On the use of Generative adversarial neural networks for computing photonic crystal fiber optical properties

Aimen Zelaci, Ahmet Yaşlı, Cem Kalyoncu, and Hüseyin Ademgil

Abstract

Our aim is to surpass the performance of the simulations conducted using Full-Vectorial Finite Element Method (FV-FEM) to design, optimise and evaluate a Surface Plasmon Reasonance (SPR) based Photonic Crystal Fiber (PCF) sensor performance, using deep learning. By these simulation techniques one must calculate the confinement loss for a specific wavelength and a particular set of geometric properties of the PCF sensor, then repeat the process for different analytes. In order to find the best fit distribution we have to make as much calculations per wavelength as possible to ensure the correct location of the peak value of the confinement loss. This process is then repeated for different geometric set that consists of more than 6 elements to find the best set that maximizes the distance between confinement loss distributions, in other words to find the best PCF sensor. Thus this way is not only time costing, consuming hours to days, but also inaccurate sometimes. In contrast, a deep neural network model does all of this astonishingly in under a second.

I. Introduction

In this paper we propose the use of artificial neural networks (ANN) to compute the loss in photonic crystal fibers in a wide range of configurations. Additionally, in order to minimize the required amount of samples to train the ANN, we employed generative adversarial networks (GAN). As discussed in the experiments section, this setup significantly reduces the error in the prediction compared to the state-of-the-art methods used in the literature.

Importance of PCF and SPR, written by AY or HA

Machine learning techniques are coming to the forefront of many fields [ref], surpassing the human performance in many tasks, namely automatic speech recognition, image recognition, natural language processing, drug discovery and toxicology. Additionally, ANN can approximate any function (Universality theorem)[ref]. This fact propelled researchers to widen the applications of ANN even further, including the study of nanophotonic structures [ref], optimization of photonic crystal nanocavities [ref], and more recently, computing optical properties of a photonic crystal fiber [ref].

One of the most difficult challenges that deep learning models face is that they benefit from large amounts of data to train, which may be costly to acquire [ref]. One of the solutions to overcome this issue is to artificially expand the original dataset by the means of generative networks. Introduced by Goodfellow et al., Generative Adversarial Networks (GAN) [ref], proved to be successful in tasks such as synthetic data generation [ref] and synthetically augment the data for deep learning based systems [ref]. In this paper we focus on determining one of the propagation features of a multi-channel Photonic Crystal Fiber (PCF) sensor based on Surface Plasmon Resonance (SPR), that is the confinement loss, by using an Artificial Neural Network model, and a special kind generative networks to expand our limited data set for the purpose of improving the accuracy of our ANN model.

Literature survey

This paper is organized as follows. Section II details the use of GAN to generate additional training samples for ANN as well as the proposed neural network architecture. Photonic crystal fiber design that is used for testing is described in details in section III. Detailed analysis of the experimental results are discussed in section IV. Finally, concluding remarks are made in section V.

II. Proposed method

We will first train ANN model to predict the confinement loss of a SPR based PCF using the original data set collected. Then we will augment the original training data set with generated samples by the GAN, to improve the performance of the ANN model. Previous works [ref, ref], demonstrated that random data augmentation actually had weakened the performance of the model. Hence, active researches are ongoing to find ways to optimally sample the generated data [ref], in other words to filter out the ”good” generated data after training the GAN. In this paper, we will use a simple if-statement to discard values that fall out of our desired ranges for both the independent variables($n_{analyte}, \lambda, \Lambda, d1, d2, d3$), and the confinement loss. In the following sub-sections we discuss the networks architectures.

A. Artificial neural network design

The ANN model is a fully-connected feed-forward MLP (Multi Layer Perceptron), consisting of an input layer, an output layer, and 5 hidden layers, 50 neurons for each hidden layer. Taking ReLU (Rectified Linear Unit) as the activation function, Adam [ref] as the optimizer and the mean squared error as the loss/cost function. To reduce overfitting we use the Early stopping method [reference], that is by saving checkpoints where the best validation mean squared error occurred as we iterate. To accelerate training and mitigate the problem of internal covariate shift Batch Normalization algorithm was used [ref].

B. Generative Adversarial network design

GANs consist of two neural networks, as shown in figure (num). The first of which called a discriminator, that gives an estimate of the probability that a given input is real or generated (fake). Whereas the second network is referred to as the generator, which outputs data samples from a random noise vector called a latent variable usually given the symbol z supplied at its input. The error between the discriminator's output and the actual labels (The real data samples vectors all labelled as 1: the probability of being real) would then be measured by the means of a chosen metric. Introduced by Arjovsky et al [reference] the Wasserstein distance metric (or the earth mover distance) proved to be very effective, instead of discriminating whether an input is real or generated, the discriminator provides a criticism of how far the generated data from the real data is, hence the discriminator network is referred to as the critic in the WGANs. Throughout this paper we will use yet the improved WGAN, the WGAN with Gradient penalty [reference]. The improved WGAN converges in a stable manner with least efforts made to tune the hyper- parameters of both networks constituting the GAN, which can be very daunting. The WGAN-GP architecture chosen in this work is as follows: both the Critic and the generator networks are a fully-connected feed-forward MLP, having 4 hidden layers, with $minibatchsize \times 2^{numberoflayers}$ neurons for each hidden layer in the critic network, whereas $minibatchsize \times 2^2$ neurons for each hidden layer in the generator network. The input vector for the Critic network consists of the six parameters discussed earlier for the ANN model, in addition the corresponding loss value, giving a total of seven inputs. The generator is supplied with random noise vector having a dimension of 7, and outputs a vector that resembles that of the input to the critic network. LeakyReLU and ReLU were chosen as the activation functions for the Critic and Generator respectively and Adam as the optimizer for both networks. Finally Batch Normalization technique was applied to the Generator.

III. Photonic crystal fiber design

PCF design details, written by AY

IV. Experiments

A. Experimental setup

A labelled data-set of only 432 samples was collected in this work, through simulations using FV-FEM. The length of the data-set made the task of building ANN model look quiet impossible, however the results were satisfying to some extent. The data-set consists of the wavelength λ, index of refraction $n_{analyte}$, air-hole to air-hole distance Λ, and the air holes radii per ring $d1$, $d2$ and $d3$, taken as our independent variables. The labels are the confinement loss of the PCF. The set consists of nine different configurations of the geometric properties (Λ, d1, d2, d3), for each configuration the confinement loss was calculated for 3 different analytes (Water (n=1.33), Ethanol (n=1.35) and several commercial hydrocarbon mixtures (n=1.36) [35]). Seven configurations were randomly selected for training both the WGAN-GP and the ANN, one configuration was held out for validation, and the last configuration for testing. This data was preprocessed before fed in the networks. The indices of refraction are very close, which made it quiet difficult for the neural networks to differentiate between them, after many trials, the best choice was to take only the tens digit of $134, 135, 136$, giving $4, 5, 6$. Finally, we transform the confinement loss to the log scale.

Metrics used in the comparisons

B. Performance of ANN

We train the ANN model for more than 2000 epochs, starting from the original data-set. Then we augment the data by 1000 generated samples from our WGAN-GP, which will be demonstrated in the next section. The following table summarizes the hyper-parameters chosen for the ANN model:

Length of the training data-set	Learning rate	Batch size	β_1	β_2
336	1×10^{-04}	12	0.9	0.999
1000+336	1×10^{-04}	12	0.9	0.999
2000 + 336	2×10^{-04}	16	0.9	0.999
3000 + 336	2.5×10^{-04}	16	0.9	0.999

Table I: Hyper-parameters chosen for the ANN model

The MSE on the training sets ranged from 0.0030 to 0.0050. For all data sets the MSE on the training sets decreased in an acceptable manner, as shown in the following figure.

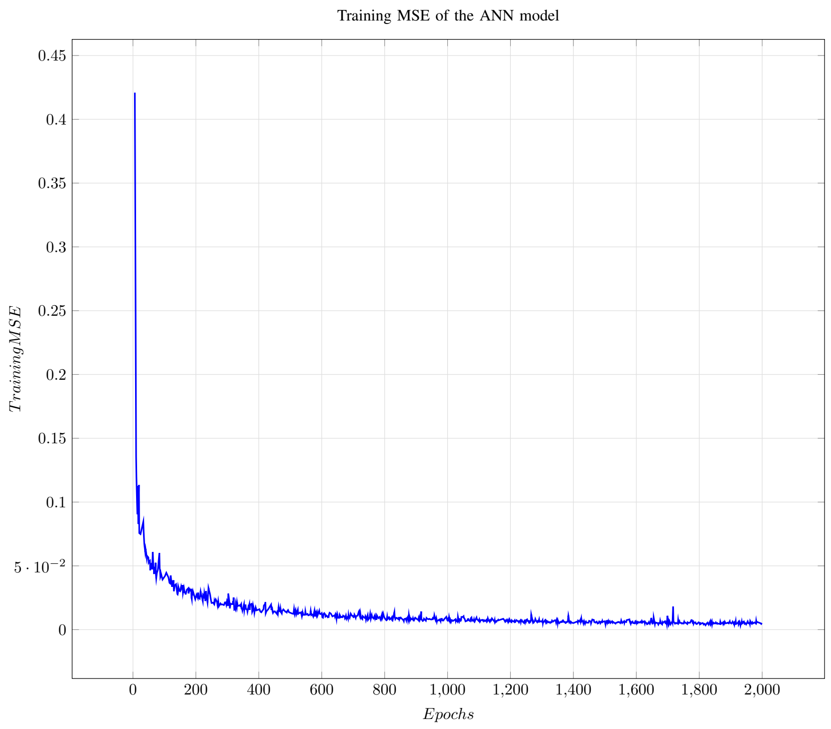

Figure 1

Next, we plot the predictions the ANN model made after training using only the original data-set.

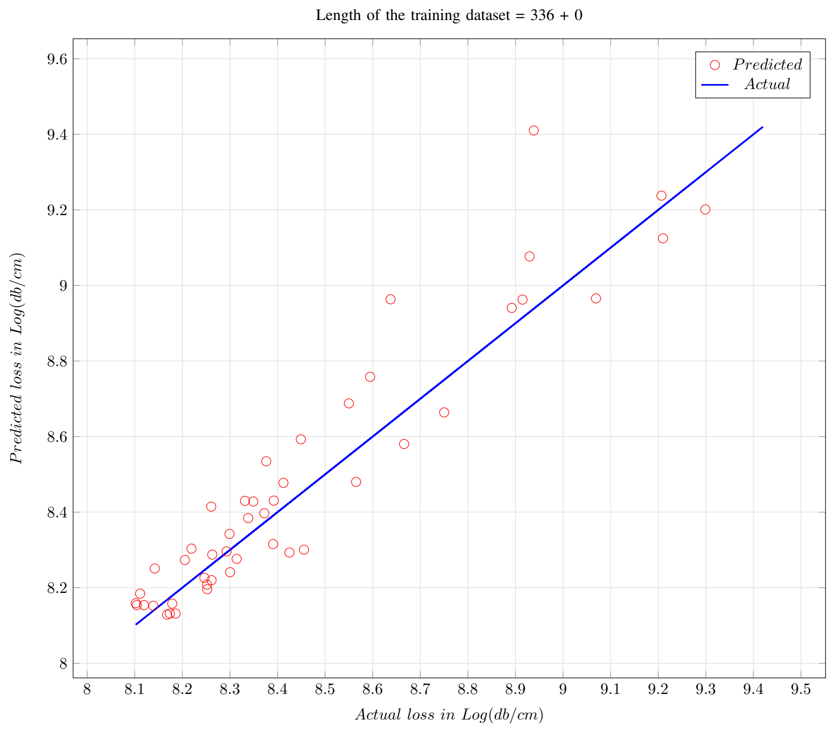

Figure 2

As you can see, the ANN made somewhat good predictions of the low values of the confinement loss. A close inspection into the data samples collected shows that points are concentrated in the low values interval, i.e the ANN picked up the pattern quiet easily. However, in the higher values interval, the ANN could not see a pattern because points are distant from each other to some extent, even though we transformed the confinement loss to the log scale to actually minimize these distances.

C. Performance boost of GAN

During training the metric used to monitor the WGAN-GP is the loss function of the Critic network. Its convergence signals the ending of the training phase, shown in figure(num). In this sub-section we discuss the results of augmenting the real data set with Synthetic data. Figure(num) demonstrates the predictions made by the ANN model after training with different sizes of the Synthetic data set. This might seem paradoxical to what we mentioned earlier, that ANN model benefit from large amounts of data. There are several reasons that is driving our ANN model predictions in the wrong way. Even though our WGAN-GP seemed well trained, there still a domain gap between the real data and the generated data, or the expansion of data is growing in the wrong direction to that of entirely containing the real data. Furthermore, the generated data might lack realism. Or the sampling strategy adopted is poor[ref, ref, ref]. After all, with 1000 generated data samples the ANN model made quiet accurate predictions with which we are satisfied, that is we have achieved we set out to accomplish at the beginning, to train an ANN model and make accurate prediction on a never-seen-before data.

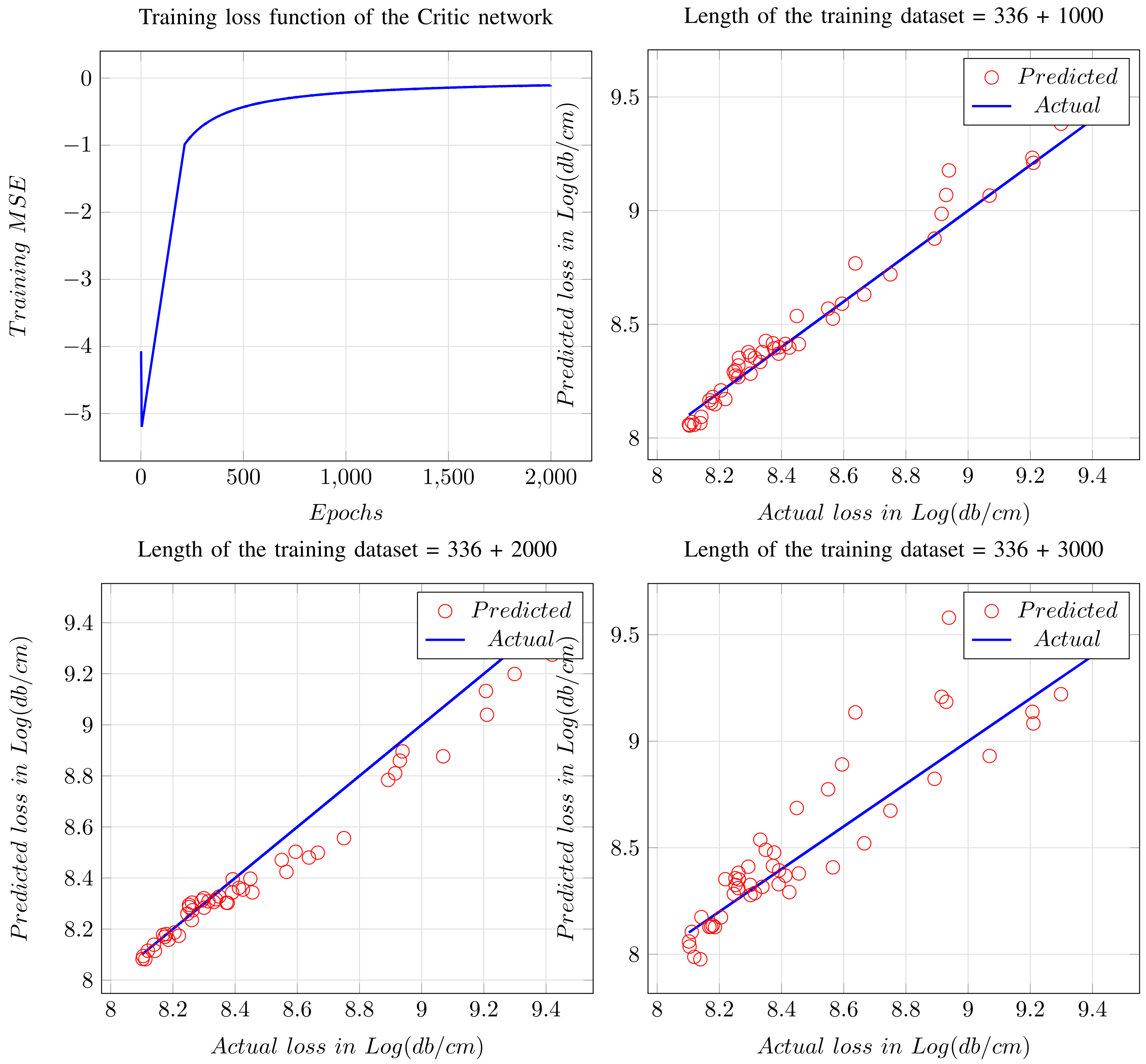

Figure 3

It can readily be seen that by augmenting with 1000 samples the ANN model made the best predictions, which then can be better viewed when we plot the confinement loss of the PCF versus the wavelength(λ) in figure(num).

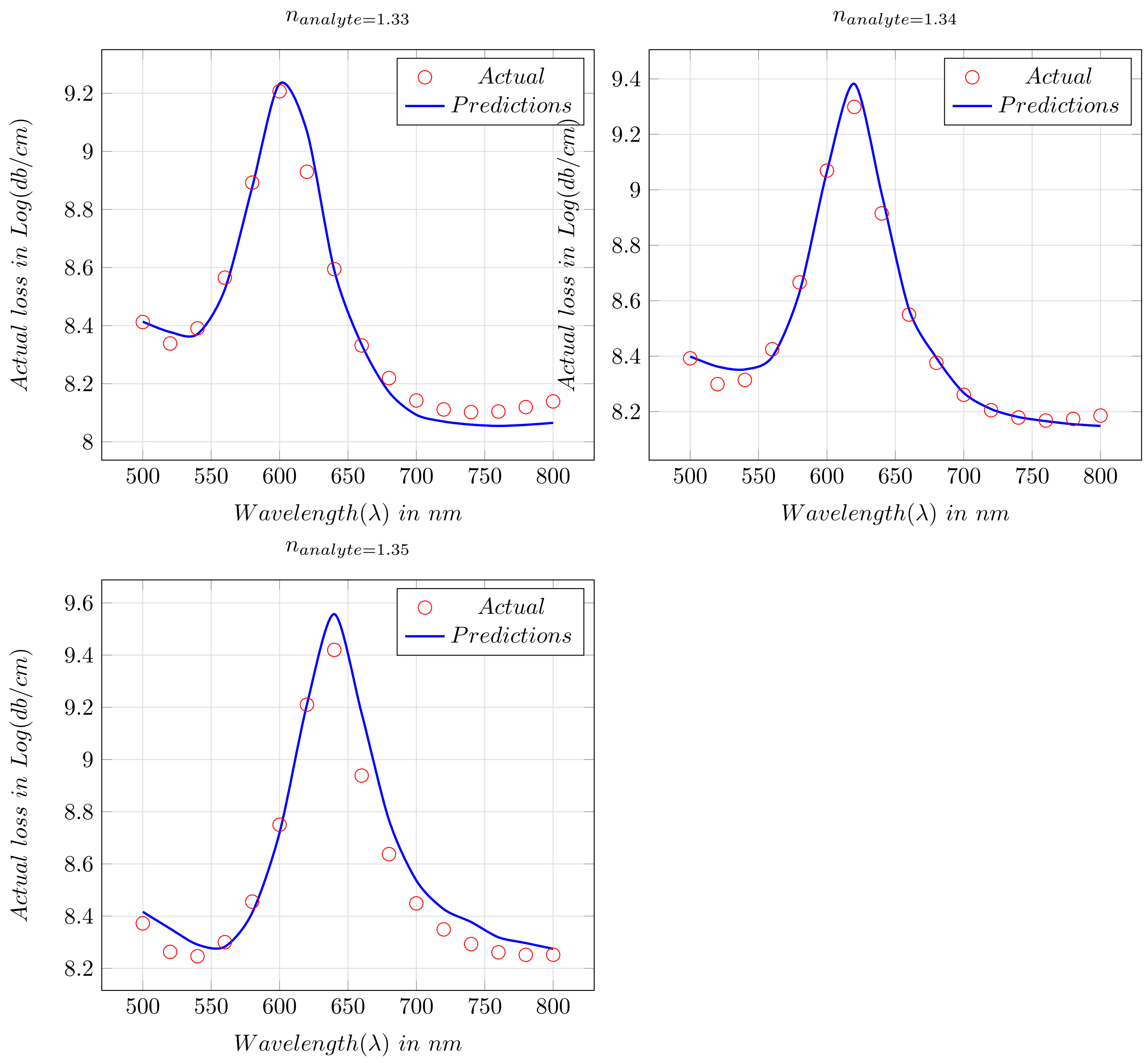

Figure 4

D. Computational performance

Training and execution time of ANN versus simulation method

This work had been conduct on a laptop having: Intel(R) Core(TM) i7-5600 CPU @2.60GHz (4 CPUs) and 8GB RAM. The code for this work is available at: https://github.com/Aimen-Zelaci/SPRPCF_ANN/.

The elapsed training time of an ANN model depends on the parameters of the model, for example, the number of hidden layers and hidden neurons, batch size, data set size, epochs and framework used to code the model. In this work we fixed the number of epochs and executed training using Tensorflow/Keras and Pytorch frameworks, by varying the mini-btach size and the data set size, as demonstrated in Table(num). It is worth mentioning the differences between using full batch or mini-batch. In full batch training the whole data set is passed to the model for each iteration. In contrast, mini-batch training, the data set is split into small mini-batches of equal sizes. Previous works have firmly proved the advantages of using the correct mini-batch size as oppose to large batch sizes[ref]. Training runtime is longer using mini-batches. However, mini-batches allow for more frequent gradient calculations, that results in more stable and reliable training [ref]. Perhaps one of the main downsides of using large batch size is the poor generalization resulting in a phenomena know as the ”generalization gap”[ref]. Active research is on going to actually use large batch sizes while mainting the performance of the model and shortning the training runtime [ref]. A closely related recent work (Machine learning approach for computing optical properties of a photonic crystal fiber, SUNNY CHUGH, AAMIR GULISTAN, SOUVIK GHOSH, AND B.M.A. RAHMAN, DEC 2019), conducted their expirements using full batch to train the model. In this works we will use the latter paper availabe code on Github for Pytorch expirements, with slight modifications. Finally, we compared the performance of the ANN/WGAN models using CPU and GPU. As Table (num) demonstrates, GPU based training outperformes that of CPU based. With the emergence of Cloud Computing technology, one can rent GPU based instances of high specs with pennies on the dollar.

	data set size	mini batch size	Time elapsed (sec)		data set size	Time elapsed (sec)
Mini batches	336 + 0	8	136	Full batch	336 + 0	96
	336 + 1000	8	388		336 + 1000	162
	336 + 2000	16	563		336 + 2000	165
	336 + 3000	20	958		336 + 3000	263

Table II: Training performance using Tensorflow/Keras

	data set size	mini batch size	Time elapsed (sec)		data set size	Time (sec)
Mini batches	336 + 0	8	359	Full batch	336 + 0	14
	336 + 1000	8	1348		336 + 1000	57
	336 + 2000	16	1379		336 + 2000	107
	336 + 3000	20	1504		336 + 3000	195

Table III: Training performance using Pytorch

WGAN EXCUTION WITH GPU VS CPU BY CEM

Next, we compare the testing runtime of the ANN model to that of simulation methods used to compute the confinement loss for 3 different analytes and 16 wavelenghts, i.e a total of 48 values of the confinement loss, demonstrated in Table(num).

	Time elapsed
ANN	0.79 sec
Simulation	7872 sec = 2.18 hours

Table IV: ANN vs Simulation test runtime performance

V. Conclusion

about the improved speed

reduced amount of training samples

increased generality of the system

Machine learning approaches has an inherit strength on top of classical simulation methods: they could model hidden parameters in a system which we have no knowledge about. Therefore, not only ANN can improve the speed of PCF simulation, but also accuracy of the simulations. However, this final claim requires further analysis and experimentation.