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 propose a deep learning approach that surpasses simulations conducted using Full-Vectorial Finite Element Method (FV-FEM) used to design, optimise and evaluate a Surface Plasmon Reasonance (SPR) based Photonic Crystal Fiber (PCF) sensor performance. Through 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. To eventually find the best set of geometric properties that maximises the distance between the confinement loss distributions of different analytes, and thus find the most sensible PCF sensor. In order to find these distributions we have to make as much calculations per wavelength as possible to ensure the correct location of the peak value of the confinement loss. Simulation methods are not only time costing and daunting, consuming hours to days, but also inaccurate sometimes. In contrast, a deep neural network model astonishingly performs these calculations in milliseconds to seconds, quiet accurately, when it is well trained with an accurate data set.

I. Introduction

Importance of PCF and SPR, written by AY or HA

Machine Learning (ML) techniques are coming to the forefront of many fields, surpassing the human performance in many tasks, namely automatic speech recognition, image recognition, natural language processing, drug discovery and toxicology. Artificial Neural Networks can approximate any function by virtue of the Universality theorem: No matter what the input is there is to be a guaranteed ANN architecture that computes the desired outputs. This fact propelled researchers to widen the applications of ANN even further, including the study of nanophotonic structures [1], optimization of photonic crystal nanocavities [2], and more recently, computing optical properties of a photonic crystal fiber [3].

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. One of the solutions to overcome this issue is to artificially expand the original training dataset by the means of generative networks. Introduced by Goodfellow et al., Generative Adversarial Networks (GAN) [4], proved to be successful in data generation [5, 6, 7, 8, 9]. 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 [10, 11], 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 [12], 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. Before we proceed, it is worth mentioning that the choices made for the parameters of the networks are by no means a robust set of rules, rather, they are rules of thumb and heuristics, as no such rules have been yet developed for ANN models.

A. Artificial neural network design

The ANN model is a fully-connected feed-forward MLP (Multi Layer Perceptron). Table(num) summarizes the details of the model consisting of an input layer. To reduce over fitting we use the Early stopping method, by saving checkpoints where the best validation mean squared error occurred as we iterate before the model starts to over fit. To accelerate training and mitigate the problem of internal covariate shift Batch Normalization algorithm [14] was used.

Number of hidden layers	5
Number of neurons per hidden layers	50
Activation function	Rectified Linear Unit (ReLU)
Optimizer	Adam[13]
Loss function	Mean Squared Error (MSE)

Table I: Detailes of the ANN model

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 at the discriminator's output would then be measured by the means of a chosen metric. Introduced by Arjovsky et al [15] 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 [16]. 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. In this work, both the Critic and the generator are fully connected feed forward MLP models. Tables(num, num) summarize the details of both networks. The input vector to the Critic network consists of the six parameters discussed earlier for the ANN model, in addition to the corresponding confinement 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.

Number of hidden layers	4
Number of neurons per hidden layers	$minibatchsize \times 2^{numberoflayers}$
Activation function	Leaky Rectified Linear Unit (LeakyReLU)
Optimizer	Adam[13]

Table II: Details of the Critic model

Number of hidden layers	4
Number of neurons per hidden layers	$minibatchsize \times 2^2$
Activation function	Rectified Linear Unit (ReLU)
Optimizer	Adam[13]
Normalization	Batch Normalization[14]

Table III: Details of the Generator model

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 latter set by 1000 generated samples by our WGAN-GP, which will be demonstrated in the next subsection. The following table summarizes the hyper-parameters chosen for the ANN model:

Length of the training data-set	Learning rate	Mini batch size	β_1	β_2
336	1×10^{-04}	8	0.9	0.999
336 + 1000	1×10^{-04}	8	0.9	0.999
336 + 2000	2×10^{-04}	16	0.9	0.999
336 + 3000	2.5×10^{-04}	20	0.9	0.999

Table IV: 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. Next, we plot the predictions the ANN model made after training using only the original. As shown in the following figure data-set.

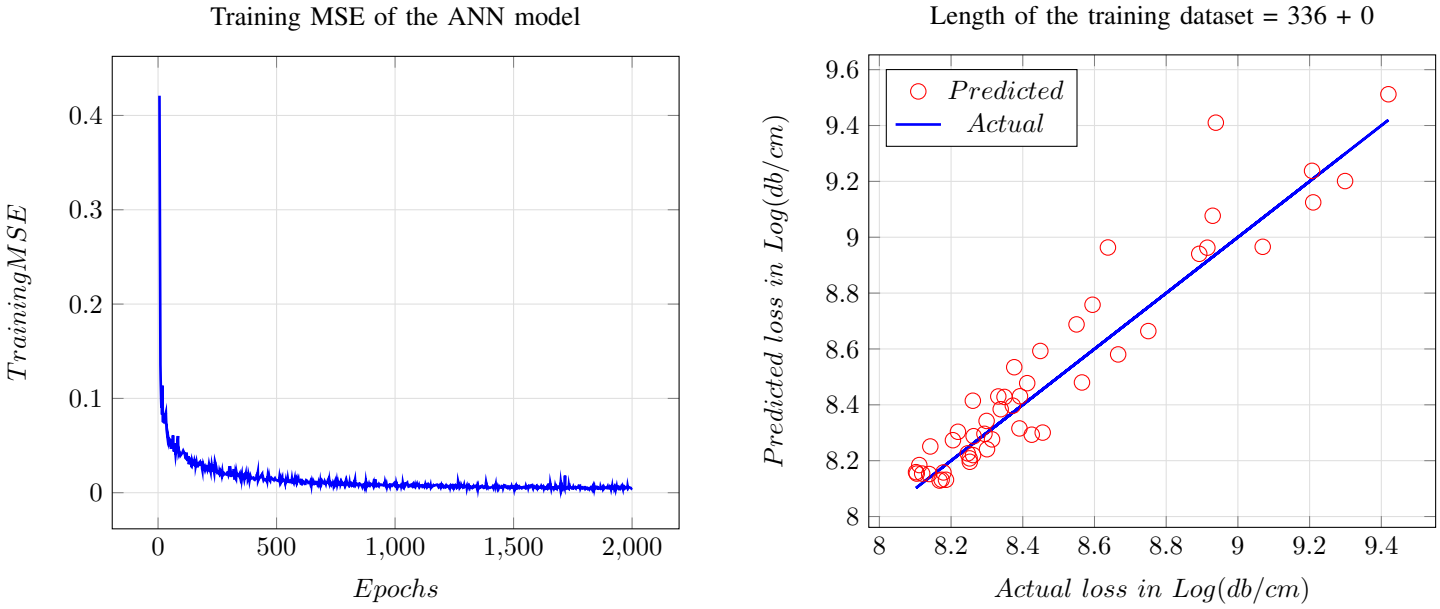

Figure 1

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 generated samples. Figure(num) demonstrates the predictions made by the ANN model after training with different sizes of the data set. This might look 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. But the most convincing reason is the limited original data set size of 336 samples, used to train the WGAN model in the first place. After all, with 1000 generated data samples the ANN model made quiet accurate predictions with which we are satisfied, that is we have achieved what we set out to accomplish at the beginning, to increase the accuracy of the ANN model by artificially expanding the original set.

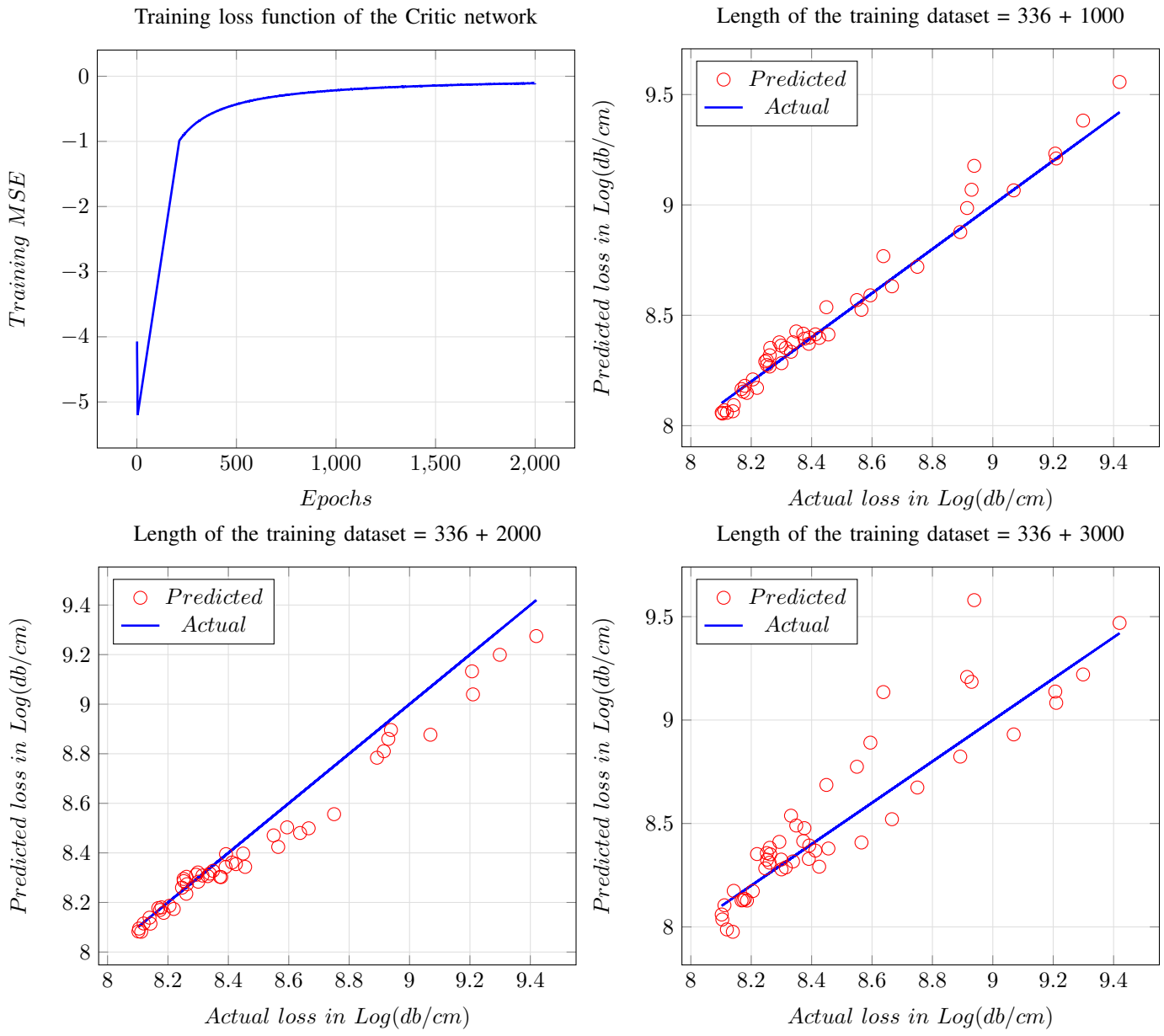

Figure 2

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).

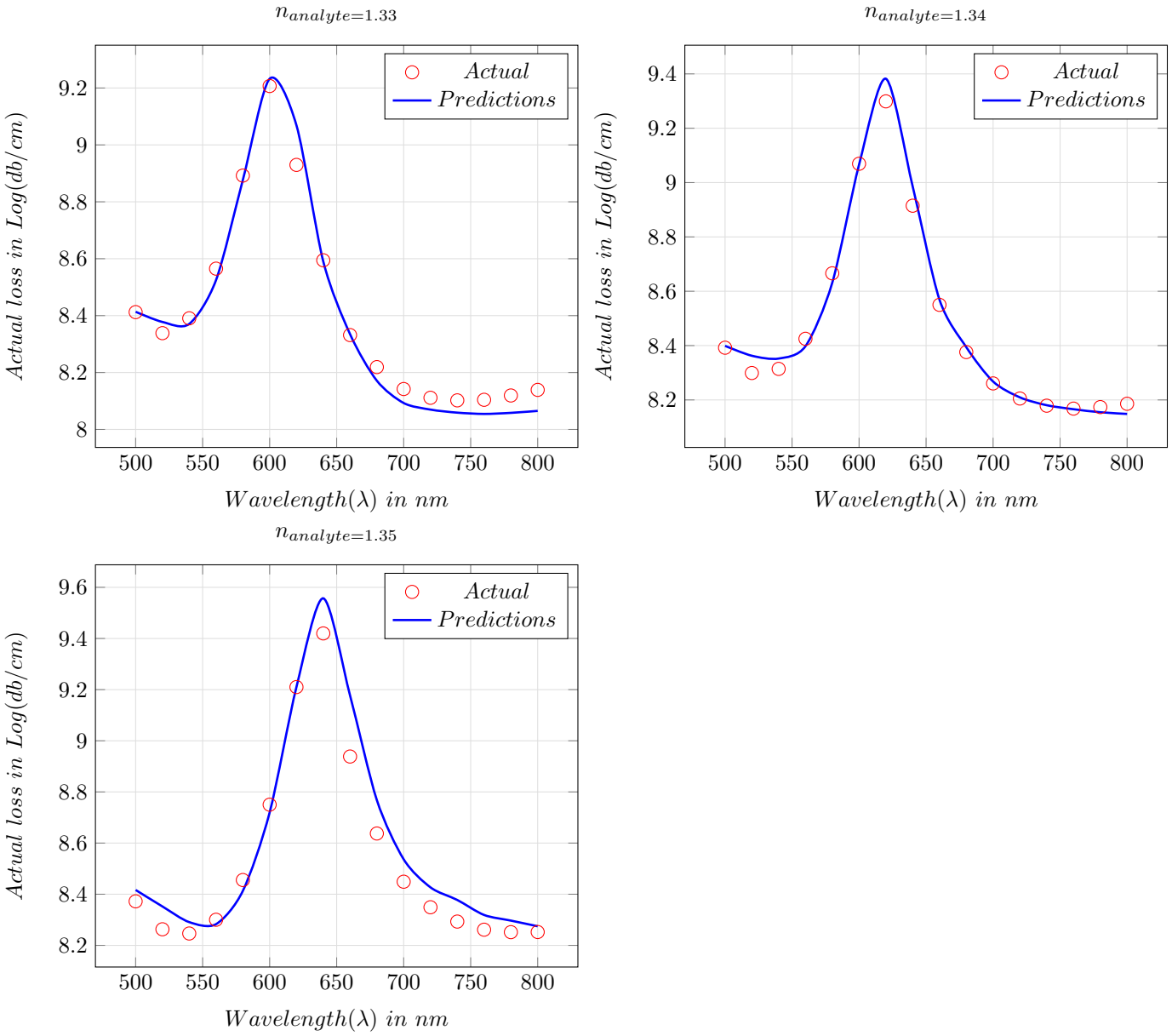

Figure 3

D. Computational performance

Training and execution time of ANN versus simulation method

This work had been conduct on a laptop:

- Intel(R) Core(TM) i7-5600 CPU @2.60GHz (4 CPUs)
- 8GB RAM
- No GPU on board

For comparaison purposes we ran the expirements on a machine with GPU on board:

CEM PC SPECS

Epirements conducted on the latter machine will be labeled as GPU based, whereas on the former as CPU based.

By taking advantage of Cloud Computing technology, one does not have to spend large amounts of money to obtain high-specs machine, rather, rent readily available deep learning instances in the cloud for pennies on the dollar.

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, mini batch size, data set size, number of epochs, the framework used to code the model, and of course the specifications of the machine. In this work we fixed the number of epochs to 2000 and executed training using Tensorflow(Keras) and Pytorch frameworks. As demonstrated in Table(num). It is worth mentioning the differences between training with full batch or mini batches. 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, then passed to the model seperately, thus for each epoch there will be exactly length of the data set devided by the mini batch size steps, and hence slowing down the training process. Previous works [17, 18] have firmly proved the advantages of using the correct mini batch size as oppose to large batch sizes. Mini batches allow for more frequent gradient calculations, that results in more stable and reliable training. Perhaps one of the main downsides of using large batch size is the poor generalization resulting in a phenomena know as the ”generalization gap”. Active research [19] is on going to actually use large batch sizes while mainting the performance of the model and shortning the training runtime. 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 work we will use the latter paper availabe code on Github for Pytorch expirements, with slight modifications to fit our purposes. Finally, we compare the performance of the WGAN model, using Tensorflow(Keras) framework, of CPU based to that of GPU based.

	data set size	mini batch size	Time elapsed (sec)		data set size	Time elapsed (sec)
Mini batches	336 + 0	8	214	Full batch	336 + 0	23
	336 + 1000	8	744		336 + 1000	34
	336 + 2000	16	685		336 + 2000	45
	336 + 3000	20	861		336 + 3000	57

Table V: Training performance using Tensorflow(Keras). CPU based training.

	data set size	mini batch size	Time elapsed (sec)		data set size	Time elapsed (sec)
Mini batches	336 + 0	8	1315	Full batch	336 + 0	36
	336 + 1000	8	6453		336 + 1000	98
	336 + 2000	16	4751		336 + 2000	195
	336 + 3000	20	5132		336 + 3000	336

Table VI: Training performance using Pytorch. CPU based training.

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). This result is absolutely staggering. The ANN model calculated 16 confinement losses in just 0.17 seconds, while simulations took about 2 hours. Therefore, a well trained ANN model can be used to accurately find the best PCF geometric properties set that maximisez the distance between the confinement loss distributions of each analyte in a matter of seconds to minutes. Hence, accelerate the search for the best PCF sensor.

	Time elapsed
ANN	0.17 sec
Simulation	7872 sec = 2.18 hours

Table VII: 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.