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*—Photonic crystal fibers (PCF) for specific applications are designed and optimized by both industry experts and researches. However, the potential number of combinations possible for a single application is very large. This issue combined by the speed of the commonly used Full Vectorial Finite Element Method (FV-FEM) causes the task to take significant amount of time. As stated in the previous works, artificial neural networks (ANN) can predict the result of numerical simulations much faster. However, there are two issues with the methods proposed previously. Namely, the required number of samples for training and the overall accuracy of these methods. In this paper, we propose the use of generative adversarial networks (GAN) to augment the real dataset to train an ANN model. Experimental analysis suggest that the proposed combination not only accurately predicts the confinement loss even with limited amount of data but also GAN can be used to improve existing methods in the literature. Finally, it is shown that this system can predict the confinement loss over a range of analytes and wavelengths in a completely new set of geometric configuration.**

I. Introduction

Importance of PCF and SPR, written by AY or HA

Machine Learning (ML) techniques are becoming a common tool in many fields, surpassing human performance in many tasks, namely automatic speech recognition, image recognition, natural language processing, drug discovery and toxicology. Additionally, ANN can approximate any function proven by Universality theorem [1]. This fact propelled researchers to widen the applications of ANN even further, including the study of nanophotonic structures [2], optimization of photonic crystal nanocavities [3], and more recently, computing optical properties of a photonic crystal fiber [4].

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) [5], proved to be successful in data generation [6]–[10].

In this paper, we focus on estimating confinement loss, one of the propagation features of multi-channel Photonic Crystal Fiber (PCF) sensors, using artificial neural networks. Specifically, we have based our system on Surface Plasmon Resonance (SPR). However, in the experiments section we have demonstrated that the designed system is generic enough to apply to multiple PCF designs. The most important contribution of this research is the use of GAN phase, where the available data is expanded to be used in the training phase.

!! Literature survey !!

Use of ANN in PCF

SPR in PCF

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

In this section details of the proposed method is discussed. Training of the system contains two phases: a GAN phase and regression training phase. At the start of the training, a GAN is trained to generate additional data by using training samples. These generated samples are filtered to ensure they fall within the applicable range. Original training samples and remaining samples are joined to train a fully-connected feed-forward multi layer perceptron neural network. At the end of the training, the ANN that is trained for the regression task will be used to decide the containment loss of a set of input parameters. This architecture is illustrated in Figure 1. The details of the proposed GAN and ANN architecture is discussed in the following subsections.

A. Generative adversarial network design

Generative adversarial networks are introduced in [5]. We have employed GAN to augment the number of samples that are used in training. A GAN is composed of two neural networks: generator and discriminator. The aim of the generator is to transform fully randomized data into data that follows the distribution of the original dataset. Discriminator assesses the performance of the generator and provides feedback for training. Instead

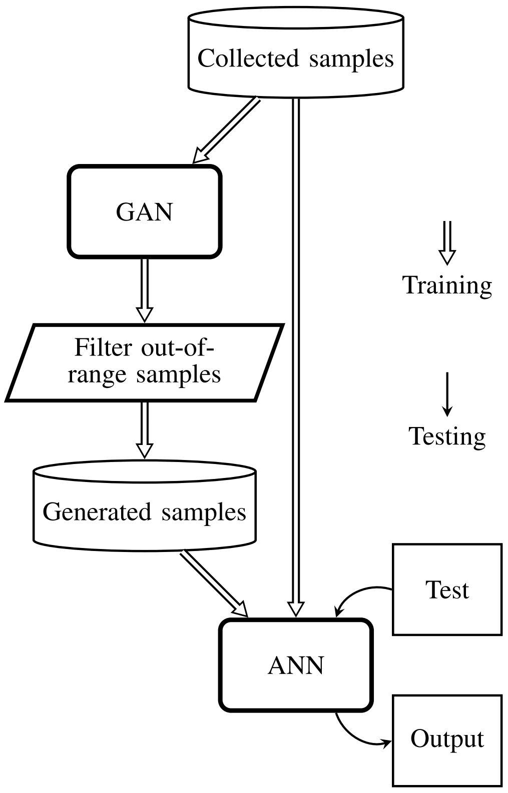


Figure 1: System architecture

of directly training generator, it is trained through this feedback. This paradigm avoids over-fitting the data.

It is possible to use different metrics for training in GAN [references here], for this project we have selected WGAN Wasserstein distance metric. This variant has been proposed by Arjovsky et al in [11]. In this system, discriminator is named as critic and it measures the distance between the generated data and the real data. The reason behind choosing WGAN over other methods is to be able to determine a stopping criteria. In a regular GAN system, training is stopped when the generated data is deemed viable by an observer. Since GAN is often used in generation of image, video or audio, using a human in the loop is effective. However, in our problem, it is not viable for a human to judge the generated data. Automating this procedure to remove the human in-the-loop can lead to over or under-fitting, which in turn degrades the performance. WGAN uses an adaptive stopping criteria that does not have the issue mentioned above. Additionally, we have selected to incorporate Gradient penalty to improve WGAN proposed in [12]. This improved WGAN system converges in a stable manner without having to fine tune hyper-parameters of the system. The flowchart of this system is given in Figure 2.

In this work, both the Critic and the generator are fully connected feed forward MLP models. The details of the networks are given in Table I. The output from this system should be input and output pairs that will be used to train ANN part of the system. In our PCF system we have 6 input parameters and a single output parameter making a total of 7 parameters that will be used in the GAN phase. In Section IV, we have experimented using a different PCF system with different inputs, resulting a different number of parameters for the system. The generator is supplied with the same number of parameters as its expected output, that is 7 for our PCF system. These parameters are generated using Gaussian noise and is called latent variable. Once the training phase is complete, the generator and the filter is used to augment the number of training samples that are available for ANN.

Previous works [13], [14], demonstrated that it is possible for random data augmentation to weaken the performance of the model. Hence, it is important to sample the generated data in way to prevent performance degradation [15]. In the proposed system, we have included a filtering step for the generated data. This step uses a simple condition to discard the values that fall out of the desired range, for both the independent variables ($n_{analyte}, \lambda, \Lambda, d1, d2, d3$), and the confinement loss.

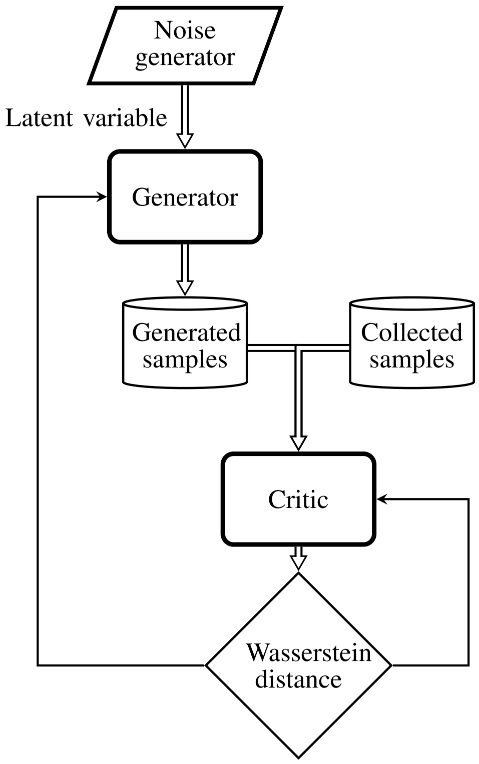


Figure 2: WGAN training

B. Artificial neural network design

The ANN model employed in this research is a fully-connected feed-forward Multi Layer Perceptron (MLP) consists of an input layer, an output layer, and 5 hidden layers. The ANN is designed and trained to estimate containment loss in a regression configuration. We have applied log_{10} to confinement loss in order to keep its numeric stability across a wide range. Each hidden layer contains 50 neurons and uses Rectified Linear Unit (ReLU) activation function. Table I summarizes the details of this model.

We have adopted Adam [16] as the optimizer and the mean squared error (MSE) as the loss/cost function. In addition, we use Batch Normalization algorithm [17] to accelerate training, reduce the effects of initial randomized state and mitigate the problem of internal covariate shift. In proposed system, the ANN is trained using the

Table I: Details of ANN models

Parameter	**Generator**	**Critic**	**Regressor**
Hidden layers	5	5	6
Neurons in hidden layers	2^2 $batchsize$	2^{layers} $batchsize$	50
Activation function	ReLU	Leaky ReLU	ReLU
Optimizer	Adam [16]	Adam	Adam
Input normalization	None	Z-score (for collected samples)	Z-score
Layer normalization	-	Batch [17]	Batch
Loss function	-Critic(generated)	Critic(generated) - Critic(collected)	MSE

samples from the original dataset and the augmented samples generated by the GAN phase.

III. Photonic crystal fiber design

A labelled dataset of only 432 samples was collected in this work, through simulations using FV-FEM. This dataset 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 three different analytes (Water (n=1.33), Ethanol (n=1.35) and several commercial hydrocarbon mixtures (n=1.36)).

We will fix the pattern and geometrical shape of the cladding air holes, and vary 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 $d3$. Then the results of this study can be carried to different types of SPR-based PCF sensors, using other suitable deep learning approaches such as Recurrent and Convolutional neural networks.

PCF design details, written by AY

IV. Experiments

A. Experimental setup

In order to prove the effectiveness of the proposed setup, we have performed multiple experiments over two datasets. We have built the first dataset that is used in the experiments. This is an SPR PCF configuration ... and collected using FV-FEM. Details of this dataset is explained in Section III and it is referred as SPR dataset. Having nine configurations, we have performed 9-fold testing on this dataset. Each fold tests a single configuration using the other configurations as training data. Slicing the data this way allowed us to guide the network to predict the loss on a completely different configuration. Please note that we have applied log_{10} to confinement loss to scale it.

Additionally, we have used the dataset that has been used in [4]. This dataset contains over 1000 samples. Inputs to this data is .... and outputs are .... We have only used log of confinement loss as the output. This dataset is referred as PCF. Details of the datasets that are used in the experiments are given in Table II.

We have used mean square error as in the comparisons. All algorithms are implemented using TensorFlow library in Python and experiments are performed on laptop with Intel i7-5600 CPU and 8GB of RAM. TensorFlow library is set to use only CPU for the experiments.

Table II: Datasets that are used in experiments

	SPR	PCF
Samples	432	1117
Configurations	9	-
Parameters	6	5
Output	log_{loss}	log_{loss}
Testing	48 (1 configuration)	10%
Testing method	9-fold	10-fold

In the following subsections, we have performed experiments to show the performance of the base ANN system by its own comparing with the state-of-the-art method in the literature, improvement gained by employing GAN phase, and finally the computational cost of the overall system compared to the simulator. All numeric comparison results are averaged over multiple executions.

B. *Performance of ANN*

In this set of experiments, we have analyzed the performance of the proposed ANN design. The first experiment involves in training loss over SPR dataset. In this experiment, we have trained ANN model for up to 2000 epochs and reported the training loss in Figure 3. According to this experiment, training our ANN architecture more than 1000 epochs does not yield to any significant improvement. Therefore, we have set the training epoch limit to 1000 in subsequent experiments. Additionally, it is clear that the use of augmented samples from the GAN phase reduces the oscillations in the training error.

We have performed experiements comparing the proposed ANN architecture with the state-of-the-art method proposed in [4]. Results of this experiment is demonstrated in Figure 4. Please note that in the experiment involving SPR dataset, ANN models are predicting a new configuration and the number of samples available for training is lower. Therefore, they both have higher error rate compared to the PCF dataset.

CK: continue here..

C. *Performance boost of GAN*

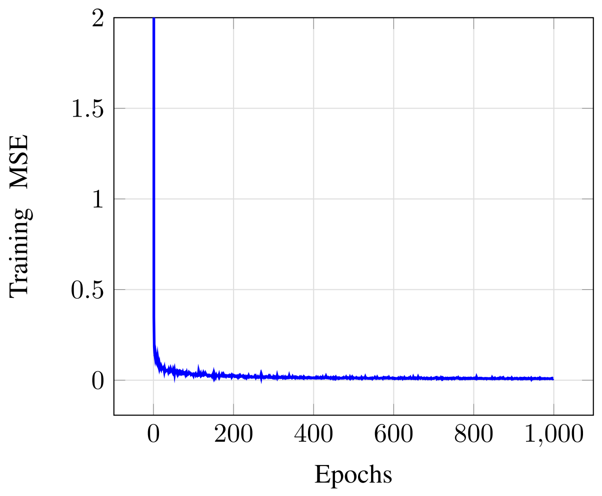
During training, the metric used to monitor the WGAN-GP is the loss function of the Critic network. Its convergence signals the ending of the training phase, shown in Figure(4). In this subsection we discuss the results of augmenting the real dataset with generated samples. Figure(4) demonstrates the predictions made by the ANN model after training with different sizes of the dataset. This might look paradoxical to what we mentioned earlier, that ANN model benefit from large amounts of data. There are several reasons for why our ANN model is driven in the wrong direction. 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 training dataset 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 improve the accuracy of the ANN model by artificially expanding the original set.

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(5). The most important result in these experiments is that the ANN model predicted the correct

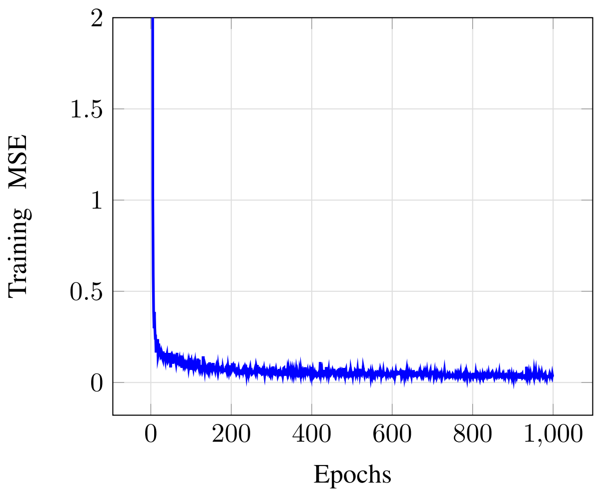
Needs fixing

Table III: Hyper-parameters chosen for the ANN model, where β_1 and β_2 are a part of Adam parameters

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



(a) With augmentation



(b) Without augmentation

Figure 3: Training MSE of the ANN model with (a) and without (b) GAN phase obtained from SPR dataset

location of the peak value of the confinement loss distributions for the three different analytes used.

D. *Computational performance*

Training and execution time of ANN versus simulation method

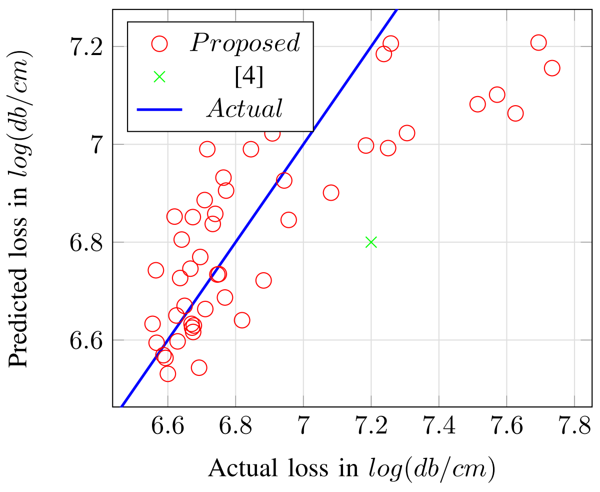
Epirements conducted on the latter machine will be labeled as GPU based, whereas on the former as CPU based.

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.

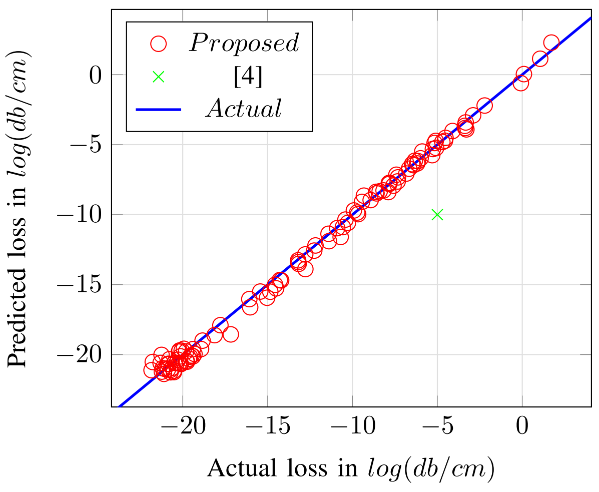
The code for this work is available at: https://github.com/Aimen-Zelaci/SPRPCF_ANN/.

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, dataset size, number of epochs, the framework used to code the model, and of course the specifications of the machine.

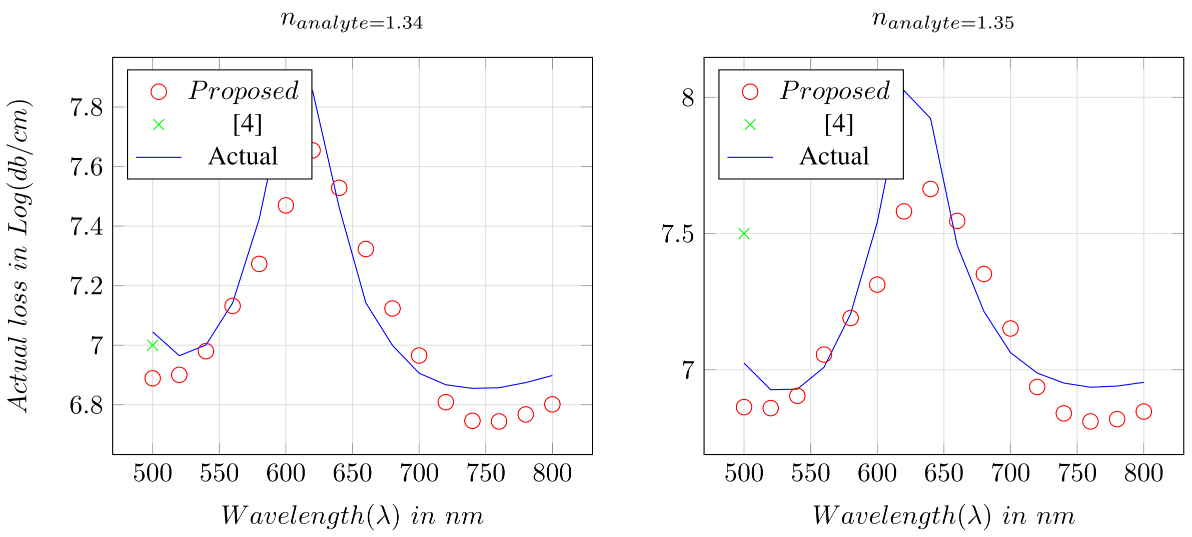
In this work we fixed the number of epochs to 2000 and executed training using Tensorflow(Keras) and Pytorch frameworks. As demonstrated in Tables(5,6). It



(a) SPR dataset



(b) PCF dataset



(c) Configuration estimation on SPR dataset

Figure 4: Testing results without GAN augmentation phase

is worth mentioning the differences between training with full batch or mini batches. In full batch training, the whole dataset is passed to the model for each iteration. In contrast, mini-batch-training, the dataset 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 dataset devided by the mini batch size steps, and hence slowing down the training process. Previous works [18], [19] 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 [20] 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 [4], 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.

	dataset size	mini batch size	Time elapsed (sec)
Mini batches	336 + 0	8	214
	336 + 1000	8	744
	336 + 2000	16	685
	336 + 3000	20	861

	dataset size	Time elapsed (sec)
Full batch	336 + 0	23
	336 + 1000	34
	336 + 2000	45
	336 + 3000	57

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

	dataset size	mini batch size	Time elapsed (sec)
Mini batches	336 + 0	8	1315
	336 + 1000	8	6453
	336 + 2000	16	4751
	336 + 3000	20	5132

	dataset size	Time elapsed (sec)
Full batch	336 + 0	36
	336 + 1000	98
	336 + 2000	195
	336 + 3000	336

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

	Time elapsed
ANN	0.17 sec
Simulation	7872 sec = 2.18 hours

Table VI: ANN vs Simulation test runtime performance

V. Conclusion

about the improved speed

reduced amount of training samples

increased generality of the system

This Idea is very important In this work we built a feed forward ANN model that learned the confinement loss given the geometric properties of a SPR-based PCF sensor with fixed pattern and geometrical shape of the cladding air holes. For different types of sensors, Convolutional and Reccurent neural networks can be of great advantage, to discover the spatial insights of variety of patterns and geometrical shape of the cladding air holes. The latter will be conducted in future work.

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.

References

[1] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251 – 257, 1991. [Online]. Available: http://www.sciencedirect.com/science/article/pii/089360809190009T

[2] Y. Kiarashinejad, M. Zandehshahvar, S. Abdollahramezani, O. Hemmatyar, R. Pourabolghasem, and A. Adibi, "Knowledge discovery in nanophotonics using geometric deep learning," *Advanced Intelligent Systems*, vol. 2, no. 2, p. 1900132, 2020.

[3] T. Asano and S. Noda, "Optimization of photonic crystal nanocavities based on deep learning," *Optics express*, vol. 26, no. 25, pp. 32 704–32 717, 2018.

[4] S. Chugh, A. Gulistan, S. Ghosh, and B. Rahman, "Machine learning approach for computing optical properties of a photonic crystal fiber," *Optics Express*, vol. 27, no. 25, pp. 36 414–36 425, 2019.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[6] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *International conference on information processing in medical imaging*. Springer, 2017, pp. 146–157.

[7] Z. Zheng, L. Zheng, and Y. Yang, "Unlabeled samples generated by gan improve the person re-identification baseline in vitro," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3754–3762.

[8] M. Frid-Adar, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "Synthetic data augmentation using gan for improved liver lesion classification," in *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*. IEEE, 2018, pp. 289–293.

[9] F. H. K. d. S. Tanaka and C. Aranha, "Data augmentation using gans," *arXiv preprint arXiv:1904.09135*, 2019.

[10] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv preprint arXiv:1712.04621*, 2017.

[11] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[12] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in neural information processing systems*, 2017, pp. 5767–5777.

[13] S. Ravuri and O. Vinyals, "Seeing is not necessarily believing: Limitations of biggans for data augmentation," 2019.

[14] K. Shmelkov, C. Schmid, and K. Alahari, "How good is my gan?" in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 213–229.

[15] B. Bhattarai, S. Baek, R. Bodur, and T.-K. Kim, "Sampling strategies for gan synthetic data," *arXiv preprint arXiv:1909.04689*, 2019.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[18] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," *arXiv preprint arXiv:1804.07612*, 2018.

[19] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv preprint arXiv:1609.04836*, 2016.

[20] E. Hoffer, I. Hubara, and D. Soudry, "Train longer, generalize better: closing the generalization gap in large batch training of neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 1731–1741.