Received 28 March 2022; revised 19 April 2022; accepted 20 April 2022. Date of publication 27 April 2022; date of current version 5 May 2022.

Digital Object Identifier 10.1109/OJAP.2022.3170798

Antenna Design Using a GAN-Based Synthetic Data Generation Approach

OAMEED NOAKOASTEEN (Graduate Student Member, IEEE), JAYAKRISHNAN VIJAYAMOHANAN, ARJUN GUPTA (Member, IEEE), AND CHRISTOS CHRISTODOULOU (Life Fellow, IEEE)

Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM 87131, USA

CORRESPONDING AUTHOR: O. NOAKOASTEEN (e-mail: oameednkn@unm.edu)

ABSTRACT In this paper, we propose the use of GANs as learned, data-driven knowledge database that can be queried for rapid synthesis of suitable antenna designs given a desired response. As an example, we consider the problem of designing the Log-Periodic Folded Dipole Array (LPFDA) antenna for two non-overlapping ranges of Q-factor values. By representing the antenna with the vector of its structural parameters and considering each desirable range of the Q-factor as a class, we transform our problem to that of generating new samples from a given class. We develop two alternative models, a Conditional Wasserstein GAN and a label-switched library of vanilla Wasserstein GANs and train them with a dataset of features and their associated labels (parameter vectors and Q-factor range). The main component of these models is a generator network that learns to map a normally distributed noise vector along with a binary label to the vector of parameters of candidate structures. We demonstrate that in inference mode, these models can be relied upon for fast generation of suitable designs.

INDEX TERMS Artificial neural networks, electromagnetics, machine learning.

I. INTRODUCTION

RAPID increase of computing power has transformed computational physics. In recent years, taking advantage of the vast amount of data that can be generated using muti-physics simulations, Deep Neural Network (DNN) based data-driven approaches have produced fast surrogate models for conventional numerical methods in both computational fluid dynamics [1]–[6] and computational electromagnetics [7]–[11]. DNN based approaches are also being applied to inverse problems, i.e., designing a suitable structure given a desired (mechanical or electromagnetic) response [12], with notable advances in the design of airfoils [13] and RF/nanophotonic metasurfaces [14]–[18].

In this work, we describe the inverse design of the LPFDA antenna for two non-overlapping Q-factor ranges as querying a learned, data-driven, knowledge database for suitable designs given the desired range. We utilize Generative Adversarial Networks (GANs) to generate new samples that are similar to a dataset of LPFDA antennas that was put together from some favorable designs. By training a GAN on this dataset, it learns an estimate of its data-generating distribution and therefore, the GAN effectively captures the knowledge that is gained from many design iterations and optimization rounds. The Q-factor becomes an important consideration in the miniaturization of antennas and given the extensive amount of time that is required to search the space of all possible designs for the near-optimal candidates, it is reasonable to investigate the feasibility of using GANs to accelerate this process. From a machine learning perspective, we view each desirable range of Q-factor as a *class* and cast the problem of antenna design in terms of generating samples (in the form of a vector of structural parameters) from each class. We choose GANs, because they can produce new samples from the same distribution as the dataset that they were trained with, and draw on a large number of suitable designs as our training dataset and experiment with two alternative architectures. The first, depicted in Fig. 1(b), is a Conditional Generative Adversarial Network (CGAN). In this model, the conditional generator accepts a noise vector and a binary class label at its input and maps them to a vector of structural parameters from the class that is

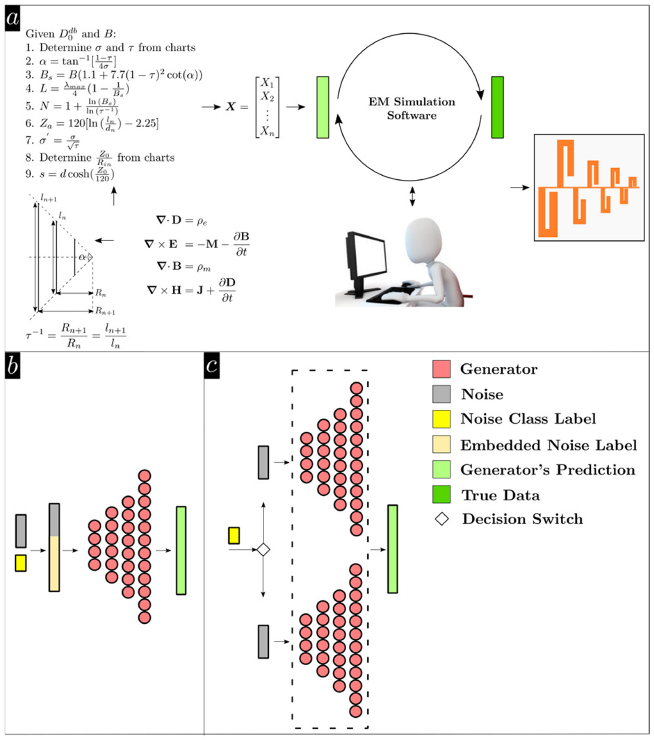

a
Given D_0^{db} and B:
1. Determine σ and τ from charts
2. $\alpha = \tan^{-1}[\frac{1-\tau}{4\sigma}]$
3. $B_s = B(1.1 + 7.7(1-\tau)^2 \cot(\alpha))$
4. $L = \frac{\lambda_{max}}{4}(1 - \frac{1}{B_s})$
5. $N = 1 + \frac{\ln(B_s)}{\ln(\tau^{-1})}$
6. $Z_a = 120[\ln(\frac{l_n}{d_n}) - 2.25]$
7. $\sigma' = \frac{\sigma}{\sqrt{\tau}}$
8. Determine $\frac{Z_0}{R_{in}}$ from charts
9. $s = d\cosh(\frac{Z_0}{120})$
$X = [X_1, X_2, \ldots, X_n]$
EM Simulation Software
l_{n+1}
l_n
α
R_n
R_{n+1}
$\tau^{-1} = \frac{R_{n+1}}{R_n} = \frac{l_{n+1}}{l_n}$
$\nabla \cdot \mathbf{D} = \rho_e$
$\nabla \times \mathbf{E} = -\mathbf{M} - \frac{\partial \mathbf{B}}{\partial t}$
$\nabla \cdot \mathbf{B} = \rho_m$
$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t}$
b
c
Generator
Noise
Noise Class Label
Embedded Noise Label
Generator's Prediction
True Data
Decision Switch

FIGURE 1. (a) conventional design procedure (b) the Conditional GAN in inference (testing) mode (c) the library of label-switched vanilla GANs in inference (testing) mode.

indicated by the noise class label. The second, depicted in Fig. 1(c), is a label-switched library of vanilla GANs. In this model, a separate GAN is trained for each class. In inference mode, the appropriate one is chosen, based on the noise class label, to make predictions. These architectures, when trained, exhibit good approximation power in generating candidate structures and can be used as fast design generators.

In summary, our contribution has been a novel data-driven approach to inverse design of antennas in which a CGAN or a library of vanilla GANs serve as a learned database. When a desired response (here a desired Q-factor range) is required, the learned database is asked to produce some new antenna designs that exhibit that response.

II. RESEARCH METHODOLOGY

A. MOTIVATION

The conventional design procedure, depicted in Fig. 1(a), comprises two phases. In the first phase, an initial value for the vector of structural parameters is obtained using approximate analytical expressions that are derived from application of Maxwell's equations to the structure under investigation. In the second phase, a multitude of full-wave simulations are carried out, guided by the insights of an experienced designer, that aim to achieve the desired EM response by methodically updating the parameter vector. In deep learning, GANs are models that learn the data-generating distribution of a dataset and are, therefore, able to generate new samples from it [19]. The CGAN is an extension of the original (vanilla) GAN that can direct the data generation process by conditioning the model on additional information, specifically, the class labels [20]. In other words, CGAN can be instructed to only generate samples from a specific class of the dataset. These observations inspire the current work: given a dataset of near-optimal structures, the learned data-generating distribution captures the knowledge obtained through the design and optimization process. Therefore, the GAN can be queried as a knowledge database to produce new and similar structural designs. This approach, can be a suitable alternative to the complicated and time consuming design procedure described above. To assess the performance of the proposed methodology, we consider the problem of designing the LPFDA antenna for specific ranges of Q-factor values. The designs that are of interest to us are based on bounds that were established, through many rounds of optimization, on certain geometric features of LPFDA that would result in desirable Q-factor values in certain UAV applications [21], [22]. Using an EM simulation software, various designs for two separate Q-factor ranges are generated and are used to train the two alternative architectures described above.

B. PROPOSED ARCHITECTURES

The proposed models, at training phase, are shown in Fig. 2. For GANs, training is an adversarial process in which a discriminator network estimates whether a given example is produced by the generator (fake) or is from the dataset (real). As training progresses, the generator learns increasingly better approximations to the distribution of data and it will become harder for the discriminator to differentiate between the output of the generator and the actual dataset. The generator of the CGAN accepts a noise vector (which is sampled from the normal distribution) and a binary class label as inputs. The label is provided as an integer and is then embedded (i.e., projected onto a higher-dimensional space) and concatenated with the noise vector. This mixture of noise and class information is mapped to the vector of structural parameters after passing through the generator. The generators in the library of vanilla GANs, accept only a noise vector (which is sampled from the normal distribution) and map it to the vector of structural parameters. In the CGAN's case, the desired class of the generated samples, i.e., Q-factor range, is conveyed to the generator via the noise class label. In the vanilla GAN's case, during inference, the noise class label is used to choose the generator that was trained for the desired class from the library.

We implement all generators and discriminators with fully-connected layers because the data structure of our training examples is a vector of discrete values (fold gap, feed separation, etc). Layer weights are initialized using Variance Scaling method [23] and its activations are Batch-Normalized [24] and passed through ReLU non-linearity. The widths of generator's stack of layers are 128/256/512/10 (last layer's width can vary depending on the shape of

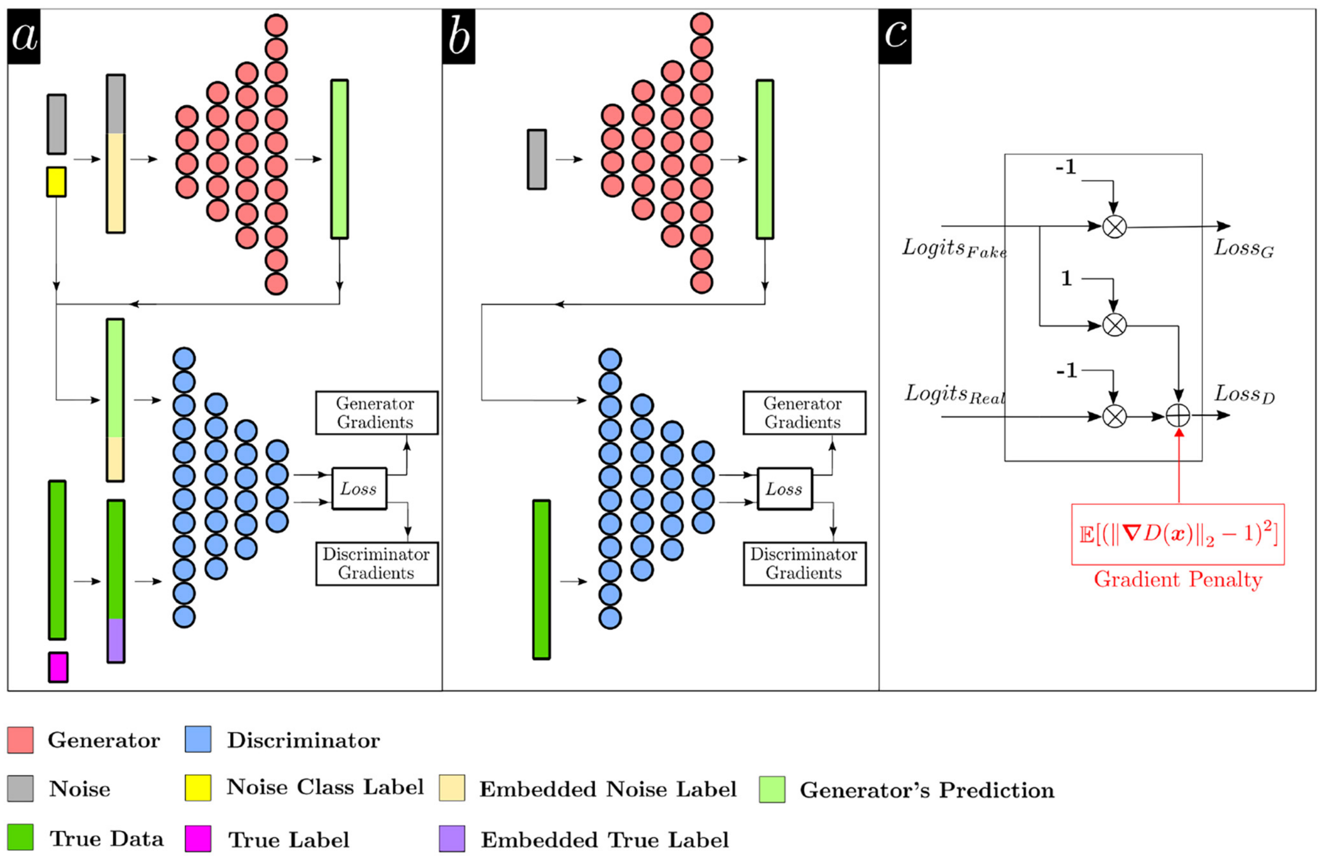


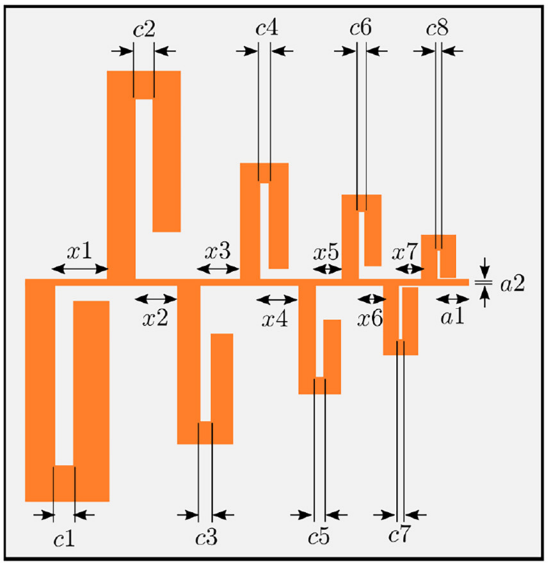
FIGURE 2. Training process of GANs. The generators and discriminators are shown as stacks of layers with gradually varying widths. The noise, vector of structural parameters and labels are shown as stand-alone vectors. Arrows indicate the direction of data flow (a). Conditional GAN (b). vanilla GAN (c). Wasserstein loss function; shown in red is the regularizer term, i.e., gradient penalty, for the WGAN-GP scheme in which $D(x)$ is the discriminator's response to samples from straight lines between real and generated data distributions.

data) and the widths of discriminator's stack of layers are 512/256/128/1. The last layer of the generators are implemented with sigmoid non-linearity while that of the discriminators are implemented using linear activation.

Training of the original GAN is very slow and unstable and the practical *tricks* that were introduced to improve it (flipping the fake label for generator's loss, label smoothing etc.) [25], [26] failed to remedy the problem. The Wasserstein GAN (WGAN) [27] provides a significant improvement in training stability by (1) replacing the original GAN's loss with Wasserstein loss as a measure of the closeness of the predicted distribution and data distribution (2) constraining the discriminator to the set of 1-Lipschitz Continuous functions by confining (clipping) its weights to a compact space $[-c, c]$ (3) updating the discriminator weights to optimality before each update of the generator weights and (4) using RMSprop algorithm [28] for optimization. It must be noted that, the norm of the gradient of a K-Lipschitz Continuous function is at most K at any point on that function. This implies that the function's variations are bounded by a cone formed by lines with slopes $\pm K$. Enforcing K-Lipschitz Continuity on the discriminator ensures that the Wasserstein loss is valid, continuous and differentiable which leads to training stability [29], [30]. WGAN's training was further improved [31] by enforcing the 1-Lipschitz Continuity constraint through a *Gradient Penalty* (GP) instead of weight clipping. In this improved scheme, WGAN-GP, weight clipping and batch normalization are removed from the discriminator and a regularizer, the gradient penalty, is added to to the discriminator's loss and ADAM algorithm [32] is used for optimization. For the purposes of this study, we have adopted the WGAN scheme for our Conditional GAN and the WGAN-GP scheme for the library of vanilla GANs as detailed in Section III. At this point it is worth reiterating that GAN and Conditional GAN refer to architectures while Wasserstein loss and Gradient Penalty, as used in designations WGAN and WGAN-GP, refer to certain strategies for stable training of these architectures.

C. TRAINING DATASET

The geometry of the LPFDA is shown in Fig. 3. The array is designed to operate from 350 *MHz* to 1 *GHz* and its elements are printed on a substrate with $\epsilon_r = 2.33$ and $h = 0.786$ *mm*. The structure is fed from the bottom using a coaxial cable that drives the center conductor. Ten structural parameters were used to form the antenna's vector representation $\boldsymbol{X} = [x_2, x_3, x_4, x_5, c_2, c_3, c_4, c_5, a_1, a_2]^T$ while x_1, x_6, x_7, c_1, c_6, c_7 and c_8 were not used in the dataset and were kept constant



Parameter	Range (*mm*)	
$x1$	34.99	−
$x2$	29.00	38.00
$x3$	26.00	37.00
$x4$	23.00	30.00
$x5$	22.00	28.00
$x6$	19.30	−
$x7$	18.30	−
$c1$	15.83	−
$c2$	13.00	18.00
$c3$	12.00	15.00
$c4$	9.00	12.00
$c5$	8.00	12.00
$c6$	7.20	−
$c7$	6.00	−
$c8$	4.60	−
$a1$	1.50	2.70
$a2$	5.50	6.50

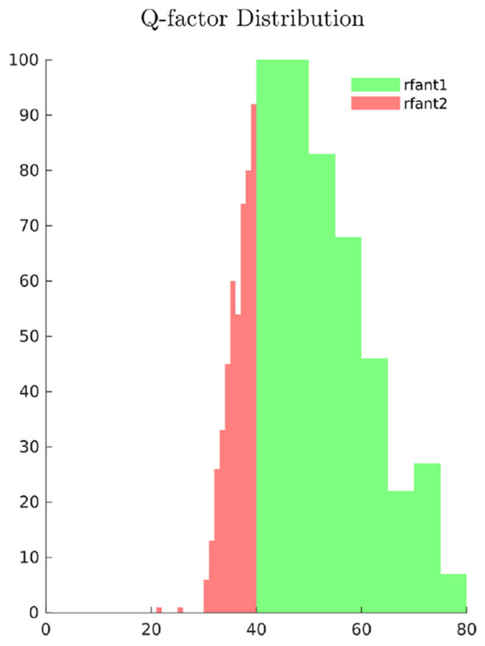


FIGURE 3. The structural parameters of the Log-Periodic Folded Dipole Array.

in the design (x_1 and c_1 fix the lower half of the bandwidth). The Q-factor of an antenna can be calculated from its input impedance $Z(\omega_0) \equiv R(\omega_0) + jX(\omega_0)$ using Eq. (1):

$$Q(\omega_0) = \frac{\omega_0}{2R(\omega_0)}\sqrt{(R'(\omega_0))^2 + \left(X'(\omega_0) + \frac{|X(\omega_0)|}{\omega_0}\right)^2} \quad (1)$$

in which the prime denotes differentiation with respect to frequency [33]. Since LPFDA is a wide-band antenna, we define its Q-factor as the average of Q-factors at several frequencies sampled in its operational bandwidth.

Three datasets were constructed based on the ranges of interest for Q-factor values. In the first dataset, named *rfant 1*, $40 < Q < 80$ and in the second dataset, named *rfant 2*, $Q < 40$. These two datasets were used to train each vanilla GAN separately. The third dataset, named *rfant 3*, is the union of the previous two, with label 1 assigned to examples from *rfant 1* and label 0 assigned to examples from *rfant 2*. This dataset was used to train the CGAN (it must be noted that these 0 and 1 labels denote the class of data and are different from those that are used in the loss function to designate fake and real examples). To generate the datasets, CST Microwave Studio was used to simulate 500 designs (with an average run time of half an hour per design) for each of *rfant 1* and *rfant 2* by randomly selecting values for each parameter from a certain range as shown in Fig. 3.

III. NUMERICAL EXPERIMENTS AND RESULTS

We implemented our models using TensorFlow [34]. In the first phase of our experiments, we used the MNIST (Modified National Institute of Standards and Technology) hand-written digits dataset [35], which is a standard benchmark for computer vision tasks, to validate our architectures and implementation. This provides a visual way to verify that the code setup is working as intended. In other words, it provides a visual proof of convergence for the architecture [36]. These experiments were carried out using an NVIDIA Tesla K40M GPU. It became clear through these experiments that WGAN-GP could not be satisfactorily extended to a conditional architecture, therefore we chose the scheme that worked best for each setting. The results are shown in Fig. 4. As can be seen from the figure, the generators of both the CGAN and the vanilla GAN can generate plausible examples from the dataset and the training metrics (generator loss, discriminator loss and the gradient penalty) are all stable and converge.

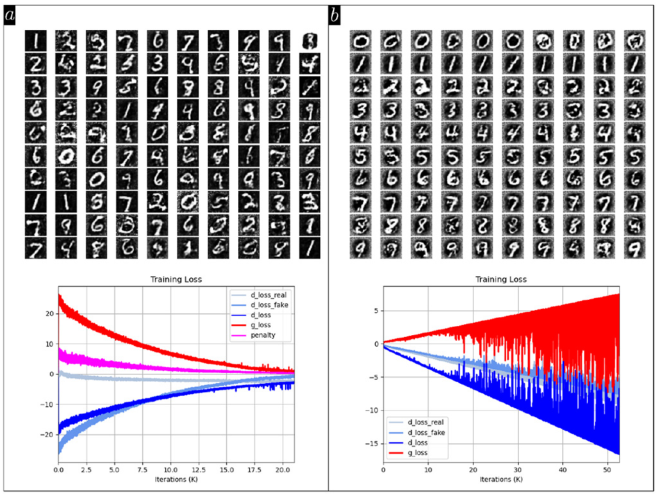


FIGURE 4. Validation of implementations using the MNIST dataset. Plotted are the discriminator's loss on samples from the dataset, *d_loss_real*, discriminator's loss on samples from the generator, *d_loss_fake*, discriminator's total loss, *d_loss*, generator's loss, *g_loss*, and the gradient penalty, *penalty*. (a) vanilla WGAN-GP (b) Conditional WGAN. Ten samples from a single class are shown in each row.

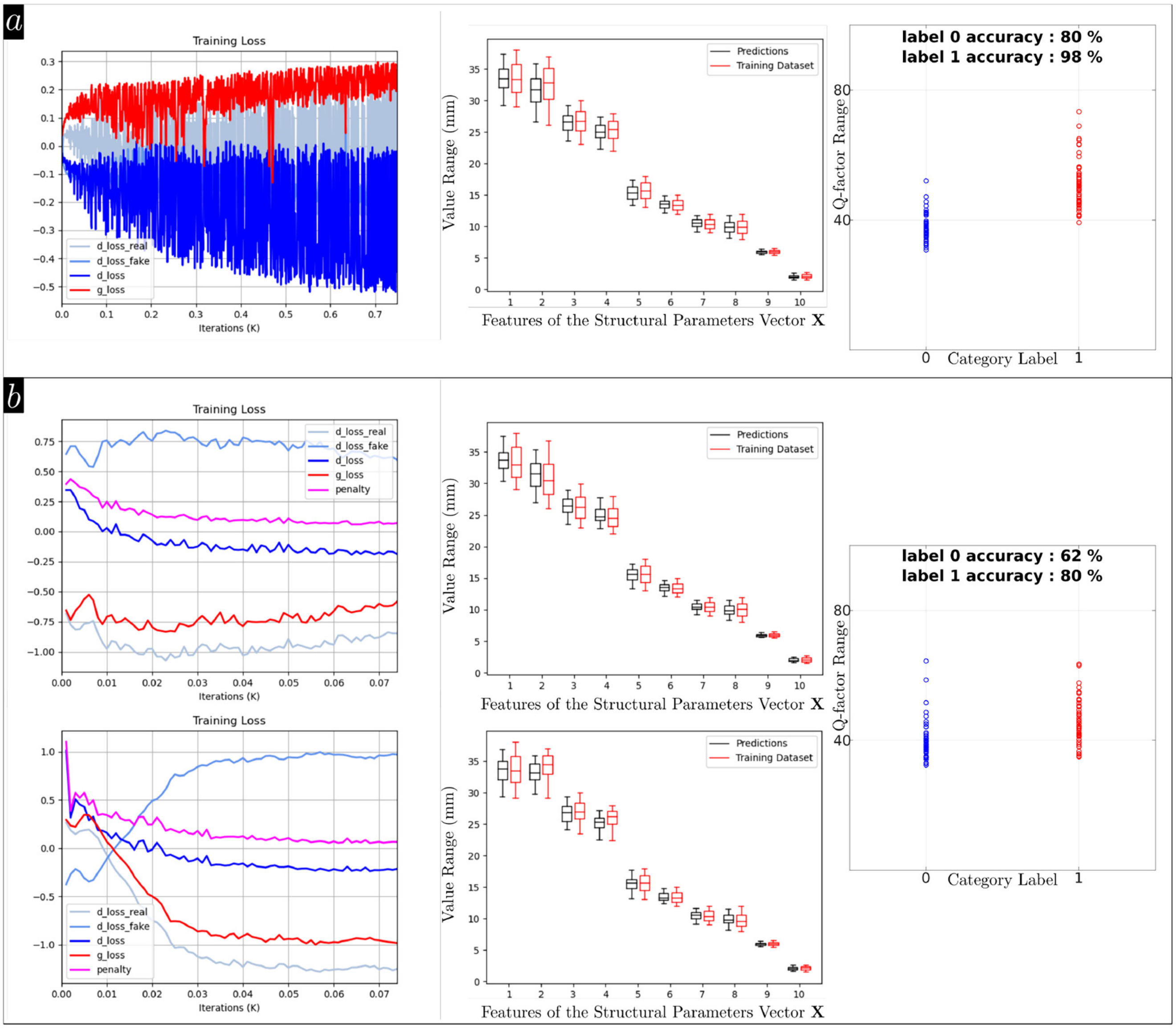


FIGURE 5. **Training loss, statistics (box plots) and accuracy (scatter plots) of the predictions using LPFDA dataset (a) the Conditional GAN and (b) the library of vanilla GANs.**

In the second phase, we trained our networks on LPFDA dataset (*rfant 1*, *rfant 2*, *rfant 3*) using a Desktop PC with Intel Core i7 Processor. Since our training examples are vectors that are formed by putting together various geometric parameters, each element of the vector (or each column of data in the 500×10 dataset) must be considered a separate data channel and scaled with its own statistics (this is similar to scaling each of the Red, Green and Blue channels of a color image separately with its own statistics). To implement this scaling scheme, in a pre-processing step, each column of the dataset was normalized using its own minimum and maximum values (min-max feature scaling) and at inference time, network predictions were scaled back up using the same statistics. For the CGAN, the training time is approximately 10 minutes (for 125 epochs and a batch size of 128). For each vanilla GAN the training time is approximately 1.5 minutes (for 25 epochs and a batch size of 128); Therefore, training the entire library of vanilla GANs takes about 3 minutes.

After training, we used each model to generate 50 predictions for each class. The total run time for each model is about 37 milliseconds (i.e., about 0.4 milliseconds for each generated example). We assessed the accuracy of the predicted structural parameters by simulating them in CST Microwave Studio and calculating the corresponding Q-factor values. Comparisons between the statistics of the predictions and those of the dataset along with the accuracy of the predictions are shown in Fig. 5. Box plots show that both models have demonstrated very good ability in learning the statistics of each parameter; the minimum and maximum of the predictions always fall within the interval defined by those of the dataset and there are significant similarities between the first quartile, median and the third quartile of the two. Scatter plots show that the CGAN outperforms the library of label-switched vanilla GANs. The accuracy of the predictions for labels 0 and 1 are 80% and 98%, respectively, in the case of the CGAN while they are 62% and 80% in the case of the library of label-switched vanilla GANs. The superior performance of CGAN is expected since it is experimentally demonstrated that using side information, e.g., class labels, can significantly improve the quality of generated samples [37]. However, In both cases the accuracy of label 0 is lower than that of label 1; this uneven level of accuracy can be attributed to the much narrower Q-factor distribution of label 0 compared to that of label 1.

IV. CONCLUSION

As computing power increases and physics simulations become more readily available, data-driven models are being utilized in a growing number of ways to facilitate the conventional tasks of simulation and design. In this paper, we introduced a novel inverse design approach in which the optimal structural parameters for a desired response are obtained by querying a learned database of similar optimal structures rather than applying direct optimization (as is the case with the conventional procedures). We successfully demonstrated that, the GAN can be utilized as a learned, data-driven antenna database and queried for fast generation of new designs similar to the ones that it was trained with. The proposed models can learn the data-generating distribution of the structural parameters of suitable designs and can be utilized to generate similar structures from the same distribution. As a further generalization of this approach, a large dataset of antennas with desired responses over a multitude of measures (e.g., Q-factor, radiation pattern, operational bandwidth, half power beam-width, etc.) can be gathered and used to train machines that provide a larger breadth of responses in any given application.

REPRODUCIBILITY

The datasets, code and results can be found at the *GitLab* repository dedicated to this work: https://gitlab.com/oameed/unm_rfant_dloptant.

ACKNOWLEDGMENT

The authors would like to thank the University of New Mexico Center for Advanced Research Computing, supported in part by the National Science Foundation, for providing the high-performance computing resources used in this work.

REFERENCES

[1] W. E. Sorteberg, S. Garasto, A. S. Pouplin, C. D. Cantwell, and A. A. Bharath, "Approximating the solution to wave propagation using deep neural networks," 2018, *arXiv:1812.01609*.

[2] Y. Xie, E. Franz, M. Chu, and N. Thuerey, "TempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1–15, 2018.

[3] B. Kim, V. C. Azevedo, N. Thuerey, T. Kim, M. Gross, and B. Solenthaler, "Deep fluids: A generative network for parameterized fluid simulations," *Comput. Graph. Forum*, vol. 38, no. 2, pp. 59–70, 2019.

[4] S. Fotiadis, E. Pignatelli, M. L. Valencia, C. Cantwell, A. Storkey, and A. A. Bharath, "Comparing recurrent and convolutional neural networks for predicting wave propagation," 2020, *arXiv:2002.08981*.

[5] J. Wen, H. Ma, and X. Luo, "Deep generative smoke simulator: Connecting simulated and real data," *Visual Comput.*, vol. 36, no. 7, pp. 1385–1399, 2020.

[6] J. Xu and K. Duraisamy, "Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics," *Comput. Methods Appl. Mech. Eng.*, vol. 372, Dec. 2020, Art. no. 113379.

[7] O. Noakoasteen, S. Wang, Z. Peng, and C. Christodoulou, "Physics-informed deep neural networks for transient electromagnetic analysis," *IEEE Open J. Antennas Propag.*, vol. 1, pp. 404–412, 2020.

[8] W. Tang *et al.*, "Study on a Poisson's equation solver based on deep learning technique," in *Proc. IEEE Electr. Design Adv. Packag. Syst. Symp. (EDAPS)*, 2017, pp. 1–3.

[9] S. Qi, Y. Wang, Y. Li, X. Wu, Q. Ren, and Y. Ren, "Two-dimensional electromagnetic solver based on deep learning technique," *IEEE J. Multiscale Multiphys. Comput. Techn.*, vol. 5, pp. 83–88, 2020, doi: 10.1109/JMMCT.2020.2995811.

[10] P. R. Wiecha and O. L. Muskens, "Deep learning meets nanophotonics: A generalized accurate predictor for near fields and far fields of arbitrary 3D nanostructures," *Nano Lett.*, vol. 20, no. 1, pp. 329–338, 2019.

[11] M. Martínez-Ramón, A. Gupta, J. L. Rojo-Álvarez, and C. G. Christodoulou, *Machine Learning Applications in Electromagnetics and Antenna Array Processing*. Boston, MA, USA: Artech House, 2021.

[12] J. Jiang, M. Chen, and J. A. Fan, "Deep neural networks for the evaluation and design of photonic devices," *Nat. Rev. Mater.*, vol. 6, pp. 679–700, Aug. 2021. [Online]. Available: https://doi.org/10.1038/s41578-020-00260-1

[13] E. Yilmaz and B. German, "Conditional generative adversarial network framework for airfoil inverse design," in *Proc. AIAA Aviation Forum*, 2020, p. 3185.

[14] D. Liu, Y. Tan, E. Khoram, and Z. Yu, "Training deep neural networks for the inverse design of nanophotonic structures," *ACS Photon.*, vol. 5, no. 4, pp. 1365–1369, 2018.

[15] Z. Liu, D. Zhu, S. P. Rodrigues, K.-T. Lee, and W. Cai, "Generative model for the inverse design of metasurfaces," *Nano Lett.*, vol. 18, no. 10, pp. 6570–6576, 2018.

[16] J. A. Hodge, K. V. Mishra, and A. I. Zaghloul, "RF metasurface array design using deep convolutional generative adversarial networks," in *Proc. IEEE Int. Symp. Phased Array Syst. Technol. (PAST)*, 2019, pp. 1–6.

[17] J. Jiang and J. A. Fan, "Global optimization of dielectric metasurfaces using a physics-driven neural network," *Nano Lett.*, vol. 19, no. 8, pp. 5366–5372, 2019.

[18] J. Jiang, D. Sell, S. Hoyer, J. Hickey, J. Yang, and J. A. Fan, "Free-form diffractive metagrating design based on generative adversarial networks," *ACS Nano*, vol. 13, no. 8, pp. 8872–8878, 2019.

[19] I. J. Goodfellow *et al.*, "Generative adversarial networks," 2014, *arXiv:1406.2661*.

[20] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, *arXiv:1411.1784*.

[21] J. Vijayamohanan, F. Ayoub, M. Patriotis, C. Christodoulou, and J. Lyke, "Peel-off and stick antennas for small unmanned aerial vehicles," in *Proc. IEEE Int. Symp. Antennas Propag. USNC-URSI Radio Sci. Meeting*, 2019, pp. 1117–1118.

[22] J. Vijayamohanan, O. Noakoasteen, A. Gupta, M. Martínez-Ramón, and C. G. Christodoulou, "On antenna Q-factor characterization with generative adversarial networks," in *Proc. IEEE Int. Symp. Antennas Propag. North Amer. Radio Sci. Meeting*, 2020, pp. 1643–1644.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1026–1034.

[24] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[25] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," *Advances in Neural Information Processing Systems*, vol. 29. Red Hook, NY, USA: Curran Assoc., 2016, pp. 2234–2242.

[26] I. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," 2016, *arXiv:1701.00160*.

[27] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," 2017, *arXiv:1701.07875*.

[28] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6A overview of mini-batch gradient descent," *Cited On*, vol. 14, no. 8, p. 2, 2012.

[29] P. Pauli, A. Koch, J. Berberich, P. Kohler, and F. Allgöwer, "Training robust neural networks using Lipschitz bounds," *IEEE Control Syst. Lett.*, vol. 6, pp. 121–126, 2021.

[30] H. Gouk, E. Frank, B. Pfahringer, and M. J. Cree, "Regularisation of neural networks by enforcing Lipschitz continuity," *Mach. Learn.*, vol. 110, no. 2, pp. 393–416, 2021.

[31] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of Wasserstein GANs," 2017, *arXiv:1704.00028*.

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

[33] A. D. Yaghjian and S. R. Best, "Impedance, bandwidth, and Q of antennas," *IEEE Trans. Antennas Propag.*, vol. 53, no. 4, pp. 1298–1324, Apr. 2005.

[34] M. Abadi *et al.* "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems." 2015. [Online]. Available: http://tensorflow.org/

[35] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[36] L. Li, L. G. Wang, F. L. Teixeira, C. Liu, A. Nehorai, and T. J. Cui, "DeepNIS: Deep neural network for nonlinear electromagnetic inverse scattering," *IEEE Trans. Antennas Propag.*, vol. 67, no. 3, pp. 1819–1825, Mar. 2019.

[37] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2642–2651.