Hindawi
International Journal of Antennas and Propagation
Volume 2020, Article ID 2154928, 10 pages
https://doi.org/10.1155/2020/2154928

Research Article

Antenna Optimization Design Based on Deep Gaussian Process Model

Xin-Yu Zhang, Yu-Bo Tian, and Xie Zheng

School of Electronics and Information, Jiangsu University of Science and Technology, Zhenjiang 212003, Jiangsu, China

Correspondence should be addressed to Yu-Bo Tian; tianyubo@just.edu.cn

Received 6 February 2020; Accepted 3 November 2020; Published 12 November 2020

Academic Editor: Rodolfo Araneo

When using Gaussian process (GP) machine learning as a surrogate model combined with the global optimization method for rapid optimization design of electromagnetic problems, a large number of covariance calculations are required, resulting in a calculation volume which is cube of the number of samples and low efficiency. In order to solve this problem, this study constructs a deep GP (DGP) model by using the structural form of convolutional neural network (CNN) and combining it with GP. In this network, GP is used to replace the fully connected layer of the CNN, the convolutional layer and the pooling layer of the CNN are used to reduce the dimension of the input parameters and GP is used to predict output, while particle swarm optimization (PSO) is used algorithm to optimize network structure parameters. The modeling method proposed in this paper can compress the dimensions of the problem to reduce the demand of training samples and effectively improve the modeling efficiency while ensuring the modeling accuracy. In our study, we used the proposed modeling method to optimize the design of a multiband microstrip antenna (MSA) for mobile terminals and obtained good optimization results. The optimized antenna can work in the frequency range of 0.69–0.96 GHz and 1.7–2.76 GHz, covering the wireless LTE 700, GSM 850, GSM 900, DCS 1800, PCS1900, UMTS 2100, LTE 2300, and LTE 2500 frequency bands. It is shown that the DGP network model proposed in this paper can replace the electromagnetic simulation software in the optimization process, so as to reduce the time required for optimization while ensuring the design accuracy.

1. Introduction

At present, solving most of the problems concerning antennas relies on full-wave electromagnetic simulation software. However, using electromagnetic simulation software to analyze the antenna is not only complicated but also computationally expensive [1]. Therefore, many literatures have proposed that artificial neural networks (ANNs) [2], support vector machine (SVMs) [3], and Gaussian process (GP) [4, 5] can be used to analyze antenna problems. ANN can implement parallel processing, self-learning, and nonlinear mapping, but its structure is relatively complicated, which requires a large amount of electromagnetic simulation data, and it is difficult to determine with poor generalization ability [6]. SVM has many unique advantages in solving small samples and nonlinear problems [7] and also has many disadvantages such as difficult selection of kernel parameters, easy overfitting, and prediction output without probabilistic significance [8]. As the machine learning (ML) method has developed rapidly in recent decades, GP has a good adaptability to deal with complex problems such as high dimensions, small samples, and nonlinearities, which is also easier to implement than SVM and ANN. Otherwise, its hyperparameters can be obtained adaptively, and its predicted output value is also of probability significance [9]. Therefore, the GP model can be used as a fast surrogate to obtain accurate full-wave analysis in antenna design, which can greatly reduce the time required for accurate simulation in antenna design while ensuring model accuracy [10]. However, for the GP modeling method, the biggest limitation is that it has relatively high requirements on training data. Therefore, it often uses high-accuracy discrete data sets to ensure that the model has sufficient prediction accuracy. Meanwhile, for calculation with the same amount of training data, GP requires more time [11].

Convolutional neural network (CNN) is a type of feedforward neural network (FNN) that includes convolution calculations and has a deep structure, and it is also one of the representative algorithms of deep learning (DL). In DL, CNN can be understood as a deep neural network (DNN) that can reduce the dimension of data [12] while retaining the value of data, which is widely used in computer vision [13] and natural language processing [14], for its convolutional layer can perform feature extraction on the data and the data passed to the pooling layer can also be used to perform feature selection and information filtering on the data. In the entire CNN, fully connected layers can be regarded as a "classifier." If we say that the convolutional layer, pooling layer, and activation function are used to map the original data to the feature space of hidden layer, then the fully connected layers can map the "distributed feature representation" learned by the CNN network to the sample label space, which is consistent with the ability of the GP to map nonlinear complex problems to high-dimensional space. However, since the traditional GP requires a large number of covariance calculations, the training efficiency of GP model will be very low once there is a large input data dimension. However, the CNN's convolutional layer and pooling layer can reduce the dimensions of the data while retaining the feature value of the data. Based on the above situation, a deep GP (DGP) network modeling method combining CNN and GP models is proposed. Simultaneously, particle swarm optimization (PSO) algorithm is used to optimize the parameters of the DGP network model when training the model. Considering the current research situation, the application of PSO algorithm to optimize CNN and GP has been very mature [15, 16]. Comparing with the traditional error backpropagation (BP) optimization of CNN, PSO is very flexible in optimizing model parameters [17]. Therefore, PSO is selected to optimize the DGP model in this study, while the mean-squared error of the difference value between the prediction output of the model and the training output is used as the fitness function of PSO. In this paper, we applied the proposed DGP model to the design of a multiband antenna [18] for the mobile terminal and obtained good optimization results.

2. Deep Gaussian Process Network Model

2.1. Convolutional Neural Network. The basic structure of CNN consists of input layers, convolutional layers, pooling layers, fully connected layers, and output layers, among which the convolutional layers and the pooling layers usually have multiple layers according to the actual problem. The traditional CNN is composed of forward pass and back propagation, so BP is used to optimize the parameters of the NN and train the NN. We use PSO to optimize the parameters. Figure 1 is a schematic diagram of the structure of the convolutional layer and the pooling layer in the one-dimensional CNN. In this figure, the top layer is the pooling layer, the middle layer is the convolutional layer, and the bottom layer is the input layer of the convolutional layer. The neurons in the convolutional layer constitute each feature surface, and each neuron is connected to the local region of the feature surface of the next layer through a set of convolution kernel. Then, the local weight value is calculated and transferred to a nonlinear activation function. In addition, the weight values of the same feature surface are shared. The parameters and complexity of the model can be reduced by weight value sharing and local connection, making the network easier to train. In order to combine with the GP, we use PSO to optimize the parameters. Therefore, we need to find out the number of parameters that need to be optimized and sort out their positions in the CNN.

Take Figure 1 as an example. The feature surfaces are connected by a 1×3 convolution kernel. If we want to use an equation to represent the connection weight value of the ith neuron on the input feature surface m and the jth neuron on the output feature surface n, the following weight value sharing equation can be obtained: $\omega_{m(i)n(j)}$. An input feature surface can be mapped to the corresponding output feature surface through the translation of a convolution kernel of a fixed size, the convolution operation is performed with all neurons on the input feature surface, and then it can be mapped after weighting and activation. Because of the property of translation convolution kernel, the weight values can be shared for adjacent neurons in the corresponding position, which actually use the same weight value in the convolution kernel. Through the convolution operation, the number of neurons in the feature surface of the convolutional layer or the size of the feature surface satisfies the following formula:

$$\text{OutSize} = \left(\frac{\text{InSize} - \text{CSize}}{\text{CInterval}} + 1\right), \tag{1}$$

where OutSize represents the number of neurons on output feature surface, InSize represents the number of neurons on input feature surface, CSize is the size of the convolution kernel, and CInterval represents the sliding translation step size of the convolution kernel. The number of parameters that can be trained by the convolutional layer is as follows [19]:

$$\text{CPN} = (\text{InSize} \times \text{CSize} + 1) \times \text{OutSize}, \tag{2}$$

where CPN is the number of training parameters and 1 is the number of thresholds, usually only one shared threshold is set for each layer. The activation function in CNN generally uses sigmoid function, tanh function, etc.

The pooling layer is generally constructed on the next layer of the convolutional layer. It also consists of multiple feature surfaces, each of which corresponds to the unique feature surface of the previous layer. The feature of the pooling layer is that it does not change the number of feature surfaces. The number of neurons on the output feature surface of the pooling layer is calculated as follows:

$$\text{OutSize} = \left(\frac{\text{InSize}}{\text{DSize}}\right), \tag{3}$$

where DSize is the size of the pooling kernel. The numerical output formula of any neuron on the output feature surface of the pooling layer is as follows:

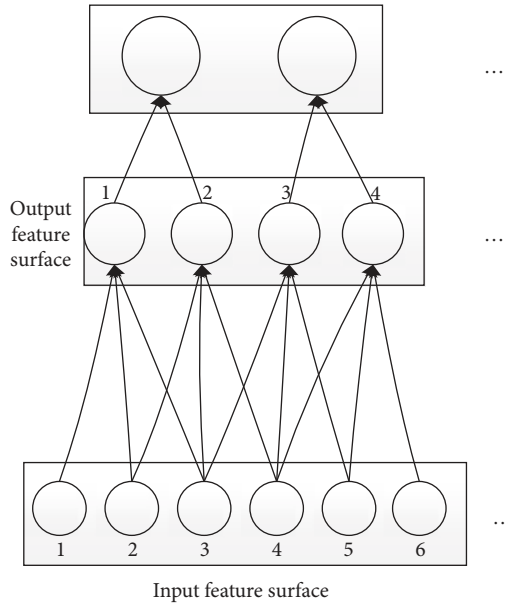

Figure 1: Schematic diagram of convolutional layer and pooling layer.

$$y_{nk}^{\text{out}} = f\left(x_{np}^{\text{in}}, x_{n(p+1)}^{\text{in}}\right), \tag{4}$$

where y_{nk}^{out} is the kth neuron of the nth output surface of the pooling layer, x_{np}^{in} is the pth neuron of the nth input face, and $f(\cdot)$ can be classified into average pooling and maximum pooling according to the different pooling methods. The parameters that need to be optimized are usually in the convolutional layer and the fully connected layer. In the pooling layer, we choose the largest pooling or average pooling, and there are no training parameters [20]. Finally, we replaced the fully connected layer of the traditional CNN with a GP model, so we will not discuss the parameters of the fully connected layer here.

2.2. Gaussian Process. The GP describes a functional distribution. It is a set of infinite random variables, and any subset of these variables conforms to the Gaussian distribution. Its properties can be determined by the average value function $u(\mathbf{x}) = E[Y(\mathbf{x})]$ and the covariance function $C(\mathbf{x}, x') = E[(Y(\mathbf{x}) - u(\mathbf{x}))(Y(x') - u(x'))]$. So, the GP can be defined as follows:

$$f(\mathbf{x}) \sim GP\left(u(\mathbf{x}), C(\mathbf{x}, x')\right), \tag{5}$$

where $\mathbf{x}, x' \in \mathbf{X}$ refers to any d-dimensional vector.

Assume the finite data set $\mathbf{D} = \{(\mathbf{x}^i, \mathbf{t}^i), i = 1, \ldots, n\}$ containing n observed values as the training sample of the Gaussian model, and the observed target value t is polluted by additive noise ε that follows the normal distribution. Then the model can be expressed as follows:

$$t^i = f\left(\mathbf{x}^i\right) + \boldsymbol{\varepsilon}^i, i = 1, \ldots, n \quad R^d \longrightarrow \mathbf{R}, \tag{6}$$

where $\mathbf{x}^i \in \mathbf{X}$ represents the $d \times n$ dimensional training input matrix composed of training input vectors; t^i represents the training output vector composed of the corresponding n training output scalars; and ε refers to the random variable that follows the normal distribution, that is,

$$\boldsymbol{\varepsilon} \sim N\left(0, \boldsymbol{\sigma}_n^2\right). \tag{7}$$

Joint Gaussian prior distribution composed of n training outputs t and n^* testing output t^* is as follows:

$$\begin{bmatrix} \mathbf{t} \\ t^* \end{bmatrix} \sim N\left(0, \begin{bmatrix} \mathbf{C}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I} & \mathbf{C}(\mathbf{X}, \mathbf{x}^*) \\ \mathbf{C}(\mathbf{X}, \mathbf{x}^*) & \mathbf{C}(\mathbf{x}^*, \mathbf{x}^*) \end{bmatrix}\right), \tag{8}$$

where $\mathbf{C}(\mathbf{X}, \mathbf{x}^*)$ is the $n \times n^*$ order covariance matrix between training input and testing input samples and $\mathbf{C}(\mathbf{x}^*, \mathbf{x}^*)$ is the covariance matrix of the testing input sample itself.

On the premise that the testing point $\mathbf{x}^*$ and the training set d is given, the purpose of Bayesian prediction probability is to calculate the probability $P(t^*|tDn, q\mathbf{x}^*)$. Based on Bayesian posterior probability formula, we can get

$$t^*|\mathbf{x}^*, \mathbf{D} \sim N\left(u_{t^*}, \sigma_{t^*}^2\right), \tag{9}$$

where the expected value and variance of t^* are as follows:

$$u_{t^*} = \mathbf{C}\left(\mathbf{x}^*, \mathbf{X}\right)\left(\mathbf{C}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{t}, \tag{10}$$

$$\sigma_{t^*}^2 = \mathbf{C}\left(\mathbf{x}^*, \mathbf{x}^*\right) - \mathbf{C}^T\left(\mathbf{x}^*, \mathbf{X}\right)\left(\mathbf{C} + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{C}\left(\mathbf{x}^*, \mathbf{X}\right). \tag{11}$$

The covariance function of the GP must meet the Mercer condition, that is, for any point set, a non-negative positive definite covariance matrix can be guaranteed. This study

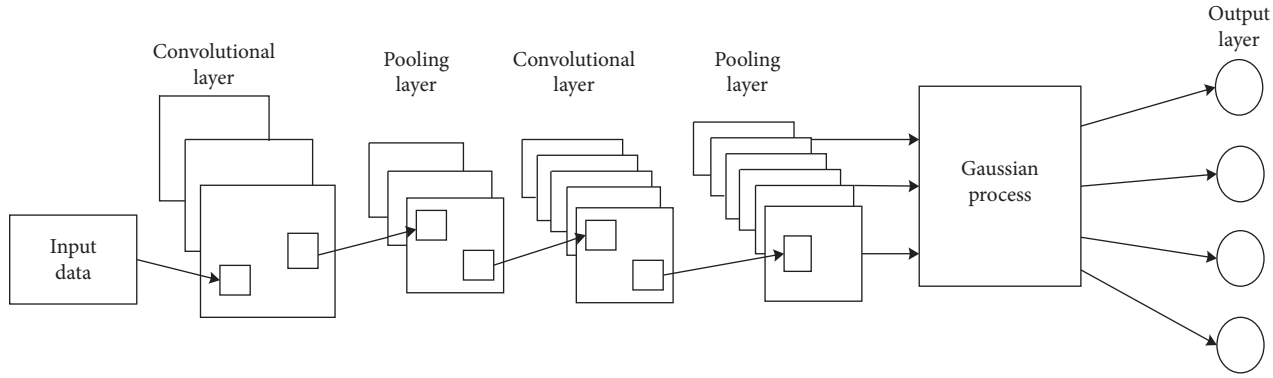

Figure 2: Deep Gaussian process network structure.

chooses the Ardmatern52 covariance function as the covariance function of the GP:

$$\mathbf{C}\left(\mathbf{x}_i, \mathbf{x}_j | t\theta\right) = \sigma_f^2\left(1 + \sqrt{5}\,\mathbf{r} + \frac{5}{3}\mathbf{r}^2\right)\exp\left(-\sqrt{5}\,\mathbf{r}\right), \tag{12}$$

where

$$r = \sqrt{\sum_{m=1}^{d} \frac{\left(x_{im} - x_{jm}\right)^2}{\sigma_m^2}}. \tag{13}$$

where σ_f^2 is the signal variance. The properties of the average function and covariance function of the GP are determined by a set of hyperparameters, which is also the only parameter that needs to be determined for the GP [21].

2.3. Particle Swarm Optimization. We adopted PSO algorithm for optimization. PSO algorithm is easy to implement, simple, with less parameters, and can effectively solve the global optimization problems [22]. In the standard PSO algorithm, the particle swarm consists of particles, and the position of each particle is assumed to be a possible prepared solution to the problem in the dimensional search space. The particle updates its flight track based on its inertia, optimal position, and swarm optimal position.

The basic idea of the PSO algorithm is to accelerate each particle to approach the best position of itself and the swarm. In the solution space, the starting position and speed of the particles will be randomly set. During the iterative search process, the algorithm will record the best positions experienced by individual particles and swarms and the corresponding fitness function values. The speed and position update formula of the particle swarm algorithm is as follows:

$$v_{i,d}^{k+1} = v_{i,d}^{k} + c_1 \mathrm{rand}\left(p_{i,d}^{k} - x_{i,d}^{k}\right) + c_2 \mathrm{rand}\left(p_{g,d}^{k} - x_{i,d}^{k}\right), \tag{14}$$

$$x_{i,d}^{k+1} = x_{i,d}^{k} + v_{i,d}^{k+1}, \tag{15}$$

where c_1 and c_2 refer to the learning factor and the acceleration constant; rand () is a random number between (0, 1); $v_{i,d}^{k}$ and $x_{i,d}^{k}$ refer to the d-dimensional speed and position of particle i in the kth iteration; $p_{i,d}^{k}$ refers to the position of the individual extreme value of particle i in the dth dimension; and $p_{g,d}^{k}$ refers to the position of the global extreme value of the swarm in the dth dimension. In this paper, we optimized the parameters of the proposed DGP model globally using the PSO algorithm in the training of the model, so that the prediction accuracy of the model after the training is completed can replace the traditional electromagnetic simulation software. After that, the PSO algorithm is used again to optimize the antenna based on the trained model.

2.4. The Proposed Deep Gaussian Process. The deep Gaussian process (DGP) network model is the combination of the CNN and the GP, which is shown in Figure 2. The GP replaces the fully connected layer of the CNN, while retaining the input layer, output layer, convolutional layer, and pooling layer of the CNN. The convolutional layer is used to retain the feature quantity of the input data, the pooling layer is used to reduce the data dimension, and the GP is used to predict the output of the object. The overall structure of the model is evolved from LeNet-5 [23] (a common conventional structure of CNN). However, it is more flexible than LeNet-5 because its overall structure has been improved with two or more layers of convolutional layers and pooling layers. Otherwise, the specific number of layers can be set according to actual needs or multiple convolutions can be carried out according to the size of the data or the order of convolutions and pooling can be changed.

In the DGP network modeling method, the samples required for model training, that is, the training input and training output, can be obtained by the electromagnetic simulation software HFSS. In this paper, VBScript language is used to realize the data exchange between MATLAB software and HFSS software, which makes the acquisition of training data more concise and automatic. After obtaining the training data, it would be uniformly normalized. Assume that each group of input data is $\mathbf{x}$ with the size of $1 \times n$ and the size is $1 \times (n-1)$ after passing the convolutional layer with the convolution kernel of 1×2. The data are then passed through the activation function before inputting the pooling layer, which can turn the linear data into discrete data. After adding nonlinear factors, the network model's ability to understand the problem can be improved [24]. Therefore, in

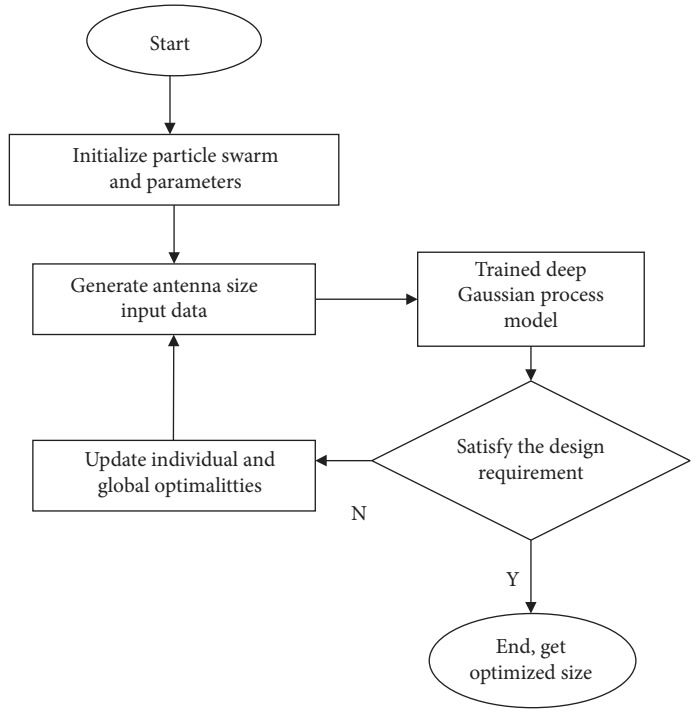

Figure 3: Flowchart of optimal design.

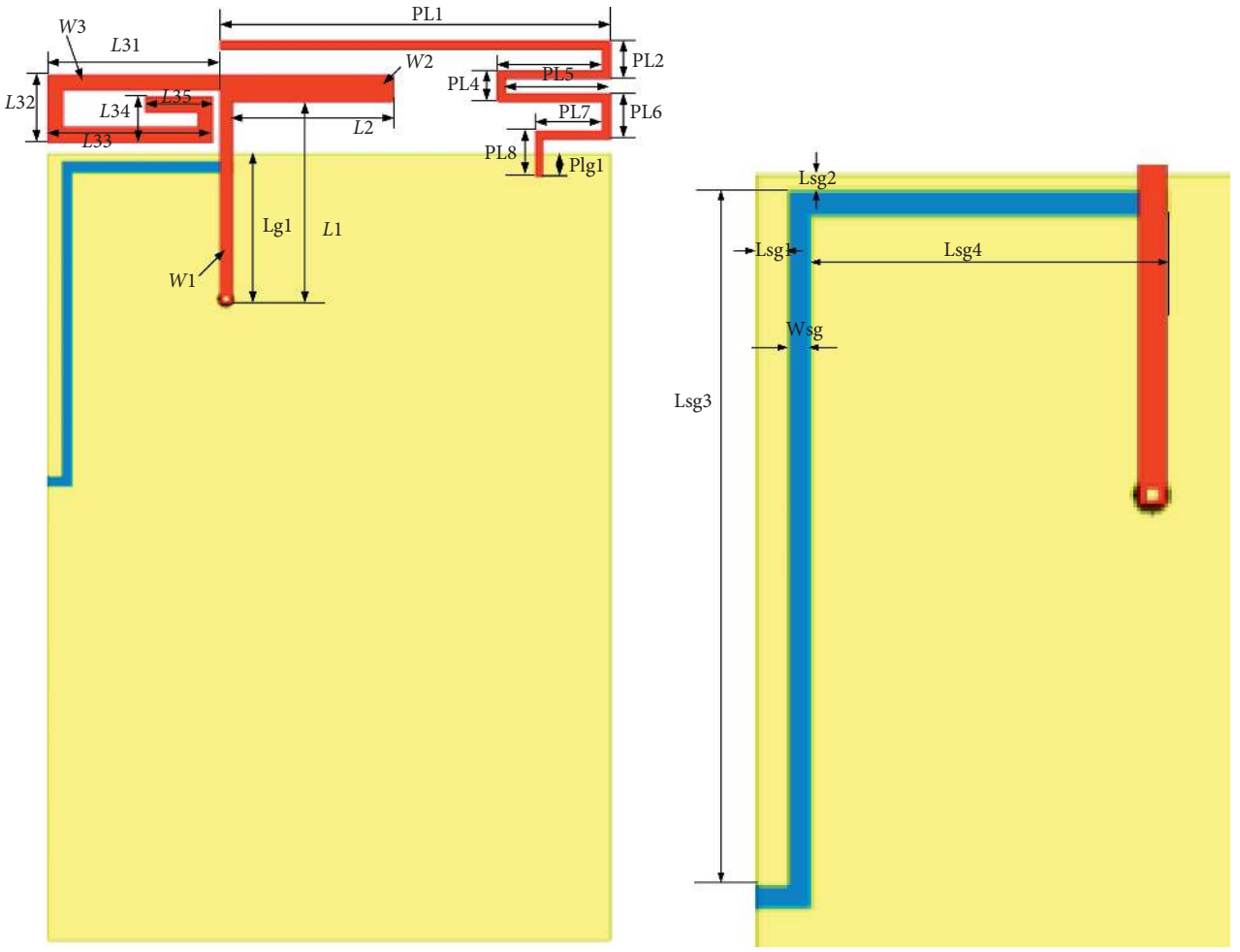

Figure 4: Schematic diagram of the multiband microstrip antenna.

Table 1: Optimization parameters of the multiband microstrip antenna.

Variable name	Variable range (unit: mm)	Variable name	Variable range (unit: mm)
$L32$	8.5~10	PL4	3.5~4.5
$L33$	15~22	PL5	12~18
$L35$	6~12	PL6	5~7
$W1$	1~2	PL7	7~12
$W2$	3~4	PL8	5~7
$W3$	1.5~2.5	Plg1	3.5~4.5
$L1$	27~29	Lsg1	1~3
$L2$	21~24	Lsg2	1~3
PL1	50~54	Lsg3	40~45
PL2	4.5~5.5	Wsg	1~2

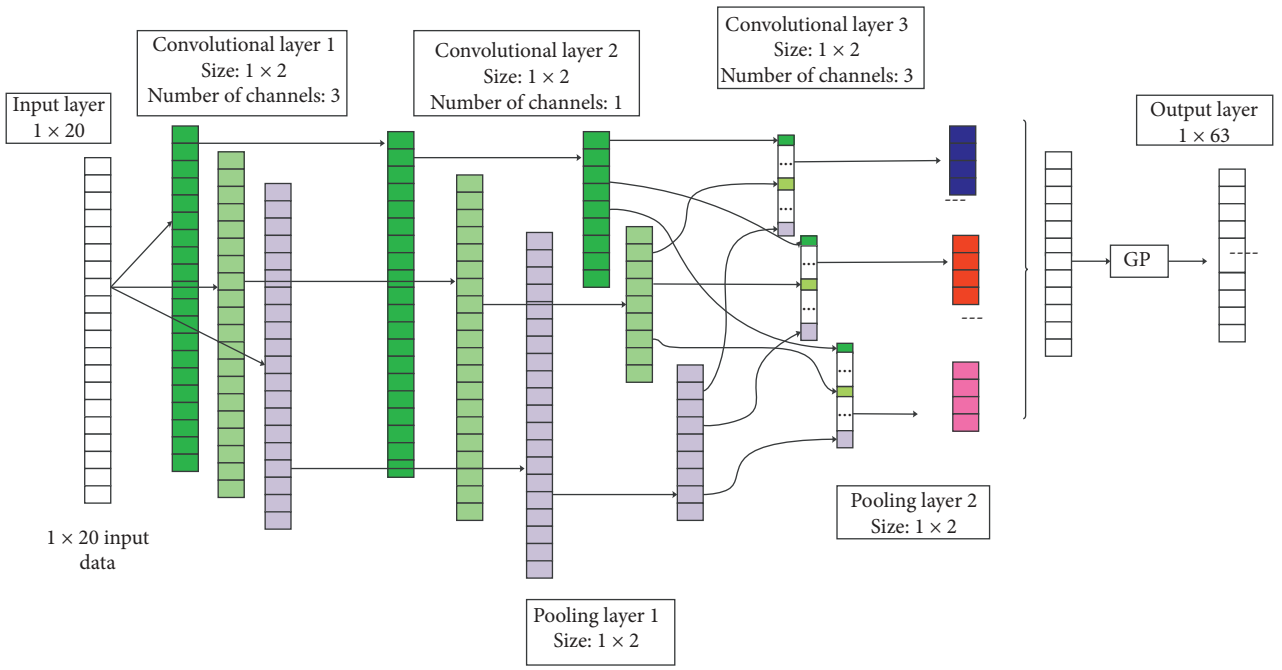

Figure 5: The DGP network model structure.

Table 2: Optimized parameters of the multiband microstrip antenna.

Variable name	Variable range (unit: mm)	Variable name	Variable range (unit: mm)
$L32$	9.66	PL4	4.02
$L33$	17.96	PL5	17.66
$L35$	10.05	PL6	6.27
$W1$	1.09	PL7	11.79
$W2$	3.26	PL8	6.75
$W3$	1.65	Plg1	3.74
$L1$	27.56	Lsg1	1.58
$L2$	22.32	Lsg2	2.34
PL1	52.11	Lsg3	43.47
PL2	4.95	Wsg	1.068

this paper, sigmoid is used as the activation function. Otherwise, the input data completed by convolutions and pooling are used as the input of the DGP, and the mean-squared error of the output of the DGP and the training output is used as the fitness function for PSO-based training. Finally, the output of the model is reversely normalized to get the real predicted output value given by the model.

The trained DGP network can finally be used for antenna optimization design. The process of optimization design is shown in Figure 3.

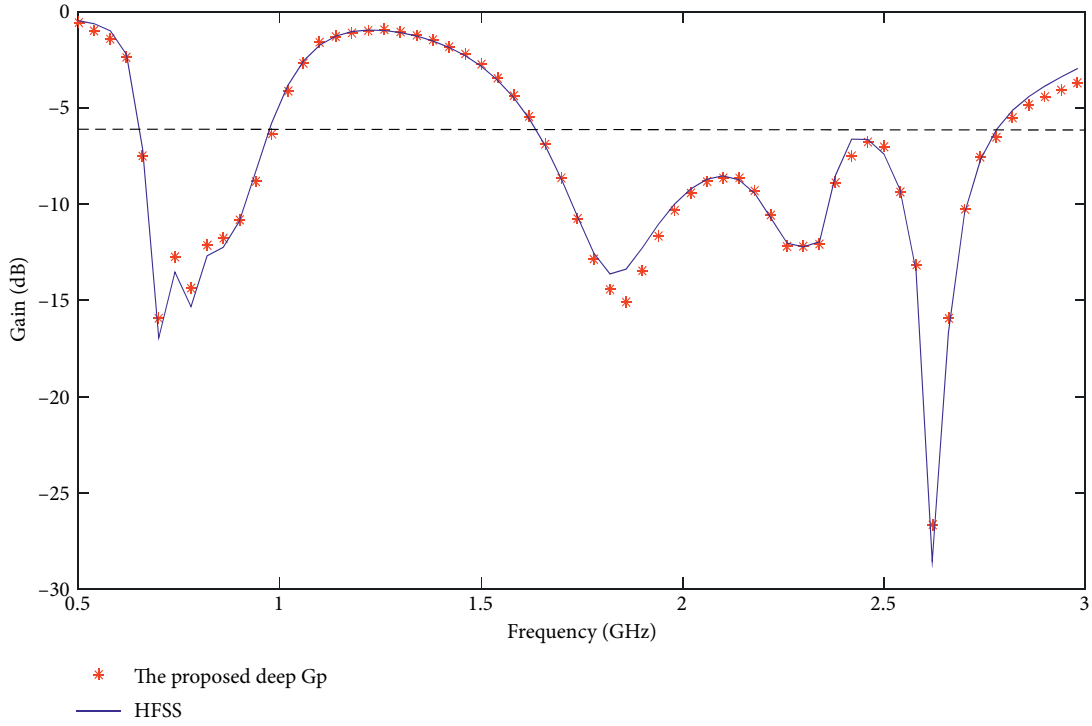

FIGURE 6: S_{11} comparison results of the multiband microstrip antenna.

3. Multiband Microstrip Antenna Application

3.1. Design of Multiband Antenna. In recent years, the 4G system of LTE has matured and developed worldwide, and the 5G communication technology has also been gradually and widely used [25]. Under this background, the performance requirements of antennas are increasing [26]. To meet the requirements of many wireless communication standards, the antennas of mobile terminal should cover multiple frequency bands or broadband [27]. In addition, with the popularity of ultrathin mobile phones, the space of the antenna is limited, so it is of research value to design the antenna in terms of small size and frequency band [28].

The antenna in [18] is changed from a T-shaped monopole antenna, and the size of the FR4 substrate is 75 (width) mm × 120 (length) mm × 0.8 (thickness) mm. The right side of the T-shaped antenna is a parasitic open strip, and the left side is a slot etched on the ground. The overall structure is shown in Figure 4. Through the simulation of HFSS electromagnetic software, we know the antenna has 4 resonant frequency bands. In this paper, we use the DGP network to perform optimization design of the microstrip antenna (MSA) to make it in the S_{11} less than 6 dB to cover the impedance bandwidth of 270 MHz (0.69 to 0.96 GHz) and 1.06 GHz (1.7 to 2.76 GHz), so that we can cover the wireless LTE 700, GSM 850, GSM 900, DCS 1800, PCS1900, UMTS 2100, LTE 2300, and LTE 2500 frequency bands.

3.2. Model Training and Prediction. During the modeling process, the 20 size parameters of the antenna (as shown in Table 1) are used as variables and randomly combined into 200 groups of different antenna parameters as input data of the DGP network. HFSS is transferred for simulation, and the obtained simulation results are taken as the training output to train the DGP network model. The mean-squared error is used as the fitness function of PSO during the training process. If the model prediction accuracy does not meet the requirements, then we would continue to train iteratively by PSO until the model meets the accuracy requirements.

The proposed DGP network model used here has 3 convolutional layers and 2 pooling layers. The size of the convolution kernel of each convolutional layer is 1×2, the number of channels of convolutional layer 1 is 3, the number of channels of convolutional layer 2 is 1, the number of channels of convolutional layer 3 is 3, and the size of pooling layer of each layer is 1×2. Figure 5 shows the specific structure of the DGP network model for multiband antennas. After the input data of 1×20 enter the model, the dimension size after the pooling layer 2 is 1×4 for every channel, which can greatly reduce the training time of GP and improve the training efficiency of GP. After a series of convolutions and pooling processes, for the GP, at this point, the input training data size is 1×12, and the output data are the S_{11} amplitude corresponding to the frequency points sampled in the frequency band. The specific frequency band range is 0.5 GHz–3 GHz, and the sampling interval is 0.04 GHz, with 63 frequency points in each group.

After training, we use PSO to optimize the design. The number of particles in the PSO algorithm is 20, the maximum number of iterations is 500, the acceleration constant is $c_1 = c_2 = 2$, the inertia weight w is 1, and the fitness function is less than −6 dB in the frequency range of

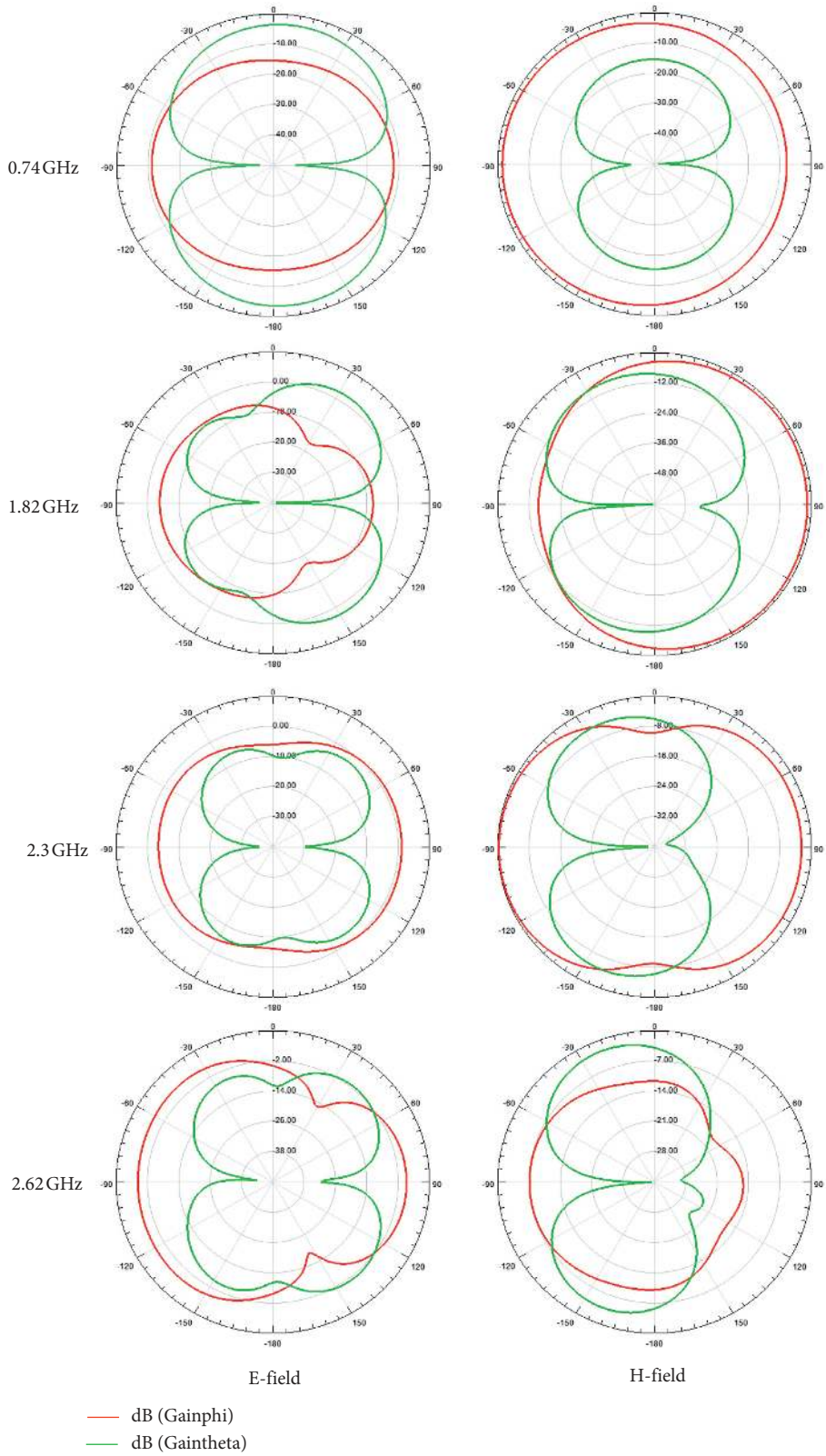

Figure 7: Field pattern of the antenna.

0.69–0.96 GHz and 1.7–2.76 GHz. The optimized size parameters are shown in Table 2. In order to verify the validity and accuracy of the model, Figure 6 shows the comparison results between the S_{11} predicted by the proposed method and the simulation results of the electromagnetic simulation software HFSS, and Figure 7 is the field pattern of the

antenna in its 4 frequency points. The above results prove that the optimized antenna can meet the optimization objectives.

4. Conclusion

This study proposes a modeling method based on deep Gaussian process networks. In the framework of deep learning, this paper first utilizes the advantages of convolutional neural network and Gaussian process and creatively combines these advantages. Then, we take advantage of the convolutional neural network to reduce the input data dimension without losing data characteristics and finally use the Gaussian process adaptability to the nonlinear problem to predict antenna frequency, so as to guarantee the accuracy and reduce the calculation time, thereby improving the efficiency. Meanwhile, the proposed deep Gaussian process network model combined with the PSO algorithm is adopted to conduct optimization design. The optimized results are very close to the results obtained by the HFSS high-fidelity model simulation, indicating that the modeling method is sufficiently reliable. The optimized antenna size meets the requirements of the index, showing that the method has practical value in the antenna optimization design.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant no. 61771225, the Natural Science Foundation of Jiangsu Province of China under Grant no. BK20190956, and the Qinglan Project of Jiangsu Higher Education.

References

[1] S. Koziel and A. Bekasiewicz, "On reduced-cost design-oriented constrained surrogate modeling of antenna structures," *IEEE Antennas and Wireless Propagation Letters*, vol. 16, pp. 1618–1621, 2017.

[2] Y. Chen, Y.-B. Tian, and F.-Y. Sun, "KBNN based on coarse mesh to optimize the EBG structures," *International Journal of Antennas and Propagation*, vol. 1, no. 4, pp. 155–164, 2017.

[3] F. Y. Sun, Y. B. Tian, G. B. Hu, and Q. Y. Shen, "DOA estimation based on support vector machine ensemble," *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, vol. 32, no. 5, p. e2614, 2019.

[4] X. Chen, Y. Tian, T. Zhang, and J. Gao, "Differential evolution based manifold gaussian process machine learning for microwave filter's parameter extraction," *IEEE Access*, vol. 8, pp. 146450–146462, 2020.

[5] J. Gao, Y. Tian, X. Zheng, and X. Chen, "Resonant frequency modeling of microwave antennas using Gaussian process based on semisupervised learning," *Complexity*, vol. 2020, Article ID 3485469, 12 pages, 2020.

[6] L. Yuan, X.-S. Yang, C. Wang, and B.-Z. Wang, "Multibranch artificial neural network modeling for inverse estimation of antenna array directivity," *IEEE Transactions on Antennas and Propagation*, vol. 68, no. 6, pp. 4417–4427, 2020.

[7] S. Fei-Yan, T. Yu-Bo, and R. Zuo-Lin, "Modeling the resonant frequency of compact microstrip antenna by the PSO-based SVM with the hybrid kernel function," *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, vol. 29, no. 6, pp. 1129–1139, 2016.

[8] N. T. Tokan and F. Gunes, "Support vector characterisation of the microstrip antennas based on measurements," *Progress in Electromagnetics Research B*, vol. 5, pp. 49–61, 2008.

[9] Y.-X. Xu and Y.-B. Tian, "Optimal design of conical horn antenna based on GP model with coarse mesh," *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, vol. 43, no. 4, pp. 717–724, 2019.

[10] J. P. Jacobs and S. Koziel, "Two-stage framework for efficient gaussian process modeling of antenna input characteristics," *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 2, pp. 706–713, 2014.

[11] X. Fan, Y. Tian, and Y. Zhao, "Optimal design of multiband microstrip antennas by self-renewing fitness estimation of particle swarm optimization algorithm," *International Journal of Antennas and Propagation*, vol. 2019, Article ID 2176518, 9 pages, 2019.

[12] J. Xu, B. Xu, P. Wang et al., "Self-taught convolutional neural networks for short text clustering," *Neural Networks*, vol. 88, pp. 22–31, 2017.

[13] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu et al., "Convolutional neural networks for medical image analysis: full training or fine tuning?" *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.

[14] T. Shen, T. Zhou, G. Long et al., "DiSAN: directional self-attention network for rnn/cnn-free language understanding," in *Proceedings of the National Conference on Artificial Intelligence*, pp. 5446–5455, New Orleans, USA, August 2018.

[15] X. -H. Fan, Y.-B. Tian, and Y. Zhao, "Optimal design of microwave devices by fitness-estimation-based particle swarm optimization algorithm," *The Applied Computational Electromagnetics Society Journal*, vol. 33, no. 11, pp. 1259–1267, 2018.

[16] F. Zhang, Z. Yan, and Z. Du, "Preoperative planning for the multi-arm surgical robot using pso-gp-based performance optimization," in *Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, Singapore, May 2017.

[17] F. Chen and Y. b. Tian, "CUDA-based PSO-trained neural network for computation of resonant frequency of circular microstrip antenna," *International Journal of Computational Science and Engineering*, vol. 14, no. 3, pp. 211–221, 2017.

[18] J. Y. Deng, S. Y. Liu, and D. Q. Sun, "Multiband antenna for mobile terminals," *International Journal of RF and Microwave Computer-Aided Engineering*, vol. 29, no. 11, pp. 745–754, 2019.

[19] J. Gu, Z. Wang, J. Kuen et al., "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2017.

[20] H.-C. Shin, H. R. Roth, M. Gao et al., "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[21] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Machine learning of linear differential equations using gaussian processes," *Journal of Computational Physics*, vol. 348, pp. 683–693, 2017.

[22] Yu-Bo Tian, *Particle Swarm Optimization Algorithm and Electromagnetic Applications*, Science Press, Beijing, China, 2014.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[24] P. M. Long and H. Sedghi, "On the effect of the activation function on the distribution of hidden nodes in a deep network," *Neural Computation*, vol. 31, no. 12, pp. 2562–2580, 2019.

[25] M. Lauridsen, L. C. Gimenez, I. Rodriguez, and T. B. Sorensen, "Mogensen from LTE to 5G for connected mobility," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 156–162, 2017.

[26] H.-D. Mogensen, H.-W. Yang, and C.-Y.-D. Sim, "Single open-slot antenna for LTE/WWAN smartphone application," *IEEE Transactions on Antennas and Propagation*, vol. 65, no. 8, pp. 4278–4282, 2017.

[27] R. Wu and Q.-X. Chu, "Multi-mode broadband antenna for 2G/3G/LTE/5G wireless communication," *Electronics Letters*, vol. 54, no. 10, pp. 614–616, 2018.

[28] S. Verma, L. Mahajan, R. Kumar, H. S. Saini, and N. Kumar, "A small microstrip patch antenna for future 5G applications," in *Proceedings of the 2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, pp. 7–9, Noida, India, September 2016.