I. INTRODUCTION

With the concurrent rise of artificial intelligence and quantum information science, these two fields are merging in a synergistic manner. In this growing trend, some works try to design new theoretical models based on quantum algorithms to improve classical machine learning for desired quantum speed-up [1–10]. At the same time, with the ever-increasing complexity of quantum systems, advanced quantum information technologies also require more powerful tools for data processing and data analysis. Therefore, on the other hand, we urgently need to combine existing classical machine learning techniques to solve practical, but difficult, problems in quantum information science, such as tomography [11–13], classifying quantum states [14–16], quantum metrology [17–19] and quantum control [20–22].

Quantum key distribution (QKD) [23, 24] is by far the most practical technology in quantum information. It allows two distant parties (Alice and Bob) to establish secure keys against any eavesdropper. In recent decades, various QKD protocols have been proposed one after another [25]. Among them, continuous-variable (CV) QKD has its own distinct advantages at a metropolitan distance [26] due to the use of common components of coherent optical communication technology. In addition, owing to the extraordinary spectral filtering capability inherent in homodyne [27] or heterodyne [28] measurements, the crosstalk in wavelength division multiplexing (WDM) channels would be effectively suppressed. In this way, a single optical fiber containing multiplexing of hundreds of QKD channels and co-propagation with classical data channels will be possible, leading integration into existing communication network more efficient. In CV-QKD, discrete modulation technology has attracted much attention [29–40] because of its ability to reduce the requirements for modulation devices. However, due to the lack of symmetry, the corresponding security proof is more complicated. Therefore, its security proof has so far mainly relied on numerical methods [32–37, 41]. In fact, numerical methods are widely used to calculate the secure key rate in a large number of quantum key distribution protocols [42–46], including device-independent [47] and measurement-device-independent [48] protocols.

However, calculating the key rate by numerical methods requires minimizing a convex function over all eavesdropping attacks related with the experimental data [45, 46]. The efficiency of this optimization depends on the number of parameters of the QKD protocol. For example, in discrete modulation CV-QKD, the number of parameters is generally 1000 – 3000 depending on the different choices of photon-number cutoff [33]. This leads to the corresponding optimization may take minutes or even hours [41]. On the other hand, with the development of global quantum networks [49] , there is an increasing interest in implementing the QKD protocols on moving platforms [50–52], which requires low latency and low cost for calculating key rates. Therefore, it is especially important to develop tools for calculating the key rate that are more efficient than numerical methods.

In this work, we take the homodyne detection discrete-modulated CV-QKD [33] as an example to construct neural networks capable of predicting the secure key rate for the purpose of saving time and resource consumption. We apply our neural networks to a test set obtained at different excess noises and distances. Excellent accuracy and time savings are observed after adjusting the hyper-
parameters. Our work provides an end-to-end learning model that directly predicts the key rate after our neural networks are trained. Through some open source deep learning frameworks for on-device inference, such as TensorFlow Lite [53], our model can be easily deployed on devices at the edge of the network, such as mobile devices, embedded Linux, or microcontrollers.

II. DISCRETE-MODULATED CV-QKD

To clearly show the problem we try to solve, we briefly introduce the main ideas of discrete-modulated CV-QKD and give the convex optimization problem of finding its key rates in this section. See Ref. [33] and Appendix A for a detailed description of discrete-modulated CV-QKD.

The protocol involves two parties, Alice and Bob. Alice randomly prepares one of the four coherent states and sends it to Bob via an untrusted quantum channel. Bob measures the received coherent state using homodyne detection. After repeating $N$ rounds, Alice and Bob perform sifting, parameter estimation, error correction and privacy amplification over the classical authentication channel to obtain the final secure key rates. The key rate formula in the asymptotic limit can be expressed according to Refs. [42, 43] as

$$R^\infty = \min_{\rho_{AB} \in S} D(\mathcal{G}(\rho_{AB}) \parallel \mathcal{Z}(\mathcal{G}(\rho_{AB}))) - p_{pass} \deltaEC,$$  

where $D(\rho || \sigma) = \text{Tr} (\rho \log_2 \rho) - \text{Tr} (\rho \log_2 \sigma)$ is the quantum relative entropy; $\rho_{AB}$ is the bipartite state of Alice and Bob; $\mathcal{G}$ is the mapping to describe the post-processing of the bipartite state $\rho_{AB}$; $\mathcal{Z}$ is a pinching quantum channel for reading out the results of the key rate mapping; $S$ is the set of all density operators that match the experimental observations; $p_{pass}$ is a sifting factor that determines how many rounds of data are used for generating keys; $\deltaEC$ represents the amount of information leakage per bit in the error-correction process.

The key to finding the secure key rates is to find the minimum value of $D(\mathcal{G}(\rho_{AB}) \parallel \mathcal{Z}(\mathcal{G}(\rho_{AB})))$, since $p_{pass} \deltaEC$ is a fixed quantity. The associated optimization problem is [33]

$$\begin{align*}
\text{minimize} & \quad D(\mathcal{G}(\rho_{AB}) \parallel \mathcal{Z}(\mathcal{G}(\rho_{AB}))) \\
\text{subject to} & \quad \text{Tr} [\rho_{AB} (|x\rangle \langle x|_A \otimes \hat{q})] = p_x \langle \hat{q} \rangle_x, \\
& \quad \text{Tr} [\rho_{AB} (|x\rangle \langle x|_A \otimes \hat{p})] = p_x \langle \hat{p} \rangle_x, \\
& \quad \text{Tr} [\rho_{AB} (|x\rangle \langle x|_A \otimes \hat{n})] = p_x \langle \hat{n} \rangle_x, \\
& \quad \text{Tr} [\rho_{AB} (|x\rangle \langle x|_A \otimes \hat{d})] = p_x \langle \hat{d} \rangle_x, \\
& \quad \text{Tr} [\rho_{AB}] = 1, \\
& \quad \rho_{AB} \geq 0, \\
& \quad \text{Tr}_B [\rho_{AB}] = \sum_{i,j=0}^3 \sqrt{p_{ij}} |\varphi_j \rangle |i\rangle \langle j|_A,
\end{align*}$$

where $|x\rangle \langle x|_A$ is a local projective measurement operator of Alice’s side, where $x \in \{0, 1, 2, 3\}$; $\hat{q} = \frac{1}{\sqrt{2}} (\hat{a}^\dagger + \hat{a})$, where $\hat{a}$ and $\hat{a}^\dagger$ are the annihilation and creation operators of a single-mode state, respectively; $\hat{p} = \frac{1}{\sqrt{2}} (\hat{a}^\dagger - \hat{a})$; $\hat{n} = \frac{1}{2} (\hat{q}^2 + \hat{p}^2 - 1) = \hat{a}^\dagger \hat{a}; \hat{d} = \hat{q}^2 - \hat{p}^2 = \hat{a}^2 + (\hat{a}^\dagger)^2$; $|\hat{q}\rangle_x, |\hat{p}\rangle_x, |\hat{n}\rangle_x$ and $|\hat{d}\rangle_x$ represent the corresponding expectation values of the operators $\hat{q}, \hat{p}, \hat{n}$ and $\hat{d}$ acting on $\rho_B^D$, respectively; $\rho_B^D = \frac{1}{p_x} \text{Tr}_A [\rho_{AB} (|x\rangle \langle x|_A \otimes \text{id}_B)]$ is the state of Bob after Alice has performed measurement $|x\rangle \langle x|$ on $\rho_{AB}$, and $p_x$ is the corresponding probability; $\text{id}_B$ is the identity transformation acting on system $B$.

The first four constraints in Eq. (2) are derived from experimental observations. The fifth and sixth constraints are conditions that the density matrix must satisfy. The seventh constraint comes from the fact that Alice’s states do not change because they do not go through insecure quantum channels.

The optimization problem in Eq. (2) is to find the optimal $\rho_{AB}$ in $S$ such that $R^\infty$ is minimized. $\rho_{AB}$ is infinite-dimensional because the attacker has the ability to arbitrarily perturb the optical mode sent by Alice into an infinite-dimensional state to send to Bob. To solve this optimization problem using numerical methods, we need to apply the photon-number cutoff assumption to $\rho_{AB}$ to ensure that the number of variables is in a reasonable range. A detailed description of this method can be found in Ref. [33].

After applying the photon-number cutoff assumption, the optimization problem in Eq. (2) can be solved by applying the numerical method in Refs. [33, 43], but this is very time consuming. In this work, to reduce the time to predict secure key rates, we use the key rates obtained by the numerical method in Refs. [33, 43] as labels to train our neural networks.

III. NEURAL NETWORKS FOR PREDICTING THE KEY RATES

We use artificial neural networks to predict the key rates of discrete-modulated CV-QKD. The general spirit
of the work is to encode the optimization problem in Eq. (2) on the loss function of a feedforward neural network, and train the neural network by minimizing this loss function. The trained neural network can be seen as a mapping, which has learned the structure of the training set. For new instances, the neural network outputs the results directly via mapping, unlike traditional numerical methods that perform complex searches. As a result, the trained neural network saves a great deal of time, while ensuring a good level of accuracy. A more detailed description of neural networks can be found in the Ref. [54].

A four-layer neural network model is designed to predict the key rates of discrete-modulated CV-QKD (Fig. 1). The input layer of the network has 29 neurons, which are used to receive the training inputs. The first hidden layer and the second hidden layer of the network have 400 and 200 neurons respectively, and their activation functions are the tanh function and the sigmoid function, respectively. The output layer has only one neuron, which is used to predict secure key rates.

To train our neural networks, we generate the data set containing 552,000 training inputs \{\vec{x}_i\} and 552,000 corresponding labels \{y_i\} using the numerical method in Refs. [33, 43]. Each \vec{x}_i \in \{\vec{x}\} represents a vector of 29 variables, and label \text{y}_i represents the corresponding key rate. There are 16 variables in each \vec{x}_i that are the right parts of the first four restrictions of Eq. (2), 12 variables in each \vec{x}_i are non-diagonal elements of the right side matrix of the fifth restriction of Eq. (2), and the remaining variable is excess noise \xi. The 29 variables in each \vec{x}_i can be calculated in the experiment by using experimental parameters and experimental observations. In our simulation, These random training inputs \{\vec{x}_i\} are generated directly from seven experimental parameters (transmission distance \text{L}, light intensity \text{\mu}, excess noise \xi, and probability p0, p1, p2 and p3) and the following method.

When the excess noise \xi is within 0.002 – 0.014, we first generate a two-dimensional grid with excess noise and distance in the horizontal and vertical coordinates, respectively. Specifically, the value of the distance is between 0 and 100 km in a step of 5 km. The value of the excess noise is between 0.002 and 0.014 in a step of 0.001. Then, each grid point is sampled 80 times. With each sampling, the excess noise fluctuates around the exact value and the float range is 0.0005 up and down. Once the excess noise for this sampling is determined, the light intensity will take a value every 0.01 between 0.35 and 0.60. Each sampling needs to generate 25 training inputs with a positive key rate, otherwise the current round of sampling will be discarded and restarted. In this way, 2000 training inputs are generated on each grid point. Correspondingly, a total of 520,000 training inputs are generated on this two-dimensional grid. When the excess noise \xi is 0.015, a similar two-dimension grid is generated. But we only sample to 80 km, so only 32,000 training inputs are generated. In this way, we collect a total of 552,000 samples for training neural networks with the excess noise \xi between 0.002 and 0.015. Using the above numerical approach, we generate the data set on the blade cluster system of the High Performance Computing Center of Nanjing University. We consume over 40,000 core hours and the node we used contains 4 Intel Xeon Gold 6248 CPUs, which involves immense computational power.

To improve the convergence speed and accuracy of our neural networks, we preprocess the training inputs \{\vec{x}_i\} and the corresponding labels \{\text{y}_i\} before training the neural networks. To demonstrate the necessity of data pre-processing, we use the network structure shown in Fig. 1 to do a controlled experiment with the mean square error as the loss function. With the excess noise of 0.002 – 0.005, the absolute values of the relative deviations between the key rates predicted by our neural networks and the corresponding key rates obtained by the numerical method do not exceed 25% after the data pre-processing (Fig. 2), whereas the absolute values of the relative deviations exceed 400% without the data pre-processing. Here the relative deviation is the absolute deviation between predicted value and true value divided by the true value. A detailed description of the data pre-processing can be found in Appendix B.

A new loss function is specifically designed to make key rates predicted by our neural networks as information-theoretically secure as possible, rather than using the traditional mean squared error as a loss function. The expression of the loss function is as follows:
FIG. 2. Relative deviations before and after data pre-processing. We use the network structure shown in Fig. 1 with the mean square error as the loss function to compare the results of data pre-processing (a) and without data pre-processing (b). The data set is generated under the excess noise of 0.002 – 0.005, and is split into a training set containing 158000 samples and a test set containing 2000 samples. The horizontal coordinate represents the different samples in the test set. The vertical coordinate represents the relative deviations between the key rate predicted by our neural network and the key rate obtained by the numerical method at each sample.

\[
C = \frac{1}{n} \sum_{i=1}^{n} \gamma \left( e_i^* + \max(e_i^*, -\log_{10}(\varepsilon)) - (1 - \gamma)(\min(e_i^*, 0)) \right),
\]

where \( n \) is the number of the training inputs. \( e_i^* = y_i^{\text{pre}} - y_i^{\text{true}} \) is the residual error between the pre-processed label \( y_i^{\text{pre}} \) and the corresponding output \( y_i^{\text{true}} \) of the neural networks.

The minimum function part in Eq. (3) is the penalty term and is used to make the key rates predicted by the neural network as information-theoretically secure as possible. On the other hand, the part consisting of the maximum function and the squared term in Eq. (3) is used to bound the upper limit of \( e_i^* \) to obtain higher key rates. The parameter \( \gamma \) is used to balance the effects of the two parts.

The performance of the neural networks is related to hyperparameters \( \gamma \) and \( \epsilon \). Without loss of generality, we take the examples of neural networks with the excess noise \( \xi \) between 0.002 and 0.005 (Fig. 3). When \( \gamma = 0.20 \) and \( \epsilon = 0.80 \), key rates predicted by the neural network are strictly lower than those obtained by the numerical method in Refs. [33, 43], which means that the key rates predicted by the neural network are information-theoretically secure. Meanwhile, the absolute values of the relative deviations are mainly distributed between 0.05 and 0.20 (Fig. 3a-b). Fig. 3c-f plot the corresponding results for the hyperparameters \( \gamma = 0.20, \epsilon = 0.90 \) and \( \gamma = 0.80, \epsilon = 0.80 \), respectively. Note that the partial key rates predicted by the neural networks under

\( \gamma = 0.20, \epsilon = 0.90 \) and \( \gamma = 0.80, \epsilon = 0.80 \) are higher than key rates obtained by the numerical method. This indicates that the performance of neural networks trained with hyperparameters \( \gamma = 0.20, \epsilon = 0.90 \) and \( \gamma = 0.80, \epsilon = 0.80 \) is not as good as that of neural network trained with hyperparameters \( \gamma = 0.20 \) and \( \epsilon = 0.80 \). Therefore, we need to carefully tune hyperparameters of the neural networks to ensure their stable performance.

The 552,000 data generated by the numerical method are split into a training set containing 524,400 data and a test set containing 27,600 data. The procedure of data preprocessing follows hard data splitting. The Adam optimization algorithm [55] is used to train our neural networks. The initial learning rate is set to 0.001. For each training, we set 200 epoches and 256 batch sizes. In addition, techniques such as early stopping and dropout [56] are used to prevent overfitting. The relative deviations of the trained network on the test set and the training set have similar distribution, which indicates that the model has good generalization performance.

IV. RESULT

We use our neural network to predict, given the optimal light intensity, key rates of discrete-modulated CV-QKD at different distances and different excess noises after training the neural network under \( \gamma = 0.20 \) and \( \epsilon = 0.80 \) according to our method described above. As shown in Fig. 4, we compare the key rates with the corresponding key rates obtained by the numerical method
in Refs. [33, 43]. The results show that all key rates predicted by the neural network are strictly lower than those obtained by the numerical method. It is worth noting that the relative deviations between them are basically within 20% (Relevant data can be found in Appendix C).

To illustrate the more general case, we test the above mentioned test set containing 27,600 samples. The results show that the number of samples, for which the key rates predicted by the neural network are lower than the corresponding results calculated by the numerical method, is 27379. Namely, the probability that the key rate predicted by the neural network of the sample in the above mentioned test set is secure is as high as 99.2%.

Our neural networks show greater advantages over the numerical method on time and resource consumption. We compare the time required to predict the key rates with our neural network and the time required to calculate the key rates with the numerical method on a high-performance personal computer with a 3.3GHz AMD Ryzen 9 4900H and 16GB of RAM (Fig. 5). The neural network is 6 – 8 orders of magnitude faster than the numerical method for predicting the key rates of the discrete-modulated CV-QKD within 0 – 100 km for excess noise $\xi = 0.008 – 0.012$. In addition, as the excess noise increases, the speed of the neural network increases even more. Refer to Appendix C for more detailed data.

V. DISCUSSION

We have constructed neural networks and shown that these neural networks can predict the information-theoretically secure key rates of homodyne detection discrete-modulated CV-QKD with a great probability (up to 99.2%) at a distance of 0 – 100 km and an excess noise of no more than 0.015. In particular, with the excess noise up to 0.008 or more, the speed of our method is at least improved by six orders of magnitude compared to that of the numerical method in Refs. [33, 43]. For example, it takes an average of 190 seconds to numerically calculate the point with the excess noise $\xi$ around 0.008, which greatly affects the efficiency of QKD systems to calculate the secure key rate. In contrast, the neural networks can calculate tens of thousands of key rates in one second. Considering that it takes a certain amount of time for QKD system to collect data, the speed of predicting the key rates by the neural networks completely meets practical applications. This advantage brings us one step closer to achieving low latency for discrete modulated CV-QKD on a low-power platform.

Recently, there are two main types of situations in which machine learning is used in QKD. One is used to experimental parameter optimization [57, 58] and the other is used to assist experimental control [59–61]. They
all use machine learning to replace traditional optimization or feedback control algorithms, which are significantly different from our work. For the first time, we apply machine learning methods to predict key rates of QKD. This poses a greater challenge than parameter optimization with machine learning methods. This is because the parameters predicted by the neural networks are substituted into numerical or analytical methods to find the corresponding key rates, which naturally ensures that the key rates are information-theoretically secure. However, the key rates obtained by neural networks do not guarantee this naturally, which forces us to redesign the loss function and seek better data preprocessing methods to guarantee the acquired key rate with information-theoretic security.

We expect that larger excess noises and longer distances will require deeper networks, more sophisticated loss functions, and more detailed data preprocessing methods to improve the performance of neural networks on the training set. At the same time, more training data are necessary to improve the generalization ability of the neural networks. For deep neural networks, the rapid growth or rapid disappearance of the transmitted gradient will hinder the optimization process, the debugging process is therefore highly technical. The debugging process can be guided by monitoring the activation function values of the neurons and the histograms of those gradient [54].

Our machine learning approach is at least six orders of magnitude faster than the numerical method at predicting the secure key rates of homodyne detection discrete-modulated CV-QKD with the excess noise up to 0.008 or more. However, training our neural network is still time consuming. This is because we need to use traditional numerical methods to obtain a number of key rates as the training set of the neural networks. In particular, the performance of our neural network is dependent on the choice of hyperparameters γ, ϵ and initial learning rate. This means that we may need to train several times to get suitable neural networks. To make our machine learning method more intelligent, further work is necessary to design another neural network to automatically find the most suitable hyperparameters. We have also tried other machine learning methods, such as boosting decision trees. These methods have smaller relative deviations, but have greater variances. We have left the fusion of these methods to future research.

The important contribution of our work is that it opens the door to using classical machine learning to predict QKD key rates. In particular, our ideas and methods are very easy to be generalized to other QKD protocols. We expect that our work will stimulate further researches to help most QKD systems run on low-power chips in mobile devices.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (Grant No. 61801420); the Key-Area Research and Development Program of Guangdong Province (Grant No. 2020B0303040001); the Fundamental Research Funds for the Central Universities (Grant No. 020414380182). The authors would like to
Appendix A: Discrete-modulated CV-QKD

According to Ref. [33], homodyne detection discrete-modulated CV-QKD is described below:

(1) State preparation.—Alice prepares a coherent state $|\psi_k\rangle$ from the set $\{|\alpha\rangle, | -\alpha\rangle, |i\alpha\rangle, | -i\alpha\rangle\}$ according to the probability of $[p_A/2, p_A/2, (1 - p_A)/2, (1 - p_A)/2]$, where $\alpha \in R$ is a pre-determined amplitude and $k$ is the number of rounds. Then Alice sends the state $|\psi_k\rangle$ to Bob.

(2) Measurement.—Bob performs a homodyne measurement on the received state. He chooses to measure a convolution quadratures for each one of the four coherent states sent by Alice. Then they calculate the secret key rate based on the convex optimization problem in Eq. (8).

(3) Announcement and sifting.—After repeating the first two steps $N$ times, Alice and Bob communicate via the classical authentication channel and divide the obtained data into the following four subsets:

$$
\mathcal{I}_{qq} = \{k \in [N] : |\psi_k\rangle \in \{|\alpha\rangle, |-\alpha\rangle\}, b_k = 0\},
\mathcal{I}_{qD} = \{k \in [N] : |\psi_k\rangle \in \{|\alpha\rangle, |-\alpha\rangle\}, b_k = 1\},
\mathcal{I}_{pD} = \{k \in [N] : |\psi_k\rangle \in \{|i\alpha\rangle, |-i\alpha\rangle\}, b_k = 0\},
\mathcal{I}_{pp} = \{k \in [N] : |\psi_k\rangle \in \{|i\alpha\rangle, |-i\alpha\rangle\}, b_k = 1\},
$$
(A1)

where $[N]$ denotes the set of all integers from 1 to $N$. Then Alice and Bob randomly select a subset $\mathcal{I}_{\text{key}}$ of size $m$ from $\mathcal{I}_{qq}$ for generating keys. The key string $X = (x_1, x_2, \ldots, x_m)$ at Alice is also determined according to the following rules:

$$
\forall j \in [m], \quad x_j = \begin{cases} 
0 & \text{if } |\psi_{f(j)}\rangle = |\alpha\rangle, \\
1 & \text{if } |\psi_{f(j)}\rangle = |-\alpha\rangle,
\end{cases}
$$
(A2)

where $f(j)$ is a function that maps from $\mathcal{I}_{\text{key}}$ to $\mathcal{I}_{qq}$. The remaining data in $\mathcal{I}_{qq}$, $\mathcal{I}_{qD}$, $\mathcal{I}_{pD}$ and $\mathcal{I}_{pp}$ are integrated into the set $\mathcal{I}_{\text{test}}$ and used for parameter estimation.

(4) Parameter estimation.—Alice and Bob perform parameter estimation based on the data in $\mathcal{I}_{\text{test}}$. First, they calculate the first and second moments of $q$ and $p$ quadratures for each one of the four coherent states sent by Alice. Then they calculate the secret key rate based on the convex optimization problem in Eq. (8).

If the result shows that the key rate is equal to 0, Alice and Bob abort the protocol and start over. Otherwise, they continue with the next step.

(5) Reverse reconciliation key map.—The key string $Z = (z_1, z_2, \ldots, z_m)$ at Bob is determined according to Bob’s measurement outcome $y_k$ in step 2 and the following rules:

$$
z_j = \begin{cases} 
0 & \text{if } y_{f(j)} \in [\Delta_e, \infty), \\
1 & \text{if } y_{f(j)} \in (-\infty, \Delta_e], \\
\bot & \text{if } y_{f(j)} \in (-\Delta_e, \Delta_e),
\end{cases}
$$
(A3)

where $\Delta_e \geq 0$ is determined by the postselection of data. Alice and Bob then pick out the location of the symbol $\bot$ and remove the data at that location by classical communication. The set $X$ and $Z$ after removing $\bot$ is the raw key string.

(6) Error correction and privacy amplification.—Alice and Bob choose a suitable error-correction protocol and a suitable privacy-amplification protocol to generate secret key rates.

The key rate can be calculated using the well-known Devetak-Winter formula [62] in the asymptotic limit and under collective attacks. To apply this formula, we transform the prepare-and-measure protocol into the entanglement-based protocol.

Alice prepares the state according to the ensemble $\{|\varphi_x\rangle, p_x\}$ in the prepare-and-measure protocol. In the equivalent entanglement-based protocol, Alice prepares the bipartite state in the form of $|\Psi_{AA'}\rangle = \sum_x \sqrt{p_x} |\varphi_x\rangle_A |\varphi_x\rangle_{A'}$. Here Alice keeps $|x\rangle_A$ in register $A$ and sends $A'$ to Bob. $|\varphi_x\rangle_{A'}$ changes as it passes through an insecure quantum channel. The process can be described by a completely positive and trace-preserving map $E_{A'\rightarrow B}$. The bipartite state $\rho_{AB}$ thus transforms into

$$
\rho_{AB} = (\text{id}_A \otimes E_{A'\rightarrow B})(|\Psi\rangle \langle \Psi|)_{AA'},
$$
(A4)

where $\text{id}_A$ is the identity transformation acting on $A$. Under reverse reconciliation [63], the key rate formula can be expressed according to Refs. [42, 43] as

$$
R^\infty = \min_{\rho_{AB} \in S} D(G(\rho_{AB}) || Z(G(\rho_{AB}))) - p_{\text{pass}} \delta_{EC}.
$$
(A5)

Appendix B: Details of data pre-processing

To improve the performance of our neural networks, we preprocess the training inputs $\{\bar{x}_i\}$ before training the neural networks. The process can be expressed as

$$
x^*_ij = \frac{x_{ij} - \bar{x}_j}{\sigma_j},
$$
(B1)

where $x_{ij}$ represents the $j$-th component of the $i$-th sample; $\bar{x}_j$ and $\sigma_j$ are the mean and variance of the $j$-th component in all sample, respectively; $x^*_{ij}$ is the $j$-th component of the $i$-th sample after being preprocessed.

The pre-processed data $\{x^*_{ij}\}$ follow a standard normal distribution with a mean of 0 and a variance of 1. The
TABLE I. Relative deviations between key rates predicted by our neural network and the corresponding key rates obtained by the numerical method for the given optimal light intensity at different distances and different excess noises.

| L (km) | Relative deviations |
|-------|---------------------|
|       | \( \xi = 0.002 \) | \( \xi = 0.005 \) | \( \xi = 0.008 \) | \( \xi = 0.011 \) | \( \xi = 0.014 \) |
| 5     | 0.27               | 0.16               | 0.15               | 0.16               | 0.15               |
| 10    | 0.17               | 0.14               | 0.14               | 0.12               | 0.12               |
| 15    | 0.14               | 0.12               | 0.11               | 0.10               | 0.10               |
| 20    | 0.12               | 0.10               | 0.09               | 0.08               | 0.09               |
| 25    | 0.11               | 0.09               | 0.08               | 0.07               | 0.10               |
| 30    | 0.09               | 0.09               | 0.06               | 0.08               | 0.11               |
| 35    | 0.08               | 0.07               | 0.06               | 0.09               | 0.11               |
| 40    | 0.07               | 0.07               | 0.08               | 0.10               | 0.13               |
| 45    | 0.07               | 0.08               | 0.09               | 0.10               | 0.14               |
| 50    | 0.08               | 0.09               | 0.10               | 0.11               | 0.14               |
| 55    | 0.09               | 0.10               | 0.11               | 0.12               | 0.14               |
| 60    | 0.09               | 0.11               | 0.11               | 0.12               | 0.14               |
| 65    | 0.10               | 0.11               | 0.12               | 0.13               | 0.14               |
| 70    | 0.10               | 0.11               | 0.12               | 0.13               | 0.13               |
| 75    | 0.10               | 0.12               | 0.13               | 0.13               | 0.15               |
| 80    | 0.11               | 0.13               | 0.13               | 0.14               | 0.17               |
| 85    | 0.11               | 0.13               | 0.14               | 0.14               | 0.20               |
| 90    | 0.11               | 0.14               | 0.14               | 0.15               | 0.19               |
| 95    | 0.11               | 0.14               | 0.14               | 0.15               | 0.19               |
| 100   | 0.11               | 0.14               | 0.14               | 0.14               | 0.06               |

TABLE II. Time consumption of the neural network versus the numerical method with the excess noise \( \xi = 0.008 \), 0.010 and 0.012. NM and NN are the abbreviations of numerical method and neural network, respectively. L is the distance between Alice and Bob.

| L(km) | \( \xi = 0.008 \) | \( \xi = 0.010 \) | \( \xi = 0.012 \) |
|-------|-------------------|-------------------|-------------------|
|       | L(km) NM(s) NN(s) | L(km) NM(s) NN(s) | L(km) NM(s) NN(s) |
| 5     | 1.42 \times 10^2  | 5.4 \times 10^{-2} | 2.16 \times 10^2  | 3.11 \times 10^{-2} |
| 10    | 7.86 \times 10^1  | 7.25 \times 10^{-3} | 10.1 \times 10^2  | 4.7 \times 10^{-3}  |
| 15    | 1.04 \times 10^2  | 6.6 \times 10^{-3} | 1.2 \times 10^2   | 4.15 \times 10^{-3} |
| 20    | 1.09 \times 10^2  | 6.5 \times 10^{-3} | 2.37 \times 10^2  | 3.4 \times 10^{-3}  |
| 25    | 2.12 \times 10^2  | 6.6 \times 10^{-3} | 6.39 \times 10^2  | 2.4 \times 10^{-3}  |
| 30    | 1.98 \times 10^2  | 5.9 \times 10^{-3} | 3.3 \times 10^2   | 4.7 \times 10^{-3}  |
| 35    | 2.34 \times 10^2  | 5.9 \times 10^{-3} | 3.1 \times 10^2   | 4.65 \times 10^{-3} |
| 40    | 2.47 \times 10^2  | 5.7 \times 10^{-3} | 4.18 \times 10^2  | 4.6 \times 10^{-3}  |
| 45    | 2.50 \times 10^2  | 6.1 \times 10^{-3} | 2.73 \times 10^2  | 4.35 \times 10^{-3} |
| 50    | 2.62 \times 10^2  | 6.35 \times 10^{-3} | 6.24 \times 10^2  | 4.55 \times 10^{-3} |
| 55    | 2.74 \times 10^2  | 6.5 \times 10^{-3} | 6.5 \times 10^2   | 4.3 \times 10^{-3}  |
| 60    | 2.68 \times 10^2  | 6.65 \times 10^{-3} | 5.28 \times 10^2  | 4.3 \times 10^{-3}  |
| 65    | 2.55 \times 10^2  | 6.7 \times 10^{-3} | 5.48 \times 10^2  | 4.2 \times 10^{-3}  |
| 70    | 2.72 \times 10^2  | 6.55 \times 10^{-3} | 4.82 \times 10^2  | 5.3 \times 10^{-3}  |
| 75    | 2.60 \times 10^2  | 6.7 \times 10^{-3} | 4.78 \times 10^2  | 4.1 \times 10^{-3}  |
| 80    | 2.30 \times 10^2  | 6.0 \times 10^{-3} | 4.19 \times 10^2  | 4.35 \times 10^{-3} |
| 85    | 2.34 \times 10^2  | 5.7 \times 10^{-3} | 3.63 \times 10^2  | 4.35 \times 10^{-3} |
| 90    | 1.99 \times 10^2  | 5.75 \times 10^{-3} | 3.48 \times 10^2  | 4.1 \times 10^{-3}  |
| 95    | 1.72 \times 10^2  | 5.75 \times 10^{-3} | 2.92 \times 10^2  | 4.35 \times 10^{-3} |
| 100   | 1.54 \times 10^2  | 5.85 \times 10^{-3} | 2.43 \times 10^2  | 4.65 \times 10^{-3} |

The process removes dimension restrictions and facilitates comparison of features of different dimensions. Since the maximum difference between different key rates in these samples is 4 orders of magnitude, we preprocess the labels as follows to speed up the training process of the neural networks:

\[
y_i^* = -\log_{10}(y_i),
\]

where \( y_i^* \) is the label corresponding to the \( i \)-th sample after being preprocessed. Note that the outputs predicted by the neural networks trained with pre-processed labels
Algorithm 1: Training stage  
Input: \( \{(x_i, y_i)\} \)  // Original training data set of discrete-modulated CV-QKD collected from the numerical method. \( x_i \) is feature vector containing 29 variables, and \( y_i \) is the corresponding key rate.  
Input: \( \gamma, \epsilon \)  // Two hyperparameters in our self-designed loss function.  
Output: \( \{\theta_r\} \)  // The final learned weights of the neural network \( \mathcal{R} \).  

Preprocessing \( \{\tilde{x}_i^r\} \leftarrow \{\tilde{x}_i\} \)  
- Calculate the mean vector \( \bar{x} \) of \( \{\tilde{x}_i\} \)  
- Calculate the variance vector \( \bar{\sigma} \) of \( \{\tilde{x}_i\} \)  
if \( \sigma_j = 0 \) then  
\[ x_{ij}^r = x_{ij} \]  
else  
\[ x_{ij}^r = (x_{ij} - \mu_j)/\sigma_j \]  
Preprocessing \( \{y_i^r\} \leftarrow \{y_i\} \)  
\[ y_i^r = -\log_{10}(y_i) \]  
Train the neural network under \( \{\gamma, \epsilon\} \) with \( \{(\tilde{x}_i^r, y_i^r)\} \)  
return \( \{\theta_r\} \)  

Algorithm 2: Inference stage  
Input: \( \{\tilde{x}_i\} \)  // A set of original feature vectors containing 29 variables collected from experiment.  
Output: \( \{y_i\} \)  // A set of corresponding key rates predicted by Neural network \( \mathcal{R}_c \).  

Preprocessing \( \{\tilde{x}_i^r\} \leftarrow \{\tilde{x}_i\} \)  
for \( \tilde{x}_i^r \in \{\tilde{x}^r\} \) do  
\[ y_i^r = \mathcal{R}_c(\tilde{x}_i^r) \]  
\[ y_i = 10^{-y_i^r} \]  
end  
return \( \{y_i\} \)  

\[ y_i^p = 10^{-y_i^r}, \]  
where \( y_i^r \) and \( y_i^p \) are the output value and the predicted key rate of the neural networks for the \( i \)-th sample, respectively.  

Appendix C: Detailed data on the time consumption of neural networks and the numerical method  

Table I shows relative deviations between key rates predicted by our neural network and the corresponding key rates obtained by the numerical method for the given optimal light intensity at different distances and different excess noises. This table is a supplement to Fig. 4.  

Table II shows the specific data of the time consumption of the neural network and the numerical method with excess noise \( \xi \) of 0.008, 0.010 and 0.012. In the numerical method, each point with the excess noise \( \xi \) around 0.01 takes 200 seconds on average, which greatly affects the efficiency of QKD system to calculate the secure key rate. In contrast, the neural network can calculate tens of thousands of key rates in one second. Considering that it takes a certain amount of time for QKD system to collect data, the speed of predicting the key rates by the neural networks completely meet practical applications.
[49] C. Simon, Towards a global quantum network, Nat. Photon. 11, 678 (2017).

[50] R. Bedington, J. M. Arrazola, and A. Ling, Progress in satellite quantum key distribution, npj Quantum Inf. 3, 30 (2017).

[51] S.-K. Liao, W.-Q. Cai, W.-Y. Liu, L. Zhang, Y. Li, J.-G. Ren, J. Yin, Q. Shen, Y. Cao, Z.-P. Li, et al., Satellite-to-ground quantum key distribution, Nature 549, 43 (2017).

[52] H.-Y. Liu, X.-H. Tian, C. Gu, P. Fan, X. Ni, R. Yang, J.-N. Zhang, M. Hu, J. Guo, X. Cao, et al., Optical-relayed entanglement distribution using drones as mobile nodes, Phys. Rev. Lett. 126, 020503 (2021).

[53] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems (2015), software available from tensorflow.org.

[54] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning (MIT Press, Cambridge, Mass., 2016).

[55] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[56] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res 15, 1929 (2014).

[57] F.-Y. Lu, Z.-Q. Yin, C. Wang, C.-H. Cui, J. Teng, S. Wang, W. Chen, W. Huang, B.-J. Xu, G.-C. Guo, et al., Parameter optimization and real-time calibration of a measurement-device-independent quantum key distribution network based on a back propagation artificial neural network, J. Opt. Soc. Am. B 36, B92 (2019).

[58] W. Wang and H.-K. Lo, Machine learning for optimal parameter prediction in quantum key distribution, Phys. Rev. A 100, 062334 (2019).

[59] W. Liu, P. Huang, J. Peng, J. Fan, and G. Zeng, Integrating machine learning to achieve an automatic parameter prediction for practical continuous-variable quantum key distribution, Phys. Rev. A 97, 022316 (2018).

[60] J.-Y. Liu, H.-J. Ding, C.-M. Zhang, S.-P. Xie, and Q. Wang, Practical phase-modulation stabilization in quantum key distribution via machine learning, Phys. Rev. Applied 12, 014059 (2019).

[61] H.-M. Chin, N. Jain, D. Zibar, U. L. Andersen, and T. Gehring, Machine learning aided carrier recovery in continuous-variable quantum key distribution, npj Quantum Inf. 7, 20 (2021).

[62] I. Devetak and A. Winter, Distillation of secret key and entanglement from quantum states, Proc. R. Soc. A 461, 207 (2005).

[63] F. Grosshans, N. J. Cerf, J. Wenger, R. Tualle-Brouri, and P. Grangier, Virtual entanglement and reconciliation protocols for quantum cryptography with continuous variables, Quantum Inf. Comput. 3, 535 (2003).