Machine Learning for Quantum Key Distribution Network Optimization

Student: Shek Lun Leung

Supervisor: Erik Svanberg

Co-Supervisor: Giulio Foletto, Vaishali Adya

Examiner: Katia Gallo

Abstract

Optimizing parameters is crucial for maximizing the performance of Quantum Key Distribution (QKD) systems, but traditional numerical methods are computationally prohibitive for real-time applications, especially on resource-constrained platforms like drones or single-board computers. This study investigates the efficacy of neural networks (NNs) as a high-speed alternative to Dual Annealing (DA) for determining optimal operational parameters (signal/decoy intensities μ_k, probabilities P_{μ_k}, basis choice P_X) for the finite-key decoy-state BB84 protocol. We first establish performance benchmarks using DA optimization across fiber lengths (0–200 km) and block sizes ($n_X = 10^4$ to 10^9), achieving secure distances up to 180 km, consistent with literature expectations [1]. However, DA proves computationally intensive, requiring approximately 45 minutes per 1,000 points on a multi-core CPU. We then trained a four-layer feed-forward neural network using PyTorch, leveraging GPU acceleration (Apple Silicon MPS backend) to complete 5,000 epochs on a 6,000-point dataset (generated via DA) in roughly 20 minutes. The trained NN demonstrates excellent predictive accuracy, closely matching the optimized parameters and secret key rates (R) across the trained n_X range and generalizing effectively to unseen intermediate block sizes (e.g., $n_X = 5 \times 10^8$), with relative SKR errors remaining within an acceptable $\pm 6\%$ even near transmission limits. Most significantly, NN inference for 100 operating points takes approximately 1 second, representing a substantial 270-fold speedup compared to the estimated 4.5 minutes required by DA. This result validates the potential of NNs to enable rapid, on-the-fly parameter adjustments, making sophisticated QKD optimization practical for dynamic scenarios and low-power devices. Future work could involve refining NN architectures for edge-case accuracy and incorporating more comprehensive device models to further bridge the gap towards robust, high-performance quantum networks.

Contents

I. Introduction

Quantum Key Distribution (QKD) offers a path to fundamentally secure communication, enabling two parties to establish shared secret keys protected by the laws of quantum mechanics [2, 3]. To realize this potential in practical systems, rigorous security analysis incorporating real-world factors like finite data block sizes (n_X) and channel losses is essential. Seminal works, such as Lim et al. (2014), have provided crucial finite-key security bounds for widely used protocols like decoy-state BB84, allowing for accurate calculation of the achievable secure key rate (R) [1]. However, maximizing this key rate necessitates careful optimization of protocol parameters, including signal and decoy state intensities (μ_k) and their respective probabilities (P_{μ_k}), tailored to specific channel conditions and block sizes.

This optimization process presents a significant computational challenge. Traditional numerical methods, such as local search algorithms or global optimizers like Dual Annealing, must iteratively evaluate the complex key rate function across a multi-dimensional parameter space [4]. As demonstrated in preliminary analyses (Section D), this can require substantial computation time, potentially tens of minutes per operating point on standard multi-core CPUs, rendering these methods impractical for dynamic scenarios requiring real-time adaptation. This bottleneck is particularly acute for emerging QKD applications on resource-constrained platforms, such as drones, mobile devices, or satellites involved in ground communication, where rapid parameter adjustments are critical to counteract fluctuating environmental conditions (e.g., atmospheric turbulence, varying distances) with limited computational power [5].

Recent advancements in machine learning, particularly deep learning, have shown promise in addressing computational bottlenecks across various scientific domains, including quantum communication. Relevant prior work includes the application of neural networks for noise filtering in continuous-variable QKD [6], stabilizing system parameters like light intensity [7], and, pertinent to this study, predicting optimal QKD parameters to bypass costly online optimization [8]. These studies underscore the potential for machine learning to enhance the efficiency and adaptability of QKD systems.

Building upon this potential, this study focuses on developing and evaluating a neural network approach specifically designed to accelerate the determination of optimal parameters for finite-key decoy-state BB84. Our objective is to demonstrate that a trained neural network can accurately predict the near-optimal parameter set significantly faster than traditional optimization methods like Dual Annealing. We rigorously assess the trade-off between the computational speed-up offered by the neural network and the accuracy of the resulting predicted key rates, particularly evaluating its performance across a wide range of block sizes (n_X) and transmission distances, including its generalization capability to unseen conditions. The ultimate aim is to validate the feasibility of using neural networks for real-time, on-the-fly parameter tuning in practical, potentially resource-constrained, QKD implementations.

II. Background on QKD

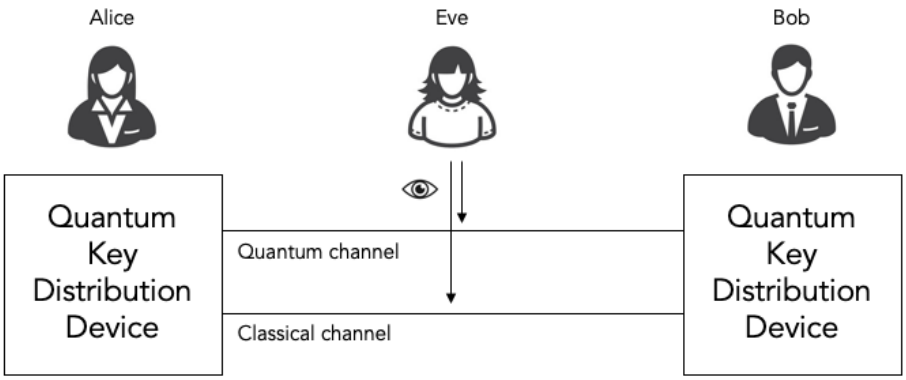

Figure 1: Illustration of a Quantum Key Distribution (QKD) system. Alice and Bob communicate securely using QKD devices over a quantum and classical channel. The quantum channel is used to transmit qubits, while the classical channel is employed for public communication. An eavesdropper, Eve, may attempt to intercept the quantum transmission, but any such attempt introduces detectable disturbances, ensuring security. Extracted from [9].

In 1984, Bennett and Brassard proposed the BB84 protocol for secure quantum key distribution (QKD) between Alice and Bob in an insecure environment [2]. Since then, the scheme has been widely studied and advanced both theoretically and practically. However, practical implementations of the BB84 protocol differ from the original proposal. Instead of using ideal single-photon sources, most systems employ weak pulsed laser sources, which can emit pulses containing more than one photon. This vulnerability allows an eavesdropper (Eve) to exploit the photon-number-splitting (PNS) attack, making high-loss channels particularly susceptible [10]. To counter the PNS attack, most BB84 implementations adopt the decoy-state method. In this approach, Alice randomly varies the mean photon number of each laser pulse sent to Bob, making it difficult for Eve to exploit photon-number-dependent attacks. By analyzing their shared data, Alice and Bob can detect photon-number-dependent loss in the channel. This method makes the BB84 protocol secure against PNS attacks, while also improving secret key rates and increasing tolerance to channel losses.

For an asymmetric coding BB84 protocol, the bases $\mathbf{X}$ and $\mathbf{Z}$ are chosen with probabilities P_X and $P_Z = 1 - P_X$, respectively. The secret key is extracted from the events that Alice and Bob both choose on the $\mathbf{X}$ basis. The protocol is based on the transmission of phase-randomized laser pulses and uses two-decoy settings. The intensity is randomly set to be one of the intensity μ_1, μ_2, μ_3 and satisfies the following condition for secure key generation. The intensity levels are subject to optimization, with μ_3 being fixed as a vacuum state. We consider a fiber-based QKD system model. Alice's laser pulses are set to one of the intensity levels, which are optimized within the constraints:

$$\mu_1 > \mu_2 + \mu_3 \tag{1}$$
$$\mu_2 > \mu_3 \geq 0 \tag{2}$$

Bob uses an active measurement setup with two superconducting nanowire single-photon detectors with a detection efficiency of $\eta_{\text{Bob}} = 10\%$, a dark count probability of $P_{\text{dc}} = 2.7 \times 10^{-7}$. The channel transmittance, η_{ch}, is calculated based on the fiber length L (km) and the attenuation coefficient, α.

$$\eta_{ch} = 10^{-\frac{\alpha L}{100}} \tag{3}$$

we assume $\alpha = 0.2$ dB/km. The fiber length is restricted to the range $L \in [0, 200]$.

The system transmittance is calculated with the detector efficiency, η_{Bob}, and channel transmittance, η_{ch}.

$$\eta_{sys} = \eta_{Bob}\eta_{ch} \tag{4}$$

Alice begins by randomly selecting a bit value, which is recorded as y_i. Then, she chooses a basis $a_i \in [\mathbf{X}, \mathbf{Z}]$ with probabilities P_X for the $\mathbf{X}$ basis and $P_Z = 1 - P_X$ for the $\mathbf{Z}$ basis. Additionally, she selects an intensity level $k_i \in \mathcal{K} = [\mu_1, \mu_2, \mu_3]$ with probabilities P_{μ_1}, P_{μ_2}, and $P_{\mu_3} = 1 - P_{\mu_1} - P_{\mu_2}$. Based on these choices, a laser pulse is prepared and sent to Bob via a quantum channel.

Bob selects a basis $b_i \in \{\mathbf{X}, \mathbf{Z}\}$ with probabilities P_X and $P_Z = 1 - P_X$, respectively. A measurement is then performed, and the outcome is recorded as y_i'. The measurement setup typically involves two single-photon detectors, resulting in four possible outcomes: $[0, 1, \emptyset, \perp]$. Here, 0 and 1 represent bit values, while $\emptyset$ denotes a no-detection event and $\perp$ represents a double-detection event. If the outcome is 0, 1, or $\emptyset$, the corresponding value is assigned to y_i'. If the outcome is $\perp$, a random bit value is assigned to y_i'.

During basis reconciliation, Alice and Bob publicly announce their basis and intensity choices over an authentic channel. They identify and verify the following sets for all $k \in \mathcal{K}$:

$$\mathcal{X}_k := \{i : a_i = b_i = \mathbf{X} \wedge k_i = k \wedge y_i' \neq \emptyset\}, \quad |\mathcal{X}_k| \geq n_{X,k} \tag{5}$$
$$\mathcal{Z}_k := \{i : a_i = b_i = \mathbf{Z} \wedge k_i = k \wedge y_i' \neq \emptyset\}, \quad |\mathcal{Z}_k| \geq n_{Z,k} \tag{6}$$

These steps are repeated until these conditions are met. The total number of laser pulses sent by Alice until these conditions are fulfilled is denoted as N.

First, a raw key pair (X_A, X_B) is generated by selecting a random sample of size n_X, where:

$$n_X = \sum_{k \in \mathcal{K}} n_{X,k}, \quad \mathcal{X} = \bigcup_{k \in \mathcal{K}} \mathcal{X}_k \tag{7}$$

Here, n_X is the post-processing block size. In this protocol, all intensity levels are utilized for key generation.

Second, the sets $\mathcal{Z}_k$ are announced, and the corresponding number of bit errors $m_{Z,k}$ is computed.

Third, the vacuum events ($s_{X,0}$), single-photon events ($s_{X,1}$) and phase errors ($c_{X,1}$) are calculated. Vacuum events ($s_{X,0}$)are instances where no photons are detected, corresponding to the intensity level μ_3, which is set close to zero to simulate a vacuum state. Single-photon events ($s_{X,1}$) occur when exactly one photon is emitted, which is crucial for secure key generation as it minimizes the information available to an eavesdropper. Phase errors ($c_{X,1}$) refer to errors that occur in the phase of the single-photon states, which can compromise the security of the key if not properly accounted for.

If the required conditions for secure key generation, such as sufficient sample sizes and error rates, are not met, the protocol is aborted. Otherwise, post-processing proceeds.

Alice and Bob first perform an error-correction step that reveals at most λ_{EC} bits of information. This step is designed to correct for a predetermined error rate to ensure that both parties have identical keys. Following this, they conduct an error-verification step using a two-universal hash function. This step publishes a minimal amount of information to verify that their keys are identical, with a probability ϵ_{hash} that a pair of non-identical keys passes the verification. The information published during this step is typically very small and can be represented as $\log_2 \frac{1}{\epsilon_{\text{hash}}}$. Finally, conditioned on passing the error-verification step, Alice and Bob perform privacy amplification on their keys to extract a secret key pair (S_A, S_B), where $|S_A| = |S_B| = l$ bits. Conditioned on passing the checks in the error-estimation and error-verification steps, an ϵ_{sec}-secret key of length

$$l = s_{X,0} + s_{X,1} - s_{X,1}h(\phi_X) - \lambda_{EC} - 6\log_2\left(\frac{2}{\epsilon_{\text{sec}}}\right) - \log_2\left(\frac{2}{\epsilon_{\text{hash}}}\right) \tag{8}$$

can be extracted, where the binary entropy function $h(x)$ is defined as:

$$h(x) = -x\log_2 x - (1-x)\log_2(1-x). \tag{9}$$

The number of vacuum events in $\mathbf{X}_A$ satisfies:

$$s_{X,0} \geq \tau_0 \frac{\mu_2 n^-_{X,\mu_3} - \mu_3 n^+_{X,\mu_2}}{\mu_2 - \mu_3}, \tag{10}$$

Where the Poisson probability that Alice prepares an n-photon state is:

$$\tau_n = \sum_{k\in\mathcal{K}} \frac{p_k e^{-k}\mu_k^n}{n!}. \tag{11}$$

The maximum and the minimum values of $n^{\pm}_{X,k}$ are:

$$n^{\pm}_{X,k} = \frac{e^k}{p_k}\left[n_{X,k} \pm \sqrt{\frac{n_X}{2}\ln\frac{21}{\epsilon_{\text{sec}}}}\right], \quad n^{\pm}_{Z,k} = \frac{e^k}{p_k}\left[n_{Z,k} \pm \sqrt{\frac{n_Z}{2}\ln\frac{21}{\epsilon_{\text{sec}}}}\right], \quad \forall k \in \mathcal{K}. \tag{12}$$

The number of single-photon events in $\mathbf{X}_A$ is:

$$s_{X,1} \geq \frac{\tau_1\mu_1\left[n^-_{X,\mu_2} - n^+_{X,\mu_3} - \frac{\mu_2^2-\mu_3^2}{\mu_1^2}\left(n^+_{X,\mu_1} - \frac{s_{X,0}}{\tau_0}\right)\right]}{\mu_1(\mu_2-\mu_3) - \mu_2^2 + \mu_3^2}. \tag{13}$$

Similarly, we calculate the number of vacuum events $s_{Z,0}$ and single-photon events $s_{Z,1}$ for $\mathcal{Z} = \bigcup_{k\in\mathcal{K}} \mathcal{Z}_k$ using Eqs. (10) and (13) with statistics from the Z basis. Additionally, the number of bit errors $\nu_{Z,1}$ associated with single-photon events in Z is required:

$$\nu_{Z,1} \leq \tau_1 \frac{m^+_{Z,\mu_2} - m^-_{Z,\mu_3}}{\mu_2 - \mu_3}, \tag{14}$$

Where:

$$m^{\pm}_{Z,k} = \frac{e^k}{p_k}\left[m_{Z,k} \pm \sqrt{\frac{m_Z}{2}\ln\frac{21}{\epsilon_{\text{sec}}}}\right], \quad \forall k \in \mathcal{K}, \tag{15}$$

And:

$$m_Z = \sum_{k\in\mathcal{K}} m_{Z,k}. \tag{16}$$

The phase error rate of single-photon events in $\mathbf{X}_A$ is [11]:

$$\phi_X = \frac{c_{X,1}}{s_{X,1}} \leq \frac{\nu_{Z,1}}{s_{Z,1}} + \gamma\left(\epsilon_{\text{sec}}, \frac{\nu_{Z,1}}{s_{Z,1}}, s_{Z,1}, s_{X,1}\right), \tag{17}$$

Where:

$$\gamma(a,b,c,d) = \sqrt{\frac{(c+d)(1-b)b}{cd\ln 2}\log_2\left(\frac{c+d}{cd(1-b)b}\frac{21^2}{a^2}\right)}. \tag{18}$$

The key rate becomes:

$$R = \frac{l}{N}, \tag{19}$$

where l is the length of the final secret key (in bits) and N is the total number of pulses sent. For more detailed derivation, please read the Appendix.

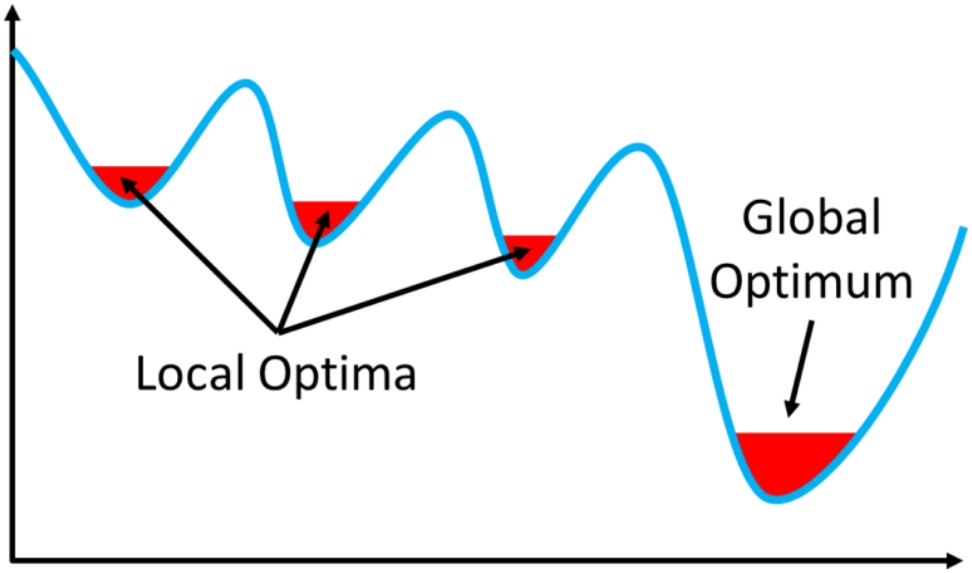

Figure 2: Local and global minima in optimization, showing how local search can converge prematurely. The given protocol has a non-convex key rate versus parameters function, such that the optimization problem is a convex optimization and a local search works. Extracted from [12].

III. Background on Optimization

Optimization in quantum key distribution (QKD) aims to maximize the secret key rate $R = l/N$ by tuning parameters like pulse intensity (μ_k) and its selection probability (P_{μ_k}). Local search algorithms (e.g., gradient descent) refine initial guesses incrementally, excelling in convex landscapes but struggling with non-convexity [1] (Fig. 2). Global search methods (e.g., simulated annealing) explore the full parameter space, addressing this at a higher cost. Table 1 compares their traits.

Feature	**Local Search**	**Global Search**
Search Scope	Local region	Entire parameter space
Efficiency	Fast, efficient	Slow, resource-intensive
Risk of Local Minima	High	Low
Applicability	Convex functions	Non-convex landscapes

Table 1: Local vs. Global Search Algorithms

Dual Annealing extends simulated annealing by pairing probabilistic exploration with local refinement. It uses a temperature schedule $T(t)$ to accept suboptimal solutions early, avoiding local minima, then refines via Nelder-Mead as T decreases. Nelder-Mead, a derivative-free method, minimizes functions [13], suiting QKD optimization.

To address QKD's non-convex landscape, we employ Dual Annealing to globally search parameters (e.g., μ_k, P_{μ_k}), identifying regions maximizing R, followed by Nelder-Mead for precision. This hybrid approach outperforms standalone simulated annealing by combining global exploration with local refinement to achieve higher secret key rates (R) with improved efficiency (faster convergence).

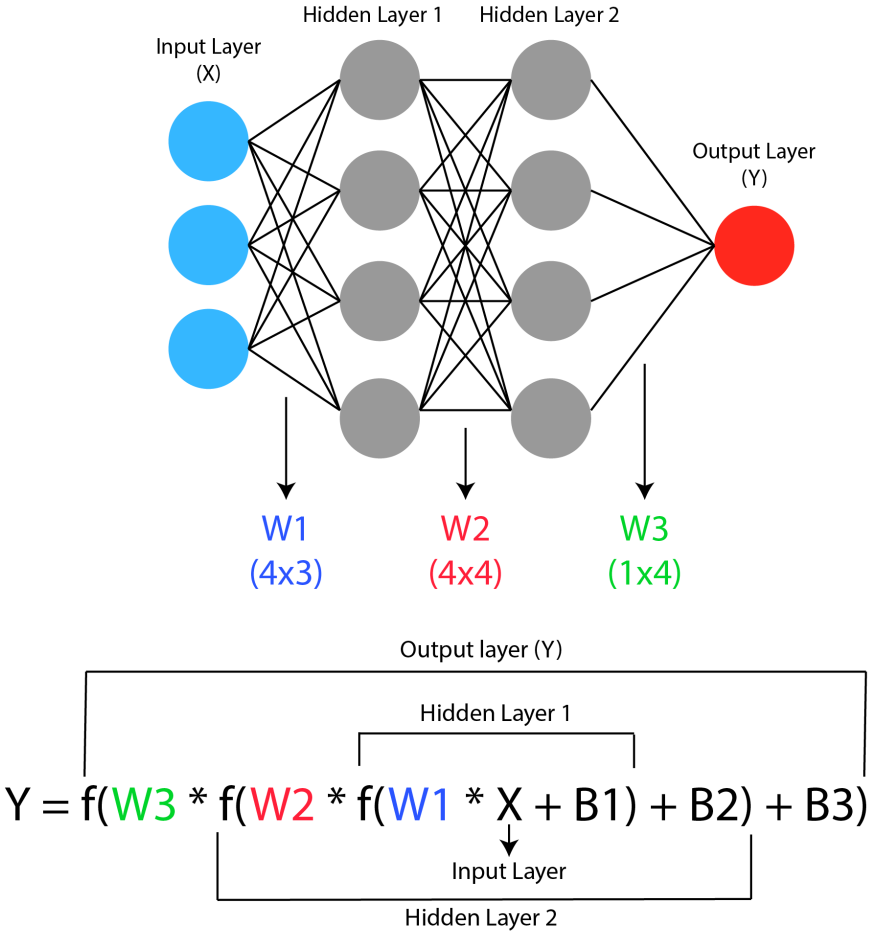

Figure 3: Artificial neural network with weighted inputs and activation function. Extracted from [14].

IV. Background on Neural Networks

Neural networks (NNs), inspired by biological neurons, are powerful tools for optimizing complex problems, such as tuning quantum key distribution (QKD) parameters (e.g., intensities μ_k, probabilities P_{μ_k}) to maximize the secret key rate $R = \frac{l}{N}$. An NN consists of interconnected units called neurons, organized into layers: an input layer that receives features, hidden layers that perform computations, and an output layer that produces predictions. Each neuron computes an output as $\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i w_i\right)$, where x_i are inputs (e.g., QKD parameters like L, P_{dc}), w_i are weights, w_0 is a bias, and $g(\cdot)$ is a non-linear activation function, such as ReLU ($f(x) = \max(0, x)$) [15]. Data flows through the layers via weighted connections, with biases added at each step to adjust the computations. For example, Figure 3 illustrates a feedforward neural network with an input layer (X) of 3 nodes, two hidden layers (Hidden Layer 1 and Hidden Layer 2) with 4 nodes each, and an output layer (Y) with 1 node. The network transforms inputs through weight matrices W1 (4×3), W2 (4×4), and W3 (1×4), with bias terms B1, B2, and B3 applied at each layer. An activation function f is applied at each step. This architecture can model the complex optimization landscape of QKD, where numerical optimization maximizes the secret key rate R over parameters like μ_k and P_{μ_k}, enabling the NN to predict optimal settings efficiently from valid optimization data. In this project, the NN is trained on a dataset of valid numerical values of the optimized parameters and secret key rate from dual annealing, to predict their corresponding optimized values instantly for new experimental parameters, significantly accelerating QKD optimization.

[1] A non-convex function has multiple local minima (as shown in the figure with "Local Minima" and "Global Optimum") This means the optimization landscape is complex, with several "valleys" (minima) where a local search might get stuck, potentially missing the global optimum.

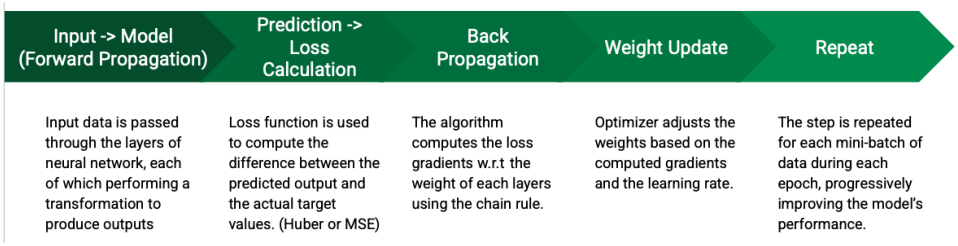

Figure 4: Forward and backward propagation in NN training.

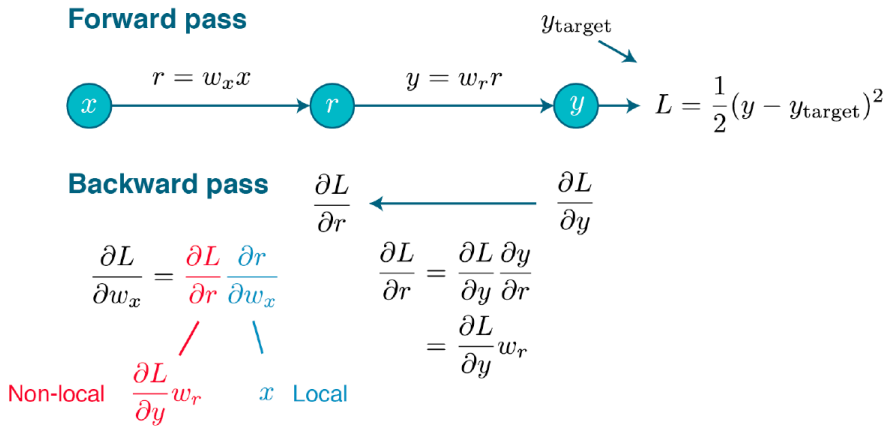

Figure 5: The details calculation of forward pass and backward pass. Extracted from [16].

A. Training Process and Propagation

The training of a neural network involves forward propagation and backward propagation, as illustrated in Figures 4 and 5. During forward propagation, input data is passed through the network, with each neuron computing the weighted sum of its inputs followed by an activation function to produce outputs. The output from each layer serves as input to the next layer until the final output is obtained. This predicted output is compared to the actual target values using a loss function, such as Mean Squared Error (MSE), which quantifies the difference between predicted and actual values. In backward propagation, the loss is propagated back through the network, and gradients of the loss function concerning each weight are computed using the chain rule of differentiation. An optimization algorithm, such as Adam, then updates the weights and biases based on the computed gradients and the learning rate. An epoch is one complete pass of the training dataset through the neural network. This process repeats over multiple epochs, gradually minimizing the error and improving the network's accuracy.

B. Application to QKD

In this work, a fully connected NN with three hidden layers (16, 32, 16 neurons, ReLU) takes inputs like distance (L), dark count (P_{dc}), misalignment (e_d), and signals (n_X), outputting five decoy-state parameters. Trained with Adam, it adapts to QKD's dynamic conditions, outperforming traditional optimization by rapidly predicting near-optimal R values in simulations More details on the method are explained in the next section.

V. Method

A. Simulation of Key Rate

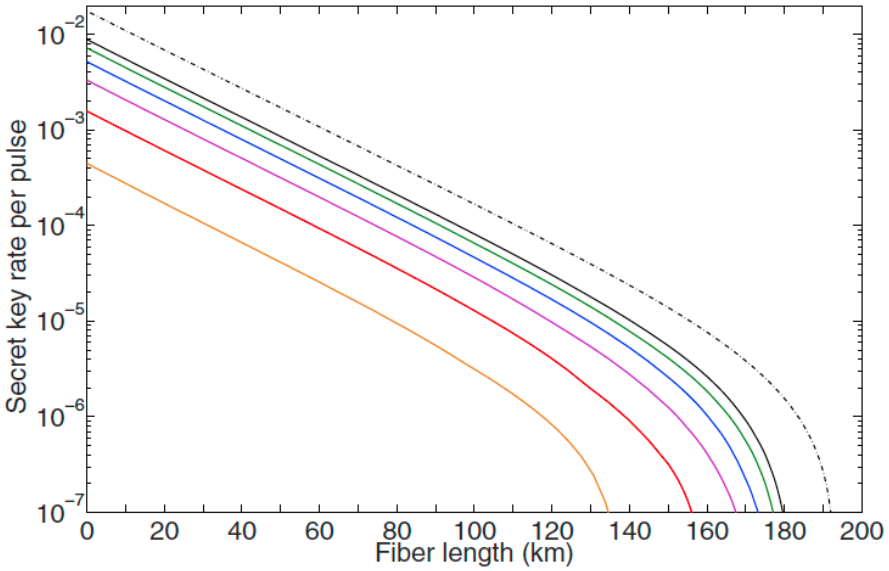

Figure 6: Secret key rate vs. fiber length for $n_X = 10^4$ to 10^9, from [1]. The secret key rate (color solid lines) is numerically optimized (in logarithmic scale) for fixed postprocessing block size $n_X = 10^s$ with $s = 4, 5, \ldots, 9$ (from left to right). The asymptotic secret key rate (dashed line) corresponds to the asymptotic secret key rate, i.e., in the limit of infinitely long keys. The number of laser pulses sent by Alice can be approximated with the secret key rate and the block size, i.e., $N \leq \frac{n_X}{R}$. The image is extracted from [1].

The QKD key rate was calculated using the BB84 decoy state protocol to establish a performance baseline before optimization, verifying whether the key rate's decay with fiber length matches the trend in [1]. This validation step computes the secret key rate $R = \frac{l}{N}$ (Eq. 19 from Section "Background on QKD") across fiber lengths $L = 0$ to 200 km with 1000 discrete points and detected events $n_X = 10^4$ to 10^9, spanning six logarithmic scales, representing short-range laboratory tests to long-distance operational networks. The number of laser pulses sent by Alice, N, is estimated as $N = \frac{n_X}{P_X^2}$, where P_X is the probability of a pulse being detected in the X basis, accounting for basis choice, channel loss, and detector efficiency. Implemented in JAX (Python 3.9.20), the simulation leverages efficient array operations and potential automatic differentiation, enabling rapid computation of R under realistic conditions. The parameter n_X represents detected events in the X basis, critical for estimating error rates and single-photon yields, a cornerstone of decoy-state security analysis. Key parameters were sourced from [1], with security ensured by secrecy error $\epsilon_{\text{sec}} = 10^{-10}$ and correction error $\epsilon_{\text{hash}} = 10^{-15}$, adhering to sufficiently secure QKD standards. Detailed specifications are provided in Appendix A.

Core calculations were encapsulated in a modular script, `QKD_functions.py`, enhancing code reusability and maintainability. Detection rates were modeled as:

$$D_k = 1 - (1 - 2P_{\text{dc}})e^{-\eta_{\text{sys}}\mu_k},$$

using fixed intensities $\mu_k = [0.6, 0.2, 2 \times 10^{-4}]$ and probabilities $P_{\mu_k} = [0.05, 0.6, 0.35]$. These intensities were selected to balance signal strength for key generation with a low value to detect multi-photon eavesdropping attempts, a strategy validated in decoy-state literature. Basis probabilities were set at $P_X = 0.5$ and $P_Z = 0.5$, providing an initial symmetric configuration for key extraction and error checking. Error correction was calculated via:

$$\lambda_{\text{EC}} = n_X f_{\text{EC}} h(e_{\text{obs}}),$$

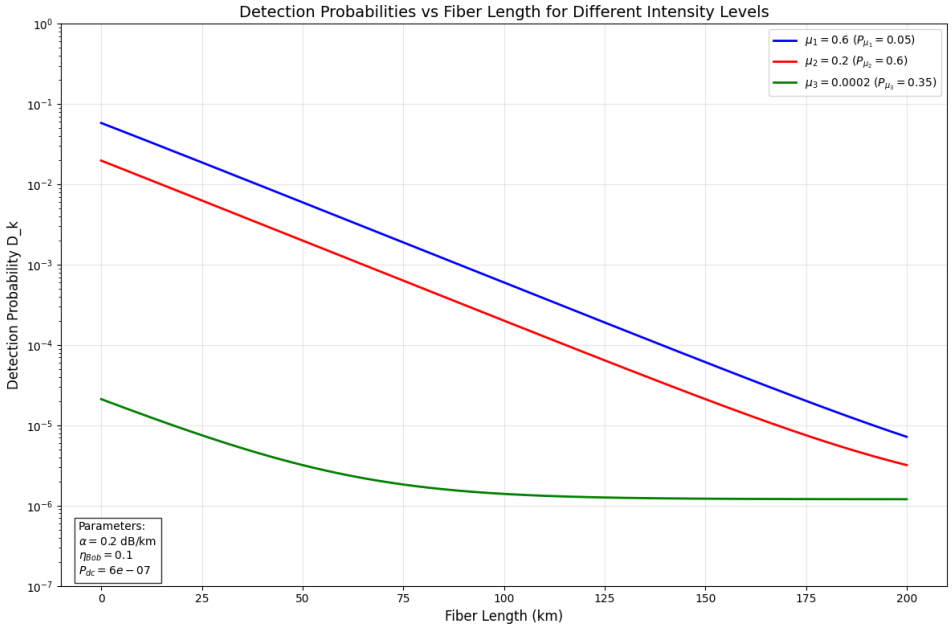

Figure 7: Detection probabilities vs. fiber length for different intensity levels in the BB84 decoy-state QKD protocol, as simulated for fiber lengths $L = 0$ to 200 km. The curves represent detection rates $D_k = 1-(1-2P_{dc})e^{-\eta_{sys}\mu_k}$ for fixed intensities $\mu_k = [0.6, 0.2, 2\times10^{-4}]$ with corresponding probabilities $P_{\mu_k} = [0.05, 0.6, 0.35]$. These intensities balance key generation efficiency with multi-photon eavesdropping detection, aligning with the simulation parameters from [1] and supporting the key rate analysis in Fig. 6.

where $f_{\text{EC}} = 1.16$ reflects typical reconciliation inefficiency, and $e_{\text{obs}} = \frac{m_x}{n_X}$ is the observed error rate, with $h(e_{\text{obs}})$ as the binary entropy function measuring information loss. This term quantifies bits sacrificed to correct errors, directly impacting the net key length l.

The variables to be optimized are a set of 5 parameters, $[\mu_1, \mu_2, P_{\mu_1}, P_{\mu_2}, P_X]$, where μ_1, μ_2 are the signal and decoy intensities, and P_{μ_1}, P_{μ_2}, P_X are the probabilities of sending them and the basis choice probability. We unite these 5 parameters into one parameter vector $\vec{p}$.

The photon intensities μ_1, μ_2 (with $\mu_3 = 2\times10^{-4}$ fixed to simulate the effect of sending an imperfect vacuum state) and probabilities $P_{\mu_1}, P_{\mu_2}, P_{\mu_3}$ (with $P_{\mu_3} = 1-P_{\mu_1}-P_{\mu_2}$), are constrained within bounds $\mu_k \in [0,1]$ and $P_{\mu_k} \in [0,1]$ to maintain realistic photon emissions and probability sums. The basis probability P_X is optimized, with $P_Z = 1-P_X$, adjusting the trade-off between key extraction in the X basis and error estimation in the Z basis. Fixed parameters include $\alpha = 0.2$ dB/km (fiber attenuation), $\eta_{Bob} = 0.1$ (detector efficiency), $P_{dc} = 6\times10^{-7}$ (dark count probability), and $f_{EC} = 1.16$ (error correction efficiency), reflecting typical experimental conditions validated in prior QKD studies.

The calculation of the secret key rate depends not only on the optimizable parameters $\vec{p}$ but also on the experimental parameters $\vec{e}$ that represent the physical conditions of the QKD system. These experimental parameters are transformed into a normalized form to enhance optimization stability and efficiency:

$$e_1 = \frac{L}{100}, \tag{20}$$

$$e_2 = -\log_{10} P_{dc}, \tag{21}$$

$$e_3 = e_d \times 100, \tag{22}$$

$$e_4 = \log_{10} n_X, \tag{23}$$

where L is the fiber length (km), P_{dc} is the dark count probability, e_d is the misalignment error rate, and n_X is the total number of pulses received in the X basis. These transformations normalize the magnitudes—e.g., scaling L from 0–200 km to 0.1–2 reduces its numerical dominance in the optimization

process.

$$\text{Input} : \vec{e} = [e_1, e_2, e_3, e_4] = [L, P_{dc}, e_d, n_X] \tag{24}$$
$$\text{Output} : \vec{p} = [p_1, p_2, p_3, p_4, p_5] = [\mu_1, \mu_2, P_{\mu_1}, P_{\mu_2}, P_X] \tag{25}$$

This preprocessing facilitates the mapping from experimental conditions $\vec{e}$ to optimal protocol parameters $\vec{p}$, which can be conceptualized as:

$$\vec{e} \to \vec{p} \tag{26}$$

Therefore, the QKD secret key rate can be expressed as a function of both sets of parameters:

$$\mathrm{R} = \mathrm{R}(\vec{e}, \vec{p}) \tag{27}$$

Where R is the objective function dependent on the experimental parameters $\vec{e}$, which cannot be controlled by the users, and the user parameters $\vec{p}$, which can be adjusted to optimize performance.

This calculates the rate for a given fixed set of parameters and experimental parameters. To calculate the optimal rate, we need to calculate

$$\mathrm{R}_{\max}(\vec{e}) = \max_{\vec{p} \in P} \mathrm{R}(\vec{e}, \vec{p}) \tag{28}$$

Where P represents the feasible parameter space constrained by physical limitations. By maximizing R, we also determine the set of optimal parameters $\vec{p}_{\text{opt}}$. Note that $\vec{p}_{\text{opt}}$ is a function of $\vec{e}$ only, and the key objective in QKD optimization is to find these optimal parameters based on the given experimental conditions:

$$\vec{p}_{\text{opt}} = \arg\max_{\vec{p} \in P} \mathrm{R}(\vec{e}, \vec{p}) \tag{29}$$

B. Optimization

The optimization framework maximizes the secret key rate (R) and determines optimal parameters ($\mu_1, \mu_2, P_{\mu_1}, P_{\mu_2}, P_X$) for the decoy-state BB84 protocol using a Jupyter Notebook. Implemented in JAX, it defines system parameters, formulates an objective function, and employs a three-phase optimization strategy combining global and local search methods.

The objective function, implemented via JAX's `objective` function, maximizes $R = \frac{l}{N}$, accounting for channel losses (dependent on fiber length L), detection rates (influenced by detection efficiency η_{Bob}), and errors (e.g., dark counts, misalignment). JAX's automatic differentiation provides gradient insights into parameter impacts on R, while Just-In-Time (JIT) compilation optimizes repeated evaluations for efficiency.

The optimization process proceeds in three phases. In the first phase, dual annealing's simulated annealing performs a global search within parameter bounds (e.g., $\mu_k \in [0, 1]$, $P_{\mu_k} \in [0, 1]$), using a cooling schedule to explore broadly at high temperatures and escape local optima. Initial guesses, listed in Table 2 for block sizes $n_X = 10^4$ to 10^9, are derived from prior optimization runs, ensuring near-optimal starting points by using parameters from the previous fiber length iteratively. In the second phase, dual annealing's embedded local search, using the L-BFGS-B method, refines solutions as the temperature decreases, targeting high-R regions. In the third phase, the Nelder-Mead algorithm performs a separate local optimization, initializing from the dual annealing result. This gradient-free method uses simplex transformations (e.g., reflection, expansion, contraction) to converge precisely to the optimal parameters.

Block Size n_X	μ_1	μ_2	P_{μ_1}	P_{μ_2}	P_X
10^4	0.65	0.15	0.05	0.61	0.425
10^5	0.62	0.24	0.10	0.70	0.55
10^6	0.68	0.30	0.14	0.74	0.66
10^7	0.55	0.34	0.15	0.75	0.75
10^8	0.54	0.375	0.16	0.775	0.83
10^9	0.52	0.40	0.18	0.785	0.88

Table 2: Initial guess values for different block sizes n_X.

Six datasets, each containing 1,000 data points, are generated for block sizes $n_X = 10^4, 10^5, 10^6, 10^7, 10^8, 10^9$, with each point including optimized parameters and R for fiber lengths (0–200 km). An unseen dataset of 100 points for $n_X = 5 \times 10^8$, not included in training, tests neural network generalization across new block sizes. The `joblib` library parallelizes objective evaluations across four cores, reducing runtime by distributing the workload. JAX's JIT compilation optimizes the `optimal_parameters` and `objective` functions, pre-compiling them into efficient machine code to minimize overhead during iterative calls, ensuring rapid convergence to the optimal parameter set stored in `optimized_params`.

C. Neural Network

A neural network (NN) was developed to predict optimal parameters for the decoy-state BB84 quantum key distribution (QKD) protocol, offering a data-driven alternative to traditional optimization techniques. Implemented in PyTorch within a Jupyter Notebook, the NN harnesses the Metal Performance Shaders (MPS) backend on an M2 Pro device to accelerate computations, leveraging GPU capabilities for efficient training. The architecture features an input layer with 4 neurons, accepting normalized experimental inputs $(L, P_{\text{dc}}, e_d, n_X)$, representing fiber length, dark count probability, misalignment error rate, and total pulse count, respectively. Three fully connected hidden layers, with 16, 32, and 16 neurons, respectively, use the Rectified Linear Unit (ReLU) activation function to introduce non-linearity, enabling the model to learn intricate patterns in the data. The output layer, with 5 neurons, predicts the optimal parameters $\mu_1, \mu_2, P_{\mu_1}, P_{\mu_2}, P_X$, which govern the signal and decoy intensities and basis probabilities critical to maximizing the secret key rate R.

Figure 8 illustrates the neural network architecture designed to optimize QKD parameters. The network takes four input features: the distance between Alice and Bob (L), the dark count probability (P_{dc}), the basis misalignment (e_d), and the number of signals received on basis X (n_X). These inputs are processed through three dense hidden layers with 16, 32, and 16 neurons, respectively, using ReLU activation functions. The final output layer, employing a linear activation function, predicts five parameters critical to QKD: the signal and decoy state intensities (μ_1, μ_2), the probabilities of sending these states (P_{μ_1}, P_{μ_2}), and the probability of using the X basis (P_X).

The methodology for training and testing the neural network (NN) to optimize QKD parameters is illustrated in Figure 9. The process begins with a generator program that produces a dataset (detailed below) of experimental parameters $\vec{e}_{\text{popt}}$, each paired with optimal parameters $\vec{p}_{\text{opt}}$, derived through dual annealing optimization. The dataset is used to train the neural network via the NN trainer program, which predicts optimal parameters $\vec{p}_{\text{pred}}$ from new experimental data $\vec{e}_{\text{test}}$, representing unseen scenarios. A validation program computes the secret key rates $R(\vec{e}, \vec{p}_{\text{opt}})$ and $R(\vec{e}, \vec{p}_{\text{pred}})$, comparing the performance of the predicted parameters against optimized benchmarks to assess the neural network's predictive accuracy.

The training dataset was constructed from the optimization runs referenced in the previous section, covering block sizes from 10^4 to 10^9. Each data point pairs the normalized experimental parameters $\vec{e}_{\text{popt}}$ (defined earlier) with the corresponding optimized parameters $\vec{p}_{\text{opt}}$. The experimental parameters were normalized for all datasets (training, validation, and test), and the optimized parameters were scaled

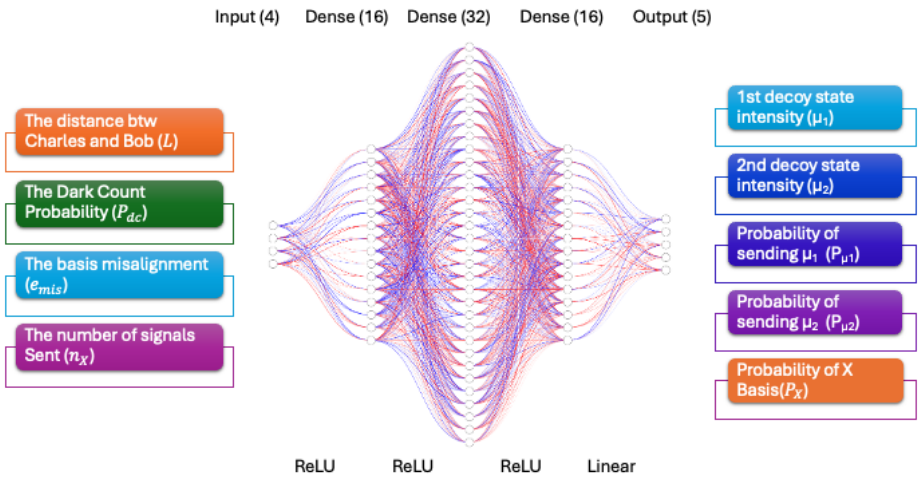

Figure 8: Neural network architecture for optimizing QKD parameters. The network maps four input features to five output parameters through three hidden layers with ReLU activation and a linear output layer.

to [0, 1], preserving their relative magnitudes and facilitating stable training. The dataset, totaling 6,000 points (1,000 per block size), was filtered to include only scenarios with positive key rates and fiber lengths up to 200 km, as the BB84 protocol with the chosen parameters (Table 3) yields zero key rates beyond this distance, reflecting practical QKD constraints. It was divided into an 80% training set (4,800 points) and a 20% validation set (1,200 points), with random shuffling to eliminate bias and ensure representative sampling across conditions.

Training employed the Mean Squared Error (MSE) loss function to minimize the difference between predicted and target parameters, computing both training loss [2] and validation loss [3] during optimization. The Adam optimizer, starting with a learning rate [4] of 10^{-3}, iteratively adjusts the model's weights over 5,000 epochs, adapting to the data's complexity. A learning rate scheduler reduces the rate by a factor of 0.1 if validation loss stalls for 5 epochs, preventing overfitting and promoting convergence. Batches of 64 samples were processed using a data loader, leveraging the MPS device's parallel processing to expedite training. The model refines its predictions of $\mu_1, \mu_2, P_{\mu_1}, P_{\mu_2}, P_X$ through backpropagation, iteratively minimizing the training loss and capturing the relationship between experimental conditions and optimal settings.

For final testing, a test dataset with a block size of $n_X = 5 \times 10^8$, not included in the training or validation datasets, was generated, comprising 100 points across varying fiber lengths (0–200 km). Test inputs $\vec{e}_{\text{test}}$ were processed using the same scaling applied during training, ensuring consistency. The trained model predicted parameters $\vec{p}_{\text{pred}}$, which were inverse-transformed to their original scales and used to compute key rates via the analytical QKD model. These predicted rates were compared to optimized benchmarks, with results visualized as plots of key rates and parameters across fiber lengths for the first and last training epochs. Loss curves for the training and validation phases were also plotted, demonstrating convergence and validating the model's performance. This approach efficiently predicts optimal QKD parameters, leveraging extensive data and normalization to generalize across block sizes, offering a robust tool for practical QKD applications.

[2] Training loss is the error on the training dataset, indicating how well the model fits the training data.
[3] Validation loss is the error on a separate validation dataset, assessing generalization to unseen data.
[4] The learning rate controls the step size of parameter updates during training, influencing the speed and stability of convergence.

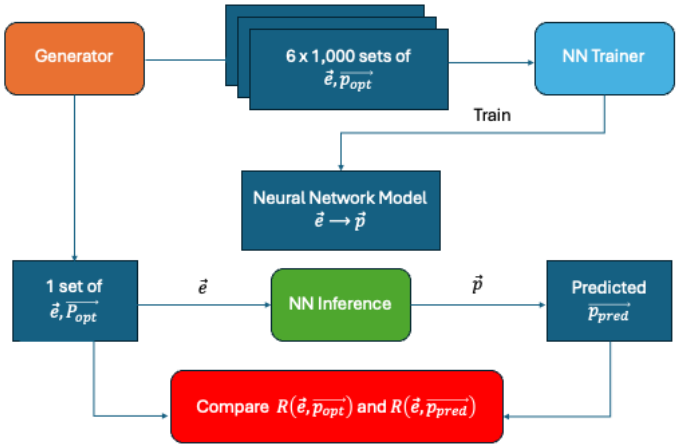

Figure 9: Data flow for training and testing the NN in optimizing QKD parameters, with programs and data represented by rounded and rectangular boxes, respectively.

VI. Results

A. Verification of BB84 Key Rate Script Implementation

Before proceeding to parameter optimization, the core implementation of the decoy-state BB84 secret key rate (R) calculation was verified. This involved simulating the key rate using a set of fixed, plausible parameters (detailed in Table 3) across fiber lengths $L \in [0, 200]$ km and for various block sizes $n_X = 10^s$, with $s = 4, 5, \ldots, 9$. The simulation employs three intensity levels (one signal, two decoys) as specified. The goal of this step was to ensure the script produces physically reasonable results and trends consistent with established literature, using [1] as a reference point for expected optimized performance.

Experimental Parameters $\vec{e}$	**Fixed Protocol Parameters $\vec{p}$**
$\alpha = 0.2$ dB/km	$\mu_1 = 0.6$
$\eta_{\text{Bob}} = 0.1$	$\mu_2 = 0.2$
$P_{\text{dc}} = 6 \times 10^{-7}$	$\mu_3 = 2 \times 10^{-4}$
$e_{\text{mis}} = 5 \times 10^{-3}$	$P_{\mu_1} = 0.05$
$f_{\text{EC}} = 1.16$	$P_{\mu_2} = 0.6$
$\epsilon_{\text{sec}} = 10^{-10}$	$P_{\mu_3} = 1 - P_{\mu_1} - P_{\mu_2} = 0.35$
$\epsilon_{\text{cor}} = 10^{-15}$	$P_X = 0.5$
$n = 1$ (pulses per event)	$P_Z = 1 - P_X = 0.5$

Table 3: Fixed experimental and protocol parameters used for simulation verification (Figure 10). Note that μ_k and P_{μ_k} values are representative examples and not necessarily optimal.

The results of this simulation using fixed parameters are shown in Figure 10. The key rate R exhibits the expected exponential decay with increasing fiber length L, consistent with the Beer-Lambert law governing channel loss ($\eta \propto e^{-\alpha L}$). The slope on the logarithmic scale aligns with the specified attenuation coefficient $\alpha = 0.2\,\text{dB/km}$. Furthermore, the plot correctly captures the dependency on block size n_X: larger n_X values yield higher key rates and extend the maximum achievable distance due to reduced finite-size effects. A quantitative comparison with the optimized results reported by Lim et al.

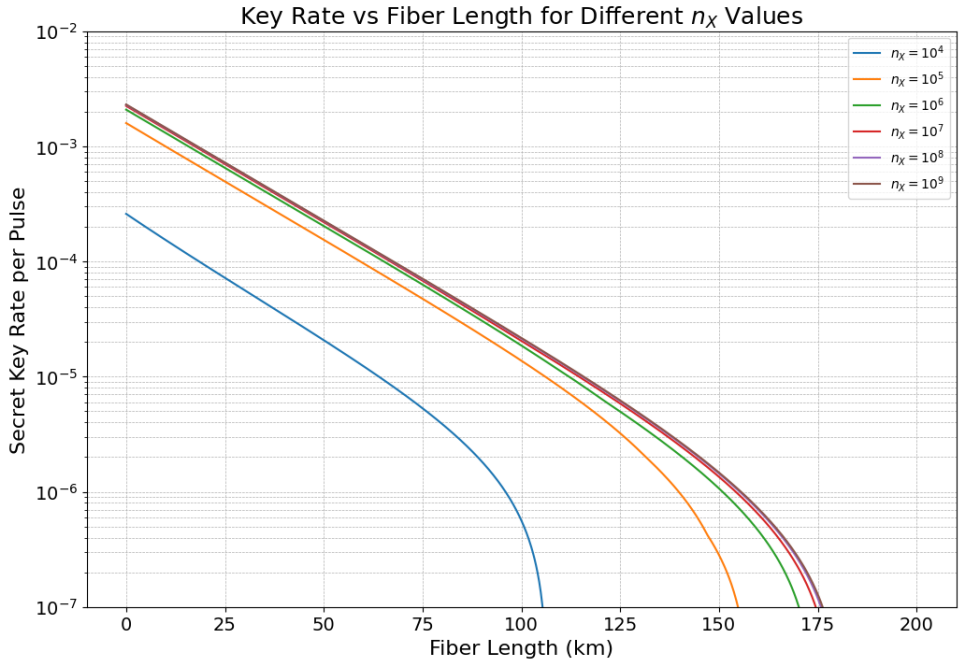

Figure 10: Secret key rate (R, logarithmic scale) vs. fiber length simulated using the fixed parameters from Table 3 for block sizes $n_X = 10^4$ to 10^9.

[1] (shown previously in Figure 6) provides context. At 100 km, our simulation with fixed parameters yields R values from $\sim 10^{-5.5}$ ($n_X = 10^4$) to $\sim 10^{-4.4}$ ($n_X = 10^9$). While these are lower than the optimized rates shown in Figure 6 (which are closer to $\sim 10^{-5}$ to $\sim 10^{-4}$), the magnitudes are reasonably close, and the relative improvement with n_X is similar. The maximum secure distances obtained with these fixed parameters range from approximately 105 km ($n_X = 10^4$) to 175 km ($n_X \geq 10^7$), again showing the correct trend but generally falling short of the fully optimized distances seen in Figure 6 (130–170+ km). This discrepancy in absolute performance is expected, as the parameters in Table 3 were chosen as representative values and were not optimized for each distance and n_X. However, the qualitative agreement in the decay shape, the correct scaling with n_X, and the reasonable proximity of the rates and distances to literature benchmarks confirm the fundamental correctness of the underlying simulation script. This verification provides confidence before applying the Dual Annealing optimization to find the truly optimal parameters presented in subsequent sections.

n_X	100 km			150 km		
	Key Rate (R)	$\log_{10}(R)$	Gain (%) vs. Prev. n_X	Key Rate (R)	$\log_{10}(R)$	Gain (%) vs. Prev. n_X
10^4	5.53×10^{-7}	-6.26	N/A	< 0	N/A	N/A
10^5	1.37×10^{-5}	-4.86	2372.3	2.85×10^{-7}	-6.55	N/A
10^6	1.85×10^{-5}	-4.73	35.5	1.07×10^{-6}	-5.97	276.9
10^7	2.04×10^{-5}	-4.69	10.0	1.35×10^{-6}	-5.87	26.0
10^8	2.13×10^{-5}	-4.67	4.5	1.44×10^{-6}	-5.84	6.8
10^9	2.16×10^{-5}	-4.67	1.4	1.47×10^{-6}	-5.83	2.0

Table 4: Optimized secret key rates (R) and percentage gains achieved by increasing block size (n_X) at selected fiber lengths.

Table ?? quantifies the impact of increasing the block size n_X on the simulated secret key rate (R) at representative fiber lengths of 100 km and 150 km. As expected from finite-key security analyses, larger block sizes significantly enhance performance by reducing statistical fluctuations and tightening security bounds. The most dramatic improvement occurs when moving from very small block sizes;

for example, at 100 km, increasing n_X from 10^4 to 10^5 boosts the key rate by over 2300%. However, the table clearly illustrates the principle of diminishing returns. The percentage gain achieved from each subsequent order-of-magnitude increase in n_X progressively shrinks. At 100 km, the gain drops to 35.5% ($10^5 \rightarrow 10^6$), then 10.0% ($10^6 \rightarrow 10^7$), and finally to just 1.4% ($10^8 \rightarrow 10^9$). A similar trend is observed at 150 km, where substantial initial gains (e.g., 276.9% from $10^5 \rightarrow 10^6$) decrease to only 2.0% ($10^8 \rightarrow 10^9$). This occurs because the dominant statistical uncertainties scale roughly logarithmically with n_X; once n_X reaches 10^7 or 10^8, the statistical precision is already high, and further increases yield only marginal reductions in security penalties [1]. This stabilization is visually represented in Figure 13, where the optimized key rate curves for $n_X = 10^7, 10^8$, and 10^9 become closely clustered. While larger n_X mitigates statistical limitations, the ultimate reach is constrained by fundamental noise sources. The stabilization of the maximum secure distance around 175-180 km for $n_X \geq 10^7$ (seen in Figure 13) indicates that beyond this point, noise factors like detector dark counts ($P_{dc} = 6 \times 10^{-7}$) and channel-induced errors amplified by loss dominate over any remaining statistical uncertainty, preventing further extension even with larger block sizes. It is also worth contrasting these optimized results with the baseline performance obtained during the script verification using fixed parameters (Table 3, Figure 10). Specific choices in that simulation, such as the high proportion of decoy states ($P_{\mu_2} = 0.6$) and balanced basis choice ($P_X = 0.5$), contribute to its characteristic performance curve, which, while validating the script's physics, yields significantly lower rates and shorter distances than achieved through optimization. Furthermore, idealized assumptions common to many simulations, such as neglecting afterpulsing ($P_{ap} = 0$), may lead to somewhat optimistic rate estimations compared to real-world experimental capabilities. Accurately modeling practical QKD systems requires considering both sophisticated parameter optimization and realistic device characterization.

B. Optimization

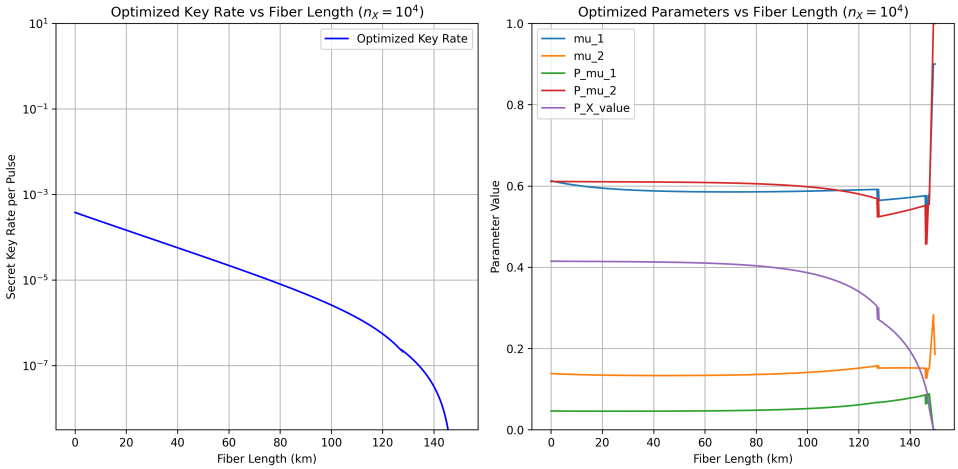

Figure 11: Left: Optimized key rate R (bits per pulse) versus fiber length L (km) for $n_X = 10^4$ pulses, showing exponential decay with a maximum secure distance of 145 km. Right: Optimized parameters (μ_1, μ_2, μ_3, P_{μ_1}, P_{μ_2}, P_X) versus fiber length, highlighting adaptive adjustments by the dual annealing algorithm, with notable fluctuations beyond 125 km due to extreme attenuation.

The results of the secret key rate (SKR) optimization for a relatively small block size of $n_X = 10^4$ pulses are detailed in Figure 11. The figure illustrates the achievable key rate and the corresponding optimal parameter settings as a function of fiber transmission distance.

The left panel shows the optimized SKR on a logarithmic scale. Starting from approximately $10^{-3.5}$ bits per pulse at zero distance, the key rate decreases steadily due to channel loss ($\alpha = 0.2$ dB/km). Initially, up to about 100 km, the decay appears roughly linear on this log plot, characteristic of exponential loss dominating. Beyond 100-120 km, the decline sharpens noticeably as the signal weakens relative to detector noise (P_{dc}) and finite-size effects become more pronounced, limiting the precision of parameter estimation and achievable security bounds. The secure transmission effectively ceases around 145 km, where the SKR drops below the practical threshold (e.g., 10^{-8}).

The right panel reveals the optimal strategy encoded in the parameters $(\mu_1, \mu_2, P_{\mu_1}, P_{\mu_2}, P_X)$ required to achieve the maximal SKR shown left.

The right panel reveals the optimal strategy encoded in the parameters $(\mu_1, \mu_2, P_{\mu_1}, P_{\mu_2}, P_X)$ required to achieve the maximal SKR. In the low-loss, stable regime (approximately 0-100 km), the parameters remain relatively constant: the signal state intensity μ_1 (blue) holds near 0.6, the decoy intensity μ_2 (orange) is kept low (0.15), the decoy probability P_{μ_2} (red) is high (0.6), the signal probability P_{μ_1} (green) is consequently low (0.05), and the X-basis measurement probability P_X (purple) sits around 0.4, indicating a slight preference for Z-basis measurements. As channel loss increases beyond 100 km, entering the transition regime (100-140 km), obtaining reliable error estimates becomes paramount. The most notable adaptation is the significant decrease in P_X, implying an increased reliance on Z-basis measurements ($P_Z = 1 - P_X$) to gather sufficient statistics for both key generation and robust QBER estimation in the noisier channel, even with fewer direct X-basis samples; concurrently, P_{μ_1} begins a gradual increase. Finally, in the near-cutoff regime ($>$ 140 km), where the SKR rapidly collapses, the optimizer makes drastic adjustments. Sharp fluctuations and spikes become evident, particularly in μ_1, μ_2, and P_{μ_2}, likely representing the algorithm pushing parameters to their boundaries to satisfy security constraints $(\epsilon_{\text{sec}}, \epsilon_{\text{cor}})$ and extract any residual key rate amidst overwhelming loss and noise relative to the small block size ($n_X = 10^4$). The specific complex behavior of P_{μ_2} near the cutoff may also reflect critical interactions within the security proofs' statistical bounds. In essence, Figure 11 demonstrates how numerical optimization adapts the QKD protocol parameters to maximize the secure key rate under the constraints imposed by channel physics, detector imperfections, and finite-statistics security analysis. The parameter variations, especially the reduction in P_X at intermediate distances and the sharp changes near the cutoff, highlight the protocol's dynamic response to increasing channel attenuation for this specific, small block size.

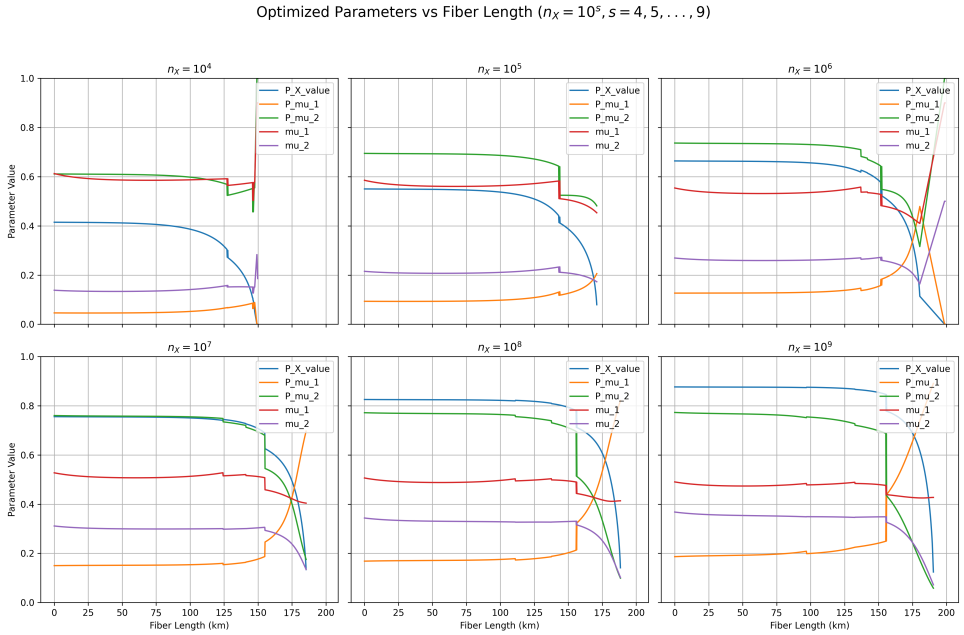

Figure 12: Optimized parameters $(\mu_1, \mu_2, P_{\mu_1}, P_{\mu_2}, P_X)$ versus fiber length (0–200 km) for varying total pulse counts $n_X = 10^4$ to 10^9. The plots illustrate adaptive adjustments determined by the dual annealing algorithm in response to channel conditions and available statistics.

Figure 12 provides a comparative overview of the optimized parameter strategies across different total pulse counts (n_X) ranging from 10^4 to 10^9. A key observation is the strong dependence of the optimal parameter settings not only on the fiber length (channel loss) but also on the available block size, n_X. Across all presented n_X values, a common pattern emerges: a region of relative parameter stability at shorter fiber lengths gives way to significant adjustments and volatility as the transmission distance approaches its respective maximum limit. However, the extent of these regions and the specific parameter values differ notably with n_X. At higher n_X (e.g., $10^8, 10^9$), the region of stability extends further, and the subsequent parameter transitions occur at longer distances (around 150-180 km) compared to lower n_X cases (e.g., $10^4, 10^5$), where transitions begin earlier (around 100-145 km). This directly reflects the

enhanced statistical precision afforded by larger block sizes, which allows the system to securely tolerate higher channel losses and thus achieve greater secure distances. Examining individual parameters reveals consistent strategic adaptations. The X-basis probability, P_X, invariably decreases at longer distances for all n_X, indicating a universal strategy to shift measurement focus towards the Z-basis for more robust error estimation when noise becomes significant. However, the onset of this decrease shifts to longer distances with increasing n_X. Similarly, the signal probability P_{μ_1} tends to increase near the cutoff distance, likely to maximize raw key generation potential when security bounds tighten. The decoy intensity μ_2 and its probability P_{μ_2} exhibit complex behavior, particularly the sharp spikes or dips near the maximum distance. These extreme adjustments likely represent the optimizer pushing parameter boundaries to meet stringent security constraints ($\epsilon_{\text{sec}}, \epsilon_{\text{cor}}$) under high noise and finite-size limitations, with the volatility being somewhat less pronounced for larger n_X where statistical estimates are inherently more stable.

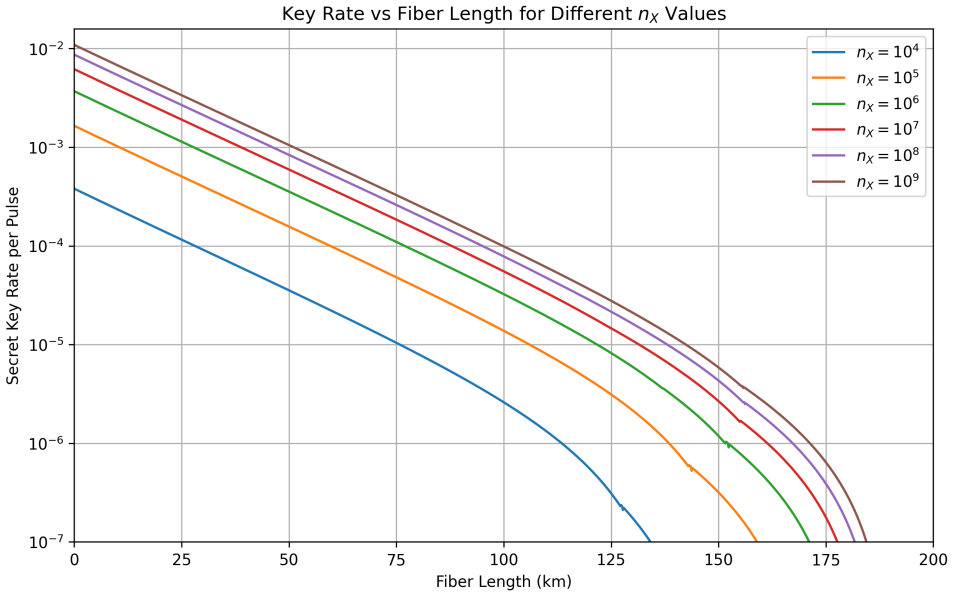

Figure 13: Optimized Secret Key Rate vs. Fiber Length for Different Total Pulse Counts (n_X). The key rate is plotted on a logarithmic scale ($\log_{10}$) for n_X values ranging from 10^4 to 10^9.

Figure 13 consolidates the results of the secret key rate (SKR) optimization, presenting the maximum achievable SKR as a function of fiber length for various total pulse counts (n_X) from 10^4 to 10^9. This provides a comprehensive view of the system's performance potential under different statistical conditions. As universally expected, the SKR for all n_X values exhibits a monotonic decrease with increasing fiber length, plotted here on a logarithmic scale. This decline is fundamentally driven by exponential signal attenuation in the fiber ($\eta = 10^{-\alpha L/10}$ with $\alpha = 0.2$ dB/km) and the increasing relative impact of detector noise at longer distances. The most critical insight from this figure is the pronounced effect of the total pulse count, n_X, on performance. At any given fiber length, a larger n_X consistently yields a higher SKR. Furthermore, increasing n_X significantly extends the maximum secure transmission distance – the point where the SKR drops below a practical threshold (e.g., 10^{-8}). This distance expands from approximately 145 km for $n_X = 10^4$ to about 180 km for $n_X = 10^9$. The underlying reason for this improvement is the mitigation of finite-size effects in the security analysis. Larger datasets (n_X) allow for more precise estimation of channel parameters (like error rates and phase error bounds) with tighter confidence intervals, reducing the statistical penalties subtracted during privacy amplification and leading to a higher net secure key rate [17]. Observe also that while each order-of-magnitude increase in n_X provides a substantial boost, the vertical spacing between the curves slightly diminishes at the highest n_X values (e.g., between 10^8 and 10^9). This suggests diminishing returns as the performance gradually approaches the theoretical asymptotic limit achievable with an infinite number of pulses. These optimized key rate curves represent the outcome of the parameter tuning process (detailed in Figure 12), where variables like intensities and basis choices were specifically adjusted by the dual annealing algorithm for each combination of L and n_X to maximize the secure throughput under rigorous security constraints.

A comparative analysis of secret key rate (SKR) performance under different optimization conditions is

facilitated by examining Figures 6, 10, and 13. All depict the SKR versus fiber length for the decoy-state BB84 protocol, assuming a channel loss of $\alpha = 0.2\,\mathrm{dB/km}$. Figure 6, sourced from optimized results presented by Lim et al. [1], serves as a literature benchmark. It illustrates SKRs starting around 10^{-2} bits per pulse at zero distance and extending beyond 175 km before dropping below 10^{-7}, showcasing the potential reach under their specific optimization and parameter assumptions. Figure 10 represents results from our own analytical simulation, likely employing simplified or partially optimized parameters. This yields noticeably lower performance, with initial SKRs ranging from $\sim 10^{-3.5}$ ($n_X = 10^4$) to only $\sim 10^{-2.5}$ ($n_X = 10^9$), and a maximum reach that appears somewhat limited compared to the benchmark, although the plot extends to 200 km showing rates down to 10^{-7}. Finally, Figure 13 presents the outcome of our comprehensive parameter optimization using the Dual Annealing algorithm. These results show a marked improvement over the partially optimized simulation in Figure 10. The initial SKR for $n_X = 10^9$, for instance, reaches $\sim 10^{-2}$, significantly higher than the $\sim 10^{-2.5}$ in the analytical simulation and closely matching the literature benchmark in Figure 6. More importantly, the maximum secure distances achieved via Dual Annealing (ranging from 145 km for $n_X = 10^4$ to approximately 180 km for $n_X = 10^9$, considering a $\sim 10^{-8}$ threshold) are substantially greater than those implied by less optimized approaches and are consistent with state-of-the-art expectations shown in Figure 6. The superior performance demonstrated in Figure 13 stems directly from the effectiveness of the Dual Annealing optimization in finding optimal parameter sets $(\mu_1, \mu_2, P_{\mu_1}, P_{\mu_2}, P_X)$ tailored for each specific fiber length and n_X value, as detailed previously in Figure 12. The adaptive strategies, such as adjusting basis choices (P_X) and intensities (μ_k) in response to increasing attenuation, allow the protocol to maximize the secure throughput and extend the operational range significantly compared to using fixed or only partially optimized parameters. This comparison clearly underscores the critical importance of thorough, dynamic parameter optimization for achieving maximal performance in practical QKD systems.

However, given the computational expense associated with methods like Dual Annealing, we explore the potential of neural networks as a high-speed alternative for determining these optimal settings. The specific experimental parameters adopted for generating the optimization data used to train and evaluate this neural network approach, are detailed in the next section.

C. Neural Network

The parameter values used in this study are based on [8], which specifies 50 km, $P_{dc} = 1.75 \times 10^{-7}$, $e_d = 0.0287$, $N = 2.99 \times 10^{12}$, $\alpha = 0.2\,\mathrm{dB/km}$, and $f_e = f_{EC} = 1.16$. We adopt the following additional parameters:

$$\eta_{Bob} = 0.1 \qquad P_{ap} = 0.2$$
$$\epsilon_{sec} = 10^{-10} \qquad \epsilon_{cor} = 10^{-15}$$

Since [8] notes $e_d \in [11, 14]$ and $N \in [10^{11}, 10^{14}]$, we optimize with fixed post-processing block sizes $n_X = 10^s$ for $s = 6, 7, 8, 9$, enabling comparison with their results.

Figure 14 presents a detailed comparison between the optimal parameters for the decoy-state BB84 protocol, found via numerical optimization (dual annealing, solid lines), and the parameters predicted by the trained neural network (dotted lines/markers) after 5,000 epochs. The comparison is shown for varying total signal counts, $n_X = 10^s$ ($s = 4, \ldots, 9$), as a function of fiber length. The most salient feature is the excellent fidelity of the neural network's predictions; the predicted parameter values closely track the numerically optimized ones across the entire range of fiber lengths and n_X values considered. This demonstrates the network's successful learning of the complex mapping from input conditions (related to fiber length, n_X, etc., encoded in the features e_1 to e_4) to the five optimal operational parameters $(\mu_1, \mu_2, P_{\mu_1}, P_{\mu_2}, P_X)$. Observing the parameter dynamics, we note that for a given n_X, the optimal parameters often remain relatively stable over shorter fiber lengths before exhibiting sharp changes near the maximum achievable distance where the secret key rate (SKR) drops below a practical threshold (e.g., 10^{-8}). For example, with lower signal counts ($n_X = 10^4$ to 10^6), parameters like P_X (blue) hover around 0.5–0.6, while μ_1 (red) is initially near 0.7–0.8. These values hold reasonably steady until approximately 145–155 km, beyond which they adapt rapidly. Conversely, for larger signal counts ($n_X = 10^7$ to 10^9), the system can tolerate higher losses, extending the secure transmission distance. This is reflected in

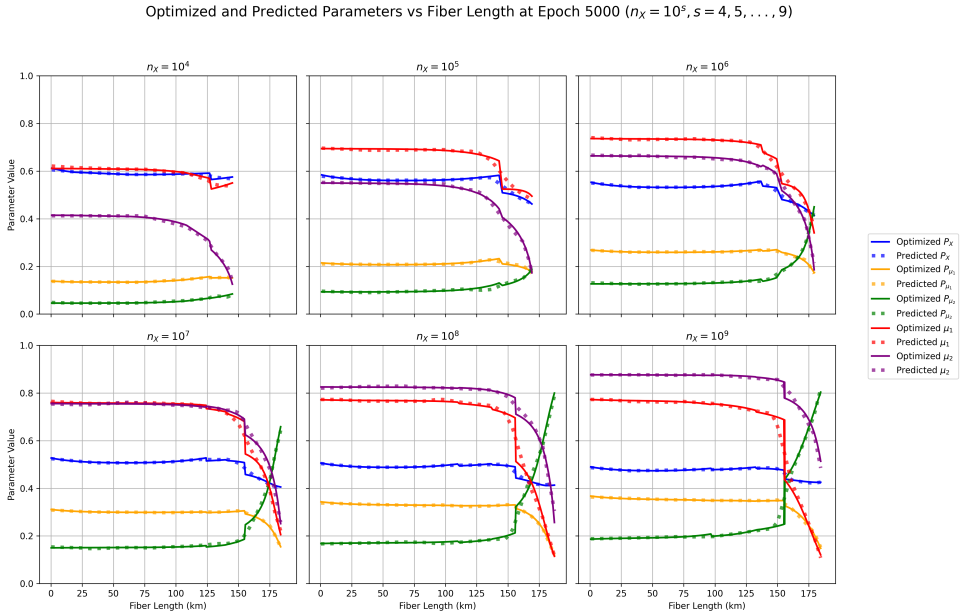

Figure 14: Predicted and Optimized Parameters vs. Fiber Length for Different n_X Values ($n_X = 10^s, s = 4, 5, \ldots, 9$).

the plots where the sharp parameter adaptations occur at longer fiber lengths, typically between 160–175 km. This behavior aligns with the corresponding key rate performance shown in Figure 15. The neural network accurately captures this dependence on n_X, further validating its utility for generating appropriate parameters across different operational regimes, which is crucial for real-time optimization in practical QKD systems.

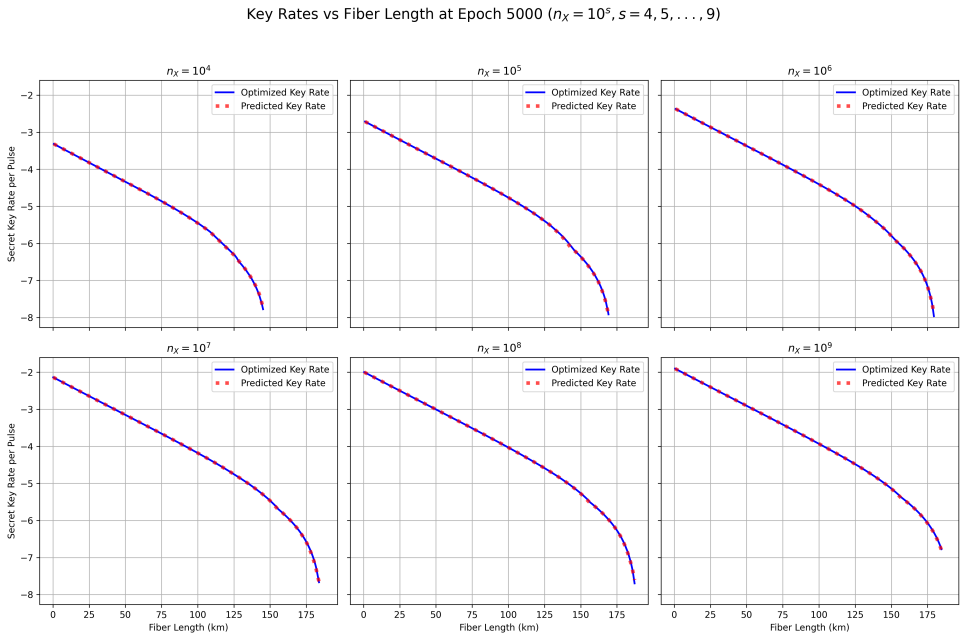

Figure 15: Predicted and Optimized Key Rates vs. Fiber Length for Different n_X Values ($n_X = 10^s, s = 4, 5, \ldots, 9$) at Epoch 5000.

Building upon the accurate parameter predictions demonstrated in Figure 14, Figure 15 illustrates the resulting secret key rate (SKR) performance. It compares the SKR achieved using the numerically optimized parameters (solid blue lines) against the SKR obtained by applying the parameters generated by the neural network (red markers) after 5,000 training epochs, across the range of total signal counts $n_X = 10^4$ to 10^9. The optimized SKR follows the expected trend for decoy-state BB84, decreasing

quasi-logarithmically with transmission distance. Higher signal counts (n_X) yield significantly better performance, both in terms of initial key rate (ranging from approximately $10^{-3.5}$ for $n_X = 10^4$ to 10^{-2} for $n_X = 10^9$ at zero distance) and maximum secure distance (extending from roughly 150 km for $n_X = 10^4$ to about 175 km for $n_X = 10^9$, considering a typical threshold SKR of 10^{-8}). The most striking feature of this figure is the remarkable overlap between the predicted and optimized key rates. The SKR calculated using the neural network's output parameters (red markers) aligns almost perfectly with the maximum achievable SKR (blue lines) obtained via numerical optimization. This high degree of accuracy persists across all tested n_X values and the full range of fiber lengths where a positive key rate is achievable.

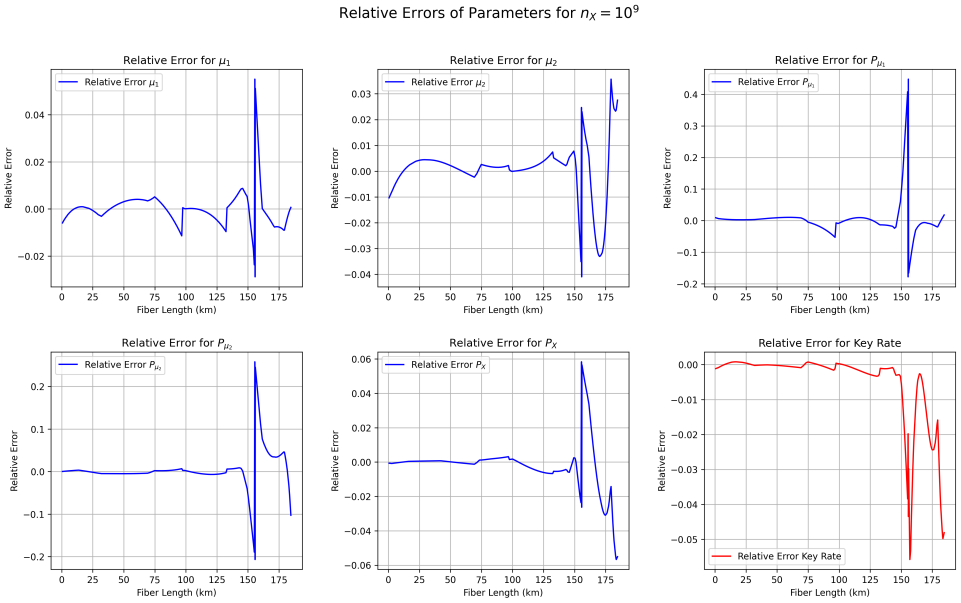

Figure 16: Relative errors of predicted versus optimized parameters ($\mu_1, \mu_2, P_{\mu_1}, P_{\mu_2}, P_X$) and secret key rate for $n_X = 10^9$, at epoch 5,000 of neural network training.

To quantitatively assess the neural network's prediction accuracy, Figure 16 plots the relative errors between the predicted and optimized parameters, as well as the resulting secret key rate (SKR), specifically for the high signal count case of $n_X = 10^9$. The relative error is calculated as (Predicted - Optimized) / Optimized. Over a significant portion of the transmission range, approximately 0 to 150 km, the network exhibits high precision. The relative errors for the intensities (μ_1, μ_2) and probabilities ($P_{\mu_1}, P_{\mu_2}, P_X$) predominantly remain within a narrow band, typically below $\pm 2\%$. This exceptional agreement in the stable operating region underpins the close match between predicted and optimized SKR observed in Figure 15 over these distances. The relative error in the SKR itself is consistently negative but very small in magnitude, generally less than 1%, indicating a slight, but negligible, underestimation by the NN-derived parameters in this regime. As the fiber length approaches the maximum secure distance (175 km for $n_X = 10^9$), the relative errors for the parameters become notably larger and more volatile. This region corresponds to the area where the optimal parameters undergo rapid adjustments to maximize the dwindling SKR, as seen in Figure 14. Peak relative errors reach approximately 5% for μ_1, 3% for μ_2, 6% for P_X, -20% for P_{μ_2}, and a significant 40% for P_{μ_1}. These larger deviations suggest the network finds it more challenging to perfectly replicate the optimization results where parameter sensitivity is high and optimal values may change abruptly. It is also worth noting that relative errors can be amplified when the optimized parameter value itself becomes small near the cutoff. Despite the pronounced spikes in some parameter errors near the maximum distance, the relative error in the final SKR remains relatively well-contained. While exhibiting oscillations, the SKR relative error peaks at approximately -6%. A prediction accuracy leading to SKR deviations within $\pm 5 - 10\%$ is often considered acceptable, and sometimes excellent, in practical QKD analysis and system modeling, especially when compared to other system uncertainties. Therefore, even in the most challenging, low-rate regime near the cutoff distance, the neural network maintains a reasonable level of accuracy for the primary figure of merit, the secret key rate. This further reinforces its potential as a reliable tool for efficient parameter estimation in QKD systems.

After 5,000 training epochs, the neural network demonstrates significant promise for real-time parameter optimization in decoy-state BB84 QKD. As shown by the close alignment in Figure 14 and Figure 15, the network accurately predicts both the optimal operational parameters ($\mu_1, \mu_2, P_{\mu_1}, P_{\mu_2}, P_X$) and the resultant Secret Key Rate (SKR) across a wide range of signal counts ($n_X = 10^4$ to 10^9) and fiber lengths. A quantitative analysis of the relative errors for the representative case $n_X = 10^9$ (Figure 16) confirms this high accuracy over most of the transmission distance (approx. 0–150 km), where relative errors for parameters typically stay below $\pm 2\%$, and the SKR relative error remains consistently small, generally less than 2%. While the relative errors for individual parameters increase notably near the maximum transmission distance (150–175 km), where optimal parameters change rapidly to counteract high noise, the impact on the final SKR is well-controlled. Despite observing peak relative errors as high as 40% for certain parameters like P_{μ_1} in this challenging regime, the relative error in the SKR itself remains bounded, peaking at approximately -6%. This level of deviation in the key rate is well within the $\pm 10\%$ margin often considered acceptable for practical QKD applications. Crucially, contrary to potential misinterpretations, the SKR relative error does not exhibit extreme spikes nor does it drastically reduce the predicted secure distance. The network successfully tracks the positive key rate close to the original 175 km limit found via optimization. Therefore, the trained network provides reliable SKR predictions across the entire operational range.

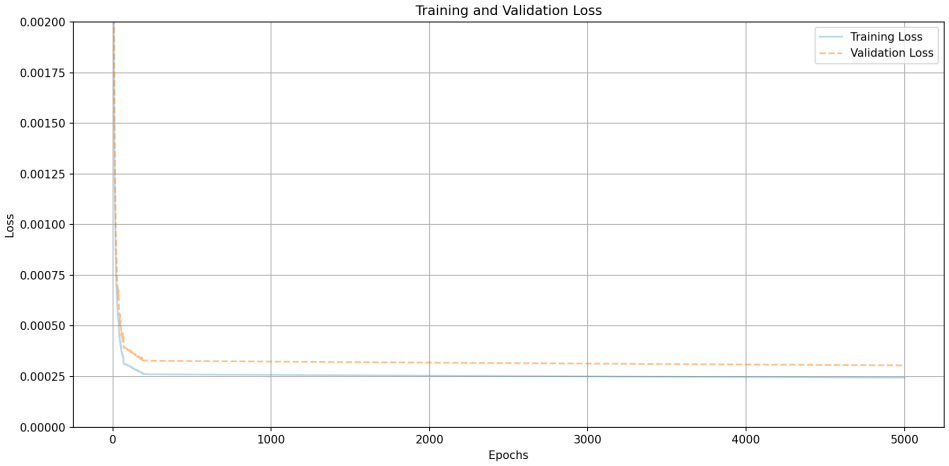

Figure 17: Training and Validation Loss vs. Epochs. The plot illustrates the convergence behavior of the neural network model over 5,000 epochs.

Figure 17 displays the training and validation loss curves over the 5,000 training epochs, providing insight into the model's learning process and generalization capability. Both loss curves exhibit an extremely rapid decrease within the initial phase, approximately the first 100–200 epochs. Starting from very high initial values, the losses plummet, indicating that the model quickly learns the dominant features and patterns from the training data using the initial, higher learning rate. Following this steep decline, the rate of improvement slows considerably. The model continues to refine its parameters, with both training and validation losses gradually reaching a stable plateau around epoch 300. After this point, further training yields minimal changes in loss. The training loss converges to a value of approximately 2.7×10^{-4}, while the validation loss stabilizes slightly higher, at around 3.1×10^{-4}. This asymptotic behavior, reached relatively early in the 5,000-epoch training run, suggests that the model converged effectively. Crucially, the validation loss closely tracks the training loss throughout the entire process, consistently maintaining only a small gap between the two curves. The training loss remains below the validation loss, as expected, but the lack of divergence or significant widening of the gap indicates good generalization. There is no evidence of overfitting; the validation loss does not increase, even during the extended plateau phase from epoch 300 to 5,000. The observed convergence pattern is consistent with the use of the 'ReduceLROnPlateau' learning rate scheduler. The rapid initial learning phase likely occurred with the starting learning rate, while the stabilization around epoch 300 suggests the scheduler reduced the learning rate upon detecting stagnation in the validation loss, allowing the model to settle into a stable minimum. The extended training duration primarily served to confirm the stability of this convergence. Overall, the loss curves indicate a successful training process resulting in a well-generalized model.

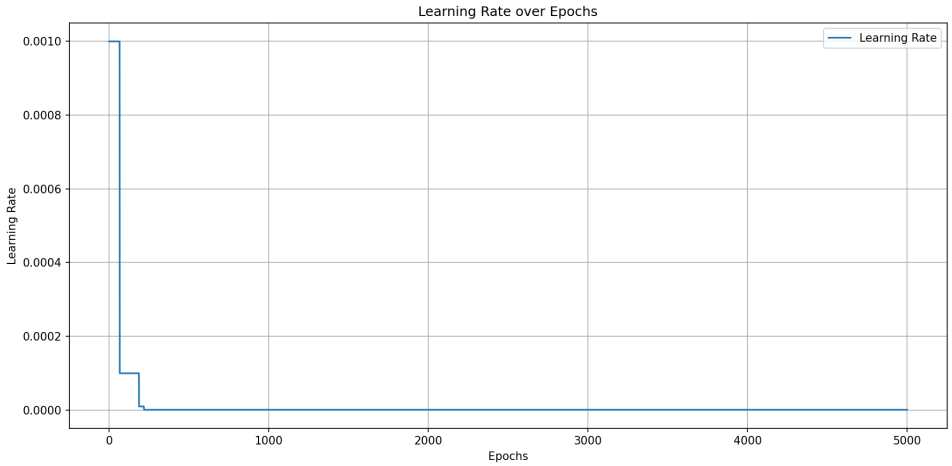

Figure 18: Learning Rate Evolution over Epochs. This plot illustrates the adaptive adjustments made by the learning rate scheduler during the 5,000-epoch training process.

Figure 18 tracks the learning rate used by the optimizer throughout the training. This schedule reflects the behavior of the 'ReduceLROnPlateau' scheduler, which dynamically adjusts the learning rate based on the validation loss performance. The training commences with an initial learning rate of 10^{-3} (0.001). This relatively high value facilitates rapid initial learning, allowing the model to make significant progress in reducing the loss during the early epochs, corresponding to the steep decline seen in Figure 17. The learning rate remains constant at 10^{-3} for approximately the first 150-200 epochs. Subsequently, as the improvement in validation loss likely began to slow, the scheduler triggered the first reduction, decreasing the learning rate by a factor of 10 (consistent with the scheduler's 'factor=0.1' setting) to 10^{-4}. This pattern repeats: the learning rate holds steady at 10^{-4} for another period before dropping to 10^{-5}, and then again holds before dropping to 10^{-6}. Each step-down signifies that the validation loss had plateaued for a number of epochs defined by the scheduler's 'patience' parameter, prompting a reduction to enable finer adjustments to the model's weights. The final reduction brings the learning rate to 10^{-6} at approximately epoch 500. This value corresponds to the minimum learning rate ('$min_lr = 1e-6$') set for the scheduler. Once this floor is reached, the learning rate remains fixed at 10^{-6} for the remainder of the 5,000 epochs, regardless of further validation loss behavior. This ensures that the model continues making very small, stable updates, focusing on fine-tuning without the risk of divergence associated with higher learning rates, especially after convergence has been largely achieved (around epoch 500, as seen in Figure 17). The plot thus confirms the effective operation of the adaptive learning rate strategy, balancing rapid initial convergence with stable refinement in later stages.

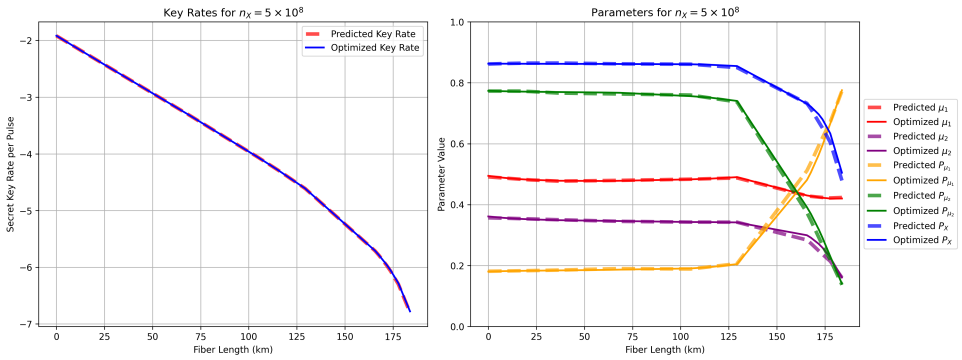

Figure 19: Evaluation of Neural Network Generalization: Predicted vs. Optimized Key Rates and Parameters for unseen $n_X = 5 \times 10^8$. Left: Secret key rate (SKR) comparison. Right: Operational parameter $(\mu_1, \mu_2, P_{\mu_1}, P_{\mu_2}, P_X)$ comparison. Solid lines/markers represent optimized values (via dual annealing); dashed lines/markers represent NN predictions after 5,000 epochs.

Figure 19 specifically assesses the trained neural network's ability to generalize to conditions not explicitly present in the training set ($n_X = 10^4, \ldots, 10^9$). It compares the network's predictions against

numerically optimized results for an intermediate signal count of $n_X = 5 \times 10^8$, using an unseen test dataset of 100 points. The left panel confirms excellent generalization in terms of the primary performance metric, the Secret Key Rate (SKR). The predicted SKR (dashed red line) closely overlays the optimized SKR benchmark (solid blue line) across the entire fiber length range. Both follow the expected logarithmic decay, starting near 10^{-2} bits per pulse at 0 km and decreasing to below 10^{-7} around 175 km, demonstrating that the network accurately translates its parameter predictions into reliable key rate estimates even for this novel n_X value. The right panel delves into the underlying parameter predictions. Consistent with observations within the training range (cf. Figure 14), the predicted parameters (dashed lines) exhibit high fidelity to the optimized values (solid lines). In the stable region up to approximately 150 km, the network accurately captures parameters like $P_X \approx 0.85$, $\mu_1 \approx 0.5$, $\mu_2 \approx 0.35$, and $P_{\mu_2} \approx 0.75$. More importantly, the network also successfully replicates the sharp parameter transitions occurring beyond 150 km, necessary for maximizing the SKR under high loss. This includes the decrease in P_X and μ_2, and the sharp increase in P_{μ_1} as the protocol adapts near the maximum distance. Collectively, these results strongly indicate that the neural network has learned the underlying physical relationships governing the optimal parameter choices, rather than simply memorizing the training data points. Its ability to accurately interpolate and predict performance for an unseen intermediate n_X value validates its potential as a robust and computationally efficient alternative to repeated numerical optimization for real-time parameter control in practical QKD systems operating under varying conditions.

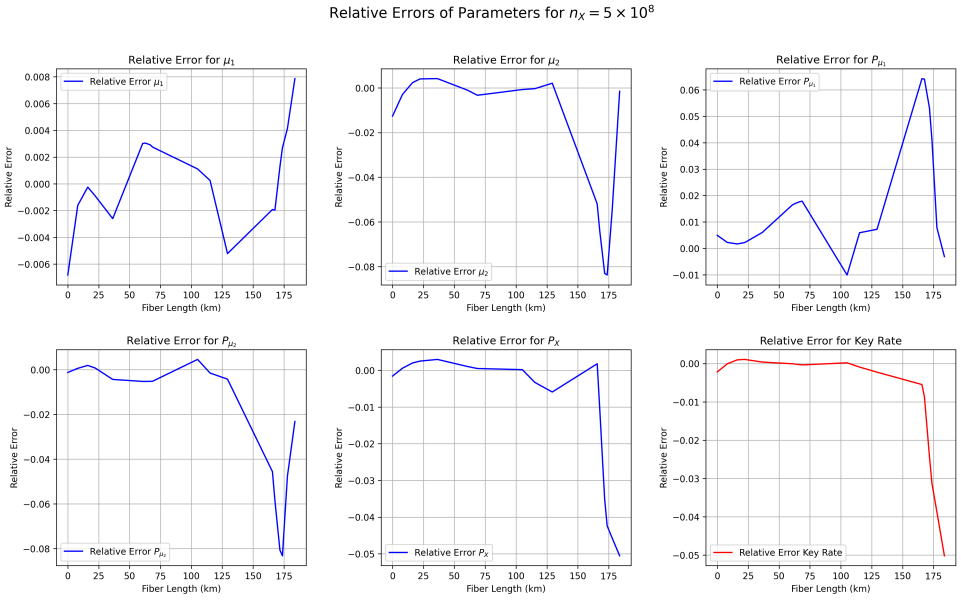

Figure 20: Relative errors of predicted versus optimized parameters $(\mu_1, \mu_2, P_{\mu_1}, P_{\mu_2}, P_X)$ and secret key rate (SKR) for the unseen case $n_X = 5 \times 10^8$.

To quantitatively evaluate the neural network's generalization performance shown qualitatively in Figure 19, Figure 20 presents the relative errors between predicted and optimized values for the unseen $n_X = 5 \times 10^8$ scenario. Across the majority of the operational range (approximately 0 to 150 km), the prediction errors are remarkably small. The relative errors for all five parameters $(\mu_1, \mu_2, P_{\mu_1}, P_{\mu_2}, P_X)$ generally fluctuate within a tight band of $\pm 1\%$, with many points even closer to zero. This high level of precision directly corresponds to the near-perfect overlap observed in the parameter and SKR plots (Figure 19) over these distances. Consistent with behavior observed within the training data range (e.g., Figure 16), the relative errors exhibit an increase in magnitude and volatility as the fiber length approaches the maximum transmission distance (150–175 km). This reflects the increased sensitivity and rapid changes required of the optimal parameters in the high-loss regime. Specifically, peak relative errors near the cutoff reach approximately 0.8% for μ_1, -7% for μ_2, 6.5% for P_{μ_1}, -8% for P_{μ_2}, and -4.5% for P_X. While noticeable, these parameter deviations are still reasonably constrained. Crucially, the relative error in the Secret Key Rate remains very well-behaved throughout. It stays exceptionally close to zero (negligibly small negative values) up to 150 km. Beyond this point, as parameter errors increase, the SKR relative error trends downward but only reaches a maximum underestimation of approximately -5% near the 175 km mark. This level of accuracy in the final key rate is well within typical tolerance levels $(\pm 5 - 10\%)$ considered acceptable for QKD system analysis and practical implementation. Therefore,

this quantitative analysis confirms the neural network's strong generalization capability. It accurately predicts parameters and, most importantly, the resulting SKR for unseen intermediate conditions, even maintaining acceptable SKR accuracy in the challenging high-attenuation regime near the maximum distance. While minor parameter deviations increase at the very edge of the operational range, they do not translate into significant inaccuracies in the key rate, validating the network's robustness for this application.

D. Computational Efficiency and Practical Considerations

Aspect	**Details (BB84 Protocol)**
Evaluation Platform	MacBook Pro M2 Pro (12-core CPU, 19-unit GPU)
Optimization Method	Dual Annealing (SciPy implementation)
Optimization Performance	CPU-bound (10 cores used); approx. 45 min per 1,000 data points
Neural Network Training	GPU-accelerated (19-unit GPU); approx. 5 min 30 s (5,000 epochs on 6,000 points)
Neural Network Inference	GPU-accelerated; approx. 10 s per 1,000 data points

Table 5: Comparison of computation times for QKD parameter optimization using Dual Annealing versus Neural Network training and inference on the specified hardware.

A critical factor for the practical deployment of QKD systems is the ability to rapidly determine optimal operating parameters in response to changing channel conditions. We evaluated the computational performance of the traditional numerical optimization approach (Dual Annealing) against the trained neural network on an ARM-based MacBook Pro M2 Pro platform, with key results summarized in Table 5. The Dual Annealing optimization, implemented via SciPy, is inherently CPU-bound. Utilizing 10 CPU cores, it required approximately 45 minutes to optimize the parameters for 1,000 different operating points (representing varying fiber lengths in this study). Extrapolating, generating the full training dataset of 6,000 points took roughly 270 minutes (4.5 hours).

This method suffers from several limitations regarding real-time application and even offline data generation. Firstly, its intensive CPU usage consumes significant system resources, potentially limiting the capacity available for other critical tasks like system control or real-time data processing [5]. Secondly, Dual Annealing lacks support for GPU acceleration; standard implementations like SciPy's are CPU-only and cannot exploit the parallel processing capabilities of modern GPUs, whether they are Apple Silicon, NVIDIA, AMD, or integrated Intel units [18]. This lack of acceleration is compounded by the algorithm's inherently sequential nature, which resists efficient parallelization and further hinders potential speed improvements [4, 19]. Platform-specific issues can also arise; for instance, on ARM architectures like Apple Silicon, reliance on potentially non-native optimization libraries might incur emulation overhead via Rosetta 2 on macOS, impacting performance [20]. Furthermore, practical execution can be hampered by occasional, unpredictable stalling of the optimization process itself. Such occurrences necessitate manual intervention to determine whether the process is genuinely stuck or just progressing slowly, often leading to restarts and adding significant uncertainty and hidden time costs to the already lengthy task of generating large datasets. These factors render Dual Annealing unsuitable for applications demanding rapid parameter updates.

In stark contrast, the neural network approach shifts the computational burden. While generating the initial training dataset via Dual Annealing is time-consuming, this is an offline, one-time cost. The neural network training itself, leveraging the M2 Pro's 19-unit GPU via PyTorch's MPS backend, was highly efficient, completing 5,000 epochs on the 6,000-point dataset in just 5 minutes and 30 seconds. The most significant advantage lies in the inference speed. Once trained, the neural network predicted the optimal parameters for an unseen test set of 100 points in approximately 1 second. To achieve the same result, Dual Annealing would require roughly 4.5 minutes (45 minutes / 10). This represents a substantial speedup of approximately 270 times for the neural network inference compared to direct optimization. Furthermore, trained neural networks can be deployed using lightweight frameworks opti-

mized for efficient inference on diverse hardware, including ARM CPUs and GPUs, enhancing portability and reducing runtime dependencies [21, 22]. While numerical optimization like Dual Annealing directly searches the solution space and can yield high precision, the neural network provides an approximation. However, as demonstrated in Figures 14 to 20, the network achieves excellent accuracy, with resulting key rate deviations well within acceptable limits ($\approx \pm 5\%$) even near transmission cutoffs. Given the dramatic improvement in computational speed, the neural network presents a highly practical and effective solution for real-time parameter tuning in dynamic QKD environments, striking an excellent balance between accuracy and efficiency.

VII. Conclusion

This study successfully addressed the challenge of efficient parameter optimization for decoy-state BB84 quantum key distribution (QKD), aiming to enhance its feasibility for real-time applications, especially on resource-constrained platforms. We first established the performance characteristics through comprehensive numerical optimization using Dual Annealing. This process confirmed the expected exponential decay of the secret key rate (R) with distance and quantified the significant, albeit diminishing, benefits of increasing the block size (n_X) – for instance, the gain in R drops from over 2300% ($10^4 \to 10^5$) to just 1.4% ($10^8 \to 10^9$) at 100 km (Table **??**). The optimization effectively maximized performance, achieving secure distances ranging from approximately 145 km ($n_X = 10^4$) to 180 km ($n_X = 10^9$), extending the reach compared to non-optimized approaches and aligning well with literature benchmarks.

However, the practical utility of Dual Annealing for dynamic QKD is limited by its computational demands. Requiring roughly 45 minutes per 1,000 points on a capable multi-core CPU and lacking GPU acceleration, it is ill-suited for rapid parameter updates. Practical challenges, including unpredictable stalling during execution, further add to the time cost, especially for generating the large datasets needed for alternative approaches.

To overcome this speed limitation, we developed and evaluated a neural network model trained on the optimized parameter data. While the offline training requires an initial investment (approx. 5.5 minutes for 5,000 epochs on 6,000 points using GPU acceleration), the crucial advantage lies in its inference speed. The trained network predicted optimal parameters for 100 unseen test points ($n_X = 5 \times 10^8$) in approximately 1 second – a dramatic 270-fold speedup compared to the estimated 4.5 minutes required by Dual Annealing for the same task.

Critically, this computational efficiency did not come at a significant cost to accuracy. The neural network demonstrated excellent performance, accurately predicting both the optimized parameters and the resulting secret key rate across the full range of trained n_X values (Figures 14, 15) and generalizing effectively to the unseen $n_X = 5 \times 10^8$ case (Figure 19). Quantitative analysis confirmed that relative errors in the predicted SKR were typically below 2% over most distances, reaching a maximum deviation of only about -6% near the transmission cutoff (Figure 20). This level of precision is well within acceptable margins for practical QKD implementation, and importantly, the network did not significantly underestimate the key rate nor reduce the achievable secure distance. While minor deviations in individual parameters were observed near the cutoff, the network reliably predicted the key operational outcome.

In conclusion, this work validates the use of neural networks as a highly effective and computationally efficient tool for real-time parameter estimation in decoy-state BB84 QKD. While Dual Annealing provides high precision suitable for offline analysis or systems with stable conditions, the neural network's rapid inference capabilities make it vastly preferable for dynamic scenarios and deployment on low-power devices. Future work could focus on refining the network's accuracy specifically at the extreme edge of the operational range, perhaps through targeted data augmentation or architectural enhancements, and incorporating more realistic device models, including detector imperfections like afterpulsing, to bridge the gap towards practical, high-performance quantum network deployment.

A. Appendix

A. Details in Key Rate Equation

The secret key rate calculation is based on the work in [1]. The key rate R is defined as:

$$R = \frac{l}{N}, \tag{30}$$

where l is the length of the final secret key (in bits) and N is the total number of pulses sent.

Detection and Probability Models

The expected detection rate (excluding after-pulse contributions) for intensity level k is:

$$D_k = 1 - (1 - 2P_{dc})e^{-\eta_{sys}\mu_k}, \tag{31}$$

where $\eta_{sys} = \eta_{ch}\eta_{Bob}$ is the system transmittance, μ_k is the mean photon number for intensity level $k \in \{1, 2, 3\}$, and P_{dc} is the dark count probability.

The probability of detecting n photons given intensity μ_k follows a Poisson distribution:

$$P(n|\mu_k) = \frac{\mu_k^n e^{-\mu_k}}{n!}. \tag{32}$$

The joint probability of a detection event and intensity μ_k is:

$$P(\text{det}, \mu_k) = D_k P_{\mu_k}, \tag{33}$$

where P_{μ_k} is the probability of choosing intensity μ_k, satisfying $\sum_k P_{\mu_k} = 1$.

X-Basis Analysis

The probability of detection in the X basis for intensity μ_k is:

$$P(\text{det}, \mu_k, X) = P(\text{det}, \mu_k)P_X^2, \tag{34}$$

with the total detection probability in the X basis given by:

$$\sum_k P(\text{det}, \mu_k, X). \tag{35}$$

The conditional probability of intensity μ_k given a detection in the X basis is:

$$P(\mu_k|\text{det}, X) = \frac{P(\text{det}, \mu_k, X)}{\sum_k P(\text{det}, \mu_k, X)}. \tag{36}$$

The expected number of events for intensity μ_k in the X basis is:

$$n_{X,\mu_k} = n_X \times P(\mu_k|\text{det}, X), \tag{37}$$

where $n_X = \sum_k n_{X,\mu_k}$ is the total number of events in the X basis. The total number of pulses sent is estimated as:

$$N = \frac{n_X}{\sum_k P(\text{det}, \mu_k, X)}. \tag{38}$$

Z-Basis Analysis

Similarly, for the Z basis:

$$\begin{aligned} P(\text{det}, \mu_k, Z) &= P(\text{det}, \mu_k) P_Z^2, \\ \sum_k P(\text{det}, \mu_k, Z) &= \sum_k P(\text{det}, \mu_k, Z), \\ P(\mu_k|\text{det}, Z) &= \frac{P(\text{det}, \mu_k, Z)}{\sum_k P(\text{det}, \mu_k, Z)}, \\ n_{Z,\mu_k} &= n_Z \times P(\mu_k|\text{det}, Z), \\ n_Z &= \sum_k n_{Z,\mu_k}, \end{aligned} \tag{39}$$

with an alternative expression:

$$n_{Z,\mu_k} = N D_k P_{\mu_k} P_Z^2. \tag{40}$$

Finite-Key Security Bounds

To account for finite-key effects, security bounds for n_{X,μ_k} and n_{Z,μ_k} are:

$$n_{X,k}^{\pm} = \frac{e^k}{p_k} \left[n_{X,k} \pm \sqrt{\frac{n_X}{2} \ln \frac{21}{\epsilon_{\text{sec}}}} \right], \quad n_{Z,k}^{\pm} = \frac{e^k}{p_k} \left[n_{Z,k} \pm \sqrt{\frac{n_Z}{2} \ln \frac{21}{\epsilon_{\text{sec}}}} \right], \tag{41}$$

where ϵ_{sec} is the security parameter, and $k \in \mathcal{K}$.

The probability that Alice prepares an n-photon state is:

$$\tau_n = \sum_{k \in \mathcal{K}} \frac{p_k e^{-\mu_k} \mu_k^n}{n!}. \tag{42}$$

Decoy-State Analysis

Using decoy-state analysis, the number of vacuum and single-photon events in the X basis ($s_{X,0}$ and $s_{X,1}$) are bounded as:

$$s_{X,0} \geq \tau_0 \frac{\mu_2 n_{X,\mu_3}^- - \mu_3 n_{X,\mu_2}^+}{\mu_2 - \mu_3}, \tag{43}$$

$$s_{X,1} \geq \frac{\tau_1 \mu_1 \left[n_{X,\mu_2}^- - n_{X,\mu_3}^+ - \frac{\mu_2^2 - \mu_3^2}{\mu_1^2} (n_{X,\mu_1}^+ - \frac{s_{X,0}}{\tau_0}) \right]}{\mu_1(\mu_2 - \mu_3) - \mu_2^2 + \mu_3^2}. \tag{44}$$

Similar bounds apply to $s_{Z,0}$ and $s_{Z,1}$ in the Z basis.

Error Probability

For a dedicated fiber (basis-independent error probability), the bit error probability for intensity k is:

$$e_k = P_{dc} + e_{\text{mis}}[1 - e^{-\eta_{sys}\mu_k}] + P_{ap} \frac{D_k}{2}, \tag{45}$$

where P_{ap} is the after-pulse probability, and e_{mis} is the misalignment error probability.

The expected number of errors in the Z basis for intensity μ_k is:

$$m_{Z,k} = e_k N_Z P_{\mu_k}, \tag{46}$$

where $N_Z = NP_Z^2$, and the total number of errors in the Z basis is:

$$m_Z = \sum_{k \in \mathcal{K}} m_{Z,k}. \tag{47}$$

Security bounds on the number of errors are:

$$m_{Z,k}^{\pm} = \frac{e^{\mu_k}}{p_k} \left[m_{Z,k} \pm \sqrt{\frac{m_Z}{2} \ln \frac{21}{\epsilon_{\text{sec}}}} \right]. \tag{48}$$

The number of bit errors for single-photon events in the Z basis is bounded by:

$$v_{Z,1} \leq \tau_1 \frac{m_{Z,\mu_2}^{+} - m_{Z,\mu_3}^{-}}{\mu_2 - \mu_3}. \tag{49}$$

Phase Error Rate

The phase error rate for single-photon events in the X basis is:

$$\phi_X = \frac{c_{X,1}}{s_{X,1}} \leq \frac{v_{Z,1}}{s_{Z,1}} + \gamma \left(\epsilon_{\text{sec}}, \frac{v_{Z,1}}{s_{Z,1}}, s_{Z,1}, s_{X,1} \right), \tag{50}$$

where $c_{X,1}$ is the number of phase errors, and the finite-key correction term is:

$$\gamma \left(\epsilon_{\text{sec}}, \frac{v_{Z,1}}{s_{Z,1}}, s_{Z,1}, s_{X,1} \right) = \sqrt{\frac{(s_{Z,1} + s_{X,1})(1 - \frac{v_{Z,1}}{s_{Z,1}}) \frac{v_{Z,1}}{s_{Z,1}}}{s_{Z,1} s_{X,1} \ln 2} \log_2 \left(\frac{s_{Z,1} + s_{X,1}}{s_{Z,1} s_{X,1} (1 - \frac{v_{Z,1}}{s_{Z,1}}) \frac{v_{Z,1}}{s_{Z,1}}} \frac{21^2}{\epsilon_{\text{sec}}^2} \right)}. \tag{51}$$

Error Correction and Observed Error Rate

The observed error rate in the X basis is:

$$e_{\text{obs}} = \frac{m_X}{n_X}. \tag{52}$$

The error correction term is:

$$\lambda_{EC} = n_X f_{EC} h(e_{\text{obs}}), \tag{53}$$

where f_{EC} is the error correction efficiency, and $h(x) = -x \log_2 x - (1 - x) \log_2 (1 - x)$ is the binary entropy function.

Final Secret Key Length

The secret key length is:

$$l = s_{X,0} + s_{X,1} - s_{X,1} h(\phi_X) - \lambda_{EC} - 6 \log_2 \left(\frac{2}{\epsilon_{\text{sec}}} \right) - \log_2 \left(\frac{2}{\epsilon_{\text{cor}}} \right), \tag{54}$$

where ϵ_{cor} is the error correction security parameter. The terms $-6 \log_2 \left(\frac{2}{\epsilon_{\text{sec}}} \right)$ and $- \log_2 \left(\frac{2}{\epsilon_{\text{cor}}} \right)$ account for security overheads.

B. Details in Optimization

This appendix details the technical implementation of optimizing parameters for the BB84 Quantum Key Distribution (QKD) protocol with decoy states, aimed at maximizing the secret key rate over varying fiber lengths and detection event counts. The simulation and optimization follow the analytical model in [1].

Parameter Space

The optimization explores:

- **Fiber Lengths (L)**: 0 to 200 km, with 100 steps (2 km increments), yielding 100 unique values.
- **Detection Events (n_X)**: $[10^4, 10^5, 10^6, 10^7, 10^8, 10^9]$ (6 values), though some runs use subsets (e.g., $[10^{10}]$).
- **Combinations**: Up to $100 \times 6 = 600$ combinations of (L, n_X), varying by experiment.

Optimized parameters are $\vec{p} = [\mu_1, \mu_2, P_{\mu_1}, P_{\mu_2}, P_X]$:

- μ_1, μ_2: Mean photon numbers for two intensity levels.
- P_{μ_1}, P_{μ_2}: Probabilities of selecting μ_1 and μ_2.
- P_X: X-basis probability, with $P_Z = 1 - P_X$.
- Fixed: $\mu_3 = 2 \times 10^{-4}$, $P_{\mu_3} = 1 - P_{\mu_1} - P_{\mu_2}$.

Bounds: $\mu_1 \in [4 \times 10^{-4}, 0.9]$, $\mu_2 \in [2 \times 10^{-4}, 0.5]$, $P_{\mu_1}, P_{\mu_2}, P_X \in [10^{-12}, 1 - 10^{-12}]$.

Fixed system parameters are listed in Table 6:

Table 6: Simulation parameters for the decoy-state BB84 protocol.

Parameter	**Value**	**Description**
α	0.2 dB/km	Fiber attenuation coefficient
η_{Bob}	0.1	Detector efficiency at Bob's side
Y_0	6×10^{-7}	Dark count probability
e_d	5×10^{-3}	Misalignment error rate
P_{ap}	0	After-pulse probability
f_{EC}	1.16	Error correction efficiency
ϵ_{sec}	10^{-10}	Secrecy error tolerance
ϵ_{cor}	10^{-15}	Correlation error tolerance

Optimization Process

The secret key rate $R = l/N$ is maximized using a hybrid approach:

- **Global Search**: `dual_annealing` (SciPy) explores the parameter space, avoiding local optima. Initial guess varies by n_X (e.g., $[0.54, 0.375, 0.16, 0.775, 0.83]$ for $n_X = 10^8$).

- **Local Refinement**: `minimize` with Nelder-Mead refines the solution (`maxiter=10000`, `xatol=1e-10`, `fatol=1e-10`).

The objective function is:

$$\text{Objective} = -R(\vec{p}, L, n_X), \tag{55}$$

computed via custom functions in `QKD_Functions.py`, including channel transmittance (η_{ch}), system efficiency (η_{sys}), detection probabilities (D_k), counts (n_{X,μ_k}), errors (m_{X,μ_k}), bounds ($S_{X,0}$, $S_{X,1}$, $v_{Z,1}$), and final key rate.

Validation against [1] shows close agreement (e.g., $R = 1.23 \times 10^{-3}$ vs. 1.25×10^{-3} at $L = 0\,\text{km}$, $n_X = 10^9$).

Parallelization

Optimization is parallelized using `joblib.Parallel` and `concurrent.futures.ThreadPoolExecutor` (12 threads), processing batches (size 16) with progress tracked via `tqdm-joblib`.

Framework Selection: JAX

JAX is used for its efficiency in numerical optimization:

- **Features**: JIT compilation, automatic differentiation (`grad`), vectorization (`vmap`).
- **Configuration**: Double precision (`jax_enable_x64=True`).

Dataset Generation

Results are stored in a JSON dataset:

- **Features**: $e_1 = L/100$, $e_2 = -\log_{10}(Y_0)$, $e_3 = e_d \times 100$, $e_4 = \log_{10}(n_X)$, key rate R, optimized parameters $\vec{p}_{opt}$.
- **Format**: Grouped by n_X, sorted by L (e.g., `qkd_grouped_dataset_YYYYMMDD_HHMMSS.json`).
- **Size**: Varies (e.g., 600 entries for 100 L and 6 n_X).

The codebase is available at https://gits-15.sys.kth.se/slleung/QKD_KeyRate_ParameterOptimization/blob/main/Analysis/BB84_Parameters_2014_Analysis_Jax.ipynb.

C. Details in Neural Network

This appendix details the technical implementation of a neural network (NN) to predict optimized parameters for the BB84 Quantum Key Distribution (QKD) protocol with decoy states, based on input features derived from fiber length and detection events. The goal is to approximate the parameter optimization process described in Appendix B.

Data Preparation

The training data is sourced from `cleaned_combined_datasets.json`, containing entries grouped by n_X (detection events, 10^4 to 10^9):

- **Input Features (X)**: $[e_1, e_2, e_3, e_4]$, where $e_1 = L/100$, $e_2 = -\log_{10}(6\times10^{-7})$, $e_3 = 5\times10^{-3}\times100$, $e_4 = \log_{10}(n_X)$.
- **Target Outputs (Y)**: Optimized parameters $[\mu_1, \mu_2, P_{\mu_1}, P_{\mu_2}, P_X]$ from prior optimization.
- **Filtering**: Entries with key_rate ≤ 0 or $L > 200$ km are excluded.
- **Scaling**: X is standardized using `StandardScaler`, and Y is scaled to [0, 1] with `MinMaxScaler`. Scalers are saved as `scaler.pkl` and `y_scaler.pkl`.
- **Dataset Split**: 80% training (shuffled), 20% validation, using `torch.utils.data.random_split`.
- **DataLoader**: Batch size 64, shuffled for training, unshuffled for validation.

Evaluation data includes the full dataset and a separate test set for $n_X = 5\times10^8$ from `5e8_100_reordered_qkd_grouped_da`

Neural Network Architecture

The model (`BB84NN`) is a fully connected feedforward neural network implemented in PyTorch:

$$\text{Architecture: } 4 \xrightarrow{\text{ReLU}} 16 \xrightarrow{\text{ReLU}} 32 \xrightarrow{\text{ReLU}} 16 \xrightarrow{\text{Linear}} 5, \tag{56}$$

where:

- Input layer: 4 nodes (for e_1, e_2, e_3, e_4).
- Hidden layers: 16, 32, 16 nodes with ReLU activation.
- Output layer: 5 nodes (for $\mu_1, \mu_2, P_{\mu_1}, P_{\mu_2}, P_X$).

The model is deployed on the Metal Performance Shaders (MPS) backend (`torch.device("mps")`) for GPU acceleration on macOS.

Training Process

Training is conducted over 5000 epochs:

- **Loss Function**: Mean Squared Error (`MSELoss`).
- **Optimizer**: Adam (`optim.Adam`), initial learning rate 10^{-3}.
- **Scheduler**: `ReduceLROnPlateau`, reducing learning rate by 0.1 if validation loss plateaus for 5 epochs, minimum 10^{-6}.
- **Training Loop**: For each epoch, compute training and validation losses, update weights, and adjust learning rate.
- **Metrics**: Losses and learning rates are logged for analysis.

The model is saved as `bb84_nn_model.pth` after the first and last epochs.

Evaluation and Visualization

Evaluation occurs at epochs 1 and 5000, and post-training for $n_X = 5 \times 10^8$:

- **Prediction**: Model outputs scaled parameters, inverse-transformed via `y_scaler`, clipped to physical bounds ($\mu_1, \mu_2 \geq 10^{-6}$, probabilities in [0, 1]).
- **Key Rate Calculation**: Predicted parameters are fed into `objective`, a wrapper for `objective` (Appendix B), returning 0 on NaN or errors.
- **Subplot Visualizations**:
  - **Key Rates**: Log-scaled optimized vs. predicted key rates vs. fiber length for $n_X = 10^4$ to 10^9, in 2x3 subplots, saved as `keyrate_subplots_*.png`.
  - **Parameters**: Optimized vs. predicted parameters vs. fiber length, in 2x3 subplots with a shared legend, saved as `parameters_subplots_*.png`.
- **Final Test**: For $n_X = 5 \times 10^8$, a combined plot of key rates and parameters is generated (`keyrate_parameters_5e8_final_test.png`).
- **Metrics**: Mean Absolute Error (MAE) is computed for parameters and key rates, excluding NaN values.

Loss curves (training and validation) and learning rates are plotted and saved as `loss_plot.png` and `learning_rate_plot.png`.

Implementation Details

- **Libraries**: PyTorch for NN, NumPy for data handling, Scikit-learn for scaling and metrics, Matplotlib (Agg backend) for plotting, Joblib for scaler serialization.
- **Error Handling**: NaN key rates are filtered, with diagnostics printed.
- **Codebase**: Available at https://gits-15.sys.kth.se/slleung/QKD_KeyRate_ParameterOptimization/blob/main/Analysis/BB84_Parameters_2014_Analysis_Jax.ipynb.

References

[1] Charles Ci Wen Lim et al. "Concise security bounds for practical decoy-state quantum key distribution". In: *Physical Review A* 89.2 (2014), p. 022307.

[2] Charles H Bennett and Gilles Brassard. "Quantum cryptography: Public key distribution and coin tossing". In: *Theoretical computer science* 560 (2014), pp. 7–11.

[3] Artur K Ekert. "Quantum cryptography based on Bell's theorem". In: *Physical review letters* 67.6 (1991), p. 661.

[4] Y. Xiang et al. "Simulated Annealing with GPU Acceleration". In: *IEEE Transactions on Parallel and Distributed Systems* 24.8 (2013). Accessed: March 18, 2025, pp. 1568–1577.

[5] C. Tan et al. *Optimization Algorithms for Real-Time Systems*. Accessed: March 18, 2025. Amsterdam, Netherlands: Elsevier, 2018. ISBN: 9780123456789.

[6] Weizhao Lu et al. "Recurrent neural network approach to quantum signal: coherent state restoration for continuous-variable quantum key distribution". In: *Quantum Information Processing* 17 (2018), pp. 1–14.

[7] Weiqi Liu et al. "Integrating machine learning to achieve an automatic parameter prediction for practical continuous-variable quantum key distribution". In: *Physical Review A* 97.2 (2018), p. 022316.

[8] Wenyuan Wang and Hoi-Kwong Lo. "Machine learning for optimal parameter prediction in quantum key distribution". In: *Physical Review A* 100.6 (2019), p. 062334.

[9] Hiking and Coding. "A Cascade Information Reconciliation Tutorial". In: *Hiking and Coding* (Jan. 2020). Accessed: 2025-04-15. URL: https://hikingandcoding.com/2020/01/15/a-cascade-information-reconciliation-tutorial/.

[10] V. Scarani et al. "The Security of Practical Quantum Key Distribution". In: *Reviews of Modern Physics* 81 (2009), pp. 1301–1350.

[11] Chi-Hang Fred Fung, Xiongfeng Ma, and HF Chau. "Practical issues in quantum-key-distribution postprocessing". In: *Physical Review A—Atomic, Molecular, and Optical Physics* 81.1 (2010), p. 012318.

[12] Christoph Roser. *Local Optima Global Optimum*. Illustration. Available at: https://www.allaboutlean.com/ (accessed: March 04, 2025), Licensed under Creative Commons Attribution-ShareAlike 4.0 International License (CC-BY-SA 4.0), https://creativecommons.org/licenses/by-sa/4.0/. July 2018. URL: https://www.allaboutlean.com/.

[13] John A Nelder and Roger Mead. "A simplex method for function minimization". In: *The computer journal* 7.4 (1965), pp. 308–313.

[14] Deep Learning Demystified. "Introduction to Neural Networks - Part 1". In: *Medium* (May 2023). Accessed: 2025-04-15. URL: https://medium.com/deep-learning-demystified/introduction-to-neural-networks-part-1-e13f132c6d7e.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Accessed: March 18, 2025. Cambridge, MA: MIT Press, 2016. ISBN: 9780262035613. URL: http://www.deeplearningbook.org/.

[16] Guangyu Yang and Xiao-Jing Wang. *Artificial neural networks for neuroscientists: A primer*. June 2020. DOI: 10.48550/arXiv.2006.01001.

[17] Renyi Cai and Valerio Scarani. "Finite-Key Analysis for Practical Implementations of Quantum Key Distribution". In: *New Journal of Physics* 11.4 (2009). DOI: 10.1088/1367-2630/11/4/045024. URL: https://doi.org/10.1088/1367-2630/11/4/045024.

[18] SciPy Developers. *SciPy Documentation: dual_annealing*. Accessed: March 18, 2025. 2023. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.dual_annealing.html.

[19] Lester Ingber. "Simulated Annealing: Practice versus Theory". In: *Mathematical and Computer Modelling* 18.11 (1993). Accessed: March 18, 2025, pp. 29–57.

[20] Apple Inc. *Apple Silicon Overview*. Accessed: March 18, 2025. 2022. URL: https://www.apple.com/silicon-overview/.

[21] TensorFlow Team. *TensorFlow Lite on ARM*. Accessed: March 18, 2025. 2023. URL: https://www.tensorflow.org/lite/guide.

[22] Apple Inc. *Metal Programming Guide*. Accessed: March 18, 2025. 2022. URL: https://developer.apple.com/metal/.