ACADEMIA QUANTUM

Research Article
Published: 2025-05-09
https://doi.org/10.20935/AcadQuant7692

Reducing single-qubit gate complexity using machine-learned microwave pulses

Jaden Nola[1], Uriah Sanchez[2], Anusha Krishna Murthy[3,*], Elizabeth Behrman[1], James Steck[4]

Academic Editor: Randy Kuang

Abstract

A gate sequence of single-qubit transformations may be condensed into a single microwave pulse that maps a qubit from an initialized state directly into the desired state of the composite transformation. Here, machine learning is used to learn the parameterized values for a single driving pulse associated with a transformation of three sequential gate operations on a qubit. This implies that future quantum circuits may contain roughly a third of the number of single-qubit operations performed, greatly reducing the problems of noise and decoherence. There is a potential for even greater condensation and efficiency using the methods of quantum machine learning.

Keywords: *quantum machine learning, quantum computing, quantum circuit, error reduction, microwave pulses*

Citation: Nola J, Sanchez U, Krishna Murthy A, Behrman E, Steck J. Reducing single-qubit gate complexity using machine-learned microwave pulses. *Academia Quantum* 2025;2. https://doi.org/10.20935/AcadQuant7692

1. Introduction and background

Quantum computing is a field of growing interest with enormous potential; however, the current state of the art, called NISQ (noisy intermediate-scale quantum), is limited due to problems with both fidelity and scaling. Methods for error correction have been devised, but they require a rapidly growing number of additional qubits, called ancilla qubits, so that a smaller and smaller percentage of qubits can be used for the desired computation.

Here, we will provide a brief overview of quantum computing concepts. The pure state of a qubit, or quantum bit, can be written on the charge basis $\{|0\rangle, |1\rangle\}$ as

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle \tag{1}$$

where $0 < \theta < \pi$ and $0 < \varphi < 2\pi$ are angles on the Bloch sphere (see **Figure 1**).

Single-qubit operators move the qubit state from point to point on the Bloch sphere. Operators are called "gates" in analogy with classical logic gates, but in quantum computing, there are operators that cannot be expressed in terms of classical logic gates. For example, a Hadamard or H operator acting on a qubit in the state $|0\rangle$ or $|1\rangle$ results in a combined state of both $|0\rangle$ and $|1\rangle$ called a *superposition*, which is impossible for classical bits to obtain. Superpositions are located on the nonpolar regions of the Bloch sphere. Operators can be represented on a given basis by a matrix, which transforms or maps one state onto another. For example, the Hadamard operator acting on the state $|0\rangle$ can be written as a matrix equation on the charge basis, as shown in Equation (2):

$$H|0\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} = |+\rangle \tag{2}$$

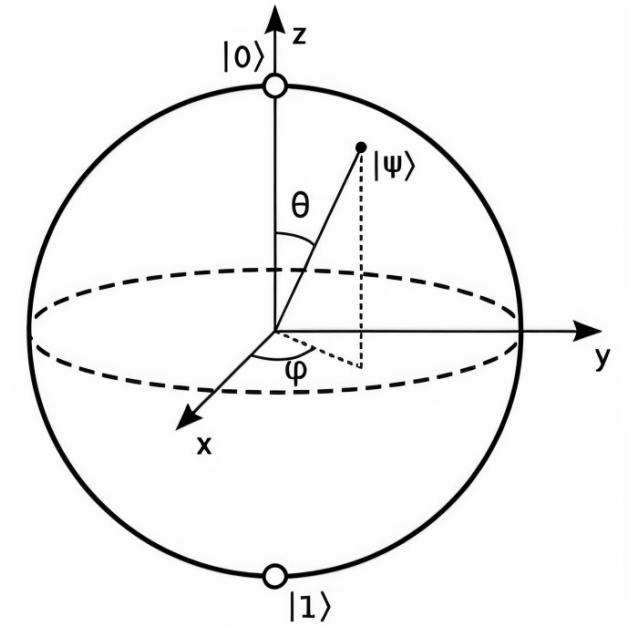

Figure 1 • The Bloch sphere, whose poles correspond to the two charge basis states $|0\rangle$ and $|1\rangle$ and whose nonpolar regions represent superposition states. The surface of the sphere represents all possible pure states, given by Equation (1) above. For example, the state $|0\rangle$ is the point at the top of the depicted sphere with the angles $\theta = \varphi = 0$; the equator represents any state in an equal superposition of $|0\rangle$ and $|1\rangle$, with $\theta = \pi/2$.

[1]Department of Mathematics, Statistics, and Physics, Wichita State University, Wichita, KS 67260, USA.
[2]Department of Biomedical Engineering, Wichita State University, Wichita, KS 67260, USA.
[3]Department of Electrical Engineering and Computer Science, Wichita State University, Wichita, KS 67260, USA.
[4]Department of Aerospace Engineering, Wichita State University, Wichita, KS 67260, USA.
*email: axkrishnamurthy@shockers.wichita.edu

All quantum gates are unitary operators, which conserve probability. In general, then, any single-qubit gate is a mapping from one point on the Bloch sphere to another, which is conventionally written as a product of rotations around the x and z axes:

$$U(\Delta\vartheta, \Delta\chi, \Delta\lambda) = R_z(\Delta\vartheta)R_x(\Delta\chi)R_z(\Delta\lambda) \tag{3}$$

This generalized rotation simplifies to the single matrix

$$\begin{pmatrix} \cos\left(\frac{\Delta\vartheta}{2}\right) & -e^{i\Delta\lambda}\sin\left(\frac{\Delta\vartheta}{2}\right) \\ e^{i\Delta\chi}\sin\left(\frac{\Delta\vartheta}{2}\right) & e^{i(\Delta\chi+\Delta\lambda)}\cos\left(\frac{\Delta\vartheta}{2}\right) \end{pmatrix} \tag{4}$$

where $\Delta\lambda$, $\Delta\chi$, and $\Delta\vartheta$ are the angles of the rotations in Equation (3). For example, $U(\frac{\pi}{2}, 0, \pi)$ is the Hadamard gate (H); alternatively, $U(0, \pi, \Delta\lambda)$ is a rotation about the z-axis of the Bloch sphere by the angle $\Delta\lambda$, which simplifies to the $R_z(\Delta\lambda)$ gate. For several decades now, we have been exploring an alternative approach that has been demonstrated to be more robust to noise and decoherence: quantum machine learning (QML), rather than algorithm programming. In QML, a quantum system is used as a quantum computer that *learns* how to perform a particular task when provided with an example dataset. In recent years, QML has emerged as a powerful tool for optimizing complex quantum systems. QML algorithms leverage the principles of quantum computing to process quantum data more efficiently than classical machine learning methods. Cerezo et al. (2022) explore the intersection of quantum computing and machine learning, with a focus on the advantages of QML in accelerating data analysis, particularly with quantum data [1]. Various QML models, such as parameterized quantum circuits (PQCs) and quantum neural networks (QNNs), are discussed while highlighting the potential for achieving quantum advantage in certain machine learning tasks, such as classification, optimization, and clustering. The study also sheds light on the challenges in QML, such as the trainability of quantum models, noise and decoherence in quantum hardware, and the need for efficient quantum algorithms. Additionally, [2] presents a theoretical framework and initial algorithms that demonstrate the significance for quantum speedup in machine learning. Furthermore, research has been conducted on combining discrete gate-based and continuous pulse-based paradigms, producing a hybrid model for variational quantum algorithms (VQAs) [3]. The core objective has been to improve the efficiency and accuracy of VQAs.

Research on QML has also been conducted at the *pulse*-level rather than the *gate*-level. However, first, we will discuss the implementation of gates as pulses on physical hardware. Quantum states for superconductor qubits are controlled by microwave pulses characterized by a time-varying voltage, $V_d(t)$, with specific parameters that result in a desired quantum transformation [4] (see **Figure 2**).

In this paper, we use supervised machine learning to train these sub-gate pulses, which requires OpenPulse permissions on the hardware to modify the microwave pulses. Such permissions are uncommon on open-source quantum hardware platforms, as most organizations with quantum hardware optimize and then fix the microwave pulse control frequency to match the resonant frequency of their systems. This fixed-frequency approach, as shown in **Figure 3**, based on IBM Research Blog's post [5], explains why pulse-level alterations by users are generally not permitted. However, IBM's *ibmq_armonk* device is an exception, allowing users to implement pulse-level commands, enabling greater flexibility in pulse-level optimization.

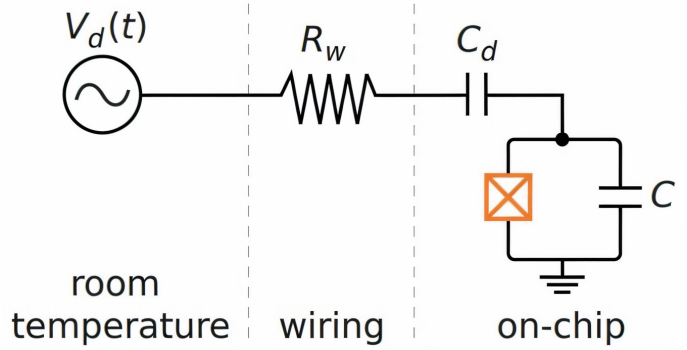

Figure 2 • A circuit diagram for a transmon qubit attached to a microwave drive line [4].

a)

```
# Select the pulse-enabled Armonk device
backend = provider.get_backend('ibmq_armonk')
# create a new pulse schedule and give it a name
schedule = pulse.Schedule(name='Frequency sweep')
# apply the drive pulse on the qubit's drive channel
schedule += drive_pulse(pulse.DriveChannel(0))
# apply a measurement pulse on the qubit's measurement channel
measure_schedule = meas_pulse(pulse.MeasureChannel(0))
# Trigger acquisition, and store results into memory slots
measure_schedule += acq_cmd([pulse.AcquireChannel(0)],
                            [pulse.MemorySlot(0)])
# The left shift `<<` means shift start of the schedule
schedule += measure_schedule << schedule.duration
# Create the frequency settings for the sweep
schedule_frequencies = [{drive_chan: freq} for freq in freq_list]

num_shots_per_frequency = 1024
job = execute(schedule, backend=backend,
              meas_level=1, meas_return='avg',
              shots=num_shots_per_frequency,
              schedule_los=schedule_frequencies)
```

b)

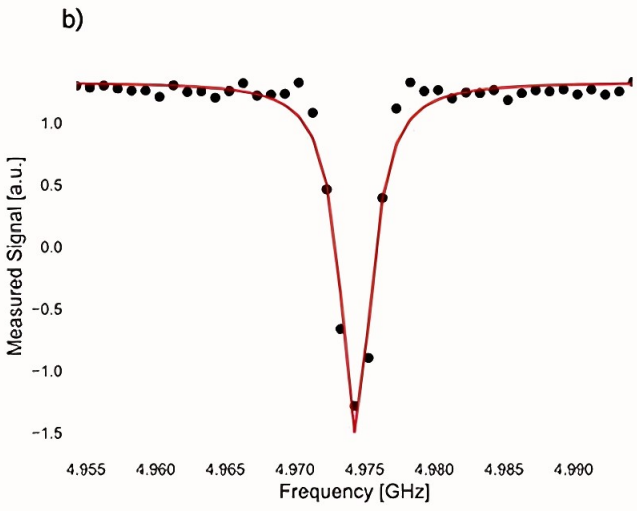

Figure 3 • Combined figure showing both the job execution and resonance sweep. (**a**) Commented code associated with running a job on IBM's ibmq_armonk device. (**b**) A graphical display of signal strength as a function of frequency to find the optimized frequency of a single qubit in the setup process [5].

The Qiskit Pulse framework [6, 7] is a Python library designed to simulate and execute pulses on quantum hardware. In this work, we used it to implement the trained pulses for single-qubit operations. The framework facilitates the conversions of quantum circuits from the gate-level to pulse-level commands, incorporating the physical details of implementation. Most quantum gates are realized through a combination of DRAG pulses (briefly discussed

later) and phase shifts in the drive channel, commonly referred to as "virtual Z-gates" [8]. However, our focus is on leveraging Qiskit Pulse to access and customize the SQUID microwave control pulses using parameters of our own choosing.

IBM's quantum hardware backends exhibit varying specifications, including time resolution and sampling rates, which are not always explicitly uniform or published across all devices. Fortunately, these details can be accessed through the backend configuration available in IBM's Qiskit platform. For our implementation, we specifically considered the time resolution, denoted as dt, which is unique to the backend we utilized.

To ensure high measurement fidelity, the duration of the readout pulse must exceed the time resolution. Given the small dt, readout pulses typically last hundreds of nanoseconds, translating to over 20,000 dt intervals. This ensures sufficient temporal coverage for accurate quantum state measurements.

In Qiskit Pulse, each pulse is represented by n discrete, complex-valued samples, $d_j, j \in \{0, 1, \ldots, n-1\}$. These samples define the shape of the microwave pulses applied to the qubit during execution. Each d_j corresponds to the value used for one timing cycle (timestep) of the quantum processor. In the jth timestep, the ideal signal applied to the drive channel, which relays the signal to the qubit [9], is

$$D_j = \mathrm{Re}\left[e^{2\pi f j \mathrm{d}t + \phi} d_j\right] \tag{5}$$

where f is the modulation frequency and ϕ is the phase. Both f and ϕ are properties of the drive channel and can be controlled via instructions provided by the user through the Qiskit Pulse interface. For the remainder of this work, "training a pulse" refers to the process of using machine learning techniques to optimize the complex values d_j, which define the microwave pulse shape. These values are iteratively trained to improve the accuracy and performance of the quantum operations, ensuring that the pulse closely matches the desired qubit control behavior.

Derivative errors can arise due to the inherent characteristics of both the pulse and the qubit, specifically from extra oscillations in the pulse. The derivative removal by adiabatic Gaussian (DRAG) pulse technique is an effective method for mitigating these errors [10]. This technique modifies the shape of the microwave pulse by combining two distinct pulses: a standard Gaussian pulse and a derivative pulse carefully designed to cancel the effects of derivative errors. The signal of d_j in pulse generation is primarily determined by the settings of the Gaussian wave, which plays a crucial role in shaping the overall pulse. A full description of the pulse can be given by four parameters: duration, amplitude, variance, and correction amplitude.

Prior work on pulse learning can be found in the development of variational quantum pulse (VQP) learning [11], where the authors proposed directly training quantum pulses for machine learning tasks. The research focused on evaluating the performance of VQP learning on binary classification tasks and the Modified National Institute of Standards and Technology (MNSIT) dataset.

Melo et al. (2023) investigated the impact of pulse-efficient transpilation using a near-term QML algorithm and cross-resonance-based hardware, aiming to improve the performance and fidelity of quantum circuits by reducing circuit schedule duration and mitigating the effects of device noise [12]. Additionally, they introduced an approach to enhance the efficiency of QML through pulse-efficient techniques. While their study broadly focused on pulse efficiency across various QML tasks, our research takes a more specialized approach by targeting the training of microwave pulses for single-qubit transformations. Furthermore, we emphasize the condensation of gate sequences into a single microwave pulse, potentially simplifying the implementation of specific qubit transformations.

Meitei et al. (2021) proposed an alternative algorithm called ctrl-VQE for state preparation using the variational quantum eigensolver (VQE) algorithm [13]. While the results of this research showed a reduction in state preparation times of roughly three orders of magnitude compared to gate-based strategies, there are challenges such as hardware limitations, optimization complexity, experimental implementation, scalability to larger systems, etc., which need further investigation. Schlud (2021) presented the relationship between supervised quantum machine learning models and kernel methods, highlighting the mathematical similarities and implications for quantum models [14].

Additionally, Rongxin and Sabre (2018) presented a hybrid quantum algorithm that employs a restricted Boltzmann machine in order to obtain molecular potential energy surfaces that are accurate [15]. However, the main challenges lie in resource requirements. The simulations for the molecules used in their study, (H_2, L_iH, and H_2O) required 13 qubits each, leading to the simulation of simple molecules requiring a significant number of qubits. Further, noise management, scalability, implementation complexity, et cetera, are some practical challenges that need to be addressed.

Gianani et al. (2021) explored the idea of embedding classical data into a quantum state, which can then be manipulated in a larger Hilbert space [16]. Optimization is carried out using gradient descent, employing automatic differential tools available in platforms such as PennyLane. These differential experimental platforms, however, come with unique constraints and challenges.

In previous work, we have shown that a quantum system, adapted with machine learning and analogous to a neural network, can design its own algorithm, using the mathematical isomorphism between the time evolution of a quantum state and the information flow in a neural network [17]. The isomorphism is as follows. A neural network consists of layers of nodes, with a typical feed-forward structure being one input layer, one or more hidden layers, and one output layer (see **Figure 4**). We can think of this diagram of the information flow as analogous to time evolution by visualizing the horizontal dimension as "time", where, naturally, the system's state at any given time is causally connected to its state at previous times. The vertical dimension represents space; different parts of the system are connected to each other.

In neural networks, the process by which a node (or neuron) produces an output based on its input and internal parameters is called "activating" the node. Quantum computing, however, does not have a direct one-on-one equivalent to the concept of "activation", but some analogous processes are available. We consider the key concepts and components of neural network activation and how they might map to quantum computing. Each node connects to every node of the next layer; each node computes a weighted sum of its inputs, in a neural network, which is then added to the bias term. This process is then followed up with passing the weighted sum through an activation function which helps determine the output of the node. In quantum computing, quantum algorithms make use of quantum gates for qubit manipulation. Quantum gates perform

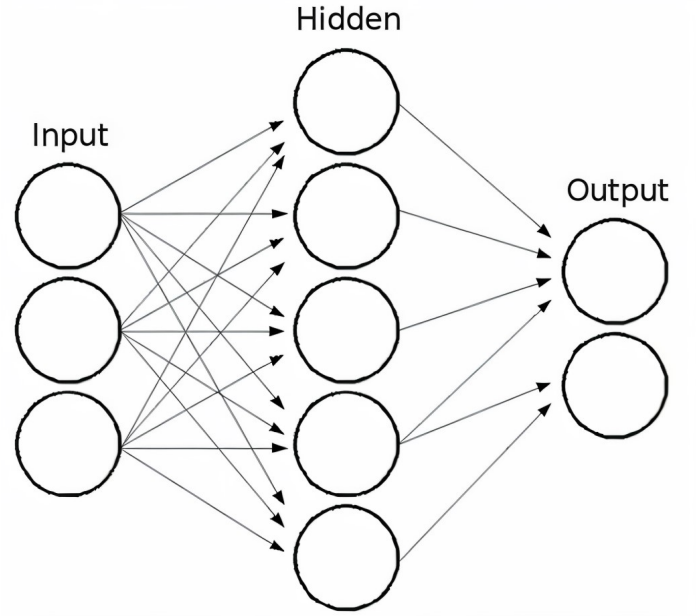

Figure 4 • A diagram of a general neural network. The structure consists of an input layer where each circle represents a simple processing unit—a "node" or "neuron". Each node in the input layer is connected to subsequent nodes in the "hidden layers" (here, there is only one hidden layer), finally connecting to the output layer on the right. In our current work, qubits represent the nodes, and the horizontal axis is time [18].

operations that alter the state of the qubits, similar to the activation function of a neural network. Similarly, in VQAs, rotational gates are parameterized akin to the activation function.

Neural networks operate by minimizing a cost function (performance index) which quantifies the deviation of the network's output from the desired output. The weights and biases are adjusted to minimize the cost function, analogous to tuning the adjustable parameters in the Hamiltonian of a quantum system. A single pass through the training dataset, consisting of inputs and corresponding expected outputs, is referred to as an *epoch*.

In our quantum system, the input and output dimensions depend on the number of qubits involved in the intended transformation. Specifically, for n qubits, these dimensions are both equal to 2^n. Since our current work focuses on a single qubit, we utilize only one hidden layer, which represents a single phase shift preceding a DRAG pulse. For multi-qubit systems, this hidden layer would include pulses driven along the control channel of the qubits. Additionally, one could imagine partitioning a sequence of pulses into "blocks" that execute sequentially, each representing a separate hidden layer.

In earlier work, we proposed an alternative method for designing algorithms by decomposing them into smaller "building blocks" derived from a universal set of gates. In this approach, the system is prepared at an initial time (the "input") and measured at a final time (the "output"), and externally adjustable parameters are trained using machine learning to improve the closeness of the output to the desired values for the intended computation. Beyond the reasons highlighted above regarding the importance of this study, further benefits are outlined below:

1. It bypasses algorithm construction [17].
2. Breaking down the computation is unnecessary [19–21].
3. Scale-up is relatively easy [22].
4. The multi-interconnectivity of the architecture results in robustness to both noise and decoherence [19, 23, 24].

The training process is illustrated in **Figure 5**. According to the Schrödinger equation, a pure quantum state evolves in time according to $|\psi(t)\rangle = e^{\frac{-i\hat{H}t}{\hbar}}|\psi(0)\rangle$. The Hamiltonian $\hat{H}$ governs the temporal evolution of the system's state. The quantum system is initially prepared in a specific state (the block representing the 'Input' in **Figure 5**), undergoes evolution under the Hamiltonian $\hat{H}$, and is subsequently measured (the block representing the 'Output'). The measured output is compared to the desired outcome for the given input, and the Hamiltonian is adjusted to reduce the error (block representing 'Modify Hamiltonian'). This process is iterated until the desired outcome is achieved, thus completing the training.

This approach to programming quantum computers has been remarkably successful in both reproducing calculations performed with other methods [17, 19] and in designing "algorithms" for which there is no known sequence of gates. This approach also has other advantages like scaleup and robustness to noise and decoherence [22, 23]. More recently, the method has been rediscovered by others under the names of the Quantum Approximate Optimization Algorithm (QAOA) [25, 26] and VQE [27, 28]. In some contexts, we are able to show that our method significantly simplifies and streamlines the desired computation by training the "whole thing" as opposed to breaking it down into building blocks [17, 20]. It is natural to ask if we can achieve even greater efficiency by applying machine learning at the sub-gate level.

In this paper, we extend our method to the level of the microwave pulses used to constitute the physical implementations of the quantum logic gates. Success suggests an increased efficiency for the overall computations, which allows the implementation of an entire transformation sequence—represented as the product of its individual gates—with far fewer pulses. Since the error rate of a quantum sequence grows with both the number of pulses and the time of operation, this would result in a lower error rate for a given calculation and, therefore, an increase in the length and depth of the possible computations. This represents yet another advantage to the quantum machine learning approach.

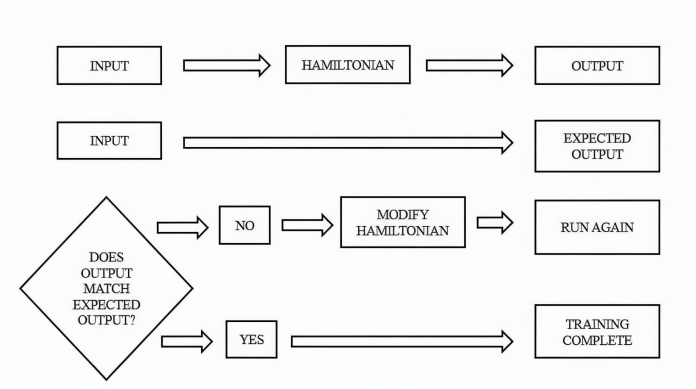

Figure 5 • The process used to reduce final error when training. Training starts with a given input, a desired output, and an adjustable Hamiltonian. The training system evolves in time according to a Hamiltonian, which can be altered to match the system's final state with the desired output.

2. Methodology

We simulated our pulses as if they were being played on IBM's *ibmq_armonk* single-qubit quantum computer. Thus, the sampling rate in our study was ≈ 4.54 GHz, corresponding to a time resolution dt = 0.022 ns. Training (input and target output) data were obtained by generating input states via Equation (1) with 10 random θ and φ pairs. The target outputs were the known resulting states after applying the known gate transformation we wanted the input state to learn. For example, the X and SX gates are well known and are given by Equations (6) and (7), respectively.

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{6}$$

$$SX = \frac{1}{2}\begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix} \tag{7}$$

Note that the SX gate is, up to an overall phase, a rotation around the x-axis by $\frac{\pi}{2}$. As such, we operated on the state vectors of the initial states in the training dataset with these matrices to obtain our desired state after performing these transformations.

In this work, we applied the DRAG pulse technique to optimize a single-qubit operation, following the method outlined by Gambetta et al. [10]. We aimed to train a model to find the duration, amplitude, variance, and correction amplitude of the pulse associated with our desired transformation. An example of such a pulse can be seen in the drive channel D0 of the schedule in **Figure 6**, while the pulse in the measurement channel M0 corresponds to the qubit measurement.

In our implementation, a ShiftPhase instruction preceded the DRAG pulse, and another ShiftPhase instruction followed but with the opposite of the angle, returning the drive phase to the value before adjustment. The ShiftPhase instruction was responsible for modifying the phase of the drive channel, which resulted in an additional phase being added to all pulses played after the instruction. In other words, this instruction modified ϕ in Equation (5). A ShiftPhase instruction was not necessary for training most of the gates shown in the Results Section below—such as the Pauli X and $\sqrt{X}$ gates (henceforth referred to as X and SX gates)—however, the Hadamard gate, H, did require this additional trained parameter.

Since the amplitude parameter could take on complex values, we trained it as two separate real values: the signed modulus, r, and the argument, α. We used the signed modulus instead of the traditional non-negative modulus to prevent discontinuities in the model. Furthermore, Qiskit Pulse prevents users from simulating a DRAG pulse whose magnitude is greater than one. Thus, instead of training the actual signed modulus, we trained an input into Equation (8), which mapped our signed modulus from the intervals $(-\infty, \infty)$ onto $(-1, 1)$.

$$\sigma^*(x) = \frac{1-e^x}{1+e^x} \tag{8}$$

In this work, we defined the effective signed modulus, r_e, and calculated the complex amplitude parameter according to the formula $r_e e^{i\alpha} = \sigma^*(r)e^{i\alpha}$. If r_e was positive, this form followed the standard exponential form for complex numbers. If r_e was negative, this could instead be rewritten as $-r_e e^{i(\alpha+\pi)}$ in order to follow the standard convention. Thus, we were required to train the following six parameters: the duration, signed modulus, argument, variance, correction amplitude, and phase.

We implemented our quantum machine learning (QML) using the Python libraries of Qiskit Dynamics (a supplementary library to Qiskit [6]) and PyTorch [29], with a single hidden layer. This hidden layer contained the six parameters described above and was the transformation defined as the qubit receiving a DRAG pulse with these parameters. We determined the final state of the qubit by using the pulse simulator from Qiskit Dynamics, which used information about the physical information about the qubits.

Our cost function for the machine learning algorithm was the infidelity (or error) between the target (σ) and actual output (ρ) density matrices given by

$$\text{Error} = 1 - F(\rho,\sigma) = 1 - \left(\text{tr}\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}}\right)^2 \tag{9}$$

where tr is the trace operator. A generic density matrix is defined by

$$\sigma = \sum_i p_i |\psi_i\rangle \langle\psi_i| \tag{10}$$

where the sum is over all pure states $|\psi_i\rangle$ on the basis, and p_i is the classical probabilities of each of the outer products so that $\sum_i p_i = 1$. Since we were only working with pure states, we could calculate the density matrix with $\rho = |\psi\rangle\langle\psi|$. Additionally, we had $\rho^2 = \rho$ and $\text{tr}(\rho^2) = 1$.

It should be noted that fidelity has a range of zero to one, which is consistent with our definition of infidelity. *Fidelity* is a measure of the closeness of the final state to the desired state, so we wished to minimize the *infidelity* (our cost function).

We began with a fresh model and random values for each parameter in the following ranges:

$$\begin{aligned} \text{phase shift}: &\ 0 \pm \pi/4 \\ r: &\ 1.73 \pm 0.7 \\ \alpha: &\ 0 \pm \pi/4 \\ \text{variance } (\sigma): &\ 80 \pm 10 \\ \text{correction amplitudes } (\beta): &\ 0.7 \pm 0.1 \\ \text{duration}: &\ 65 \pm 15 \end{aligned}$$

Note that the duration was in units of timesteps (dt) of the physical quantum computer.

We then optimized these parameters to produce a pulse that would yield our desired transformation using **stochastic gradient descent with momentum** [30], which was implemented for us by PyTorch. The gradient was also calculated by PyTorch using automatic differentiation [29], or also known as its autograd mechanism. It kept track of the gradients of operations performed on

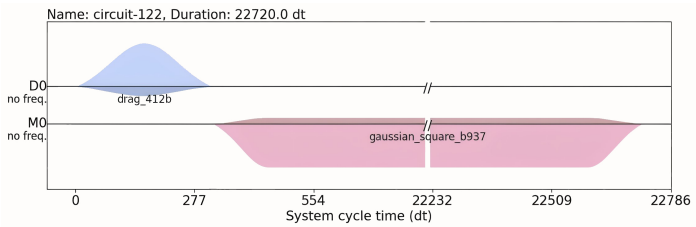

Figure 6 • Pulses corresponding to an SX-gate and measurement. The symbol "D0" refers to the drive channel and "M0" refers to the measurement channel, both of which operate on the first qubit. This image was generated using the Python library Qiskit [6].

the input tensors between modules (layers) and computed the gradient via the chain rule. In order to hook into the autograd mechanism, we implemented the application of the pulse as an `autograd.Function` child class and calculated the gradient using a first-order divided difference (i.e., simulating slightly shifted pulses multiple times).

The gradients were calculated per input state in the training dataset and then averaged and used in the gradient descent with momentum algorithm. This was to prevent overfitting on certain edge cases that may have appeared in the training dataset. This constituted an epoch, and the process repeated until the final epoch.

3. Results and discussion

We first trained some commonly used basis gates (*X*, *SX*, and *H*). The training was conducted in simulations using the Qiskit Dynamics [31, 32] simulator with *ibmq_armonk* as a model. The training was rapid and complete for all single-qubit gates attempted. Each job submitted for training consisted of 100 epochs. The gates trained were those listed in the first column of **Table 1** at the top of the next page. The final error and the resulting pulse parameters for each gate are listed in the remaining columns of **Table 1**. The outcome of the gate parameter training was the successful reduction in errors to the values of the order of 10^{-3} to 10^{-4} (as per Equation (9)) for the trained gates.

Next, we trained a "combination" gate, an example of the general single-qubit gate in Equation (3). In previous work [21], we showed that an entanglement witness trained as a time-dependent Hamiltonian [17] could be rewritten as a sequence of gates, consisting of three rotations on a single qubit interspersed with a CNOT. We used this as an example to demonstrate consolidation. The training of this gate is shown in the last entry in **Table 1**: 3-Rot $= R_y(-w_{5k+1})R_z(w_{5k+1})R_y(w_{5k+1})$, where R_y and R_z represent rotations about the y and z axes of the Bloch sphere, respectively, and the rotation angle of w_{5k+1} is determined through qubit biases and their tunneling amplitudes.

Validity was again measured using Equation (9). Infidelity for this composite transformation gate was readily trained down to 3.702×10^{-3} (0.9963 fidelity). This showed that our training methods could drive a qubit with one control pulse and produce an effect that would traditionally require a plurality of pulses. This allowed for the condensation of lengthy quantum circuits at little cost to error as well as the expedience of these circuits, as a single pulse is much faster and less error-prone than a multiplicity of pulses. The training optimized microwave pulses for more efficient and accurate single-qubit operation in quantum computing, as shown in the obtained results in **Table 1**.

In addition to training on our simulator, for experimental verification, we tested gate identities on *ibmq_armonk*, a small-scale superconducting quantum processor that is accessible through the IBM Cloud, and compared the measurement outputs for well-known gate compositions. Specifically, we replaced one of the gates in the composition with our trained gate and compared the measurement outputs of running that gate composition with the composition with native gates on the computer. Note that we set the optimization level for the Qiskit circuit/schedule transpiler to zero to ensure that gate compositions were not optimized away into the single gate that they composed. We used the default 1024 shots for measurement as defined by the Qiskit API.

Internally, to compose our pulse with other gates in Qiskit, we implemented it as a *pulse gate*. In other words, we defined a new gate to be a "black box" transformation with the action of playing a pulse with our calculated parameter. Side effects of this implementation included the inability to transpile our quantum circuits since we combined both gate- and pulse-level operations. However, since we only worked with small circuits in this work, this did not impose a problem.

We verified our pulse by decomposing the Hadamard gate into an S-SX-S composition and comparing our trained *SX*-gate pulse parameters to the native *SX*-gate implemented on the quantum processor. The resulting probabilities from our trained pulse are shown in **Figure 7**. These probabilities are nearly identical to the same gate composition, but with a native *SX* gate on the computer, which gives us confidence in having trained a pulse with at most a difference in the global phase. **Figure 8** shows a representative error vs. epoch plot for *X* and *SX* training.

The physical implementation of the transformations was challenging due to the limited availability of accredited quantum computers that support OpenPulse—an open-source framework developed by IBM for designing and implementing custom quantum gate

Table 1 • Trained single-qubit gates showing pulse parameters (duration, modulus, argument, variance, etc.) and resulting infidelity.

Gate	Duration (ns)	Signed modulus (arb.)	Effective modulus (arb.)	Argument (rad)	Variance (ns)	Correction amp. (arb.)	Phase (rad)	Infidelity
X	64.08	3.280	0.9275	−0.5188	87.36	−0.8577	0.0690	2.434×10^{-3}
SX	77.29	−1.167	−0.5254	0.3283	70.58	−1.416	−0.2190	1.097×10^{-3}
H	61.09	0.2154	0.1073	−0.5127	86.51	−1.294	2.010	2.360×10^{-4}
R_z	55.23	1.070	0.4888	−0.0253	70.86	0.7921	0.5244	1.521×10^{-3}
R_y	67.26	2.606	0.8625	−0.3393	72.96	0.7858	−0.4742	3.160×10^{-3}
R_x	63.37	1.204	0.5384	−0.2262	72.22	0.6728	−0.3898	4.744×10^{-3}
U_R_x	71.61	2.601	0.8618	0.7665	88.13	0.7125	0.3212	8.902×10^{-3}
U_R_y	51.87	2.977	0.9031	−0.6832	70.66	0.7951	−1.3309	2.757×10^{-3}
U_R_z	76.80	1.987	0.7589	0.3217	84.30	0.7418	0.1389	4.984×10^{-3}
3-Rot	79.77	2.491	0.8470	−0.2740	77.36	0.6690	0.3204	3.702×10^{-3}

pulses. While OpenPulse offers extensive control over quantum hardware, it is also complex, has a steep learning curve for new users, is error-prone, hardware-specific, and resource-intensive. In our future work, we plan to explore alternatives such as IBM Quantum Experience, Qiskit, and the Quantum Scientific Computing Open User Testbed (QSCOUT) by Sandia National Laboratories to compare and highlight different approaches to the physical implementation of these transformations. The system associated with IBM's Quantum Lab that we used, `ibmq_armonk`, was retired during the neural network training [33].

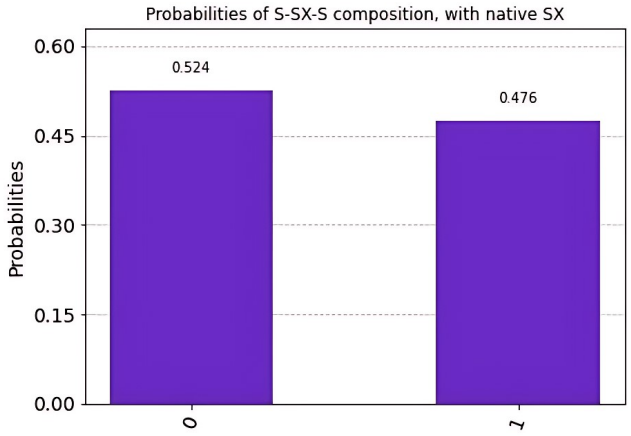

Figure 7 • A bar graph of probabilities from an S-SX-S composition using 1024 shots, with S being native gates on the computer and SX being our trained pulse [18].

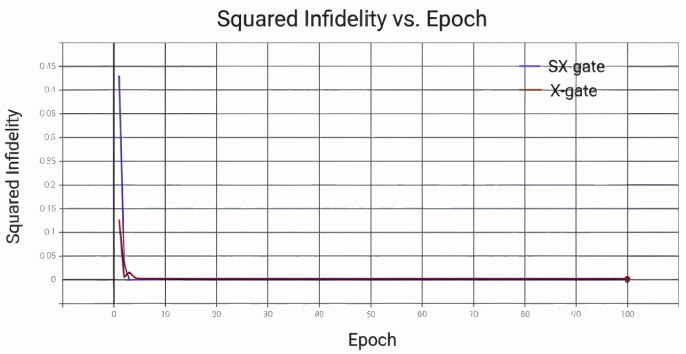

Figure 8 • Error as a function of epoch for the SX and X gates [18].

4. Future work

In this study, we focused on the implementation and evaluation of the DRAG pulse technique for single-qubit control using the Qiskit Pulse framework. While our initial results demonstrate promising improvements in gate fidelity and error reduction, a comprehensive comparison with other baseline methods, such as standard Gaussian pulses, square pulses, and optimal control pulses, remains part of our future work. Gate and measurement fidelity, leakage errors, and coherence time will be the key performance metrics that will be considered for comparison.

Further, our future work will also focus on using IBM Quantum Eagle—a 127-qubit processor—and IBM Quantum Heron—a 133-qubit processor. As the developmental trends change in terms of having more reliable and powerful quantum computers that are open-source, our experiments and findings shall also progress alongside it, providing us with ample opportunities to explore, research, and implement our approach further. Ban et al. (2023) proposed that a QNN with multi-qubit interactions could improve the scalability of QNNs and minimize network depth without affecting the approximation power [34]. The weight of the multi-qubit interaction term in the neural potential was updated using standard gradient descent. The QNN was trained to search prime numbers using a set of $2n_b$ pairs for n_b bits, where n_b is the number of bits. To improve the research further, the scalability of QNNs with multi-qubit interactions can be investigated, and potential challenges associated with the study can be addressed. Additionally, specialized training algorithms tailored to QNNs with multi-qubit interactions can be explored and developed.

While traditional optimization techniques such as GRAPE, Krotov's method, and direct search approaches provide effective solutions for pulse-level optimization, they often require precise analytical gradients and can be computationally expensive for high-dimensional systems. In contrast, our machine learning-based approach minimizes infidelity in a data-driven manner, reducing the need for exact gradients and offering greater flexibility in handling hardware imperfections. Furthermore, by condensing multiple single-qubit operations into a single pulse, our method minimizes gate decomposition errors and execution time.

Future work will also focus on a comparative analysis of our approach with GRAPE, Krotov's method, and direct search techniques to evaluate their relative performance, scalability, and robustness, particularly in the context of multi-qubit operations.

Further, training microwave pulses using machine learning may introduce computational overhead due to the iterative optimization process. In our current study, pulse training was conducted in a controlled simulation environment with a focus on improving pulse efficiency for single-qubit operations.

As future work, we aim to explore the feasibility of adapting this approach for offline training to pre-compute optimized pulses that can be deployed in real time on quantum hardware. Additionally, further investigation into the time complexity and scalability of this approach for multi-qubit systems is warranted to assess its practicality in real-time quantum operations.

5. Conclusions

Quantum learning can be applied at the algorithm level [17] and the gate level [19], as demonstrated in our previous work. In this study, we extended our method to include the pulse level, leading to a reduction in the number of operations required to achieve certain states. This reduction minimizes the errors associated with executing multiple operations and shortens the duration needed to manipulate a qubit into the desired state, as each operation requires a microwave pulse of a specific duration to drive the qubit.

Through our training methods, the time required to execute single-qubit operations can be significantly reduced, paving the way for more efficient quantum operations. However, we acknowledge that extending this approach to multi-qubit systems introduces additional challenges, such as cross-talk, increased noise, and the complexity of controlling multiple qubits simultaneously. Future

work will explore the feasibility of condensing multi-qubit operations using similar pulse-level optimization techniques. Specifically, we intend to investigate the application of this approach to two-qubit systems and assess its scalability on platforms such as IBM Quantum Eagle and IBM Quantum Heron, which offer enhanced multi-qubit capabilities.

Acknowledgments

We thank the research group for their invaluable discussions in developing this project.

Funding

This research was not funded by any grant. Two of the students were partially supported by Wichita State University through the Undergraduate Research Experience (U. Sanchez) and First-Year Research Experience (J. Nola).

Author contributions

Conceptualization and methodology, E.B. and J.S.; software, J.N. and U.S.; validation, formal analysis, writing, J.N., U.S., A.K.M., E.B. and J.S. All authors have read and agreed to the published version of the manuscript.

Conflict of interest

The authors declare no conflicts of interest.

Data availability statement

Data supporting these findings are available at https://github.com/mergemoveagree/pulselearn

Institutional review board statement

Not applicable.

Informed consent statement

Not applicable.

Additional information

Received: 2025-01-29

Accepted: 2025-04-22

Published: 2025-05-09

Academia Quantum papers should be cited as *Academia Quantum 2025*, ISSN 3064-979X, https://doi.org/10.20935/AcadQuant7692. The journal's official abbreviation is *Acad. Quant.*

Publisher's note

Copyright

References

1. Cerezo M, Verdon G, Huang H, Cincio L, Coles PJ. Challenges and opportunities in quantum machine learning. Nat Comput Sci. 2022;2:567. doi: 10.1038/s43588-022-00311-3
2. Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N, Lloyd S. Quantum machine learning. *Nature*. 2017;549:195–202. doi: 10.1038/nature23474
3. Liang Z, Song Z, Cheng J, He Z, Liu J, Wang H, et al. Hybrid gate-pulse model for variational quantum algorithms. arXiv. 2022;arXiv:2212.00661. doi: 10.48550/arXiv.2212.00661
4. Krantz P, Kjaergaard M, Yan F, Orlando TP, Gustavsson S, Oliver WD. A quantum engineer's guide to superconducting qubits. Appl Phys Rev. 2019;6(2):021318. doi: 10.1063/1.5089550
5. Asfaw A, Alexander T, Nation P, Gambetta J. Get to the heart of real quantum hardware. 2019 [cited 2025 Apr 20]. Available from: https://www.ibm.com/blogs/research/2019/12/qiskitopenpulse/
6. IBM. IBM Quantum Documentation — Qiskit Pulse Framework. [cited 2025 Apr 20]. Available from: https://docs.quantum.ibm.com/migration-guides/qiskit-2.0
7. Javadi-Abhari A, Treinish M, Krsulich K, Wood C, Lishman J, et al. Quantum computing with Qiskit. arXiv. 2024;arXiv:2405.08810 doi: 10.48550/arXiv.2405.08810
8. McKay DC, Wood CJ, Sheldon S, Chow JM, Gambetta JM. Efficient Z-gates for quantum computing. Phys Rev A. 2016;96:022330. doi: 10.1103/PhysRevA.96.022330
9. Alexander T, Kanazawa N, Egger DJ, Capelluto L, Wood CJ, Javadi-Abhari A, et al. Qiskit pulse: programming quantum computers through the cloud with pulses. Quantum Sci Technol. 2020;5:044006. doi: 10.1088/2058-9565/aba404
10. Gambetta JM, Motzoi F, Merkel ST, Wilhelm FK. Analytic control methods for high-fidelity unitary operations in a weakly nonlinear oscillator. Phys Rev A. 2011;83:012308. doi: 10.1103/PhysRevA.83.012308

11. Liang Z, Wang H, Cheng J, Ding Y, Ren H, Gao Z, et al. Variational quantum pulse learning. arXiv. 2022;arXiv:2203.17267. doi: 10.48550/arXiv.2203.17267
12. Melo A, Earnest-Noble N, Tacchino F. Pulse-efficient quantum machine learning. Quantum. 2023;7:1130. doi: 10.22331/q-2023-10-09-1130.
13. Meitei OR, Gard BT, Barron GS, Pappas DP, Economou SE, Barnes E, et al. Gate-free state preparation for fast variational quantum eigensolver simulations. NPJ Quant Inf. 2021;7:155. doi: 10.1038/s41534-021-00493-0
14. Schuld M. Supervised quantum machine learning mod-els are kernel methods. arXiv. 2021;arXiv:2101.11020. doi: 10.48550/arXiv.2101.11020
15. Xia R, Kais S. Quantum machine learning for electronic structure calculations. Nat Commun. 2018;9:4195.
16. Gianani I, Mastroserio I, Buffoni L, Bruno N, Donati L, Cimini V, et al. Experimental quantum embedding for machine learning. Adv Quantum Technol. 2021;5:2100140. doi: 10.1002/qute.202100140
17. Behrman EC, Steck JE, Kumar P, Walsh KA. Quantum algorithm design using dynamic learning. Quantum Inf Comput. 2008;8:12–29. doi: 10.48550/arXiv.0808.1558
18. Nola J, Behrman EC. Learning basis gates for one qubit using microwave pulses. 2022 [cited 2025 Apr 20]. Available from: https://soar.wichita.edu/handle/10057/23332
19. Thompson N, Steck JE, Behrman EC. A non-algorithmic approach to 'programming' quantum computers via machine learning. In: Proceedings of the 2020 IEEE International Conference on Quantum Computing and Engineering (QCE); 2020 Oct 12–16; Denver, CO, USA. p. 63–71.
20. Rethinam MJ, Javali AK, Hart AE, Behrman EC, Steck JE. A genetic algorithm for finding pulse sequences for nmr quantum computing. arXiv. 2004. doi: 10.48550/arXiv.quant-ph/0404170
21. Thompson NL, Nguyen NH, Behrman EC, Steck JE. Experimental pairwise entanglement estimation for an N-qubit sys-tem: A machine learning approach for programming quan-tum hardware. Quantum Inf Process. 2020;19:1–19. doi: 10.1007/s11128-020-02877-1
22. Behrman EC, Steck JE. Multiqubit entanglement of a general input state. Quantum Inf Comput. 2013;13:36–53. doi: 10.26421/QIC13.1-2-4.
23. Behrman EC, Nguyen NH, Steck JE, McCann M. Quantum neural computation of entanglement is robust to noise and decoherence. In: Bhattacharyya S, editor. Quantum Inspired Computational Intelligence: Research and Applications. Am-sterdam: Morgan Kaufmann, Elsevier; 2016. p. 3–32.
24. Nguyen NH, Behrman EC, Steck JE. Quantum learning with noise and decoherence: a robust quantum neural network. Quantum Mach Intell. 2020;2:1–15. doi: 10.1007/s42484-20-00013-x
25. Turati G, Dacrema MF, Paolo P. Feature selection for classifi-cation with QAOA. In: Proceedings of the 2022 IEEE Interna-tional Conference on Quantum Computing and Engineering (QCE); 2022 Sep 18–23; Broomfield, CO, USA. p. 782–785.
26. Chandarana P, Suarez P, Hegade NN, Solano E, Ban Y, Chen X. Meta-learning digitized-counterdiabatic quantum optimization. Quantum Sci Technol. 2023;8:045007. doi: 10.1088/2058-9565/ace54a
27. Tao Y, Zeng X, Fan Y, Liu J, Li Z, Yang J. Exploring accurate potential energy surfaces via integrating Variational Quantum Eigensolver with machine learning. J Phys Chem Lett. 2022;13:6420–6426. doi: 10.1021/acs.jpclett.2c01738
28. Nakayama A, Mitarai K, Placidi L, Sugimoto T, Fujii K. VQE-generated quantum circuit dataset for machine learning. arXiv. 2023;arXiv:2302.09751. doi: 10.48550/arXiv.2302.09751
29. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. Pytorch: An imperative style, high-performance deep learning library. arXiv. 2019;arXiv:1912.01703. doi: 10.48550/arXiv.1912.01703
30. Sutskever I, Martens J, Dahl G, Hinton GE. On the importance of initialization and momentum in deep learning. In: Proceedings of the 30th International Conference on Machine Learning; 2013 Dec 4–7; Miami, FL, USA. p. 1139–1147.
31. Simulating Backends at the Pulse-Level with DynamicsBackend. Qiskit Dynamics 0.5.1 documentation. [cited 2025 Apr 20]. Available from: https://qiskit-community.github.io/qiskit-dynamics/tutorials
32. Puzzuoli D, Wood C, Egger D, Rosand B, Ueda K. Qiskit Dynamics: A Python package for simulating the time dynamics of quantum systems. J Open Source Softw. 2023;8:5853. doi: 10.21105/joss.05853
33. Retired IBM Systems. [cited 2024 May 10]. Available from: https://quantum-computing.ibm.com/lab/docs/iql/manage/systems/retired-systems
34. Ban Y, Torrontegui E, Casanova J. Quantum neural networks with multi-qubit potentials. Sci Rep. 2023;13:4531. doi: 10.1038/s41598-023-35867-1.