Quantum Multiple Valued Kernel Circuits

Aviraj Sinha and Mitchell A. Thornton
Quantum Informatics Research Group, Darwin Deason Institute for Cyber Security
Southern Methodist University, Dallas, Texas, USA
{avirajs, mitch}@smu.edu

***Abstract*—Quantum kernels map data to higher dimensions for classification and have been shown to have an advantage over classical methods. In our work, we generalize recent results in binary quantum kernels to multivalued logic by using higher dimensional entanglement to create a qudit memory and show that the use of qudits offers advantages in terms of quantum memory representation as well as enhanced resolution in the outcome of the kernel calculation. Our method is not only capable of finding the kernel inner product of higher dimensional data but can also efficiently and concurrently compute multiple instances of quantum kernel computations in linear time. We discuss how this method increases efficiency and resolution for various distance-based classifiers that require large datasets when accomplished with higher-dimensioned quantum data encodings. We provide experimental results of our qudit kernel calculations with different data encoding methods through the use of a higher-dimensioned quantum computation simulator.**

***Index Terms*—multidimensional quantum computing, qudit, quantum information processing, quantum machine learning, quantum data clustering, quantum memory**

I. Introduction

Quantum machine learning (QML) is a developing field of quantum information processing in which quantum computers are used to perform machine learning tasks [1]. Many methods use quantum primitives to speed up training of classical methods such as the HHL algorithm for matrix inversion and Grover's algorithm to search for training parameters. Others use a classical computer to train a variational quantum circuit consisting of parameterized gates that are repeatedly adjusted based on a cost function; this variational circuit architecture is similar to designing a neural network structure and training its weights. The learning models that use variational circuits are an example of an exploratory approach to QML wherein the models offer a straightforward implementation in a quantum device. Kernel methods are similar to variational models; however, the main distinction is that kernel methods optimize parameters in the form of a linear combination. This means that training is accomplished in quadratic runtime due to calculating all pairwise distances between the training data. This type of calculation finds the optimal model for the desired classification task [2]. Alternatively, variational methods are more efficient and can be trained in linear time; however, there is no guarantee that an optimal measurement will be achieved and each implementation strategy is unique to the problem under consideration. The method described here focuses on exploring and improving the structure of quantum kernel computations and, in particular, the generalization of kernel calculations using qudits, also known as higher dimensional quantum digits. We show that quantum kernel computations can readily take advantage of higher dimensional qudits.

There are three parts of a quantum algorithm, data encoding, processing, and measurement. Quantum algorithms first encode data into an n-dimensional feature space known as the Hilbert space $\mathbb{H}_n$ [3] before being input into a processing portion, in this case a quantum kernel computation. The data encoding method is an important contributor to quantum speedup due to characteristics such as entanglement and superposition. The structure that represents multiple data encodings through the use of quantum superposition is described as a quantum memory. We present the benefits of using a qudit quantum memory that are realized due to the resulting higher-radix system. Using a higher-radix system allows for compact storage of encoded data as quantum superpositions with additional benefit gained due to qudits enabling more information to be represented in each individual encoding.

More specifically, using qudits instead of qubits results in benefits for quantum kernel computations since the quantum effects of superposition and entanglement result in greater data density within the quantum circuit. The result is that there will be more data encodings in superposition as well as more information stored per data encoding. This improvement in data storage naturally results in improvements in quantum machine learning by allowing parallelized computation on large datasets and on high-dimensional data encodings that represent a complex feature space. We demonstrate how creating a qudit memory increases the efficiency of a quantum kernel computation by comparing it to a kernel computation based on the use of qubits. To our knowledge, quantum machine learning methods have not taken advantage of benefits provided by using higher-dimensional qudits. There have been many quantum hardware advances that allow qudits to be represented as higher energy states. A previous implementation describes a photonic circuit that produces a pair of 96-state qudits that can be used to create a total of $96^2 = 9216$ dimensional quantum states [4]. In addition to quantum speedups, there are some practical hardware benefits of using qudits. First, the use of qudits avoids the challenge of requiring large numbers of qubits thus reducing circuit resource requirements and lessening the problems associated with decoherence. Second, creating qudits is technically achievable since it only involves removing restrictions that are usually present in binary-basis circuits that purposely restrict qubit implementations to use only two basis states.

For quantum kernel computations, the data encoding mechanism can be abstractly considered as feature mapping to a higher dimension. In this case, the quantum Hilbert space mapping is denoted as $x \rightarrow |\phi(x)\rangle$. This type of mapping to a higher dimension, known as the "kernel trick" in machine learning, eases the ability to create decision boundaries. Kernel-based mapping is created by synthesizing a circuit that evolves the basis $|0\rangle$ state into the desired output state $\mathbf{U}_\phi |0\rangle \rightarrow |\phi(x)\rangle$. The kernel itself is customized by changing the quantum data encoding strategy for creating the feature mappings. The synthesized inputs are then combined into a circuit that finds the overlap between quantum states $\langle\phi|\psi\rangle$. In method described here, we show that using higher-dimensional qudits allows us to find the overlap between multiple combinations of higher dimensional quantum states. In addition to enhanced parallelism for kernel testing, the ease of creating higher-dimensional states allows us to create kernels that are difficult to simulate on a classical computer that inherently uses only two binary states. This enhancement yields a significant benefit in comparison to a classical computing approach since it enables parallelized qudit kernel computations thus gaining a quantum advantage.

Higher-dimensional qudits additionally increase storage efficiency and thus offer further enhancements to computation speed. We describe the creation of a higher-dimensional quantum kernel circuit based on the use of qudits that calculate many groups of quantum kernel distances simultaneously. This technique requires the creation of a qudit memory as a state generation circuit and is briefly summarized. The quantum speedup in this method results from the way the data is encoded as well as the use of qudits. We compare the structure and efficiency of a qubit kernel computational circuit with that of a radix-4 qudit kernel computational circuit. Additionally, we evaluate the qudit kernel computational circuit using two different data encoding methods.

II. Previous Work

The use of kernel methods to map data into a higher-dimensional feature space through a feature mapping function for conventional electronic computing is well known in the machine learning (ML) community. A feature space is used by a ML algorithm to create classification boundaries with more ease. One way that a higher-dimensional state can be created is through the use of kernels created by an inner product computation. The inner product of two feature maps must be mapped into a higher dimensional feature space that measures distances between data points. One common classification algorithm that uses kernels is the support vector machine (SVM). The SVM separates classes of data points using a line, or in multiple dimensions, a hyperplane. More information about classical kernels can be found in [5]. Kernel methods can be used as a quantum primitive in a hybrid QML model wherein a classically running algorithm, such as SVM, queries a quantum computer for kernel calculation results [3].

Previous work investigates the usage and training of quantum kernels for small datasets using a few radix-2 qubits on quantum devices. The research of [6] explores quantum kernels that follow the pattern of inner product interference routines and describes how to use them for classification. Unlike hybrid models, this work performs the entire classification task on the quantum device; indicating that labels must be output by the quantum computer. The two classifiers created are the Hadamard classifier and the swap test classifier. The Hadamard classifier encodes data in the two output branches of superposition. When the Hadamard operator is applied, it results in overlap. In the swap test classifier, a conditional swap gate is used to create an inner product between the input and encoded weights. In both methods, an ancilla qubit is used to create the interference, a series of weights is synthesized by a series of gates and the encoded data itself, and the resulting label qubit is produced. For these methods, a projective measurement is used to create an overlap and another measurement is used for creating the label that is entangled with the data. This method makes use of the fidelity between the quantum states of the training dataset and test data.

The primitives used for interference generation are based upon the original swap test for quantum fingerprinting [7]. Quantum fingerprinting was originally described as a way of finding the overlap of two quantum states through interference using a controlled swap gate. The quantum effect of interference is used as a similarity measure of two quantum states. The idea of the qubit inner product has been adapted to machine learning to work as a kernel in a quantum support vector machine.

In [8] there is a comparison between a quantum kernel and a variational classifier implemented on a quantum computer. The design ansatz or architecture is used with multiple layers. A layer consists of Hadamard operations, a layer of parameterized rotations, then a layer of entangling gates. This variational architecture was modified to create a kernel inner product by reversing the ansatz operations (*i.e.*, conjugate transpose) and replacing the rotational weights with different values. This past work outlines the training of the kernel rotational weights through a dual quadratic program. In contrast, it is noted that for a variational algorithm, a stochastic gradient descent algorithm can be used.

Recent work has proven the quantum advantage of QML algorithms by creating a kernel for certain data types. The work of [9] constructs a dataset that is hard to classify based on the discrete logarithm problem. A kernel is trained using a similar technique as the kernel architecture and training in [8]. The result is that the quantum classifier provides high accuracy while a classical ML learner is reduced to randomly guessing.

Recently, qudits have been used to improve quantum algorithms as shown in the qudit-based Deutsch-Jozsa and quantum database search algorithms. In [10] the binary Deutsch-Jozsa algorithm is generalized to qudits. The algorithm can distinguish between constant and balanced functions in a single query. In addition, it can identify affine functions in oracles. Also in [10], a quantum database search is expanded to work with a qudit database, also referred to as quantum memory. The main benefit of working with a quantum database is the resulting enhanced parallelism of computations. Using a similar technique, our multivalued kernel method relies on a qudit memory. We will briefly describe how the qudit memory can be implemented as a state generation circuit using higher-dimensional qudit gates. We also describe the exponential improvement of the higher-dimensional quantum memory. Overall, our results show that the usage of qudits for these types of quantum primitives yield overall computational speedup when applied to QML.

III. Background

A. Qubit Kernel

As background in describing kernels and quantum memory, the principle of entanglement is briefly reviewed. A simple example of an entanglement generator, the Bell state generator, is shown in Figure 1. The qubits are both initialized to the binary computational basis state, $|0\rangle_2$ where the subscript denotes the radix or dimension of the Hilbert space. In standard linear algebraic notation, the computational basis is $\langle 0| = [1,0]$, $|0\rangle = [1,0]^\dagger$, $\langle 1| = [0,1]$, $|1\rangle = [0,1]^\dagger$. The idea is that the uppermost qubit evolves into a state of perfect superposition among $|1\rangle$ and $|0\rangle$ due to the Hadamard gate and it is then applied to the control port of a controlled-NOT gate. Thus, the controlled-NOT gate serves as an entangling gate resulting in the target qubit becoming entangled with the value of the control qubit. Entanglement causes the target value to be determined based upon the value that the control qubit collapses into after measurement. As a result, entanglement essentially allows the transmission of data between two qubits instantaneously.

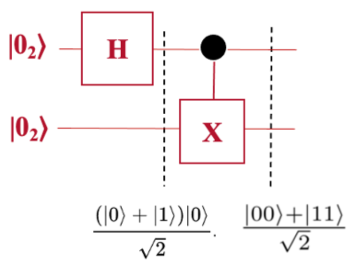
$|0_2\rangle$
$|0_2\rangle$
H
X
$\frac{(|0\rangle+|1\rangle)|0\rangle}{\sqrt{2}}$.
$\frac{|00\rangle+|11\rangle}{\sqrt{2}}$

Fig. 1: Bell State Entanglement Generator

A parametrized form for a general qubit, $|\varphi_j\rangle$, can be written in terms of the Bloch sphere angles θ_j and ϕ_j in the form shown in Equation 1 where (θ_j, ϕ_j) are identical to the standard angles describing a point in the spherical coordinate system and $i^2 = -1$.

$$|\varphi_j\rangle = \cos(\frac{\theta_j}{2})\,|0\rangle + e^{i\phi_j}\sin(\frac{\theta_j}{2})\,|1\rangle \tag{1}$$

To understand how quantum gates can be synthesized as state generation memory elements, we express them as parameterized unitary matrices in terms of (θ_j, ϕ_j). A generalized unitary matrix $\mathbf{U}_j$ is represented as a rotation matrix with two parameters as shown in Equation 3. To create the matrix we apply a series of Pauli-$\mathbf{X}$ and Pauli-$\mathbf{Z}$ operators with parameterized angles (θ_j, ϕ_j) defining the desired qubit $|\varphi_j\rangle$ as shown in Equations 2. Evolving $|0\rangle$ by applying the synthesized unitary matrix, $\mathbf{U}_j$, results in the generation of the desired quantum state that stores data in the $|\varphi_j\rangle$ state shown in Equation 4. In this sense, (θ_j, ϕ_j), can be considered as the parameters found via a training process for the state generation matrix $\mathbf{U}_j$.

$$\begin{aligned} \mathbf{R_X}(\theta) &= e^{-i\theta\mathbf{X}} = \begin{bmatrix} \cos(\frac{\theta}{2}) & -i\sin(\frac{\theta}{2}) \\ -i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix} \\ \mathbf{R_Z}(\phi) &= e^{-i\phi\mathbf{Z}} = \begin{bmatrix} e^{-i\frac{\phi}{2}} & 0 \\ 0 & e^{i\frac{\phi}{2}} \end{bmatrix} \end{aligned} \tag{2}$$

$$\begin{aligned} \mathbf{U}_j &= \mathbf{U}(\theta_j, \phi_j) = \mathbf{R_Z}(\phi_j)\mathbf{R_Y}(\theta_j) \\ &= \mathbf{R_Z}(\phi_j)\mathbf{R_X}(-\pi/2)\mathbf{R_Z}(\theta_j)\mathbf{R_X}(\pi/2) \\ &= \begin{bmatrix} \cos\left(\frac{\theta_j}{2}\right) & -\sin\left(\frac{\theta_j}{2}\right) \\ e^{i\phi_j}\sin\left(\frac{\theta_j}{2}\right) & e^{i\phi_j}\cos\left(\frac{\theta_j}{2}\right) \end{bmatrix} \end{aligned} \tag{3}$$

$$\begin{aligned} |\varphi_j\rangle = \mathbf{U}_j|0\rangle &= \begin{bmatrix} \cos\left(\frac{\theta_j}{2}\right) & -\sin\left(\frac{\theta_j}{2}\right) \\ e^{i\phi_j}\sin\left(\frac{\theta_j}{2}\right) & e^{i(\phi_j)}\cos\left(\frac{\theta_j}{2}\right) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} \cos\frac{\theta_j}{2} \\ e^{i\phi_j}\sin\left(\frac{\theta_j}{2}\right) \end{bmatrix} \\ &= \cos(\frac{\theta_j}{2})\,|0\rangle + e^{i\phi_j}\sin(\frac{\theta_j}{2})\,|1\rangle \end{aligned} \tag{4}$$

Using these concepts, we generalize the Bell state generator to implement an inner product kernel as shown in Figure 2 that is used to find the overlap of two quantum states, φ_0 and φ_1, and is a modification of the original swap test in [3]. First, the top control qubit is evolved into a state of superposition using a Hadamard gate, $|\Psi\Phi\rangle\,(t_1) = \frac{1}{\sqrt{2}}(|0\rangle\,|0\rangle + |1\rangle\,|0\rangle)$. Then the first set of gate transformations are activated when the top control qubit is $|1\rangle$ and the second when the top qubit is $|0\rangle$ due to a negative control. The first gate, control-$\mathbf{U}_0$ results in $|\Psi\Phi\rangle\,(t_2) = \frac{1}{\sqrt{2}}(|0\rangle\,|0\rangle + |1\rangle\,|\varphi_0\rangle)$ and embeds the $|\varphi_0\rangle$ quantum state. Likewise, the second gate, negative-control-$\mathbf{U}_1$, causes an evolution that embeds the $|\varphi_1\rangle$ quantum state resulting in $|\Psi\Phi\rangle\,(t_3) = \frac{1}{\sqrt{2}}(|0\rangle\,|\varphi_1\rangle + |1\rangle\,|\varphi_0\rangle)$. The second Hadamard gate computes the sum and difference of the quantum states $|\varphi_0\rangle$ and $|\varphi_1\rangle$ in different superposition branches as $|\Psi\Phi\rangle\,(t_4) = \frac{1}{\sqrt{2}}(|0\rangle\,|\varphi_1\rangle + |1\rangle\,|\varphi_1\rangle + |0\rangle\,|\varphi_0\rangle - |1\rangle\,|\varphi_0\rangle)$. When the $\mathbf{M}_0 = |0\rangle\,\langle 0|$ projective operator within the Pauli-$\mathbf{Z}$ measurement observable is applied to the uppermost qubit $|\Psi\rangle$, the probability that it collapses to $|0\rangle$ is obtained, as shown in Equation 5. We can also use the other projector of the same basis, $\mathbf{M}_1 = |1\rangle\,\langle 1|$, to obtain a second probability distribution, $\mathrm{Prob}\,[|\Psi\rangle\,(t_5) \rightarrow |1\rangle]$. These measurements result in binomially-distributed probabilities that can be used to obtain the inner product value. Multiple runs N of the quantum program referred to as "shots" can be used to efficiently obtain an estimate of the probability distribution from which the inner product can be computed with simple algebraic relationships. Specifically, $p_0 = N_0/N$ and $p_1 = N_1/N$ where N_0 and N_1 are the number of measurements of $|0\rangle$ and $|1\rangle$ respectively. Using estimates of the p_0 and p_1 values from the N shots, we obtain $\text{Re}\{\langle\varphi_0|\varphi_1\rangle\} = 2p_0 - 1$ and $\text{Re}\{\langle\varphi_0|\varphi_1\rangle\} = 1 - 2p_1$.

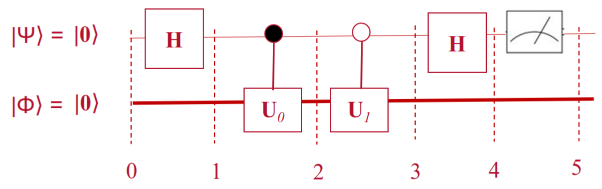

Fig. 2: Qubit Innerproduct Kernel

$$\begin{aligned}
\text{Prob}\left[|\Psi\rangle(t_5) \rightarrow |0\rangle\right] &= \langle\psi(t_4)|\mathbf{M}_0^\dagger\mathbf{M}_0|\psi(t_4)\rangle \\
&= \langle\psi(t_4)|0\rangle\,\langle 0|0\rangle\,\langle 0|\psi(t_4)\rangle \\
&= \langle\psi(t_4)|0\rangle\,\langle 0|\psi(t_4)\rangle \\
&= \frac{1}{2}(\langle\varphi_1| + \langle\varphi_0|)\frac{1}{2}(|\varphi_1\rangle + |\varphi_0\rangle) \\
&= \frac{1}{4}(\langle\varphi_0|\varphi_1\rangle + \langle\varphi_1|\varphi_0\rangle + 2) \\
&= \frac{1}{2}\Big(\text{Re}\{\langle\varphi_0|\varphi_1\rangle\} + 1\Big)
\end{aligned} \tag{5}$$

B. Data Encoding in Quantum Storage

There are many data encoding methods for storing data in a quantum state, such as angle encoding, which involves rotating qubits and storing the value in the probability of the qubits, and basis encoding, which simply stores the discrete 0 or 1 states of the qubits [3]. The data encoding method of choice is equivalent to the type of feature map of the kernel. As a result, the strength of quantum kernel-based machine learning algorithms depends on the type of data encoding strategy selected, the training method for the variational parameters that transform the encodings, as well as the effectiveness of the synthesis method. We will not focus on a particular synthesis method or training of qudits in higher dimensions but rather the overall structure of the kernel computation itself. For the evaluation of our kernel method, we will first use modulo-addition gates to create the qudit equivalent of basis encoding, then we will use random rotation matrices to represent a set of qudit angle encoded state generation operators.

C. Qudit Gates

Next, we generalize the entanglement-based kernel computation method to use qudits. The Chrestenson gate in Equation 6 is a generalized form of the Hadamard operator and is defined by a matrix that is parameterized by the radix of computation. The gate represented by the $\mathbf{C}_4$ transfer matrix allows a radix-4 basis qudit to evolve into a quantum state of equal superposition among all four basis qudits, $|0\rangle_4$, $|1\rangle_4$,$|2\rangle_4$, and $|3\rangle_4$ where the subscript "4" indicates the Hilbert vector space dimension.

$$\mathbf{C}_4 = \frac{1}{\sqrt{4}}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} \tag{6}$$

$$\mathbf{A_{2,U}} = \begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&u_{00}&u_{01}&u_{02}&u_{03}& &0&0&0\\
0&0&0&0&0&0&0&0&u_{10}&u_{11}&u_{12}&u_{13}&0&0&0&0\\
0&0&0&0&0&0&0&0&u_{20}&u_{21}&u_{23}&u_{24}&0&0&0&0\\
0&0&0&0&0&0&0&0&u_{30}&u_{31}&u_{33}&u_{34}&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1
\end{bmatrix} \tag{7}$$

The qudit controlled gate $\mathbf{A}_{h,k}$ results in the target state $|k\rangle$ when the state of the control qudit is $|h\rangle$. The process of entanglement in higher dimensional qubits has been theoretically analyzed and described in detail in [11], [12]. The implementation of a higher-dimensioned quantum entanglement operator as a state preparation circuit is described in [13]. In Equation 7, we generalize the applied matrix to contain a general radix-4 unitary $\mathbf{U}$ for the synthesis of different data encoding types. This higher dimensional unitary matrix requires expanding the number of rotational parameters. We can create the radix-4 equivalent of the rotation matrix by calculating the outer product of two radix-2 rotation matrices as shown in Equation 8. In Figure 3, the Chrestenson gate is applied to a computational basis state to generate perfect superposition of the topmost qudit, $|\Psi\Phi\rangle(t_0) = (\mathbf{C_4} \otimes \mathbf{I_4})|00\rangle_4 = \frac{1}{2}(|0\rangle_4 + |1\rangle_4 + |2\rangle_4 + |3\rangle_4)|0\rangle_4$. Next, a controlled entangling gate represented by $\mathbf{A}_{2,\mathbf{U}}$ is applied that evolves the target qudit according to a unitary operator $\mathbf{U}$ parameterized by higher-dimensional Bloch sphere angles in response to the control qudit's $|2\rangle$ probability amplitude value, shown in the black circle. $\mathbf{U}$ is synthesized in a generalized manner as the qubit-based operators previously described and is shown in Equation 8.

$$\mathbf{U} = \mathbf{U}(\theta_0, \phi_0) \otimes \mathbf{U}(\theta_1, \phi_1) = \mathbf{U}(\theta_0, \phi_0, \theta_1, \phi_1) \tag{8}$$

$$\begin{aligned}
|\Psi\Phi\rangle(t_1) &= \mathbf{A}_{2,\mathbf{U}} \times (\mathbf{C_4} \otimes \mathbf{I_4})|00\rangle_4 \\
&= \frac{1}{\sqrt{4}}(|0\rangle|0\rangle + |1\rangle|0\rangle + |2\rangle|\varphi\rangle + |3\rangle|0\rangle)
\end{aligned} \tag{9}$$

The series of qudit-controlled unitary gates form a multivalued control gate (MVCG) that is a series of $\mathbf{A}$ gates with every possible control basis value present as shown in Figure 4. The MVCG gate enables the advantages of using higher-dimensional gates. While a qubit controlled-NOT gate will only perform a single operation on the target qubit, the MVCG gate allows the selection of multiple different

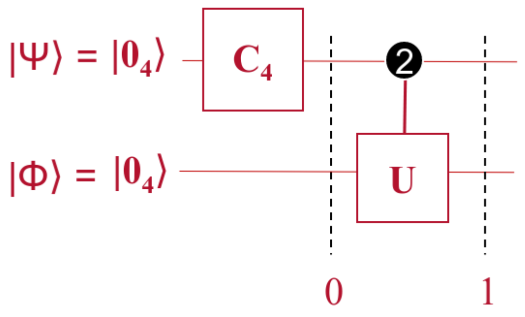

Fig. 3: Radix-4 Qudit Entanglement

entangling gates, depending on the control activation value. As a result, the MVCG is sometimes referred to as a quantum multiplexer [14]. In Equation 4, a 4-dimensional qudit MVCG is depicted with each $\mathbf{U}_k$ of dimension 4×4 representing a single operation that is applied to the target qudit based upon the value of the control qudit's probability amplitude for each basis component, $|k\rangle$. In terms of notation, $\mathbf{0}_4$ represents the 4×4 null or "all-zeros" component matrix.

$$\mathbf{MVCG_4} = \begin{bmatrix} \mathbf{U}_0 & \mathbf{0}_4 & \mathbf{0}_4 & \mathbf{0}_4 \\ \mathbf{0}_4 & \mathbf{U}_1 & \mathbf{0}_4 & \mathbf{0}_4 \\ \mathbf{0}_4 & \mathbf{0}_4 & \mathbf{U}_2 & \mathbf{0}_4 \\ \mathbf{0}_4 & \mathbf{0}_4 & \mathbf{0}_4 & \mathbf{U}_3 \end{bmatrix}$$

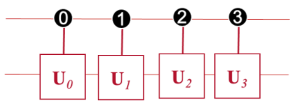

Fig. 4: Multiple Valued Control Gate

IV. Contribution

A. *Qudit Memory*

To create the multivalued kernel computation circuit, it is necessary to represent the data in a higher-dimensional quantum memory circuit. To create the higher-dimensional circuit structure we apply the $\mathbf{C}_4$ gate on the control qudits of the MVCG as shown in Figure 5. By applying the $\mathbf{C}_4$ gate to the previously described quantum multiplexer, we can create a series of entangled data and address values that are in superposition. The resulting Equation 11 represents a circuit structure that is described as a quantum memory in [15] and is represented by Equation 10. The $|j\rangle$ control qudit serves as an address qudit that is entangled with the corresponding data in the form of quantum states $|\varphi\rangle$ that are in superposition. N is the normalizing factor that results in the equiprobable probability amplitudes to cancel when squared; for our case N is 4, due to our single radix-4 qudit. Quantum memory is an essential requirement for many quantum machine learning algorithms. Due to the four control states of the qudit, the amount of data stored in the superimposed state of a single particle increases from two to four. In addition, the dimension of each individual qudit representing encoded data increases from two to four in this radix-4 example. In the small illustrative example shown in Figure 5 and described by Equation 11, there is only a single controlling address qudit $n = 1$, thus the memory data stored in superposition only represents four distinct values. When there are multiple address/controlling particles $n > 1$, the amount of data increases exponentially with respect to the value of n of the form r^n where r is the single qudit Hilbert space dimension (or radix value) and n is the number of qudits. This exponential increase in data storage can be achieved when the qudit memory circuit is generalized to utilize $n > 1$ address qudits.

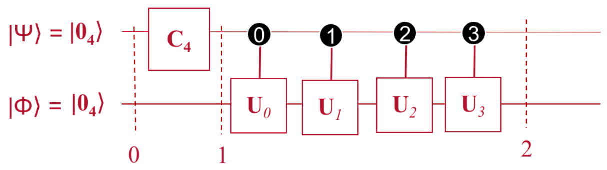

Fig. 5: Qudit Memory

$$\frac{1}{\sqrt{N}} \sum_{j=0}^{N} |j\rangle_4 |0\rangle_4 \longrightarrow \frac{1}{\sqrt{N}} \sum_{j=0}^{N} |j\rangle_4 |\varphi_j\rangle \tag{10}$$

$$\begin{aligned} &|\Psi\Phi\rangle (t_2) \\ &= \mathbf{A}_{3,\mathbf{U}} \times \mathbf{A}_{2,\mathbf{U}} \times \mathbf{A}_{1,\mathbf{U}} \times \mathbf{A}_{0,\mathbf{U}} \times (\mathbf{C_4} \otimes \mathbf{I_4}) |00\rangle_4 \\ &= \frac{1}{\sqrt{4}} (|0\rangle |\varphi_0\rangle + |1\rangle |\varphi_1\rangle + |2\rangle |\varphi_2\rangle + |3\rangle |\varphi_3\rangle) \end{aligned} \tag{11}$$

B. *Loading Quantum Data*

In addition to choosing an appropriate quantum data encoding method, quantum memory architectures must be specified for selecting multiple unitary operators $\mathbf{U}$ in quantum form that result in the corresponding quantum states $|\varphi_j\rangle$. In the case of radix-2 quantum read-only memory or QROM, the operators used to create the states are stored as described in [16], whereas in QRAM, quantum states are loaded from conventional electronic data circuits as described in [15]. In Figure 6 the quantum state generation circuit for a QROM is represented by the overall transformation circuit with transfer matrix $\mathbf{T}_{data}$ and the processing portion is likewise represented by $\mathbf{T}_{proc}$. To control which data values are accessed from the QROM, a quantum computation is performed that generates the address qudits to be applied to the QROM denoted here as $\mathbf{T}_{addr}$.

The overall algorithm representing a data processing application is specified by the collection of $\mathbf{T}_{proc_i}$ circuits with the overall transformation matrix $\mathbf{T}_{circ}$ formed as $\mathbf{T}_{circ} = \prod_{i=1}^{k} (\mathbf{T}_{addr_i} \otimes \mathbf{T}_{proc_i}) \mathbf{QROM} (\mathbf{T}_{addr_{i-1}} \otimes \mathbf{T}_{proc_{i-1}})$. The set of controlling qudits can be considered to be an address analogous to a conventional addressable memory unit. When implemented within a general data processing algorithm, the QROM can be embedded multiple times at any location in the serial cascade and would generally follow the portion of the algorithm that generates an address but precedes the portion of the algorithm that actually processes the addressed data. An example of such a structure is shown in Figure 6.

In the case of the quantum qudit memory embedded within the kernel calculation circuit, it is possible to use the four different basis states for the $\mathbf{T}_{addr}$ qudit to retrieve the

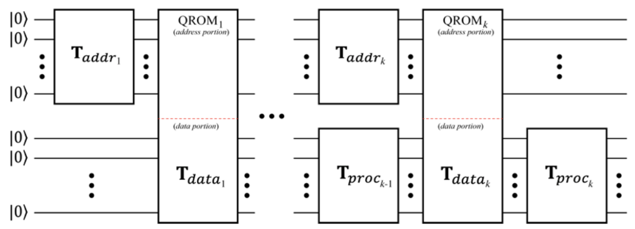

Fig. 6: Quantum Algorithm Containing k Instances of QROM Structures

corresponding stored values within $\mathbf{T}_{data}$ that is embedded within the four unitary matrices $\mathbf{U}_j$ as shown in Figure 5. To create the controlled-$\mathbf{U}_j$ gates, the process described in [17] can be used to add a control qudit to the synthesized $\mathbf{U}_j$ gates that comprise $\mathbf{T}_{data}$. Generalizing this form of QROM architecture allows an arbitrary number of quantum states to be generated from a corresponding set of synthesized $\mathbf{U}_j$ operators that, in turn, represent a large dataset. Thus, the quantum kernel algorithm can be implemented over a large set of values.

C. *Qudit Kernel*

As previously discussed, a quantum kernel computation circuit enables one to find the overlap of quantum states. In the multiple-valued case, this is the overlap between every possible combination of quantum states. In Equation 12, when the rightmost $\mathbf{C}_4$ gate of Figure 7 is applied to the controlling qudit (*i.e.* address qudit) of the quantum memory, a similar result is achieved as that of the swap test in that multiple sums and differences of each quantum state are generated. In Equation 13, a projective measurement operator is applied as described by a generalized radix-4 Pauli-$\mathbf{Z}$ observable. This observable is formed in part using the $\mathbf{M}_0$ projector where $\mathbf{M}_0 = |0_4\rangle\langle 0_4|$. The $\mathbf{M}_0$ projector yields the probability that the measured qudit collapses to state $|0_4\rangle$. The result is that the probability distribution contains the sum of every possible inner product combination in constant time as expressed in Equation 13. Normally, finding every possible combination between n points is proportional to $n(n-3)/2$. Thus, by using the qudit-based method, all combinations can be calculated in the linear time required to synthesize the circuits resulting in a quadratic runtime reduction. As a result, this circuit can compute more than a single kernel overlap in one operation and it can be used to efficiently find the overall similarity among multiple higher-dimensional quantum states.

When the quantum algorithm represented by $\mathbf{T}_{proc}$ is a kernel calculation, we can calculate the distance between multiple points in higher dimensions. In this way we can reduce the number of calls to a quantum computer since this is equivalent to finding multiple kernels at once and adding their differences. This can be used to speed up kernel enhanced algorithms that require multiple kernel calculations such as K-nearest neighbors clustering algorithms. A K-dimensional qudit can be used to create, calculate and sum K distances in the Hilbert space. We can create a classifier by encoding one of the synthesized states as ϕ_0 and finding the kernel distance from the remainder of encoded data in the cluster $\varphi_2, \varphi_3 \ldots \varphi_i$. If the resulting probability of our kernel is close to unity (1), the sample is close to that class cluster and can thus be classified as part of that class.

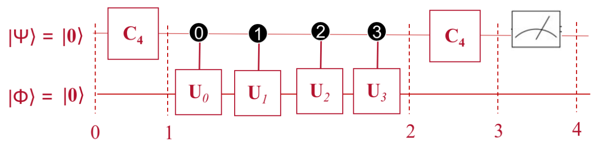

Fig. 7: Qudit Innerproduct Kernel

$$\begin{aligned}
&|\Psi\Phi\rangle(t_3) = (\mathbf{C_4} \otimes \mathbf{I_4})|\Psi\Phi\rangle(t_2) = \\
&\quad \frac{1}{\sqrt{16}}[(|0\rangle|\varphi_0\rangle) + |1\rangle|\varphi_0\rangle + |2\rangle|\varphi_0\rangle + |3\rangle|\varphi_0\rangle) + \\
&\qquad |0\rangle|\varphi_1\rangle) + i|1\rangle|\varphi_1\rangle - |2\rangle|\varphi_1\rangle - i|3\rangle|\varphi_1\rangle + \\
&\qquad |0\rangle|\varphi_2\rangle) - |1\rangle|\varphi_2\rangle + |2\rangle|\varphi_2\rangle - |3\rangle|\varphi_2\rangle + \\
&\qquad |0\rangle|\varphi_3\rangle) - i|1\rangle|\varphi_3\rangle - |2\rangle|\varphi_3\rangle + i|3\rangle|\varphi_3\rangle)]
\end{aligned} \tag{12}$$

$$\begin{aligned}
&\mathrm{Prob}\left[|\Psi\rangle(t_4) \rightarrow |0\rangle_4\right] \\
&= \langle\psi(t_3)|\mathbf{M}_0^{\dagger}\mathbf{M}_0|\psi(t_3)\rangle = \langle\psi(t_3)|0\rangle_4 \langle 0|0\rangle_4 \langle 0|\psi(t_3)\rangle_4 \\
&= \frac{1}{\sqrt{16}}(\langle\varphi_0| + \langle\varphi_1| + \langle\varphi_2| + \langle\varphi_3|)\frac{1}{\sqrt{16}}(|\varphi_0\rangle + |\varphi_1\rangle + |\varphi_2\rangle + |\varphi_3\rangle) \\
&= \frac{1}{16}(2\,|\,\langle\varphi_0|\varphi_1\rangle\,| + 2\,|\,\langle\varphi_0|\varphi_2\rangle\,| + 2\,|\,\langle\varphi_0|\varphi_3\rangle\,| \\
&+ 2\,|\,\langle\varphi_1|\varphi_2\rangle\,| + 2\,|\,\langle\varphi_1|\varphi_3\rangle\,| + 2\,|\,\langle\varphi_2|\varphi_3\rangle\,| + 4) \\
&= \frac{1}{8}(|\,\langle\varphi_0|\varphi_1\rangle\,| + |\,\langle\varphi_0|\varphi_2\rangle\,| + |\,\langle\varphi_0|\varphi_3\rangle\,| + |\,\langle\varphi_1|\varphi_2\rangle\,| \\
&+ |\,\langle\varphi_1|\varphi_3\rangle\,| + |\,\langle\varphi_2|\varphi_3\rangle\,| + 2)
\end{aligned} \tag{13}$$

V. Evaluation

To evaluate the higher-dimensional similarity metric, we synthesize unitary radix-4 gates that encode data and generate corresponding create quantum states. We implemented the qudit kernel computation circuit in Figure 7 using the Google *cirq* circuit simulator with various unitary matrices. For the first part of the experiment, shift matrices are used, also known as modulo addition operators [18]. The shift gates are actually higher-dimensional versions of the Pauli-$\mathbf{X}$ operator for radix-4. The Pauli-$\mathbf{X}$ transfer matrix applies the operation $(|\varphi\rangle + |1\rangle) \pmod 2$ to the input state $|\varphi\rangle$, where $|\varphi\rangle$ can be $|0\rangle$ or $|1\rangle$. The radix-4 equivalents of the Pauli-$\mathbf{X}$ gate are shown in Equation 14 representing $(|\varphi\rangle + |0\rangle) \pmod 4$, $(|\varphi\rangle + |1\rangle) \pmod 4$, $(|\varphi\rangle + |2\rangle) \pmod 4$ and $(|\varphi\rangle + |3\rangle) \pmod 4$. The generated quantum state $|\varphi\rangle$ can be formed from either the $|0\rangle_4$, $|1\rangle_4$, $|2\rangle_4$, or $|3\rangle_4$ radix-4 basis states. When these operators are used, the quantum states stored within the memory circuit contain multiple basis encodings that have the form of Equation 15 where $|\varphi\rangle$ is a discrete state value.

$$\mathbf{M}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{M}_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

$$\mathbf{M}_2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \mathbf{M}_3 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{14}$$

$$|\varphi\rangle_4 = |a\rangle_4 \, , a \in \{0, 1, 2, 3\} \tag{15}$$

When using the modulo addition operators as the $\mathbf{U}_j$ matrices, the data is represented by its qudit equivalent in the form of basis encoded memory. That is, it does not store values in the probability amplitudes of the quantum states, which means that the states are not in superposition. Unlike basis encoding which simply stores data as 0 or 1 discrete basis states of a qubit, the qudit data is stored as the 0, 1, 2, or 3 basis states. Likewise, rather than using control-$\mathbf{X}$ gates we use the higher-dimensional equivalent of controlled-modulo addition gates. Basis state encoding is used for the first test evaluation since it is among the simplest and easiest to interpret. We note that other, more efficient, quantum data encoding and memory circuits can also be applied such as the previously mentioned angle or probability amplitude types of encodings.

For the second set of results, we analyze the similarity metric distribution using n-dimensional random unitary matrices for $\mathbf{U}_j$. We generate random unitary matrices following the steps in [19], that are implemented in the Python library `scipy-stats`. Random generation of these matrices can be created through the randomization of the rotation parameters shown in Equation 8. When random unitary rotations are performed on a single qudit, the qudit equivalent of angle-encoded data is created. Equation 16 shows the resulting data values stored in the four probability amplitudes of a single qudit. Like qubit angle encoding, the sum of squared magnitudes of the probability amplitudes (*i.e.*, α_j values) must be equal to unity (1) due to Born's rule or, alternatively, the laws of probability theory. That is, $|\alpha_0|^2+|\alpha_1|^2+|\alpha_2|^2+|\alpha_3|^2 = 1$.

$$|\varphi\rangle_4 = \alpha_0|0\rangle_4 + \alpha_1|1\rangle_4 + \alpha_2|2\rangle_4 + \alpha_3|3\rangle_4 \tag{16}$$

A. *Qudit Basis Encoding*

In our illustrative example, four data points are stored. The results show the result of the qudit memory circuit when each unitary matrix is denoted as $\mathbf{U}_0$, $\mathbf{U}_1$, $\mathbf{U}_2$, $\mathbf{U}_3$. Since all four matrices are different, all four measurement values are in equal superposition.

The calculations using Equation 13 reveal that the similarity of the encoded quantum data states in the memory can be found by measuring the probability that the qubit collapses into state $|0\rangle_4$ upon measurement. Using different combinations of shift matrices can reveal that this is a similarity metric that can be used between multiple states.

The resulting probabilities from the example circuits show that using qudit basis encoding results in a similarity metric with five discrete levels. The similarity of the quantum states for a variety of matrix combinations is shown in table I. We observe that there are only five possible similarities between these four states when higher dimensional basis encoding is applied. The first row indicates that when all matrices are different the similarity is 0.25, when one pair are the same the metric increases to 0.3675, and when two pairs are the same the similarity metric is 0.5. Only when all operators are the same and result in the same quantum state, does the similarity metric yield a perfect unity value (1). The qubit equivalent is more strict in that it only has two levels of similarity between the quantum basis states, $|0\rangle_2$ or $|1\rangle_2$. In comparison, we show the advantage that the use of radix-4 qudits allows more flexibility for quantum state comparison approaches by providing for five possible levels of similarity.

TABLE I: Qudit Basis Encoding Similarities

Unitary Matrices Used	Measurement $P(\|0\rangle)$
$\{\mathbf{U}_0, \mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3\}$	0.25
$\{\mathbf{U}_0, \mathbf{U}_0, \mathbf{U}_2, \mathbf{U}_3\}$	0.3675
$\{\mathbf{U}_0, \mathbf{U}_0, \mathbf{U}_1, \mathbf{U}_1\}$	0.50
$\{\mathbf{U}_0, \mathbf{U}_0, \mathbf{U}_0, \mathbf{U}_3\}$	0.6250
$\{\mathbf{U}_0, \mathbf{U}_0, \mathbf{U}_0, \mathbf{U}_0\}$	1

B. *Qudit Angle Encoding*

To test the kernel calculation method for other encoding strategies such as angle encoding, we generalize qudit gates as random unitary matrices that represent generalized rotations on a hyper-Bloch sphere. When the four random unitary matrices are created and used in the circuit for 10,000 trials, a histogram in Figure 8 is created around the mean measurement probability. This result indicates that when the four unitary matrices are random, the average measurement probability is 0.25 as would be expected. This histogram estimates the probability mass function (PMF) that represents measurement outcome with respect to an observable formed to measure $|0\rangle_4$.

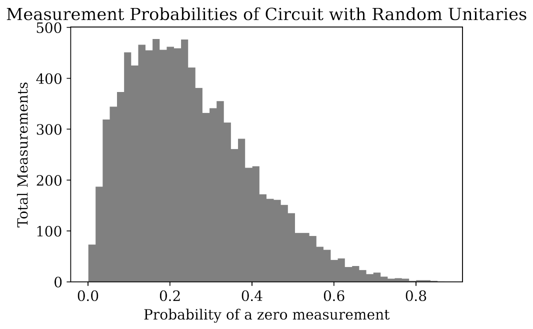

Fig. 8: Random Qudit Angle Encoding Similarities

Overall, the overlap of multiple kernel values can produce informative results that are useful for efficient classification algorithms through the use of higher-dimensional qudit processing. Unlike the usage of the modulo-addition gates representing basis encoded data, which yield discrete levels, angle encoding allows for encoding a continuous range of possible similarities that can be used to classify values with a variable degree of confidence and thus are highly applicable to the emerging area of quantum machine learning algorithms.

VI. Summary

The methods described in this work show that data can be efficiently encoded into higher dimensional qudits represented by a state preparation circuit that implements a quantum memory for qudits. The use of qudits results in an increase in dimensionality of the data as well as an increase in the amount of data in superposition; both of which are advantageous for quantum kernel computations. When the qudit memory is implemented within a quantum kernel computation circuit, we are able to not only find the inner product of a higher dimensional kernel but to also efficiently obtain multiple groups of quantum kernels in linear time. This approach is validated using the Google *cirq* tool and we have shown that multiple kernels can be calculated in parallel. This can be useful for various distance-based classifiers that require the calculation of many higher dimensional kernels. To evaluate the method, we analyzed the possible outcomes of the quantum kernel computational approach using different data encodings. We validated the method using the qudit equivalent of both basis and angle encoded quantum memory and processing. Overall, the similarity metric produced by higher dimensional data can give informative results for multiple quantum states with finer resolution achieved by angle versus basis encoded data sets.

Future work into selecting qudit data encoding methods, synthesizing the qudit circuits, and training parameters will allow for further evaluation of the qudit kernel algorithm. Comparing different classification metrics for qudits and qubits will allow for gaining more insight into the advantages of these techniques. In addition, we can expand the description of our qudit-based kernel algorithm to a multi-qudit system to explore the behavior due to growth of dataset size and its effect on qudit kernel calculations. Synthesizing a multi-qudit kernel requires more calculations to produce the multi-controlled unitary gates comprising the memory. We plan to incorporate additional multi-dimensional synthesis optimization methods into the *MustangQ* quantum circuit synthesis tool [20] to reduce the overall quantum volume of the resulting circuits in order to decrease the effects of spontaneous decoherence during a kernel calculation.

References

[1] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[2] M. Schuld, "Supervised quantum machine learning models are kernel methods," *arXiv:2101.11020v2 [quant-ph]*, pp. 1–26, Apr. 2021.

[3] M. Schuld and F. Petruccione, *Supervised Learning with Quantum Computers*. New York: Springer, 2018.

[4] M. Kues, C. Reimer, P. Roztocki, L. R. Cortés, S. Sciara, B. Wetzel, Y. Zhang, A. Cino, S. T. Chu, B. E. Little, D. J. Moss, L. Caspani, J. Azaña, and R. Morandotti, "On-chip generation of high-dimensional entangled quantum states and their coherent control," *Nature*, vol. 546, no. 7660, pp. 622–626, Jun. 2017. [Online]. Available: https://doi.org/10.1038/nature22986

[5] B. Schölkopf, A. J. Smola, F. Bach *et al.*, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[6] C. Blank, D. K. Park, J.-K. K. Rhee, and F. Petruccione, "Quantum classifier with tailored quantum kernel," *npj Quantum Information*, vol. 6, no. 1, May 2020. [Online]. Available: https://doi.org/10.1038/s41534-020-0272-6

[7] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf, "Quantum fingerprinting," *Physical Review Letters*, vol. 87, no. 16, Sep. 2001. [Online]. Available: https://doi.org/10.1103/physrevlett.87.167902

[8] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, Mar. 2019. [Online]. Available: https://doi.org/10.1038/s41586-019-0980-2

[9] Y. Liu, S. Arunachalam, and K. Temme, "A rigorous and robust quantum speed-up in supervised machine learning," *Nature Physics*, vol. 17, no. 9, pp. 1013–1017, Jul. 2021. [Online]. Available: https://doi.org/10.1038/s41567-021-01287-z

[10] Y. Fan, "A generalization of the Deutsch-Jozsa algorithm to multi-valued quantum logic," in *37th International Symposium on Multiple-Valued Logic (ISMVL'07)*. IEEE, May 2007. [Online]. Available: https://doi.org/10.1109/ismvl.2007.3

[11] K. N. Smith and M. A. Thornton, "Entanglement in higher-radix quantum systems," in *49th International Symposium on Multiple-Valued Logic (ISMVL)*. IEEE, May 2019. [Online]. Available: https://doi.org/10.1109/ismvl.2019.00028

[12] ——, "Higher dimension quantum entanglement generators," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 16, no. 1, pp. 1–21, Oct. 2019. [Online]. Available: https://dl.acm.org/doi/10.1145/3345501

[13] ——, "Entangled state preparation for non-binary quantum computing," in *2019 IEEE International Conference on Rebooting Computing (ICRC'19)*. IEEE, Nov. 2019. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8914717

[14] Y. Wang, Z. Hu, B. C. Sanders, and S. Kais, "Qudits and high-dimensional quantum computing," *Frontiers in Physics*, vol. 8, Nov. 2020. [Online]. Available: https://doi.org/10.3389/fphy.2020.589504

[15] V. Giovannetti, S. Lloyd, and L. Maccone, "Quantum random access memory," *Physical Review Letters*, vol. 100, no. 16, Apr. 2008. [Online]. Available: https://doi.org/10.1103/physrevlett.100.160501

[16] B. C. Travaglione, M. A. Nielsen, H. M. Wiseman, and A. Ambainis, "ROM-based computation: quantum versus classical," *Quantum Information and Computation*, vol. 2, no. 4, 2002.

[17] X.-Q. Zhou, T. C. Ralph, P. Kalasuwan, M. Zhang, A. Peruzzo, B. P. Lanyon, and J. L. O'Brien, "Adding control to arbitrary unknown quantum operations," *Nature Communications*, vol. 2, no. 1, Aug. 2011. [Online]. Available: https://doi.org/10.1038/ncomms1392

[18] M. A. Thornton, D. W. Matula, L. Spenner, and D. M. Miller, "Quantum logic implementation of unary arithmetic operations," in *38th International Symposium on Multiple Valued Logic (ISMVL 2008)*. IEEE, May 2008. [Online]. Available: https://doi.org/10.1109/ismvl.2008.27

[19] F. Mezzadri, "How to generate random matrices from the classical compact groups," *NOTICES of the AMS*, vol. 54, pp. 592–604, 2007.

[20] K. N. Smith and M. A. Thornton, "A quantum computational compiler and design tool for technology-specific targets," in *19th International Symposium on Computer Architecture (ISCA'19)*. ACM/IEEE, Jun. 2019. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3307650.3322262