QUANTUM VISION TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

In this work, quantum transformers are designed and analysed in detail by extending the state-of-the-art classical transformer neural network architectures known to be very performant in natural language processing and image analysis. Building upon the previous work which use parametrised quantum circuits for data loading and orthogonal neural layers, we introduce three types of quantum transformers for training and inference, including a quantum transformer based on compound matrices, which guarantees theoretical advantages of the quantum attention mechanism compared to their classical counterpart both in terms of asymptotic run time and number of model parameters. These quantum architectures can be built using shallow quantum circuits and produce qualitatively different classification models. The three proposed quantum attention layers vary on the spectrum between closely following the classical transformers and exhibiting more quantum characteristics. As building blocks of the quantum transformer, we propose a method for loading a matrix as quantum states as well as two new trainable quantum orthogonal layers adaptable to different levels of connectivity and quality of quantum computers. We performed extensive simulations of the quantum transformers on standard medical image datasets that showed competitive, and at times better performance compared to the classical benchmarks, including the best-in-class classical vision transformers. The trained quantum transformers require fewer parameters as compared to the standard classical benchmarks, confirming the predicted computational advantage of our quantum attention layers with respect to the size of the classified images. Finally, we implemented our quantum transformers on superconducting quantum computers and obtained encouraging results for up to six qubit experiments.

1 INTRODUCTION

Quantum machine learning Biamonte et al. (2017) uses quantum computation in order to provide novel and powerful tools to enhance the performance of classical machine learning algorithms. Some use parametrised quantum circuits to compute quantum neural networks and explore a higher-dimensional optimisation space Cong et al. (2019); Bharti et al. (2021); Cerezo et al. (2020), while others exploit interesting properties native to quantum circuits, such as orthogonality or unitarity Kerenidis et al. (2021); Kiani et al. (2022).

In this work, we focus on transformers, a neural network architecture proposed by Vaswani et al. (2017) which has been applied successfully to both natural language processing Devlin et al. (2018) and visual tasks Dosovitskiy et al. (2020), providing state-of-the-art performance across different tasks and datasets Tay et al. (2020). At a high level, transformers are neural networks that use an *attention* mechanism that takes into account the global context while processing the entire input data element-wise. For visual recognition or text understanding, the context of each element is vital, and the transformer can capture more global correlations between parts of the sentence or the image compared to convolutional neural networks without an attention mechanism Dosovitskiy et al. (2020). In the case of visual analysis for example, images are divided into smaller patches, and instead of simply performing patch-wise operations with fixed size kernels, a transformer learns attention coefficients per patch that weigh the attention paid to the rest of the image by each patch.

In one related work, classical transformer architectures and attention mechanisms have been used to perform quantum tomography Cha et al. (2021). Moreover, a quantum-enhanced transformer for sentiment analysis has been proposed in Di Sipio et al. (2022), and a *self-attention* mechanism for text classification has been used in Li et al. (2022). These works use standard variational quantum circuits to compute the neural networks, and the attention coefficients are calculated classically. A method for using a natively quantum attention mechanism for reinforcement learning has also been proposed in Sanches et al. (2022). Yang & Sun (2022) performed semiconductor defect detection using quantum self-attention, also using standard variational quantum circuits. We also note the proposals of Cong et al. (2019); Henderson et al. (2020) for variational circuits with similarities to convolutional neural networks for general purpose image classification.

The difference between the above-mentioned approaches and the proposed approached of this work mainly stems from the linear algebraic tools we developed which make our quantum circuits much more Noisy Intermediate-Scale Quantum (NISQ)-friendly with proven scalability in terms of run time and model parameters, in contrast to variational quantum circuit approaches used in Farhi & Neven (2018) and Cerezo et al. (2020) which lack proof of scalability Mitarai et al. (2018). This advantage in scalability of our proposed parametrised quantum circuits is made possible by the use of a specific *amplitude encoding* for translating vectors as quantum states, and consistent use of hamming-weight preserving quantum gates instead of general quantum ansatz. In addition to a quantum translation of the classical vision transformer, a novel and natively quantum method is proposed in this work, namely the *compound transformer*, which invokes Clifford Algebra operations that is hard to compute classically.

While we adapted the vision transformer architecture to ease the translation of the attention layer into quantum circuits and benchmarked our methods on vision tasks, the proposed approaches for quantum attention mechanism developed in this work can be easily adapted to apply to other fields of applications, for example in natural language processing where transformers have been proven to be particularly efficient Devlin et al. (2018).

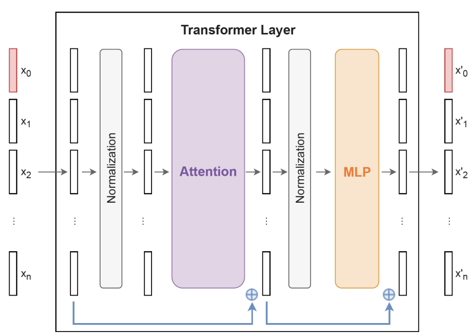

Figure 1: Single transformer layer. Dosovitskiy et al. (2020).

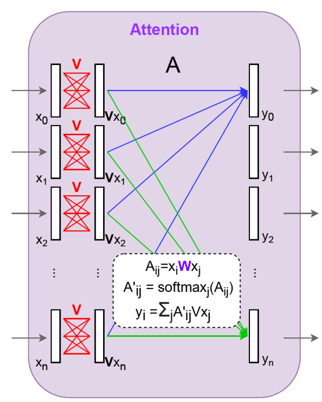

Figure 2: The attention mechanism at the heart of the transformer layer. V and W are trainable.

2 METHOD

The main ingredient in a transformer as introduced by Dosovitskiy et al. (2020) is the *attention layer*, shown in Fig.2. This attention layer is also the focus of this work which seeks to leverage quantum circuit for computational advantages. Given an input image X, we transform the input data into n patches each with dimension of d, and denote each patch i with $x_i \in \mathbb{R}^d$. The trainable weight matrix from the linear fully connected layer at the beginning of each attention layer is denoted by V. The heart of the attention mechanism, i.e. the attention coefficients which weighs each patch x_i to every other patch is denoted by $A_{ij} = x_i W x_j$, where W denotes the second trainable weight matrix.

Based on the architecture shown in Fig.2 we propose three types of quantum transformers (Sections 2.3.1, 2.3.2 and 2.3.4). Section 2.3.3 outlines the approach of combining 2.3.1 and 2.3.2 into one circuit to perform inference on the quantum circuit once the attention coefficients have been trained, while sections 2.3.1, 2.3.2 and 2.3.4 propose 3 distinct quantum architecture for both training and inference.

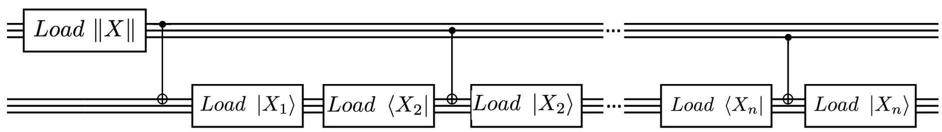

Figure 3: Data loader circuit for a matrix $X \in \mathbb{R}^{n\times d}$. The top register uses n qubits and the vector data loader to load the norms of each row, $(\|X_1\|, \cdots, \|X_n\|)$, to obtain the state $\frac{1}{\|X\|}\sum_{i=1}^{n}\|X_i\|\,|e_i\rangle$. The lower register uses d qubits to load each row $X_i \in \mathbb{R}^d$ sequentially, by applying the vector loader and their adjoint for each row X_i, with CNOTs controlled by the corresponding qubit i of the top register. Each loader on the lower register has depth $O(\log d)$.

The first quantum transformer introduced in Section 2.3.1 implements a trivial attention mechanism which where each patch pays attention only to itself while retaining orthogonality of trained weight matrices Jia et al. (2019). In the second quantum transformer introduced in Section 2.3.2, coined the Orthogonal Transformer, we design a quantum analogue for each of the two main components of a classical attention layer: a linear fully connected layer and the attention matrix to capture the interaction between patches. In Section 2.3.4, the Compound Transformer, which takes advantage of the quantum computer to load input states in superposition, is defined. For each of our quantum methods, we provide theoretical analysis of the computational complexity of the quantum attention mechanisms which is lower compared to their classical counterparts.

The fundamental building blocks for the implementation of a transformer architecture including the matrix data loader, quantum orthogonal layers, and a quantum transformer with trivial attention mechanism are introduced in Sections 2.1, 2.2 and 2.3.1.

2.1 Quantum Data Loaders for Matrices

Loading a whole matrix $X \in \mathbb{R}^{n\times d}$ in a quantum state is a powerful technique for machine learning. We build upon previously published methods to define a quantum data loader for matrices. Johri et al. (2021) designed quantum circuits to load input vectors using *unary* amplitude encoding, more specifically a basis of states of hamming weight 1 where all qubits are in state 0 except one in state 1 is used. The number of required gates to load a vector is $d - 1$. In this work, we extend their approach to build a data loader for matrices (Fig.3) where each row, X_i is loaded in superposition. The required number of gates to load a matrix is $(n - 1) + (2n - 1)(d - 1)$. The resulting state of the matrix loader shown in Fig.3 is a superposition of the form:

$$|X\rangle = \frac{1}{\|X\|}\sum_{i=1}^{n}\sum_{j=1}^{d} X_{ij}\,|e_j\rangle\,|e_i\rangle \tag{1}$$

Data loaders used for loading vectors and matrices are described in detail in the Appendix B.1.

2.2 Quantum Orthogonal Layers

The classical attention layer (Fig.2) starts with a linear fully connected layer, where each input, i.e. patch x_i, is a vector and is multiplied by a weight matrix V. To perform this operation quantumly we generalise the work of Kerenidis et al. (2021), where a quantum orthogonal layer is defined as a quantum circuit applied on a state $|x\rangle$ (encoded in the *unary* basis) to produce the output state $|Vx\rangle$. More precisely, V is the matrix corresponding to the unitary of the quantum orthogonal layer, restricted to the *unary* basis. In addition to the already existing Pyramid circuit (Fig.5) from Kerenidis et al. (2021), we define two new types of quantum orthogonal layers with different levels of expressivity and resource requirements: the butterfly circuit (Fig.4), and the X circuit (Fig.6).

Looking at Table 1, the X circuit is the most suited for noisy hardware. It requires smaller number of gates while maintaining a path from every input qubit to every output qubit. It is also less expressive with a restrained set of possible orthogonal matrices and fewer trainable parameters. The butterfly

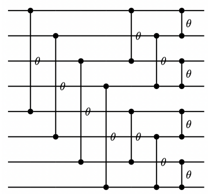

Figure 4: Butterfly

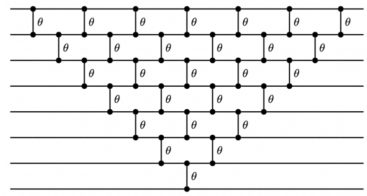

Figure 5: Pyramid

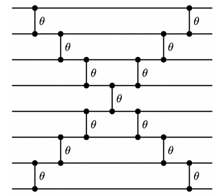

Figure 6: X

Three different types of Quantum Orthogonal Layer Circuits. Vertical lines represent two-qubit Reconfigurable Beam Splitter (RBS) gates each parameterised with one independent angle θ (Appendix B.1).

Circuit	Hardware Connectivity	Depth	# Gates
Pyramid	NN	$2n-3$	$\frac{n(n-1)}{2}$
X	NN	$n-1$	$2n-3$
Butterfly	All-to-all	$log(n)$	$\frac{n}{2}log(n)$

Table 1: Comparison of different quantum orthogonal layer circuits with n qubits. NN stands for Nearest Neighbor connectivity.

circuit requires logarithmic circuit depth, a linear number of gates, and exhibits a higher level of expressivity. It originates from the classical Cooley–Tukey algorithm Cooley & Tukey (1965) used for Fast Fourier Transform. Note that the butterfly circuit requires the ability to apply gates on all possible qubit pairs. More details are outlined in Appendix B.2.

As shown in Kerenidis & Prakash (2022), quantum orthogonal layers can be generalised to work with inputs which encode a vector on a larger basis. Namely, instead of the *unary* basis, where all qubits except one are in state 0, basis of hamming weight k can be used as well. Note a basis of hamming weight k comprises of $\binom{n}{k}$ possible states over n qubits. A vector $x \in \mathbb{R}^{\binom{n}{k}}$ can be loaded as a quantum state $|x\rangle$ using only n qubits. Since the quantum orthogonal layers are hamming weight preserving circuits, the output state from such circuits will also be a vector encoded in the same basis.

2.3 Quantum Transformers

The second component of the attention layer is the interaction between patches (Fig.2). Parameterised quantum circuits are specifically designed to learn the attention coefficients $A_{ij} = x_i^T W x_j$ by performing $x_i^T W x_j$ for a trainable orthogonal matrix W and all pairs of patches x_i and x_j. After that, a non-linearity, for example softmax, is applied to obtain each output y_i. Three different approaches for implementing the quantum attention layer are introduced in the next sections, listed in the order of increasing complexity in terms of quantum resource requirement, which reflect the degree to which quantum circuits are leveraged to replace the attention layer. A comparison between these different quantum methods is provided in Table 2, which is applicable to both training and inference.

Table 2 lists 5 key parameters of the proposed quantum architecture which reflect their theoretical scalability. Classical Vision Transformer has $O(2d^2)$ trainable parameters (See Section A), which can be directly compared with the number of trainable parameters of the proposed quantum approaches. The fixed number of parameters are required for data loading. The circuit depth together with the number of distinct circuits dictate the overall run time of the quantum architectures, which can be compared to the circuit depth of the classical transformer of $O(nd^2 + n^2d)$. The number of distinct circuits per quantum architecture indicate the possibility for each architecture to be processed in parallel, akin to multi-core CPU processing. Notice that the circuit depth shown in Table 2 already includes the data loader.

Algorithm	# Qubits	Circuit Depth	#Trainable parameters	#Fixed parameters	# Distinct circuits
A - Orthogonal Patch-wise	d	$O(\log d)$	$O(d \log d)$	$d-1$	n
B - Quantum Orthogonal Transformer	d	$O(\log d)$	$O(d \log d)$	$3(d-1)$	$n+n^2$
C - Quantum Attention Mechanism	$n+d$	$O(\log n + n \log d + \log d)$	$O(d \log d)$	$n-1+(2n-1)(d-1)$	n
D - Compound Transformer	$n+d$	$O(\log n + n \log d + \log(n+d))$	$O((n+d)\log(n+d)))$	$n-1+(2n-1)(d-1)$	1
Classical Transformer	-	$O(nd^2+n^2d)$	$O(2d^2)$	-	-

Table 2: Comparison of different quantum methods to perform a single attention layer of a transformer network. n and d stand respectively for the number of patches and their individual dimension. All quantum orthogonal layers are implemented using the butterfly circuits.

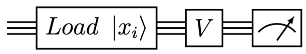

Figure 7: Quantum circuit to perform the matrix multiplication Vx_i (fully connected layer) using a data loader for x_i and a quantum orthogonal layer for V.

2.3.1 Orthogonal Patch-wise Neural Network

The *orthogonal patch-wise neural network* can be thought of as a transformer with a trivial attention mechanism, where each patch pays attention only to itself. As illustrated in Fig 7, each input patch is multiplied by the same trainable matrix V and one circuit per patch is used. Each circuit has d qubits and each patch x_i is encoded in a quantum state with a vector data loader. A quantum orthogonal layer is used to perform multiplication of each patch with V. The output of each circuit is a quantum state encoding Vx_i, a vector which is retrieved through tomography.

The computational complexity of this circuit is calculated as follows: from Section 2.1, a data loader with d qubits has a complexity of $log(d)$ steps. For the orthogonal quantum layer, as shown in Table 1, a butterfly circuit takes $log(d)$ steps, with $\frac{d}{2}log(d)$ trainable parameters. Overall, the complexity is $O(log(d))$ and the number of trainable parameters are of $O(d \log d)$. Since this circuit uses one vector data loader, the number of fixed parameters required is $d-1$.

2.3.2 Quantum Orthogonal Transformer

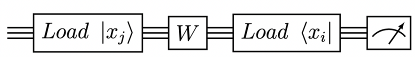

Figure 8: Quantum circuit to compute $|x_i^T W x_j|^2$, a single attention coefficient, using data loaders for x_i and x_j and a quantum orthogonal layer for W.

Looking at Fig.8, each attention coefficient $A_{ij} = x_i^T W x_j$, x_j is calculated first by loading x_j into the circuit with a vector loader followed by a trainable quantum orthogonal layer, W, resulting in the vector Wx_j. Next, an inverse data loader of x_i is applied, creating a state where the probability of measuring 1 on the first qubit is exactly $|x_i^T W x_j|^2 = A_{ij}^2$. Note the square that appears in the quantum circuit is already one type of non-linearity. Using this method, coefficients of A are always positive, which can still be learned during training as we show later in the Section 3. The estimation of A_{ij} (and therefore A'_{ij} if needed, by applying a column-wise *softmax* classically) is repeated for each pair of patches and the same trainable quantum orthogonal layer W. The computational complexity of this quantum circuit is similar to the previous one, with one more data loader.

Putting Figures 7 and 8 together: the quantum circuit presented in Section 2.3.1 is implemented to obtain each Vx_j. At the same time, each attention coefficient $|x_i^T W x_j|^2$ is computed on the quantum circuit, which is further post-processed column-wise with the softmax function to obtain the A'_{ij}. The two parts can then be classically combined to compute each $y_i = \sum_j A'_{ij} V x_j$. For computing $|x_i^T W x_j|^2$, we would require two data loaders ($2 \times (d-1)$ gates) for x_i and x_j, and one Quantum Orthogonal Layer ($d \log d$ gates in the case of Butterfly layer) for W. To obtain Vx_j, we require $d-1$ gates to load each x_j and a Quantum Orthogonal Layer ($d \log d$ gates in the case of Butterfly layer) for the matrix V.

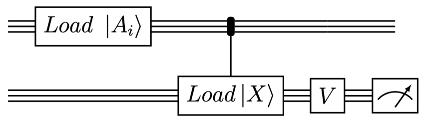

Figure 9: Quantum circuit to directly apply the attention mechanism, given each coefficient in A. The first part of the circuit corresponds to the matrix data loader from Fig.3, where $Load(\|X\|)$ is replaced by $Load(A_i)$. A quantum orthogonal layer from Section 2.2 is used for V.

2.3.3 Direct Quantum Attention

In Section 2.3.2, the output of the attention layer $y_i = \sum_j A'_{ij} V x_j$ is computed classically once the quantities A'_{ij} and $V x_j$ have been computed separately with the help of quantum circuits. During inference, where the matrices V and W have been learnt, and the attention matrix A (or A') is stored classically, *Direct Quantum Attention* implements the attention layer directly on the quantum computer. The matrix data loader from Fig.3 is used to compute each $y_i = \sum_j A_{ij} V x_j$ using a single quantum circuit.

In Fig.9, y_i, which corresponds to the output patch with index i, is computed using a quantum circuit where the qubits are split into two main registers. On the top register, the vector A_i, i^{th} row of the attention matrix A (or A'), is loaded via a vector data loader, as $\sum_j A_{ij} |e_j\rangle |0\rangle$.

Next, on the lower register, as in Fig.3, the data loader for each vector x_i, and their respective adjoint, are applied sequentially, with CNOTs controlled on each qubit i of the top register. This gives the quantum state $\sum_j A_{ij} |e_j\rangle |x_j\rangle$, i.e. the matrix X is loaded with all rows re-scaled according to the attention coefficients. As for any matrix data loader, this requires $(n-1) + (2n-1)(d-1)$ gates with fixed (non-trainable) parameters.

The last step consists of applying the quantum orthogonal layer V that has been trained before on the second register of the circuit. As previously established, this operation performs matrix multiplication between V and the vector encoded on the second register. Since the k^{th} element of the vector $V x_j$ can be written as $\sum_q V_{kq} x_{jq}$, we get:

$$\sum_j A_{ij} |e_j\rangle |V x_j\rangle = \sum_j A_{ij} |e_j\rangle \sum_k (\sum_q V_{kq} x_{jq}) |e_k\rangle = \sum_k \sum_j A_{ij} (\sum_q V_{kq} x_{jq}) |e_j\rangle |e_k\rangle \quad (2)$$

Since $y_i = \sum_j A_{ij} V x_j$, its k^{th} element can be written $y_{ik} = \sum_j A_{ij} (\sum_q V_{kq} x_{jq})$. Therefore, the quantum state at the end of the circuit can be written as $|y_i\rangle = \sum_k y_{ik} |\phi_k\rangle |e_k\rangle$ for some normalised states $|\phi_k\rangle$. Performing tomography on the second register generates the output vector y_i.

This circuit is a more direct method to compute each y_i. Each y_i uses a different A_i in the first part of the circuit. As shown in Table 2, compared with the previous method, this method requires fewer circuits to run, but each circuit requires more qubits and a deeper circuit. To analyse the computational complexity: the first data loader on the top register has n qubit and $\log n$ depth; the following $2n-1$ loaders on the bottom register have d qubits, so $(2n-1)\log d$ depth; and the final quantum orthogonal layer V implemented using a butterfly circuit, has a depth of $\log d$ and $O(d \log d)$ trainable parameters.

2.3.4 Quantum Compound Transformer

Until now, each step of the classical vision transformer has been reproduced closely by quantum linear algebraic procedures. The same quantum tools can also be used in a more natively quantum fashion, while retaining the spirit of the classical transformers, as shown in Fig.10. Instead of loading each patch independently, the Compound Transformer starts with loading all patches in superposition, and then a orthogonal matrix with trainable weights is used to apply attention to each patch in superposition Kerenidis & Prakash (2022). The mathematical formalism behind the Compound Transformer is the second-order compound matrix Horn & Johnson (2012) since both the input vector and the trainable weight matrix are no longer a simple vector or a simple matrix.

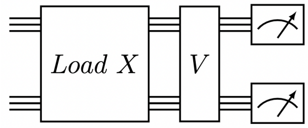

Figure 10: Quantum circuit to execute one attention layer of the Compound Transformer. We use a matrix data loader for X (equivalent to Fig.3) and a quantum orthogonal layer for V applied on both registers.

This quantum circuit has two registers: the top one of size n and the bottom one of size d. The full matrix $X \in \mathbb{R}^{n\times d}$ is loaded into the circuit using the matrix data loader from Section 2.1 with $n + d$ qubits. This could correspond to the entire image, as every image can be split into n patches of size d each. Next, a quantum orthogonal layer V is applied on both registers at the same time. The resulting state is $|Y\rangle = |\mathcal{V}^{(2)}X\rangle$, where $\mathcal{V}^{(2)}$ is the 2^{nd}-order compound matrix of V defined above. Tomography is performed for a state of the form $|Y\rangle = \frac{1}{\|Y\|}\sum_{i=1}^{n}\sum_{j=1}^{d} y_{ij}\,|e_j\rangle\,|e_i\rangle$ which is used to conclude that this quantum circuit produces transformed patches $(y_1, \cdots, y_n) \in \mathbb{R}^{n\times d}$.

To calculate the computational complexity of this circuit: the matrix data loader, detailed in Fig.3 has depth of $\log n + 2n\log d$; the Quantum Orthogonal Layer applied on $n + d$ qubits, has a depth of $log(n+d)$ and $(n+d)log(n+d)$ trainable parameters if implemented using the butterfly circuit. Since this circuit uses exactly one matrix loader, the number of fixed parameters is $(n-1)+(2n-1)(d-1)$.

In order to calculate the cost of performing the same operation on a classical computer, consider the equivalent operation of creating the compound matrix $\mathcal{V}^{(2)}$ by first computing all determinants of the matrix and then performing a matrix-vector multiplication of dimension $\binom{n+d}{2}$, which takes $O((n+d)^4)$ time. Performing this operation on a quantum computer can provide a polynomial speedup with respect to n. More generally, this compound matrix operation on an arbitrary input state of hamming weight k is quite hard to perform classically, since all determinants must be computed, and a matrix-vector multiplication of size $\binom{n+d}{k}$ needs to be applied.

Overall, the compound transformer can replace both the Orthogonal Patch-wise Network (2.3.1) and the Quantum Transformer layer (2.3.2) with one combined operation. The use of compound matrix multiplication makes this approach different from the classical transformers, while retaining some interesting properties with its classical counterpart: patches are weighted in its global context and gradients are shared through the determinants used to generate the compound matrix.

The Compound Transformer operates in a similar spirit as the MLPMixer architecture presented in Tolstikhin et al. (2021). This state-of-the-art architecture used for image classification tasks exchanges information between the different patches without using convolution or attention mechanisms. The underlying mechanism operates in two steps by first mixing the patches and then extracting the patch-wise features using fully connected layers. The Compound Transformer performs both steps at the same time.

3 Experiments

In order to benchmark the proposed methods, we applied them to a set of medical image classification tasks, using both simulations and quantum hardware experiments. MedMNIST, a collection of 12 preprocessed, two-dimensional, open source medical image datasets from Yang et al. (2020; 2021), annotated for classification tasks and benchmarking using a diverse set of classical techniques, is used to provide the complete training and validation data.

3.1 Simulation Setting

Orthogonal Patch-wise Network from Section 2.3.1, Orthogonal Transformer from Section 2.3.2, and Compound Transformer from Section 2.3.4 were trained via simulation, along with two baseline methods. The first baseline is the Vision Transformer from Dosovitskiy et al. (2020), which has been successfully applied to different image classification tasks and is described in detail in Appendix A. The second baseline is the Orthogonal Fully-Connected Neural Network (OrthoFNN), a quantum method without attention layer that has been previous trained on the RetinaMNIST dataset in Mathur et al. (2021). For each of the five architectures, one model was trained on each dataset of MedMNIST and validated using the same validation method as in Yang et al. (2020; 2021).

To ensure comparable evaluations between the five neural networks, similar architectures were implemented for all five. The benchmark architectures all comprise of three parts: pre-processing, features extraction, and post-processing. The first part is classical and pre-processes the input image of size 28×28 by extracting 16 patches ($n = 16$) of size 7×7. We then map every patch to the dimension of the feature extraction part of the neural network, by using a fully connected neural network layer to turn the 7×7 patch is a 16 dimensional feature space ($d = 16$). Note that this first feature extraction part is a single fully connected layer trained in conjunction to the rest of the architecture. For the OrthoNN networks, used as our quantum baseline, one patch of size 16 was extracted from the complete input image using a fully connected neural network layer of size 784×16. This fully connected layer is also trained in conjunction to the quantum circuits. The second part of the common architecture transforms the extracted features by applying a sequence of 4 attention layers on the extracted patches, which maintain the dimension of the layer. Moreover, the same gate layout, i.e. the butterfly circuit, is used for all circuits that compose the quantum layers. Finally, the last part of the neural network is classical, which linearly projects the extracted features and outputs the predicted label.

3.2 SIMULATION RESULTS

Network	PathMNIST		ChestMNIST		DermaMNIST		OCTMNIST		PneumoniaMNIST		RetinaMNIST	
	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC
OrthoFNN (baseline)	0.939	0.643	0.701	0.947	0.883	0.719	0.819	0.516	0.950	0.864	0.731	0.548
OrthoPatchWise	0.953	0.713	0.692	0.947	0.898	0.730	0.861	0.554	0.945	0.867	0.739	0.560
VisionTransformer (baseline)	0.957	0.755	**0.718**	**0.948**	0.895	0.727	**0.879**	**0.608**	**0.957**	**0.902**	0.736	0.548
OrthoTransformer	**0.964**	**0.774**	0.703	0.947	0.891	0.719	0.875	0.606	0.947	0.885	**0.745**	0.542
CompoundTransformer	0.957	0.735	0.698	0.947	**0.901**	**0.734**	0.867	0.545	0.947	0.885	0.740	**0.565**
Network	BreastMNIST		BloodMNIST		TissueMNIST		OrganAMNIST		OrganCMNIST		OrganSMNIST	
	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC
OrthoFNN (baseline)	0.815	0.821	0.972	0.820	0.819	0.513	0.916	0.636	0.923	0.672	0.875	0.481
OrthoPatchWise	0.830	0.827	0.984	0.866	0.845	0.549	0.973	0.786	0.976	0.805	0.941	0.640
VisionTransformer (baseline)	0.824	0.833	**0.985**	**0.888**	**0.880**	**0.596**	0.968	0.770	0.970	0.787	0.934	0.620
OrthoTransformer	0.770	0.744	0.982	0.860	0.856	0.557	0.968	0.763	0.973	0.785	**0.946**	0.635
CompoundTransformer	**0.859**	**0.846**	**0.985**	0.870	0.841	0.544	**0.975**	**0.789**	**0.978**	**0.819**	0.943	**0.647**

Table 3: Performance analysis using AUC and ACC on each test dataset of MedMNIST of our quantum architectures (Orthogonal PatchWise, Orthogonal Transformer and Compound Transformer) compared to the classical (Vision Transformer Dosovitskiy et al. (2020)) and quantum (Orthogonal FNN Kerenidis et al. (2021)) baselines described in Section 3.

A summary of the simulation results is shown in Table 3 where the area under receiver operating characteristic (ROC) curve (AUC) and the accuracy (ACC) are reported as evaluation metrics. A full comparison with the classical benchmark provided by Yang et al. (2020) is given in Appendix D, Table 6.

From Table 3, we observe that Vision Transformer, Orthogonal Transformer, and Compound Transformer architectures outperform the Orthogonal Fully-Connected and Orthogonal Patch-wise neural networks for all 12 tasks. This may be due to the fact that the latter two do not contain on any mechanism that exchange information across the patches, confirming the effectiveness of the attention mechanism to extract useful features from images. Second, all three quantum transformer networks developed in this work provide competitive performances compared to the two benchmark methods and outperform the two benchmark methods on 7 out of 12 MedMNIST datasets.

Moreover, comparisons can be made with regard to the number of trainable parameters used by each architecture. Table 5 presents a resource analysis for the quantum circuits that were simulated per layer. E.g. the Compound Transformer requires 80 trainable parameters compared to the 512 required by the Classical Vision Transformer. Note again this resource analysis focuses on the attention layer of each transformer network, and does not include parameters used for preprocessing the images (see Appendix A), other parts found in the transformer layer, nor the single layer used in the final classification (Fig.11), which are common to all simulated methods.

Overall, our quantum transformers have reached comparable levels of accuracy compared to the classical equivalent transformers, while using a smaller number of trainable parameters, providing confirmation of our theoretical predictions on a small scale. Summary of the hardware experiments are listed in Table 4 with details to be found in Appendix C.3.

Model	Classical (JAX)		IBM Simulator		IBM Hardware	
	AUC	ACC	AUC	ACC	AUC	ACC
Google AutoML (Best in Yang et al. (2021))	0.750	53.10 %	-	-	-	-
VisionTransformer (classical benchmark)	0.736	55.75 %	-	-	-	-
OrthoPatchWise (Pyramid Circuit)	0.738	56.50 %	0.731	54.75 %	0.727	51.75 %
Ortho Transformer (Pyramid Circuit)	0.729	55.00 %	0.715	55.00 %	0.717	54.50 %
Ortho Transformer with Quantum Attention	0.749	56.50 %	0.743	55.50 %	0.746	55.00 %
CompoundTransformer (X Circuit)	0.729	56.50 %	0.683	56.50 %	0.666	45.75 %
CompoundTransformer (\ Circuit)	0.716	55.75 %	0.718	55.50 %	0.704	49.00 %

Table 4: Hardware Results for RetinaMNIST using various models. Note that "\ Circuit" (see Appendix C.3.2) contains a single diagonal of trainable RBS gates. Classical (JAX): classical code run by JAX, equivalent to quantum operations. IBM Simulator: code compiled to run on actual IBM hardware and executed using their Aer Simulator.

Model	Qubits	Number of Fixed Parameters for Data Loading	Number of Trainable Parameters	Circuit Depth	Number of Distinct Circuits
Orthogonal PatchWise	16	15	32	9	16
Orthogonal Transformer	16	45	64	9 & 13	272
Compound Transformer	32	480	80	150	1

Table 5: Resource analysis on a single attention layer used for the MedMNIST simulations (Section 3.1). The number of trainable parameters for each attention layer of each quantum network are to be compared with the 512 trainable parameters per attention layer of the classical Vision Transformers. Note that the *Quantum Transformer* is using two different types of circuits per layer.

4 Conclusion

In this work, three different quantum transformers are presented: Orthogonal Patchwise implements trivial attention mechanism and is the simplest approach; Orthogonal Transformer is the most similar to the classical transformers; Compound Transformer steps away from the classical architecture with a quantum-native linear algebraic operation that cannot be efficiently done classically: multiplying a vector with a higher-dimensional *compound* matrix. Inside all these quantum transformers are the quantum orthogonal layers, which efficiently apply matrix multiplication on vectors encoded on specific quantum basis states. All circuits implementing orthogonal matrix multiplication can be trained using backpropagation detailed in Kerenidis et al. (2021).

As shown in Table 2, our quantum circuits show definite advantage in terms of computational complexity of the attention layer. In addition to theoretical analysis, we performed extensive numerical simulations and quantum hardware experiments, which shows that our quantum circuits can classify the small MedMNIST images just as well as or at times better than the state-of-the-art classical methods (Table 3). Our quantum methods have the potential to address over-fitting issues by using a small number of parameters.

While the run time of the quantum fully connected and attention layers has been theoretically proven to be advantageous, this is hard to observe on the current quantum computers due to their limited size, high level of noise, and latency of cloud access. From our hardware experiments, it can be observed that results from the current hardware become too noisy as soon as the number of qubits or the size of the quantum circuit increased (see Table 4). Hence the exact number of parameters used to run the quantum circuits is expected to change with the availability of larger quantum computers, which will allow for larger quantum operations and eliminate usage of classical preprocessing to downsize the input images.

Overall, our results are encouraging and show the benefit of using trainable quantum circuits to perform efficient linear algebra operations. This approach allows for much better control over the size of the Hilbert space that is explored by the model and provides models that are both expressive and trainable.

REPRODUCIBILITY STATEMENT

It is fairly straightforward to reproduce our results. We have used publicly available MedMNIST datasets, the benchmarks of which are also available online. The steps for preprocessing the data points to make it into patches have been well defined in Appendix A. Implementing a quantum orthogonal layer can be done both on quantum hardware (or simulator) by defining the circuit and by classically performing the quantum operations. We have used JAX to perform our classical simulations and IBM's simulator and hardware for the quantum experiments. We tried to find decent enough hyperparameters for all our models but we do not claim to have found the best ones.

REFERENCES

Kishor Bharti, Alba Cervera-Lierta, Thi Ha Kyaw, Tobias Haug, Sumner Alperin-Lea, Abhinav Anand, Matthias Degroote, Hermanni Heimonen, Jakob S Kottmann, Tim Menke, et al. Noisy intermediate-scale quantum (nisq) algorithms. *arXiv preprint arXiv:2101.08448*, 2021.

Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. *arXiv preprint arXiv:2012.09265*, 2020.

Peter Cha, Paul Ginsparg, Felix Wu, Juan Carrasquilla, Peter L McMahon, and Eun-Ah Kim. Attention-based quantum tomography. *Machine Learning: Science and Technology*, 3(1):01LT01, 2021.

Iris Cong, Soonwon Choi, and Mikhail D Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, 2019.

James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Riccardo Di Sipio, Jia-Hong Huang, Samuel Yen-Chi Chen, Stefano Mangini, and Marcel Worring. The dawn of quantum natural language processing. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8612–8616. IEEE, 2022.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Edward Farhi and Hartmut Neven. Classification with quantum neural networks on near term processors. *arXiv:1802.06002*, 2018.

Maxwell Henderson, Samriddhi Shakya, Shashindra Pradhan, and Tristan Cook. Quanvolutional neural networks: powering image recognition with quantum circuits. *Quantum Machine Intelligence*, 2(1):1–9, 2020.

Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

Kui Jia, Shuai Li, Yuxin Wen, Tongliang Liu, and Dacheng Tao. Orthogonal deep neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

S Johri, S Debnath, A Mocherla, A Singh, A Prakash, J Kim, and I Kerenidis. Nearest centroid classification on a trapped ion quantum computer. *npj Quantum Information (to appear), arXiv:2012.04145*, 2021.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pp. 5156–5165. PMLR, 2020.

Iordanis Kerenidis and Anupam Prakash. Quantum machine learning with subspace states. *arXiv preprint arXiv:2202.00054*, 2022.

Iordanis Kerenidis, Jonas Landman, and Natansh Mathur. Classical and quantum algorithms for orthogonal neural networks. *arXiv:2106.07198*, 2021.

Bobak Kiani, Randall Balestriero, Yann Lecun, and Seth Lloyd. projunn: efficient method for training deep networks with unitary matrices. *arXiv preprint arXiv:2203.05483*, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Guangxi Li, Xuanqiang Zhao, and Xin Wang. Quantum self-attention neural networks for text classification. *arXiv preprint arXiv:2205.05625*, 2022.

Natansh Mathur, Jonas Landman, Yun Yvonna Li, Martin Strahm, Skander Kazdaghli, Anupam Prakash, and Iordanis Kerenidis. Medical image classification via quantum neural networks. *arXiv preprint arXiv:2109.01831*, 2021.

Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii. Quantum circuit learning. *Physical Review A*, 98(3):032309, 2018.

Hyeonwoo Noh, Tackgeun You, Jonghwan Mun, and Bohyung Han. Regularizing deep neural networks by noise: Its interpretation and optimization. *NeurIPS*, 2017.

Fabio Sanches, Sean Weinberg, Takanori Ide, and Kazumitsu Kamiya. Short quantum circuits in reinforcement learning policies for the vehicle routing problem. *Physical Review A*, 105(6):062403, 2022.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Computing Surveys (CSUR)*, 2020.

Ilya O. Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision. In *NeurIPS*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Jiancheng Yang, Rui Shi, and Bingbing Ni. Medmnist classification decathlon: A lightweight automl benchmark for medical image analysis. *arXiv preprint arXiv:2010.14925*, 2020.

Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2: A large-scale lightweight benchmark for 2d and 3d biomedical image classification. *arXiv preprint arXiv:2110.14795*, 2021.

Yuan-Fu Yang and Min Sun. Semiconductor defect detection by hybrid classical-quantum deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2323–2332, 2022.

Xue Ying. An overview of overfitting and its solutions. In *Journal of Physics: Conference Series*, volume 1168, pp. 022022. IOP Publishing, 2019.

A VISION TRANSFORMERS

Here, the details of a classical Vision Transformers introduced by Dosovitskiy et al. (2020) are outlined. Some slight changes in the architecture have been made to ease the correspondence with quantum circuits. We also introduce important notations that will be reused in the quantum methods.

The transformer network starts by decomposing an image into *patches* and pre-processing the set of patches to map each one into a vector, as shown in Fig.12. The initial set of patches is enhanced with an extra vector of the same size as the patches, called *class embedding*. This *class embedding* vector is used at the end of the network, to feed into a fully connected layer that yields the output (see Fig.11). We also include one trainable vector called *positional embedding*, which is added to each vector. At the end of this pre-processing step, we obtain the set of n vectors of dimension d, denoted x_i to be used in the next steps.

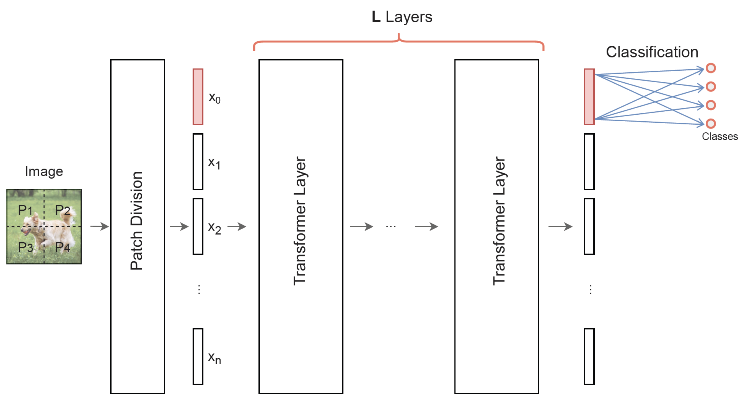

Figure 11: Representation of a Vision Transformer network architecture. Dosovitskiy et al. (2020).

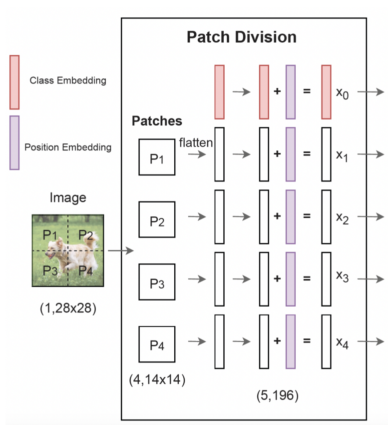

Figure 12: Patch division part of the transformer network for an image split into four patches. Note Class Embedding and Position Embedding are trainable vectors. Dosovitskiy et al. (2020).

Next, feature extraction is performed using a transformer layer Vaswani et al. (2017); Dosovitskiy et al. (2020) which is repeated L times, as shown in Fig.1. Within the transformer layer, we first apply layer normalisation over all patches x_i, and then apply the attention mechanism detailed in Fig.2. After this part, we obtain a state to which we add the initial input vectors before normalisation in an operation called *residual* layer, represented by the blue arrow in Fig.1, followed by another layer normalisation. After this, we apply a Multi Layer Perceptron (MLP), which consists of multiple fully connected linear layers for each vector that result in same-sized vectors. Again, we add the residual from just before the last layer normalisation, which is the output of one transformer layer.

After repeating the transformer layer L times, we finally take the vector corresponding to the *class embedding*, that is the vector corresponding to x_0, in the final output and apply a fully connected layer of dimension ($d \times$ number of classes) to provide the final classification result (see Fig.11). It is important to observe here that we only use the first vector outcome in the final fully connected layer to do the classification (therefore the name "class embedding").

Looking inside the attention mechanism (see Fig.2), we start by using a fully connected linear layer with trainable weights V to calculate for each patch x_i the feature vector Vx_i. Then to calculate the attention coefficients, we use another trainable weight matrix W and define the attention given by patch x_i to patch x_j as $x_i^T W x_j$. Next, for each patch x_i, we get the final extracted features as the weighted sum of all feature vectors Vx_j where the weights are the *attention coefficients*. This is equivalent to performing a matrix multiplication with a matrix A defined by $A_{ij} = x_i^T W x_j$. Note, in classical transformer architecture, a column-wise *softmax* is applied to all A_{ij} and attention coefficients $A'_{ij} = softmax_j(A_{ij})$ is used instead. Overall, the attention mechanism makes use of $2d^2$ trainable parameters, evenly divided between V and W each of size $d \times d$.

In fact, the above description is a slight variant from the original transformers proposed in Vaswani et al. (2017), where the authors used two trainable matrices to obtain the attention coefficients instead of one (W) in this work. This choice was made to simplify the quantum implementation but could be extended to the original proposal using the same quantum tools.

Computational complexity of classical attention mechanism depends mainly on the number of patches n and their individual dimension d: the first patch-wise matrix multiplication with the matrix $V \in \mathbb{R}^{d\times d}$ takes $O(nd^2)$ steps, while the subsequent multiplication with the large matrix A' takes $O(n^2 d)$. Obtaining A' from W requires $O(nd^2)$ steps as well. Overall, the complexity is $O(nd^2 + n^2 d)$. In classical deep learning literature, the emphasis is made on the second term, which is usually the most costly. Note that a recent proposal Katharopoulos et al. (2020) proposes a different attention mechanism as a linear operation that only has a $O(nd^2)$ computational complexity.

We compare the classical computational complexity with those of our quantum methods in Table 2. These running times have an real impact on both training and inference, as they measure how the time to perform each layer scales with the number and dimension of the patches.

B Quantum Tools (Extended)

B.1 Quantum Data Loaders for Matrices

RBS gates implement the following unitary:

$$RBS(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3)$$

In order to perform a machine learning task with a quantum computer, classical data (a vector, a matrix) needs to be loaded into the quantum circuit. The technique we choose for this task is called *amplitude encoding*, which uses the classical scalar component of the data as amplitudes of a quantum state made of d qubits. In particular we build upon previous methods to define quantum data loaders for matrices, as shown in Fig.3.

Johri et al. (2021) proposes three different circuits to load a vector $x \in \mathbb{R}^d$ using $d-1$ gates for a circuit depth ranging from $O(log(d))$ to $O(d)$ as desired (see Fig.13). These data loaders use the *unary* amplitude encoding, where a vector $x = (x_1, \cdots, x_d)$ is loaded in the quantum state $|x\rangle = \frac{1}{\|x\|}\sum_{i=1}^{d} x_i |e_i\rangle$ where $|e_i\rangle$ is the quantum state with all qubits in 0 except the i^{th} one in state 1 (e.g. $|0\cdots010\cdots0\rangle$). The circuit uses RBS gates: a parametrised two-qubit gate given by Eq.3.

The $d-1$ parameters θ_i of the RBS gates are classically pre-computed to ensure that the output of the circuit is indeed $|x\rangle$.

We require a data loader for matrices. Given a matrix $X \in \mathbb{R}^{n\times d}$, instead of loading a flattened vector, rows X_i are loaded in superposition. As shown in Fig.3, on the top qubit register, we first

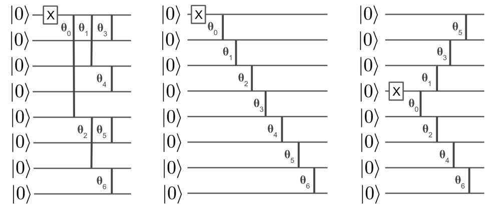

Figure 13: Three possible data loaders for d-dimensional vectors ($d = 8$). From left to right: the parallel, diagonal, and semi-diagonal circuit have respectively a circuit depth of $log(d)$, d, and $d/2$. The X gate represent the Pauli X gate, and the vertical lines represent RBS gates with tunable parameters.

load the vector $(\|X_1\|, \cdots, \|X_n\|)$ made of the norms of each row, using a data loader for a vector and obtain a state $\frac{1}{\|X\|}\sum_{i=1}^{n}\|X_i\|\,|e_i\rangle$. Then, on a lower register, we are sequentially loading each row $X_i \in \mathbb{R}^d$. To do so, we use vector data loaders and their adjoint, as well as CNOTs controlled on the i^{th} qubit of the top register. The resulting state is a superposition of the form:

$$|X\rangle = \frac{1}{\|X\|}\sum_{i=1}^{n}\sum_{j=1}^{d} X_{ij}\,|e_j\rangle\,|e_i\rangle$$

One immediate application of data loaders that construct amplitude encodings is the ability to perform fast inner product computation with quantum circuits. Applying the inverse data loader of x_i after the regular data loader of x_j effectively creates a state of the form $\langle x_i, x_j\rangle\,|e_1\rangle + |G\rangle$ where $|G\rangle$ is a garbage state. The probability of measuring $|e_1\rangle$, which is simply the probability of having a 1 on the first qubit, is $|\langle x_i, x_j\rangle|^2$. Techniques to retrieve the sign of the inner product have been developed in Mathur et al. (2021).

B.2 Quantum Orthogonal Layers

In this section, we outline the concept of quantum orthogonal layers used in neural networks, which generalises the work in Kerenidis et al. (2021). These layers correspond to parametrised circuits of n qubits made of RBS gates. More generally, RBS gates preserve the number of ones and zeros in any basis state: if the input to a quantum orthogonal layer is a vector in unary amplitude encoding, the output will be another vector in unary amplitude encoding. Similarly, if the input quantum state is a superposition of only basis states of hamming weight 2, so is the output quantum state. This output state is precisely the result of a matrix-vector product, where the matrix is the unitary matrix of the quantum orthogonal layer, restricted to the basis used. Therefore, for unary basis, we consider a $n \times n$ matrix W instead of the full $2^n \times 2^n$ unitary. Similarly for the basis of hamming weight two, we can restrict the unitary to a $\binom{n}{2} \times \binom{n}{2}$ matrix. Since the reduced matrix conserves its unitary property and has only real values, these are orthogonal matrices. More generally, we can think of such hamming weight preserving circuits with n qubits as block-diagonal unitaries that act separately on $n + 1$ subspaces, where the k-th subspace is defined by all computational basis states with hamming weight equal to k. The dimension of these subspaces is equal to $\binom{n}{k}$.

There exist many possibilities for building a quantum orthogonal layer, each with different properties. The Pyramid circuit, proposed in Kerenidis et al. (2021), is composed of exactly $n(n-1)/2$ RBS gates. This circuit requires only adjacent qubit connectivity, which is the case for most superconducting qubit hardware. More precisely, the set of matrices that are equivalent to the quantum orthogonal layers with pyramidal layout is exactly the Special Orthogonal Group, made of orthogonal matrices with determinant equal to $+1$. We have showed that by adding a final Z gate on the last qubit would allow having orthogonal matrices with -1 determinant. The pyramid circuit is therefore very general and cover all the possible orthogonal matrices of size $n \times n$.

The two new types of quantum orthogonal layers we have introduced are the butterfly circuit (Fig.4), and the X circuit (Fig.6) (Section 2.2).

There exists a method Kerenidis et al. (2021) to compute the gradient of each parameter θ_i in order to update them. This backpropagation method for the pyramid circuit takes time $O(n^2)$, corresponding to the number of gates, and provided a polynomial improvement in run time compared to the previously known orthogonal neural network training algorithms Jia et al. (2019). The exact same method developed for the pyramid circuit can be used to perform quantum backpropagation on the new circuits introduced in this paper. The run time also corresponds to the number of gates, which is lower for the butterfly and X circuits. See Table 1 for full details on the comparison between the three types of circuits.

C Medical Image Classification via Quantum Transformers (Extended)

C.1 Datasets

In order to benchmark our models, we used MedMNIST, a collection of 12 pre-processed, two-dimensional medical image open datasets Yang et al. (2020; 2021). The collection has been standardised for classification tasks on 12 different imaging modalities, each with medical images of 28×28 pixels. All three quantum transformers and two benchmark methods were trained and validated on all 12 MedMNIST datasets. For the hardware experiments, we focused on one dataset, RetinaMNIST. The MedMNIST dataset was chosen for our benchmarking efforts due to its accessible size for simulations of the quantum circuits and hardware experiments, while being representative of one important field of computer vision application: classification of medical images.

C.2 Simulations

First, simulations of our models are performed on the 2D MedMNIST datasets and demonstrate that the proposed quantum attention architecture reaches accuracy comparable to and at times better than the various standard classical models. Next, the setting of our simulations are described and the results compared against those reported in the AutoML benchmark performed by the authors in Yang et al. (2021).

C.2.1 Simulation setting MedMNIST

The JAX package Bradbury et al. (2018) was used to efficiently simulate the complete training procedure of the five benchmark architectures. The experimental hyperparameters used in Yang et al. (2021) were replicated for our benchmark: every model is trained using the cross-entropy loss with the Adam optimiser Kingma & Ba (2015) for 100 epochs, with batch size of 32 and a learning rate of 10^{-3} that is decayed by a factor of 0.1 after 50 and 75 epochs.

The 5 different neural networks were trained over 3 random seeds, and the best overall performance for each one of them was selected. The evaluation procedure is similar to the AutoML benchmark in Yang et al. (2020; 2021), and the benchmark results are shown in Table 3 where the area under receiver operating characteristic (ROC) curve (AUC) and the accuracy (ACC) are reported as evaluation metrics. A full comparison with the classical benchmark provided by Yang et al. (2020) is given in (Appendix D, Table 6).

C.2.2 Simulation results MedMNIST

From Table 3, we observe that Quantum Orthogonal and Compound Transformer architectures outperform the Orthogonal Fully-Connected and Orthogonal Patch-wise neural networks most of the time. This may be due to the fact that the latter do not rely on any mechanism that exchange information across the patches. Second, all quantum neural networks provide very competitive performances compared to the AutoML benchmark and outperform their classical counterparts on 7 out of 12 MedMNIST datasets.

Moreover, comparisons can be made with regard to the number of parameters used by each architecture, in particular for feature extraction. Table 5 presents a resource analysis for the quantum circuits that were simulated, per layer. It includes the number of qubits, the number of gates with trainable parameters, and the number of gates with fixed parameters used for loading the data. The table shows that our quantum architectures have a small number of trainable parameters per layer. The global count for each quantum method is as follows.

- Orthogonal Patch-wise Neural Network: 32 parameters per circuit, 16 circuits per layer which use the same 16 parameters, and 4 layers, for a total of 128 trainable parameters.
- Quantum Orthogonal Transformer: 32 parameters per circuit, 17 circuits which use the same 16 parameters and another 289 circuits which use another set of 16 parameters per layer, and 4 layers, for a total of 256 trainable parameters.
- Compound Transformer: 80 parameters per circuit, 1 circuit per layer, and 4 layers, for a total of 320 trainable parameters.

These numbers are to be compared with the number of trainable parameters in the classical Vision Transformer that is used as a baseline. As stated in Section A, each classical attention layer requires $2d^2$ trainable parameters, which in the simulations performed here corresponds to 512. Note again this resource analysis focuses on the attention layer of the each transformer network, and does not include parameters used for the preprocessing of the images (see Section C.2.1), as part of other transformer layers (Fig.1), and for the single layer used in the final classification (Fig.11), which are common in all cases.

More generally, performance of other classical neural network models provided by the authors of MedMNIST is compared to our approaches in Table 6 found in the Appendix. Some of these classical neural networks reach somewhat better levels of accuracy, but are known to use an extremely large number of parameters. For instance, the smallest reported residual network has approximately a total number of 10^7 parameters, and the automated machine learning algorithms train numerous different architectures in order to reach that performance.

Based on the results of the simulations in this section, quantum transformers are able to train across a number different of classification tasks, deliver performances that are highly competitive and sometimes better than the equivalent classical methods.

C.3 Quantum Hardware Experiments

Quantum hardware experiments were performed on one specific dataset: RetinaMNIST. It has 1080 images for training, 120 images for validation, and 400 images for testing. Each image contains 28×28 RGB pixels. Each image is classified into 1 of 5 classes (ordinal regression).

C.3.1 Hardware Description

The hardware demonstration was performed on two different superconducting quantum computers provided by IBM, with the smaller experiments performed on the 16-qubit *ibmq_guadalupe* machine (see Fig.14) and the larger ones on the 27-qubit *ibm_hanoi* machine. Results are reported here from experiments with four, five and six qubits; experiments with higher numbers of qubits, which entails higher numbers of gates and depth, did not produce meaningful results.

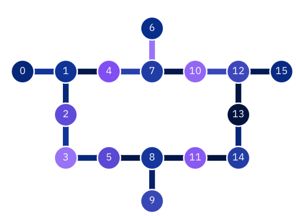

Figure 14: Connectivity of the 16-qubit ibmq_guadalupe quantum computer.

Note that the main sources of noise are the device noise and the finite sampling noise. In general, noise is undesirable during computations. In the case of a neural network, however, noise may not be as troublesome: noise can help escape local minima Noh et al. (2017), or act as data augmentation to avoid over-fitting. In classical deep learning, noise is sometimes artificially added for these purposes Ying (2019). Despite this, when the noise is too large, we also see a drop in the accuracy.

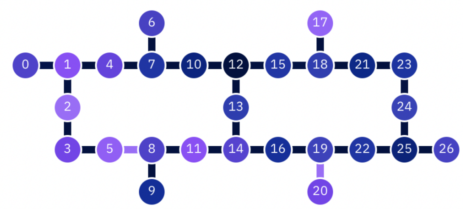

Figure 15: Connectivity of the 27-qubit ibm_hanoi quantum computer.

C.3.2 HARDWARE RESULTS

Hardware experiments were performed with four, five and six qubits to push the limits of the current hardware, in terms of both the number of qubits and circuit depth. Three quantum proposals were run: the Orthogonal Patch-wise network (from Section 2.3.1), the Quantum Orthogonal transformers (from Sections 2.3 and 2.3.3) and finally the Quantum Compound Transformer (from Section 2.3.4).

Each quantum model was trained using a JAX-based simulator, and inference was performed on the entire test dataset of 400 images of the RetinaMNIST on the IBM quantum computers.

The first model, the Orthogonal Patch-wise neural network, was trained using 16 patches per image, 4 features per patch, and one 4×4 orthogonal layer, using a 4-qubit pyramid as the orthogonal layer. The experiment used 16 different quantum circuits of 9 RBS gates per circuit per image. The result was compared with an equivalent classical (non-orthogonal) patch-wise neural network, and a small advantage in accuracy for the quantum native method could be reported.

The second model, the Quantum Orthogonal Transformer, used 4 patches per image, 4 features per patch, and an attention mechanism with one 4×4 orthogonal layer and trainable attention coefficients. 4-qubit pyramids were used as orthogonal layers. The experiment used 25 different quantum circuits of 12 RBS gates per circuit per image and 15 different quantum circuits of 9 RBS gates per circuit per image.

The third set of experiments ran the Orthogonal Transformer with the quantum attention mechanism. We used 4 patches per image, 4 features per patch, and a quantum attention mechanism that paid attention to only the neighbouring patch, thereby using a 5-qubit quantum circuit with the X as the orthogonal layer. The experiment used 12 different quantum circuits of 14 RBS gates and 2 $CNOT$s per circuit per image.

The last two quantum proposals were compared with a classical transformer network with a similar architecture and demonstrated similar level of accuracy.

Finally, the fourth experiment was performed on the *ibmq_hanoi* machine with 6 qubits, with the Compound Transformer, using 4 patches per image, 4 features per patch, and one orthogonal layer using the X layout. The hardware results were quite noisy with the X layer, therefore the same experiments were performed with a further-reduced orthogonal layer named the "\Circuit": half of a X Circuit (Fig.6) where only one diagonal of RBS gates is kept, and which reduced the noise in the outcomes. The experiment used 2 different quantum circuits of 18 RBS gates and 3 $CNOT$s per circuit per image.

Note that with the restriction to states with a fixed hamming weight, strong error mitigation techniques become available. Indeed, as we expect to obtain only quantum superpositions of unary states or states with hamming weight 2 in the case of Compound Transformers, at every layer, every measurement can be processed to discard the ones that have a different hamming weight *i.e.* states with more than one (or two) qubit in state $|1\rangle$. This error mitigation procedure can be applied efficiently to the results of a hardware demonstration, and has been used in the results presented in this paper.

The conclusion from the hardware experiments is that all quantum proposals achieve state-of-the-art test accuracy, comparable to classical networks. In particular, the quantum Compound methods on the simulator are notably more efficient than the classical networks, and they have no efficient classical equivalent. However, the current hardware is often too noisy to achieve similar performance, even with a low number of qubits.

D EXTENDED PERFORMANCE ANALYSIS

We add our results to the already existing results on the MedMNIST Yang et al. (2021) datasets in the table 6 below.

Network	PathMNIST		ChestMNIST		DermaMNIST		OCTMNIST		PneumoniaMNIST		RetinaMNIST	
	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC
ResNet-18 (28)	0.983	0.907	0.768	0.947	0.917	0.735	0.943	0.743	0.944	0.854	0.717	0.524
ResNet-18 (224)	0.989	0.909	0.773	0.947	0.920	0.754	0.958	0.763	0.956	0.864	0.710	0.493
ResNet-50 (28)	0.990	0.911	0.769	0.947	0.913	0.735	0.952	0.762	0.948	0.854	0.726	0.528
ResNet-50 (224)	0.989	0.892	0.773	0.948	0.912	0.731	0.958	0.776	0.962	0.884	0.716	0.511
auto-sklearn	0.934	0.716	0.649	0.779	0.902	0.719	0.887	0.601	0.942	0.855	0.690	0.515
auto-keras	0.959	0.834	0.742	0.937	0.915	0.749	0.955	0.763	0.947	0.878	0.719	0.503
auto-ml	0.944	0.728	0.914	0.948	0.914	0.768	0.963	0.771	0.991	0.946	0.750	0.531
VisionTransformer	0.957	0.755	0.718	0.947	0.895	0.727	0.923	0.830	0.957	0.902	0.749	0.562
OrthoFNN	0.939	0.643	0.701	0.947	0.883	0.719	0.819	0.516	0.950	0.864	0.731	0.548
OrthoPatchWise	0.953	0.713	0.692	0.947	0.898	0.730	0.861	0.554	0.945	0.867	0.739	0.560
OrthoTransformer	0.964	0.774	0.703	0.947	0.891	0.719	0.875	0.606	0.947	0.885	0.745	0.542
CompoundTransformer	0.957	0.735	0.698	0.947	0.901	0.734	0.867	0.545	0.947	0.885	0.740	0.565

Network	BreastMNIST		BloodMNIST		TissueMNIST		OrganAMNIST		OrganCMNIST		OrganSMNIST	
	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC
ResNet-18 (28)	0.901	0.863	0.998	0.958	0.930	0.676	0.997	0.935	0.992	0.900	0.972	0.782
ResNet-18 (224)	0.891	0.833	0.998	0.963	0.933	0.681	0.998	0.951	0.994	0.920	0.974	0.778
ResNet-50 (28)	0.857	0.812	0.997	0.956	0.931	0.680	0.997	0.935	0.992	0.905	0.972	0.770
ResNet-50 (224)	0.866	0.842	0.997	0.950	0.932	0.680	0.998	0.947	0.993	0.911	0.975	0.785
auto-sklearn	0.836	0.803	0.984	0.878	0.828	0.532	0.963	0.762	0.976	0.829	0.945	0.672
auto-keras	0.871	0.831	0.998	0.961	0.941	0.703	0.994	0.905	0.990	0.879	0.974	0.813
auto-ml	0.919	0.861	0.998	0.966	0.924	0.673	0.990	0.886	0.988	0.877	0.964	0.749
VisionTransformer	0.824	0.833	0.985	0.888	0.880	0.596	0.968	0.770	0.970	0.787	0.934	0.620
OrthoFNN	0.815	0.821	0.972	0.820	0.819	0.513	0.916	0.636	0.923	0.672	0.875	0.481
OrthoPatchWise	0.830	0.827	0.984	0.866	0.845	0.549	0.973	0.786	0.976	0.805	0.941	0.640
OrthoTransformer	0.770	0.744	0.982	0.860	0.856	0.557	0.968	0.763	0.973	0.785	0.946	0.635
CompoundTransformer	0.859	0.846	0.985	0.870	0.841	0.544	0.975	0.789	0.978	0.819	0.943	0.647

Table 6: Extended Performance Analysis in metrics of AUC and ACC on each test dataset of MedMNIST where we included the results reported in Yang et al. (2021).