axioms

Article

Hybrid Quantum Vision Transformers for Event Classification in High Energy Physics

Eyup B. Unlu [1,*], Marçal Comajoan Cara [2], Gopal Ramesh Dahale [3], Zhongtian Dong [4], Roy T. Forestano [1], Sergei Gleyzer [5], Daniel Justice [6], Kyoungchul Kong [4], Tom Magorsch [7], Konstantin T. Matchev [1] and Katia Matcheva [1]

1 Institute for Fundamental Theory, Physics Department, University of Florida, Gainesville, FL 32611, USA; roy.forestano@ufl.edu (R.T.F.); matchev@ufl.edu (K.T.M.); matcheva@ufl.edu (K.M.)
2 Department of Signal Theory and Communications, Polytechnic University of Catalonia, 08034 Barcelona, Spain; marcal.comajoan@estudiantat.upc.edu
3 Indian Institute of Technology Bhilai, Bhilai 491001, Chhattisgarh, India; gopald@iitbhilai.ac.in
4 Department of Physics & Astronomy, University of Kansas, Lawrence, KS 66045, USA; cdong@ku.edu (Z.D.); kckong@ku.edu (K.K.)
5 Department of Physics & Astronomy, University of Alabama, Tuscaloosa, AL 35487, USA; sgleyzer@ua.edu
6 Software Engineering Institute, Carnegie Mellon University, 4500 Fifth Avenue, Pittsburgh, PA 15213, USA; dljustice@sei.cmu.edu
7 Physik-Department, Technische University of München, James-Franck-Str. 1, 85748 Garching, Germany; tom.magorsch@tum.de
* Correspondence: eyup.unlu@ufl.edu

Abstract: Models based on vision transformer architectures are considered state-of-the-art when it comes to image classification tasks. However, they require extensive computational resources both for training and deployment. The problem is exacerbated as the amount and complexity of the data increases. Quantum-based vision transformer models could potentially alleviate this issue by reducing the training and operating time while maintaining the same predictive power. Although current quantum computers are not yet able to perform high-dimensional tasks, they do offer one of the most efficient solutions for the future. In this work, we construct several variations of a quantum hybrid vision transformer for a classification problem in high-energy physics (distinguishing photons and electrons in the electromagnetic calorimeter). We test them against classical vision transformer architectures. Our findings indicate that the hybrid models can achieve comparable performance to their classical analogs with a similar number of parameters.

Keywords: quantum computing; deep learning; quantum machine learning; vision transformers; supervised learning; classification; large hadron collider

MSC: 68Q12; 81P68

check for updates

Citation: Unlu, E.B.; Comajoan Cara, M.; Dahale, G.R.; Dong, Z.; Forestano, R.T.; Gleyzer, S.; Justice, D.; Kong, K.; Magorsch, T.; Matchev, K.T.; et al. Hybrid Quantum Vision Transformers for Event Classification in High Energy Physics. *Axioms* **2024**, *13*, 187. https://doi.org/10.3390/axioms13030187

Academic Editor: Mariam Zomorodi

Received: 27 January 2024
Revised: 7 March 2024
Accepted: 8 March 2024
Published: 13 March 2024

1. Introduction

The first transformer architecture was introduced in 2017 by Vaswani et al. in a famous paper "Attention Is All You Need" [1]. The new model was shown to outperform the existing state-of-the-art models by a significant margin for the English-to-German and English-to-French `newstest2014` tests. Since then, the transformer architecture has been implemented in numerous fields and has become the go-to model for many different applications such as sentiment analysis [2] and question answering [3].

The vision transformer architecture can be considered as the implementation of transformer architecture for image classification. It utilizes the encoder part of the transformer architecture and attaches a multi-layer perceptron (MLP) layer to classify images. This architecture was first introduced by Dosovitskiy et al. in the paper "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" [4]. It was shown that in a multitude of datasets, a vision transformer model is capable of outperforming the state-of-the-art model ResNet152x4 while using less computation time to pre-train. Similar to their language counterparts, vision transformers became the state-of-the-art models for a multitude of computer vision problems such as image classification [5] and semantic segmentation [6].

However, these advantages come at a cost. Transformer architectures are known to be computationally expensive to train and operate [7]. Specifically, their demands on computation power and memory increase quadratically with the input length. A number of studies have attempted to approximate self-attention in order to decrease the associated quadratic complexity in memory and computation power [8–11]. There are also proposed modifications to the architecture which aim to alleviate the quadratic complexity [12–14]. A recent review of the different methods for reducing the complexity of transformers can be found in [15]. As the amount of data grows, these problems are exacerbated. In the future, it will be necessary to find a substitute architecture that has similar performance but demands fewer resources.

A quantum machine learning model might be one of those substitutes. Although the hardware for quantum computation is still in its infancy, there is a high volume of research that is focused on the algorithms that can be used on this hardware. The main appeal of quantum algorithms is that they are already known to have computational advantages over classical algorithms for a variety of problems. For instance, Shor's algorithm can factorize numbers significantly faster than the best classical methods [16]. Furthermore, there are studies suggesting that quantum machine learning can lead to computational speedups [17,18].

In this work, we develop a quantum-classical hybrid vision transformer architecture. We demonstrate our architecture on a problem from experimental high energy physics, which is an ideal testing ground because experimental collider physics data are known to have a significant amount of complexity and computational resources represent a major bottleneck [19–21]. Specifically, we use our model to classify the parent particle in an electromagnetic shower event inside the CMS detector. In addition, we will test the performance of our hybrid architecture by benchmarking it against a classical vision transformer of equivalent architecture.

This paper is structured as follows. In Section 2, we introduce and describe the dataset. The model architectures for both the classical and hybrid models are discussed in Section 3. The model parameters and the training are specified in Sections 4 and 5, respectively. Finally, in Section 6 we present our results and discuss their implications in Section 7. We consider the future directions for study in Section 8.

2. Dataset and Preprocessing Description

The Compact Muon Solenoid (CMS) is one of the four main experiments at the Large Hadron Collider (LHC), which has been in operation since 2009 at CERN. The CMS detector [22] has been recording the products from collisions between beams consisting of protons or Pb ions, at several different center-of-mass energy milestones, up to the current 13.6 TeV [23]. Among the various available CMS datasets [24], we have chosen to study data from proton–proton collisions at 13.6 TeV. Among the basic types of objects reconstructed from those collisions are photons and electrons, which leave rather similar signatures in the CMS electromagnetic calorimeter (ECAL) (see, e.g., Ref. [25] and references therein). A common task in high-energy physics is to classify the resulting electromagnetic shower in the ECAL as a photon (γ) or electron (e^{-}). In practice, one also uses information from the CMS tracking system [26] and leverages the fact that an electron leaves a track, while a photon does not. However, for the purposes of our study, we shall limit ourselves to the ECAL only.

The dataset used in our study contains the reconstructed hits of 498,000 simulated electromagnetic shower events in the ECAL sub-detector of the CMS experiment (photon conversions were not simulated) [27]. Half of the events originate from photons, while the remaining half are initiated by electrons. In each case, an event is generated with exactly one particle (γ or e^-) which is fired from the interaction point with fixed transverse momentum magnitude $|\vec{p}_T| = 50$ GeV, see Figure 1. The direction of the momentum $\vec{p}$ is sampled uniformly in azimuthal angle $-\pi \leq \varphi \leq \pi$ and pseudorapidity $-1.4 \leq \eta \leq 1.4$, where the latter is defined in terms of the polar angle θ as $\eta = -\ln\tan(\theta/2)$.

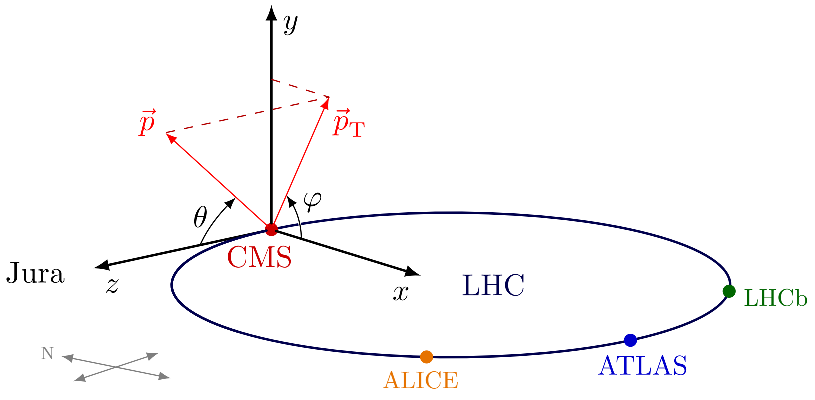


Figure 1. The CMS coordinate system against the backdrop of the LHC, with the location of the four main experiments (CMS, ALICE, ATLAS and LHCb). The z axis points to the Jura mountains, while the y-axis points toward the sky. In spherical coordinates, the components of a particle momentum $\vec{p}$ are its magnitude $|\vec{p}|$, the polar angle θ (measured from the z-axis), and the azimuthal angle φ (measured from the x-axis). The transverse momentum $\vec{p}_T$ is the projection of $\vec{p}$ on the transverse (xy) plane. Figure generated with TikZ code adapted from Ref. [28].

For each event, the dataset includes two image grids, representing energy and timing information, respectively. The first grid gives the peak energy detected by the crystals of the detector in a 32×32 grid centered around the crystal with the maximum energy deposit. The second image grid gives the arrival time when the peak energy was measured in the associated crystal (in our work, we shall only use the first image grid with the energy information.) Each pixel in an image grid corresponds to exactly one ECAL crystal, though not necessarily the same crystal from one event to another. The images were then scaled so that the maximum entry for each event was set to 1.

Several representative examples of our image data are shown in Figure 2. The first row shows the image grids for the energy (normalized and displayed in $\log_{10}$ scale), while the second row displays the timing information (not used in our study). In each case, the top row in the title lists the label predicted by one of the benchmark classical models, while the bottom row shows the correct label for that instance—whether the image was generated by an actual electron or photon.

As can be gleaned from Figure 2 with the naked eye, electron–photon discrimination is a challenging task—for example, the first and third images in Figure 2 are wrongly classified. To first approximation, the e^- and γ shower profiles are identical, and mostly concentrated in a 3×3 grid of crystals around the main deposit. However, interactions with the magnetic field of the CMS solenoid ($B = 3.8$ T) cause electrons to emit bremsstrahlung radiation, preferentially in φ. This introduces a higher-order perturbation on the shower shape, causing the electromagnetic shower profiles [29] to be more spread out and slightly asymmetric in φ.

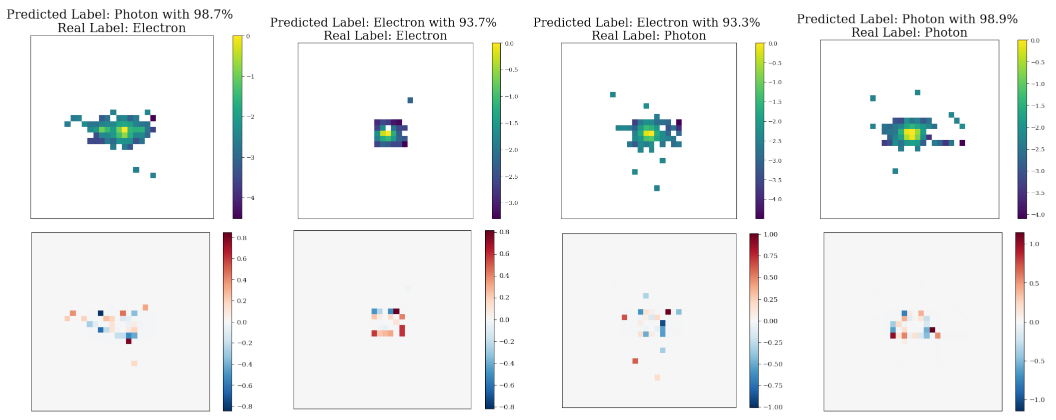


Figure 2. Four representative image grid examples from the dataset, in the (φ, η) plane. The first row shows the image grids for the energy (normalized and displayed in $\log_{10}$ scale), while the second row displays the timing information. The titles list the correct labels (real electron or real photon), as well as the corresponding labels predicted by one of the benchmark classical models (see text for more details).

3. Model Architectures

The following definitions will be used for the rest of the paper and are listed here for convenience.

- n_t: Number of tokens/patches
- d_i: Flattened patch length
- d_t: Token length
- n_h: Number of heads
- $d_h \equiv \frac{d_t}{n_h}$: Data length per head
- d_{ff}: The dimension of the feed-forward network

3.1. General Model Structure

Both the benchmark and hybrid models utilize the same architectures except for the type of encoder layers. These architectures are shown in Figure 3. As can be seen in the figure, there will be two main variants of the architecture: (a) column-pooling variant and (b) class token variant.

As the encoder layer is the main component of both the classical and the hybrid models, they will be discussed in more detail in Sections 3.2 and 3.3, respectively. The rest of the architecture is discussed here.

First, we start by dividing our input picture into n_t patches of equal area, which are then flattened to obtain n_t vectors with length d_i. The resulting vectors are afterward concatenated to obtain a $n_t \times d_i$ matrix for each image sample. This matrix is passed through a linear layer with a bias (called "Linear Embedding" in the figure) to change the number of columns from d_i to a desirable number (token dimension, referred to as d_t).

If the model is a class token variant, a trainable vector of length d_t is concatenated as the first row of the matrix at hand (module “Concat” in Figure 3b). After that, a non-trainable vector is added to each row (called the positional embedding vector). Then the result is fed to a series of encoder layers where each subsequent encoder layer uses its predecessor’s output as its input.

If the model is a class token variant, the first row of the output matrix of the final encoder layer is fed into the classifying layer to obtain the classification probabilities (“Extract Class Token” layer in Figure 3b). Otherwise, a column-pooling method (take the mean of all the rows or take the maximum value for each column) is used to reduce the output matrix into a vector, then this vector is fed into the classifying layer to obtain the classification probabilities (“Column-wise Pooling” layer in Figure 3a).

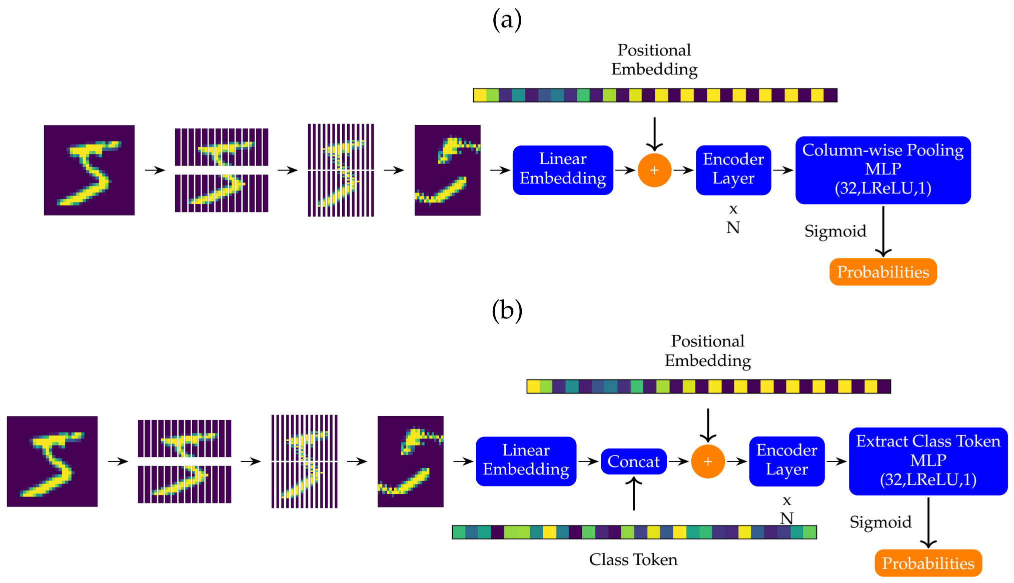


Figure 3. The architecture for the (**a**) column-wise pooling and (**b**) the class-token models. For clarity, we use an MNIST image [30] to demonstrate the process. The hybrid and the classical model differ by the architecture of their encoder layers (see Figures 4 and 5).

3.2. The Classical Encoder Layer

The structure of the classical encoder layer can be seen in Figure 4a. First, we start by standardizing the input data to have zero mean and a standard deviation of one. Afterward, the normalized data are fed to the multi-head attention (discussed in the next paragraph) and the output is summed with the unnormalized data. Then, the modified output is again normalized to have zero mean and a standard deviation of one. These normalized modified data are then fed into a multilayer perceptron of two layers with hidden layer size d_{ff} and the result is summed up with the modified data to obtain the final result.

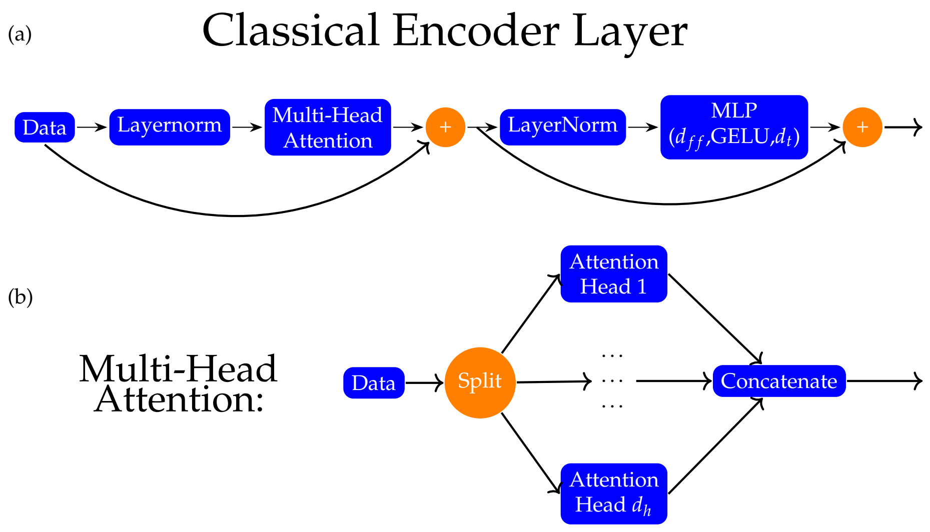


Figure 4. The classical encoder layer (**a**) and multi-head attention (**b**) architecture for the benchmark models.

The multi-head attention works by separating our input matrix into n_h many $n_t \times d_h$ matrices by splitting them through their columns. Afterward, the split matrices are fed to the attention heads described in Equations (1) and (2). Finally, the outputs of the attention heads are concatenated to obtain an $n_t \times d_t$ matrix, which has the same size as our input matrix. Each attention head is defined as

$$\text{Attention Head } (x_i; W_K^{(i)}, W_Q^{(i)}, W_V^{(i)}) = \text{SoftMax}\left(\frac{(x_i W_K^{(i)})(x_i W_Q^{(i)})^T}{\sqrt{d_h}}\right)(x_i W_V^{(i)})$$

$$W_K^{(i)} \in R^{(d_h \times d_h)}, W_Q^{(i)} \in \mathbb{R}^{(d_h \times d_h)}, W_V^{(i)} \in \mathbb{R}^{(d_h \times d_h)}\ d_h \equiv d_t / n_h; \tag{1}$$

where

$$X = \begin{bmatrix} x_1 & x_2 & \dots & x_{n_h} \end{bmatrix} \in \mathbb{R}^{(n_t \times d_t)}, \quad x_i \in \mathbb{R}^{(n_t \times d_h)} \tag{2}$$

is the input matrix.

3.3. Hybrid Encoder Layer

The structure of the hybrid encoder layer can be seen in Figure 5a. Firstly, we start by standardizing the input data to have zero mean and standard deviation of one. Afterward, the normalized data are fed to the hybrid multi-head attention layer (discussed in the next paragraph). Then, the output is fed into a multilayer perceptron of two layers with hidden layer size d_{ff}, and the result is summed up with the unnormalized data to obtain the final result.

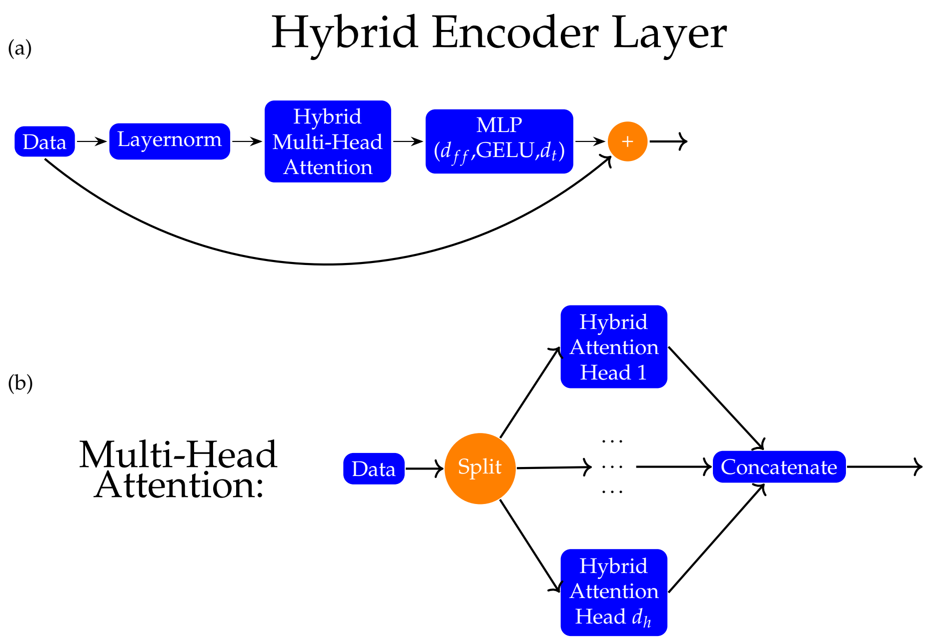


Figure 5. The hybrid encoder layer architecture (**a**) and multi-head attention (**b**) architecture for the hybrid models.

The hybrid multi-head attention works by separating our input matrix into n_h many $n_t \times d_h$ matrices by splitting them through their columns. Afterward, the split matrices are fed to the hybrid attention heads (which are described in the bulleted procedure below). Finally, the outputs of the attention heads are concatenated to obtain an $n_t \times d_t$ matrix, which has the same size as our input matrix.

The hybrid attention heads we used are almost identical to the architecture implemented in [31], "Quantum Self-Attention Neural Networks for Text Classification" by Li et al. In order to replace the self-attention mechanism of a classical vision transformer in Equation (1), we use the following procedure:

- Define x_i as the ith row of the input matrix X.

- Define the data loader operator $\hat{U}(x_i)$ as

$$|x_i\rangle \equiv \hat{U}(x_i)|0>^{(d_h)} = \bigotimes_{j=1}^{d_h} \hat{R}_x(x_{ij})\hat{H}\,|0\rangle\,, \tag{3}$$

where $\hat{H}$ is the Hadamard gate and $\hat{R}_x$ is the parameterized rotation around the x-axis.

- Apply the key circuit (data loader + key operator $\hat{K}(\theta_K)$) for each x_i and obtain the column vector K (see Figure 6).

$$K_i = \langle x_i|\,\hat{K}^\dagger(\theta_K)\hat{Z}_0\hat{K}(\theta_K)\,|x_i\rangle\,, \quad 1 \le i \le d_t, \tag{4}$$

where $\hat{Z}_i$ is a spin measurement of the ith qubit on the z direction.

- Apply the query circuit (data loader $\hat{U}(x_i)$ + query operator $\hat{Q}(\theta_Q)$) for each x_i and obtain the column vector Q (see Figure 6).

$$Q_i = \langle x_i|\,\hat{Q}^\dagger(\theta_Q)\hat{Z}_0\hat{Q}(\theta_Q)\,|x_i\rangle\,, \quad 1 \le i \le d_t. \tag{5}$$

- Obtain the so-called attention matrix using the key and the query vectors using the following expression

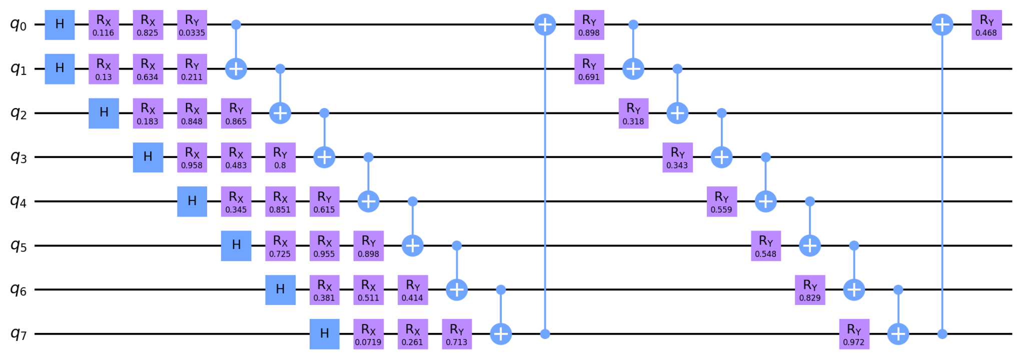
$$A_{ij} = -(Q_i - K_j)^2; \quad 1 \le i \le d_t, 1 \le j \le d_t. \tag{6}$$

- Apply the value circuit (data loader + value operator $\hat{V}(\theta_V)$) to each row of the image and measure each qubit separately to obtain the value matrix. (See Figure 7)

$$V_{ij} = \langle x_i|\,\hat{V}^\dagger(\theta_V)\hat{Z}_j\hat{V}(\theta_V)\,|x_i\rangle\,, |x_i\rangle = \hat{U}(x_i)|0_n>; \quad 1 \le i \le d_t, 1 \le j \le d_h. \tag{7}$$

- Define the self-attention operation as,

$$\text{Hybrid Attention Head: SoftMax}\left(\frac{A}{\sqrt{d_h}}\right)V. \tag{8}$$



q_0 H Rx 0.116 Rx 0.825 Ry 0.0335 Ry 0.898 Ry 0.468
q_1 H Rx 0.13 Rx 0.634 Ry 0.211 Ry 0.691
q_2 H Rx 0.183 Rx 0.848 Ry 0.865 Ry 0.318
q_3 H Rx 0.958 Rx 0.483 Ry 0.8 Ry 0.343
q_4 H Rx 0.345 Rx 0.851 Ry 0.615 Ry 0.559
q_5 H Rx 0.725 Rx 0.955 Ry 0.898 Ry 0.548
q_6 H Rx 0.381 Rx 0.511 Ry 0.414 Ry 0.829
q_7 H Rx 0.0719 Rx 0.261 Ry 0.713 Ry 0.972

Figure 6. Key and Query circuit for the $d_h = 8$ case. The first two rows of circuits load the data to the circuit ($\hat{U}(x_i)$ operator), while the rest are the parts of the trainable ansatz. Therefore, the total number of parameters for each circuit is equal to $3d_h + 1$.

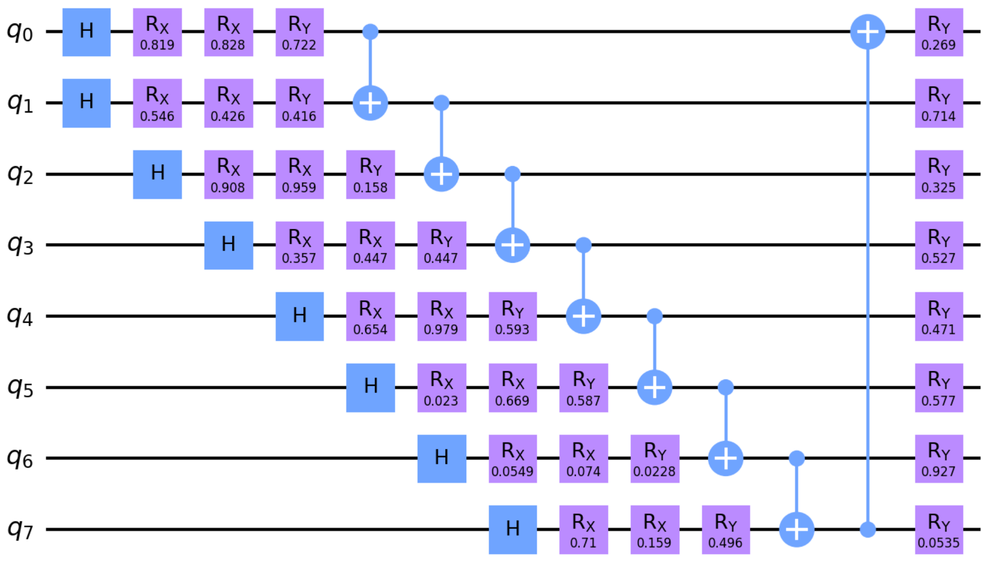


Figure 7. The value circuit used for the $d_h = 8$ case. The first two rows of circuits load the data to the circuit ($\hat{U}(x_i)$ operator), while the rest are the parts of the trainable ansatz. Therefore, the total number of trainable parameters for each circuit is equal to $3d_h$.

4. Hyper-Parameters

The number of parameters is a function of the hyper-parameters for both the classical and the hybrid models. However, these functions are different. Both models share the same linear embedding and classifying layer. The linear embedding layer contains $(d_i + 1)d_t$ many parameters and the classifying layer contains $32d_t + 65$ parameters.

For each classical encoder layer, we have n_h attention heads which all contain $3d_h^2$ parameters from the Q, K, and V layers, respectively. In addition, the MLP layer inside each encoder layer contains $2d_{ff}d_t + d_{ff} + d_t$ parameters. Overall, each classical vision transformer has $d_t(33 + d_i) + n_l(2d_{ff}d_t + d_{ff} + d_t + 3n_h d_h^2)$ parameters except for the class token variation which has extra d_t parameters.

For each hybrid encoder layer, we have n_h attention heads which all contain $9d_h + 2$ parameters from the Q, K, and V layers, respectively. Similar to the classical model, each encoder layer MLP contains $2d_{ff}d_t + d_{ff} + d_t$ parameters. Overall, each hybrid vision transformer has $d_t(33 + d_i) + n_l(2d_{ff}d_t + d_{ff} + d_t + n_h(9d_h + 2))$ parameters except for the class token variation which has extra d_t parameters.

Therefore, assuming they have the same hyper-parameters, the difference between the number of parameters for the classical and hybrid models is $n_l(d_t(3d_h - 9) - 2n_h)$.

Our purpose was to investigate whether our architecture might perform similarly to a classical vision transformer where the number of parameters are close to each other. In order to use a similar number of parameters, we picked a region of hyperparameters such that this difference is rather minimal. For all models, the following parameters were used:

- $n_l = 5$
- $d_t = 16$
- $n_t = 16$
- $n_h = 4$
- $d_h = \frac{d_i}{d_h} = 4$
- $d_{ff} = 16$.

Therefore, for our experiment the number of parameters for the classical models (4785 to 4801) is slightly more than the quantum models (4585 to 4601).

5. Training Process

All the classical parts of the models were implemented in PyTorch [32]. The quantum circuit simulations were conducted using TensorCircuit with the JAX backend [33,34]. We explored a few different hyperparameter settings before settling on the following. Each model was trained for 40 epochs, which was typically sufficient to ensure convergence, see Figures 8 and 9. The criteria for the selection of the best model iteration was the accuracy of the validation data. The optimizer used was the ADAM optimizer with learning rate $\lambda = 5 \times 10^{-3}$ [35]. All models were trained on GPUs and the typical training times were on the order of 10 min (5 h) for the classical (quantum) models. The batch size was 512 for all models as well. The loss function utilized was the binary cross entropy. The code used to create and train the models can be found at the following GitHub repository: https://github.com/EyupBunlu/QViT_HEP_ML4Sci (accessed on 7 March 2024).

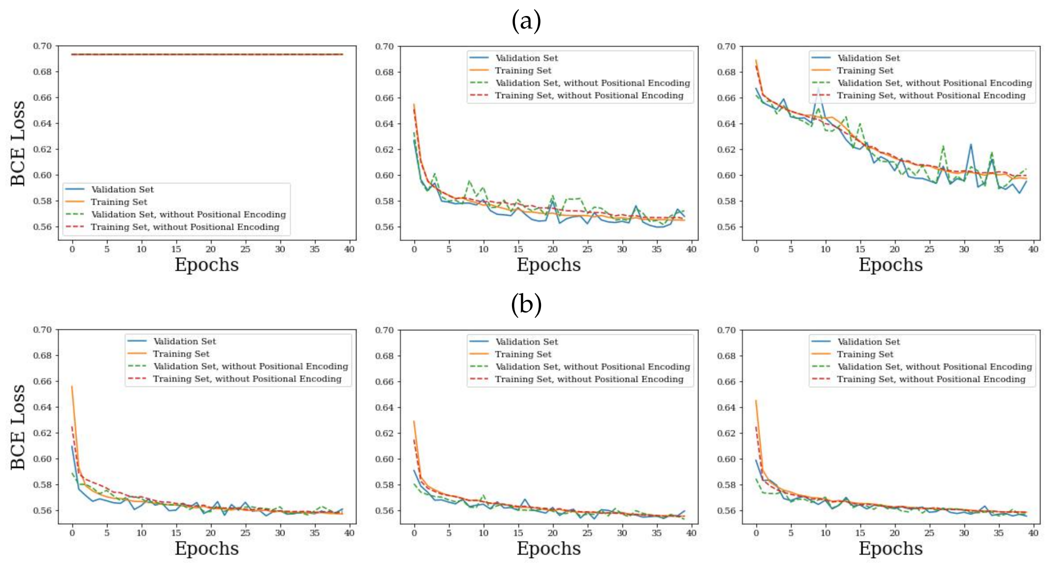


Figure 8. BCE loss on the validation and training set during training for the (**a**) quantum and (**b**) classical models. From left to right, each column corresponds to a different model variant: class token (left column), column max (middle column) and column mean variant (right column). For each plot, the blue (orange) line corresponds to the validation (training) set loss for the model with positional encoding, whereas the dashed green (red) line corresponds to the validation (training) set loss for the model without positional encoding layer.

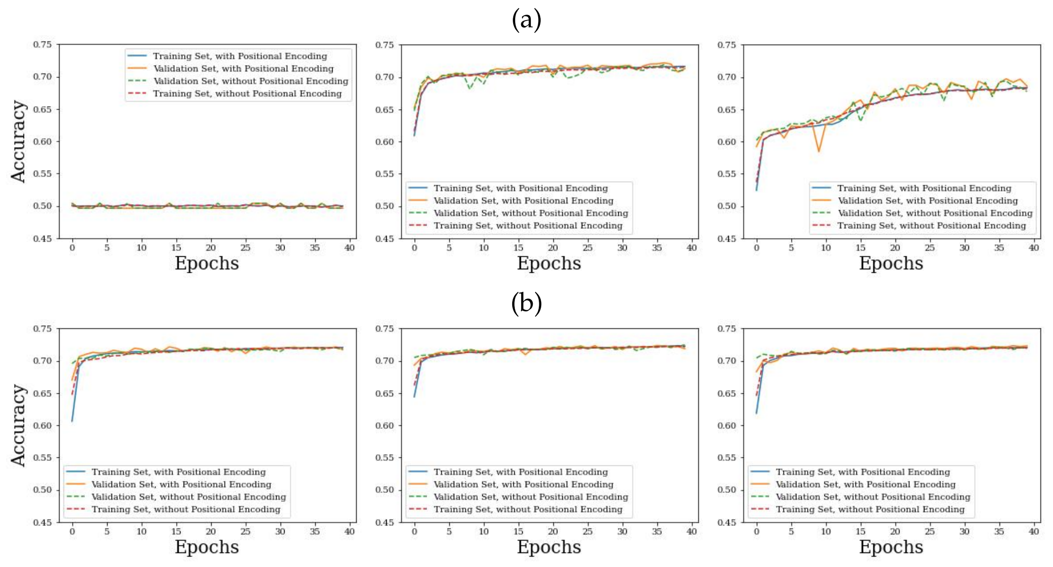


Figure 9. The same as Figure 8, but for the accuracy on the validation and training set during training for the (**a**) quantum and (**b**) classical models.

6. Results

The training loss and the accuracy of the validation and training data are plotted in Figures 8 and 9, respectively. In addition, the models were compared on several metrics such as the accuracy, binary cross-entropy loss, and AUC (area under the ROC curve) on the test data. This comparison is shown in Table 1.

Table 1. Comparison table for the models. The accuracy, the BCE loss and the AUC score were calculated on the test data. For each entry, the first number corresponds to the classical model, whereas the second one corresponds to the hybrid model. For each variant and metric, the best value is shown in bold.

Model	**Positional Embedding**	**Accuracy (Cls/Hybrid)**	**BCE Loss (Cls/Hybrid)**	**AUC Score (Cls/Hybrid)**	**Trainable Parameters (Cls/Hybrid)**
With Class Token	Yes	**0.717**/0.502	**0.564**/0.6931	**0.780**/0.501	4801/4601
With Class Token	No	**0.720**/0.502	**0.561**/0.6931	**0.783**/0.500	4801/4601
Column Max (CMX)	Yes	**0.718**/**0.718**	**0.562**/0.565	**0.783**/0.779	4785/4585
Column Max (CMX)	No	**0.722**/0.718	**0.557**/0.565	**0.786**/0.779	4785/4585
Column Mean (CMN)	Yes	**0.720**/0.696	**0.559**/0.592	**0.784**/0.751	4785/4585
Column Mean (CMN)	No	**0.720**/0.692	**0.560**/0.595	**0.783**/0.748	4785/4585

7. Discussion

As seen in Table 1, the positional encoding has no significant effect on the performance metrics. In retrospect, this is not that surprising, since the position information is already used in the linear embedding in Figure 3 (we thank the anonymous referee for this clarification.). We note that the CMX variant (either with or without positional encoding) performs similarly to the corresponding classical model. This suggests that a quantum advantage could be achieved when extrapolating to higher-dimensional problems and datasets since the quantum models scale better with dimensionality.

On the other hand, Table 1 shows that hybrid CMN variants are inferior to their hybrid CMX counterparts for all metrics. This might be due to the fact that taking the mean forces each element of the output matrix of the final encoder layer to be relevant, unlike the CMX variant, where the maximum values are chosen. This could explain the larger number of epochs required to converge in the case of the hybrid CMN (see Figures 8 and 9). It is also possible that the hybrid model lacks the expressiveness required to encode enough meaningful information to the column means.

Somewhat surprisingly, the training plots of the hybrid class token variants (upper left panels in Figures 8 and 9) show that the hybrid class token variants did not converge during our numerical experiments. The reason behind this behavior is currently unknown and is being investigated.

8. Outlook

Quantum machine learning is a relatively new field. In this work, we explored a few of the many possible ways that it could be used to perform different computational tasks as an alternative to classical machine learning techniques. As the current hardware for quantum computers improves further, it is important to explore more ways in which this hardware could be utilized.

Our study raises several questions that warrant future investigations. First, we observe that the hybrid CMX models perform similarly to the classical vision transformer models that we used for benchmarking. It is fair to ask if this similarity is due to the comparable number of trainable parameters or the result of an identical choice of hyper-parameter values. If it is the latter, we can extrapolate and conclude that as the size of the data grows, hybrid models will still perform as well as the classical models while having a significantly fewer number of parameters.

It is fair to say that both the classical and hybrid models perform similarly at this scale. However, the hybrid model discussed in this work is mostly classical, except for the attention heads. The next step in our research is to investigate the effect of increasing the fraction of quantum elements of the model. For instance, the conversion of feed-forward layers into quantum circuits such as the value circuit might lead to an even bigger advantage in the number of trainable parameters between the classical and hybrid models.

Although the observed limitations in the class token and column mean variants might appear disappointing at first glance, they are also important findings of this work. It is worth investigating whether this is due to the nature of the dataset or a sign of a fundamental limitation in the method.

Author Contributions: Conceptualization, E.B.U.; methodology, M.C.C., G.R.D., Z.D., R.T.F., S.G., D.J., K.K., T.M., K.T.M., K.M. and E.B.U.; software, E.B.U.; validation, M.C.C., G.R.D., Z.D., R.T.F., T.M. and E.B.U.; formal analysis, E.B.U.; investigation, M.C.C., G.R.D., Z.D., R.T.F., T.M. and E.B.U.; resources, E.B.U., K.T.M. and K.M.; data curation, G.R.D., S.G. and T.M.; writing—original draft preparation, E.B.U.; writing—review and editing, S.G., D.J., K.K., K.T.M. and K.M.; visualization, E.B.U.; supervision, S.G., D.J., K.K., K.T.M. and K.M.; project administration, S.G., D.J., K.K., K.T.M. and K.M.; funding acquisition, S.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 using NERSC award NERSC DDR-ERCAP0025759. S.G. is supported in part by the U.S. Department of Energy (DOE) under Award No. DE-SC0012447. K.M. is supported in part by the U.S. DOE award number DE-SC0022148. K.K. is supported in part by US DOE DE-SC0024407. Z.D. is supported in part by College of Liberal Arts and Sciences Research Fund at the University of Kansas. Z.D., R.T.F., E.B.U., M.C.C., G.R.D. and T.M. were participants in the 2023 Google Summer of Code.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The dataset used in this analysis is described in [27] and is available at https://cernbox.cern.ch/index.php/s/FbXw3V4XNyYB3oA (accessed on 12 March 2024) and https://cernbox.cern.ch/index.php/s/AtBT8y4MiQYFcgc (accessed on 12 March 2024). The code used to create and train the models can be found at https://github.com/EyupBunlu/QViT_HEP_ML4Sci (accessed on 12 March 2024).

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

AUC	Area Under the Curve
BCE	Binary Cross Entropy
CMN	Column Mean
CMS	Compact Muon Solenoid (experiment)
CMX	Column Max
ECAL	Electromagnetic Calorimeter
GPU	Graphics processing unit
LHC	Large Hadron Collider
MHA	Multi-Head Attention
MLP	Multi-Layer Perceptron
MNIST	Modified National Institute of Standards and Technology database
ROC	Receiver Operating Characteristic

References

1. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In Proceedings of the NIPS'17: 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 6000–6010.
2. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* **2020**, *21*, 1–67.
3. Jun, C.; Jang, H.; Sim, M.; Kim, H.; Choi, J.; Min, K.; Bae, K. ANNA: Enhanced Language Representation for Question Answering. In Proceedings of the 7th Workshop on Representation Learning for NLP, Dublin, Ireland, 26 May 2022; pp. 121–132. [CrossRef]
4. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In Proceedings of the 2021 International Conference on Learning Representations, Virtual, 3–7 May 2021.
5. Yu, X.; Xue, Y.; Zhang, L.; Wang, L.; Liu, T.; Zhu, D. NoisyNN: Exploring the Influence of Information Entropy Change in Learning Systems. *arXiv* **2023**, arXiv:2309.10625.
6. Fang, Y.; Wang, W.; Xie, B.; Sun, Q.; Wu, L.; Wang, X.; Huang, T.; Wang, X.; Cao, Y. EVA: Exploring the Limits of Masked Visual Representation Learning at Scale. In Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 18–22 June 2023; IEEE: Piscataway, NJ, USA, 2023. [CrossRef]
7. Tuli, S.; Dedhia, B.; Tuli, S.; Jha, N.K. FlexiBERT: Are current transformer architectures too homogeneous and rigid? *J. Artif. Intell. D* **2023**, *77*, 39–70. [CrossRef]
8. Gupta, A.; Berant, J. Value-aware Approximate Attention. *arXiv* **2021**, arXiv:2103.09857.
9. Xiong, Y.; Zeng, Z.; Chakraborty, R.; Tan, M.; Fung, G.; Li, Y.; Singh, V. Nyströmformer: A Nyström-Based Algorithm for Approximating Self-Attention. *arXiv* **2021**, arXiv:2102.03902.
10. Dao, T.; Fu, D.Y.; Ermon, S.; Rudra, A.; Ré, C. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. *arXiv* **2022**, arXiv:2205.14135.
11. Peng, H.; Pappas, N.; Yogatama, D.; Schwartz, R.; Smith, N.; Kong, L. Random Feature Attention. In Proceedings of the 2021 International Conference on Learning Representations, Virtual, 3–7 May 2021.
12. Kitaev, N.; Kaiser, Ł.; Levskaya, A. Reformer: The Efficient Transformer. *arXiv* **2020**, arXiv:2001.04451.
13. Zaheer, M.; Guruganesh, G.; Dubey, A.; Ainslie, J.; Alberti, C.; Ontanon, S.; Pham, P.; Ravula, A.; Wang, Q.; Yang, L.; et al. Big bird: Transformers for longer sequences. In Proceedings of the NIPS'20: 34th International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 6–12 December 2020; Curran Associates Inc.: Red Hook, NY, USA, 2020.
14. Choromanski, K.; Likhosherstov, V.; Dohan, D.; Song, X.; Gane, A.; Sarlos, T.; Hawkins, P.; Davis, J.; Mohiuddin, A.; Kaiser, L.; et al. Rethinking Attention with Performers. *arXiv* **2022**, arXiv:2009.14794.
15. Fournier, Q.; Caron, G.M.; Aloise, D. A Practical Survey on Faster and Lighter Transformers. *ACM Comput. Surv.* **2023**, *55*, 1–40. [CrossRef]
16. Shor, P.W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* **1997**, *26*, 1484–1509. [CrossRef]
17. Servedio, R.A.; Gortler, S.J. Equivalences and Separations Between Quantum and Classical Learnability. *SIAM J. Comput.* **2004**, *33*, 1067–1092. [CrossRef]

18. Dunjko, V.; Briegel, H.J. Machine learning & artificial intelligence in the quantum domain: A review of recent progress. *Rep. Prog. Phys.* **2018**, *81*, 074001. [CrossRef]
19. The HEP Software Foundation; Albrecht, J.; Alves, A.A.; Amadio, G.; Andronico, G.; Anh-Ky, N.; Aphecetche, L.; Apostolakis, J.; Asai, M.; Atzori, L.; et al. A Roadmap for HEP Software and Computing R&D for the 2020s. *Comput. Softw. Big Sci.* **2019**, *3*, 7.
20. HSF Physics Event Generator WG; Valassi, A.; Yazgan, E.; McFayden, J.; Amoroso, S.; Bendavid, J.; Buckley, A.; Cacciari, M.; Childers, T.; Ciulli, V.; et al. Challenges in Monte Carlo Event Generator Software for High-Luminosity LHC. *Comput. Softw. Big Sci.* **2021**, *5*, 12.
21. Humble, T.S.; Perdue, G.N.; Savage, M.J. Snowmass Computational Frontier: Topical Group Report on Quantum Computing. *arXiv* **2022**, arXiv:2209.06786.
22. CMS Collaboration; Chatrchyan, S.; Hmayakyan, G.; Khachatryan, V.; Sirunyan, A.M.; Adam, W.; Bauer, T.; Bergauer, T.; Bergauer, H.; Dragicevic, M.; et al. The CMS Experiment at the CERN LHC. *JINST* **2008**, *3*, S08004. [CrossRef]
23. CMS Heavy-Ion Public Physics Results. Available online: https://twiki.cern.ch/twiki/bin/view/CMSPublic/PhysicsResultsHIN (accessed on 6 March 2024).
24. Public CMS Data Quality Information. Available online: https://twiki.cern.ch/twiki/bin/view/CMSPublic/DataQuality (accessed on 6 March 2024).
25. Benaglia, A. The CMS ECAL performance with examples. *JINST* **2014**, *9*, C02008. [CrossRef]
26. CMS Collaboration; Chatrchyan, S.; Khachatryan, V.; Sirunyan, A.M.; Tumasyan, A.; Adam, W.; Bergauer, T.; Dragicevic, M.; Erö, J.; Fabjan, C.; et al. Description and performance of track and primary-vertex reconstruction with the CMS tracker. *JINST* **2014**, *9*, P10009.
27. Andrews, M.; Paulini, M.; Gleyzer, S.; Poczos, B. End-to-End Event Classification of High-Energy Physics Data. *J. Phys. Conf. Ser.* **2018**, *1085*, 042022. [CrossRef]
28. CMS Coordinate System. Available online: https://tikz.net/axis3d_cms/ (accessed on 6 March 2024).
29. Sempere Roldan, P. Quality Control and Preparation of the PWO Crystals for the Electromagnetic Calorimeter of CMS. Ph.D. Thesis, University of Santiago de Compostela, Santiago, Spain, 2011.
30. LeCun, Y.; Cortes, C. MNIST Handwritten Digit Database. 2010. Available online: http://yann.lecun.com/exdb/mnist/ (accessed on 10 January 2024).
31. Li, G.; Zhao, X.; Wang, X. Quantum Self-Attention Neural Networks for Text Classification. *arXiv* **2022**, arXiv:2205.05625.
32. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*; Curran Associates, Inc.: Red Hook, NY, USA, 2019; pp. 8024–8035.
33. Bradbury, J.; Frostig, R.; Hawkins, P.; Johnson, M.J.; Leary, C.; Maclaurin, D.; Necula, G.; Paszke, A.; VanderPlas, J.; Wanderman-Milne, S.; et al. JAX: Composable Transformations of Python+NumPy Programs. Available online: https://github.com/google/jax (accessed on 6 March 2024).
34. Zhang, S.X.; Allcock, J.; Wan, Z.Q.; Liu, S.; Sun, J.; Yu, H.; Yang, X.H.; Qiu, J.; Ye, Z.; Chen, Y.Q.; et al. TensorCircuit: A Quantum Software Framework for the NISQ Era. *Quantum* **2023**, *7*, 912. [CrossRef]
35. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2017**, arXiv:1412.6980.