How Do Quantum and Photonic Deep Learning Platforms Compare to Current GPU Hardware?

Aarav Dudhia

Received September 25, 2024
Accepted October 28, 2024
Electronic access November 15, 2024

Moore's Law discloses that the number of transistors on a chip doubles every two years. As a result, transistor size significantly decreases and it becomes harder to continuously shrink their size. The laws of quantum physics start to govern transistors at such sizes, which could render current GPU systems useless in the future. Therefore, alternative systems, such as photonic and quantum systems are being researched. This paper uses systems that have importance placed on neural networks represented by hardware and also tested in experiments with data. This paper compares these three systems (classical, photonic, and quantum) in relation to neural networks and deep learning overall in four areas: speed, energy consumption, accuracy, and versatility. A Figure of Merit (FoM) is also utilized for more direct comparison in accuracy. Classical GPUs, known for their high versatility and ability to parallel process, dominate current ML tasks due to their extensive customization capabilities via languages like Complete Unified Device Architecture (CUDA). Photonic systems offer significant improvements in speed, such as only 570 picoseconds for end-to-end classification in one photonic system, while light could traverse the other photonic chip in about 31 picoseconds. Due to the speed of the systems, their energy consumptions (2.1375 nJ and 9.32 nJ respectively) are much lower than the classical GPU's value of 153580 J. However, these photonic chips only have 2 to 4 classes, and there are currently many scalability problems related to the inherent nature of photons (drop in accuracy due to photon loss). Quantum systems have many theoretical benefits, such as quicker runtime for linear algebra problems, but are not as developed as the photonic chips, as execution can take over 30 hours and an energy consumption of >32.4 MJ. Based on its versatility FoM value (81.5), classical GPUs remain the best option for neural networks, while photonic systems show potential because of their speed, energy consumption and decent FoM values (64.8 and 51.7 respectively). Meanwhile, there are possible benefits with quantum systems, however they are not a frontrunner in the near future because of a high energy consumption and low FoM (46.3).

Introduction

Moore's Law states that the number of transistors on a chip will double every two years. However, as the number of transistors increase, their size becomes increasingly smaller, approaching atomic scales, making it difficult to continuously shrink them. Additionally, at atomic scales, quantum physics starts to govern these transistors, as opposed to classical physics, which would then require a complete revamp of our current GPU systems for Deep Learning (DL) purposes. As of 2024, we are beginning to see the end of Moore's Law, as the rate of improvement from each predecessor chip is starting to decrease, and already, alternative systems have been researched [1]. Photonic and quantum technologies have emerged as promising platforms to possibly replace current systems and enhance DL models efficiency.

This paper aims to compare photonic, quantum, and traditional GPU-based neural networks to see whether the alternative systems are capable replacements of GPU systems. These three architectures will be compared in 4 different areas: Speed, Energy Consumption, Accuracy, and Versatility.

To properly compare the three systems, it is important to understand the idea of a neural network, which is the foundation of deep learning and what the systems are representing. A neural network is made up of many neurons arranged in interconnected layers. Every neuron takes in information, applies a weight to it, sums the weighted inputs, and then uses an activation function (such as sigmoid or ReLu) to decide its output, as seen in Figure 1. This is a computationally demanding operation in traditional neural networks, particularly as the number of layers and neurons increases [2].

Photonic systems offer benefits in terms of speed and energy efficiency by utilizing components based on light to carry out computations [4,5]. Similar to this, qubits, the equivalent of a classical binary bit in quantum mechanics, are used by quantum computing to analyze data in ways that traditional computers are unable to, possibly providing large speedups for training and inference [6]. The capacity of quantum and photonic systems to handle deep neural networks more efficiently than electronic-based platforms is the main emphasis of this study's evaluation of the potential efficiency improvements of these systems in machine learning. Finally, all of these novel systems

© The National High School Journal of Science 2024

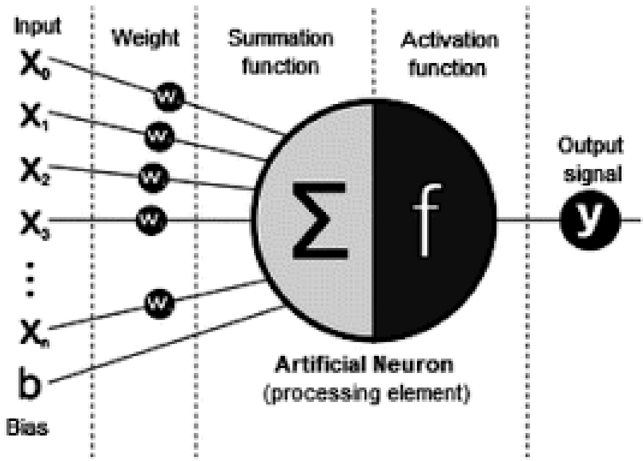


Figure 1: Image of a neuron[3]

are benchmarked to traditional GPU architectures that are widely used today for DL applications. The research paper will also use a figure of merit that is made up using the accuracy of the system and the number of classes to properly compare each other.

Methodology

To find relevant research papers regarding photonic and quantum neural networks, esteemed journals such as Nature, IEEE, SpringerLink, Quantum Journal were reviewed, along with their own references. A strong emphasis was put on research that placed importance on the hardware aspects of their neural networks. This was determined by the experimental data that was showcased in the research and seeing if that data could be applied in the figure of merit of this literature review. Although there were other research papers that mentioned using photonics or quantum systems to enable neural networks, the selected papers match the guidelines set by the search strategy the best.

Discussion

Classical GPUs

Currently, classical neural networks are run using graphic processing units (GPUs), such as NVIDIA Blackwell. First of all, it is essential to note the reasons why GPUs are used for neural networks instead of central processing units. GPUs can lower the time required for training, which involves matrix multiplication, because of their ability to parallel process. However, this parallel processing comes at the cost of higher power consumption because of the operations running in parallel. While CPUs have multiple cores, GPUs have thousands of cores. Being able to parallel process quickly and efficiently is crucial for training algorithms, such as backpropagation[7,8,9,10]. See the Appendix for more information regarding backpropagation.

GPUs are well equipped with handling matrix and vector calculations and have much more bandwidth than CPUs, which allows a high rate of memory transfer between the tensor cores (NVIDIA-specific cores that enable mixed-precision computing in high-performance tasks) of the system[12].

Additionally, based on the figure above, it can be seen that the cache, which is used for temporary storage, is more localized and closer to the ALUs in the GPU meaning an increased memory/computation speed compared to CPUs. For example, the NVIDIA Blackwell HGX B200 & B100 Tensor Core GPUs have 8 GPU cores, where each core has 8 Tb/s of memory bandwidth for an aggregate memory bandwidth of 64 Tb/s[13]. Increased memory bandwidth allows the trained weights to be fetched quickly, enabling the neural network to perform the desired classification. As a comparison, the theoretical maximum bandwidth for an Intel Core X-Series Processor is 0.094 Tb/s, which is considerably less than the 8 Tb/s memory bandwidth of a single Blackwell core[14].

Another advantage of the Blackwell GPU is the customizability that it offers, which can support a wide variety of neural networks. The C-based programming language developed by NVIDIA, CUDA (Complete Unified Device Architecture), allows the GPU to act as a general parallel processing platform instead of just focusing on graphics. CUDA facilitates parallel processing and allows multiple threads to be run at a given time, as shown in Figure 2. In the scheme of neural network training, these threads relate to the calculations regarding forward passing, loss calculation, and backward propagation. Having the ability to define the networks with software allows the neural networks run on the NVIDIA GPUs to be highly customizable and variable. In other words, it's easy to create networks of different sizes (inputs, weights, layers, etc.). Therefore, the main advantage of these GPU

 © The National High School Journal of Science 2024

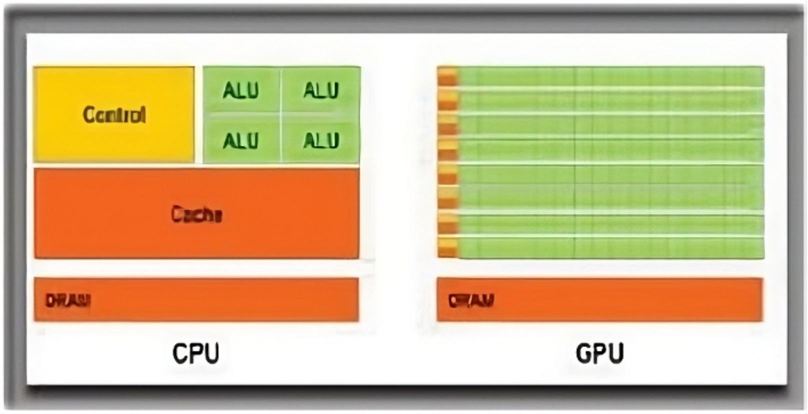


Figure 2: CPU Layout vs GPU Layout[11]

architectures is that the neural network structure is not married to the hardware and is thus highly versatile[11].

Even with these advantages with classical GPUs, there are still limitations that are present. GPUs are limited by both their clock speed and memory access time[13]. As neural networks grow larger and more complex, these constraints have a significantly greater impact. Additionally, raw data (which is also ever-increasing) would have to be converted to an electrical, digitized format, increasing the need for large memory and decreasing the processing speed[5].

Photonic Systems

Photonic neural networks have been proposed as an alternative to current GPU-powered neural networks to address the aforementioned bandwidth limitations present in GPU-based networks. Photonic processing allows for high bandwidths as optical signals are no longer limited by parasitic capacitances and resistance, therefore making PDNNs (photonic deep neural networks) a strong candidate for quick and energy-efficient processors (as quick as 570 picoseconds)[5]. Additionally, photonic systems offer lower energy consumption as operations can happen in a much shorter time as optical signal processing is determined by how quickly light can propagate through the system, which is typically close to the speed of light. As energy is power * time, less amount of time relates to lower energy consumption. Meanwhile, electrical signal processing happens in the digital domain, thus the maximum operations per second is limited by a clock of a few GHz[5]. While the classical GPUs used backpropagation calculations, the researchers for the photonic chip used propagation (the reverse of backpropagation) for image classification and used gradient descent for training.

Setup of Chip (An On-chip Photonic Deep Neural Network For Image Classification) As opposed to GPUs, the photonic chip described in An On-chip Photonic Deep Neural Network For Image Classification is analog-based, meaning that a continuous range of values is used instead of binary values (0 and 1). The structure of the neural network is similar to the classical one, as there are input layers, a hidden layer, and an output layer. However, the photonic chip differs in that light is coupled from the target image, using an array of grating couplers, into the nanophotonic waveguide. This coupling prevents the need to take a digital photo, store that photo in memory, and then retrieve it from memory when the classification needs to happen. In the photonic chip, applying weights (multiplication) to the inputs is done using PIN attenuators (value < 1), meaning that the value will always be less than the original input, and there will always be less signal than the start. This lossy behavior is one drawback of this architecture. After applying weights to the inputs, addition is performed by photodetection using SiGe photodiodes, creating photocurrents that are summed electrically by Kirchhoff's Current Law (KCL).

Then a linear trans-impedance amplifier converts the current (weighted sum) to a voltage. Finally, a micro-ring modulator (MRM) acts like the rectified linear unit (ReLu) function and is the non-linear activation function seen in classical neural networks[5]. Figure 1 of An On-chip Photonic Deep Neural Network For Image Classification shows the diagram and structure of the photonic neural network. As this chip is experimental and not fully scalable, only a two-class and a four-class dataset were able to be tested. For the two-class classification, the authors used 216 letters per iteration with an equal split between 'p' and 'd', while for the four-class they used 432 letters per iteration with an equal split between 'p', 'd', 'a', and 't'. The results show promise as the two-class dataset had an accuracy of 93.8%, while the four-class dataset was 89.8% accurate. But as a comparison, the four-class dataset had been tested with a Python Keras environment and recorded an accuracy of 96%[5].

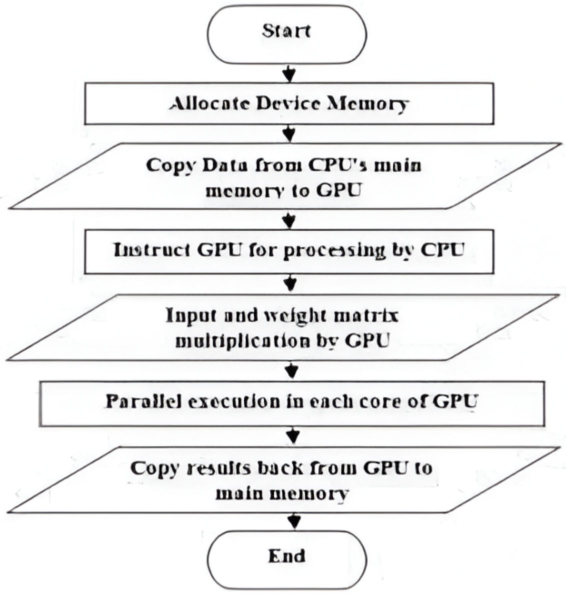


Figure 3: CUDA Architecture Flow [11]

Background on Deep Learning with Coherent Nanophotonic Circuits

The fully optical neural networks (ONNs) described in this research paper are an exciting alternative to current systems because linear transformations can occur at a rate of more than 100 GHz, and the ONN relies mainly on matrix multiplication. Power consumption can be near zero in regard to calculating matrices, similar to how a simple lens can execute a Fourier transform without any use of power. The authors claim that a future system may be designed without having to burn as much power for optical phase modulation. However, their current system uses 10 mW per modulator. However, these architectures rely on bulk components such as fibers and require a laser, which is not the case for traditional GPU-based neural networks. Using a laser also increases system power. Because this system is integrated into a photonic chip, the neural network does not require bulky components. Therefore, the necessity for phase stability and large neuron counts serve as a hindrance. However, the theoretical approach with nanophotonic circuits can facilitate forward propagation two times quicker than current electronic or hybrid systems with the speed being proportional to the neuron count [4].

Theoretical Approach

The abstract idea of the neural network is similar to the other networks previously mentioned, with an input layer connected to at least one hidden layer and then an output layer [2]. Weights and biases are present as part of the matrix multiplications along with a nonlinear activation function. Backpropagation also optimizes and trains the neural network. The input is first processed to a high-dimensional vector on a computer, and the signals are then encoded in the optical pulses' amplitudes, implementing an ONN. Each layer of the ONN has two parts: the optical interference units (OIUs) for matrix multiplication and optical nonlinearity units (ONUs) for nonlinear activation functions. Figure of 2 of Deep Learning with Coherent Nanophotonic Circuits showcases the schematic and a micrograph illustration of the complete ONN. In theory, the neural network can have arbitrary depth and dimensions in the optical domain [4]. This is important as this type of scalability would not be possible in classical electronic systems.

OIU and Matrix Multiplication

Using singular value decomposition (SVD), a matrix (M) can be decomposed in the following form $M = U\Sigma V^{\dagger}$: U is an $m \times m$ unitary matrix, $\sum$ is an $m \times n$ rectangular diagonal matrix (non-negative real numbers), and $V^{\dagger}$ is the complex conjugate of the n x n matrix V. U and $V^{\dagger}$ can both be implemented with beamsplitters and phase shifters, while $\sum$ can be implemented with optical attenuators. The authors claim that with non-heater-based phase shifters, the chip can passively operate on the supplied input light. While the authors claim a future chip does not consume power, power is still required for generating the input light [4].

The ONU

This paper proposes using saturable absorption as nonlinear. A realistic saturable absorption is a dye, semiconductor graphene,

 © The National High School Journal of Science 2024

or saturable amplifier. However, this paper does not use nonlinearity optically. Rather, they simulate the saturable absorption via a computer. This is different from the purpose of computing something at light speed since, ultimately, the system is going to be limited by the slow speed of the classical CPU. Therefore, this system cannot be an alternative as of right now, as it relies on a classical system.

Time Estimation for Energy Consumption

From Figure 2 of Deep Learning with Coherent Nanophotonic Circuits, it can be approximated that the chip is $\sim$1.6mm long. For silicon, the waveguide index is 4, so the speed of light is c/4, which can be used to find the total time for light to travel $\sim$1.6mm in silicon. $\frac{1.6\,\mathrm{mm}\times 4}{c} = 21\,\mathrm{ps}$ Then assume,there is 100GHz PD bandwidth. Therefore, photonic to electrical conversion is 1/100GHz, which is 10ps. So total time through a photonic chip is 21ps + 10ps = 31ps.

Experiment

A two-layer neural network was used to train vowel recognition. Three hundred sixty data points were generated by 90 different variables (half of the data was used in training with backpropagation). The ONN got 138/180 (76.7%), while the 64-bit computer got 165/180 (91.7%). Both systems were good at classifying but the ONN had limited resolution, decreasing the accuracy. Through the experiment with the ONN, a different training method was explored as backpropagation can be inefficient with recurrent and convolutional neural networks. Hence, ONNs use forward propagation with the finite difference method. Forward propagation could be two times faster than classical GPU neural networks because of the clock speed limitation already discussed by the other source. The matrix multiplication that is done fully optically also helps at that point. Maintaining the phase modulator requires 10 mW, but with a material change, no power may be required. Then, total power consumption would be limited by physical size, spectral bandwidth (THz), and photodetection rate (100 GHz). To match the optical network's computational speed, a computer would need to do $2m \cdot N^2 \cdot 10^1 1$ FLOPs (floating point operations per second), where m is the number of layers and N is the number of nodes[4].

Scalability Issues with Photonic Systems

Although these photonic systems can be implemented through experiments and have a lot of theoretical benefits, there are still scalability issues currently. Firstly, there is the problem of photon loss, which is when photons scatter and fail to reach their destination. As the number of photons increases, the magnitude of error increases. Additionally, integrating a large number of optical components, such as beam splitters, modulators, and detectors, into a single photonic circuit is still a significant challenge. The size of these circuits are constrained by the physical space needed for these components. Along with size concerns, fabricating such a photonic chip is quite complex and as the number of components grows, it becomes difficult to ensure consistency from all the elements. Finally, noise and crosstalk will always remain in photonic systems, reducing accuracy and making the system even harder to implement[15].

Quantum-Classical Hybrid Neural Networks

Hybrid neural networks (HNNs) utilize the components of both classical systems and quantum circuits to decrease computation time dramatically. Problems in fields, such as cryptography, chemistry, data analysis, and different optimization problems can be solved better by quantum computers, thus also likely to benefit from quantum machine learning (QML)[16].

To be able to use backpropagation with a quantum circuit (QC), an algorithm to calculate a gradient is necessary: gradient $= \mathrm{QC}(\Theta+\varepsilon) - \mathrm{QC}(\Theta-\varepsilon)$, so that the QC can be differentiable as part of the backpropagation algorithm. In regular backpropagation training algorithms, the chain rule is used to calculate the derivatives (gradients). However, to work with the QCs, the formula above is required, where Θ is a parameter of the QC and ε is the shift. The gradient is evaluated as the difference between the QC at the right shift $(\Theta+\varepsilon)$ and left shift $(\Theta-\varepsilon)$.

As quantum computers are not mainstream yet, the neural networks presented in Hybrid classic-quantum neural networks for image classification are created classically using the PyTorch library[16]. For quantum simulation and realistic QCs, the authors used the Qiskit software development kit, which limits this approach as a present-day alternative as the quantum part is reliant on a GPU. The neural network in the experiment consists of two convolutional layers, three linear layers, and a QC. The convolutional layers (both use the ReLU activation function) can convert the input images to vectors with 400 elements. Next, two of the linear layers also employ ReLU, where the third one uses tanh to normalize the outputs between -1 and 1. These initial layers decrease the number of input variables from 400 to the number of qubits in the quantum layer. The values after the third linear layer are multiplied by π because QCs rely on qubit shifts, which are measured by rotation angles. Sigmoid is then used on the quantum layer outputs as an activation function. For the control and reference model, an entirely classical neural network was created by replacing the QC with a linear layer. While the last QC layer utilizes quantum logic gate circuits to perform the multiplication/addition required for the network, the first classical layers still rely on the classical GPU-based system for the computations[16].

Data and Experiment

The datasets that the HNN was tested on include CIFAR10, and CIFAR100. CIFAR10 contains 60000 32x32 RGB images of 10 different classes (split 5000 for training and 1000 for testing in each class), while CIFAR100 contains 100 classes that can be divided into 20 superclasses. Each class has 500 images for training and 100 testing images for a total of 60000 images. For general information, a class represents a possible outcome that an input can belong to, while a superclass is a broader, generalized category that encompasses classes and is useful for maintaining a hierarchy. For example, lions, tigers, and elephants can be considered as classes, while animals would be a super class [16].

The figure below displays the QC simulated. Figures 5a and 5b below relate to the loss and accuracy plots during training for the Ry quantum circuit model with 4 qubits for solving 10 classes of the CIFAR10 dataset (300 images were used for training, while 50 images were used for validation per each class). Meanwhile, figures 5c and 5d relate to a more complex QC that has 7 qubits to solve 100 classes of the CIFAR100 dataset (100 images were used for training, while 50 images were used for validation per class). The training process took 30 hours to complete, as only a quantum simulator on a computer was used. Figure 5e demonstrates when a QC goes wrong, the accuracy drops, and the training process stops. These cases were ignored in the final results [16].

Overall, HNNs are capable of solving multiclass image classification problems, but as the number of classes increases, the accuracy decreases in comparison to classical neural networks.

As the number of classes, or the number of possible outputs, increases, the hybrid neural network's accuracy decreases, while the classical neural network stays more consistent. This is apparent in Table 1 of Hybrid classic-quantum neural networks for image classification.

The computational speed also decreased, as some attempts for the CIFAR100 dataset took several days to train. However, QC development is still very young, but there is potential in combining them with current hardware, such as CPUs, GPUs, and tensor processing units. Future plans include creating a more feasible approach and using more complex datasets [16].

Theoretical Benefits of Quantum Systems

Theoretically, quantum systems can solve certain problems faster than classical algorithms because of the qubits' inherent qualities of superposition and entanglement. Linear algebra problems, such as the one involving matrices, can be done quicker. For example, to solve CX decomposition, which is when a matrix is decomposed and approximated using a subset of its columns, quantum decomposition algorithms offer a speedup in runtime $O(n^1.75)$, compared to classical algorithms $O(n^2)$ [17].

Data handling is another deep learning concept where quantum systems can, theoretically, perform better classical counterparts through quantum clustering algorithms, such as the quantum-KNN algorithm, quantum annealing algorithm, and a Variational Quantum Eigensolver [18] .

Scalability Issues with Quantum Systems

Quantum systems have a lot of theoretical potential, but currently they are limited to either computer simulation or specially designed labs and there are many challenges with implementing them. First of all, quantum systems are extremely sensitive to noise, affecting the performance and accuracy of the qubits in a system. As a result, to perform computations, such as Shor's Algorithm, it is estimated that millions of physical qubits are required. Additionally, qubits require temperatures of absolute zero to function, which is a significant engineering challenge currently [19].

Results

As a direct comparison between the systems is not possible, the results can't provide any definitive claim, but they do have some importance. Additionally, the datasets are very specific which prevent apples-to-apples comparison between architectures. However, the FoM can be used as a first order mechanism to compare the different systems. Based on Table 2, currently, classical GPUs are best for deep learning because they have higher accuracy, high versatility (because of programming languages like CUDA), and the highest Figure of Merit (FoM = the difference between the accuracy of the system and the standard probability of correctly guessing an output based on the number of classes). However, the photonic systems discussed have significant potential based on the increase in speed as seen in An on-chip photonic deep neural network for image classification and Deep learning with coherent nanophotonic circuits and the same can be said with energy consumption. The photonic systems show extreme promise for low energy consumption for end-to-end classification, as the numbers for both systems are multitudes smaller than the energy consumption of a normal GPU during the execution of the neural network. However, the versatility is important to note, as the system described in An on-chip photonic deep neural network for image classification can only be used with a couple of handwritten letters, and the input data set size is limited by the hardware (input grating coupler array). The system in Deep learning with coherent nanophotonic circuits can be used in more applications because it primarily focuses on matrix multiplication, but it still relies on a computer, which limits the benefits that a full optical system could provide. Inherent photonic problems, such as photon loss and the requirement of fitting in multiple optical components in a single chip are

© The National High School Journal of Science 2024

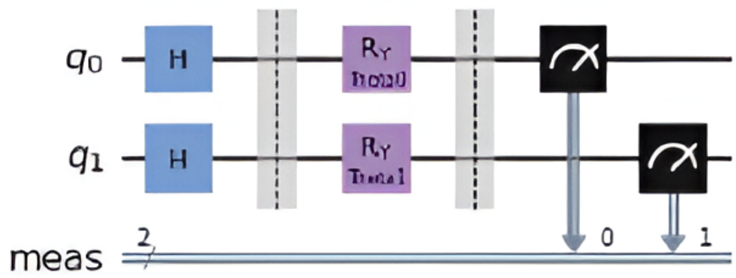
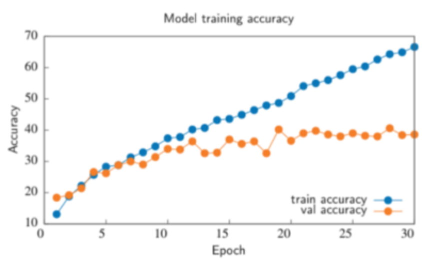


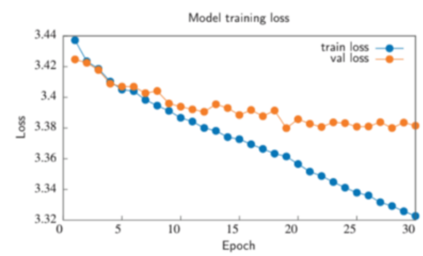
Figure 4: 2-qubit QC with one trainable parameter per qubit [16]

Table 1 [16]

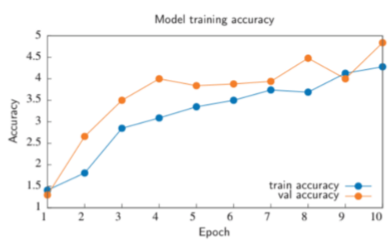
Classes	Hybrid NN accuracy	Classical NN accuracy
0-3 classes	79.60%	67.40%
0-5 classes	52.00%	76.40%
0-7 classes	56.58%	71.20%
0-9 classes	45.13%	70.10%



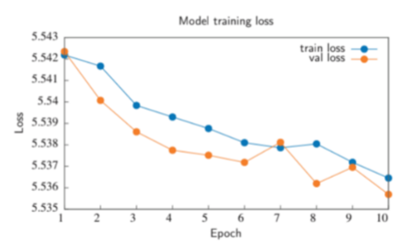
a. Train and validation loss during model training on CIFAR10



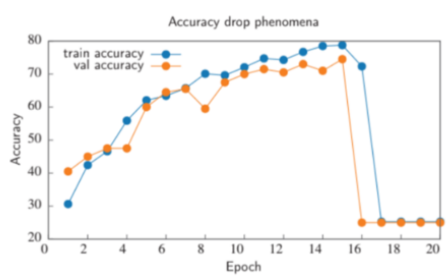
b. Train and validation accuracy during model training on CIFAR10



c. Train and validation loss during model training on CIFAR100



d. Train and validation accuracy during model training on CIFAR100



e. When a point in the quantum layer goes wrong, the network accuracy drops drastically and the network training process stops. The reason for this is unknown, but it is suspected that it might be related to K-fold cross-validation, where one K-fold attempt was much worse than other K-folds

Figure 5: 5a-5d relate to training and validation results for different datasets, while 5e showcases the accuracy drop phenomena [16]

Table 2: Key Metrics Compared

	Classical GPU*	Photonic	Photonic	Hybrid- Quantum
Speed	2194 s	HS: 570 ps LS: 1us	31 ps	>30 hrs
Power Consumption	70 W	HS: 3.75 W*** LS: 2 W***	560 mW**** + 300 W*****	300 W*****
Energy Consumption	153580 J	HS: 2.1375 nJ LS: 2 uJ	9.32 nJ	>32.4 MJ
# of classes	14	2, 4	4	3
Accuracy (A)	88.60%	2: 93.8%, 4: 89.8%	76.70%	79.60%
Versatility	High	Low	Medium	Medium
Dataset	THUC*	Handwritten letters: "t", "p", "d", "a"	Vowel recognition, "i", "e", "y", "u"	CIFAR10, and CIFAR100
FoM: $A - \frac{1}{\#of classes} \times 100$	81.5	43.8, 64.8	51.7	46.3

hurdles that still need to be overcomed. As of right now, quantum systems require the most work, as they too rely heavily on computers for quantum simulation and can take over 30 hours to train completely. There is versatility with the hybrid-quantum system in Hybrid classic-quantum neural networks for image classification, as many datasets were able to be tested with the software qubits. Theoretically, there is potential in these quantum systems, as there can be an improvement on runtime for ML problems, such as CX composition, but physical implementation is still far away, due to noise and temperature sensitivity. Although we are approaching the theoretical limit set by Moore's Law, classical GPU systems are the best for current deep learning applications, while photonic platforms may be implemented in the near future[20] implements a CNN on a GPU and CPU. The CNN uses a Chinese Character dataset, THUC, which has 14 classes and reports an accuracy of 88.6% in[21].

**default batch size is 128

***HS - high speed; LS - low speed

****10mW * 56

*****The power consumption is based on the computers used in the experiment, which are not given. A standard wattage of 300 is assumed.

References

1 Contrary, *The evolution of computer chips*, https://www.contrary.com/foundations-and-frontiers/evolution-of-chips, n.d.

2 NVIDIA Developer, *Artificial Neural Network*, https://developer.nvidia.com/discover/artificial-neural-network, n.d.

3 Santhiyaarun, *Neuron in deep learning*, 2024, https://medium.com/@santhiyaarun124/neuron-in-deep-learning-f33b9fc4f270.

4 Y. Shen, N. C. Harris, S. Skirlo *et al.*, *Nature News*, 2017.

5 F. Ashtiani, A. J. Geers and F. Aflatouni, *Nature News*, 2022.

6 Microsoft Azure, *What is a qubit?*, https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-a-qubit, n.d.

7 NVIDIA Technical Blog, *A data scientist's guide to gradient descent and backpropagation algorithms*, 2022, https://developer.nvidia.com/blog/a-data-scientists-guide-to-gradient-descent-and-backpropagation-algorithms.

8 J. Jordan, *Neural networks: Training with backpropagation*, 2019, https://www.jeremyjordan.me/neural-networks-training/.

9 3Blue1Brown, *What is backpropagation really doing? | Chapter 3, deep learning*, 2017, https://www.youtube.com/watch?v=Ilg3gGewQ5U&t=308s, Video.

10 S. Kostadinov, *Understanding backpropagation algorithm*, 2019, https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd.

11 N. Singh and S. P. Panda, IEEE conference publication.

12 Nvidia, *Nvidia tensor cores: Versatility for HPC AI*, https://www.nvidia.com/en-us/data-center/tensor-cores/, n.d.

13 Nvidia, *Nvidia Blackwell architecture technical brief*, https://cdn.prod.website-files.com/61dda201f29b7efc52c5fbaf/6602ea9d0ce8cb73fb6de87f_nvidia-blackwell-architecture-technical-brief.pdf, n.d.

14 Intel, *Theoretical maximum memory bandwidth for Intel® Core™ processors*, https://www.intel.com/content/www/us/en/support/articles/000056722/processors/intel-core-processors.html, n.d.

15 K.-L. Hsieh and S.-K. Hwang, IEEE Photonics Conference (IPC), 2016, pp. 27–28.

16 Y. Trochun, S. Stirenko, O. Rokovyi *et al.*, IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems, 2021, pp. 968–972.

17 A. Prakash, *Quantum algorithms for linear algebra and machine learning*, 2014, https://www2.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-211.pdf.

18 P. Bermejo and R. Orús, *Scientific Reports*, 2023, **13**, 13284.

19 J. E. Bourassa, R. N. Alexander, M. Vasmer *et al.*, *Quantum*, 2021, **5**, 392.

20 Y. Sun, Z. Ou, J. Chen *et al.*, *SpringerLink*, 1970.

21 THUCTC, , http://thuctc.thunlp.org/, n.d.

 © The National High School Journal of Science 2024

Appendix

Backpropagation is a method of training a neural network by minimizing the cost function (uses the difference between predicted results and actual results) to adjust weights and biases with a gradient descent equation. The gradient informs how to adjust the weights to minimize the cost function. Adjusting the weights in a way that gets closer to the desired output benefits the network by increasing its accuracy. This process is repeated until the cost function is at its minimum value. To avoid local minimums and truly find the global minimum of the cost function, backpropagation may be performed multiple times with different data / different initial weight values, increasing the confidence that a truly global minimum is found for the cost function.

Let x = input, W_n = weight, b_n = bias, and σ = activation function (sigmoid).

The output of a given layer n is equal to: $z_n = W_n \cdot x + b_n$ and after activation: $a_n = \sigma(z_n)$.

Suppose the layer $n+1$ is the output layer. The output of the layer $n+1$ is equal to: $z_{n+1} = W_{n+1} \cdot a_n + b_{n+1}$, and after activation, the final output is: $\hat{y} = \sigma(z_{n+1})$.

To calculate the cost function, the mean squared error formula is used to measure the difference between the predicted output $\hat{y}$ and the actual output y: $C = \frac{1}{2}(\hat{y} - y)^2$.

Updating the weights requires the derivative (gradient) of the cost function: $\frac{\partial C}{\partial \hat{y}} = \hat{y} - y$.

Using the chain rule, we can compute the derivative of the cost function with respect to z_{n+1}: $\frac{\partial C}{\partial z_{n+1}} = \frac{\partial C}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_{n+1}}$.

Since $\hat{y} = \sigma(z_{n+1}) = \frac{1}{1+e^{-z_{n+1}}}$, we have $\frac{\partial \hat{y}}{\partial z_{n+1}} = \hat{y}(1-\hat{y})$, and so $\frac{\partial C}{\partial z_{n+1}} = (\hat{y} - y) \cdot \hat{y}(1-\hat{y})$.

Now, we can find the derivative of the cost function with respect to W_{n+1}: $\frac{\partial C}{\partial W_{n+1}} = \frac{\partial C}{\partial z_{n+1}} \cdot \frac{\partial z_{n+1}}{\partial W_{n+1}}$.

Since $\frac{\partial z_{n+1}}{\partial W_{n+1}} = a_n$, we have $\frac{\partial C}{\partial W_{n+1}} = (\hat{y} - y) \cdot \hat{y}(1-\hat{y}) \cdot a_n$.

For the bias b_{n+1}, $\frac{\partial C}{\partial b_{n+1}} = \frac{\partial C}{\partial z_{n+1}} \cdot \frac{\partial z_{n+1}}{\partial b_{n+1}}$, and since $\frac{\partial z_{n+1}}{\partial b_{n+1}} = 1$, we have $\frac{\partial C}{\partial b_{n+1}} = (\hat{y} - y) \cdot \hat{y}(1-\hat{y})$.

These calculations were done only for the output layer $n+1$, but similar calculations would also be done for the hidden layer or layers n.

To update the weights and biases, the learning rate (η) is required, as it controls how much the gradient changes the network in the opposite direction. A small η results in a smaller change, while a large η results in a larger change.

Finally, the gradient descent equations for updating the weight and bias of the output layer are:

$W_{n+1} = W_{n+1} - \eta \frac{\partial C}{\partial W_{n+1}}$

$b_{n+1} = b_{n+1} - \eta \frac{\partial C}{\partial b_{n+1}}$