2023 OPEN HACKATHON SUBMISSION:

GENOMIC ERROR CORRECTION AND RESEQUENCING WITH GPU-ACCELERATED QCNN

BY TEAM BCQENTANGLEMEN

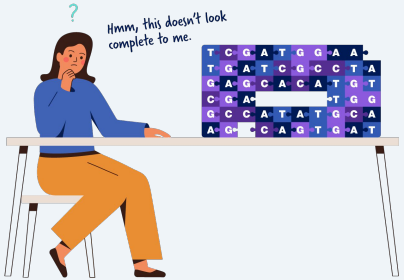
Ethan Rajkumar, Pranav Kairon, Luis Mantilla, Mohammed Mostaan

TABLE OF CONTENTS

GENOMIC SEQUENCING

Genomic sequencing = reading + analyzing a DNA sequence.
*Genome = composition of four nucleotide bases (adenine(**A**), guanine(**G**), cytosine(**C**), thymine(**T**))*

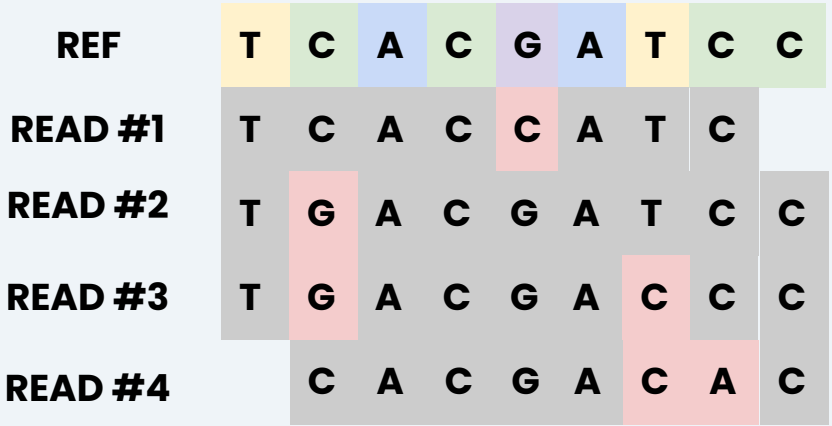
- *Reveals genetic makeup and functions of an organism*
- *Identifies genetic variations related to traits or diseases for personalized medicine*
- *Informs drug development*
- *Enhances understanding of evolution*
- *Cost has decreased, making it more accessible for research and healthcare*
- *Expected to continue impacting biology and medicine*
- *Addressing problems with genomic sequencing can lead to valuable solutions.*



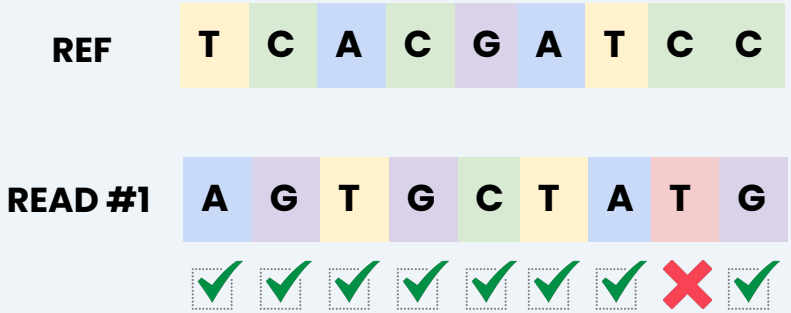
Images are taken from the University of Washington and National Genome Research Institute

PROBLEM STATEMENT

One of the significant challenges that current genomic sequencing methods face is error correction of readings, which can introduce errors and confound downstream analyses. Traditional algorithms, such as convolutional neural networks (CNNs), can be effective for error correction but may yield inconsistent results depending on the length of the sequence. CNNs can be used for:

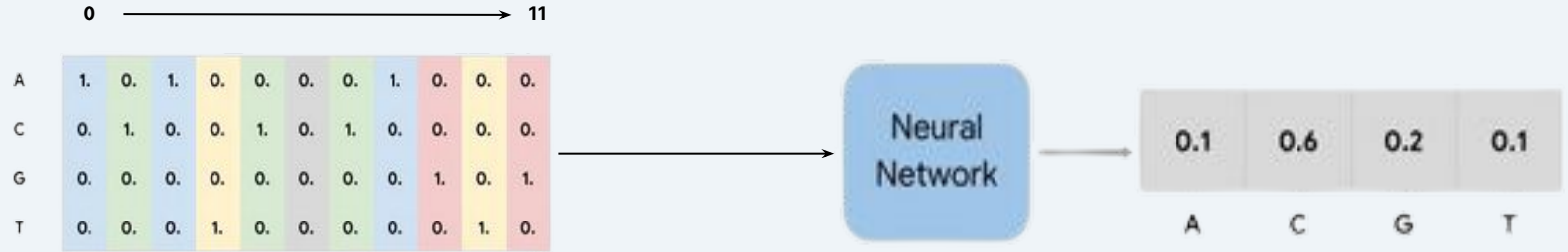


1. CONSENSUS ERROR-CORRECTION



2. SINGLE ERROR-CORRECTION

SINGLE-READ ERROR CORRECTION (SREC)



K-mer counting breaks DNA sequences into smaller k-mers by sliding a window of fixed length over the sequence.
These k-mers are typically 20-30 base pairs long and are counted and stored in a hash table, where each k-mer is associated with a count representing its frequency in the sequence data.

K-mers that occur with low frequency are considered to be errors and are corrected or removed.

If there is a missing base pair a final prediction is made based on the highest probability.

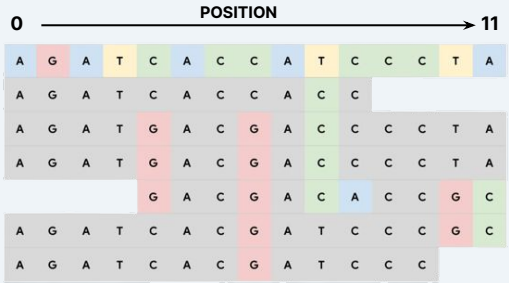
Improves accuracy, quality → Leads to better assembly and efficient downstream analyses

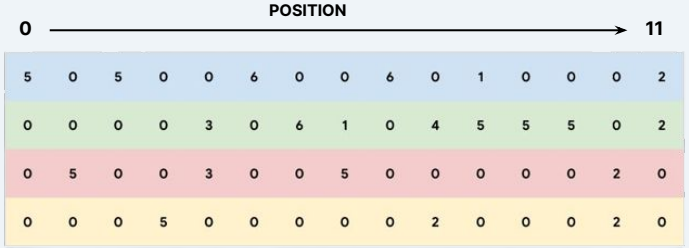
Can result in loss of low-frequency genomic variants or rare alleles → May be computationally intensive

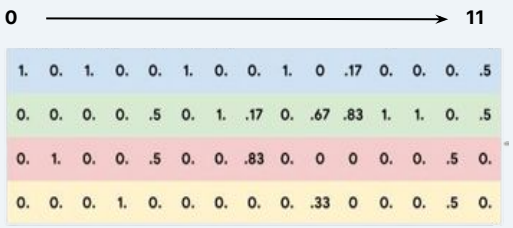
Images are taken from Google Nucleus.

CONSENSUS-BASED ERROR CORRECTION (CBEC)

compare multiple reads of the same region of DNA and use the most common nucleotide at each position as the corrected sequence.







The reference genome is segmented into anchors. Each read is aligned to the anchors using Smith-Waterman algorithm.

The alignment score depends on the number of matching bases, gaps, and a penalty for mismatched bases.

Advantages:

Relies on sufficient coverage of each DNA region to identify most common nucleotide

Some errors may not be corrected if present in all reads of a region

Reads are split into overlapping fragments, which are clustered based on their overlap patterns

Consensus sequences are generated for each cluster by identifying the most common nucleotide at each position

Consensus sequences are normalized and compared to a reference genome

Differences between the consensus and reference sequences are corrected by replacing the reference genome base with the consensus base

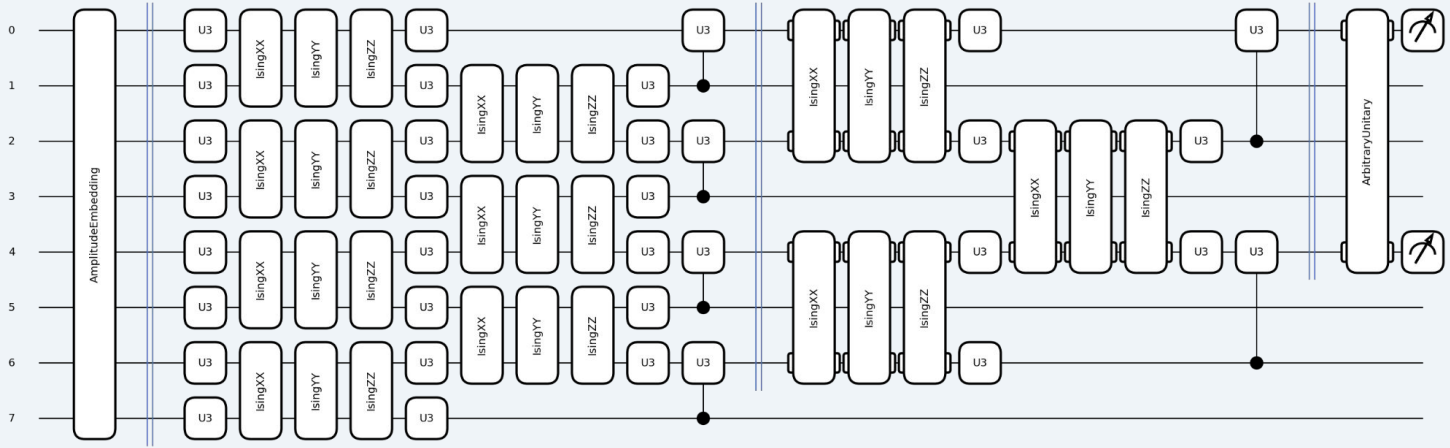
The genome is then assembled using these corrected consensus bases

This method helps improve accuracy and quality of genomic data analysis.

Images are taken from Google Nucleus.

QCNNs

QUANTUM CONVOLUTIONAL NEURAL NETWORKS



Quantum convolutional neural networks (QCNNs) are capable of performing complex computations in parallel and can potentially leverage quantum entanglement to enhance feature detection and classification. (*See Appendix for typical QCNN workflows*)

Since we are measuring the probabilities of 4 nucleotides at the end, we expect QCNN to perform quite well on such problems since it is naturally suited to output probability distributions

OBJECTIVE

Aim:

Investigate the potential of quantum neural networks (QNNs) in aiding with genomic sequencing error correction.

Milestones to Reach:

1. Convert genomic data into reference images to be fed into QCNN for gene correction using SREC

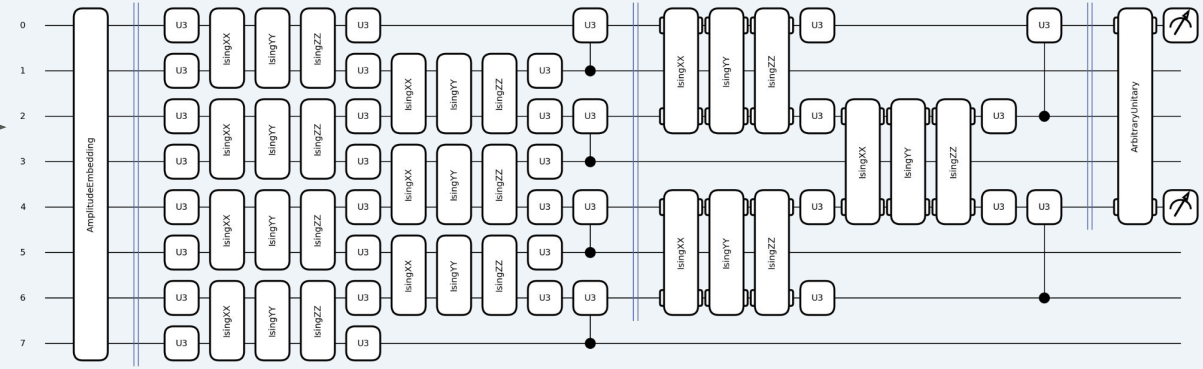
2. Develop a GPU-accelerated QNN architecture specifically for error correction in genomic sequences and train it on a large dataset with known errors.

3. Evaluate the QNN's performance on a separate set of genomic sequences with known errors and compare it to traditional error-correction methods.

4. Assess the accuracy and computational performance of the QNN-based method compared to other methods.

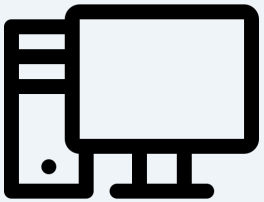
PROJECT WORKFLOW

DATASET INPUT



GPU-ACCELERATED QCNN

CLASSICAL COMPUTER



The classical computer will be used to optimize the loss function.

PROB OF AGTC

ACCURACY TESTS

INPUT IMAGES FROM SEQUENCES

The BAM files were obtained from Google Nucleus.

BAM stands for Binary Alignment Map containing the nucleotide sequence of the reads, along with additional information such as quality scores, mapping positions, and read orientations.

The following workflow was then implemented.

SEQUENCE READS

QUALITY CONTROL

GENOME ALIGNMENT

CLEANUP

VARIANT CALLING

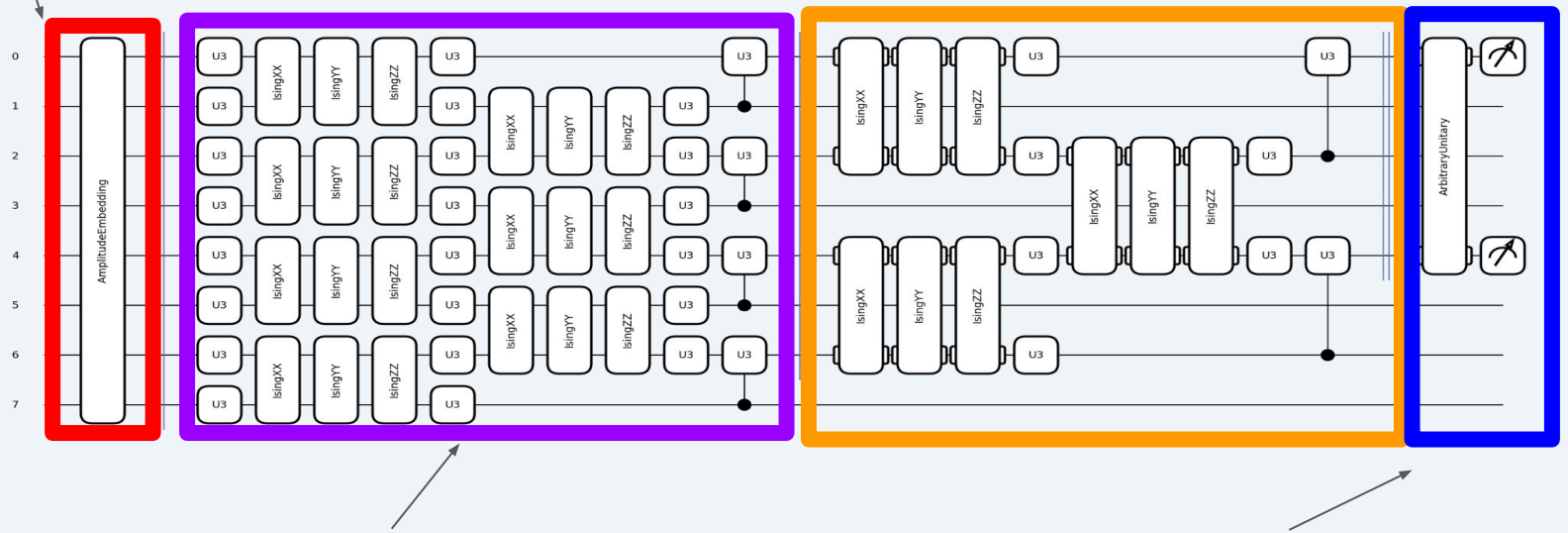
Here we use FastQ which is in orange and SAM for variant calling which is alignment of sequences

Variant Calling results in the following image which is used as our input for our QCNN model.

QCNN ALGORITHM:

DATA ENCODING: Encodes 7158 images in batches of 256 into the amplitude vector of 8 qubits.

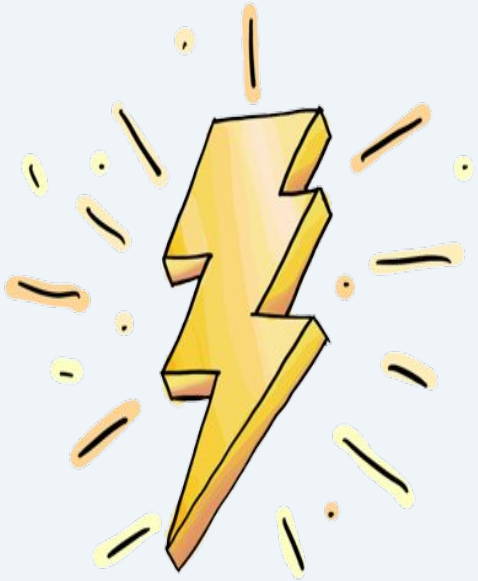
POOLING LAYER: performs data reduction by reducing the images by half. This is done through averaging or downsampling that is used to reduce the dimensionality of the data while preserving important features.



CONVOLUTIONAL LAYER: Input encoded high dimensional data goes through a convolutional layer that selects/ filters specific features in our case, for bp.

ARBITRARY UNITARY GATES:

GPU Acceleration



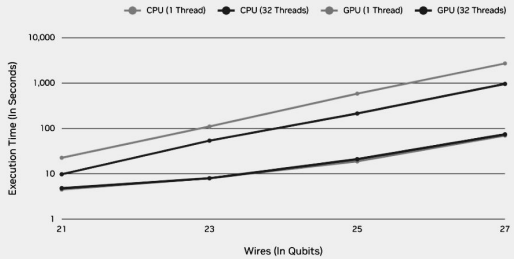
Pennylane Lightning GPU and NVDIA's QODA backend was used for statevector simulation.

Performance

State Vector Method

Quantum Machine Learning

CPU vs Single GPU (1 thread and 32 thread comparisons)



Evaluation of the Jacobian of a strongly entangling layered circuit leveraging adjoint backpropagation. Run lightning.gpu on an NVIDIA DGX A100, compared to lightning.qubit on an Epyc 7742 CPU. Results are averaged across 3 runs.

State vector simulation tracks the entire state of the system over time, through each gate operation. It's an excellent tool for simulating deep or highly entangled quantum circuits, and for simulating noisy qubits.

An **NVIDIA DGX™ A100** system with eight NVIDIA A100 80GB Tensor Core GPUs can simulate up to 36 qubits, delivering an orders-of-magnitude speedup on leading state vector simulations over a dual-socket CPU server.

cuStateVec has been adopted by leading publicly available simulators, including integrations into AWS Braket, Google Cirq's qsim simulator, the IBM Qiskit Aer simulator, and Xanadu's PennyLane Lightning simulator. Users leveraging lightning.gpu on AWS Braket experienced 900x Speedups and saved 3.5x on costs. It will soon support an even wider range of frameworks and simulators. Visit the **NVIDIA Technical Blog** for more details.

Training Curves: Classical CNN

CNN with Maximal Resources

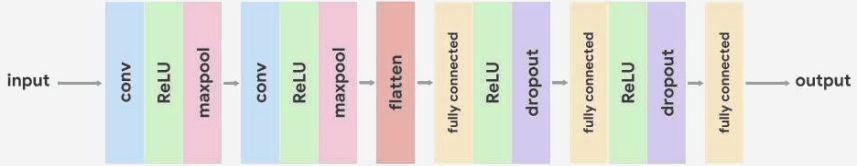
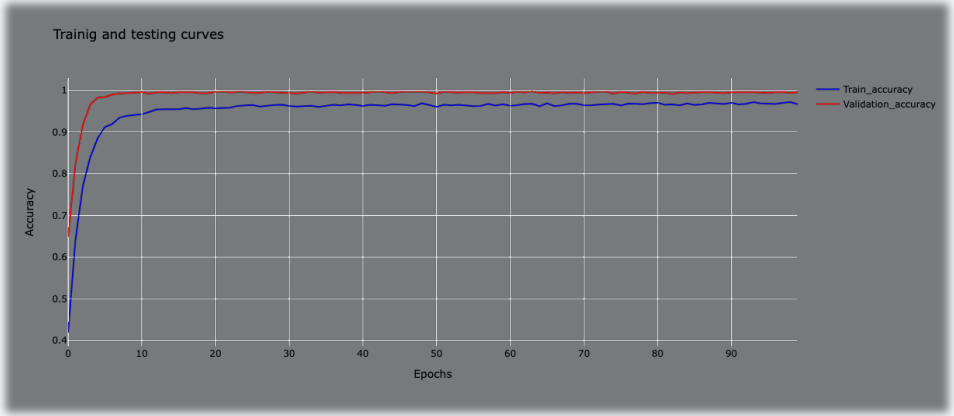


Figure 2: Neural network architecture used for error correction.

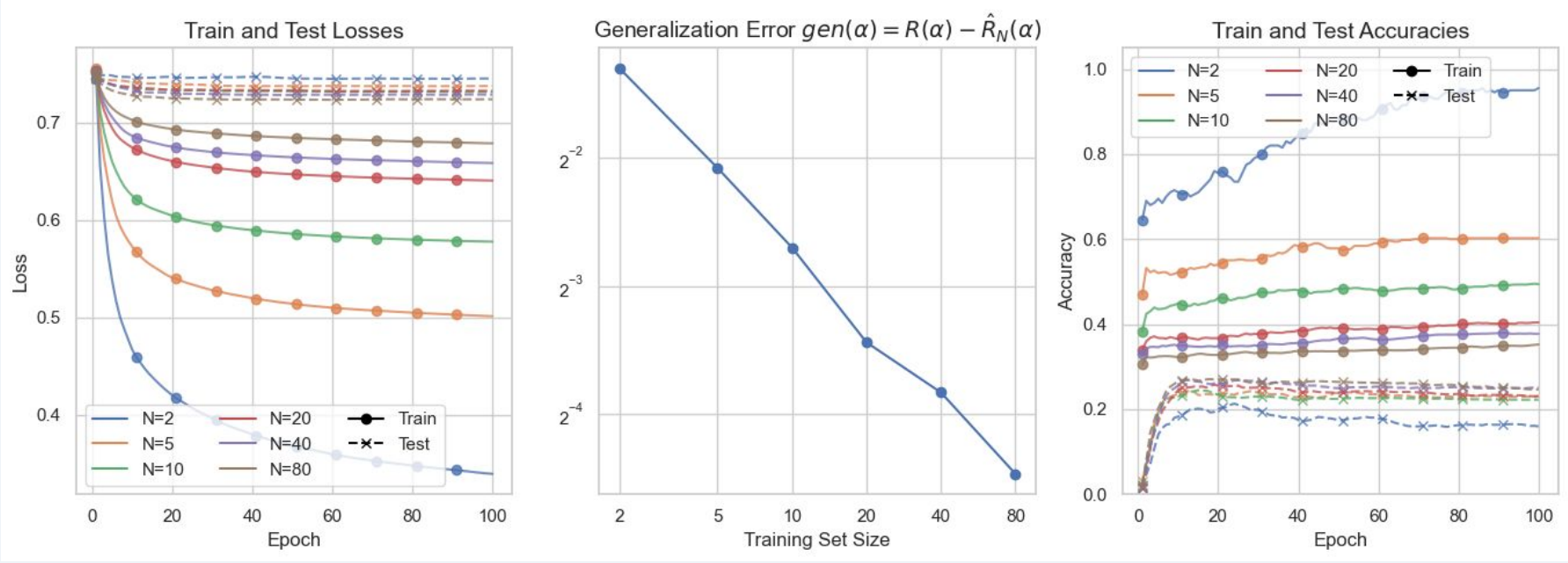


CNN with Resources equivalent to QCNN

Figure 2: Neural network architecture used for error correction.

We observe that while maximal CNN works well, CNN with fewer resources underperforms on dataset

Training Curves: Quantum CNN



Despite achieving a low accuracy of the QCNN, we see a monotonic increase in the accuracy as we increase the size of the training dataset. Here we train with 80 data points at most, as compared to the ~4000 data points used in the classical counterpart. We expect to see *better* performance when scaling

Implications

JOURNAL ARTICLE

The Unreasonable Effectiveness of Convolutional Neural Networks in Population Genetic Inference

Lex Flagel, Yaniv Brandvain, Daniel R Schrider

Molecular Biology and Evolution, Volume 36, Issue 2, February 2019, Pages 220–238,
https://doi.org/10.1093/molbev/msy224

We wanted to observe the performance of Quantum Convolutional Neural Networks on Genome sequencing datasets. We started by constructing images by reading genome sequences. We fed our data to Q-CNN and CNN. We found that while CNN performs quite well, it's performance suffers as we decrease the resources.

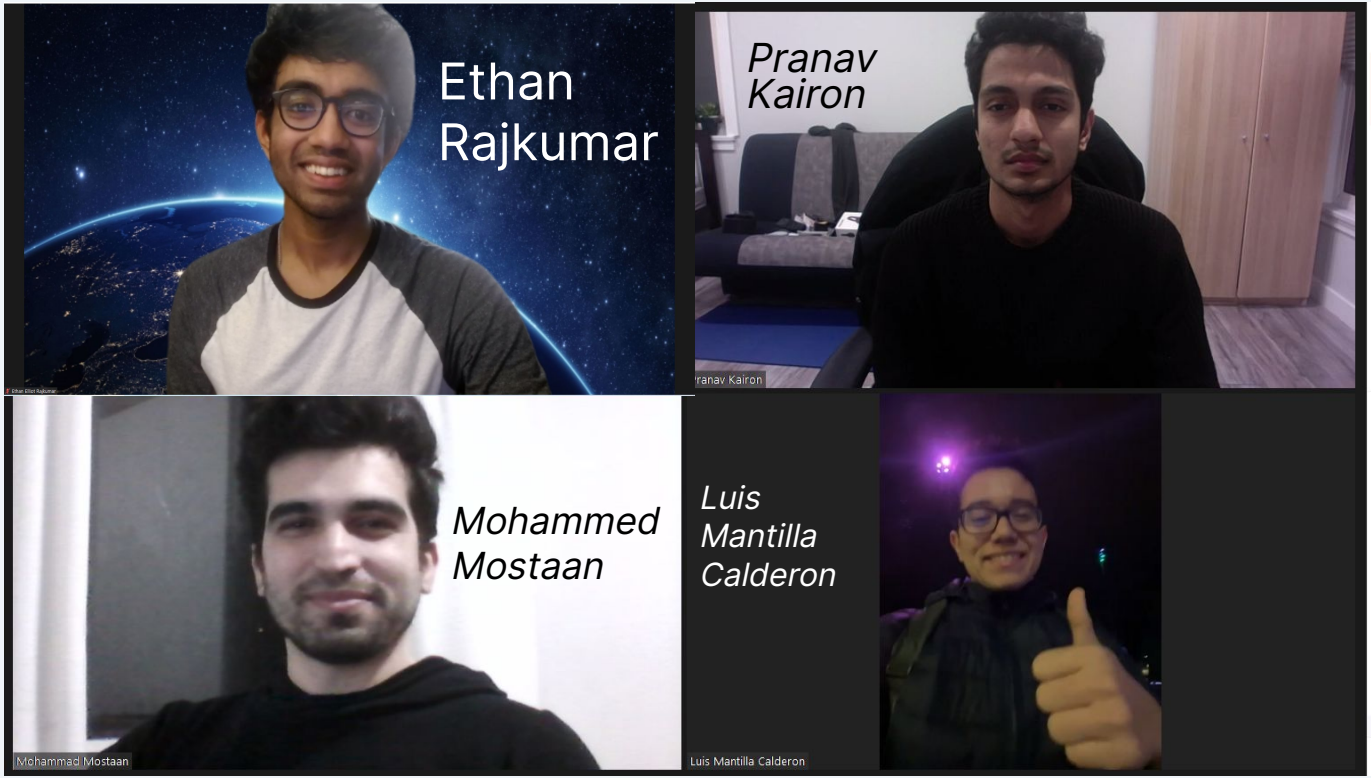
We implement our QCNN with GPU acceleration thanks to nVidia's QODA. We find that QCNN gives an average performance on our dataset. The performance can be increased by increasing the resources in terms of more pooling layers and variational parameters in QCNN

Our study opens up new avenues for application of quantum machine learning on biological datasets

Future Considerations

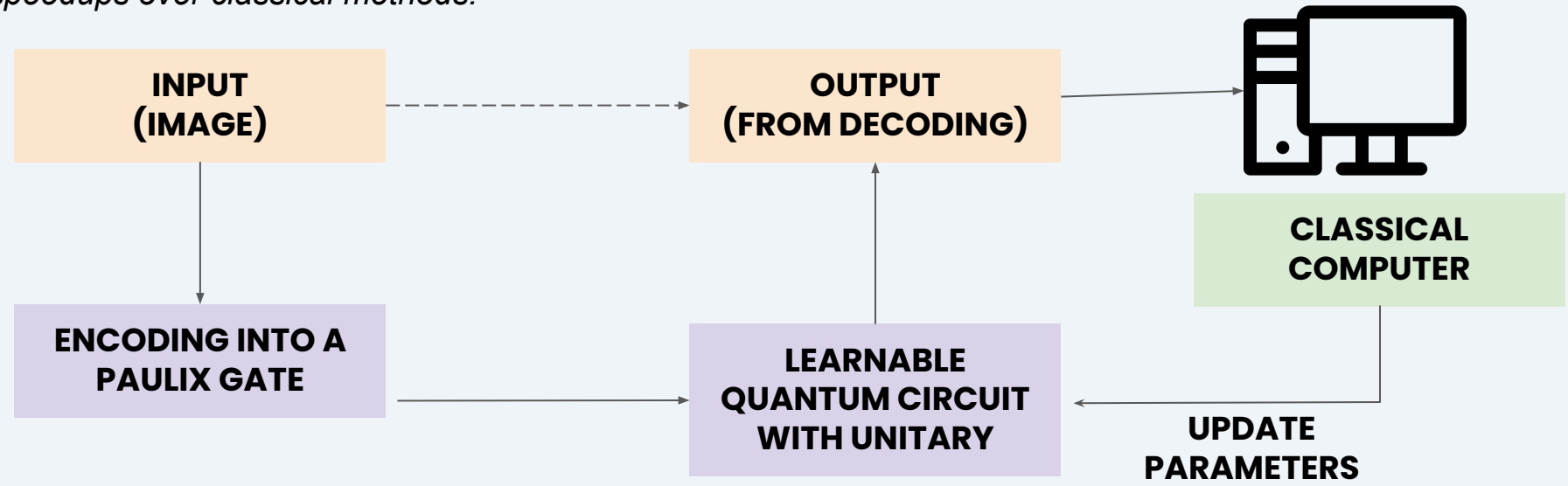
- ***Run on gate-based hardware! Compare performance with quantum statevector simulator…***
- ***Improved, variational optimization of tunable QCNN parameters***
- ***Ansatz improvements: account for additional, non-standard base pairs, more complex layers***
- ***Investigation of other models for Genome Sequencing (quantum metropolis MCMC, quantum Boltzmann machine)***

MEET THE TEAM



TYPICAL QCNN WORKFLOWS

QCNNs are designed to perform image recognition and classification tasks, and they can potentially provide exponential speedups over classical methods.



The following workflow was taken from Oh, Seunghyeok et al. "A Tutorial on Quantum Convolutional Neural Networks (QCNN)." 2020 International Conference on Information and Communication Technology Convergence (ICTC) (2020): 236-239.