Aliro™

Integrating AI and Quantum Technologies

Aliro

www.alirotech.com

Integrating AI and Quantum Technologies

Summary

In this white paper, we explore the synergies between quantum computing, artificial intelligence and machine learning, and quantum networks, providing an overview of the core technologies, the mutually beneficial potential in combining these technologies, as well as applications for enhancing computational and security capacities. We'll cover how AI and quantum technologies complement each other, offering unique opportunities to address optimization, error correction, and privacy challenges in secure networking and computing. We'll also look at future directions for using AI with quantum technology.

Introduction

Artificial intelligence and quantum networking each provide powerful capabilities as standalone technologies, revolutionizing fields from data processing and predictive analytics to network security and complex problem-solving. AI, with its ability to analyze large volumes of data and identify patterns, has quickly become a cornerstone of innovation in a variety of sectors. Meanwhile, quantum networking is creating a paradigm shift in data transmission and security, leveraging quantum mechanics to enable unprecedented levels of security to our communication networks.

How can these be used together?

The synergies that emerge when AI and quantum networking are combined are even more powerful. Each technology compensates for limitations in the other, creating a symbiotic relationship where the strengths of one amplify the other's capabilities. This white paper explores the potential synergies where AI can drive the scalability and efficiency of quantum networks, quantum networking provides the secure infrastructure needed to protect and extend AI applications, and other synergistic applications.

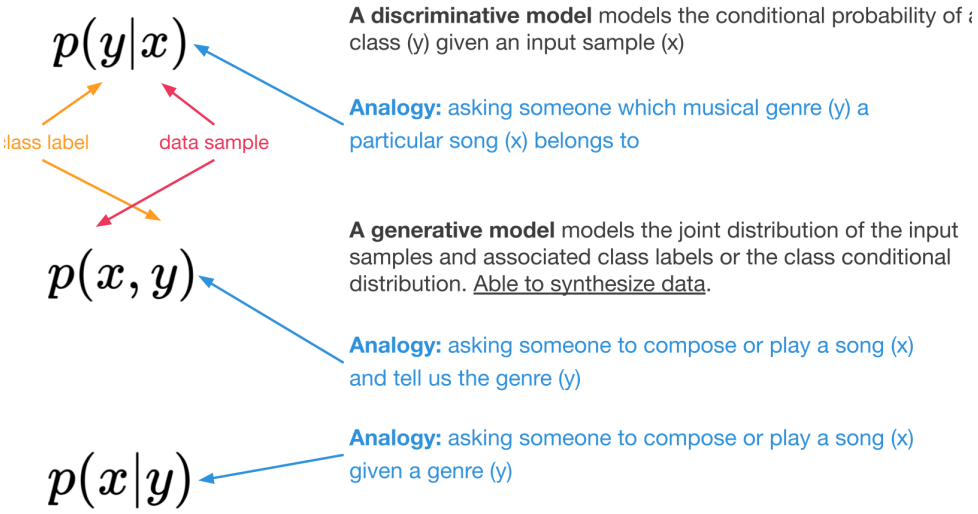
Classical Artificial Intelligence (AI) and Machine Learning(ML)

AI models can also be broadly categorized into types based on their functionality. Each model category serves distinct purposes, from data classification to the creation of new data:

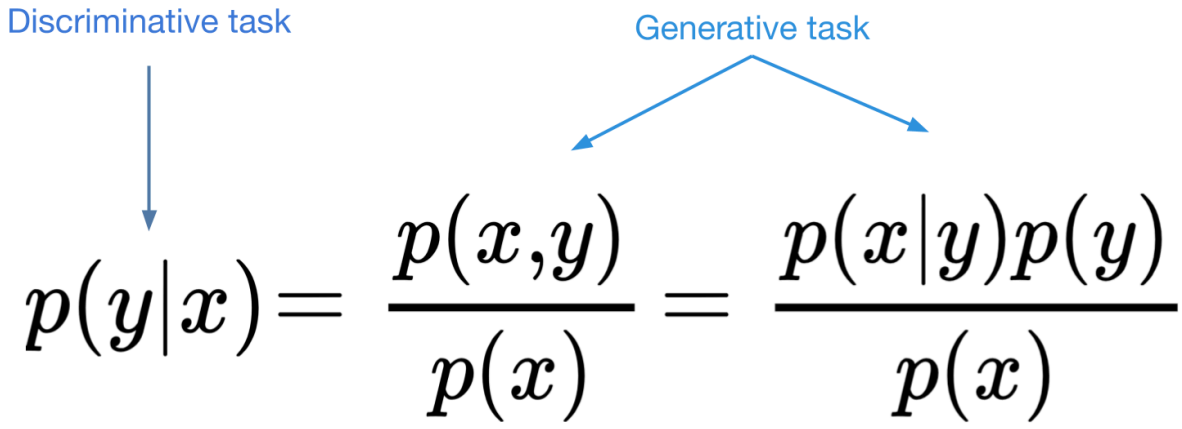
> **Discriminative Models.** These models are used for tasks like classification, and answer questions like "Given this input, what category does it belong to?" For example, imagine playing a song and then asking someone to identify its musical genre. This task is analogous to what discriminative models do in classifying data

based on observed patterns. Models that perform these classification tasks are typically referred to as classifiers.

Generative Models. These models synthesize new data. It's able to model the joint distribution of the data and its class label, and can answer questions like “Given a class, make a prediction for a sample that lies within that class.” An analogy for this would be asking someone to compose a song and then classify the genre of the song. Alternatively, we could give the genre and ask someone to compose or play a song within that genre.



It’s interesting to note the connection between discriminative and generative tasks, as shown by Bayes’ Theorem. This theorem relates the distributions used in each approach, meaning that, in some cases, a generative model can be adapted to perform discriminative tasks, however training a dedicated discriminative model is usually more effective. For this reason, we still treat discriminative and generative models as two distinct types.



AI leverages predictive models and automation to streamline and solve complex tasks that have traditionally required human expertise to accomplish. Machine Learning (ML) is a subset

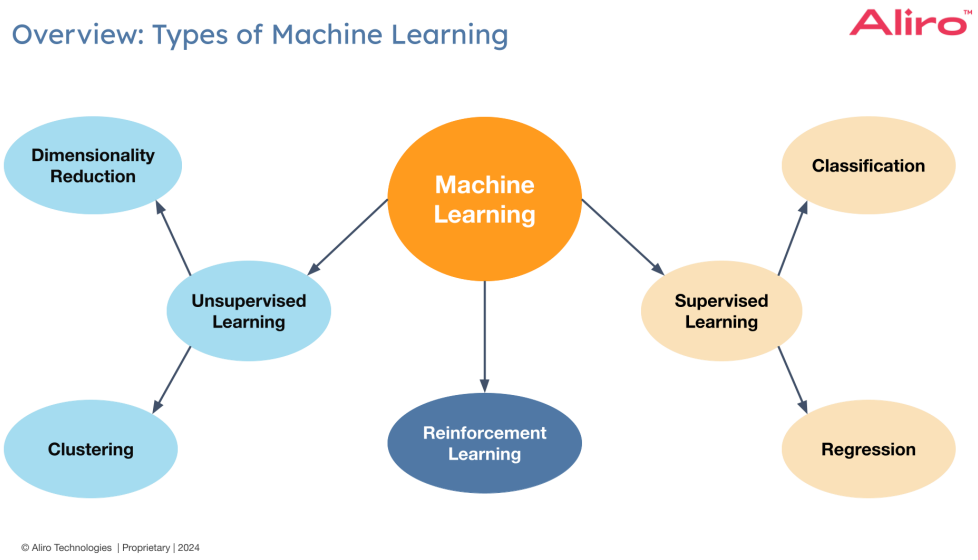
of AI that focuses on optimizing these predictions through adaptive learning. When implemented effectively, ML makes accurate predictions from learned patterns in data it's trained on.

Machine learning can be categorized into three types of statistical learning: supervised learning, unsupervised learning, and reinforcement learning. Each category provides distinct functionalities, and there are particular use cases for each type of learning.

In supervised learning, models are trained on data sets that have labeled data, which means for each data sample given there is a class category associated with that data sample in the training data. In essence, this method is giving the model the answer that a user wants it to find within the data set. Once trained, the model will be able to classify new data that's outside of the training data set, and even make predictions based on patterns learned in the training.

In unsupervised learning, the training data sets do not have labeled data. The goal of this type of training is for an AI model to be trained to uncover some patterns or statistical aspects of the data. Models trained in this way can perform tasks such as clustering, dimensionality reduction, and anomaly detection.

In reinforcement learning, an agent interacts with an environment to learn optimal behaviors. Over time, the agent learns which actions lead to the desired result. This method is commonly used in applications that require sequential decision-making, such as robotics, gaming, and autonomous navigation.



Specific Machine Learning Methods

The following sections provide specific examples of machine learning models and explain how each is applied within the AI landscape. These foundational machine learning models each have different strengths that make them valuable tools.

K-means Clustering: Uncovering Patterns in Unlabeled Data

K-means clustering is an unsupervised learning algorithm designed to identify and group data points into different sets, or clusters, without the use of labeled data. This method is helpful in exploring unstructured data and uncovering hidden groupings of data.

How K-means Clustering Works:

1. The algorithm begins with an assumed number of clusters, each represented by a centroid or center point in the feature space.
2. Each data point is assigned to a centroid based on proximity.
3. After assignment, each centroid is recalculated as the mean position of all data points in its cluster.

This iterative process continues until the clusters stabilize, meaning that data points no longer shift between clusters.

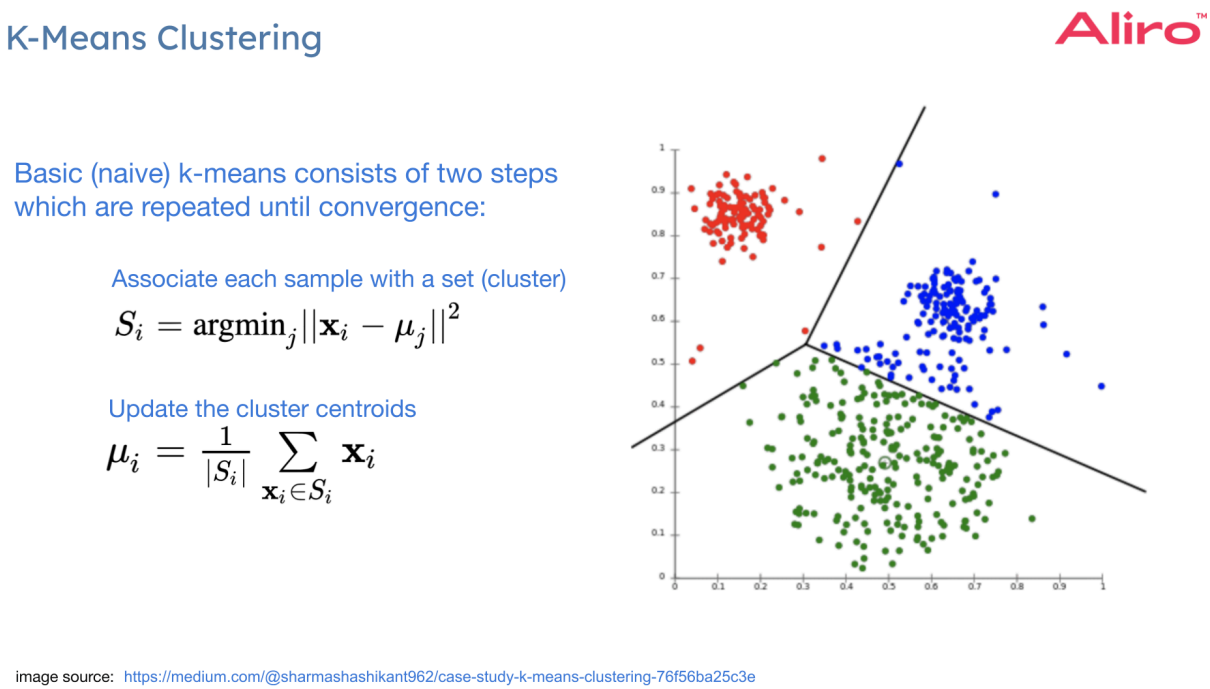


image source: https://medium.com/@sharmashashikant962/case-study-k-means-clustering-76f56ba25c3e

K-means clustering is widely used for tasks such as pattern recognition, customer segmentation, and detecting anomalies in data. Once clusters are formed, the model can be used to classify new data based on proximity to existing centroids, although it's primarily used

to understand the structure of the data rather than a strict classifier of data. As can be seen above, once the algorithm converges the data is partitioned into k different categories.

Kernel Methods: Separating Complex Data with Higher Dimensions

Kernel methods are used to distinguish data points that aren't easily separable in their original form. By mapping data to a higher-dimensional space, kernel methods create a more straightforward classification of complex datasets. Kernel functions perform two main operations: data transformation and choosing the kernel. Kernel functions, such as the radial basis function or polynomial kernel, transform data points into higher dimensions. This transformation creates new relationships between features, making it easier to separate out classes. Selecting the right kernel function (such as radial basis, polynomial, or sigmoid) is critical as this determines how the data is restructured and ultimately classified.

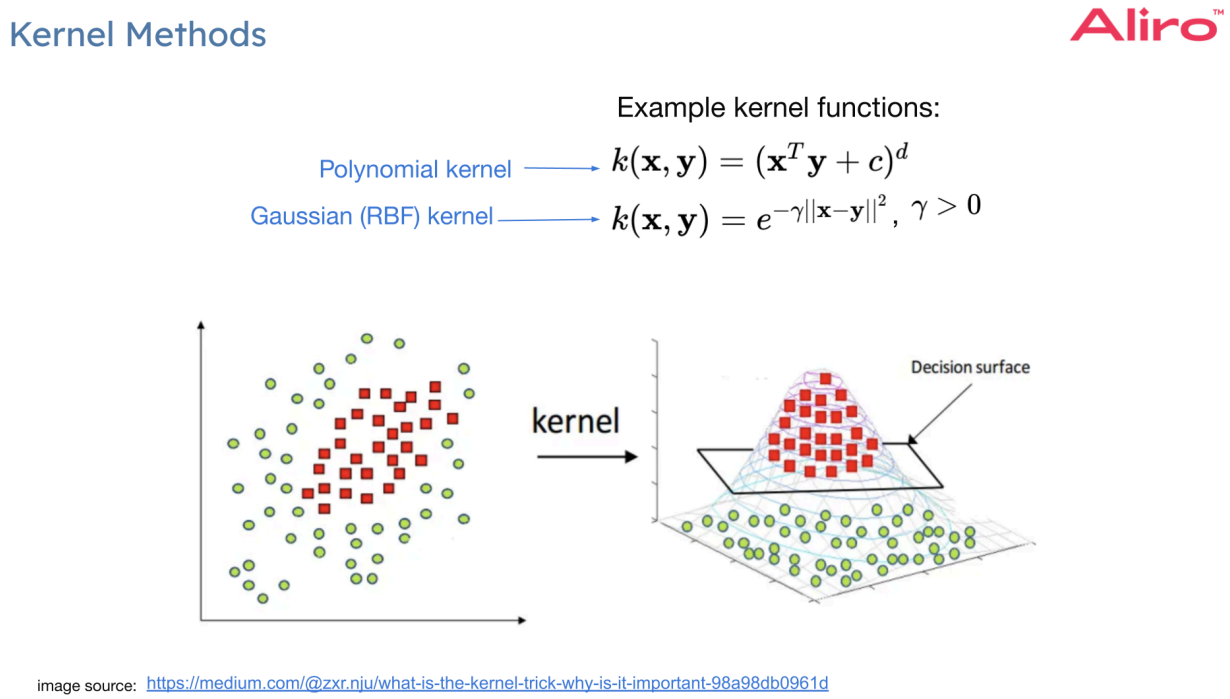


image source: https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d

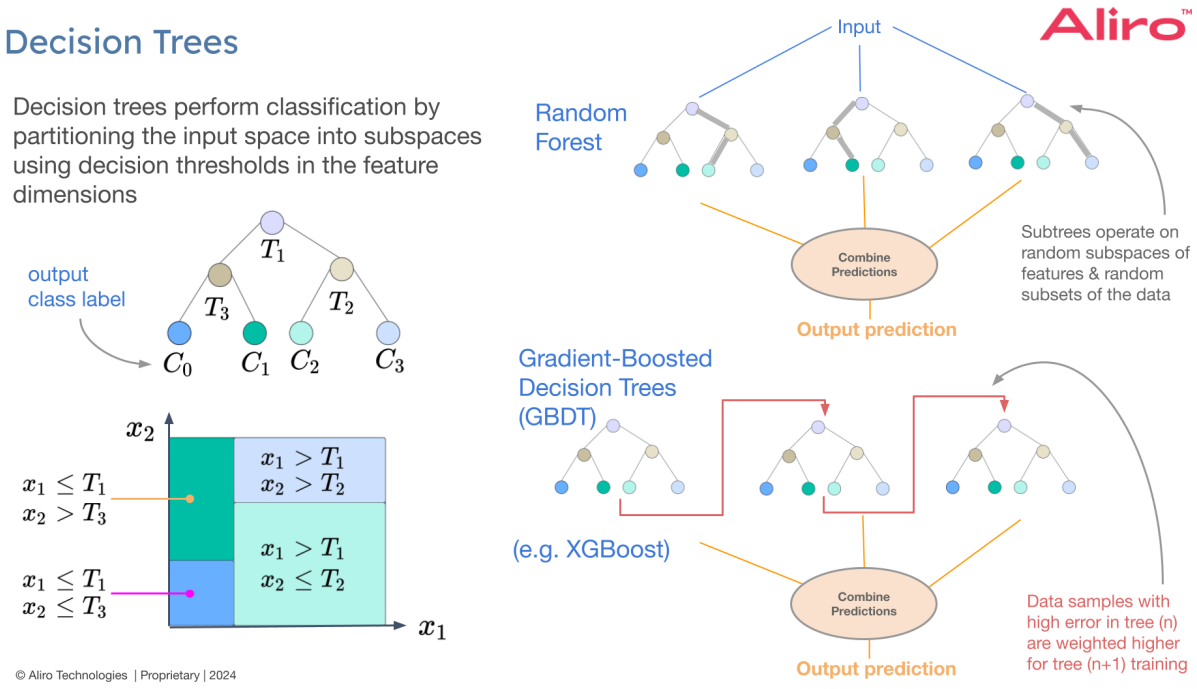
In the image above, the left part of the diagram shows some data samples, in two classes. Class One is indicated by the red samples and Class Two by the green samples. Because of the structure this data, there's no simple decision boundary: a straight line cannot be drawn to separate these two classes. By using a kernel function to map this data into higher dimensional space, in many cases a simple decision surface can be devised. The kernel mapping essentially made the data separable by mapping into this higher dimension. IN order for this method to be effective, the right kernel function needs to be used, but with the proper choice the classification is made much easier. One common application of kernel methods is in Support Vector Machines (SVMs).

Decision Trees: Classification and Regression

Decision trees are versatile models used for both classification and regression tasks. Decision trees start from a root node, splitting data into branches based on specific feature thresholds. Each split moves closer to a final classification. The model learns which thresholds minimize classification errors, making each branch more refined in categorizing data.

While decision trees are effective for many tasks, combining them yields even more sophisticated models. Two popular ensemble methods are:

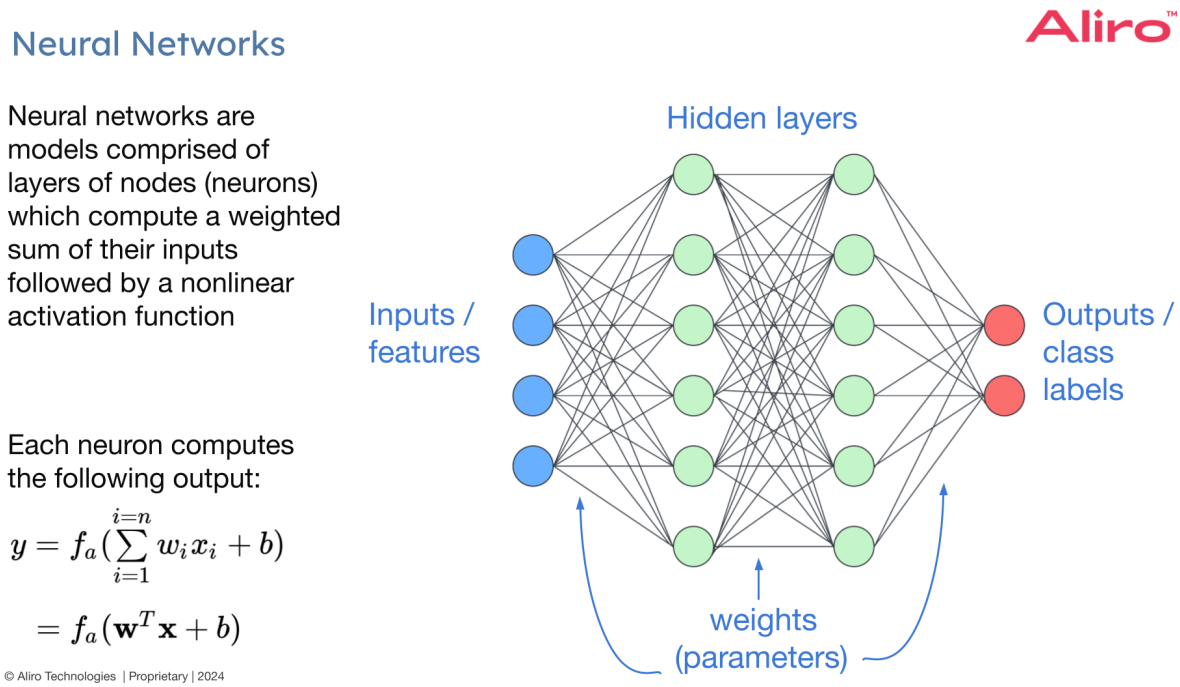
- Random Forests. Multiple decision trees are created using random subsets of data and features. Different sub-samplings of the data can then be applied to each tree, creating a strong learn model by combining the results of multiple weaker learning models.
- Gradient Boosted Trees. Decision trees are built sequentially, with each tree correcting the mistakes of the previous one. This approach focuses on reducing errors incrementally. The training data is weighted more heavily based on the errors that the previous tree made. Errors are minimized as trees become more sensitive to the errors that occurred previously. This is another way to combine decision trees into a stronger learning model.



Decision trees and their ensembles are widely used for financial modeling, risk assessment, and fraud detection.

Neural Architectures: Deep Learning with Layers of Neurons

Neural networks are among the most advanced machine learning architectures. They consist of interconnected computational nodes called neurons. Each neuron in a neural network takes a weighted sum of its input values, to which a bias term can be added. This weighted sum is passed through a nonlinear activation function that controls the range of the output. This allows the network to capture complex patterns in the data. As data flows through the network, it moves from layer to layer, ultimately reaching the output layer that gives a result. In a classification task, the neurons in the output layer produce signals. A stronger signal in one output neuron indicates a higher likelihood that the input data corresponds to one particular class over another.



Convolutional Neural Networks (CNNs) are particularly suited for tasks that involve spatial data. CNNs build on the basic neural network structure by placing constraints on the weights. This is called weight sharing, which in effect creates pattern matching filters that slide over the data. These filters are seeking out particular features in the data. The neurons in a CNN will create a strong signal when these features are detected. Where the activation is highest, these results are areas of interest - such as edges in an image. CNNs use max pooling, retaining only the highest activations / strongest signals in each region, to refine the output and discard less important details. In the final layers, referred to as fully connected layers, the network generates output neurons that correspond to the number of classification categories.

Convolutional Neural Networks (CNNs)

CNNs utilize weight-sharing to implement pattern-matching filters which operate over the data–widely used in computer vision

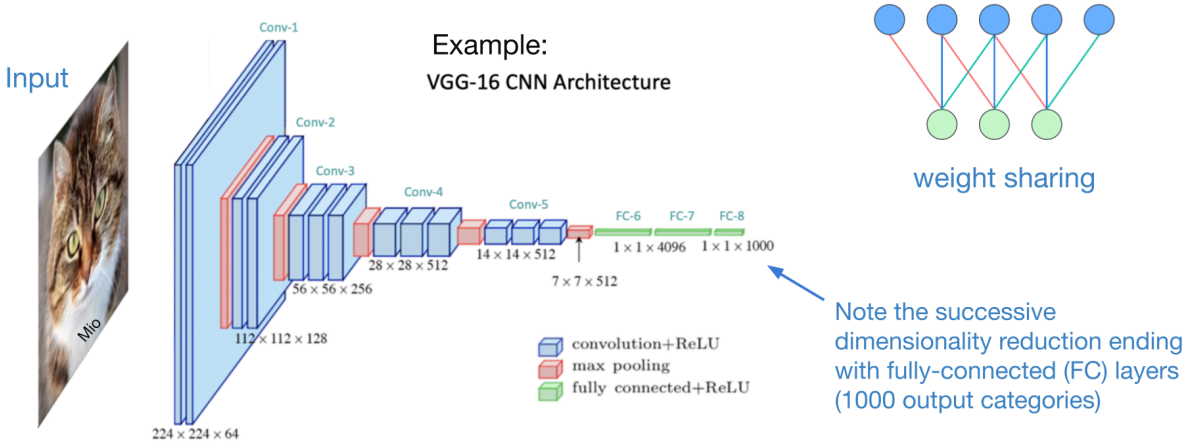


image source: https://learnopencv.com/understanding-convolutional-neural-networks-cnn/

Autoencoders are designed to learn the identity operation to produce an output that matches the input, but under a constraint that compresses the data into a lower dimensional hidden layer. The model identifies the key features of the data and then reconstructs that information. The reconstructed data can be adjusted to create different outputs. In the example below, a new image is created by making adjustments to the compressed representation.

Autoencoders

Autoencoders (AE) are models which attempt to learn the identity operation, subject to the constraint of having lower-dimensional hidden layers. This requires the model to learn an efficient "latent space" encoding of the data. There are multiple versions of AE, including convolutional autoencoders (CAE), and variational autoencoders (VAE).

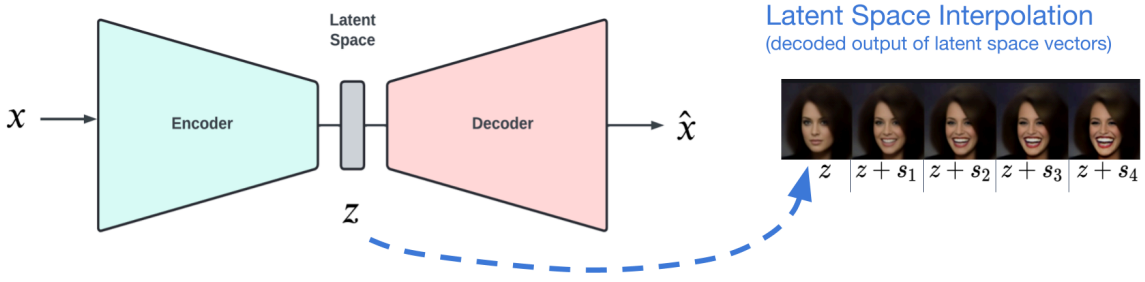


image source: https://www.bpesquet.fr/mlhandbook/algorithms/autoencoders.html

Data transformers are components used within neural architectures for learning useful representations of sequential data. This is particularly useful in Large Language Models. Data transformers employ something called multi-head self-attention mechanisms which is a matrix base way to capture relationships between different parts of a sequence. This is what enables models like ChatGPT to perform complex language tasks. Transformers are key in natural language processing but also have applications in other data types like images and video.

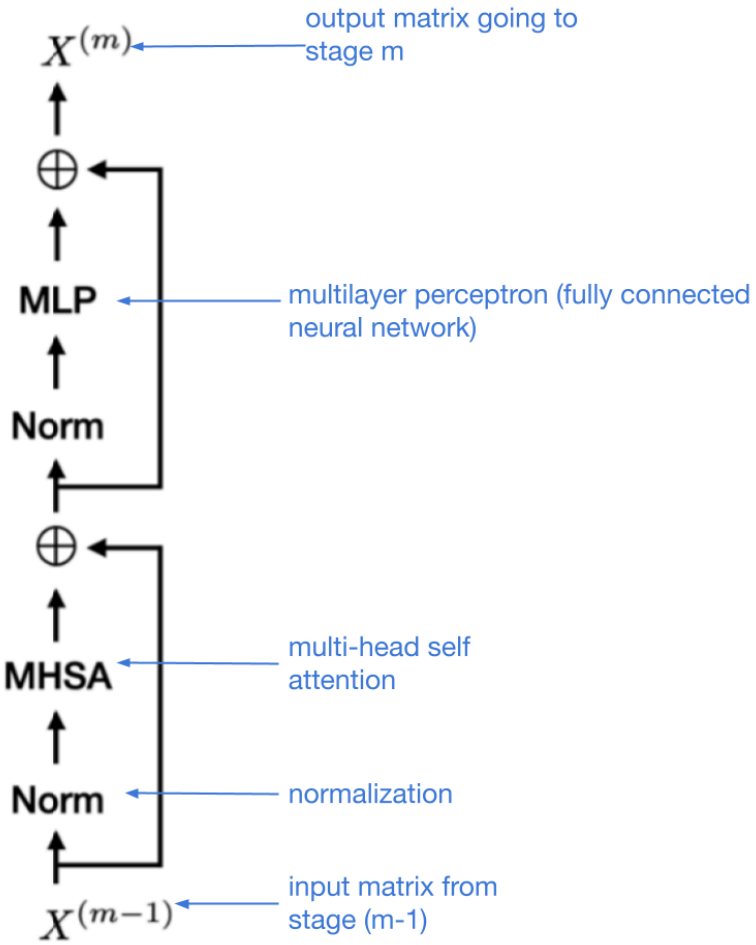
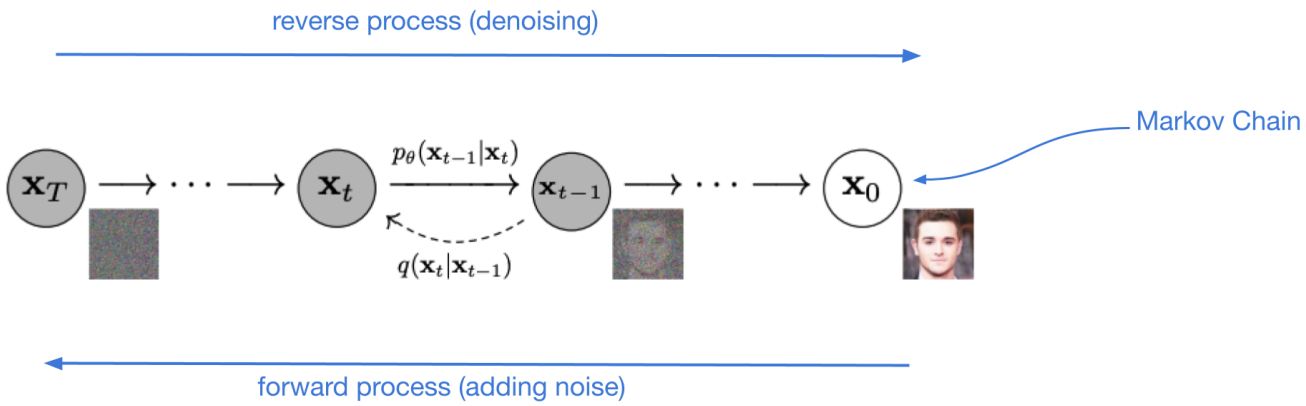


image source: https://arxiv.org/pdf/2304.10557

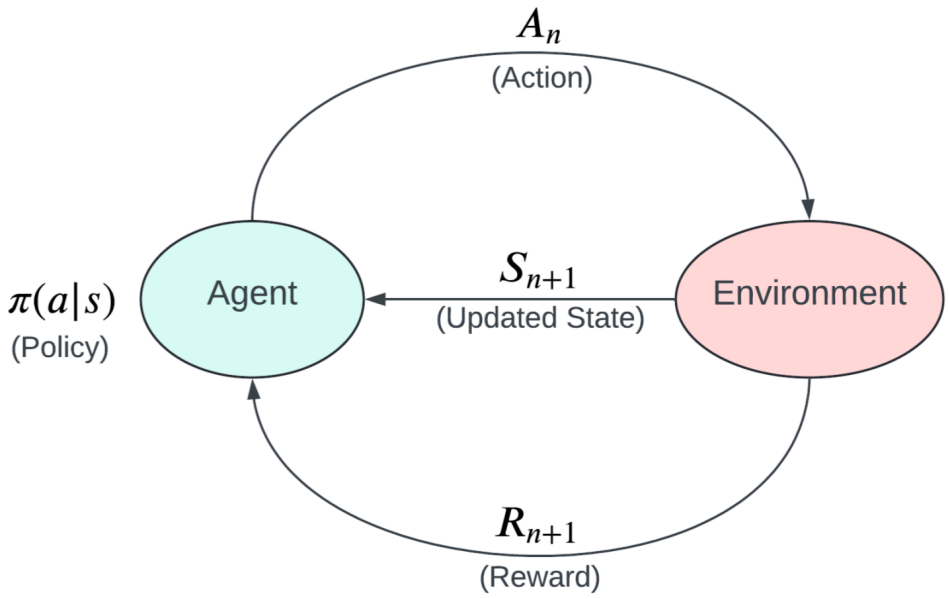
Diffusion models are inspired by thermodynamic processes, and are specifically designed for image synthesis. Diffusion models gradually learn to reverse decision process that allows them to create images. This process relies on a Markov chain and internal Gaussian distributions to refine the image through a series of small, sequential steps.

A well-known example of this approach is Stable Diffusion, which generates high-quality images from random noise.



image/reference source: https://arxiv.org/pdf/2006.11239

Reinforcement learning is a computational approach where an agent interacts with an environment to learn optimal behaviors. The agent takes actions, and the environment responds with an updated state and a reward or penalty. Over time, the agent learns a policy—a strategy for choosing actions—that maximizes its cumulative reward.



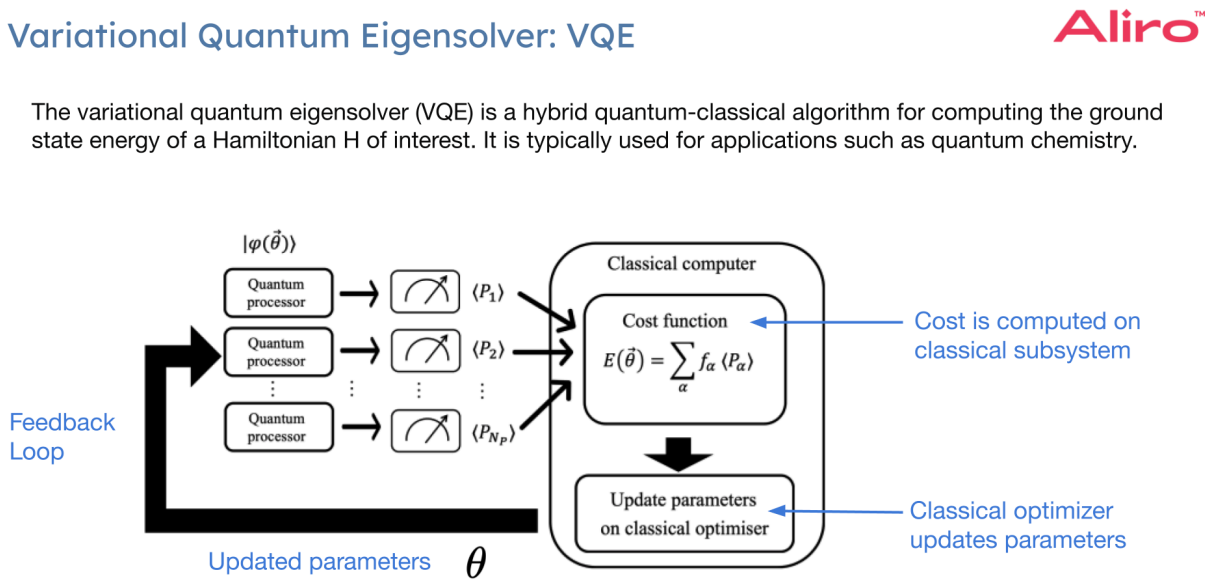
reference: https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf

Variational Quantum Algorithms

Variational quantum algorithms are a class of algorithms that combine quantum and classical computing capabilities. Here, we'll explore two key types of variational quantum algorithms: the Variational Quantum Eigensolver (VQE) and the Quantum Approximate Optimization Algorithm (QAOA). Each uses a hybrid approach that involves feedback between quantum and classical systems to refine and improve results.

The Variational Quantum Eigensolver (VQE)

The Variational Quantum Eigensolver (VQE) is a hybrid quantum-classical approach that finds the ground-state energy of complex systems, such as molecular structures and for this reason it is very useful in quantum chemistry. In the diagram below, the quantum system performs computations based on a set of parameters denoted as θ (theta). Once the quantum computation is executed, measurements are taken. The classical optimizer then takes the results of these measurements and determines how well this sampling matched the cost optimization criteria. Based on these findings, it adjusts the parameters. This process repeats and over time the quantum system converges toward a solution that closely approximates the ground state energy.

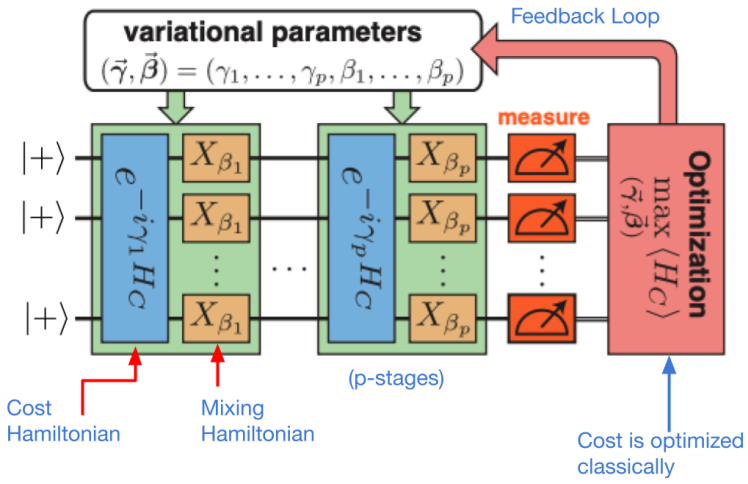


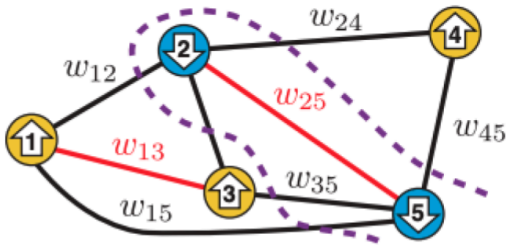
Reference: Hybrid quantum-classical algorithms and quantum error mitigation

Quantum Approximate Optimization Algorithm (QAOA)

Quantum Approximate Optimization Algorithm (QAOA) is primarily applied in discrete optimization problems, such as logistical planning or graph-based problems. One application of QAOA is the Max-Cut Problem, a type of graph problem where the goal is to divide nodes into two groups while also maximizing the number of edges between the groups.

In QAOA, the optimization problem is encoded into two parts within the quantum computation: a Cost Hamiltonian and a Mixing Hamiltonian. The Cost Hamiltonian represents the problem to be solved by encoding the specific optimization task. The Mixing Hamiltonian allows the algorithm to explore a range of possible solutions. The quantum processor samples solutions from a probability distribution, allowing it to identify potential solutions that have a high likelihood of meeting the optimization criteria. A classical optimizer then assesses the quality of the solution and feeds this information back into the quantum system, refining the solution with each iteration.





Example MaxCut Problem

Assign spin variables on the vertices such that the sum of edge weights between anti-aligned spins is maximized (black edges)

Reference: https://arxiv.org/pdf/1812.01041

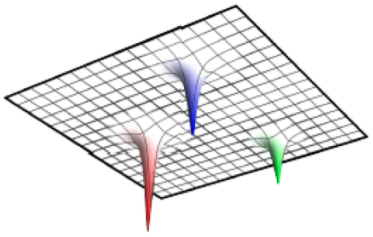
Barren Plateaus

One common issue in variational quantum algorithms like VQE and QAOA is barren plateaus. Barren plateaus occur when the gradients of the cost function become extremely small. This essentially flattens the optimization landscape. The optimizer can't determine which direction to adjust the parameters, as every possible path appears to yield the same minimal gradient values. This makes the optimization process slow and ineffective. Researchers in both classical machine learning and quantum machine learning have sought to mitigate barren plateaus, and research has shown that adjusting the choice of topologies, initial states, and activation functions in the network can help eliminate barren plateaus.

Optimization Challenge: Barren Plateaus

Aliro™

Variational optimization shares a challenge with classical ML due to the use of classical optimizers. Optimizing a cost function with respect to a model weight involves the chain rule, leading to products of factors. This can lead to a problem called barren plateaus, making it difficult to determine search direction.



$$\frac{\partial C}{\partial w_{jk}^{l}} = \sum_{mnp\ldots q} \frac{\partial C}{\partial a_{m}^{L}} \frac{\partial a_{m}^{L}}{\partial a_{n}^{L-1}} \frac{\partial a_{n}^{L-1}}{\partial a_{p}^{L-2}} \cdots \frac{\partial a_{q}^{l+1}}{\partial a_{j}^{l}} \frac{\partial a_{j}^{l}}{\partial w_{jk}^{l}}$$

All paths from initial weight to final Cost

Long products lead to very small gradients

The rate-of-change of the Cost (C) with respect to a specific weight (w) in a neural network or variational model

Mitigation:

- Initial state choice
- Limit circuit depth or use skip connections
- Choice of activation functions (a)
- Choice of ansatz (topology)

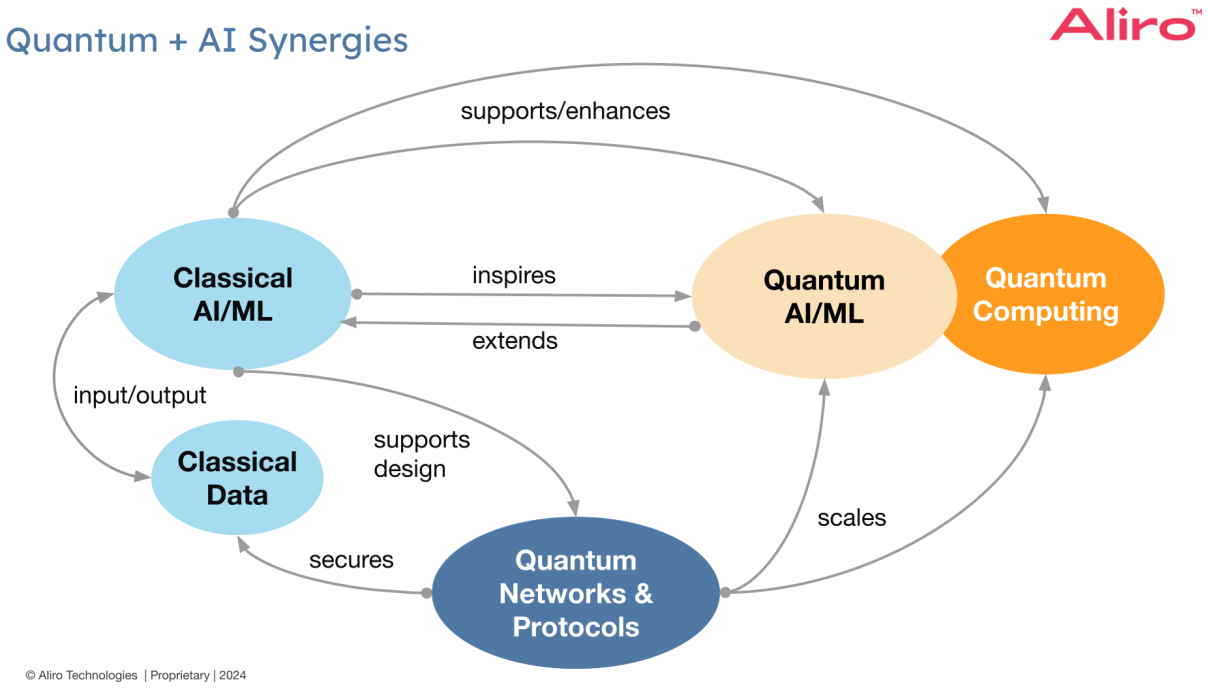
Classical vs Quantum:

Both vanishing and exploding gradients are possible with classical ML but with QML the bounded nature of unitary operations tends to prevent exploding gradients. QML achieves nonlinearity through entanglement or measurements.

Image source: https://arxiv.org/pdf/2405.05332

Synergies Between Classical AI and Quantum Technologies

The intersection of classical and quantum methodologies provides a multitude of beneficial synergies.



Classical AI Inspiring Quantum Machine Learning

A significant synergy exists in how classical machine learning (ML) methods inspire quantum machine learning. For many classical ML algorithms, researchers have developed quantum versions. Some examples of these adaptations include quantum neural networks, quantum convolutional neural neural networks, quantum auto encoders, quantum support vector machines, quantum diffusion models and so forth. These quantum adaptations allow classical algorithms to run directly on quantum data, or the quantum versions improve the performance of the algorithm.

Scaling Quantum Computing Through Quantum Networking

Building a large, monolithic quantum computer is extremely challenging due to technical limitations. Many believe that the future of scalable quantum computing lies in quantum networking. Through quantum networking, smaller quantum processors can be interconnected to form larger, distributed systems.

This networking approach is instrumental in scaling quantum computing, as it allows for powerful distributed quantum computing setups, which in turn benefit quantum machine learning by providing greater computational resources.

Enhancing Quantum Computing with Classical AI

One of the central challenges in quantum computing is dealing with noise, which can interfere with accurate calculations. Classical AI offers solutions for managing and mitigating this noise through quantum error correction and quantum error mitigation techniques.

Quantum error correction involves adding redundant qubits to detect and correct errors in calculations. Classical machine learning models assist by interpreting complex error syndromes (non-destructive measurements that detect errors) to determine the corrective actions required for data integrity. Post-processing methods in quantum error mitigation can enhance the results of quantum computing.For example, methods exist for using ML to generate a prediction of the ideal noiseless result of a quantum computation derived from the noisy measured output values. By using classical AI for error correction and mitigation, quantum systems become more resilient, leading to more accurate results, particularly in quantum machine learning applications.

Designing Quantum Networks with Classical ML

Designing efficient quantum networks is complex, involving multiple parameters, degrees of freedom, and a wide variety of algorithms. Classical ML, especially reinforcement learning, aids in optimizing network configurations and protocols for certain networking tasks, such as how long to hold an entanglement before regenerating it. This enhances the stability and efficiency of quantum networks. Some reinforcement learning systems have also autonomously discovered new protocols for entanglement distribution, even outperforming manually designed methods in specialized cases such as where two noise channels have different kinds of noise on them

Privacy Enhancements in AI through Quantum Networking

Data privacy remains a critical area in AI, as machine learning models trained on data can sometimes reveal sensitive information. In some cases, specific information can be inferred from the model parameters themselves. Quantum networks enable the secure transmission of data with information-theoretic security, which is theoretically immune to interception. In addition to protecting data in transit, blind quantum computing architectures that are enabled by quantum networks can facilitate even greater privacy protections. In blind quantum computing, users can execute quantum algorithms with quantum data in such a way that the quantum server cannot extract any information about the data or the algorithm.

These advancements enable privacy-preserving AI, allowing sensitive data to be analyzed without risk of exposure—a valuable breakthrough for sectors requiring high confidentiality, such as finance and healthcare.

Future Directions and Security Implications

The convergence of AI, quantum computing, and advanced secure networks marks a transformative shift in computational and security capabilities. These technologies individually provide enormous benefits, but together they provide even more opportunities for advancement.

Entanglement-based quantum networks are being built today by a variety of organizations for a variety of use cases – benefiting organizations internally, as well as providing great value to an organization's customers. Telecommunications companies, national research labs, and systems integrators are just a few examples of the organizations Aliro is helping to leverage the capabilities of quantum secure communications.

Building entanglement-based quantum networks that use entanglement is no easy task. It requires:

- Emerging hardware components necessary to build the network.
- The software necessary to design, simulate, run, and manage the network.
- A team with expertise in the fundamental science of entanglement-based quantum networks and classical networking.
- Years of hard work and development.

This may seem overwhelming, but Aliro is uniquely positioned to help you build your quantum network. The steps you can take to ensure your organization is meeting the challenges and leveraging the benefits of the quantum revolution are part of a clear, unified solution already at work in networks like the EPB Quantum Network℠ powered by Qubitekk in Chattanooga, Tennessee.

AliroNet™, the world's first full-stack entanglement-based network solution, consists of the software and services necessary to ensure customers will fully meet their advanced secure networking goals. Each component within AliroNet™ is built from the ground up to be compatible and optimal with entanglement-based networks of any scale and architecture. AliroNet™ is used to simulate, design, run, and manage quantum networks as well as test, verify, and optimize quantum hardware for network performance. AliroNet™ leverages the expertise of Aliro personnel in order to ensure that customers get the most value out of the software and their investment.

Depending on where customers are in their quantum networking journeys, AliroNet™ is available in three modes that create a clear path toward building full-scale entanglement-based secure networks: (1) Emulation Mode, for emulating, designing, and validating entanglement-based quantum networks, (2) Pilot Mode for implementing a

small-scale entanglement-based quantum network testbed, and (3) Deployment Mode for scaling entanglement-based quantum networks and integrating end-to-end applications. AliroNet™ has been developed by a team of world-class experts.

To get started on your Quantum Networking journey, reach out to the Aliro team for additional information on how AliroNet™ can enable secure communications.

www.alirotech.com

References

Piech, Chris, and Andrew Ng. "K-Means." Stanford University, 2013, web.stanford.edu/class/cs221/handouts/kmeans.html. Accessed 2024.

Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models." arXiv, version 2, 16 Dec. 2020, https://doi.org/10.48550/arXiv.2006.11239.

Sutton, Richard S., and Andrew G. Barto. Reinforcement Learning: An Introduction. 2nd ed., in progress, 2014-2015.

Endo, Suguru, Zhenyu Cai, Simon C. Benjamin, and Xiao Yuan. "Hybrid Quantum-Classical Algorithms and Quantum Error Mitigation." Journal of the Physical Society of Japan, vol. 90, no. 3, 2021, Article ID: 032001, https://doi.org/10.7566/JPSJ.90.032001. arXiv, https://doi.org/10.48550/arXiv.2011.01382.

Zhou, Leo, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D. Lukin. "Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices." Physical Review X, vol. 10, no. 2, 2020, Article ID: 021067, https://doi.org/10.1103/PhysRevX.10.021067. arXiv, https://doi.org/10.48550/arXiv.1812.01041.

Nielsen, Michael A. Neural Networks and Deep Learning. Determination Press, 2015.

Choukroun, Yoni, and Lior Wolf. "Deep Quantum Error Correction." arXiv, version 2, 10 Dec. 2023, https://doi.org/10.48550/arXiv.2301.11930.

Liao, Haoran, Derek S. Wang, Iskandar Sitdikov, Ciro Salcedo, Alireza Seif, and Zlatko K. Minev. "Machine Learning for Practical Quantum Error Mitigation." arXiv, 29 Sept. 2023, https://doi.org/10.48550/arXiv.2309.17368.

Cantori, Simone, Andrea Mari, David Vitali, and Sebastiano Pilati. "Synergy Between Noisy Quantum Computers and Scalable Classical Deep Learning." EPJ Quantum Technology, vol. 11, no. 45, 2024, https://doi.org/10.1140/epjqt/s40507-024-00256-8. arXiv, https://doi.org/10.48550/arXiv.2404.07802.

Image references are provided directly under the image.