INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelorarbeit

Quantum-klassisches Machine Learning am Leibniz-Rechenzentrum

Daniel Seidl

Draft vom 26. September 2022

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelorarbeit

Quantum-klassisches Machine Learning am Leibniz-Rechenzentrum

Daniel Seidl

Aufgabensteller:	Prof. Dr. Dieter Kranzlmüller
Betreuer:	Tobias Guggemos Maximilian Höb
Abgabetermin:	12. Oktober 2022

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 26. September 2022

(Unterschrift des Kandidaten)

Abstract

In Zeiten von Big Data werden immer größere Datenmengen generiert, die Analyse und Auswertung selbiger stellt die Wissenschaft vor große Hindernisse: Die massiv anwachsenden Anforderungen an die Rechenleistung schränken die Effektivität von Berechnungen erheblich ein und bilden einen Engpass. Verschiedene Machine Learning (ML) Algorithmen werden eingesetzt, um die Datenmengen zu verarbeiten und aus diesen Rückschlüsse auf zukünftige Ereignisse ziehen zu können. Weiterentwicklungen der Hardware von klassischen Computern, genauer die Transistordichte in Prozessoreinheiten, erreichen nicht mehr die durch Moores Law prognostizierte Verdopplung der Leistung alle zwei Jahre. Durch die Vorteile des Quantum Computings - wie beispielsweise die durch dieses Berechnungsparadigma erreichte echte Parallelität bei Berechnungen - besteht die Möglichkeit, diese Schwierigkeiten überwinden zu können. Derzeit befinden wir uns in der Noisy Intermediate Scale Quantum (NISQ) - Ära. Durch das Rauschen in den von Quanten Computern verwendeten Quanten Gattern ist die Anzahl von zuverlässig nutzbaren Quantenbits derzeit noch beschränkt. Daher bietet es sich vorerst an, übergangsweise ein hybrides System aus klassischem Computer und Quantencomputer zu implementieren. In dieser Arbeit wird untersucht, wie eine Schnittstelle zwischen einem Linux Rechencluster am Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften (LRZ) und der Quantum Learning Machine des französischen Herstellers Atos (QLM) implementiert werden kann. Durch eine Schnittstellennutzung über Protokolle wie Secure Shell und Secure Copy, sowie der Nutzung eines Schedulers bzw. Workloadmanagers wie SLURM ist es möglich, eine Verbindung zwischen klassischem Computer und Quantencomputer herzustellen. Über diese Verbindung lassen sich automatisiert Arbeitspakete, Anweisungen und die Ergebnisse von Berechnungen austauschen, die für jeweilige weitere Arbeitsschritte verwendet werden können. Das daraus entstehende quantum-klassisch hybride System soll in Zukunft unter anderem für hybride Machine Learning Algorithmen eingesetzt werden. Das System wurde durch die automatisierte Ausführung von verschiedenen Quantenalgorithmen auf der QLM auf die Richtigkeit und Nutzbarkeit getestet. Die Resultate der Berechnungen sind korrekt und können von klassischen Algorithmen weiterverarbeitet werden. Mit der zukünftigen Entwicklung und Implementation von Convolutional Neural Networks (CNN) können effektiv Anwendungsfälle wie Bildklassifizierung oder Objekterkennung etc. am hybriden System des LRZ durchgeführt werden.

Inhaltsverzeichnis

1 Einleitung

Noch vor 40 Jahren wurden Daten analog abgelegt, große Papierstapel bildeten die Informationsgrundlagen für die verschiedensten Unterfangen, ob von Unternehmen oder Privatpersonen. Nur ein Prozent der Daten liegt in digitaler Form vor. Ein Sprung ins Jahr 2017 zeigt: Jeder Mensch erzeugt durchschnittlich über 600 Megabyte (MB) Daten pro Tag. [Wir] Hochgerechnet ergibt sich daraus eine schier unvorstellbare Menge an Daten, die über die verschiedensten Medien wie beispielsweise Messenger Dienste und soziale Medien generiert werden. Nur noch ein Prozent der Daten liegt nun analog vor, ganze 99 Prozent sind digital. [Wir] Tagtäglich werden ca. 8,5 Milliarden Suchanfragen auf Google erstellt, um auf diese zuzugreifen. [goo] Diese immer rasanter wachsenden Datenmengen helfen dabei, über Softwarelösungen gezielte Vorhersagen zu treffen und optimale Rückschlüsse basierend auf bestehenden Informationen ziehen zu können. Doch längst handelt es sich nicht mehr nur um Big Data - der Begriff Bigger Data wirkt fast schon passender.

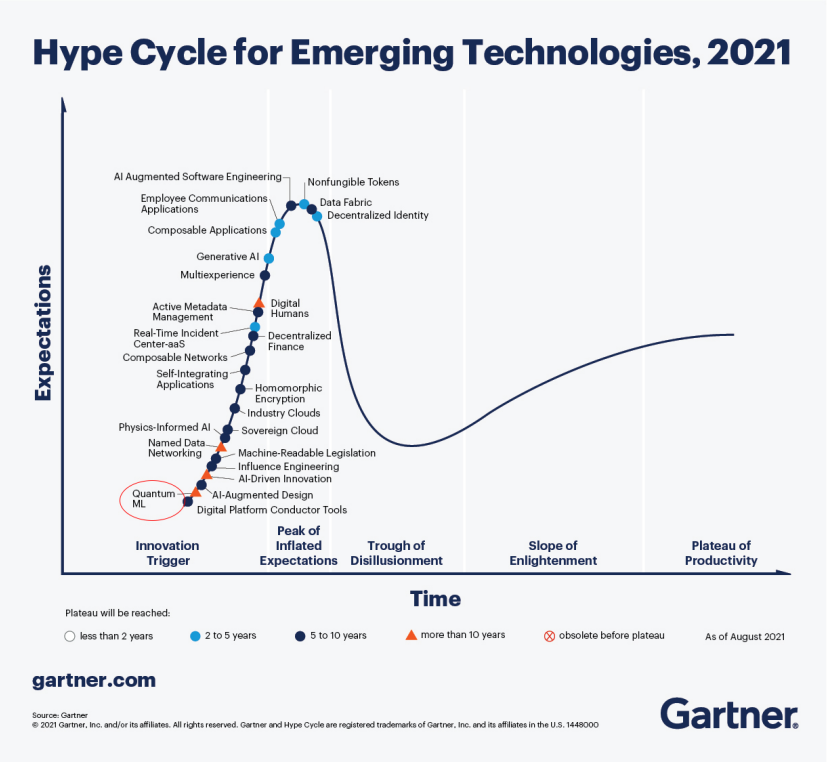

Abbildung 1.1: Gartner Hype Cycle 2021 für aufkommende Technologien, mit Quantum Machine Learning als Innovationstrigger für die kommenden Jahre [Gar]

Um diese Flut an Daten zu bewältigen, sind schon seit einigen Jahren Machine Lear-

ning (ML) - besonders Deep Learning - Algorithmen im Einsatz, um automatisiert konkrete Aussagen zu verschiedenen Datensätzen treffen zu können. Vor allem CNN haben stark an Bedeutung gewonnen und sind essenziell für verschiedenste Anwendungen, wie beispielsweise die Bilderkennung. Jedoch stoßen konventionelle, klassische Systeme, die aufwändige ML Algorithmen und (Künstliche) Neuronale Netze (NN), wie die angesprochenen CNN, implementieren, durch die hohe Anforderung an Rechenleistung an ihre Grenzen, die Laufzeit der Trainingsalgorithmen soll wünschenswerter Weise natürlich so kurz wie möglich sein. Um diese wachsenden Datenmengen zu verarbeiten, wird auch steigende Rechenleistung benötigt. Intensive Forschung treiben - unter anderem gezwungen durch diesen Flaschenhals, aber auch durch die bisher unlösbaren Probleme wie Zeiteffizienz und immer größer werdender Bedarf an Speicher - besonders das Gebiet des Quantum Computings [Fey82] voran. Dieses ist ein relativ neues Berechnungsparadigma, mit vielen Anwendungsbereichen wie beispielsweise Simulationen oder dem ML, genannt Quantum Machine Learning, was als einer der Innovationstrigger der kommenden Jahre gilt (siehe Abbildung 1.1). Durch die besonderen Eigenschaften der Quanten Mechanik, wie die Superposition und die Verschränkung ist es möglich, eine echte Parallelität beim Berechnen zu ermöglichen, welches für manche Probleme exponentielle Geschwindigkeitsvorteile verspricht.

1.1 Problemstellung

Das in München, Garching lokalisierte Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften (LRZ) [LRZc] betreibt das Münchner Wissenschaftsnetz (MWN), an welches die Technische Universität München, die Ludwig Maximilians Universität München und weitere Münchner Hochschulen angeschlossen sind. [MWN] Neben dem Hochschulnetz bietet das LRZ auch Berechnungsdienste an. Diese umfassen Zugriffe auf Linux Cluster, High Performance Cloud Systeme sowie Höchstleistungsrechner, wie den SuperMUC-NG. [HPCa] Mit diesen Supercomputern wird Wissenschaflter*innen im ganzen Land die Basis für Spitzenforschung geboten [HPCb]. Mit der Zusammenarbeit mit dem IT-Dienstleister Atos ist nun die Integration von Quanten Computern in der eigenen High-Performance-Computing Umgebung und im eigenen Netz möglich. [atoe]

1.2 Fragestellung

Die angesprochene benötigte hohe Rechenleistung spielt unter anderem für die Erdbeobachtung (englisch: Remote Sensing) im Bereich der Erdobservations-Technologien eine große Rolle, bei welcher etwa durch Kameras auf Satelliten oder durch Sonar Systeme auf Schiffen hochauflösende Aufzeichnungen der Erdoberfläche und des Meeresgrundes festgehalten werden können (siehe Abbildung 1.2). Im Anschluss werden diese Daten für verschiedenste Zwecke verwendet, so können Waldbrände aus dem All aus beobachtet werden und somit einen weitaus besseren Überblick über die Lage für Ranger schaffen. Wolkenbewegungen können nachverfolgt werden, um das Wetter vorherzusagen, das Wachstum einer Stadt und die Veränderungen von Farmland und Wäldern können über einen großen Zeitraum hinweg analysiert werden. [USG][GIS]

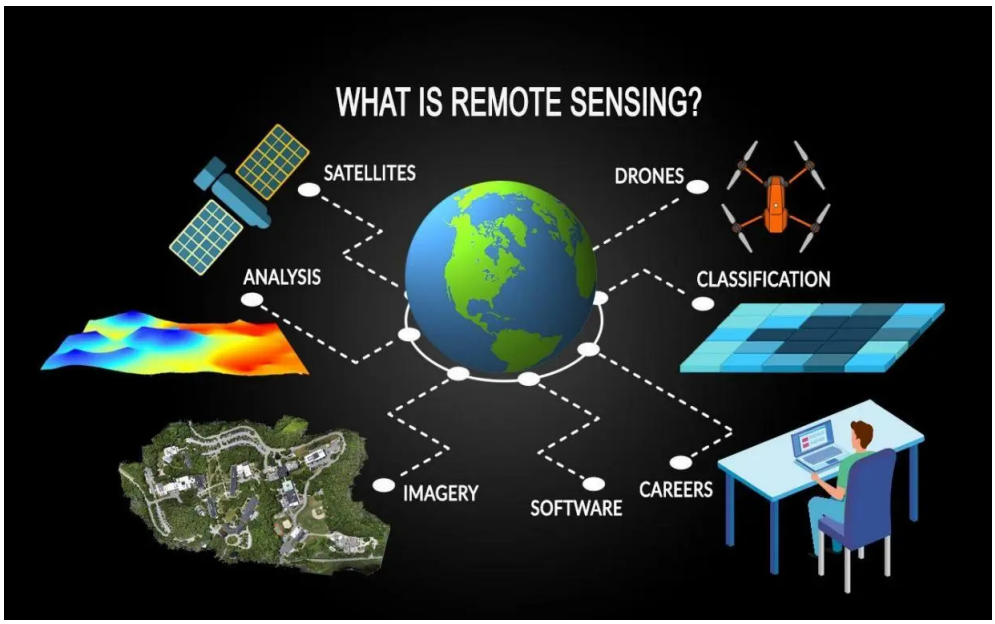

Abbildung 1.2: Einsatzgebiete von Remote Sensing [GIS]

Diese Anwendungsfälle stellen nur einen Bruchteil dessen dar, in welchen Bereichen groß angelegte Bilderkennung nützlich eingesetzt wird. Daher stellt sich die dringende Frage: Können wir schon jetzt durch ein hybrides System die Vorteile des Quantum Computings nutzen und möglicherweise eine exponentielle Beschleunigung verschiedener Arbeitsschritte verwenden? Damit gilt zu klären, wie die verschiedenen Komponenten eines möglichen Systems aussehen und wie die Datenübertragung zwischen diesen gehandhabt werden kann. Konkret auf die gegebene Infrastruktur bezogen lautet die Frage nun: Wie kann die bestehende Landschaft des LRZ mit der von Atos bereitgestellten QLM verbessert werden? Wie sieht ein quantenbeschleunigtes hybrides System in dieser Umgebung aus? Die Antwort darauf stellt den Kern der Arbeit dar und soll auf den folgenden Seiten ausgearbeitet werden.

1.3 Ziel der Arbeit

Ziel der Arbeit ist der Entwurf einer Schnittstelle zwischen dem Linux Rechencluster am LRZ und dem ersten kommerziellen Quanten Simulator, der QLM von Atos. [ATOc] Auf diese Weise soll ein quantum-klassisches Hybrid ermöglicht werden, wodurch in Zukunft verschiedene geeignete, rechenintensive Schritte eines Algorithmus auf den Quanten Teil des Systems ausgelagert werden sollen, während der klassische Teil des Systems die weniger aufwändigen Schritte übernehmen kann. Die Schnittstelle soll es möglich machen, Quanten Algorithmen im großen Stil auf einem Verbund von Hochleistungsrechner und Quanten Computer laufen zu lassen. Im nächsten Abschnitt 1.4 wird ein grober Überblick über die einzelnen Teile der Arbeit präsentiert.

1.4 Aufbau der Arbeit

Der Rest dieser Arbeit ist wie folgt strukturiert: In Kapitel 2 werden wichtige Grundlagen hinsichtlich Quanten Computing, ML und dem System, das beide Konzepte miteinander verknüpft, geklärt, um das Ziel und die Wichtigkeit eines hybriden Systems verständlich zu machen. Verwandte Arbeiten werden im Unterkapitel 2.4 vorgestellt. Anschließend wird in Kapitel 3 das Konzept zur Implementierung eines quantum-klassischen Hybrids erläutert und folglich in Kapitel 4 auf einem System bestehend aus einem Linux-Rechencluster des LRZ und der QLM von Atos eingesetzt. Das System wird in Kapitel 5 anhand der Implementation von verschiedenen Algorithmen evaluiert. Die damit zu treffenden Aussagen und der weitere Ausblick finden sich am Schluss in Kapitel 6.

2 Grundlagen und verwandte Arbeiten

Im folgenden Kapitel werden die für den weiteren Verlauf benötigten Grundlagen der Arbeit gelegt. Insbesondere werden die wichtigsten Begriffe aus dem Quantum Computing oder auch Quantencomputing (QC) erläutert und Konzepte des ML erklärt. Die Kombination der beiden Forschungsbereiche, das sogenannte Quantum Machine Learning (QML), wird anschließend erklärt, da es die Grundlage des Anwendungsfalles darstellt.

2.1 Quantencomputing Grundlagen

Das Berechnungsmodell des QC basiert auf der physikalischen Theorie der Quantenmechanik, mit welcher man versucht, das Verhalten der Materie im atomaren und subatomaren Bereich zu beschreiben. Die Konzepte wurden von bedeutenden Physikern wie beispielsweise Werner Heisenberg, Erwin Schrödinger, Paul Dirac und auch John von Neumann - welchem wir die in heutigen klassischen Computern verwendete Von-Neumann-Architektur zu verdanken haben - ausgearbeitet. [cheb] In dieser Arbeit werden die Grundlagen bezüglich der Quantenmechanik nur sehr oberflächlich behandelt.

QC unterscheidet sich fundamental zu klassischen Berechnungen. Mit Quantencomputern ist es möglich, Probleme auf eine andere Art und Weise zu lösen, als es mit klassischen Computern der Fall ist. Bei bestimmten Problemen hat ein Quantencomputer gegenüber einem klassischen Computer erhebliche Vorteile. Um dieses Verhalten zu verstehen, müssen wir uns mit den wichtigsten Konzepten der Quanten Mechanik vertraut machen. Diese bestehen vor allem aus der Superposition, der Verschränkung und der Interferenz. [NC16] Wichtige Literatur für ein besseres Verständnis von Quantum Computing bietet "Quantum Computing verstehen" von Matthias Homeister [Hom05] und "Quantum Computation and Quantum Information" von Isaac Chuang und Michael Nielsen. [NC16].

2.1.1 Quantenbit

Allgemein ist das Bit die atomare Einheit in der Informationstechnologie. Das System kann sich zu jedem gegebenen Zeitpunkt nur in einem Zustand zur gleichen Zeit befinden, so befindet sich ein System mit einem Bit entweder im Zustand 0 oder im Zustand 1. [NC16] Analog bildet auch das Qubit, das Bit in der Quantum Informationswelt, die Grundeinheit des gesamten Systems. Reell umgesetzt werden können diese Qubits in Form von photonischer Polarisation, Spin Zuständen oder etwa Ladungszustände in supraleitenden Systemen. [LZX+20] In der Quantenmechanik ist es üblich, Zustände, in denen sich ein Element befindet, in Klammern der Form $|\phi\rangle$ zu setzen. Folglich existieren für die Qubits die Zustände $|0\rangle$ und $|1\rangle$. Man bezeichnet dies als Dirac-Notation, benannt nach dem Physiker Paul Dirac. [Hom05] In der Quanten Theorie gelten Objekte, die mit dieser Notation ($|OBJEKT\rangle$) beschrieben und abgeschlossen sind, als Zustände, Vektoren oder als ein Ket. [McM07]

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{2.1}$$

Um sich das Verhalten eines Qubits visuell besser vorstellen zu können, kann man die sogenannte Blochsphäre zurate ziehen (siehe Abbildung 2.1). Man kann sich das Qubit als einen Punkt auf der Oberfläche dieser Kugel vorstellen, beschreiben lässt sich dieser Punkt durch eine Vektorschreibweise, wie beispielsweise in Gleichung 2.2 aufgezeigt. Ein Qubit kann die Zustände 0 und 1 repräsentieren, befindet sich das Qubit am einen Pol der Sphäre, so ist der Zustand beispielsweise 0, befindet sich das Qubit am anderen Pol der Sphäre, so ist der Zustand 1. Zusammengeschlossen bilden mehrere Qubits ein Quantenregister.

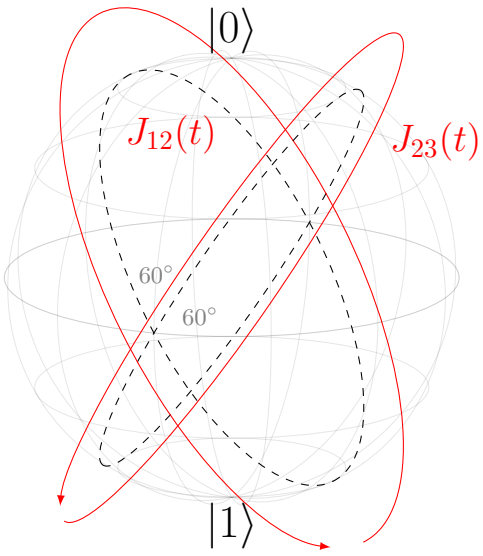

Abbildung 2.1: Darstellung einer Blochsphere

2.1.2 Superposition

Ein normales Bit kann sich entweder genau im Zustand 0 oder im Zustand 1 befinden. Dies gilt allerdings nicht für Qubits, da sich der Zustand dieser auch durch eine Linearkombination der beiden Zustände beschreiben lässt. Dieses Verhalten nennt sich Superposition und stellt eine Überlagerung von Zuständen dar. [Hom05]

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.2}$$

Alpha und Beta sind komplexe Zahlen, die die sogenannte Wahrscheinlichkeitsamplitude darstellen. Addiert man den quadrierten Betrag der beiden Amplituden auf, so erhält man immer 1. [Hom05]

$$|\alpha|^2 + |\beta|^2 = 1 \tag{2.3}$$

Bei einem klassischen Bit kann man den Zustand auslesen. Um den Zustand eines Qubit zu ermitteln, muss dieses gemessen werden. Die Amplituden α und β sind ausschlaggebend für das Ergebnis, das Qubit befindet sich mit einer Wahrscheinlichkeit von $|\alpha|^2$ in Zustand $|0\rangle$ und mit einer Wahrscheinlichkeit von $|\beta|^2$ in Zustand $|1\rangle$. Misst man ein Qubit, so wird seine Superposition zerstört, der Schritt ist irreversibel. Man misst für ein Qubit entweder genau den Zustand $|0\rangle$ oder genau den Zustand $|1\rangle$. [Hom05]

Im Beispiel der Blochsphäre aus Abbildung 2.1 würde der Punkt, der das Qubit darstellt, im Falle einer Superposition von Zuständen nun nicht mehr zwingend an einem der beiden Pole liegen, welche die Zustände $|0\rangle$ und $|1\rangle$ darstellen. Das Qubit kann auf einem beliebigen Punkt der Sphäre liegen und wird durch die Linearkombination der beiden Zustände beschrieben.

2.1.3 Quanten Gatter

Als Nächstes gilt es, Qubits und Ihren jeweiligen Zustand zu manipulieren. Eine Berechnung ist allgemein eine Folge von Zuständen des Rechners. Der aktuelle Zustand eines Systems kann durch einen Rechenschritt in einen Folgezustand überführt werden. Bei einem Beispiel von einem Bit, das derzeit im Zustand 0 liegt, kann der Zustand entweder unverändert bleiben, oder auf 1 gesetzt werden. Mithilfe mehrerer Bits und dem Einsatz von Logikgattern kann man nun mehrere Manipulationen an den Startzuständen der Bits vornehmen und eine Logik abbilden. (siehe Abbildung 2.2)

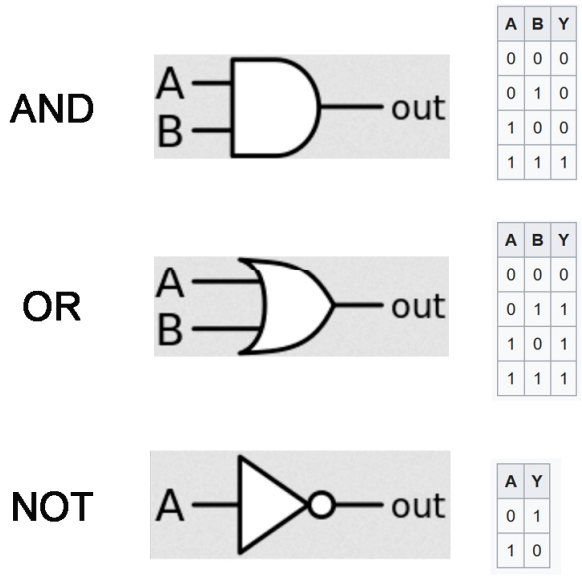

Abbildung 2.2: Beispiele klassischer Logikgatter (AND-, OR-, und NOT-Gatter) (nach [log])

Die Manipulation am Zustand eines Qubit wird mithilfe von Quantengattern durchgeführt, wobei der Übergang zwischen verschiedenen Zuständen im Gegensatz zum klassischen Computer mit den klassischen Gattern besonderen Bedingungen unterliegt. Ein Quantengatter führt eine unitäre Transformation aus. [Hom05] Die möglichen Rechenschritte auf einem Qubit werden durch quadratische Matrizen beschrieben, genauer durch 2×2-Matrizen, die unitär sind. Unitäre Matrizen beschreiben mathematisch, wie sich ein abgeschottetes Quantensystem ändert. Aus einer unitären Matrix lässt sich besonders einfach die inverse Matrix beziehungsweise die Umkehrabbildung ableiten. [McM07]
Durch die Eigenschaft, dass Quantenschaltkreise umkehrbare Berechnungen durchführen, ist die Anzahl der Eingabebits, die vom Quantengatter transformiert werden, genauso groß wie die Anzahl der Ausgabebits desselben Gatters. Der Zusammenschluss der einzelnen Quantengatter bildet dann einen Quantenschaltkreis. Die wichtigsten unitären Transformationen beziehungsweise Quantengatter sind das Hadamard Gatter, das C-NOT Gatter und die Pauli Matrizen, mit denen sich X Y und Z Gatter realisieren lassen (siehe Abbildung 2.3). [Hom05]

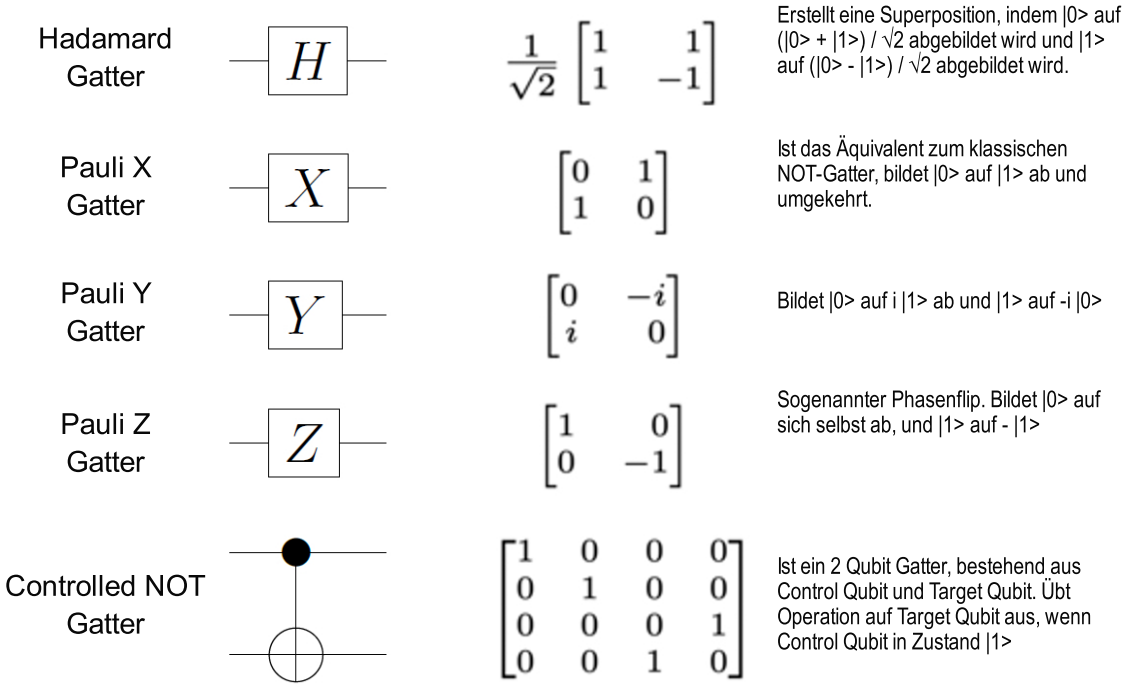

Abbildung 2.3: Beispiele Quantengatter (Hadamard-, X-, Y-, Z- und CNOT-Gatter) (nach [quac])

2.1.4 Verschränkung

In klassischen Computern sind die einzelnen Bits unabhängig voneinander. Der Zustand eines Bits beeinflusst nicht den Zustand der anderen Bits. Dies verhält sich bei Quantencomputern anders: Mehrere Qubits können miteinander verschränkt werden und Teil eines größeren Quantensystems oder Quantenzustands sein. Manipuliert man eines der verschränkten Qubits, so wird der Ausgang der Messung eines zweiten, mit dem ersten Qubit verschränkten,

Qubits beeinflusst. Dieses Verhalten nennt man Verschränkung und gilt als das Charakteristikum der Quantenwelt. [Hom05]

Um dieses Konzept zu veranschaulichen, können wir uns folgenden Quantenschaltkreis mit zwei Qubits und zwei Quantengattern vorstellen:

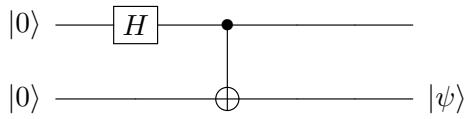

Tabelle 2.1: Verschränkung zweier Qubits

Auf das Quantenregister bestehend aus den beiden Qubits, die sich beide jeweils im Anfangszustand $|0\rangle$ befinden, werden Quantengatter angewendet. Das erste Qubit wird durch ein Hadamard Gatter transformiert und im Anschluss folgt eine Transformation beider Qubits durch das C-NOT Gatter. (siehe Abbildung 2.1) Das Hadamard Gatter überführt das Qubit A in eine Superposition, das heißt bei Messung ist das Ergebnis $|0\rangle$ oder $|1\rangle$, mit jeweils genau 50 Prozent Wahrscheinlichkeit. Messen wir für Qubit A den Zustand $|0\rangle$, so ist der Folgezustand und somit der Zustand des Quantenregisters $|00\rangle$. Messen wir für Qubit A den Zustand $|1\rangle$, so ist der Folgezustand des Quantenregisters $|11\rangle$. Somit ist der Zustand von Qubit B abhängig vom Zustand von Qubit A. [Hom05]

Durch die besonderen Eigenschaften von Superposition und Verschränkung ist es einem Quantencomputer möglich, alle möglichen Zustände bzw. eine Wahrscheinlichkeitsverteilung dieser gleichzeitig betrachten zu können (2^n, wobei n die Anzahl der Qubits repräsentiert), während ein klassischer Computer jeden dieser Zustände nacheinander berechnen muss. Dies ermöglicht eine Parallelität in der Berechnung. Durch die Messung der Zustände bzw. des gesamten Zustands des Systems erhält man nur eine einzige Lösung, die Wellenfunktion des Gesamtsystems kollabiert und die Superposition der einzelnen Qubits verloren. [Hom05]

2.1.5 Interferenz

In der Quantenmechanik wird der Zustand von Partikeln und somit auch mit einer Wellenfunktion beschrieben. Durch externe Einflüsse kann die Wellenfunktion eines Qubits gestört werden. Dies ist die sogenannte Interferenz. Verschränkt man mehrere Qubits miteinander, addieren sich die Wellenfunktionen der einzelnen Qubits miteinander und ergeben die allgemeine Wellenfunktion des Systems, in dem sie sich befinden. [Hom05]

Die bei der Kombination von einzelnen Wellenfunktionen auftretende Interferenz kann sowohl konstruktive als auch destruktive Auswirkungen haben, d. h. beträgt der Wert von zwei kombinierten Wellenfunktion an der gleichen Stelle jeweils einen positiven Wert, so wird dieser Wert verstärkt. Bei einer destruktiven Interferenz gleichen sich ein positiver Wert der einen Wellenfunktion an einer bestimmten Stelle mit einem negativen Wert der selben Mächtigkeit der zweiten Wellenfunktion aus. Allgemein können sich also die Amplituden eines Quantensystems aufaddieren oder auslöschen. [Hom05]

2.1.6 Dekohärenz

Um das System und seine Wellenfunktion verlässlich aufrecht und kohärent zu halten, muss es idealerweise nahezu perfekt isoliert und in manchen Systemen sogar auf annähernden Nullpunkt heruntergekühlt werden. Nur so ist es möglich, bei Messung des Quantensystems einen verlässlichen Wert zu erhalten. Die Betonung liegt hier auf nahezu, da es durch den Vorgang der Messung, bei dem offensichtlich eine Interaktion mit dem Quantensystem stattfinden muss, nicht zu einer perfekten Isolation kommen kann. Quantensysteme sind sehr empfindlich auf äußere Beeinträchtigungen und Rauschen der Umgebung. So wird der Zustand eines Qubit, wenn es zu stark in Bewegung ist oder von externen Störungen beeinflusst wird, viel stärker schwankend sein und somit eine Messung unzuverlässiger machen. Je größer nun das System aus Qubits ist, desto größer ist der Grad des Rauschens und der Grad der Unzuverlässigkeit, dem die einzelnen Qubits unterliegen. Das Ergebnis der Messung der Zustände wird somit mehr und mehr verfälscht, was die Vorteile des Quantum Computings folglich zunichtemachen kann. [Hom05]

2.2 Machine Learning

Das Thema künstliche Intelligenz ist hochaktuell und auch heute schon kaum noch aus unserem Alltag wegzudenken. Von Suchmaschinen wie Google und intelligenten Algorithmen zur Empfehlung von Videos auf Youtube, über Sprachassistenten wie Siri, Alexa bis hin zu selbstfahrenden Autos, die nicht mehr nur eine Einparkhilfe anbieten - künstliche Intelligenz unterstützt die Menschheit im alltäglichen Leben. Oftmals wird dieser Begriff allerdings inflationär benutzt und falsch verstanden. Man muss stark unterscheiden zwischen schwacher und starker Intelligenz, Programme, die heutzutage als intelligent bezeichnet werden, sind dies meist nur in einer einzelnen Nischendisziplin, wie beispielsweise der Schachcomputer DeepBlue. [Chea] Denn auch wenn diese künstliche Intelligenz den damaligen Schachweltmeister Kasparov schlagen konnte, so ist es ihr nicht möglich, als Suchmaschine zu agieren oder Videoempfehlungen durchzuführen. Um auf lange Sicht von einer schwachen künstlichen Intelligenz bis zu einer starken oder sogar allgemeinen künstlichen Intelligenz zu kommen, ist es nötig, der Maschine, dem Modell oder dem Programm schrittweise beizubringen, wie sie/es sich zu verhalten hat. Die damit zusammenhängenden Techniken werden grob unter dem Begriff maschinelles Lernen ML zusammengefasst. Beim maschinellen Lernen werden Lernalgorithmen implementiert, mithilfe derer das Programm in Trainingsdaten bestimmte Muster erkennen kann und anhand dieser erlernten Regeln folgende Testdaten korrekt bestimmen kann. [RN10] Der Ansatz ist - ähnlich wie bei einem Menschen - die Generierung von Wissen aus Erfahrung. [Fra] Es gibt verschiedene unterschiedliche Ansätze beim maschinellen Lernen, die je nach Problemstellung und Informationsgrundlage gewählt werden können.

2.2.1 Überwachtes Lernen

Einer der Ansätze wird als überwachtes Lernen bezeichnet, mit welchem Klassifikation und Regression auf Datensätze angewendet wird. Bei diesem Lernverfahren benötigt man einen Datensatz D, der sowohl aus Attributensätzen $A = \{A_0, A_1, ..., A_{|n-1|}\}$ (die jeweils beliebig viele Einzelattribute enthalten können), als auch aus sogenannten Klassenattributen/Labeln, $L = \{L_0, L_1, ..., L_{|n-1|}\}$ besteht. $|n|$ stellt die Anzahl von Attributensätzen und dazugehöri-

gem Label dar. Im bestmöglichen Fall existiert für jeden Satz Attribute A des Datensatzes D ein Label L, anhand dessen das jeweilige Objekt eindeutig klassifiziert werden kann. Ausschlaggebend für die Klassifikation ist die Beziehung zwischen Attributensatz und Label. [Liu11]
Der Datensatz wird in Trainingsdaten und Testdaten aufgeteilt, wobei der Trainingsdatensatz für das Training des Modells eingesetzt wird, während man mit dem Testdatensatz beispielsweise die korrekte Klassifizierung des Modells verifiziert wird.

Der überwachte Lernalgorithmus erhält einen Datensatz D mit Attributensätzen A und Labeln L als Eingabewerte. Die Label L stellen die korrekten Ausgabewerte dar. Nach der Extraktion von Merkmalen aus den Attributensätzen folgt das Training des Modells. Das Ziel des überwachten Lernens ist es, eine Funktion zur Klassifizierung einer Instanz, basierend auf seinem jeweiligen Attribut A und dem dazugehörigen Label L, zu generieren. Hierbei wird der Algorithmus kontinuierlich von selbst verbessert, sodass mit zunehmender Anzahl von Trainingsdaten eine immer bessere Performance und höhere Genauigkeit beim korrekten Klassifizieren erzielt wird. [Liu11] Im Anschluss wird das Modell nach Erreichen einer festgelegten Erfolgsquote mithilfe des Testdatensatzes überprüft und verifiziert.

2.2.2 Unüberwachtes Lernen

Ein weiterer Ansatz ist das unüberwachte Lernen. Dieser Ansatz arbeitet mit - oftmals sehr unstrukturierten - Datensätzen, die keinerlei Labels enthalten müssen und somit keine Auskunft besteht, wie die Daten korrekt klassifiziert werden sollen. Im Gegensatz zu dem überwachten Lernen reicht hierbei eine hohe Menge an Trainingsdaten aus, anhand derer Muster und Strukturen im Datensatz erkannt werden sollen. überwachte Lernalgorithmen organisieren Instanzen in verschiedene Gruppen, die dieselben Ähnlichkeitsmerkmale aufweisen. Diese Gruppen werden auch Cluster genannt. [Liu11]

Beim unüberwachten Lernen erhält das Modell einen Datensatz D mitsamt der einzelnen Attributsätze A als Eingabewerte. Nach der Merkmalsextraktion werden die Instanzen in Cluster aufgeteilt. Ein Cluster ist eine Sammlung von Instanzen eines Datensatzes, die basierend auf ihren Attributsätzen eine Ähnlichkeit zueinander aufweisen und in ihren Attributsätzen Unterschiede zu anderen Clustern desselben Datensatzes aufweisen. Die einzelnen Instanzen können als Datenpunkte in einem mehrdimensionalen Raum interpretiert werden, wobei die Anzahl der Dimensionen gleich der Anzahl an Attributen pro Attributsatz ist. [Liu11]

2.2.3 Neuronale Netze - Neural Networks

Überwachtes und Unüberwachtes Lernen wird in vielen Anwendungsfällen genutzt und bietet eine große Unterstützung in verschiedenen Gebieten. Ein Teilbereich des Machine Learnings ist das Deep Learning, oder tiefes Lernen. Hierbei ist das Ziel, anhand von künstlichen neuronalen Netzen das menschliche Gehirn und die Prozesse, die in ihm ablaufen, darzustellen. Diese neuronalen Netze werden dann zur Implementierung von Machine Learning verwendet. [SHH21] Neuronale Netze werden unter anderem bei der Analyse von Bild- und Videodaten,

sowie Sprach- und Textdaten verwendet. So werden diese beispielsweise bei Gesichts- und Objekterkennung eingesetzt. [Fra]

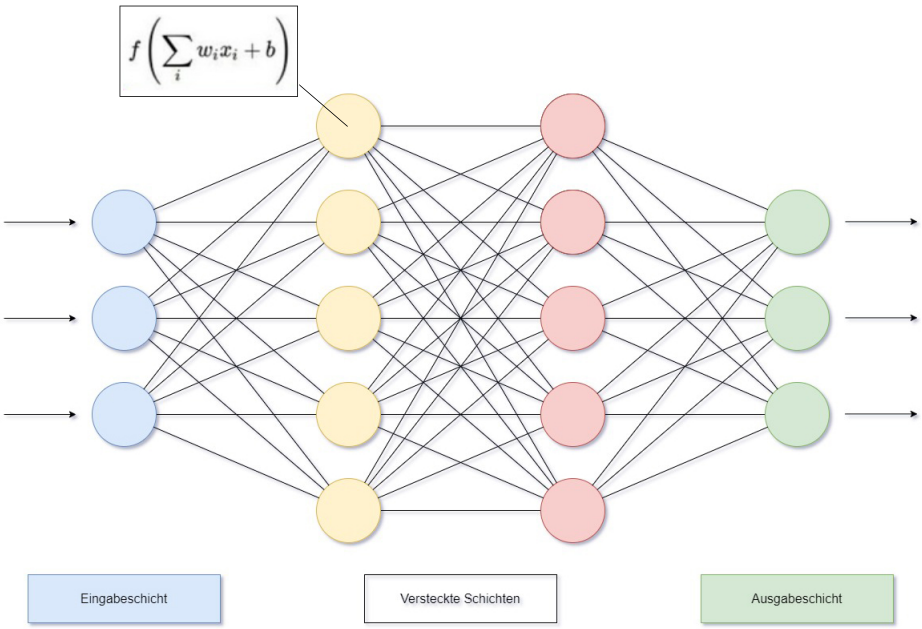

Abbildung 2.4: Visualisierung eines neuronalen Netzes

Ein neuronales Netzwerk ist eine Funktion, die aus kleinen Bausteinen namens Neuronen besteht. Diese elementaren Bausteine stellen jeweils eine simple, nicht lineare Funktion dar, welche einen oder mehrere Eingabewerte auf eine reelle Zahl abbildet (siehe Abbildung 2.4. Die Eingabewerte setzen sich aus den Werten x der vorangegangenen Neuronen, den Gewichten w, die den Verbindungen zugewiesen werden und dem Bias b, der festgelegt werden kann, zusammen. Nach Summierung der Werte für alle vorangegangenen Neuronen wird eine Aktivierungsfunktion f angewandt. Somit halten Neuronen nach der angewandten Funktion jeweils eine auf den Bereich von null bis eins normalisierte Zahl, die die Aktivierung der Zelle symbolisiert. Diese Neuronen sind in verschiedene Schichten in dem neuronalen Netz angeordnet. Ein neuronales Netz besteht aus einer Eingabeschicht, beliebig vielen versteckten Schichten und einer Ausgabeschicht. (siehe Abbildung 2.4) Für die Eingabeschicht kann beispielsweise ein Bild bzw. die einzelnen Pixel eines Bildes auf jeweils ein Neuron kodiert werden. Dementsprechend besteht die Eingabeschicht eines neuronalen Netzwerks zur Bilderkennung aus genau so vielen Neuronen, wie jedes Eingabebild Pixel enthält. [Nie18] Bei dem weitverbreiteten und oftmals für die Bilderkennung verwendeten MNIST Datensatz [Den12], einem Datensatz aus handgeschriebenen Ziffern, bestehen die einzelnen Bilder des 60.000 Instanzen großen Datensatzes aus 28x28 Pixeln. Somit umfasst die Eingabeschicht für die Klassifizierung dieser Instanzen nach Multiplikation der 28x28 Pixel 784 Neuronen. Die Eingabedaten werden nun mithilfe einer Aktivierungsfunktion auf den Bereich zwischen 0.0 und 1.0 normalisiert, sodass im Beispiel des MNIST Datensatzes jeder Pixel einen Graustufenwert enthält, wobei ein Wert gegen 0.0 einen schwarzen Pixel und ein Wert gegen 1.0 einen weißen Pixel repräsentiert.[Nie18] (siehe Abbildung 2.5)

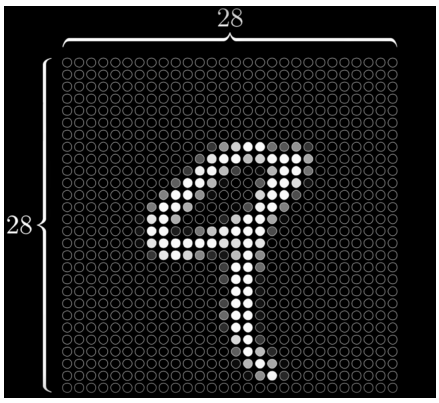

Abbildung 2.5: Kodierung von Pixelwerten in Neuronen [3Bl]

Diese Eingabedaten werden, wie in Abbildung 2.5 gezeigt, in die Neuronen kodiert und folglich in beliebig viele und manuell einstellbar große sogenannte versteckte Schichten weiter gegeben. Die versteckten Schichten enthalten eine festgelegte Anzahl von Neuronen. Ein Neuron einer Schicht N ist mit allen Neuronen der Schicht N-1 und der Schicht N+1 verbunden, zwischen ihnen findet ein Informationsaustausch statt. Eine weitere Schicht ist die erwähnte Ausgabeschicht, mithilfe dessen das Netzwerk kategorisieren oder klassifizieren kann, zu welcher Gruppe beispielsweise das Eingabebild zuzuordnen ist. Im Fall des MNIST Datensatzes besteht die Ausgabeschicht aus genau zehn Neuronen, die die Ziffern 0-9 abbilden. (siehe Abbildung 2.6) Ein höherer Wert eines Neurons in diesem Layer gibt eine höhere Wahrscheinlichkeit der korrespondierenden Ziffer als Klassifikation für das Eingabebild an. Die Verbindungen oder auch Kanten zwischen den einzelnen Neuronen erhalten jeweils eine Gewichtung in Form eines Skalars, welche mit dem Wert des Neurons, von dem die Verbindung ausgeht, verrechnet wird. Der daraus resultierende Wert wird als Eingabewert für das Neuron zu dem die Verbindung führt verwendet. Das Training des Neuronalen Netzes funktioniert dann durch eine Aktualisierung der jeweiligen einzelnen Kantengewichtungen, bis ein gewünschtes Verhalten - im beschriebenen Fall eine korrekte Klassifizierung der Eingabezahl - erreicht wird. [Nie18] Eine Vielzahl von aneinander gereihten versteckten Schichten nennt sich Deep Learning.

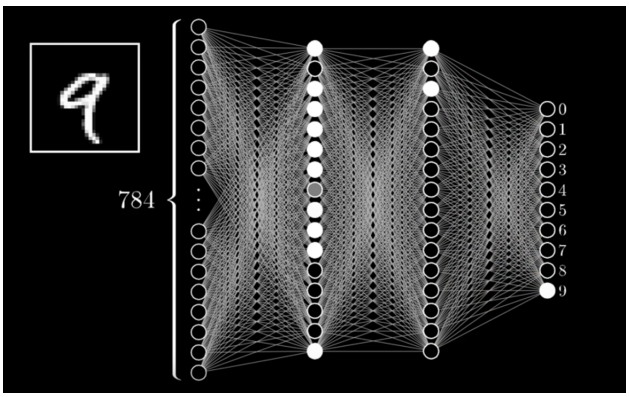

Abbildung 2.6: Neuronales Netz zur Klassifizierung der Instanz aus Abbildung 2.5 [3Bl]

Convolutional Neural Networks

Convolutional Neural Networks CNN sind eine Unterklasse von Neuronalen Netzen, die besonders bei computerbasiertem Sehen zum Einsatz kommen, genauer gesagt für Aufgaben wie beispielsweise Bild-, Objekt- und Gesichtserkennung. Dabei lernt das CNN automatisch verschiedene räumliche Hierarchien von Mustern, sowohl simple Muster wie beispielsweise Kanten und Ecken, aber auch komplexe Muster, die aus vielen einzelnen simplen Mustern bestehen. [ON15]

CNNs sind ähnlich aufgebaut wie die zuvor vorgestellten Neuronalen Netze, haben allerdings zusätzliche Baublöcke beziehungsweise Schichten. Die wichtigsten Schichten des CNNs sind die Convolution Schicht (deutsch: Faltung), die Pooling Schicht (deutsch: Zusammenlegen) und die Fully Connected Schicht (deutsch: vollständig verbunden). Eine typische CNN Architektur besteht aus sich wiederholenden Paaren von Convolution Schichten und Pooling Schichten, und einer oder mehreren Fully Connected Schichten. Die Convolution Schicht bildet eine lineare Operation ab, die für die Merkmalsextraktion verwendet wird. Hierbei wird ein sogenannter Kernel in Form eines Tensors - oftmals eine 3x3 Matrix - über das Eingabebild bewegt und für jede Position wird der Tensor des Eingabebilds mit dem Tensor des Kernels verrechnet. Bei diesem Schritt wird ein elementweises Produkt oder auch Hadamard-Produkt berechnet. Im Anschluss werden die Werte aufsummiert. Dadurch entsteht die gewünschte Faltung, die extrahierte Merkmalsmatrix hat im Vergleich zum Eingabebild eine reduzierte Höhe und Breite. (siehe Abbildung 2.7)Durch wiederholte Anwendung mit verschiedenen Kernels kann eine beliebige Anzahl von Merkmalsmatrizen erzeugt werden, die jeweils unterschiedliche Merkmale des Eingabebilds repräsentieren. [YNDT18]

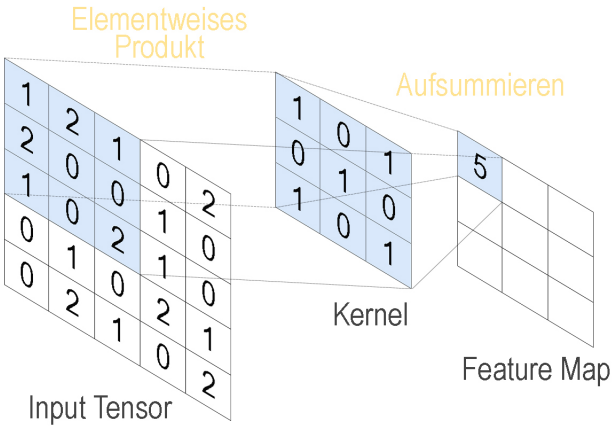

Abbildung 2.7: Faltungsvorgang mithilfe eines Kernels (nach [YNDT18])

Die Ausgaben der linearen Convolution- oder Faltungsoperationen werden dann an eine nicht-lineare Aktivierungsfunktion weitergegeben, die die Eingabewerte auf einen bestimmten Zahlenbereich berichtigt, um eine Aktivierung eines jeweiligen Neurons korrekt darzustellen. Eine der am meisten verwendeten Aktivierungsfunktionen ist dabei die Rectified Linear Unit (ReLu).

$$ReLu = f(x) = max(0, x) \tag{2.4}$$

Auf die Convolution Schicht folgt die Pooling Schicht, mit welcher man die Dimensionalität der Merkmalsmatrizen reduzieren und folglich die Zahl der Parameter minimieren kann. Hierbei ist eine der am meisten verwendeten Techniken das Max Pooling, bei welchem ein bestimmter Bereich festgelegter Größe auf den größten in diesem Bereich vorkommenden Wert reduziert wird. (siehe Abbildung 2.8) Eine oft verwendete Größe ist ein Kernel der Größe 2x2 und einer Schrittweite von 2, womit die Dimension der Merkmalsmatrix um den Faktor 2 verkleinert werden kann. [YNDT18]

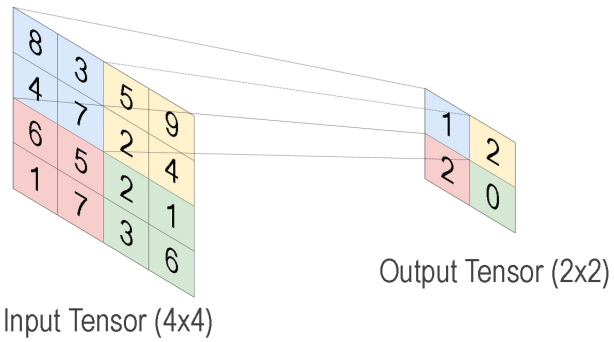

Abbildung 2.8: Pooling Vorgang mit einem 2x2 Kernel und einer Schrittweite von 2 (nach [YNDT18])

Die Merkmalsmatrizen der letzten Convolution Schicht und Pooling Schicht werden folglich in eine ein-dimensionale Form gebracht, beispielsweise in Form eines Vektors. Die Werte der Merkmalsmatrizen werden als Eingabewerte für die Fully Connected Schichten verwendet, wobei die letzte Fully Connected Schicht genau so viele Neuronen besitzt, wie es Klassen gibt, in die es das Eingabebild zu klassifizieren gilt. [YNDT18]

2.3 Quantum Machine Learning

Nachdem ML durch die hohen Anforderungen an die Berechnungsleistung an seine Grenzen kommt und das Paradigma des Quantencomputings neben weiteren Vorteilen beispielsweise echte Parallellität verspricht, wirkt eine Kombination aus beiden Berechnungsmodellen als sehr vielversprechend. [ZN20] Quantum Machine Learning basiert grundlegend auf den beschriebenen Konzepten von Quantum Computing und maschinellem Lernen. Hierbei ist das Ziel, klassische Daten und die Daten eines Quantencomputers zu kombinieren und von einem Quantum Machine Learning Algorithmus berechnen zu lassen.

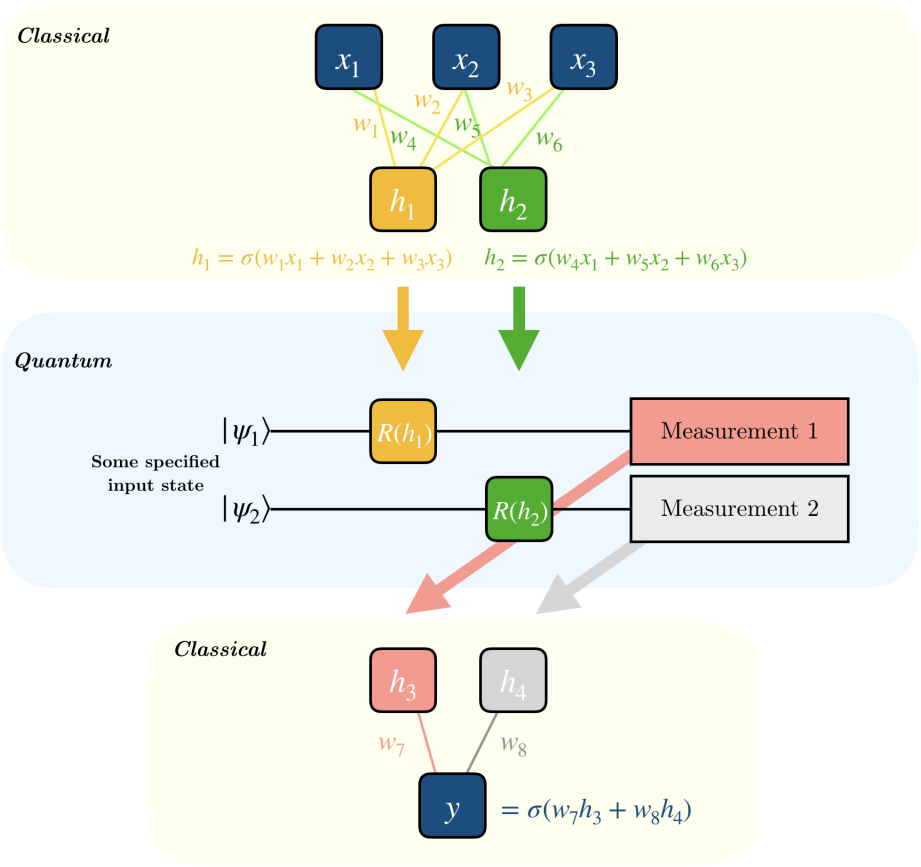

Abbildung 2.9: Konzept eines Quantum-klassischen hybriden Neuronalen Netzes [Qisc]

Da die Quantenprozessoren wie aufgezeigt bisher relativ klein und noch sehr störanfällig sind, bzw. ein starkes Rauschen innerhalb der Quantengatter aufweisen, ist es derzeit noch nicht möglich Quantum Machine Learning auf einem Quantenprozessor selbst auszuführen. Aus diesem Grund müssen Quantenprozessoren in der NISQ Ära gemeinsam mit klassischen Prozessoren arbeiten, um effektive Berechnungen durchzuführen, die für verschiedene Anwendungsfälle benötigt werden. Hierbei werden einige Rechenschritte auf einem klassischen Computer ausgeführt, während geeignete Aufgaben, die am ehesten von Quanten Computing profitieren, auf den Quantum Computer ausgelagert werden (siehe Abbildung 2.9). Es ist zu bemerken, dass die Aufgaben für den Quanten Computer nicht allzu komplexe Berechnungen abbilden. Komplexe Berechnungen werden durch die Dekohärenz des Quantenzustandes erschwert, je komplexer die Berechnung, desto mehr Operationen müssen durchgeführt werden und desto höher die Fehlerrate. Vielmehr nutzt man die echte Parallelität des Quantencomputers, um einfache Berechnungen parallel durchzuführen, wie beispielsweise eine Orakel Funktion. Denkbar sind hier Anwendungen wie beispielsweise das Lösen von Problemen der Linearen Algebra oder auch Ausführen von Quantum Principal Component Analysis, für welche bewiesen wurde, dass ein Quantencomputer gegenüber einem klassischen Computer eine exponentielle Beschleunigung aufweist. [quaa]
Das Ziel ist, klassische Daten und die Daten eines Quantencomputers zu kombinieren und von einem Quantum Machine Learning Algorithmus berechnen zu lassen. Verschiedene Machine Learning Algorithmen wurden bereits umkonzipiert, um auf hybriden Systemen bestehend aus klassischem Computer und Quantencomputer zu laufen. Dazu gehören sowohl der Quantum Support Vector Machine (SVM) Algorithmus, das Quantum K-Means Clustering und auch Quantum Neuronale Netze. [ZN20]

2.4 Verwandte Arbeiten

In den vorangegangenen Kapiteln wurde das vielversprechende Berechnungsparadigma des Quantum Computings vorgestellt. Mit den Eigenschaften der Quantenmechanik wie Superposition und Verschränkung lässt sich eine echte Parallellität in der Berechnung von Algorithmen erreichen. Eine in Zukunft immer weiter steigende Anzahl an Qubits führt zu einer exponentiellen Steigerung der möglichen Leistung des Systems, da pro Qubit die vorige Rechenleistung mit noch einem Qubit weniger verdoppelt wird. Eine Verkleinerung von Transistoren und somit eine erhöhte Transistorendichte auf einem Prozessorchip wird immer schwieriger. Darauf aufbauende Lernalgorithmen werden dadurch weniger schnell entwickelt.

High Performance Computing und Acceleratoren

Eine Lösung dieses Problems stellt ein heterogenes System aus CPU (Central Processing Unit) und GPU (Graphical Processing Unit) zur Berechnung von Algorithmen dar. GPU Computing ist mittlerweile zum Standard im Machine Learning geworden, bei leistungsintensiven und rechenaufwändigen Prozessen werden die Teile des Algorithmus, die eine erhöhte Leistung benötigen, auf die GPU ausgelagert. Besitzen die leistungsstärksten CPUs derzeit 12-16 Kerne, so verfügt eine GPU bereits über mehrere tausende Kerne, die dann letztendlich für eine parallelle Datenverarbeitung genutzt werden können. Während GPUs nicht zwingend erforderlich für Machine Learning sind und Modelle auch nur mit einer CPU

trainiert und evaluiert werden können, arbeitet eine GPU besonders effizient für verschiedene Aufgabenbereiche. GPUs sind eine beliebte Addition, um die CPU in Machine Learning Tasks zu unterstützen. Die hohe Anzahl der Kerne und die damit verbundene Parallellität unterstützen die CPU in rechenintensiven Schritten, sodass eine deutliche Beschleunigung des Trainings erreicht werden kann. Diesen Aspekt will man für das Quantum Machine Learning ebenfalls ausnutzen. Daher werden große Anstrengungen unternommen, um Quantum Machine Learning Algorithmen zu entwickeln. [LLZC15] [Sch15]

Quanten Frameworks

Für das Arbeiten mit Quantencomputern gibt es bereits mehrere Frameworks, die in den letzten Jahren veröffentlicht wurden. Cirq ist beispielsweise ein Open-Source Framework von Google, das es ermöglicht, Quanten Computer zu programmieren. Mithilfe der Python Software Library lassen sich Quantenschaltkreise erstellen und manipulieren, die sowohl auf NISQ Geräten, also Quantencomputer in der NISQ Ära, als auch in von klassischen Computern simulierten Umgebungen einsetzbar sind. [cira] [cirb] Michael Broughton et al. publizierten 2021 eine Arbeit zu TensorFlow Quantum (TFQ) , einer Open-Source Library und ein Software Framework um mit Quantendaten zu arbeiten. Die Kernidee war die Kombination von Cirq und Googles TensorFlow, einer Machine Learning Plattform, um eine Brücke zwischen Quantencomputing und maschinellem Lernen zu bilden. Mit TFQ lassen sich Prototypen von quantum-hybriden Modellen bauen, die dann für ML Algorithmen eingesetzt werden können. [BVM+20] Weitere Beispiele sind das Entwicklungskit Qiskit [qisb] und das von Atos entwickelte Quantenframework AQASM [aqa]. Diese Frameworks beschleunigen die Forschung und fördern das Verständnis des Arbeitens mit Quantencomputer und dem Quantum Machine Learning.

Quantum Machine Learning

Verschiedene Forscher haben in den letzten Jahren im Rahmen von akademischen Arbeiten bewiesen, dass der Einsatz eines Quanten Computers für Machine Learning Algorithmen erhebliche Vorteile bringt. [CB18] Der wohl am weitesten verfolgte Ansatz in der Umsetzung dieser Quantum Machine Learning Algorithmen beruht auf der Annahme, dass klassische Computer auch in Zukunft verwendet werden, um beispielsweise einzelne Details eine Quantum Computers zu steuern. Somit besteht der Ansatz aus einer hybriden Kombination aus einem klassischen und einem Quanten Computer. Es existieren verschiedene Arbeiten zu klassischen Machine Learning Algorithmen, die umgebaut wurden um auf Quantencomputern zu funktionieren und dessen Eigenschaften ausnutzen können. In [RML14] haben Rebentrost et al beispielsweise Support-Vector-Machines implementiert, Khan et al stellen in [KAVL19] eine Quanten Version des K-Means Algorithmus vor.

Auch Neuronale Netze werden auf Ihre Tauglichkeit in Bezug auf eine Quantumumgebung untersucht. Li et al setzen ein Quantum Deep Convolutional Neural Network ein, um eine mehrklassige Bildklassifizierung zu ermöglichen. Ähnliche Ansätze nur mit einem anderen Framework haben Fan et al. in [FSZ22] verfolgt und dabei gezeigt, wie mit einem QC-CNN effektiv wichtige Features aus Bildern extrahiert werden können und somit Klassifizierungen

von Eingabebildern ermöglicht werden. Weitere Arbeiten mit quantum-klassisch hybriden Algorithmen finden sich unter [TRN+21], [ZG21] und [HSPC20].

Zusammenfassung

Es wurden bereits mehrere Möglichkeiten zur Beschleunigung und Unterstützung von Prozessoren untersucht. Diese vorangegangenen Studien bilden eine Grundlage für ein hybrides System mit einem Quantencomputer. Für das Quantencomputing wurden verschiedene Frameworks entwickelt, um Quantenschaltkreise erstellen und manipulieren zu können und um Berechnungen auf echter Quantenhardware programmieren zu können. Mithilfe der Kombination mit herkömmlichen ML Frameworks oder der Nutzung durch Quantumframeworks wie TFQ ist es möglich, QML Algorithmen zu entwickeln und zu implementieren.
Die aus den verwandten Arbeiten gewonnenen Einsichten sollen nun für die Umsetzung eines hybriden Systems am LRZ genutzt werden. In dieser Arbeit werden folgende Bereiche untersucht.

- Implementierung eines heterogenen, quantum-klassischen Hybrids in einer bereits existierenden Umgebung basierend auf einem Linux Cluster als High Performance Computing Lösung (klassischer Part)
- Schnittstellenausarbeitung um Zugang zu Quantum Parts in Form der ATOS Quantum Learning Machine zu ermöglichen
- Scheduling und Workloadmanagement in der Hybridumgebung
- Datentransfer zwischen klassischem und Quanten Part

3 Konzept

In diesem Kapitel wird ein Konzept zu einem idealen System erläutert, welches ein quantum-klassisches Hybrid implementiert, um mit heterogenem Computing Quantum Algorithmen wie zukünftig beispielsweise auch Quantum Machine Learning Algorithmen berechnen und Modelle trainieren zu können.

3.1 Konzeptvisualisierung

In Abbildung 3.1 wird konzeptuelles Systemdiagram für ein quantum-klassisch hybrides System dargestellt.

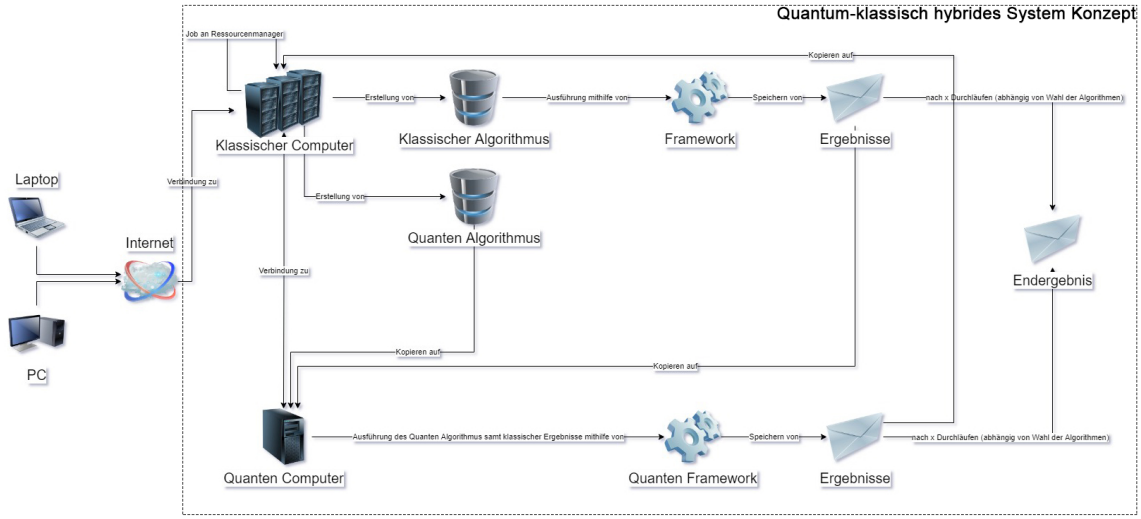

Abbildung 3.1: Systemdiagram zu Konzept für quantum-klassisch hybrides System

3.2 Hardware

Im Folgenden werden die Hardwarekomponenten eines quantum-klassisch hybriden Systems aufgelistet.

3.2.1 Klassischer Computer

Für den klassischen Teil des Systems wird ein klassischer Computer eingesetzt. Dieser besteht wiederum aus diversen Einzelkomponenten, die beim Trainieren und Anwenden von Modellen benötigt werden.

CPU/GPU

Wie schon in den verwandten Arbeiten zu lesen, gibt es verschiedene Ansätze bezüglich der Hardware, um Aufgaben wie Machine Learning zu bearbeiten. Als das Herzstück eines jeden Computers gilt die Central Processing Unit, kurz CPU. Allgemein konnte man feststellen, dass andere Hardwarekomponenten sich als deutlich besser für Machine Learning Zwecke geeignet herausstellten, als eine herkömmliche, in einem typischen Desktop-Computer verbaute CPU. Diese umfassen beispielsweise Grafikprozessoren, Application-specific Integrated Circuits (ASIC) oder auch Field Programmable Gate Arrays (FPGA). Für ein Training von Machine Learning Modellen im großen Stil eignet sich jedoch auch ein modernes System eines Hochleistungsrechners, beziehungsweise eines Hochleistungsrechnerverbunds. [hwr] Durch den Verbund mehrerer Rechner und somit deren Prozessoren wird die Anzahl der verfügbaren Kerne drastisch erhöht. Verschiedene derzeit weltweit eingesetzte Supercomputer verfügen über mehrere tausend Prozessoren mit teils sogar jeweils mehreren hundert Kernen, was zu einer Gesamtanzahl an Kernen von einigen hunderttausend bis millionen Kerne kumuliert. Ein Beispiel hierfür ist der Hochleistungsrechner Summit von IBM. [sum] Dieser besteht aus 9216 IBM POWER9 Prozessoren mit jeweils 22 Kernen, zusätzlich sind in diesem System 27.648 Grafikprozessoren von NVIDIA verbaut, die die Rechenleistung nochmal um ein Vielfaches steigern. Multithreading und die hohe Anzahl an Kernen ermöglichen solchen Supercomputer und Rechenzentren eine sehr leistungsstarke Performance im Bereich des Machine Learnings. [sum]

RAM

Die nächste für das System unabdingbare Komponente ist der Arbeitsspeicher (englisch Random Access Memory - RAM). Datensätze für das Machine Learning bestehen im Optimalfall aus so vielen einzelnen Proben oder auch oft Samples genannt, wie möglich. Ein Beispiel hierfür ist der vergleichsweise ressourcen- und speicherarme Datensatz MNIST, welcher handgeschriebene Ziffern beinhaltet. Jedes Sample dieses Datensatzes hat eine Größe von 28x28 Pixeln, insgesamt besteht der Datensatz aus 60.000 Trainingsproben und 10.000 Testproben [Den12]. Andere Datensätze können sowohl die Einzelbildgröße als auch die Größe des Gesamtdatensatzes um ein Vielfaches übertreffen. Diese Daten in Arrays zu laden erfordert eine erhebliche Größe des Arbeitsspeichers. Auch die folgenden Berechnungen der einzelnen Schritte eines Machine Learning oder Deep Learning Algorithmus benötigen ausreichend Arbeitsspeicher. Der angesprochene Supercomputer Summit führt beispielsweise pro Prozessor 512 GigaByte DDR4 Arbeitsspeicher, mit einer Gesamtkapazität von 10 PetaBytes. Allgemein gilt für den Arbeitsspeicher genauso wie für den Prozessor bzw. die Anzahl der Prozessoren und Kerne: Je mehr, desto besser. [hwr] [sum]

Festplattenspeicher

Durch die Daten-intensive Natur von Machine Learning, insbesondere in Hinblick auf die Anwendungsfälle, für die ein System mit einem solchen Hochleistungsrechner eingesetzt werden soll, ist es wichtig, ausreichend Speicherplatz für etwaige Applikationen und Modelle bereitzustellen. Dieser ist vorrangig für das lokale Speichern von Trainingsdaten vorgesehen, welche recht viel Speicherplatz verbrauchen können. Allerdings bewegt sich dies im Rahmen von einigen TeraByte, ein Hochleistungsrechner wie etwa der zum Vergleich herangezogene Summit von IBM verfügt sogar über 250 PetaByte, was selbst höhere Anforderungen

als 100 TeraByte Speicherplatz um ein Vielfaches abdeckt. Allgemein gilt hier, je größer der Datensatz der Trainingsdaten und je hochauflösender beispielsweise für Klassifikationszwecke verwendete Trainingsbilder sind, desto größer ist dementsprechend der für die Daten benötigte Speicherplatz. Die beim Training der verschiedenen ML Modelle verwendeten Datensätze können entweder gesammelt im Dateisystem des Rechenverbunds oder Supercomputers abgelegt werden, oder aber partioniert auf den einzelnen Nodes, die für das Training eingesetzt werden. [INN+21]

3.2.2 Quanten Computer

Als zweiten Part für das quantum-klassische Hybrid benötigen wir einen Quantencomputer. Hier bieten sich in der NISQ Ära zwei Möglichkeiten an. Die erste Möglichkeit ist ein Quantum Computer mit echten, physischen Qubits, welche stand heute auch 'reale' Qubits genannt werden können - ideale Qubits wären hierbei fehlerfrei und mit optimalerweise maximal hoher Dekohärenzzeit, was jedoch mit derzeitigem technischen Stand noch nicht möglich ist. Die zweite Option stellt ein Quantensimulator dar, mithilfe dessen das Verhalten eines Quanten Computers simuliert werden kann.

Quanten Computer

Verschiedene Unternehmen bemühen sich um die Entwicklung und den Fortschritt eines Quanten Computers. Mehrere unterschiedliche Ansätze werden hierbei verfolgt, um echte Qubits zu erzeugen. Die dabei bekanntesten und bisher am meisten erforschten umfassen Ansätze mit optischen Qubits, mit Qubits abgebildet durch Ionenfallen und supraleitenden Qubits. Der derzeitige Fortschritt kann der Einfachheit halber an der Anzahl der bisher erzeugten und einsetzbaren Qubits gemessen werden. Für einen präziseren Vergleich von Quanten Computern müssen unter anderem die Zustände der Qubits in Bezug auf Qualität und Anfälligkeit bezüglich Rauschen berücksichtigt werden.
Das den Ansatz der optischen Qubits verfolgende Unternehmen Xanadu stellt Entwicklern über ihre Photonic Quantum Cloud Platform Systeme mit bis zu 24 Qubits zur Verfügung [xan], der aktuellste Chip von Xanadu arbeitet bereits mit 40 Qubits. Das Unternehmen IonQ arbeitet mit Ionenfallen, um Qubits zu realisieren. Die Ionenfallen von Ionq verfügen mittlerweile über 32 Qubits, mit Schätzungen von 50-60 Qubits in den kommenden 12 Monaten. [ion] Intel, Google und IBM arbeiten mit dem Ansatz der supraleitenden Qubits. Das hierbei am - was Qubit Zahlen angeht - fortgeschrittenste Unternehmen ist IBM, deren Eagle Prozessor mit 127 Qubits arbeitet. [ibma]

Quanten Simulator

Um Prototypen für Quantenschaltkreise und Quantenalgorithmen zu entwerfen ist es möglich einen Quanten Simulator anstelle eines echten Quanten Computers einzusetzen. Hierbei werden die Quanten Schaltkreise auf einem klassischen Computer simuliert. Mithilfe dieser frühen Prototypen von Quanten Simulatoren ist es möglich, die Entwicklung und den Einsatz von zukünftigen Quanten Computern zu unterstützen. Diese sind besonders nützlich für den Einsatz in der NISQ Ära, um durch Rauschen beeinflusste niedrigzahlige Qubit Systeme zu analysieren und Schlussfolgerungen für die weitere Entwicklung von Quantum Technik ziehen zu können. Bis zu 50 Qubits können derzeit auf diese Art und Weise von Quanten Simulatoren simuliert werden. [ABC+21] [ibmb]

3.3 Architektur

Nachfolgend wird die Architektur beschrieben, die für den Aufbau eines funktionierenden quantum-klassischen Hybrids benötigt wird. Dazu gehören in erster Linie Verbindungsaufbau und Informationsaustausch zwischen beiden Parts.

3.3.1 Informationstransfer

Da das System aus den beiden Hauptkomponenten klassischer Computer und Quanten Computer besteht, ist es nötig, einen Workflow aufzusetzen, sodass ein Informationsfluß zwischen beiden Teilen hergestellt werden kann. Es muss eine Möglichkeit geben, von einem Systemteil wie beispielsweise dem klassischen Computer auf den jeweils anderen - den Quanten Computer - zuzugreifen. Hierbei bietet es sich an Netzwerkprotokolle zu nutzen, die es einem Nutzer erlauben, sich auf einen Computer innerhalb eines anderen Netzwerks zu schalten. Nach Herstellung der Verbindung von Host A zu Host B muss ein Datentransfer ermöglicht werden, mit dem auszuführende Programme, Ergebnisse dieser Programme und weitere Dateien oder Informationen von Host A zu Host B und vice versa übertragen werden können.

3.3.2 Scheduler

Um den Ablauf von rechenintensiven nicht-hybriden und hybriden Programmen, die auf einem Supercomputer ausgeführt werden sollen, zu steuern und besonders um die Vielzahl an freien Rechenkernen ausnutzen zu können, bedarf es eines Steuerprogramms (englisch scheduler). Dieser Scheduler ist für die Planung und Verteilung eines Programms und seinen Threads oder Prozessen verantwortlich. Mithilfe dieser Lösung werden alle verfügbaren Computer Ressourcen bestmöglich genutzt, sodass die Leistung der CPU oder in diesem Fall eines Verbunds von CPUs maximiert werden kann. Somit kann eine echte Parallellisierung in der Berechnung von Programmen erreicht werden.

Über eine Schnittstelle müssen Workflows und verschiedene zu bearbeitende Anweisungen definierbar sein, welche folglich gepackt werden und an andere Maschinen verteilt werden können. Für das geplante Machine Learning sind oft wiederholte Schritte nötig, sodass eine automatische Abwicklung der aufkommenden Berechnungen erforderlich ist.

3.4 Software

Für den bloßen Informationsaustausch zwischen den beiden Subsystemen ist keine weitere Software erforderlich. Möchte man nun Machine Learning Modelle auf dem hybriden System einsetzen, ist es nötig, weitere Software, Programmiersprachen und Frameworks zu verwenden.

3.4.1 Framework für klassischen Computer

Wie in 2.4 Verwandte Arbeiten beschrieben, gibt es bereits verschiedene Frameworks, die Machine Learning Algorithmen vereinfacht haben und das Arbeiten mit diesen deutlich erleichtern. Als im Bereich des Machine Learnings vorherrschende Programmiersprache gilt hierbei Python, auf welcher die entwickelten Frameworks aufbauen. Hierbei gibt es nun mehrere verschiedene Ansätze, mithilfe derer sich klassische Machine Learning Algorithmen wie

Regression, K-nearest neighbor oder etwa Support Vector Machines (SVM) implementieren lassen - beispielsweise durch Frameworks wie Scikit-learn. Andere Libraries wie pyTorch [pyt], TensorFlow [ten] und das auf letzterem aufbauende Framework von Keras [ker]ermöglichen zusätzlich den Einsatz von Neuronalen Netzen.

3.4.2 Framework für Quanten Computer

Es wurden bereits Frameworks entwickelt, um mit Quanten Computern arbeiten zu können und Algorithmen für Quanten Schaltkreise zu entwerfen. Xanadu bietet eine full-stack Python Bibliothek namens 'Strawberry Fields' für das Arbeiten mit photonischen Quanten Computern an [str] , eine weitere Bibliothek namens 'Pennylane' wurde von Xanadu für das Quantum Machine Learning entwickelt. [pen] Google hat für das Entwickeln und Optimieren von Quanten Schaltkreisen das Framework 'Cirq' herausgebracht. Diese können sowohl auf Quanten Computern als auch auf Quanten Simulatoren eingesetzt werden. Somit ist es möglich, das Rauschen der derzeitigen NISQ Geräte zu simulieren und bei der Ausführung der Schaltkreise miteinzuberechnen. [cira] Das ebenfalls von Google entwickelte Framework 'TensorFlow Quantum' ermöglicht den Einsatz von Quantum Machine Learning. [tfq]. Das Unternehmen IBM bietet mit 'Qiskit' ebenfalls ein Open-Source Quantum Framework. [qisb] Somit bietet sich für das Erstellen und Nutzen eines quantum-klassisch hybriden Systems eine größere Auswahl an möglicher Software und Frameworks, um Quanten Schaltkreise realisieren zu können.

4 Implementierung

Das im vorigen Kapitel beschriebene Konzept eines quantum-klassischen Hybridsystems soll nun aufgebaut werden. Die Implementierung des Systems erfolgt in diesem Fall in der bereits bestehenden Umgebung LRZ. Das LRZ in Garching ist das wissenschaftliche Rechenzentrum für beide Münchener Universitäten, die Technische Universität München und die Ludwig-Maxmimlians-Universität, sowie ein nationales und europäisches Höchstleistungsrechenzentrum. [lrza] Das Leibniz-Rechenzentrum hat derzeit mehrere Hochleistungsrechner und auch Höchstleistungsrechner, wie beispielsweise den SuperMUC-NG in Betrieb. Die Hochleistungsrechner als einzelne Segmente bilden im Zusammenschluss den Linux Cluster des LRZ. Mithilfe der Hoch- und Höchstleistungsrechner ist es möglich, Forschungs- und Entwicklungsarbeiten auf dem Gebiet der Informatik durchzuführen und neue Technologien und Anwendungsfelder zu erschließen. [lrzb]

4.1 Zusammenfassung der Komponenten

In Abbildung 4.1 wird das Systemdiagram zum hybriden System am LRZ dargestellt.

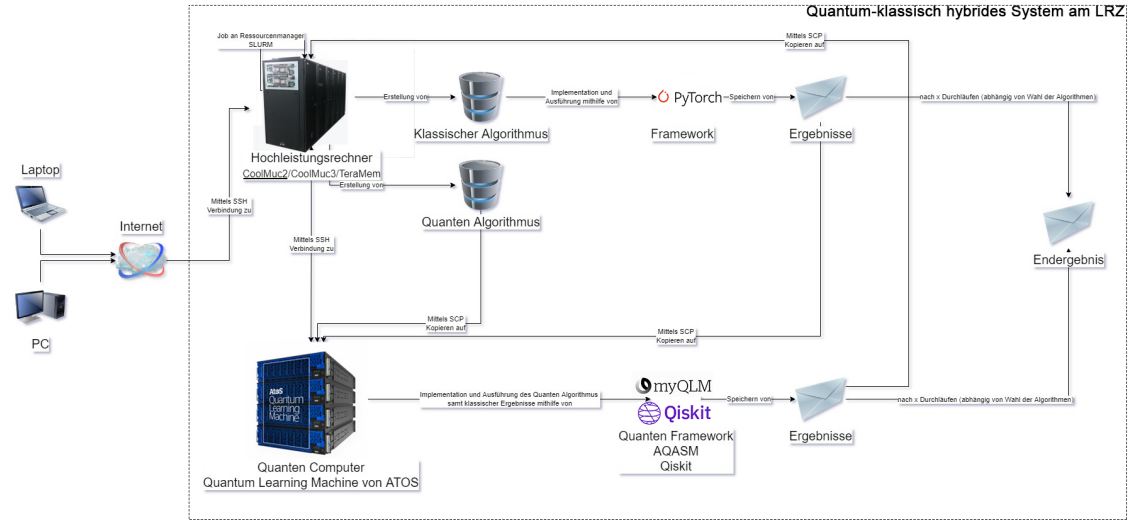

Abbildung 4.1: Systemdiagram für quantum-klassisch hybrides System am LRZ

4.2 Hardware

In diesem Abschnitt werden die Hardwarekomponenten des quantum-klassisch hybriden Systems aufgeführt.

4.2.1 Linux Cluster

Für das Hybrid wird das Linux Cluster des Leibniz-Rechenzentrums als klassischer Computer eingesetzt. Das Cluster besteht aus mehreren einzelnen Segmenten, die jeweils verschiedene Verbindungstypen und verschiedene Shared Memory Größen haben. Für die Implementierung kann beliebig aus den folgenden drei Segmenten des Clusters ausgewählt werden, welches dann für die klassischen Berechnungen des jeweiligen hybriden Programms zuständig sein wird. [lrzd]

Large Memory Teramem

Der Large Memory Teramem ist ein Intel Broadwell basierter Server mit 6 TeraByte Shared Memory. Teramem besteht nur aus einer einzigen Node, einem Prozessor des Typs Intel Xeon E7-8890 v4, der über 96 Kerne verfügt. Jeder dieser 96 Kerne besitzt jeweils 64 GigaByte DDR4 Arbeitsspeicher, was zu einem Gesamtspeicher von 6 TeraByte führt. Hiermit gilt Teramem als das einzige System des LRZ, das über mindestens 1 TeraByte Arbeitsspeicher in einer einzigen Node verfügt. Die Kernfrequenz beträgt 2.6 GigaHertz und die Leistung des Teramem liegt bei 13 TeraFlop/sekunde. (Siehe Abbildung 4.2)[ter]

Hardware	
Number of nodes	1
Cores per node	96
Hyperthreads per core	1
Core nominal frequency	2.6 GHz
Memory (DDR4) per core	64 GB
Memory per Node	6 TBV
Performance	13 TFlops
Software (OS and development environment)	
Operating system	SLES12 SP2 Linux
MPI	Intel MPI 2017, alternatively OpenMPI
Compilers	Intel icc, icpc, ifort 2017
Performance libraries	MKL, TBB, IPP
Tools for performance and correctness analysis	Intel Cluster Tools

Abbildung 4.2: Spezifikationen des Large Memory Teramem [ter]

CoolMUC-2

Der CoolMUC-2 ist ein Hochleistungsrechner und Nachfolger des CoolMUC-1. Das Cluster besteht aus 812 'Haswell' Intel Xeon E5-2697 v3 , mit jeweils 28 Kernen. Anders als der Teramem arbeitet dieses System mit 2 Hyperthreads pro Kern, die Frequenz von 2.6 GigaHertz bleibt gleich. Jede der 812 Nodes verfügt über 64 GigaByte DDR4 Arbeitsspeicher, mit einer Bandbreite von 120 GigaByte/sekunde. Die Bandbreite zwischen den einzelnen Nodes liegt bei 13.64 GigaByte/sekunde. Die Spitzenleistung des Systems liegt bei 1400

TeraFlop/sekunde. Das System wird sowohl für serielle als auch für parallele Berechnungen verwendet. (Siehe Abbildung 4.3) [cooa]

Hardware	
Number of nodes	812
Cores per node	28
Hyperthreads per core	2
Core nominal frequency	2.6 GHz
Memory (DDR4) per node	64 GB (Bandwidth 120 GB/s - STREAM)
Bandwidth to interconnect per node	13,64 GB/s (1 Link)
Bisection bandwidth of interconnect (per island)	3.5 TB/s
Latency of interconnect	2.3 µs
Peak performance of system	1400 TFlop/s
Infrastructure	
Electric power of fully loaded system	290 kVA
Percentage of waste heat to warm water	97%
Inlet temperature range for water cooling	30 ... 50 °C
Temperature difference between outlet and inlet	4 ... 6 °C
Software (OS and development environment)	
Operating system	SLES15 SP1 Linux
MPI	Intel MPI 2019, alternatively OpenMPI
Compilers	Intel icc, icpc, ifort 2019
Performance libraries	MKL, TBB, IPP
Tools for performance and correctness analysis	Intel Cluster Tools

Abbildung 4.3: Spezifikationen des CoolMUC-2 [cooa]

CoolMUC-3

Seit 2017 hat das Leibniz-Rechenzentrum den Hochleistungsrechner CoolMUC-3 in Betrieb, den Nachfolger des zuvor beschriebenen CoolMUC-2. Dieses System ist ausgestattet mit der ersten Standalone Prozessor Generation von Intel und verwendet die ebenfalls von Intel für 'High-Performance-Computing' entwickelte Vernetzungstechnologie Omni-Path, welche deutlich niedrigere Latenzzeiten, höhere Informationsraten und eine höhere Bandbreite liefert. Das System besteht aus 148 'Knights Landing' Intel Xeon Phi 7210-F-Prozessoren, mit jeweils 64 Kernen und 4 Hyperthreads pro Kern. Die Kernfrequenz liegt bei diesem System im Vergleich mit den zuvor vorgestellten Systemen mit 1.3 GigaHertz bei nur der Hälfte. (Siehe Abbildung 4.4) [coob]

Hardware	
Number of nodes	148
Cores per node	64
Hyperthreads per core	4
Core nominal frequency	1.3 GHz
Memory (DDR4) per node	96 GB (Bandbreite 80.8 GB/s)
High Bandwidth Memory per node	16 GB (Bandbreite 460 GB/s)
Bandwidth to interconnect per node	25 GB/s (2 Links)
Number of Omnipath switches (100SWE48)	10 + 4 (je 48 Ports)
Bisection bandwidth of interconnect	1.6 TB/s
Latency of interconnect	2.3 µs
Peak performance of system	459 TFlop/s
Infrastructure	
Electric power of fully loaded system	62 kVA
Percentage of waste heat to warm water	97%
Inlet temperature range for water cooling	30 ... 50 °C
Temperature difference between outlet and inlet	4 ... 6 °C
Software (OS and development environment)	
Operating system	SLES12 SP2 Linux
MPI	Intel MPI 2017, alternatively OpenMPI
Compilers	Intel icc, icpc, ifort 2017
Performance libraries	MKL, TBB, IPP
Tools for performance and correctness analysis	Intel Cluster Tools

Abbildung 4.4: Spezifikationen des CoolMUC-3 [coob]

4.2.2 Atos Quantum Learning Machine

Atos ist ein französischer Dienstleister, unter anderem im Bereich des High Performance Computings. Im Rahmen eines der ersten bedeutenden Programme für Quanteninformatik in Europa wurde die Atos Quantum Learning Machine entwickelt. [atod] Die QLM bietet Umgebungen für das Programmieren, das Optimieren und die Simulation von Quanten Algorithmen. Die QLM wird als Paket aus Hardware und Softwarelösung angeboten, mit dem bis zu 41 Qubits simuliert werden können. [atoa] Seit 2021 verfügt das Leibniz-Rechenzentrum über eine der kommerziell verfügbaren Quantum Learning Machine von Atos, welche im LRZ Quantum Integration Centre (QIC) installiert wurde. Damit gehört das LRZ zu den ersten Rechenzentren weltweit, die ein Quanten Computing System in eine bestehende Hochleistungsrechnerumgebung implementieren. [atob] Diese QLM (siehe Abbildung 4.5) werden wir im Rahmen der Arbeit als Quanten Computer einsetzen.

Abbildung 4.5: Die Quantum Learning Machine (QLM) von Atos [atod]

4.3 Architektur

Um die Kommunikation zwischen den einzelnen Systemkomponenten herzustellen und zu gewährleisten wird eine Architektur bestehend aus Software und Protokollen benötigt.

4.3.1 Informationstransfer

Im Folgenden werden die Protokolle erläutert, die für den Informationstransfer zwischen den Komponenten verwendet werden.

Secure Shell

Für die Interaktion der beiden Teile des quantum-klassischen Hybrids ist eine Verbindung zwischen den Systemen nötig. Über die Verbindung sollen Dateien zum Berechnen von hybriden Algorithmen verteilt werden. Dazu gehören sowohl Programmschritte wie ein auszuführendes Skript als auch die Ergebnisse der Berechnung dieser. Das Secure Shell (SSH) Protokoll ist ein Software Paket, welches eine sichere Administrationsumgebung innerhalb eines unsicheren Netzwerks ermöglicht. Mithilfe von Verschlüsselung wird die Verbindung zwischen zwei Systemen gesichert; Kommandos, Outputs und Datentransfers sind geschützt. Mit diesem Protokoll lässt sich eine sichere Verbindung zwischen zwei Teilen eines quantum-klassischen Hybrids herstellen. [ssh]

SSH wird in diesem System verwendet, um eine Verbindung zwischen den einzelnen Komponenten herzustellen, und somit den Programmstart von Algorithmen auf dem jeweiligen System zu initialisieren.

Secure Copy für sicheren Datentransfer

Für den Kopieren und den Transfer von Daten bzw. Dateien zwischen Systemen innerhalb eines Netzwerks kann das Secure Copy (SCP) Protokoll [scpb] verwendet werden. Das Protokoll baut auf dem SSH Protokoll auf. Für die Ausführung von Secure Copy muss eine Instanz eines SSH-Servers auf dem System aktiv sein, auf das Dateien kopiert werden sollen, um eine Dateiübertragung mit SCP zu ermöglichen. Im Fall von einem Kopieren von System A zu System B und zurück muss dementsprechend auf beiden Systemen ein SSH-Server laufen.

Das Protokoll bietet eine verschlüsselte Übertragung und gewährleistet die Vertraulichkeit, die Integrität und die Authentizität der zu übertragenden Daten. [scpa]

SCP wird in diesem System verwendet, um einerseits erstellte Algorithmen (in diesem Fall den auf dem klassischen Computer bzw. dem Rechencluster erstellten Quanten Algorithmus) von dem klassischen Computer auf den Quanten Computer zu kopieren und andererseits um bidirektional die mithilfe der Algorithmen berechneten Ergebnisse auszutauschen, die dann für weitere Berechnungen auf dem anderen System verwendet werden können.

Mit Einbindung eines von Linux Cluster und QLM gemeinsam geteilten Speichers ist die Nutzung von SCP obsolet und kann übersprungen werden.

4.3.2 SLURM als Workload-Manager/-Scheduler

Als Prozess-Scheduler oder Prozess Steuerungsprogramm verwenden wir die Software Simple Linux Utility for Resource Management (SLURM). Diese Open-Source Software ermöglicht für Linux Cluster beliebiger Größe ein fehlertolerantes und hochskalierbares Ressourcen Management und Steuerungssystem. Arbeitspakete werden als sogenannte Jobs verpackt, die mithilfe von SLURM mittels Batch Skripts für einen Linux Cluster geplant und dort ausgeführt werden können. Die Hauptfunktionen von SLURM sind sowohl das Zuweisen von Zugängen zu bestimmten exklusiven oder nicht-exklusiven Ressourcen innerhalb des verwendeten Linux Clusters, als auch ein Framework für das Starten, Ausführen und Kontrollieren von parallel ausgeführten Arbeitspaketen, sowie die Vermittlung von Ressourcen bezüglich anstehender noch ausstehender Arbeitsschritte. [slu]

Jede Node des verwendeten Linux Clusters verfügt über eine laufende Instanz eines slurmd Daemons, welcher sich in der Queue befindene Arbeitsschritte empfangen und abarbeiten kann. Nach Ausführung werden von der jeweiligen Node die Ergebnisse der Arbeitsschritte zurückgegeben und die Node ist für weitere anstehende Arbeitsschritte verfügbar. [slu] Für die Ausführung von Arbeitspaketen verwendet man ein Batch-Skript, bzw. ein Shell-Skript. In diesem definiert man Parameter, wie beispielsweise die Anzahl der zu verwendenden Nodes, die Anzahl der zu verwendenden Kerne, Multithreading pro Prozessorkern und die letztendlich auszuführenden Arbeitsschritte. Diese können in Form eines Skripts oder Programms beliebiger Programmersprache hinterlegt werden.

4.4 Software

Für das Umsetzen des quantum-klassisch hybriden Systems wird neben der für Architekturzwecke verwendeten Protokolle anwendungsspezifische Software eingesetzt.

4.4.1 Machine Learning Framework

Wie im Konzept beschrieben gibt es mehrere Frameworks, mit welchen Machine Learning Bibliotheken verwendet werden können. Für ein quantum-klassisches Hybrid spielt die Wahl des Frameworks keine große Rolle, es kann beliebig zwischen den einzelnen wie TensorFlow oder pyTorch gewählt werden. Das letztendliche Framework der Wahl hängt von persönlichen Präferenzen ab und kann dann beliebig mittels eines Skripts auf dem klassischen Teil

des Systems - dem Linux Cluster - ausgeführt werden. In diesem Anwendungsfall wird das Framework pyTorch verwendet.

4.4.2 Quantum Computing Framework

Für die Implementierung von Quantenschaltkreisen bieten sich zwei Möglichkeiten.

Atos Quantum Assembler

Als das Quantum Computing Framework wird das von ATOS entwickelte Framework Atos Quantum Assembler (AQASM) gewählt. [aqa] AQASM ist die von Atos ausgearbeitete Implementation der universellen Quantum Software Sprache Quantum Assembler (QASM). Das Framework kann sowohl auf Quanten Simulatoren wie der Quantum Learning Machine von Atos, als auch auf echten, physikalischen Quanten Computern eingesetzt werden. Mithilfe dieser Sprache ist es möglich, Quanten Schaltkreise zu beschreiben und zu generieren. Neue individualisierte Quanten Schaltkreise können entworfen werden und mit bestehenden Schaltkreisen kombiniert werden. [atoa] Eine Interoperabilität mit anderen Quantum Frameworks wie Cirq, Qiskit oder OpenQASM ist ermöglicht.

Qiskit

Eine weitere verwendete Alternative bietet das bereits angesprochene Quantum Computing Framework Qiskit. Durch die zuvor erwähnte Interoperabilität von Atos Quantum Software Stack myQLM ist es möglich, verschiedene andere Frameworks mit AQASM zu verwenden und die mit einem anderen Framework erstellten Schaltkreise zu übersetzen. Mithilfe eines Binders kann ein ausgearbeiteter Schaltkreis in einen QLM Schaltkreis umgewandelt werden. [qisa]

5 Evaluation

Nachdem die einzelnen Komponenten des quantum-klassischen Hybrids aufgelistet wurden, werden diese nun für einen technischen Versuch verwendet, um eine Praktikabilität der Implementation von hybriden Algorithmen auf dem hybriden System des LRZ zu testen.

5.1 Versuchsaufbau

Im Folgenden wird der Versuchsaufbau mitsamt einzelner Schritte beschrieben.

Schritt 1 - Verbindungsherstellung

Der erste Schritt ist eine Verbindungsherstellung mit dem Linux Cluster des LRZ. Diese Verbindung wird durch das Netzwerkprotokoll SSH ermöglicht. Von der Außenwelt ist eine Verbindung zum Linux Cluster nur über eine Login Node möglich (lxlogin1.lrz.de, ... , lxlogin8.lrz.de). Abhängig von der Login Node erhält man Zugriff auf die verschiedenen in *4.2.1 Linux Cluster* aufgelisteten Systeme und Segmente des Linux Clusters. Von der Command Line eines Rechners kann man nun beispielsweise einen SSH Befehl mit Nutzernamen und Angabe der Login Node ausüben. Mit autorisiertem Benutzernamen erhält man nach Passwort-Authentifizierung Zugang zu einem Teil des Linux Clusters am LRZ.

Schritt 2 - SLURM Batch Skript

Nach erfolgreicher Verbindungsherstellung mit dem Linux Cluster erscheint der Welcome-Screen des Leibniz Supercomputing Centre (LRZ). Über die Kommandozeile hat man Zugriff auf die Nutzerumgebung. Um extensive und berechnungsintensive Programme auf dem Linux Cluster laufen zu lassen, ist es die 'Best Practice', SLURM Batch Skripte für produktive Jobs zu verwenden. Mithilfe des erstellten Skripts wird die Ausführung der darin enthaltenen Arbeitsanweisungen, wie beispielsweise weitere Programme oder Kopierjobs, durch den Workload Manager angestartet und von diesem an die verschiedenen Nodes innerhalb des im Batch Skript definierten Linux Cluster Segments verteilt. Innerhalb des SLURM Batch Skripts werden alle nötigen Informationen definiert, die für den Workload Manager notwendig sind. Neben dem Job Namen und möglichen E-Mail Benachrichtigungen gibt man mit dem Schlüsselbegriff $SBATCH$ weitere technische Angaben wie die Anzahl der zu verwendenden Nodes, die Anzahl der zu startenden Tasks und die Anzahl der zu verwendenden CPU Kernen pro Task an. Weitere Einstellungsmöglichkeiten sind der SLURM Dokumentation zu entnehmen.

Schritt 3 - Berechnung der Arbeitspakete auf klassischem Computer

Das Batch Skript kann verschiedene zu berechnende Unterprogramme enthalten. Ein quantumklassisch hybrider Algorithmus enthält sowohl Teile die auf einem klassischen Part ausgeführt werden, als auch Teile die auf einem Quanten Part ausgeführt werden. Der klassische Teil des hybriden Algorithmus kann in ein oder mehrere Python Skript Dateien abgespeichert werden, deren Ausführung man innerhalb des SLURM Batch Skripts konfiguriert. Die Menge und Reihenfolge der Auflistung, möglicher weiterer Verbindungsherstellung und Abarbeitung von Arbeitspaketen für klassischen Computer und Quanten Computer können je nach Quantum Algorithmus variieren. Die Aufgaben für den klassischen Teil werden, nachdem der SLURM Batch Job gestartet wurde, an das der jeweiligen verwendeten Login Node entsprechende Linux Cluster übermittelt. Ebenfalls werden die Informationen bezüglich definierten Einstellungen wie die Anzahl der zu verwendenden Nodes an das Cluster weitergegeben. Die Anzahl festgelegter Nodes beginnt dann die Berechnungen abzuarbeiten.

Schritt 4 - Kopieren von Daten von klassischem Computer zu Quanten Computer

Für die Ausführung von Quantenalgorithmen, die innerhalb des Batch Skripts an SLURM weitergegeben wurden, müssen diese - wenn nicht schon auf dem Quanten Computer befindlich - auf den Quanten Computer kopiert werden. Dies wird durch das Protokoll SCP ermöglicht. Durch Angabe des Kürzels 'scp' und von Quelle und Zielspeicherort wird - nach einer Passwortabfrage - die gewünschte Datei vom Quellrechner auf den Quanten Computer kopiert. (scp QUANTENALGORITHMUS.py USERNAME@qlm: /)

Schritt 5 - Verbindung zu Quanten Computer und Ausführung von Quantenprogramm

Die Verbindung und Ausführung der Quantenalgorithmen wird durch das Protokoll SSH ermöglicht. Mithilfe des Protokolls ist neben der Verbindung zu einem Zielhost auch die folgende Ausführung von Befehlen möglich. Somit kann ein Pythonskript, welches Quantenalgorithmen enthält, entfernt auf dem Quanten Computer ausgeführt werden. Durch Angabe des Kürzels 'ssh', der Adresse des Servers, auf den verbunden werden soll, dem Befehl 'python' und dem Quellspeicherort des Quantenprogramms wird entfernt die Datei ausgeführt. (ssh qlm 'python QUANTUMALGORITHMUS.py') Die Ergebnisse werden in einer Datei abgespeichert, sodass diese an den klassischen Computer übermittelt werden können und von diesem für weitere Berechnungen verwendet werden können.

Schritt 6 - Kopieren von Daten von Quanten Computer zu klassischem Computer

Solange klassischer Computer und Quanten Computer keinen gemeinsam geteilten Speicher nutzen, müssen nun die in Schritt 5 berechneten Quantenergebnisse an den klassischen Computer übermittelt werden. Dies passiert analog zu Schritt 4 mittels des Protokolls SCP.

Schritt 7 - Berechnung mit Quantendaten auf klassischem Computer

Nach Übermittlung der Daten von Quanten Computer auf klassischen Computer können die in SLURM angegebenen weiteren Programme die Daten für weitere Berechnungen weiterverwenden. Dies erfolgt analog zur Beschreibung in Schritt 3.

Weitere Schritte

Die Schritte 1-7 können beliebig oft und abhängig von der Beschaffenheit des quantum-klassisch hybriden Gesamtalgorithmus wiederholt werden.

5.2 Implementation von Quantengattern und Quantenalgorithmen

Nachdem die Infrastruktur durch die vorangegangenen beschriebenen Schritte gewährleistet wird, können nun durch einfaches Austauschen des jeweiligen Quantenalgorithmus verschiedene Algorithmen getestet werden. Hierbei wird die Korrektheit der Ergebnisse der Algorithmen, sowie die Laufzeit auf dem Quantenrechner und die allgemeine Dauer von Einreichen bis Start eines SLURM-Jobs analysiert.

5.2.1 Anwendung eines Hadamard Gatters

Der erste Test besteht aus der Anwendung eines Hadamard Gatters auf ein einzelnes Qubit. Ein Testprogramm wird auf dem klassischen Computer ausgeführt, im Anschluss wird über SSH auf der QLM ein Quantenprogramm ausgeführt, das einen Schaltkreis mit einem Hadamard Gatter erstellt.

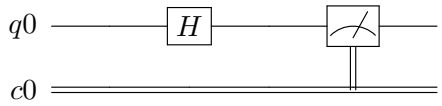

Tabelle 5.1: Schaltkreis mit einfachem Hadamard-Gatter

Das Qubit wird im Zustand $|0\rangle$ initialisiert. Der Schaltkreis ist in Tabelle 5.1 abgebildet. Die Ergebnisse des Quantenprogramms werden für 1024 Versuche gemessen und in Form von Zählungen der einzelnen gemessenen Zustände dokumentiert. Zusätzlich werden die durchschnittliche Laufzeit des Programms auf einer lokalen privaten Maschine (ØAQASM) und im Vergleich dazu die durchschnittliche Laufzeit auf der QLM (ØQLM) in Millisekunden (ms) angegeben.

Ergebnisse Hadamard Gatter mit nbshots = 1024			
Optimales Resultat	Resultat QLM	ØAQASM	ØQLM
$\|0\rangle\,(prob: 0.499)\,\|1\rangle\,(prob: 0.499)$	$\|0\rangle = 513, \|1\rangle = 511$	4.963ms	56.13ms

Tabelle 5.2: Ergebnisse die Anwendung eines Hadamard-Gatters

Die Ergebnisse der Anwendung eines Hadamard Gatters (siehe Tabelle 5.2 zeigen, dass beinahe exakt die Wahrscheinlichkeit von 50% für jeden der beiden Zustände erreicht wurde. Erhöht man die Anzahl der Versuche (nbshots), so wird dies näherungsweise immer genauer.

5.2.2 Verschränkung zweier Qubits

Für den zweiten Test werden zwei Qubits miteinander verschränkt. Der Zustand zweier miteinander verschränkter Qubits wird auch Bell Zustand genannt. Bei diesem Vorgang wird ein Schaltkreis mit zwei Qubits erstellt, bei welchem beide Qubits mit $|0\rangle$ initialisiert werden. Anschließend wird auf das erste Qubit ein Hadamard-Gatter angewandt, und folglich ein CNOT-Gatter, mit dem ersten Qubit als Kontroll-Qubit und dem zweiten als Ziel-Qubit. Der Bell Zustand hat eine 50 prozentige Wahrscheinlichkeit im Zustand $|00\rangle$ gemessen zu werden und eine 50 prozentige Wahrscheinlichkeit im Zustand $|11\rangle$ gemessen zu werden.

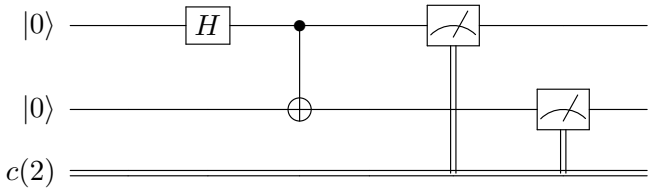

Tabelle 5.3: Schaltkreis für einen Bell Zustand (Verschränkung zweier Qubits)

Der Schaltkreis ist in Tabelle 5.3 abgebildet. Die Ergebnisse des Quantenprogramms werden für 1024 Versuche gemessen und in Form von Zählungen der einzelnen gemessenen Zustände dokumentiert. Zusätzlich werden die durchschnittliche Laufzeit des Programms auf einer lokalen privaten Maschine (ØAQASM) und im Vergleich dazu die durchschnittliche Laufzeit auf der QLM (ØQLM) in ms angegeben.

Verschränkung zweier Qubits mit nbshots = 1024			
Optimales Resultat	Resultat QLM	ØAQASM	ØQLM
$\|00\rangle$ $(prob : 0.499)$ $\|11\rangle$ $(prob : 0.499)$	$\|00\rangle = 512/1024, \|11\rangle = 510/1024$	6.058ms	45.702ms

Tabelle 5.4: Ergebnisse für die Verschränkung zweier Qubits

Ergebnisse für die Verschränkung zweier Qubits (siehe Tabelle 5.4 zeigen eine annähernd 50-prozentige Wahrscheinlichkeit für die beiden Zustände $|00\rangle$ und $|11\rangle$. Die anderen, in der Ergebnistabelle ausgelassenen Ergebnisse treten durch Fehler in der Quantenberechnung auf.

5.2.3 Deutsch-Josza Algorithmus

Als der erste Quantenalgorithmus, der eine bessere Performance als ein klassischer Algorithmus aufweisen konnte, gilt der Deutsch-Jozsa Algorithmus. [DJ92] Mit diesem Algorithmus wird bestimmt, ob eine versteckte boolsche Funktion entweder ausgeglichen oder konstant ist. Als Resultat gibt der Algorithmus eine 0 zurück, für den Fall, dass die Funktion konstant ist, ansonsten ist das Resultat 1. Die Funktion ist balanciert, wenn die Funktion für die Hälfte aller Inputs eine 0 zurück gibt, und für die andere Hälfte eine 1. Gibt die Funktion NUR 0 oder 1 zurück, so ist diese konstant.

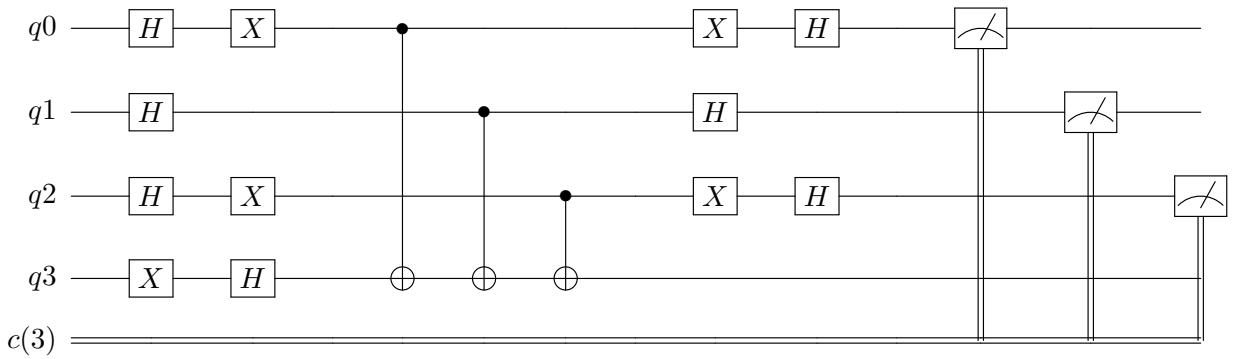

Tabelle 5.5: Schaltkreis für den Deutsch-Josza Algorithmus mit Bitstring $s = 101$

Für einen Bitstring der Länge 3 werden 4 Qubits initialisiert, wir wählen den Bitstring s mit 101, welcher auf die ersten drei Qubits kodiert wird. Es ist ersichtlich, dass der Bitstring nicht konstant ist, sondern näherungsweise ausgeglichen. Die Lösung zu diesem Bitstring sollte eine Messung im Zustand $|111\rangle$ sein, was eine ausgeglichene Funktion indiziert. Der Schaltkreis ist in Tabelle 5.5 abgebildet. Die Ergebnisse des Quantenprogramms werden für 1024 Versuche gemessen und in Form von Zählungen der einzelnen gemessenen Zustände dokumentiert. Zusätzlich werden die durchschnittliche Laufzeit des Programms auf einer lokalen privaten Maschine (ØAQASM) und im Vergleich dazu die durchschnittliche Laufzeit auf der QLM (ØQLM) in ms angegeben. (nach [quab])

Deutsch-Josza Algorithmus mit nbshots = 1024			
Optimales Resultat	Resultat QLM	ØAQASM	ØQLM
$\|111\rangle\,(prob:1)$	$\|111\rangle = 1001/1024$	14.426ms	52.041ms

Tabelle 5.6: Ergebnisse für den Deutsch-Josza Algorithmus

Wie in Tabelle 5.6 zu sehen ist, ist das deutlich am häufigsten berechnete Resultat $|111\rangle$ (Optimallösung, die hier zeigt, dass der Inputstring s ausgeglichen ist und nicht konstant denselben Output liefert, da das Ergebnis der Funktion nicht $|000\rangle$ beträgt). Die anderen, in der Ergebnistabelle ausgelassenen Ergebnisse treten durch Fehler in der Quantenberechnung auf.

5.2.4 Bernstein-Vazirani Algorithmus

Der Bernstein-Vazirani Algorithmus [BV93] kann als eine Erweiterung des Deutsch-Josza Algorithmus gesehen werden. Eine Funktion mit einem Bitstring als Input wird ausgeführt, das Ergebnis ist entweder 0 oder 1. Anders als bei seinem Vorgänger wird bei dem Bernstein-Vazirani Algorithmus der Inputstring mit einem weiteren String s bitweise multipliziert. Ziel des Algorithmus ist es, den String s zu ermitteln. Im Vergleich zum klassischen Algorithmus, der für einen Bitstring der Länge n n Funktionsaufrufe benötigt um zur Lösung zu gelangen, findet der Quantumalgorithmus die Lösung mit nur einem Funktionsaufruf. (nach [quab])

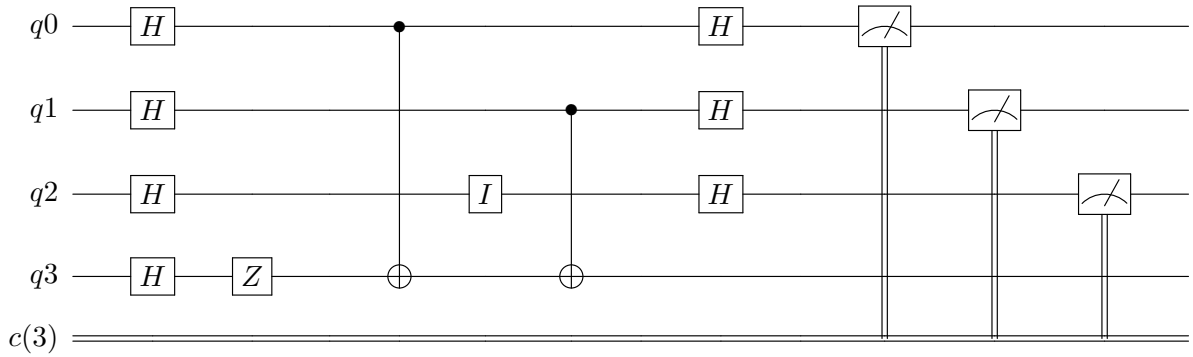

Tabelle 5.7: Schaltkreis für den Bernstein-Vazirani Algorithmus mit Bitstring $s = 011$

Für einen Bitstring der Länge 3 werden 4 Qubits im Zustand $|0\rangle$ initialisiert. Wir wählen den versteckten Bitstring s mit 011. Der Schaltkreis ist in Tabelle 5.7 abgebildet. Die Ergebnisse des Quantenprogramms werden für 1024 Versuche gemessen und in Form von Zählungen der einzelnen gemessenen Zustände dokumentiert. Zusätzlich werden die durchschnittliche Laufzeit des Programms auf einer lokalen privaten Maschine (ØAQASM) und im Vergleich dazu die durchschnittliche Laufzeit auf der QLM (ØQLM) in ms angegeben.

Bernstein-Vazirani Algorithmus mit nbshots = 1024			
Optimales Resultat	Resultat QLM	ØAQASM	ØQLM
$\|011\rangle$ $(prob : 1)$	$\|011\rangle = 1007/1024$	13.333ms	49.727ms

Tabelle 5.8: Ergebnisse für den Bernstein-Vazirani Algorithmus

Durch die Ergebnisse in Tabelle 5.8 lässt sich ableiten, dass der versteckte String s mit 011 in den meisten Fällen durch den Algorithmus erkannt wurde. Die anderen, in der Ergebnistabelle ausgelassenen Ergebnisse treten durch Fehler in der Quantenberechnung auf.

5.2.5 Quantenteleportation

Unter Quantenteleportation versteht man die Übermittlung von Quanteninformationen von einem Sender zu einem weiter entfernten Empfänger. Inhalt der Quanteninformationen ist beispielsweise der Zustand eines Qubits, bei welchem die Amplituden α und β transferiert werden. Durch die Verwendung eines verschränkten Qubit-Paares und zwei klassischen Bits kann der Zustand eines Qubits von Sender zu Empfänger geschickt werden, ohne dass das No-Cloning Theorem - welches in der Quantenmechanik besagt, dass keine Kopien von Quantenzuständen angefertigt werden können - verletzt wird. (nach [quab])

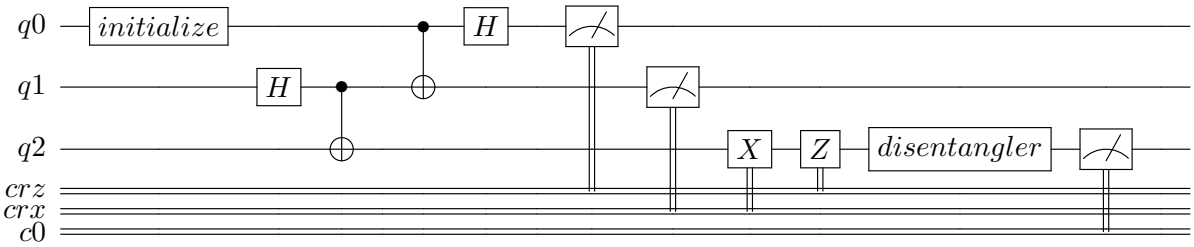

Tabelle 5.9: Schaltkreis für eine Quantenteleportation

Für die Quantenteleporation verwenden wir 3 Qubits und 2 klassische Bits, der Sender besitzt Qubit $q1$ und der Empfänger besitzt Qubit $q2$. Die Qubits werden im Zustand $|0\rangle$ initialisiert. Das Qubit, welches verschickt werden soll ist das Qubit $q0$ und wird in einem Zustand ψ initialisiert. Der Sender, der anfangs die Qubits $q0$ und $q1$ besitzt, misst diese und speichert die Ergebnisse in den beiden klassischen Bits. Diese klassischen Bits werden an den Empfänger übermittelt, der basierend auf den Ergebnissen dieser verschiedene Gatter auf das in seinem Besitz befindliche Qubit $q2$ anwendet. Danach befindet sich das Qubit $q2$ im selben Zustand wie das Qubit $q0$, dessen Zustand somit erfolgreich teleportiert wurde (Der Zustand des ursprünglichen ersten Qubit ist durch die Messung in den festen Zustand $|0\rangle$ oder $|1\rangle$ kollabiert). Der Schaltkreis ist in Tabelle 5.9 abgebildet. Die Ergebnisse des Quantenprogramms werden für 1024 Versuche gemessen und in Form von Zählungen der einzelnen gemessenen Zustände dokumentiert. Zusätzlich werden die durchschnittliche Laufzeit des Programms auf einer lokalen privaten Maschine (ØAQASM) und im Vergleich dazu die durchschnittliche Laufzeit auf der QLM (ØQLM) in ms angegeben. (nach [quab])

Quantenteleportation mit nbshots = 1024			
Optimales Resultat	Resultat QLM	ØAQASM	ØQLM
$\|\_\_0\rangle$ $(prob : 1)$	$\|000\rangle = 240/1024$, $\|010\rangle = 243/1024$, $\|100\rangle = 256/1024$, $\|110\rangle = 267/1024$	41.106ms	85.860ms

Tabelle 5.10: Ergebnisse für die Quantenteleportation

Der Algorithmus erkennt wie in Tabelle 5.10 zu sehen mit den deutlich am höchsten gemessenen Wahrscheinlichkeiten die markierten Zustände $|101\rangle$ und $|110\rangle$. Die anderen, in der Ergebnistabelle ausgelassenen Ergebnisse treten durch Fehler in der Quantenberechnung auf.

Es besteht eine Chance von annähernd 100 Prozent, dass Qubit q2 im Zustand $|0\rangle$ gemessen wird, was das erwartete Resultat ist und zeigt, dass die Quantenteleportation erfolgreich war. Diese Ergebnisse können noch besser visualisiert werden, indem man eine Visualisation über je eine Blochsphäre pro Qubit verwendet.

5.2.6 Grover Algorithmus

Der Grover Algorithmus [Gro96] ist ein Suchalgorithmus für unstrukturierte Datensätze. Mithilfe des Algorithmus ist eine quadratische Beschleunigung der Laufzeit möglich. Allgemein wird davon ausgegangen, dass es in einem Datensatz beliebiger Größe einen oder mehrere Lösungskandidaten gibt, nach denen gesucht werden soll. Um diesen Lösungskandidaten zu finden, muss ein klassischer Algorithmus durchschnittlich die Hälfte aller Einträge des Datensatzes durchforsten, um die Lösung zu finden, in den schlechtesten Fällen findet der klassische Algorithmus diesen sogar erst an der letzten durchsuchten Position. (nach [quab])

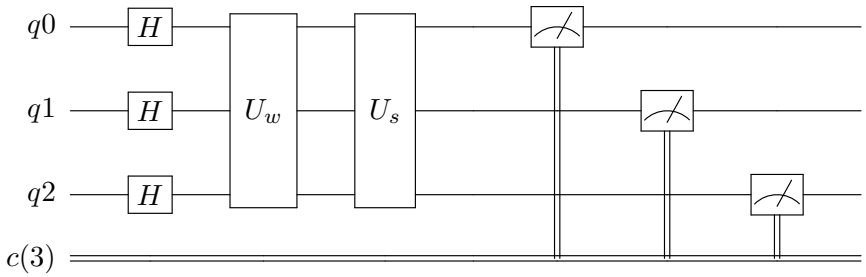

Tabelle 5.11: Schaltkreis für den Grover Algorithmus mit Markierungen der Zustände $|101\rangle$ und $|110\rangle$

Wir erstellen einen Schaltkreis für den Grover Algorithmus mit 3 Qubits, was bedeutet, dass in einem Suchraum von insgesamt 8 verschiedenen Möglichkeiten gesucht werden kann. Es werden zwei verschiedene Zielzustände gewählt, die im Orakel U_w markiert werden. Diese Zustände sind $|101\rangle$ und $|110\rangle$. Der Schaltkreis ist in Tabelle 5.11 abgebildet. Die Ergebnisse des Quantenprogramms werden für 1024 Versuche gemessen und in Form von Zählungen der einzelnen gemessenen Zustände dokumentiert. Zusätzlich werden die durchschnittliche Laufzeit des Programms auf einer lokalen privaten Maschine (ØAQASM) und im Vergleich dazu die durchschnittliche Laufzeit auf der QLM (ØQLM) in ms angegeben.

Grover Algorithmus mit nbshots = 1024			
Optimales Resultat	Resultat QLM	ØAQASM	ØQLM
$\|101\rangle$ ($prob$: 0.499) $\|110\rangle$ ($prob$: 0.499)	$\|101\rangle = 491/1024, \|110\rangle = 480/1024$	20.724ms	60.768ms

Tabelle 5.12: Ergebnisse für den Grover Algorithmus

Der Algorithmus erkennt wie in Tabelle 5.12 zu sehen mit den deutlich am höchsten gemessenen Wahrscheinlichkeiten die markierten Zustände $|101\rangle$ und $|110\rangle$. Die anderen, in der Ergebnistabelle ausgelassenen Ergebnisse treten durch Fehler in der Quantenberechnung auf.

5.2.7 Shors Algorithmus

Der Algorithmus von Peter Shor [Sho96] ist einer der bekanntesten Quantenalgorithmen und wird dazu genutzt, eine Zahl in ihre Primfaktoren zu zerlegen. Das in der Mathematik schon lange bekannte Zerlegen in Primfaktoren wird durch klassische Algorithmen in exponentieller

Laufzeit gelöst. Der Quantumalgorithmus hierzu löst das Problem in polynomialer Laufzeit. Kryptografietechniken wie die Verschlüsselungsmethode RSA basieren auf der schweren Lösbarkeit der Primfaktorzerlegung von großen Zahlen. Noch ist der Shor Algorithmus jedoch keine Gefahr für die Sicherheit dieser Technik, die technischen Voraussetzungen sind noch zu hoch, zu viele Qubits werden für den Algorithmus benötigt. In diesem Beispiel wird ein verwandtes Problem gelöst, das sich mit der Berechnung einer periodischen Funktion beschäftigt. Die Periode einer Funktion $f(x) = a^x \, mod \, N$ ist die kleinste Zahl r, sodass $a^r \, mod \, N = 1$. (nach [quab])

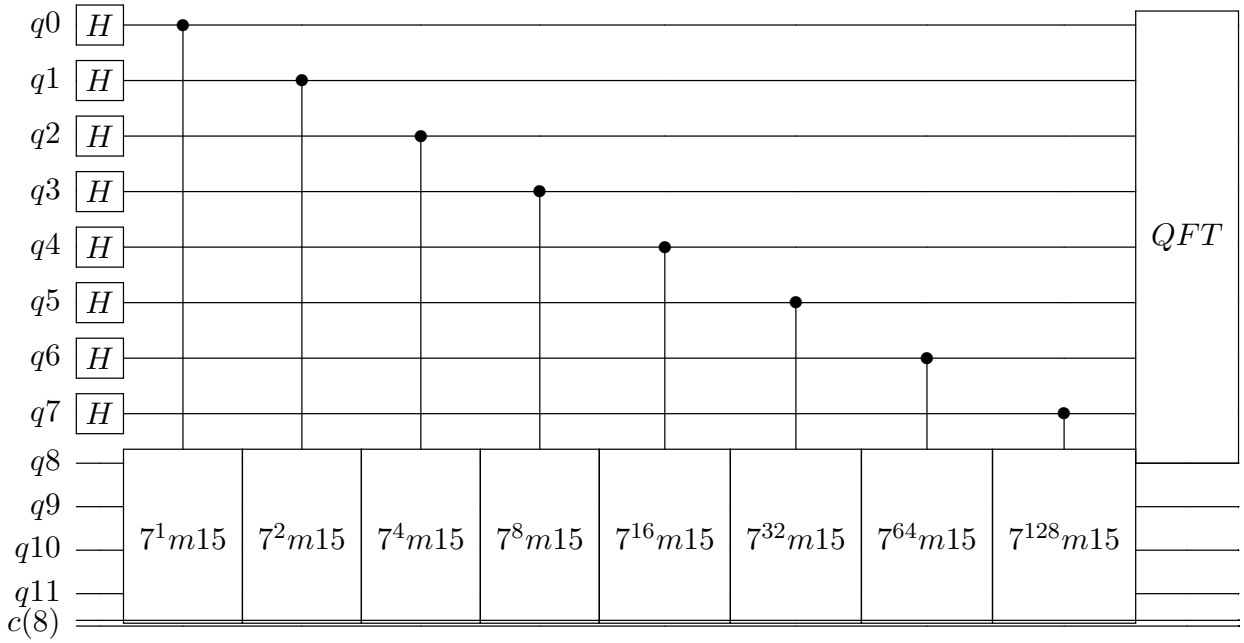

Tabelle 5.13: Schaltkreis für den simplifizierten Shor Algorithmus (Periodenberechnung) mit 8 Qubits + 4 temporelle Qubits, q0-q7 werden im Anschluss auf das QFT-Gatter auf die klassischen Qubits c(8) gemessen. Modulo ist mit m abgekürzt

Im Beispiel werden die Werte $a = 7$ und $N = 15$ gewählt. Es werden 8 Qubits benötigt, weitere 4 Qubits werden für die Applikation der unitären Gatter verwendet. Der Schaltkreis ist in Tabelle 5.13 abgebildet. Für Die Ergebnisse des Quantenprogramms werden für 1024 Versuche gemessen und in Form von Zählungen der einzelnen gemessenen Zustände dokumentiert. Zusätzlich werden die durchschnittliche Laufzeit des Programms auf einer lokalen privaten Maschine (ØAQASM) und im Vergleich dazu die durchschnittliche Laufzeit auf der QLM (ØQLM) in ms angegeben.

Shor Algorithmus mit nbshots = 1024			
Optimales Resultat	Resultat QLM	ØAQASM	ØQLM
$\|01000000\rangle$ $(prob : 0.499)$	$\|00000000\rangle = 252/1024$		
$\|11000000\rangle$ $(prob : 0.499)$	$\|01000000\rangle = 251/1024$		
	$\|10000000\rangle = 262/1024$		
	$\|11000000\rangle = 259/1024$	10920.949ms	18679.163ms

Tabelle 5.14: Ergebnisse für den simplifizierten Shor Algorithmus

Für die Berechnung des Schaltkreises für den vereinfachten Shor Algorithmus wurde ohne simuliertes Quantengatterrauschen gerechnet. Durch das Rauschen und die Tiefe des Schaltkreises konnten keine verwendbaren Ergebnisse extrahiert werden. Zusätzlich hat die Berechnung des Schaltkreises mit Rauschen sechs mal so lang gedauert, wie ohne das simulierte Rauschen. Die Messungen ergeben verschiedene Zustände und korrespondieren mit den gemessenen Phasen der Qubits (siehe 5.14. Die Resultate in Bitform werden umgewandelt in Ihre Dezimalrepräsentation, beispielsweise ergibt 01000000 = 64 in Dezimalrepräsentation. Diese Zahl wird nun durch die maximal darstellbare Zahl der Anzahl an Qubits geteilt, mit acht Qubits ergeben sich 256 Zustände, also rechnet man in diesem Fall 64/256. Dieser Bruch kann in die kleinsten Denominator runtergebrochen werden, daraus folgt 1/4. Der kleinste Denominator bringt die korrekte Lösung für r mit $r = 4$. Jedoch besteht offensichtlich das Risiko von fehlerhaften Berechnungen, welches zwei der vier Ergebnisse erklärt. Dies ist jedoch nicht nur QLM-spezifisch.

5.2.8 Weitere Ergebnisse

Die durchschnittliche Wartezeit vom Einreichen bis zum Start der SLURM-Jobs mit den verschiedenen Algorithmen beträgt über je 10 Durchläufe pro Algorithmus nur etwa eine Sekunde. Hierbei hat der Algorithmus selbst keine Auswirkung auf die Wartezeit. Die Laufzeit der SLURM-Jobs ist abhängig von der Menge an klassischen Algorithmen und Quantenalgorithmen, sowie deren Komplexität. Die auf der QLM berechneten Ergebnisse wurden mit einem simulierten Rauschen berechnet. Ausführungen der Schaltkreise ohne dieses simulierte Rauschen ergaben jeweils immer - mit Ausnahme des Shor Algorithmus - die Optimallösung, was die Korrektheit der Funktionsweise der QLM beweist.

6 Zusammenfassung und Ausblick

In dieser Arbeit wurde erfolgreich ein System bestehend aus dem Linux Cluster am Leibniz-Rechenzentrum und der Quantum Learning Machine von ATOS zur Ausführung von quantum-klassisch hybriden Algorithmen eingesetzt. Der hierfür erstellte Workflow initialisiert mittels eines Batch-Skripts einen Job auf dem Rechnerverbund, der klassische Teile eines Algorithmus auf einer festgelegten Anzahl von Cluster-Nodes berechnen lässt. Durch eine Secure Shell Verbindung vom Linux Cluster zur QLM und das Kopieren von Anweisungen beziehungsweise Quantenschaltkreisen auf die QLM können Quantum Berechnungen durchgeführt werden. Die Ergebnisse der Simulation der Schaltkreise können dann zurück auf den Linux Cluster kopiert werden, wo es den Prozessoren des Clusters möglich ist, mit diesen Quanten Parts weiterzuarbeiten. Diese Schritte können je nach hybridem Algorithmus beliebig oft und in beliebiger Reihenfolge abgearbeitet werden.
Der Hauptnutzen in diesem Hybrid liegt in der parallelen Arbeitsweise des Quantencomputers, durch den verschiedene rechenintensive Schritte deutlich schneller bearbeitet werden können. Allgemein soll also die Geschwindigkeit der Berechnung maximiert werden. Diese hängt stark von zwei Faktoren ab: Wie schnell können Informationen von A nach B gelangen, also allgemeiner der Datentransfer; und wie schnell können diese transferierten Informationen verarbeitet werden, also die Datenberechnung. In dem in dieser Arbeit aufgezeigten System ist deutlich, dass der Datentransfer zwischen den beiden Komponenten - klassisch und quantum - einen Flaschenhals bildet, an welchem einiges an Geschwindigkeit hinsichtlich Berechnungen verloren geht. Durch die von Linux Cluster und QLM gemeinsame Nutzung eines Speichers werden die Kopiervorgänge obsolet und das Problem hinsichtlich eines Flaschenhalses beim Datentransfer gelöst.

6.1 Fazit

Für die Evaluation des Systems wurden erfolgreich verschiedene Quantenalgorithmen ausgeführt und deren Ergebnisse an einen klassischen Algorithmus weitergegeben. Hierbei wurde gezeigt, dass die QLM korrekte Ergebnisse liefert und somit einen validen Bestandteil in einem quantum-klassisch hybriden System in der Umgebung des LRZ darstellt. Auffällig sind die deutlich höheren Laufzeiten auf der QLM im Vergleich zu Berechnungen mit dem AQASM Framework auf einem privaten Testrechner. Diese Probleme gilt es auch in Zukunft zu lösen. Ein CNN selbst wurde im Rahmen dieser Arbeit nicht implementiert.

6.2 Zukünftige Arbeit

Die hier implementierte Schnittstelle kann für eine erfolgreiche Nutzung des hybriden Systems zwischen Linux Rechencluster und der QLM verwendet werden. Es ist nun möglich, verschiedene quantum-klassische Algorithmen einzusetzen und weitere hybride Algorithmen

zu entwickeln. Das verwendete AQASM Format ermöglicht sowohl eine Interoperabilität mit Quantum Computing Frameworks wie Qiskit, oder gängigen Machine Learning Frameworks wie TensorFlow, als auch eine Ausführung des Programmcodes auf Quantencomputern mit echten Qubits. Somit können mittels dieser Schnittstelle in Zukunft anspruchsvolle Quantum Machine Learning Algorithmen für verschiedene Anwendungsfälle wie Bildklassifizierung eingesetzt werden, ob, wie in dieser Arbeit auf einem System mit der QLM mit bis zu 30 simulierten Qubits, oder mit weiteren Quantencomputern mit realen, physischen Qubits. Für diesen Fall muss ein jeweiliges Quantensystem analog in das bestehende Hybrid eingebaut werden. Mögliche Verbesserungen der Arbeit bestehen unter anderem aus der Optimierung der Schnittstelle und des Workloadmanagements, und dem Erstellen von nützlichen hybriden Algorithmen, wie beispielsweise die Entwicklung und Nutzung von CNNs für Aufgaben wie die Bild- und Objekterkennung.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AQASM Atos Quantum Assembler

MB Megabyte

ms Millisekunden

GB Gigabyte

TB Terabyte

LRZ Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften

CNN Convolutional Neural Networks

NN (Künstliche) Neuronale Netze

MWN Münchner Wissenschaftsnetz

QLM Quantum Learning Machine des französischen Herstellers Atos

QML Quantum Machine Learning

ML Machine Learning

NISQ Noisy Intermediate Scale Quantum

SSH Secure Shell

SCP Secure Copy

SLURM Simple Linux Utility for Resource Management

QC Quantum Computing oder auch Quantencomputing

Qubit Quantumbit oder auch Quantenbit

MNIST Modified National Institute of Standards and Technology

TFQ TensorFlow Quantum

Literaturverzeichnis

[3Bl] 3Blue1Brown: *Aber was *ist* nun ein neuronales Netzwerk? | Teil 1, Deep Learning.* https://www.youtube.com/watch?v=aircAruvnKk&ab_channel=3Blue1Brown

[ABC+21] Altman, Ehud ; Brown, Kenneth R. ; Carleo, Giuseppe ; Carr, Lincoln D. ; Demler, Eugene ; Chin, Cheng ; DeMarco, Brian ; Economou, Sophia E. ; Eriksson, Mark A. ; Fu, Kai-Mei C. u. a.: Quantum simulators: Architectures and opportunities. In: *PRX Quantum* 2 (2021), Nr. 1, S. 017003

[aqa] *The AQASM format.* https://myqlm.github.io/aqasm.html, . – Accessed: 2022-19-4

[atoa] *Atos Brochure Quantum Learning Machine.* https://atos.net/wp-content/uploads/2019/11/Brochure_Quantum_Learning_Machine_Atos.pdf, . – Accessed: 2022-15-07

[atob] *Atos Delivers 'Quantum Learning Machine' to Leibniz Supercomputing Centre (LRZ).* https://www.hpcwire.com/off-the-wire/atos-supports-the-leibniz-supercomputing-centre-with-the-atos-qlm/, . – Accessed: 2022-15-07

[ATOc] *ATOS Quantum Learning Machine.* https://atos.net/en/solutions/quantum-learning-machine, . – Accessed: 2021-11-19

[atod] *Atos Quantum Learning Machine.* https://atos.net/de/2017/pressemeldungen_2017_11_14/atos-quantum-learning-machine-unterstuetzt-forschungen-des-u-s-energieministeriums, . – Accessed: 2022-15-07

[atoe] *Kooperation Atos und LRZ.* https://atos.net/de/2021/pressemeldungen_2021_03_18/atos-unterstuetzt-das-leibniz-rechenzentrum-mit-der-atos-qlm-beim-vorantr . – Accessed: 2022-07-22

[BV93] Bernstein, Ethan ; Vazirani, Umesh: Quantum complexity theory. In: *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, 1993, S. 11–20

[BVM+20] Broughton, Michael ; Verdon, Guillaume ; McCourt, Trevor ; Martinez, Antonio J. ; Yoo, Jae H. ; Isakov, Sergei V. ; Massey, Philip ; Halavati, Ramin ; Niu, Murphy Y. ; Zlokapa, Alexander u. a.: Tensorflow quantum: A software framework for quantum machine learning. In: *arXiv preprint arXiv:2003.02989* (2020)

[CB18] Cortese, John A. ; Braje, Timothy M.: Loading classical data into a quantum computer. In: *arXiv preprint arXiv:1803.01958* (2018)

[Chea] *25 Jahre "Deep Blue schlägt Kasparov“*. https://de.chessbase.com/post/25-jahre-deep-blue-schlaegt-kasparov, . – Accessed: 2022-1-17

[cheb] *Quanten Mechanik erklärt*. https://www.chemie.de/lexikon/Quantenmechanik.html, . – Accessed: 2022-20-1

[cira] *Cirq on GitHub*. https://github.com/quantumlib/Cirq, . – Accessed: 2022-19-4

[cirb] *Quantum Framework Cirq*. https://quantumai.google/cirq, . – Accessed: 2022-19-4

[cooa] *CoolMUC2*. https://doku.lrz.de/display/PUBLIC/CoolMUC-2, . – Accessed: 2022-15-07

[coob] *CoolMUC3*. https://doku.lrz.de/display/PUBLIC/CoolMUC-3, . – Accessed: 2022-15-07

[Den12] Deng, Li: The mnist database of handwritten digit images for machine learning research. In: *IEEE Signal Processing Magazine* 29 (2012), Nr. 6, S. 141–142

[DJ92] Deutsch, David ; Jozsa, Richard: Rapid solution of problems by quantum computation. In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 439 (1992), Nr. 1907, S. 553–558

[Fey82] Feynman, R.P.: Simulating physics with computers. In: *Int J Theor Phys* (1982), Nr. 21, 467–488. https://doi.org/10.1007/BF02650179

[Fra] *Maschinelles Lernen: Kompetenzen, Forschung, Anwendung*. https://www.bigdata-ai.fraunhofer.de/de/publikationen/ml-studie.html, . – Accessed: 2022-1-25

[FSZ22] Fan, Fan ; Shi, Yilei ; Zhu, Xiao X.: Earth Observation Data Classification with Quantum Classical Convolutional Neural Network. In: *IGARSS 2022*, 2022

[Gar] *Gartner Hype Cycle 2021*. https://www.gartner.de/de/artikel/3-themen-im-hype-cycle-fuer-emerging-technologies-2021, . – Accessed: 2022-07-22

[GIS] *Remote Sensing - The definitive guide*. https://gisgeography.com/remote-sensing-earth-observation-guide/, . – Accessed: 2022-01-10

[goo] *Google search statistics*. https://www.oberlo.com/blog/google-search-statistics, . – Accessed: 2022-04-09

[Gro96] Grover, Lov K.: A fast quantum mechanical algorithm for database search. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, S. 212–219

[Hom05] Homeister, Matthias: *Quantum Computing verstehen - Grundlagen, Anwendungen, Perspektiven.* Vieweg, 2005 (Computational intelligence). – I–XI, 1–299 S. – ISBN 978–3–528–05921–7

[HPCa] *HPC am LRZ.* https://www.tum.de/die-tum/aktuelles/pressemitteilungen/details/34959, . – Accessed: 2022-07-22

[HPCb] *Schnellster Supercomputer Deutschlands.* https://doku.lrz.de/display/PUBLIC/High+Performance+Computing, . – Accessed: 2022-07-22

[HSPC20] Henderson, Maxwell ; Shakya, Samriddhi ; Pradhan, Shashindra ; Cook, Tristan: Quanvolutional neural networks: powering image recognition with quantum circuits. In: *Quantum Machine Intelligence* 2 (2020), Nr. 1, S. 1–9

[hwr] *Hardware recommendations for machine learning projects.* https://www.cherryservers.com/blog/how-to-choose-hardware-for-your-machine-learning-project, . – Accessed: 2022-19-4

[ibma] *IBM Quantum computing roadmap.* https://research.ibm.com/blog/quantum-development-roadmap, . – Accessed: 2022-19-4

[ibmb] *IBM simulates a 56-Qubit Machine.* https://spectrum.ieee.org/ibms-quantum-leap-simulates-56qubit-machine, . – Accessed: 2022-19-4

[INN+21] Ibrahim, Khaled Z. ; Nguyen, Tan ; Nam, Hai A. ; Bhimji, Wahid ; Farrell, Steven ; Oliker, Leonid ; Rowan, Michael ; Wright, Nicholas J. ; Williams, Samuel: Architectural Requirements for Deep Learning Workloads in HPC Environments. In: *2021 International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)* IEEE, 2021, S. 7–17

[ion] *IonQ Quantum Computer.* https://www.forbes.com/sites/moorinsights/2021/09/14/ionq-delivers-a-new-architecture-and-glass-quantum-computer-chip-a-year-e?sh=4e5329155dda, . – Accessed: 2022-19-4

[KAVL19] Khan, Sumsam U. ; Awan, Ahsan J. ; Vall-Llosera, Gemma: K-means clustering on noisy intermediate scale quantum computers. In: *arXiv preprint arXiv:1909.12183* (2019)

[ker] *Keras open source DL framework.* https://keras.io/, . – Accessed: 2022-19-4

[Liu11] Liu, Bing: *Web data mining exploring hyperlinks, contents, and usage data.* Berlin : Springer, 2011 http://dx.doi.org/10.1007/978-3-642-19460-3. – ISBN 9783642194603 3642194605

[LLZC15] Li, Peilong ; Luo, Yan ; Zhang, Ning ; Cao, Yu: Heterospark: A heterogeneous cpu/gpu spark platform for machine learning algorithms. In: *2015 IEEE International Conference on Networking, Architecture and Storage (NAS)* IEEE, 2015, S. 347–348

[log] *Logikgatter*. https://de.wikipedia.org/wiki/Logikgatter, . – Accessed: 2022-02-08

[lrza] *Aufgaben des LRZ*. https://badw.de/die-akademie/satzungen-und-statuten/geschaeftsordnung-fuer-das-leibniz-rechenzentrum-lrz.html, . – Accessed: 2022-19-4

[lrzb] *Hoch- und Höchstleistungsrechner am LRZ*. https://doku.lrz.de/pages/viewpage.action?pageId=64815239, . – Accessed: 2022-19-4

[LRZc] *Linux Cluster am LRZ*. https://doku.lrz.de/display/PUBLIC/Linux+Cluster, . – Accessed: 2021-12-12

[lrzd] *Linux Cluster Segmente am LRZ*. https://doku.lrz.de/display/PUBLIC/Linux+Cluster+Segments, . – Accessed: 2022-02-08

[LZX+20] Li, YaoChong ; Zhou, Ri-Gui ; Xu, RuQing ; Luo, Jia ; Hu, WenWen: A quantum deep convolutional neural network for image recognition. In: *Quantum Science and Technology* 5 (2020), Nr. 4, S. 044003

[McM07] McMahon, David: *Quantum computing explained*. John Wiley & Sons, 2007

[MWN] *MWN Überblick*. https://www.lrz.de/services/netz/mwn-ueberblick/, . – Accessed: 2022-07-22

[NC16] Nielsen, Michael A. ; Chuang, Isaac L.: *Quantum Computation and Quantum Information (10th Anniversary edition)*. Cambridge University Press, 2016. – I–XXXI, 1–676 S. – ISBN 978–1–10–700217–3

[Nie18] Nielsen, Michael A.: *Neural Networks and Deep Learning*. Version: 2018. http://neuralnetworksanddeeplearning.com/

[ON15] O'Shea, Keiron ; Nash, Ryan: *An Introduction to Convolutional Neural Networks*. http://dx.doi.org/10.48550/ARXIV.1511.08458. Version: 2015

[pen] *PennyLane is a cross-platform Python library for differentiable programming of quantum computers. Train a quantum computer the same way as a neural network*. https://pennylane.readthedocs.io/en/stable/, . – Accessed: 2022-19-4

[pyt] *PyTorch open source ML framework*. https://pytorch.org/, . – Accessed: 2022-19-4

[qisa] *MyQLM Interoperabilität mit Qiskit*. https://myqlm.github.io/myqlm_specific/qiskit.html#module-qat.interop.qiskit.converters, . – Accessed: 2022-15-07

[qisb] *Qiskit*. https://qiskit.org/, . – Accessed: 2022-19-4

[Qisc] *Quanten Machine Learning with Qiskit*. https://qiskit.org/textbook/ch-machine-learning/machine-learning-qiskit-pytorch.html, . – Accessed: 2022-20-1

[quaa] *Applications for Quantum Computers.* https://blog.paperspace.com/beginners-guide-to-quantum-machine-learning/, . – Accessed: 2022-08-08

[quab] *Quantum Algorithms for the NISQ era.* https://qiskit.org/textbook/ch-algorithms/index.html, . – Accessed: 2022-08-08

[quac] *Quantum logic gate.* https://en.wikipedia.org/wiki/Quantum_logic_gate, . – Accessed: 2022-02-08

[RML14] Rebentrost, Patrick ; Mohseni, Masoud ; Lloyd, Seth: Quantum support vector machine for big data classification. In: *Physical review letters* 113 (2014), Nr. 13, S. 130503

[RN10] Russell, Stuart ; Norvig, Peter: *Artificial Intelligence: A Modern Approach.* 3. Prentice Hall, 2010

[Sch15] Schlegel, Daniel: Deep machine learning on Gpu. In: *University of Heidelberg-Ziti* 12 (2015)

[scpa] *Secure Copy.* https://en.wikipedia.org/wiki/Secure_copy_protocol, . – Accessed: 2022-15-07

[scpb] *Secure Copy Manual.* https://linux.die.net/man/1/scp, . – Accessed: 2022-15-07

[SHH21] Srikumar, Maiyuren ; Hill, Charles D. ; Hollenberg, Lloyd C. L.: *Clustering and enhanced classification using a hybrid quantum autoencoder.* http://dx.doi.org/10.48550/ARXIV.2107.11988. Version: 2021

[Sho96] Shor, Peter W.: Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. In: *SIAM J.Sci.Statist.Comput. 26* (1996), Nr. 2

[slu] *SLURM Workload Manager.* https://slurm.schedmd.com/overview.html, . – Accessed: 2022-15-07

[ssh] *Secure Shell.* https://www.ssh.com/academy/ssh/command, . – Accessed: 2022-15-07

[str] *Strawberry fields, a cross-platform Python library for simulating and executing programs on quantum photonic hardware.* https://strawberryfields.ai/, . – Accessed: 2022-19-4

[sum] *Summit and Sierra Supercomputers.* https://www.ibm.com/thought-leadership/summit-supercomputer/, . – Accessed: 2022-19-4

[ten] *TensorFlow open source ML framework.* https://www.tensorflow.org/, . – Accessed: 2022-19-4

[ter] *Large Memory Teramem.* https://doku.lrz.de/display/PUBLIC/Large+Memory+Teramem, . – Accessed: 2022-15-07

[tfq] *TensorFlow Quantum - Framework für hybrides quantenklassisches ML*. https://www.tensorflow.org/quantum, . – Accessed: 2022-19-4

[TRN+21] Tüysüz, Cenk ; Rieger, Carla ; Novotny, Kristiane ; Demirköz, Bilge ; Dobos, Daniel ; Potamianos, Karolos ; Vallecorsa, Sofia ; Vlimant, Jean-Roch ; Forster, Richard: Hybrid quantum classical graph neural networks for particle track reconstruction. In: *Quantum Machine Intelligence* 3 (2021), Nr. 2, S. 1–20

[USG] *Remote Sensing - What is it, and what is it used for?* https://www.usgs.gov/faqs/what-remote-sensing-and-what-it-used, . – Accessed: 2022-01-10

[Wir] *Big Data oder: Viel hilft viel?!* https://www.wirtschaftsforum.de/expertenwissen/was-ist-denn-data-science/big-data-oder-viel-hilft-viel, . – Accessed: 2022-12-1

[xan] *Xanadu Photonic Quantum Computer*. https://www.newswire.ca/news-releases/xanadu-releases-world-s-first-photonic-quantum-computer-in-the-c
html, . – Accessed: 2022-19-4

[YNDT18] Yamashita, Rikiya ; Nishio, Mizuho ; Do, Richard Kinh G. ; Togashi, Kaori: *Convolutional neural networks: an overview and application in radiology*. http://dx.doi.org/10.1007/s13244-018-0639-9. Version: 2018

[ZG21] Zhao, Chen ; Gao, Xiao-Shan: Qdnn: deep neural networks with quantum layers. In: *Quantum Machine Intelligence* 3 (2021), Nr. 1, S. 1–9

[ZN20] Zhang, Yao ; Ni, Qiang: Recent advances in quantum machine learning. In: *Quantum Engineering* 2 (2020), Nr. 1, e34. http://dx.doi.org/https://doi.org/10.1002/que2.34. – DOI https://doi.org/10.1002/que2.34