DHBW Stuttgart

AK HAW

Quantum Computing Algorithmen und Programmierung für Anwendungen

fpt e.V.
Fachbereichstag Physikalische Technologien

Prof. Dr. Gerhard Hellstern | DHBW Stuttgart – Center of Finance

Referent: Prof. Dr. Gerhard Hellstern

- Seit 2018: Professor an der Dualen Hochschule Baden-Württemberg in Stuttgart, Fakultät Wirtschaft, Center of Finance
- 1998 Promotion in Theoretische Kernphysik an der Uni Tübingen
- 20 Jahre Erfahrung in der Finanzindustrie:
  Deutsche Bank, DZ Bank, Deutsche Bundesbank
- **Forschung**: Anwendung von Quantum Computing: v.a. Quantum Machine Learning, Optimierung
  Education von „Quanten für Nicht-Physiker“

Überblick

1. Quantencomputing für die Praxis
2. Vom Schaltkreismodell zu Algorithmen
3. Programmierung eines Quantencomputer
4. Optimierung mittels Quantencomputer
5. Optional: Quanten Machine Learning
6. Ressourcen: Literatur & MooCs

1. QUANTENCOMPUTING FÜR DIE PRAXIS

QUANTENCOMPUTER

Wenn komplexe Aufgaben in 3:20 Minuten statt 10.000 Jahren gelöst werden

Auf dem Forschungsfeld der Quantencomputer könnte Google ein Durchbruch gelungen sein. Experten erwarten einen Schub für die Zukunftstechnologie.

Quantencomputer „Q System One"
Amerikanische Unternehmen wie IBM forschen intensiv an der Technologie.
(Foto: dpa)

Düsseldorf, San Francisco. Der große Durchbruch oder die große

„Quantum Supremacy" oder „Quantum Advantage"

NewScientist

Sign in | Enter search keywords

News Features Newsletters Podcasts Video Comment Culture Crosswords | This week's magazine

Health Space Physics Technology Environment Mind Humans Life Mathematics Chemistry Earth Society

Technology

Google's claim of quantum supremacy has been completely smashed

Google's Sycamore quantum computer was the first to demonstrate quantum supremacy – solving calculations that would be unfeasible on a classical computer – but now ordinary machines have pulled ahead again

By Matthew Sparkes

3 July 2024

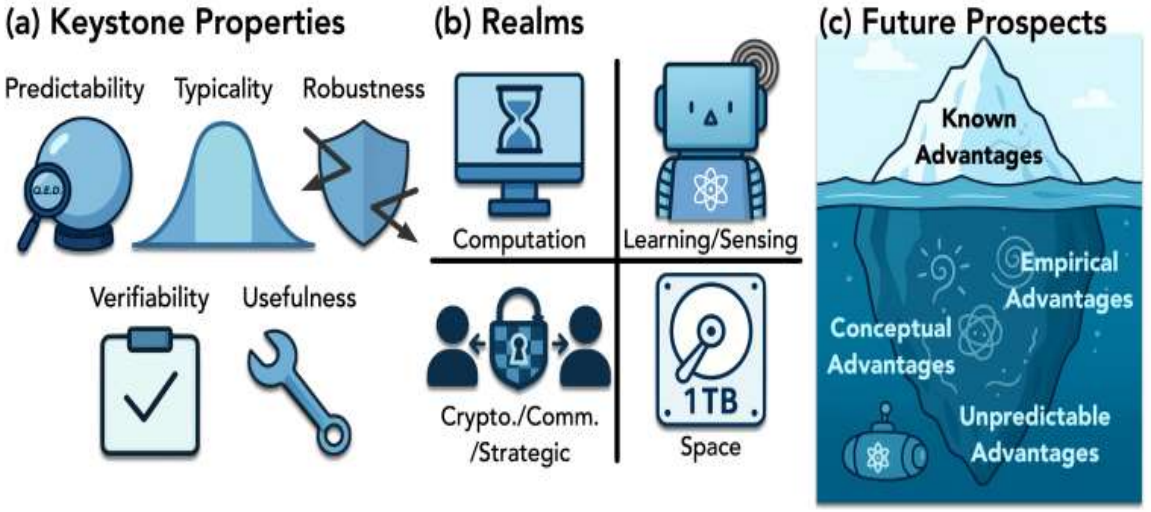
The vast world of quantum advantage

Hsin-Yuan Huang,[1,2] Soonwon Choi,[3] Jarrod R. McClean,[2] and John Preskill[1,4]

[1] *California Institute of Technology, Pasadena, California 91125, USA*
[2] *Google Quantum AI, Venice, California 90291, USA*
[3] *Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA*
[4] *AWS Center for Quantum Computing, Pasadena, California 91125, USA*

(Dated: August 14, 2025)

The quest to identify quantum advantages lies at the heart of quantum technology. While quantum devices promise extraordinary capabilities, from exponential computational speedups to unprecedented measurement precision, distinguishing genuine advantages from mere illusions remains a formidable challenge. In this endeavor, quantum theorists are like prophets attempting to foretell the future, yet the boundary between visionary insight and unfounded fantasy is perilously thin. In this perspective, we examine our mathematical tools for navigating the vast world of quantum advantages across computation, learning, sensing, and communication. We explore five keystone properties: predictability, typicality, robustness, verifiability, and usefulness that define an ideal quantum advantage, and envision what new quantum advantages could arise uitous quantum technology. We prove that some quantum advantages are in using classical resources alone, suggesting a landscape far richer than what see. While mathematical rigor remains our indispensable guide, the ultima technologies may emerge from advantages we cannot yet conceive.



Beweis, dass bestimmte Adavantage-Fragen einen funktionierenden Quantencomputer erfordern!

Standortbestimmung ….

- **Hype** um Quantencomputing seit ca. 7-8 Jahren; obwohl schon seit vielen Jahren daran gearbeitet wird
- Googles Supremacy-Debatte hat das Thema breit bekannt gemacht
- Neben Google sind auch Microsoft, IBM, Baidu sowie eine ganze Reihe von Spezialunternehmen (z.B. D-Wave oder Rigetti) involviert
- Unternehmen aus den Bereichen Mobilität (Daimler, VW, BMW, ..), Logistik. Pharmazie und Materialforschung (BASF, …) sowie Finance (Commerzbank, Barclays …) beschäftigen sich aktiv mit dem „Thema“
- Quantum Computing als Tool für Solid State Physik, Gittereichtheorien, etc.

…. und Thesen daraus

- Quantencomputing **KÖNNTE** bereits in wenigen Jahren „klassische“ Computer bei bestimmten Aufgaben ablösen
- Aus theoretischer Sicht (→ Komplexitätstheorie) lässt sich für bestimmte Problemtypen ein sog. „Speed-Up“ ggü klassischen Algorithmen erreichen:
  - Bestimmte Optimierungs-Probleme
  - Bestimmte Simulations-Probleme
  - Bestimmte Machine Learning-Probleme

Welche genau ist Thema aktueller Forschung…

Type of scaling	Time to solve problem				
Classical algorithm with exponential runtime	10 secs	2 mins	330 years	3300 years	Age of the universe
Quantum algorithm with polynomial runtime	1 min	2 mins	10 mins	11 mins	~24 mins

Andere „Hoffnungen“ sind höhere Genauigkeit etc.

…. und Thesen daraus

- Schwierigkeit = Reiz: Programmierung von Quanten-Computer ist „ganz anders“ ☺
  - Man beginnt (derzeit) „ganz unten“ im Software-Stack → Qubits statt Bits
  - Ausgehend von Qubits muss der Stack grds. neu definiert werden

→ Bedarf an Fachkräften unterschiedlicher Qualifikation (Physiker, Ingenieure, Informatiker, ..)

→ Die auch die Anwendungsseite verstehen und die Kombination „Klassisches Computing & Quantencomputing“ beherrschen

→ Hoher Bedarf an „Wissen in den Unternehmen“ über dieses Thema

Wer beschäftigt sich mit Quantencomputing?

2. VOM SCHALTKREISMODELL ZU ALGORITHMEN

Das Schaltkreismodell

- Quantencomputing: Basisbausteine sind **Quantenbits** oder **Qubits**

- Darstellung: $|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}$ und $|1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$ als „Basiszustände“

  bzw. $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix}\alpha\\\beta\end{pmatrix}$ als allgemeiner „Zustand“

  mit $\alpha, \beta \in \mathbb{C}$ und der Bedingung $|\alpha|^2 + |\beta|^2 = 1$

  → **Superposition der beiden Basiszustände mit ihren jeweiligen Wahrscheinlichkeiten $|\alpha|^2$ und $|\beta|^2$**

- Dirac‘sche „Bra - Ket-Schreibweise“ als Abkürzung (!!) für Vektoren:

  $|\psi\rangle = \begin{pmatrix}\alpha\\\beta\end{pmatrix}$ „ket“ und $\langle\psi| = (\alpha^*, \beta^*)$ „bra“ mit $\langle\psi|\psi\rangle = 1$

Sog. Qudits als Erweiterung → drei oder mehr Basiszustände

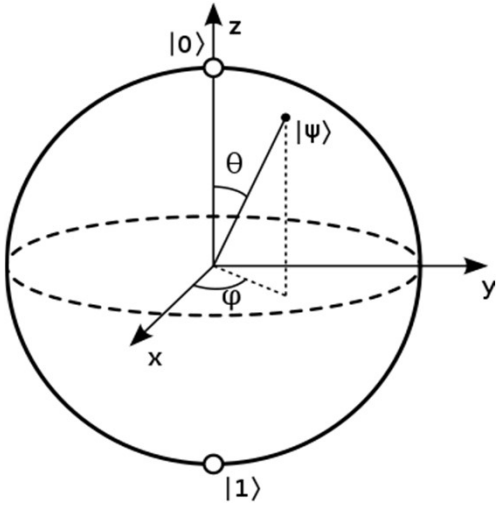
Das Schaltkreismodell

- Visualisierung eines Qubits über die sog. Blochkugel:
  mit der alternativen Parametrisierung

$$|\psi\rangle = \cos(\theta)\,|0\rangle + e^{i\varphi}\sin(\theta)\,|1\rangle$$

- **Achtung**: Auf der Blochkugel stehen die Basiszustände $|0\rangle$ und $|1\rangle$ nicht senkrecht aufeinander, sondern zeigen in unterschiedliche Richtungen!!

- Blochkugel visualisiert die beiden Freiheitsgrade eines Qubits, hier: φ und θ.



- Note: Blochkugel hilft Intuition für ein Qubit zu entwickeln;
  leider gibt es kein schönes Äquivalent für zwei oder mehr Qubits ☹

Abbildung: https://commons.wikimedia.org/wiki/File:Bloch_sphere.svg

Das Schaltkreismodell

- *Veränderungen an einem Qubit entsprechen dann „Drehungen des Vektors auf der Blochkugel“.*

- Drehungen von Vektoren werden in der Mathematik durch unitäre Matrizen beschrieben.

- Beispiel:
  - „Drehe einen Vektor, der zum Nordpol zeigt in einen Vektor, der zum Südpol zeigt“, d.h. $|0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \rightarrow |1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$

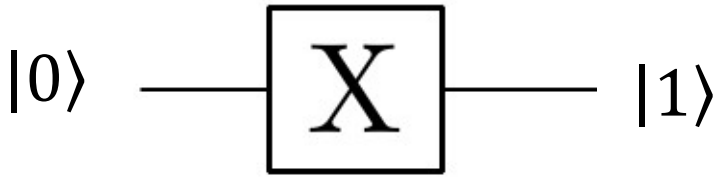
    Dies wird mit der Matrix $X = \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$ erreicht: $\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix} \cdot \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix}$

- Analogie: AND, OR, XOR, …. Gatter

Das Schaltkreismodell

- Matrix $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ wird „X-Gatter“ (engl. Gate) oder „Not-Gatter“ genannt
  - Kompaktere Schreibweise: $X|j\rangle = |j \oplus 1\rangle$ mit $j = \{0,1\}$ und $\oplus$=„Addition modulo 2“

- Grafische Darstellung:
  - Durchgezogene Linien = Qubits von „links nach rechts“
  - Operationen = Gatter, die auf die Qubits wirken:



  - Manchmal wird das X-Gatter auch so visualisiert: $-\oplus-$

Das Schaltkreismodell

Beispiel: Hadamard-Gatter $\mathrm{H} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$

mit $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ und $H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$

Dieses produziert aus **einem** Basis-Qubit, die Superposition der **beiden** Basis-Qubits

– Die Superpositionszustände haben eigene Bezeichnungen:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle$$
$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle$$

– Mathematisch: Anwendung von H auf die beiden Basiszustände führt zu einem Wechsel in eine „äquivalente Basis“

Das Schaltkreismodell

- Allgemeine Parametrierung der Ein-Qubit-Gatter:

$$U(\theta,\phi,\lambda) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i\phi+i\lambda}\cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

d.h. jedes der bisher gezeigten Gatter wird durch entsprechende Wahl der Winkel θ, ϕ, λ realisiert ☺

- **Spoiler**: Betrachte diese Winkel als Parameter, die „trainiert“ werden können → entsprechen den Parametern eines Neurons → „Variationelle Quantenalgorithmen“ für Quantum Machine Learning

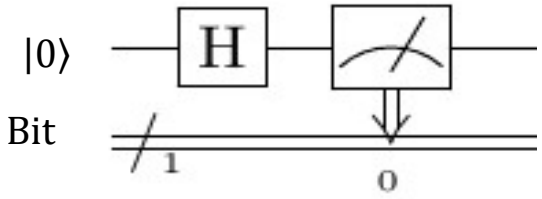
Das Schaltkreismodell

Auslesen eines Quantencomputers durch Messung von Qubits → Klassische Bits:

- $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \xrightarrow{Messu} \begin{cases} 0\rangle \text{mit Wahrscheinlichkeit} |\alpha|^2 \\ 1\rangle \text{ mit Wahrscheinlichkeit} |\beta|^2 \end{cases}$

- **Vor** der Messung ist es **unmöglich** zu wissen, wie die Messung lauten wird → Nur Wahrscheinlichkeitsaussagen möglich !!!

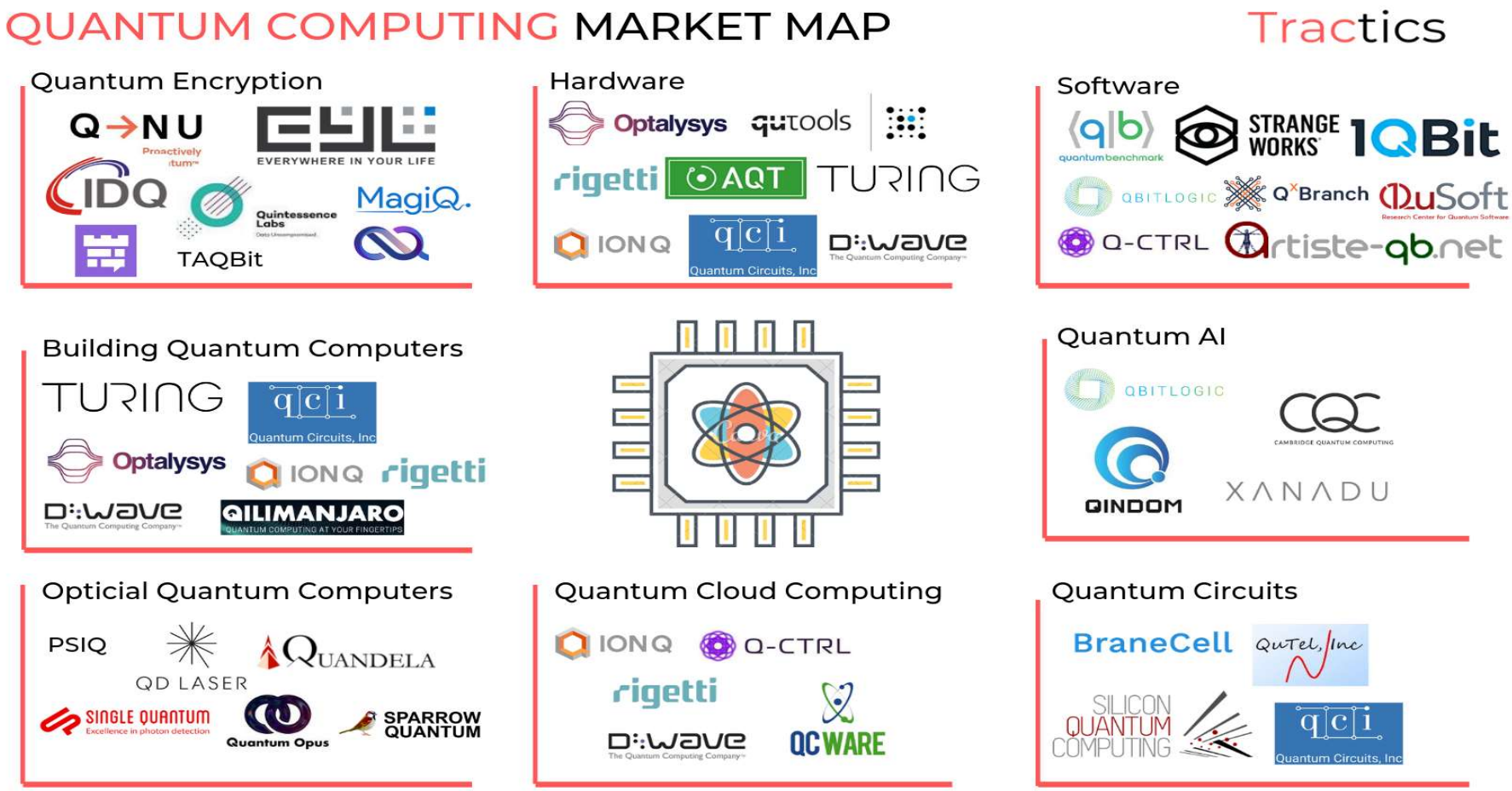
- Beispiel:



$$|0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \xrightarrow{Messung} \begin{cases} 0\rangle \text{ zu } 50\% \\ 1\rangle \text{ zu } 50\% \end{cases}$$

- **Nach** der Messung verbleibt das Qubit im gemessenen Zustand 0 oder 1

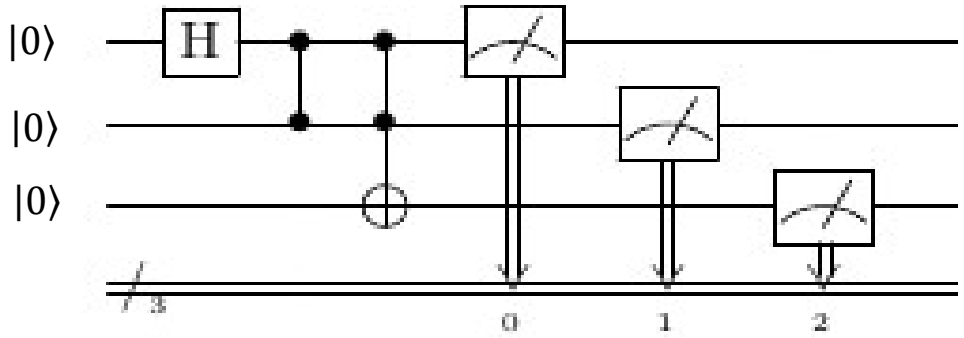
Experimentelle Realisierung von Quantencomputern



https://medium.com/datadriveninvestor/quantum-computing-73-companies-that-are-changing-the-computing-landscape-f39ebf0ccfee

Das Schaltkreismodell – Mehrere Qubits

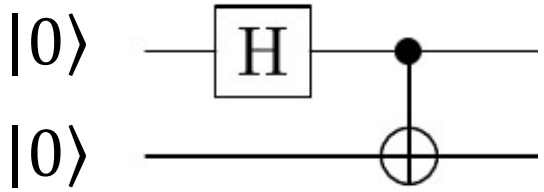
Mehrere Qubits sind der Input für einen Quantenschaltkreis:



- n QuBits gehen hinein (auf dem Bild: n=3): $|0\rangle|0\rangle|0\rangle = |000\rangle$
- n QuBits gehen auch wieder hinaus
- Quantengatter die auf eines oder mehrere Qubits wirken werden angewendet
- es wird gemessen → Ergebnis ist ein Bitstring; z.B. 101
- Nach Messung Zustand geändert zu z.B. $|1\rangle|0\rangle|1\rangle = |101\rangle$
- **Achtung**: Es gibt keine Schleifen und alle Schritte bis zur Messung sind reversibel!
- Trotzdem „universelles Berechnungsmodell“ im Sinne einer klassischen Turingmaschine

Das Schaltkreismodell

- Erzeugung eines **Verschränkten Zustandes** mit dem CNOT-Gatter:



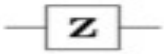
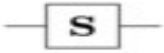
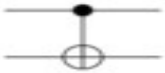
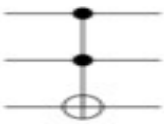
  - Rechnung:

$$|00\rangle \xrightarrow{H\ auf\ das\ 1.Qubit} \frac{1}{\sqrt{2}}(|00\rangle + |01\rangle) \xrightarrow{CNOT} \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

  - Dieser Zustand (Bell-Zustand Φ^+) lässt sich nicht als Tensorprodukt von Ein-Qubit-Zuständen schreiben

→ Wenn das erste Qubit gemessen ist, ist die Messung des zweiten Qubits determiniert ..

→ .. Egal wie weit die beiden Qubits voneinander entfernt sind!?

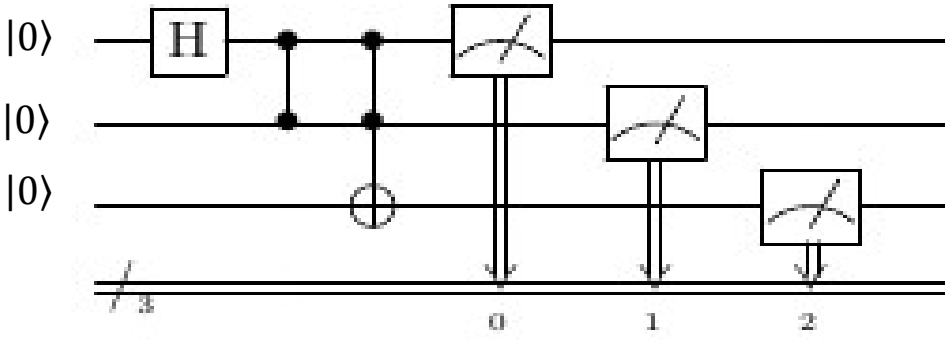
Operator	Gate(s)	Matrix
Pauli-X (X)		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)		$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

https://en.wikipedia.org/wiki/Quantum_logic_gate

3. PROGRAMMIERUNG EINES QUANTENCOMPUTER

Quanten-Software

- Programmierung eines Quantencomputers entspricht der „Manipulation“ von Qubits
- Realisiert durch Anwendung von Gattern auf eines oder mehrere Qubits
- Quanten-Algorithmus: Abfolge von Gattern und anschließende Messung:



- Ist dafür zwingend ein Quantencomputer notwendig ? Nein!

  → Berechnung eines Quantenalgorithmus entspricht i.W. Linearer Algebra mit komplexen Zahlen

  → Für eine moderate Anzahl von Qubits auf klassischem Rechner umsetzbar

Quanten-Software

- Allerdings steigt die Größe der Matrizen exponentiell mit 2^n
- Für n=50 Qubits: $2^{50} = 1125899906842624 \approx 1{,}12 \cdot 10^{15}$
- Einen echten Quantenvorteil verspricht man sich insbesondere ab n > 50 UND entsprechender Tiefe des Schaltkreises

- Design von Quantensoftware mit zwei unterschiedlichen „Backends“:
  - Für $n < 25 - 30$: Klassischer Rechner
    - Berechnung der exakten Ergebnisse eines Algorithmus über Lineare Algebra
    - Simulation eines Quantenexperiments als Zufallsexperiment
  - Quantencomputer mit entsprechender Anzahl vom Qubits
    - Physikalische Manipulation und anschließende Messung der Qubits

- Vergleich der Ergebnisse für $n < 20 - 30$ erlaubt Abschätzung der Güte

Quanten-Software

- Aber die Quantencomputer von IBM und anderer Hersteller haben doch schon ca. 100 Qubits !!??

→ Aber diese sind keine „logische“ sondern „nur“ physikalische Qubits da fehlerbehaftet:

→ „Dekohärenz“ durch Wechselwirkung mit der Umwelt

→ z.B. $|1\rangle \rightarrow |0\rangle$ oder $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \rightarrow |0\rangle$

→ Beschränkung auf „shallow“ (nicht zu tiefe Schaltkreise) mit der Hoffnung, dass trotz Fehler die Ergebnisse noch brauchbar sind …Oder:

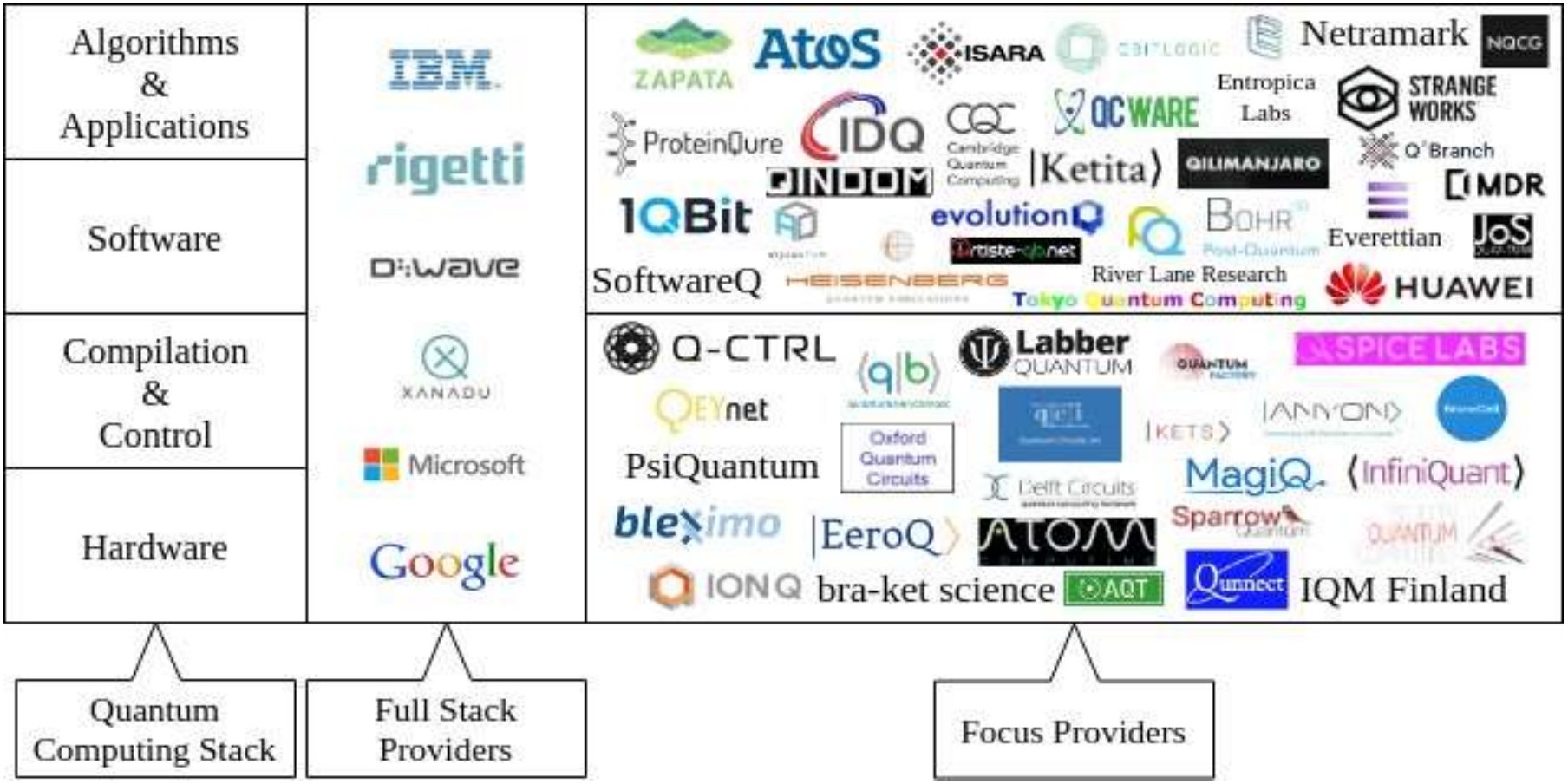
→ Korrekur der Fehler - wg. dem No-Cloning-Theorem (die auch beim klassischen Computing gemacht wird!!) hier viel aufwändiger ☹

→ Ziel: Reduktion der für ein logisches Qubit benötigter physikalischer Qubits

→ Sehr aktives Feld der Forschung!!

→ Hier: Wir nehmen an, wir hätte beliebig viele fehlerfreie Qubits ☺

Quanten-Software



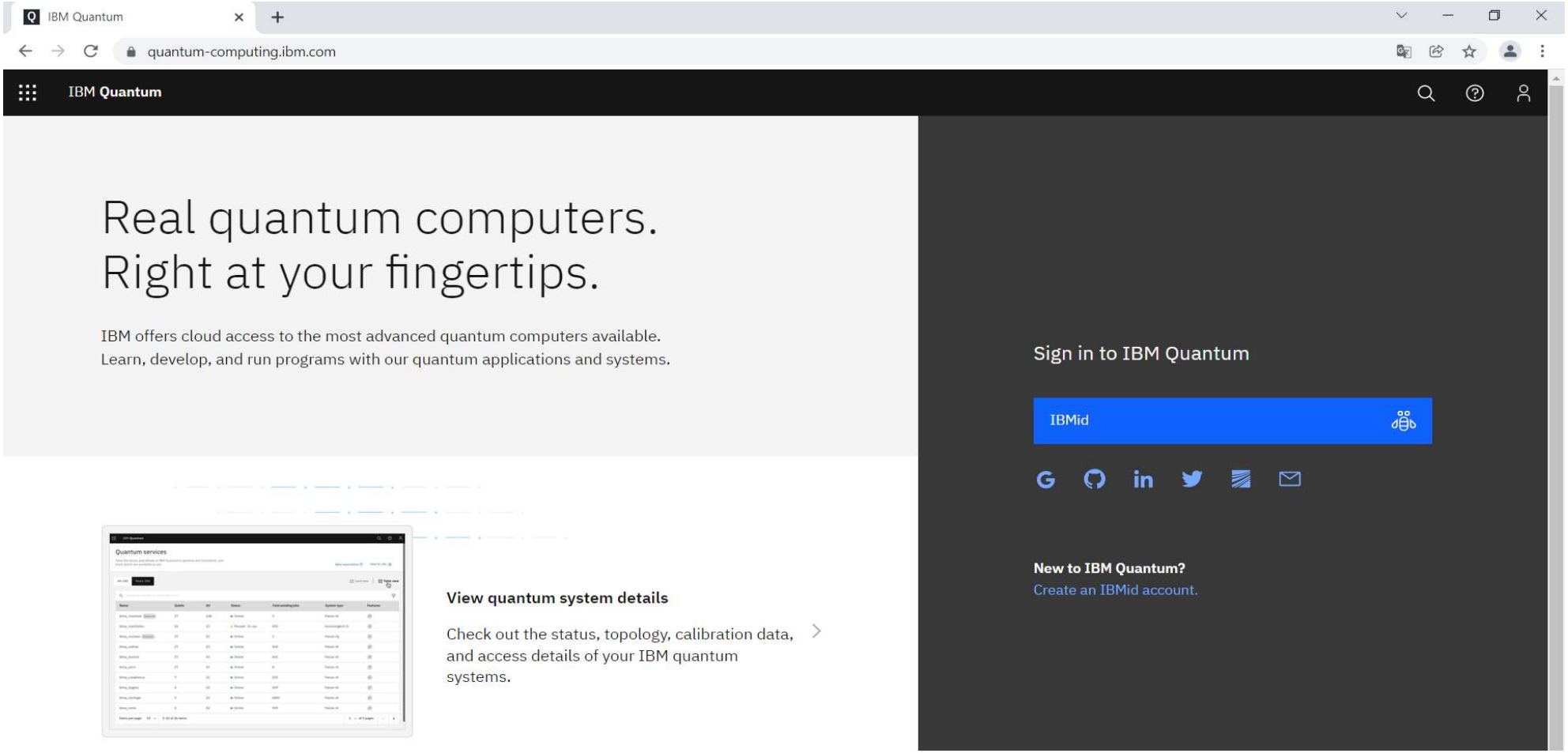
https://quantumcomputingreport.com/review-of-the-cirq-quantum-software-framework

Quanten-Software – Qiskit von IBM

Im Folgenden wird zum einen Qiskit von IBM zur Demonstration der Algorithmen verwendet:

- Qiskit ist Open-Source und kostenlos nutzbar
- Programmierung lokal (eigene Hardware) – Ansteuerung physikalischer Hardware über API
- Programmierung in Python
- Neben exakter Berechnung und Simulation stehen auch echte Quantencomputer bis ca. 100 Qubits zur Verfügung
- Online Tutorials und Material für Anfänger & Spezialisten
- Weltweit größte Community von „Quanten-Enthusiasten“ → Support und Austausch
- Auch im Unternehmenskontext wird Qiskit verwendet

Quanten-Software - Qiskit



Quanten-Software – PennyLane von Xanadu

Alternative für Quantum Machine Learning: **PennyLane**

- Xanadu ist kanadisches Quantenstartup, das neben Software photonische Quantenhardware baut
- PennyLane ist „hardware-agnostisch“, d.h. mit Quantenhardware aller gängigen Anbieter kompatibel – daneben auch extrem performante Simulatoren (z.B. auf Nvidia-GPUs)
- Problemlose Einbindung von Tensorflow und Pytorch als ML-Frameworks
- Out-of-the box Implementierung von Basiskomponenten von Quantenschaltkreisen (z.B. für Data-Embedding)
- Sehr enge Einbindung in die wissenschaftliche Forschung
- Gutes Trainingsmaterial, sowie Demo-Notebooks für typische Use-Cases bzw. neue Ideen

Einführung in Quantenalgorithmen

- **Was ist bei der Konstruktion von Algorithmen zu beachten?**
  - Während ein klassischer Bitstring der Länge in einem von 2^N möglichen Zuständen ist, kann ein N-Qubit-Zustand in allen möglichen 2^N möglichen Zuständen **gleichzeitig** sein.
  - Eine Strategie (die wir beim Grover-Algorithmus sehen werden) besteht darin, die Amplituden der verschiedenen Qubits so zu verändern, dass die Amplitude des Lösungsqubits maximiert; die anderen Amplituden minimiert werden.
  - Aufgrund der inhärent stochastischen Natur der Messung wird ein Quantenschaltkreis in der Regel mehrmals ausgeführt.

Quantenalgorithmen für Optimierung

- Nicht alle Optimierungsprobleme eigenen sich für Quantenalgorithmen ….
- … am besten eigenen sich solche sich mit **binären** Variablen
- Warum? Aus den Bits 0 und 1 werden Qubits $|0\rangle$ und $|1\rangle$
- … und bei denen die Zielfunktion z.B. quadratisch ist → sog. **QUBO**
- Praktisches Vorgehen:
  - Starte mit einem **relevanten** Problem → Hat es diese Form?
  - „Quantisiere“ das Problem
  - Löse das Problem mit geeigneten Quantenalgorithmen
- Sprache/Umgebung: Python

Ausblick auf den Workshop

- Diskussion eines Use-Case der sich als Optimierungsproblem der QUBO-Klasse formulieren lässt
- Klassische Lösungsalgorithmen als Benchmark
- „Quantisierung des Problems“
- Vorstellung von Lösungsmöglichkeiten bzw. Quantenalgorithmen
- Diskussion potenzieller Erweiterungen
- Praktische Anwendungsmöglichkeiten

4. OPTIMIERUNG MITTELS QUANTENCOMPUTER

Demo in Qiskit:

- **Qiskit-Basics**

Warum Optimierung?

- Die Lösung von **bestimmten** (!!) **Optimierungsproblemen** wird als eine der Aufgaben angesehen, bei denen nicht nur ideale fehlertolerante Quantencomputer, sondern eventuell sogar NISQ-Rechner einen Vorteil gegenüber klassischem Computing bieten „könnten“

- Warum das so ist, soll hier anhand eines einfachen Beispiels bottom-up motiviert werden

- Dieses lässt sich grundsätzlich beliebig erweitern um auch realistische Use-Cases abzudecken

- Hier soll der Weg vom Problem über die „klassische“ Lösung hin zur Quantenlösung Schritt für Schritt vorgestellt werden

Das Problem

- Die Würfel GmbH produziert und verkauft **Schachteln** in Würfelform:
- Ziel der Firma ist es, den Erlös (abzgl. Kosten) aus dem Verkauf der Schachteln zu maximieren:

Hierbei gilt (I):

- Schachtel bestehen aus Papier(0) oder Pappe (1). Für eine Pappschachtel wird im Vergleich zu einer Papierschachtel ein Preisaufschlag von 50€ verlangt.
- Schachtelkanten können verstärkt werden (1) oder nicht (0). Verstärkte Schachteln können 10€ teurer verkauft werden.
- Schachtel kann mit Schaumstoff ausgekleidet werden werden (1) oder nicht (0). Ausgekleidete Schachtel ist 20€ teurer.
- Schachtel wird bemalt (1) oder nicht (0). Bemalte Schachtel ist 30€ teurer.

→ Bei einer Pappschachtel mit verstärkten Kanten, Auskleidung und Bemalung
ergibt sich ein Mehrerlös im Verkauf von 50€+10€+20€+30€=110 €

Das Problem

– Zusätzlich sind folgende (Produktions-)Kosten bzw. zusätzliche Erlöse zu beachten (II):

- Wenn die Schachtel ausgekleidet und verstärkt wird: 30€ Kosten.
- Wenn die Schachtel aus Pappe ist und verstärkt wird: 30€ Kosten
- Wenn die Schachtel bemalt wird und gefüllt ist: Erlös steigt um 40€

→ Welche Kombination der Entscheidungsvariablen Papier/Pappe, Kantenverstärkung ja/nein; Auskleidung ja/nein; Bemalung ja/nein ist „die Beste“ ??

○ Optimierungsprobleme dieser Art heißen **QUBO** („Quadratic unconstrained optimization“) Problem

Formalisierung des Problems

- Die Entscheidungsvariablen werden im Vektor $\vec{x}$ zusammengefasst mit x_0 bis x_3:
  - $x_0 = \begin{cases} 0, & Schachtel\ ist\ aus\ Papier \\ 1, & Schachtel\ ist\ aus\ Pappe \end{cases}$
  - $x_1 = \begin{cases} 0, & Kanten\ sind\ nicht\ verstärkt \\ 1, & Kanten\ sind\ verstärkt \end{cases}$
  - $x_2 = \begin{cases} 0, & Schachtel\ ist\ nicht\ ausgekleidet \\ 1, & Schachtel\ ist\ ausgekleidet \end{cases}$
  - $x_3 = \begin{cases} 0, & Schachtel\ ist\ nicht\ bemalt \\ 1, & Schachtel\ ist\ bemalt \end{cases}$
- Hiermit lassen sich die Bedingungen unter (I) elegant formulieren als: $\boldsymbol{b^T} \cdot \vec{\boldsymbol{x}}$ mit $\boldsymbol{b^T} = (50,10,20,30)$

Formalisierung des Problems

Die Bedingungen unter (II), die also von zwei Variablen abhängen, lassen sich kompakt schreiben als: $\vec{x}^T \cdot \mathrm{C} \cdot \vec{x}$

mit der symmetrischen Matrix $C = \begin{pmatrix} 0 & -30 & 0 & 0 \\ -30 & 0 & -30 & 0 \\ 0 & -30 & 0 & 40 \\ 0 & 0 & 40 & 0 \end{pmatrix}$

Die zu optimierende Größe = Zielfunktion H ist also: $\boldsymbol{H = b^T \cdot \vec{x} + \frac{1}{2}\vec{x}^T \cdot \mathrm{C} \cdot \vec{x}}$

- Note: Durch die Ersetzung $b \rightarrow -b$ und $C \rightarrow -C$ kann aus dem Maximierungs- ein Minimierungsproblem gemacht werden
- Da $x_i \cdot x_i = x_i$ lässt sich der lineare Term auch in der Diagonale von C unterbringen

Klassische Lösung des Problems

- Suche nun die Kombination der x_i, d.h. den Vektor $\vec{x}$, welcher die Zielfunktion optimiert

- In unserem Fall gibt es 2^4=16 mögliche Kombinationen, die man alle durchprobieren kann = „Brute Force Lösung“

- Hat man allerdings 10 Variablen x_i , dann sind es schon $2^{10} = 1.024$ mögliche Kombinationen; bei 20 Variablen sind es 1.048.576 Möglichkeiten; bei 30 Variablen sind es 1.073.741.824 Möglichkeiten, so dass Brute Force an Grenzen stößt ☹

- Allerdings gibt es eine ganze Reihe von klassischen Optimierungsmethoden
(z.B. CPLEX), die man auch für solche Probleme verwenden kann …
deren Lösung aber nicht zwingend das Optimum sein muss…

Demo in Qiskit:

- **Optimierung klassisch**

„Quantisierung“ des Problems

Idee: Verwende Qubits statt Bits:

- Diese nehmen ja nicht nur die Werte 0 oder 1 an, sondern werden durch einen Zustand $|\psi\rangle = a|0 + b|1\rangle$ beschrieben

- Erst bei der Messung muss sich das Qubit „entscheiden“, ob es mit Wahrscheinlichkeit $|a|^2$ eine 0 wird oder mit Wahrscheinlichkeit $|b|^2 = 1 - |a|^2$ eine 1

- Nimm vier Qubits; führe einen geeigneten Algorithmus aus; messe die vier Qubits und generiere damit einen Bitstring der Länge vier, also z.B. [0,1,1,0]

- Was ist nun ein geeigneter Algorithmus, damit sich bei dieser Prozedur
  - zumindest mit hoher Wahrscheinlichkeit – auch der optimale Bitstring ergibt ?

Quantisierung des Problems

- Einer der Algorithmen, die man verwenden kann ist der **Quantum Approximate Optimization Algorithm (QAOA)**, der auf *Farhi, Edward, Jeffrey Goldstone, and Sam Gutmann. "A quantum approximate optimization algorithm." arXiv preprint arXiv:1411.4028 (2014)* zurückgeht.

- Ansatz: $U = e^{iH}$ Denn: Alle Manipulationen der Qubits (außer der Messung) lassen sich durch eine unitäre Matrix beschreiben; jede unitäre Matrix U besitzt diese Darstellung mit einer hermiteschen Matrix H.

- Verwende als hermitesche Matrix die quantisierte Zielfunktion des Problems:
  - $x_i \rightarrow \frac{1}{2}(1 - \hat{\sigma}_i^z)$ mit der Pauli-z-Matrix $\hat{\sigma}^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
  - $\boldsymbol{H = b^T \cdot \vec{x} + \frac{1}{2}\vec{x}^T \cdot C \cdot \vec{x}} \rightarrow \hat{H} = \sum_{i<j} J_{ij}\hat{\sigma}_i^z\hat{\sigma}_j^z + \sum_i h_i\hat{\sigma}_i^z$

Quantisierung des Problems

$$\hat{H} = \sum_{i<j} J_{ij}\hat{\sigma}_i^z\hat{\sigma}_j^z + \sum_i h_i\hat{\sigma}_i^z = \hat{H}_{Ising}$$

Aus der Zielfunktion wird dadurch ein „Hamilton-Operator“ und dieses Modell ist in der Physik als Ising-Modell oder allg. als Spin-Glas-Modell bekannt

- In unserem Problem mit vier Qubits lässt sich der Hamilton Operator als $16\ x\ 16$ Matrix darstellen …. die diagonal ist … und der kleinste (Eigen-)Wert in der Diagonale ist die gesuchte Lösung !!! ☺
- D.h. durch die „Aufblähung“ des Problems haben wir eine Darstellung gefunden in der die Lösung unmittelbar zugänglich ist ☺
- Ist unser Problem allerdings zu groß, hilft uns das auch nicht direkt weiter ☹

Quantisierung des Problems

- Indirekt aber schon, da es die Intuition vermittelt, welche Strategie wir nun verfolgen:

  Suche den kleinsten **Eigenwert** des Hamilton-Operators ….

  … und damit, suche den Zustand $|\Psi\rangle$, so dass der **Erwartungswert** des

  Hamilton-Operators minimal wird: $\langle\Psi|\hat{H}|\Psi\rangle \rightarrow min$

- Starte mit einem parametrisierten Ansatz $|\Psi(\alpha, \beta, \gamma, ..)\rangle$ und variere die Parameter so lange, bis das Minimum erreicht ist. Das ist die allg. Idee der sog. „**Variationellen Quantenalgorithmen (VQA)**".

- **QAOA** ist hiervon ein Spezialfall

Quantisierung des Problems

Bei Ansatz von QAOA lässt man sich von der Physik leiten, nämlich dem sog. **Adiabatentheorem**:
„Ein System bei dem man Änderungen nur langsam genug vornimmt bleibt immer in seinem Grundzustand“

- $\hat{H}_{QAOA} = \beta \hat{H}_{mixer} + \gamma \hat{H}_{Ising}$ mit $\hat{H}_{mixer} = \sum_i \hat{\sigma}_i^x$ dessen Grundzustand trivial ist
- Anstatt nun gemäß Adiabatentheorem das System im Grundzustand des Mixers zu starten und langsam (adiabatisch!) in den Grundzustand des Ising-Hamiltonians übergehen zu lassen, werden β und γ als zu optimierende Parameter behandelt.
- Mehr noch, $U = e^{i\hat{H}_{QAOA}}$ wird nicht nur einmal sondern k-mal mit jeweils anderen Parametern $\beta_i, \gamma_i; \mathrm{i} = 1, .., \mathrm{k}$ angewandt
- Die Parametern $\beta_i, \gamma_i; \mathrm{i} = 1, .., \mathrm{k}$ schließlich werden mit einen klassischen Optimierer bestimmt.

Quantisierung des Problems

- Dies wird oftmals folgendermaßen ausgedrückt:

- $\left|\vec{\beta},\vec{\gamma}\right\rangle = \mathrm{U}(\beta_k \hat{H}_{mixer})\mathrm{U}(\gamma_k \hat{H}_{Ising}) \dots \mathrm{U}(\beta_1 \hat{H}_{mixer})\mathrm{U}(\gamma_1 \hat{H}_{Ising})|0, \dots, 0\rangle$

  - Die Parametern $\beta_i, \gamma_i; \mathrm{i} = 1, .., \mathrm{k}$ schließlich werden mit einem klassischen Optimierer bestimmt:
    - Präpariere $\left|\vec{\beta},\vec{\gamma},\right\rangle$ mit initialen Werten für $\beta_i \; und \; \gamma_i$
    - Messe alle n Qubits → Bitstring der Länge n
    - Setze diesen Bitstring in die (klassische) Zielfunktion ein
    - Variiere $\beta_i \; und \; \gamma_i$ so lange bis ein Optimum der Zielfunktion erreicht ist
  - Und dies funktioniert ☺

Demo in Qiskit:

- **Optimierung mit dem Quantencomputer**

Erweiterungen von QUBOS

- Problemklasse „QUBO“ wirkt auf den ersten Blick sehr restriktiv …..
- Kann aber in unterschiedliche Richtungen erweitert werden:
  - **Quadratic**: Es ist möglich, auch Wechselwirkungen dritter und höherer Ordnung zu berücksichtigen → Drei- oder Mehr-Qubit-Gates, die sich allerdings wieder auf elementare (d.h. maximal Zwei-Qubit-) Gates reduzieren lassen
  - **Unconstrained**: Nebenbedingungen z.B. $\sum_i x_i = B$ lassen sich über einen quadratischen Penalty-Term einbauen.
  - **Binary**: Rationale Zahlen lassen sich über ihre Binärschreibweise darstellen; ebenso reelle Zahlen → Zahl der benötigten Qubits wächst allerdings sehr schnell ☹

Wo spielen diese Probleme eine Rolle?

- Auf https://blog.xa0.de/post/List-of-QUBO-formulations/ finden sich 112(!) Optimierungsprobleme, die sich als QUBO formulieren lassen:
  - Viele bekannte „**klassische NP-schwere“** Optimierungsaufgaben lassen sich als QUBO formulieren: z.B. Max-Cut, Maximum 2- und 3-Sat, Graph Colouring, Traveling Salesman, …
  - Optimierungsaufgaben im Rahmen des klassischen Machine Learning: z.B. Support Vector Machines, k-means Clustering, …
  - Sowie Optimierungsaufgaben bei anwendungsnahmen Use-Cases: z.B. Settlement von Wertpapieren, Portfoliooptimierung, Routenplanung, Job Shop Scheduling, ….

Wo spielen diese eine Rolle?

https://blog.xa0.de/post/List-of-QUBO-formulations/

jrnl · home about list my ventures publications LinkedIn

List of QUBO formulations

About the author:
Daniel is CTO at rhome GmbH, and Co-Founder at Aqarios GmbH. He holds a M.Sc. in Computer Science from LMU Munich, and has published papers in reinforcement learning and quantum computing. He writes about technical topics in quantum computing and startups.

Below a list of 112 optimization problems and a reference to mulation of each problem is shown. While a lot of these problems are classical op blems from mathematics (also mostly NP-hard problems), there are interestingly a ns for Machine Learning, such as the L1 norm or linear regression. Graph base problems often encode the graph structure (adjacency matrix) in the QUBO, while such as Number Partitioning or Quadratic Assignment encode the problem matrices s.t. non-linear system of equations can be found in the QUBO. The quadratic unconstrained binary optimization (QUBO) problem itself is a NP-hard optimization problem that is solved by finding the ground state of the corresponding Hamiltonian on a Quantum Annealer. The ground state is found by adiabatically evolving from an initial Hamiltonian with a well-known ground state to the problem Hamiltonian.

This is by no means the definite list of all QUBO formulations out there. This list will grow over time.

For 20 of these problems the QUBO formulation is implemented using Python (including unittests) in the QUBO-NN project. The Github can be found here.

Problem	QUBO formulation
Number Partitioning (NP)	Glover et al.[2]
Maximum Cut (MC)	Glover et al.[2]
Minimum Vertex Cover (MVC)	Glover et al.[2]
Set Packing (SP)	Glover et al.[2]
Set Partitioning (SPP)	Glover et al.[2]
Maximum 2-SAT (M2SAT)	Glover et al.[2]
Maximum 3-SAT (M3SAT)	Dinneen et al.[9]
Graph Coloring (GC)	Glover et al.[2]
General 0/1 Programming (G01P)	Glover et al.[2]
Quadratic Assignment (QA)	Glover et al.[2]
Quadratic Knapsack (QK)	Glover et al.[2]
Graph Partitioning	Lucas[7]

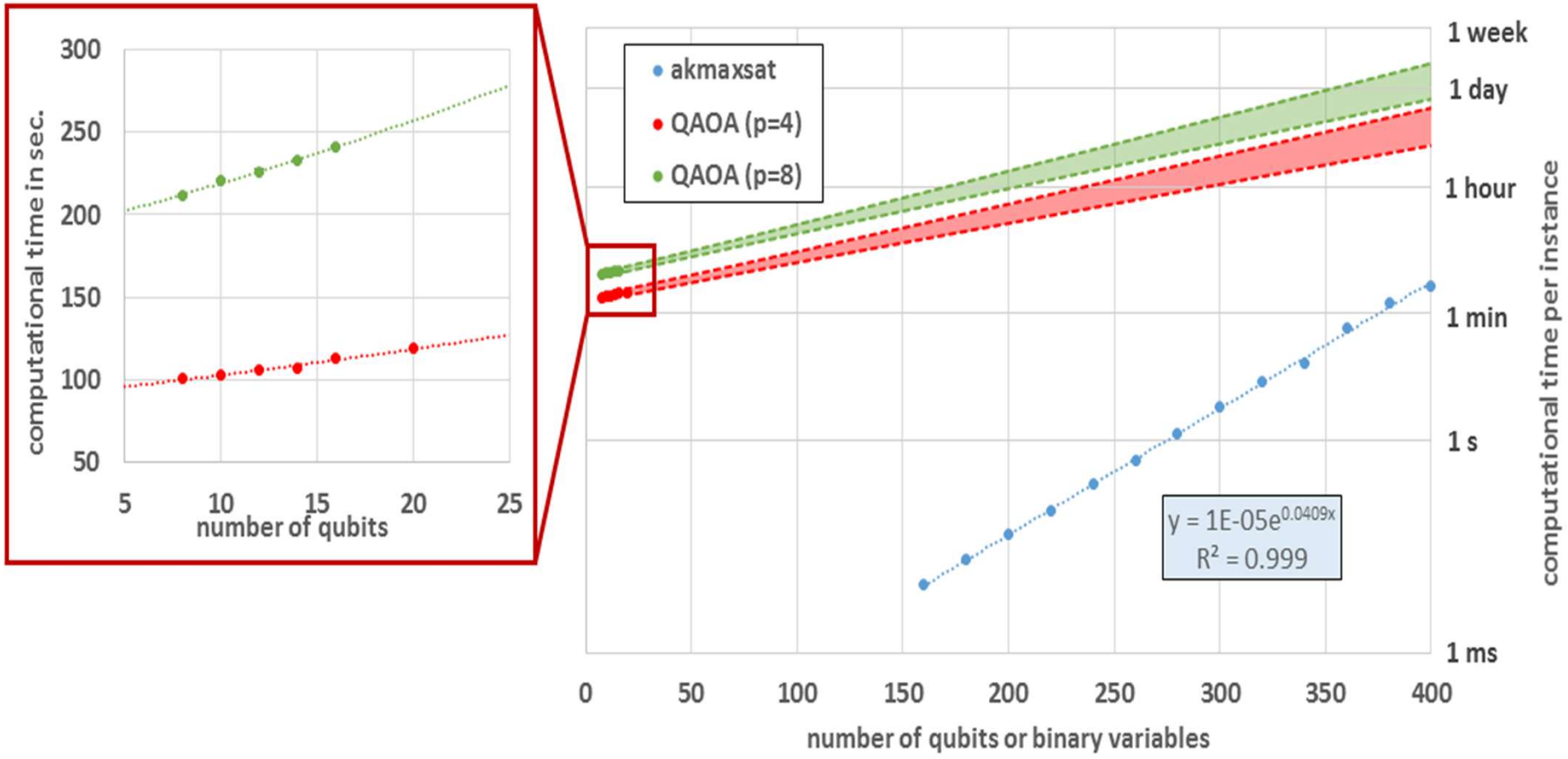
Guter Ausgangs-punkt für die Bearbeitung eigener Problem-stellungen !

Quantenvorteil bei der Lösung von QUBOs?

- Mathematischer Beweis eines Quantenvorteils bei der Lösung von QUBOs gibt es bisher nicht. Polynomiale Lösung wird allgemein auch nicht erwartet. Ggf. „kleinerer Exponent“

- Am intensivsten theoretisch und praktisch untersucht ist der QAOA-Algorithmus:
  - Exakt mathematische Aussagen über die Güte gibt es bisher nur für z.B. k=1
  - Und/oder bestimmte Spezialfälle, d.h. bestimmte Struktur der Wechselwirkungsmatrix
  - Rechnungen auf echter Quantenhardware bisher nur bis ca. 50 Qubits; ein Bereich, der auch noch klassisch idR gut lösbar ist

Quantenvorteil in der Zukunft?



Guerreschi, Matsuura, Sci. Rep. 2019

Offene Forschungsfragen

- Inwieweit hängt ein etwaiger Quantenvorteil von der Problemstruktur (=Wechselwirkungsmatrix) ab?

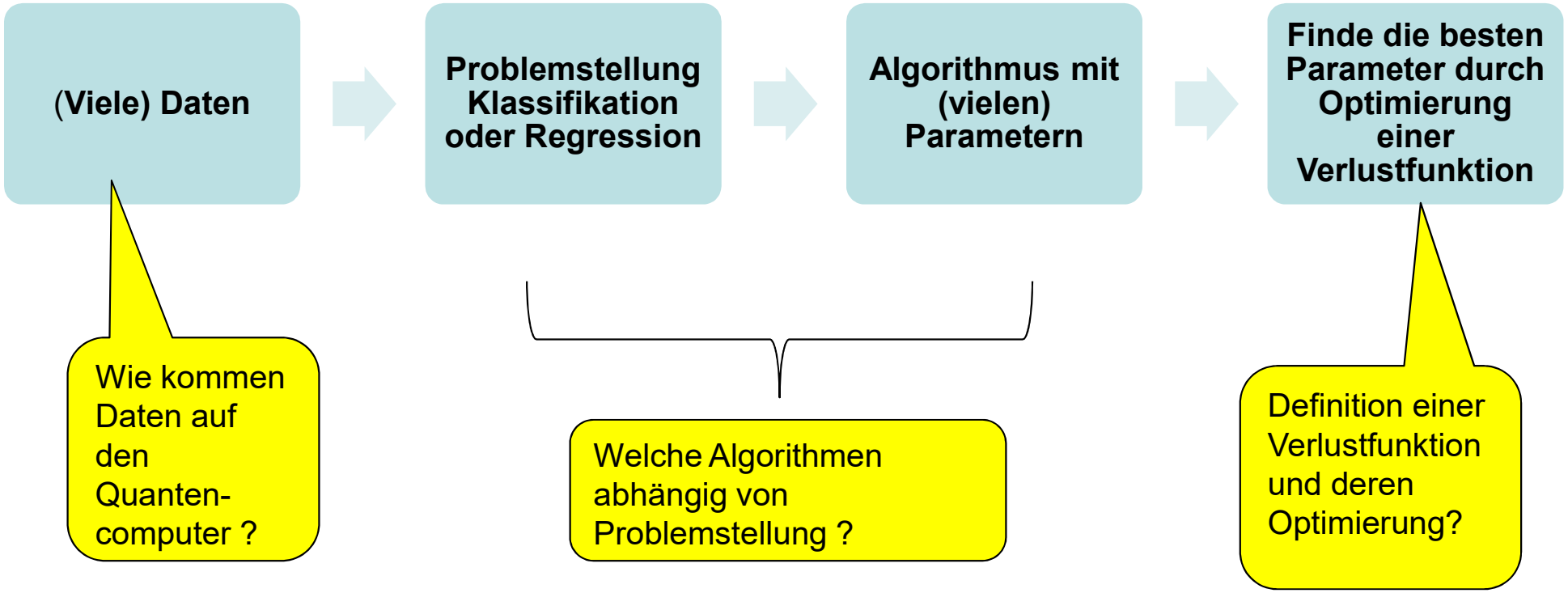
- Wie fehlerresilient ist z.B. der QAOA-Algorithmus und wie lässt sich dies ggf. durch Modifikationen am Algorithmus erhöhen?*

- Wie lassen sich reale Probleme aus der Anwendung (Wirtschaft und Technik) schneller und einfacher in die QUBO-Form bringen?

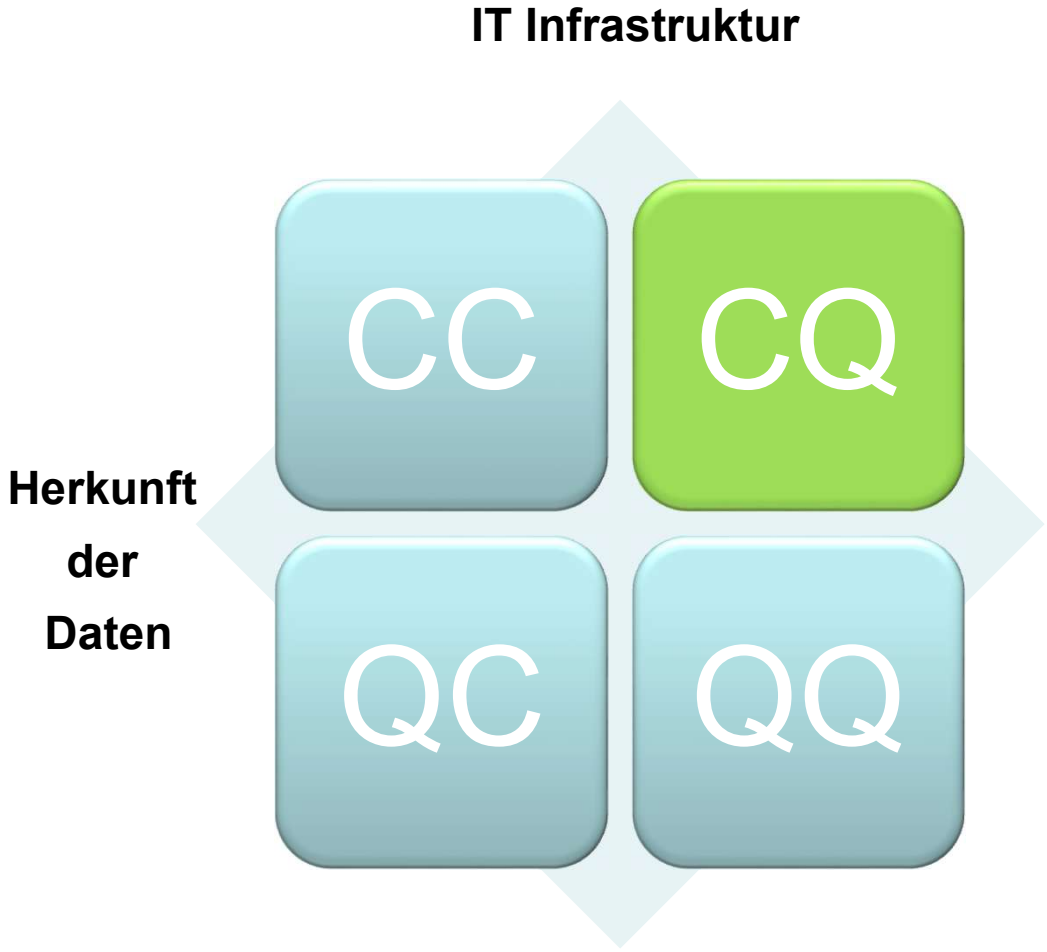
* Benchmarking the performance of portfolio optimization with QAOA,
S. Brandhofer, D. Braun, V. Dehn, G. Hellstern, M. Hüls, et al. Quant.Inf.Proc. 22 (2023) 1, 25

OPTIONAL:
5. QUANTEN MACHINE LEARNING

Machine Learning → Quantum Machine Learning



Problemstellung



Weitere Differenzierung innerhalb „CQ“:

Algorithmen für (zukünftige) ideale Quantencomputer:

- Anwendung der bekannten „Basisalgorithmen“ (Grover, QFT, …) mit dem Ziel einer Komplexitäts-/Geschwindigkeitsreduktion gegenüber klassischen Algorithmen des maschinellen Lernens

- Jackpot: Gibt es einen exponentiellen Speedup, d.h. Reduktion von exponentieller in polynomiale Laufzeit?
  - Mathematisch lückenlose Beweise hier extrem tückisch …. (vgl. z.B. A quantum-inspired classical algorithm for recommendation systems, E. Tang, arXiv:1807.04271)
  - Vorteil wird z.T. zunichte gemacht, durch das Problem, die klassischen Daten „auf den Quantencomputer“ zu bringen

Weitere Differenzierung innerhalb „CQ“:

Algorithmen für NISQ-Rechner:

- Ausgangsfragen:
  - Wie kann man vorhandene Quanten-Hardware für maschinelles Lernen nutzen?
  - Macht es Sinn den Quantencomputer als „Algorithmus“ zu verwenden und zusammen mit klassischen Komponenten zu kombinieren?
  - Gibt es bestimmte Daten, bei denen es einen „Vorteil“ geben könnte?

  → Geschwindigkeits- / oder Komplexitätsvorteil wird i.A. eher nicht erwartet

  → Hoffnung auf Performancegewinn im Sinne von z.B. bessere Trennschärfe bei Klassifikationsproblemen

  → Grundsätzlich eher heuristisches Thema – Theorie dazu noch am Anfang

NISQ-kompatible ML-Algorithmen

Quanten unterstützte Support Vector Maschines

- SVM benötigt eine sog. Kernel-Matrix
- Dies stellt die Informationen über die (verallgemeinerten) Abstände zwischen den Datenpunkten bereit und erlaubt insbesondere ein Einbettung der Daten in einen hochdimensionalen Raum
- Es konnte gezeigt werden, dass wenn Kernelmatrix mit einem Quantenrechner erzeugt wird, Einbettungen jenseits der klassischen Möglichkeiten erzielbar sind (Supervised learning with quantum-enhanced feature spaces,
  V. Havlíček, et al Nature volume 567, 209–212 (2019)
- Ob diese nicht-trivialen Einbettungen tatsächlich einen Vorteil bieten, ist allerdings (noch) offen

NISQ-kompatible ML-Algorithmen

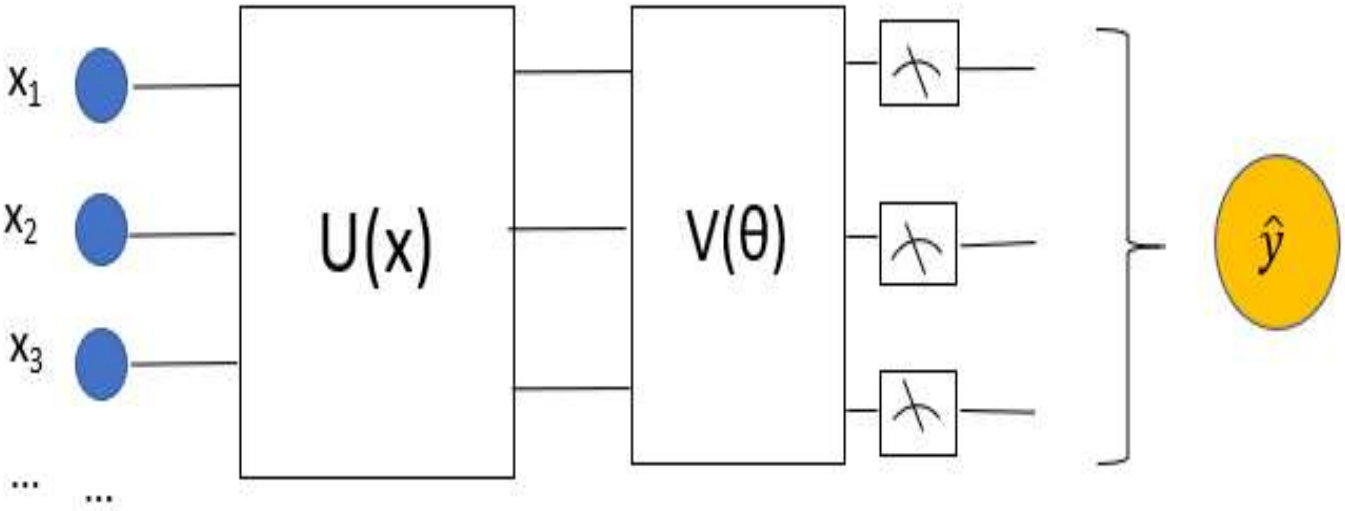
Quanten Neuronale Netze (QNN)

- Klassische Daten werden mit einer unitären Matrix **U(x)** in den Quantencomputer „geladen“ = Data Encoding
  - Hierfür stehen unterschiedliche Möglichkeiten im Raum
  - Teilweise schlechte Skalierungseigenschaften falls Daten viele Merkmale haben
- Parametrisierbare unitäre Matrix **V($\boldsymbol{\theta}$)** mit einer Schichtenstruktur in Analogie zu klassischen neuronalen Netzen
- Messung der Qubits und klassisches Post-Processing um die Messergebnisse auf die Zielvariable abzubilden (Klassifikation oder Regression)
- Klassische Feedback-Schleife um mit einer Verlustfunktion die Parameter $\boldsymbol{\theta}$ zu adjustieren

→ Umsetzung ??

NISQ-kompatible ML-Algorithmen

Quanten Neuronale Netze (QNN)



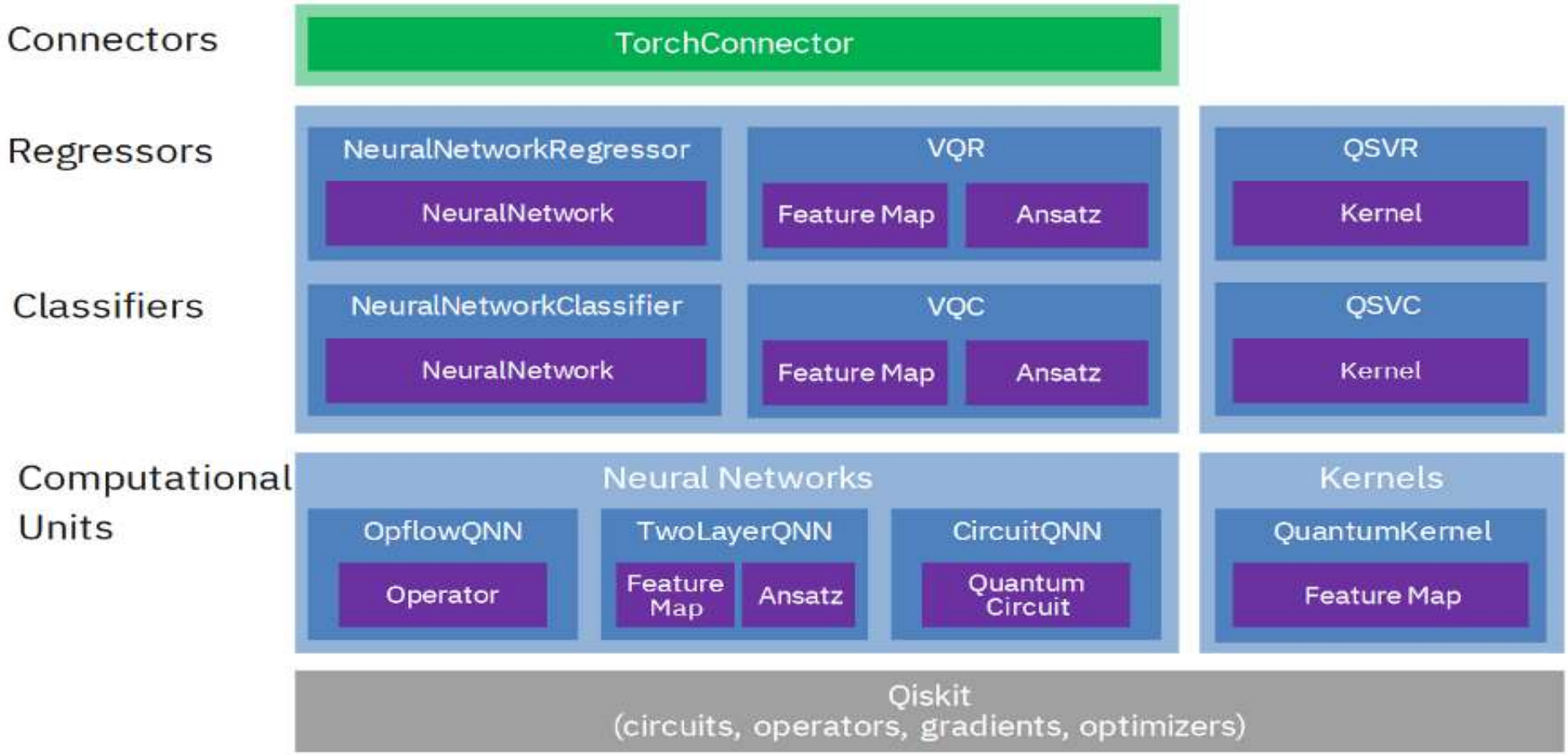
Ziel:
Weniger „Speedup“ – sondern „bessere“ Algorithmen, d.h. z.B. genauere Klassifizierung von Daten

Anforderungen an Software-Tools für QML

Für QML werden Quantenschaltkreise aber auch klassische Komponenten (Post-processing, Feedback-Schleife für Parameter etc.) benötigt

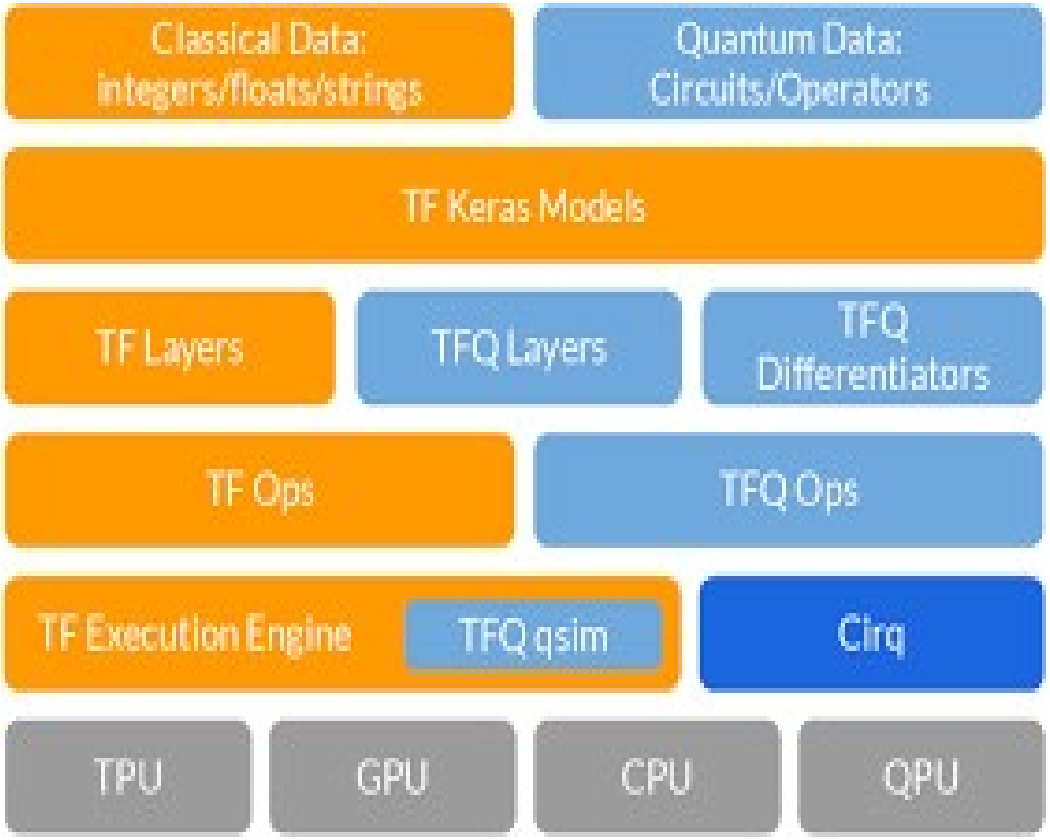
- Für letztere gibt es in der ML-Welt out-of-the-box Lösungen → Lassen sich diese nutzen?
- Unterstützung bei der Konfiguration von **U(x)** und **V($\boldsymbol{\theta}$)** – wie kann man schnell zwischen unterschiedlichen Realisierungen wechseln?
- Für die Trainingsphase muss Schaltkreis SEHR OFT durchlaufen werden → hohe Performance der Quanten-Simulatoren notwendig!
- Lassen sich ggf. auch unterschiedliche physikalische Backends (mit unterschiedlichen Technologien z.B. Photonisches QC) anbinden?
- Existiert ein Ökosystem an Entwicklern und Nutzern für den Austausch?

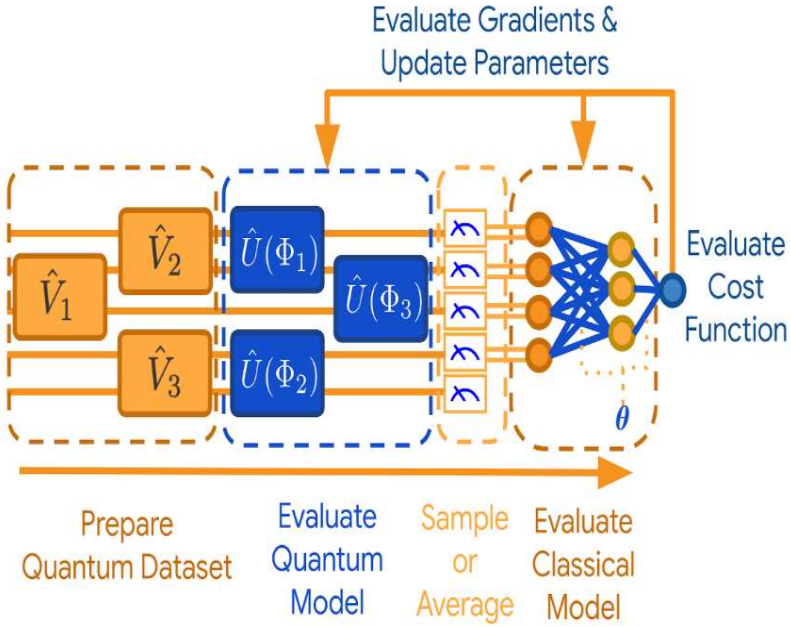
Qiskit von IBM



Quelle: IBM

Tensorflow Quantum von Google Research





Vgl. TensorFlow Quantum: A Software Framework for Quantum Machine Learning arXiv:2003.02989, 2020

Tensorflow Quantum von Google Research

Einschätzung

- Pro:
  - Umfangreiche Dokumentation, viele Notebooks mit Beispielen
  - Darunter auch nicht-triviale Beispiele, z.B. Transfer Learning oder CNNs
  - Simulatoren extrem schnell
  - Nahtlose Anbindung an Tensorflow/Keras und deren Möglichkeiten
- Contra:
  - Setzt auf cirq auf → geringere Verbreitung als z.B. Qiskit
  - Um die Möglichkeiten zu nutzen → Functional API von Tensorflow und intensive Beschäftigung mit dem Innenleben von TF und Keras
  - Auf die Quantenhardware von Google ausgerichtet

PennyLane von XanaduAI

- XanaduAI beschäftigt sich fast ausschließlich mit QML → Fokussierung statt umfänglicher Quantensoftwarelösung
- Neben Software für QML („Pennylane“) Entwicklung von photonischer Hardware und entsprechende Software-Tools hierzu („Strawberry Fields“)
- Dezidiert Hardware-agnostischer Ansatz mit dem unterschiedliche Simulatoren, aber auch Quantenhardware über eine entsprechende API angebunden werden können
- Einbindung von Tensorflow/Keras bzw. Pytorch problemlos möglich, um deren Vorteile zu nutzen

Data-Encoding U(x)

Diskussion folgender Möglichkeiten:

- Basis Encoding
- Amplituden Encoding
- Angle Encoding
- SU(2) Encoding

Basis Encoding

Basis Encoding als konzeptionell einfachste Möglichkeit

– Stelle Daten als Binärzahlen dar und kodiere diese dann in entsprechende Qubits:

Input Daten		
Sample:	Merkmal x_1	Merkmal x_2
1	5	6
2	4	1

Input Daten	
Sample:	Quantenzustand
1	$\|101\ 110\rangle$
2	$\|100\ 001\rangle$

Input Daten		
Sample:	Binäres Merkmal x_1	Binäres Merkmal x_2
1	101	110
2	100	001

Amplituden Encoding

Amplituden Encoding

- Verwende die klassischen Daten als Amplituden eines Quantenzustandes

Merkmal x_1	Merkmal x_2	Merkmal x_3	Merkmal x_4
6	-10,3	13,52	4,2

Normierung

Merkmal x_1	Merkmal x_2	Merkmal x_3	Merkmal x_4
$^{6}/_{18,51}$	$^{-10,3}/_{18,51}$	$^{13,52}/_{18,51}$	$^{4,2}/_{18,51}$

- Amplituden kodierter Zustand mit log2(n) Qubits für n-dimensionalen Punkt:

$$\frac{6}{18,51}|00\rangle - \frac{10,3}{18,51}|01\rangle + \frac{13,52}{18,51}|10\rangle + \frac{4,2}{18,51}11\rangle$$

Angle Encoding

Angle Encoding

- Verwende die klassischen Daten als Argumente einer Pauli-Y-Rotation

Merkmal x_1	Merkmal x_2	Merkmal x_3	Merkmal x_4
6	-10,3	13,52	4,2

Faktor 1/2

Merkmal x_1	Merkmal x_2	Merkmal x_3	Merkmal x_4
$6/2$	$-10{,}3/2$	$13{,}52/2$	$4{,}2/2$

$$\bigotimes_{i=1}^{4} R_y(x_i)|0000\rangle = \begin{pmatrix} \cos(3)|0\rangle \\ \sin(3)|1\rangle \end{pmatrix} \otimes \begin{pmatrix} \cos(-5{,}15)|0\rangle \\ \sin(-5{,}15)|1\rangle \end{pmatrix} \otimes \begin{pmatrix} \cos(6{,}76)|0\rangle \\ \sin(6{,}76)|1\rangle \end{pmatrix} \otimes \begin{pmatrix} \cos(2{,}1)|0\rangle \\ \sin(2{,}1)|1\rangle \end{pmatrix}$$

- Im Vergleich zum Amplituden Encoding mehr Qubits aber Gate-Komplexität geringer

SU(2) Encoding

SU(2) Encoding

– Eine unitäre Transformation eines Qubits kann durch drei Winkel parametrisiert werden (vgl. Quantenalgorithmen und ihre Implementierung (1))
→ kodiere jeweils drei Merkmale in einem Qubit

Merkmal x_1	Merkmal x_2	Merkmal x_3
6	-10,3	13,52

θ	ϕ	λ
6	-10,3	13,52

$$U(\theta,\phi,\lambda) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i\phi+i\lambda}\cos\left(\frac{\theta}{2}\right) \end{pmatrix} = \begin{pmatrix} \cos\left(\frac{6}{2}\right) & -e^{13{,}52i}\sin\left(\frac{6}{2}\right) \\ e^{-10{,}3i}\sin\left(\frac{6}{2}\right) & e^{-10{,}3i+13{,}52i}\cos\left(\frac{6}{2}\right) \end{pmatrix}$$

Data-Encoding U(x)

Neben den hier diskutierten Encoding Methoden gibt es noch weitere, die z.T. auch in PennyLane implementiert sind:

- Hamiltonian Encoding
- IQP Encoding
- QAOA Encoding
- ….

Die Frage nach der „besten“ Encoding Strategie ist ein aktuelles Forschungsgebiet *(vgl. z.B. Effect of data encoding on the expressive power of variational quantum-machine-learning models, M. Schuld et al, Phys. Rev. A 103, 032430, 2021)*

Demo in PennyLane:

- **Data Encoding**

Variationeller Schaltkreis V(θ)

- Auf die in den Circuit eingebetteten Daten wird eine Folge von Gattern mit (freien) Parametern angewandt
- Konzeptionell entspricht dies dem Vorgehen eines neuronalen Netzes
- Es lässt sich zeigen, dass mit diesem Vorgehe jede hinreichend gutartige Funktion approximiert werden kann (→ Regressionsproblem) bzw. dass damit eine Trennung in unterschiedliche Klassen (→ Klassifikationsproblem) möglich ist
- Welche Abfolge von Gattern allerdings am „besten sind“, ist a priori nicht klar ☹ Auch dies ist ein aktuelles Forschungsgebiet!

Variationeller Schaltkreis V(θ)

- Der Schaltkreis kann prinzipiell aus 1-Qubit Gates und Gatter für die Verschränkung CX oder CZ aufgebaut werden

- Um die Komplexität zu erhöhen werden die Gatter nicht nur einmal angewandt sondern in Schichten geordnet, die dann wiederholt werden → entspricht der Layer-Struktur eines neuronalen Netzes

- Es hat sich herausgestellt, dass es vorteilhaft sein kann, die Daten nicht nur am Anfang des Schaltkreises, sondern mehrmals einzuspeisen, sog. „Data re-uploading“ (vgl. *Data re-uploading for a universal quantum classifier, A. Pérez-Salinas, Quantum 4, 226 (2020))*

Schaltkreis mit Basic Entangling

- Auf jedes Qubit wird parallel nur ein elementares Gatter (RX, RY oder RZ) mit jeweils einem freien Parameter („Winkel der Drehung“) angewandt
- Danach werden jeweils Qubits in einer Kette/Ring verschränkt: 1. Qubit mit 2. Qubit, 2. Qubit mit 3. Qubit, …. ,letztes Qubit mit 1. Qubit
- Die so definierte Schicht („Layer“) wird mehrmals angewandt
- Anzahl der Parameter: Anzahl Qubits * Anzahl Schichten

Schaltkreis mit Strongly Entangling

- Auf jedes Qubit wird parallel ein allgemeines SU(2) Gatter $U(\theta, \phi, \lambda)$ mit jeweils drei freien Parameter angewandt.
- Danach werden jeweils Qubits in einer Kette/Ring verschränkt: 1. Qubit mit 2. Qubit, 2. Qubit mit 3. Qubit, …. ,letztes Qubit mit 1. Qubit
- Die so definierte Schicht („Layer“) wird mehrmals angewandt
- Anzahl der Parameter: 3 * Anzahl Qubits * Anzahl Schichten

Mögliche Erweiterungen

- Verwende statt CX und CZ parameterabhängige 2-Qubit Gates
- Verschränke nicht nur „benachbarte Qubits“ sondern alle Qubits mit allen Qubits
- Teile die Schichten in Blöcke auf, die in sich homogen sind
- Für einen systematischen Vergleich 19 (!) unterschiedlicher Ansätze für vier Qubits, vgl. *„Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms”, S. Sim et al, Advanced Quantum Technologies 2.12 (2019): 1900070*

Demo in PennyLane:

- **Variationeller Schaltkreis**

Kombination zum Gesamtbild

Nachdem wir U(x) und V(θ) spezifiziert haben, fehlt nur noch:

1. Die Messung eines / oder aller Qubits → Bit / Bitstring

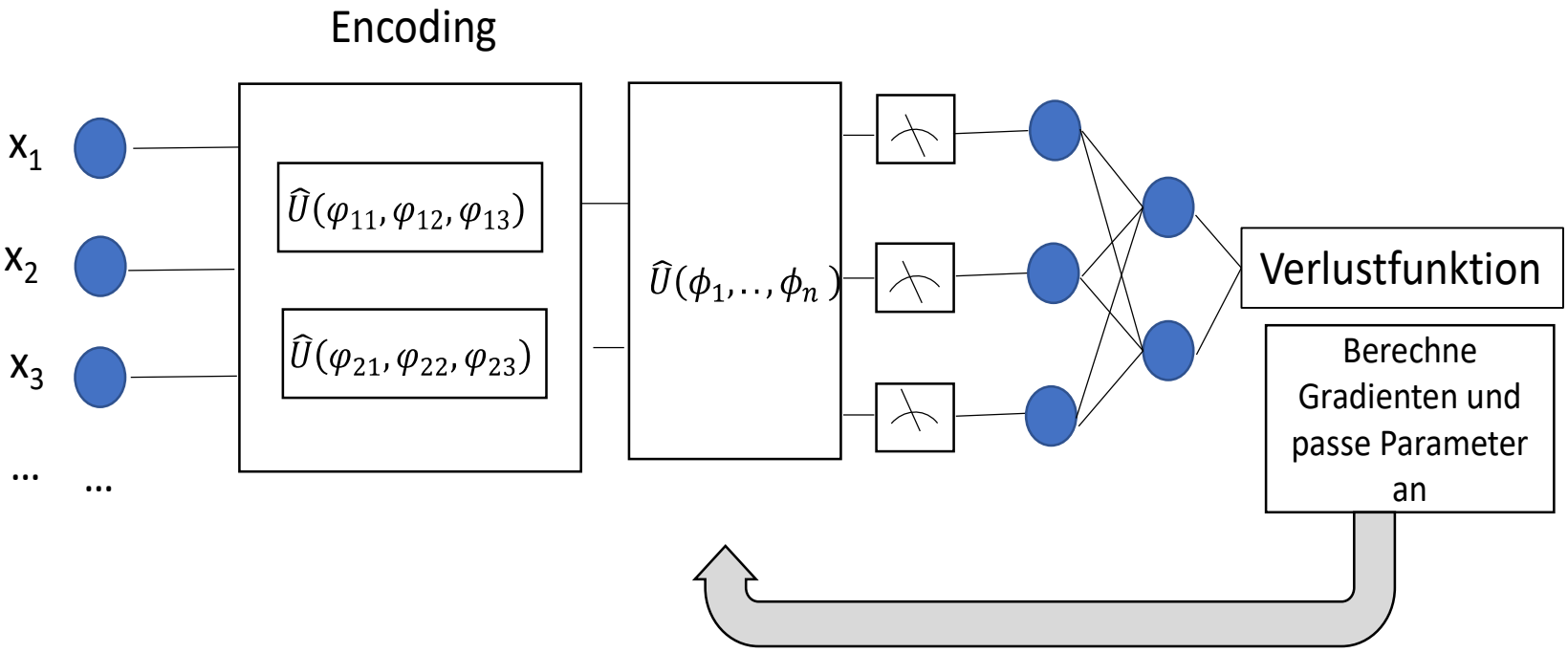
2. Das klassische Postprocessing: Bit/Bitstring → Zielwert des Problems, d.h. z.B. Label bei einem Klassifikationsproblem

3. Definition und Auswertung einer Verlustfunktion → RMSE bei Regression oder CrossEntropy bei Klassifikation

4. Klassische Optimierung um die Parameter des Schaltkreises solange anzupassen bis die Verlustfunktion das Minimum annimmt

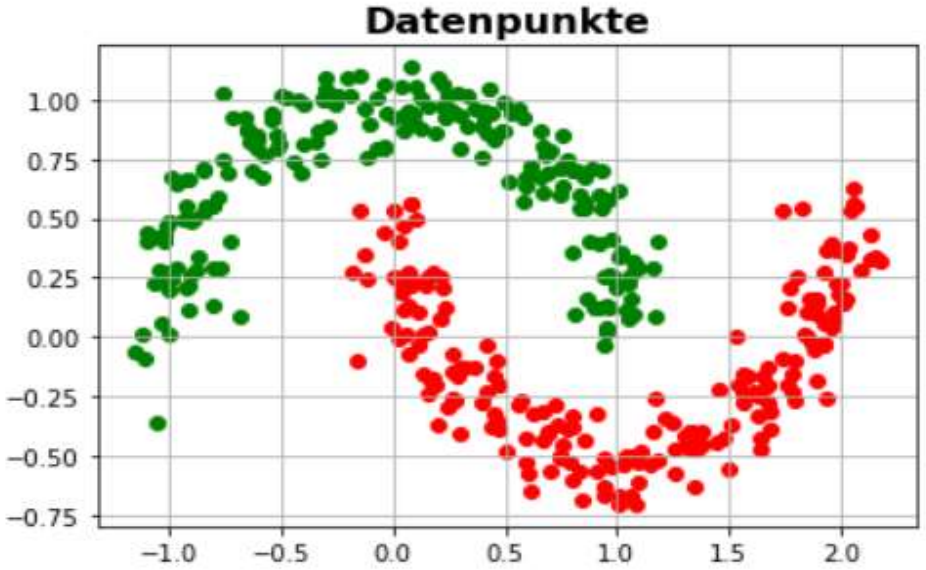
→ Schritte 2. bis 4. soll ein klassisches ML-Framework, z.B. Pytorch übernehmen.

Kombination zum Gesamtbild



Klassifizierung von 2-Dim Daten

- Beispielhaft soll der zweidimensionale Datensatz „Moon“ mit 400 Punkten klassifiziert werden:

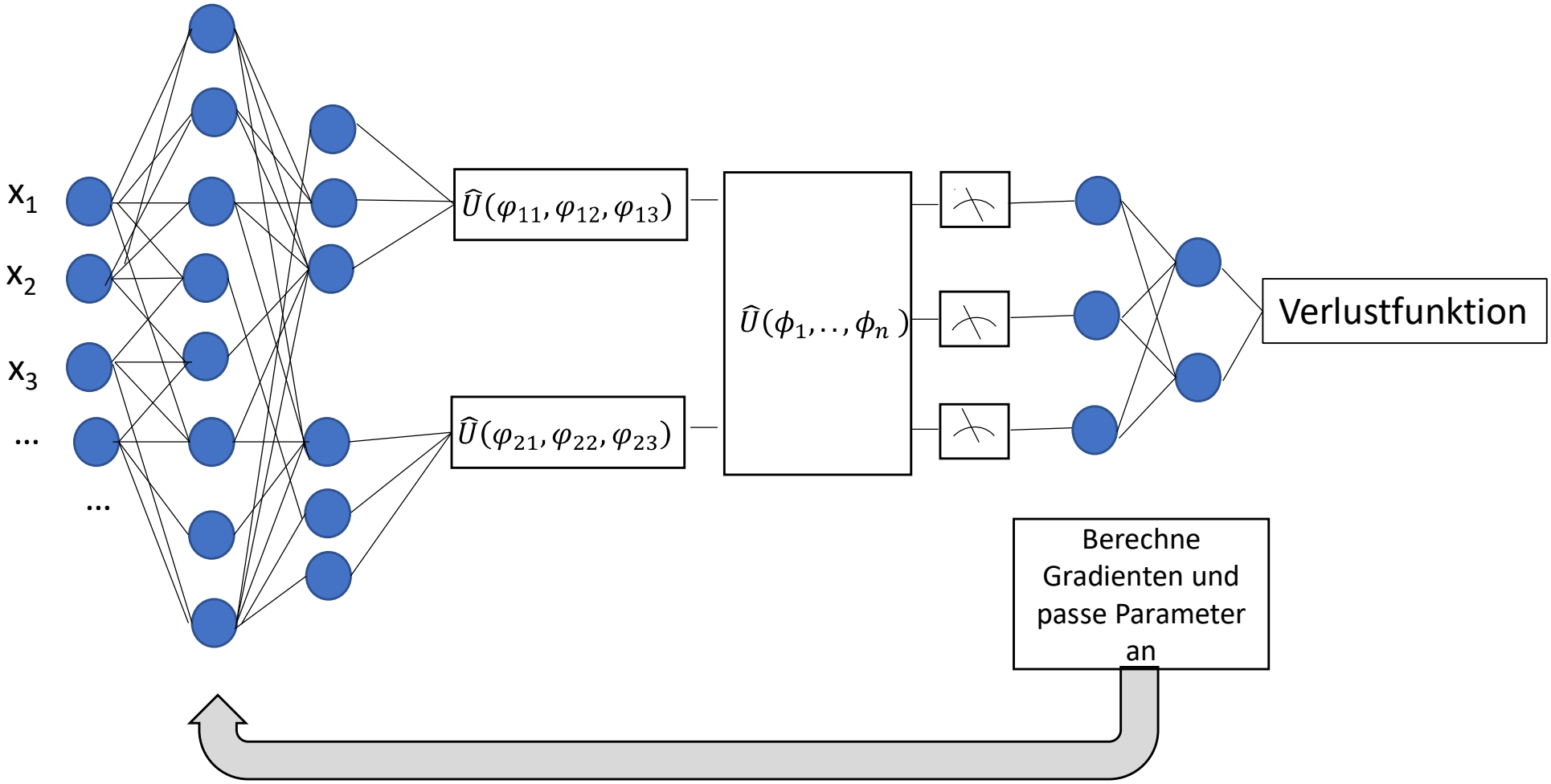


- Eine lineare Klassifizierung der Daten ist nicht möglich
- Es handelt sich um ein binäres Klassifizierungsproblem mit der Zielvariablen „grün“ oder „rot“ bzw. 0 oder 1.

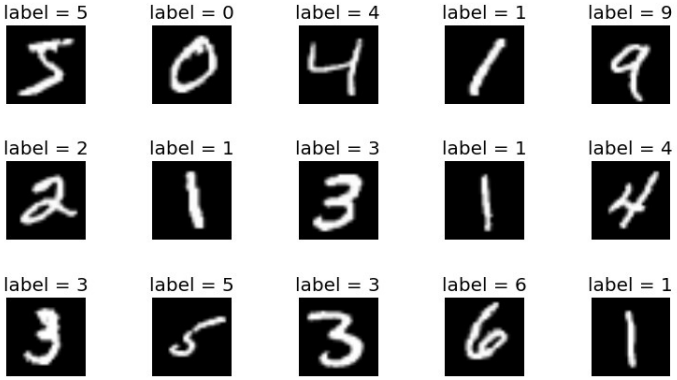
Klassifizierung von 2-Dim Daten

- Wie gleich gezeigt wird, ist ein Klassifizierungsmodell mit einer Genauigkeit > 95% auf Trainings- und Testdaten möglich ….
- … was allerdings in der gleichen Größenordnung liegt wie die Genauigkeit eines „klassischen“ Machine Learning Modells
- Gibt es aber Konstellationen, bei denen ein Quantenmodell GENAUERE Ergebnisse liefert als ein klassisches Modell ?? Eventuell sogar mit weniger Trainingsschritten?
- Dies ist eine offene Forschungsfrage – wobei die sehr starke Vermutung besteht, dass es letztlich an den Daten bzw. deren Struktur liegen wird

Hybrides Netzwerke aus klassischen und Quanten-Schichten

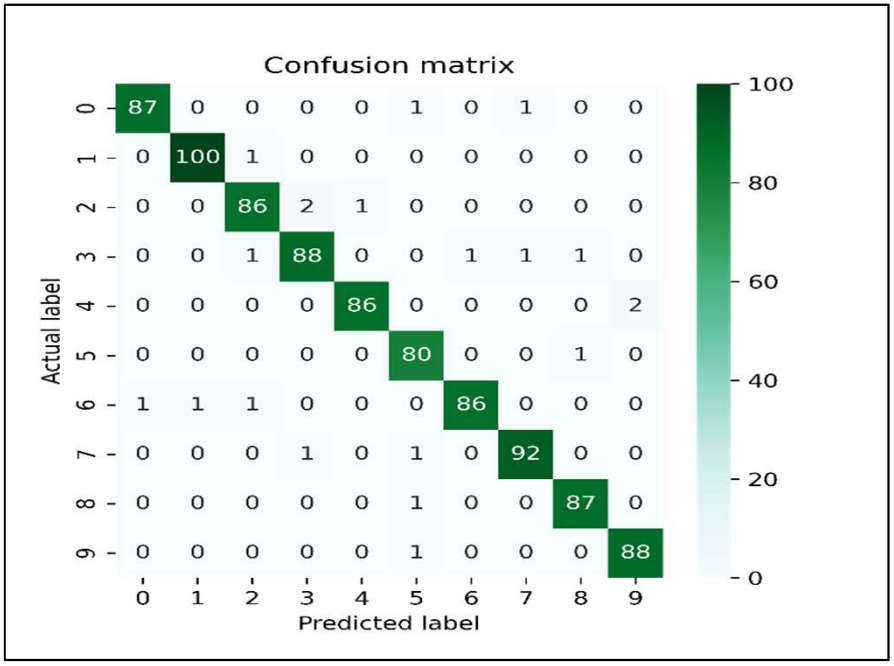


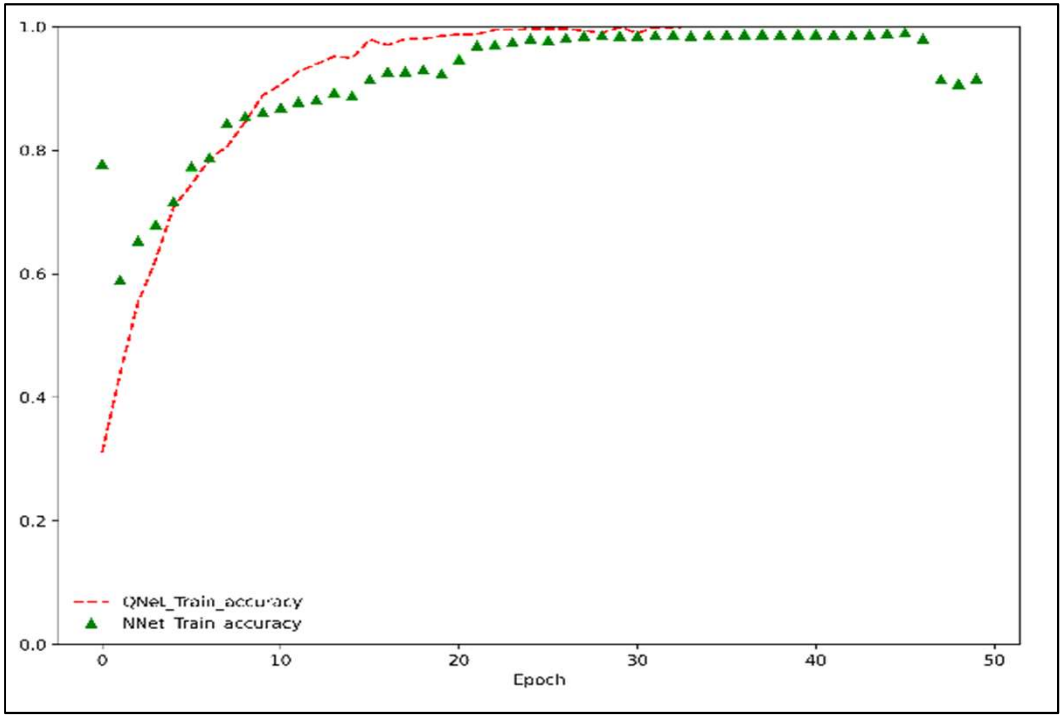
Ergebnisse



MNIST Data

60.000 graue Bilder (784 Pixels = Merkmale pro Bild) mit den handschriftlichen Ziffern 0,…..,9; 1.000 images wurden für das Training verwendet:





Demo in PennyLane:

- **Implementierung eines QNN**

Ressourcen: Literatur

DAS Standardwerk:

- Quantum Computation and Quantum Information, Michael A. Nielsen & Isaac L. Chuang, Cambridge University Press, 2010

Leichte Einführung:

- Quantum Computing verstehen , Grundlagen – Anwendungen –Perspektiven, Matthias Homeister, Springer 2022
- **Quantencomputing kompakt, Bettina Just, Springer 2020**

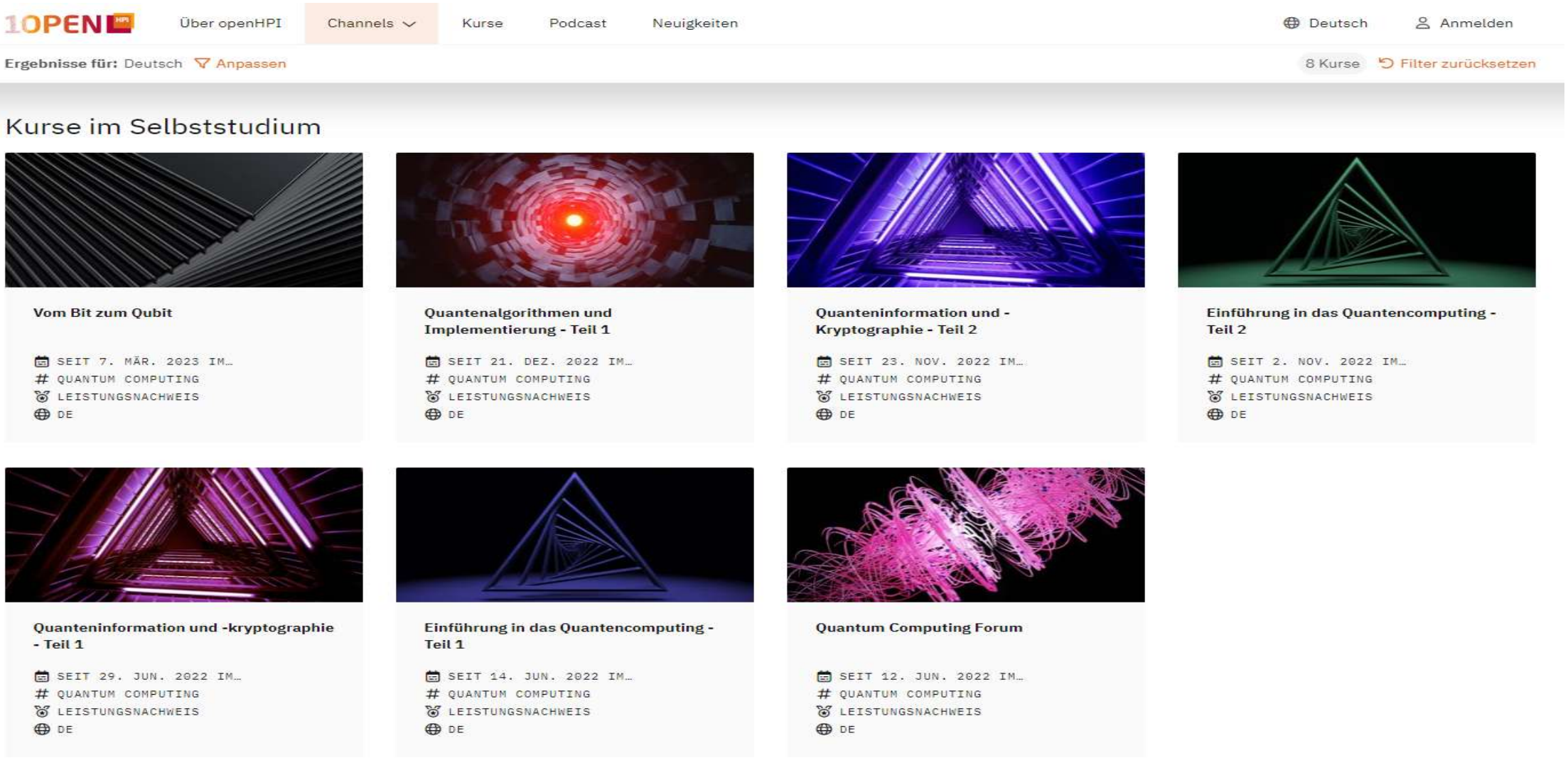
Inkl. Programmierung:

- Dancing with Qubits, Robert S. Sutor, Packt Publishing, 2023
- Quantum Computing: An Applied Approach, Jack D. Hidary, Springer, 2019

Fokussierung QML:

- Supervised Learning with Quantum Computers, Maria Schuld & Francesco Petruccione, Springer, 2023

Ressourcen: https://open.hpi.de/channels/quantum

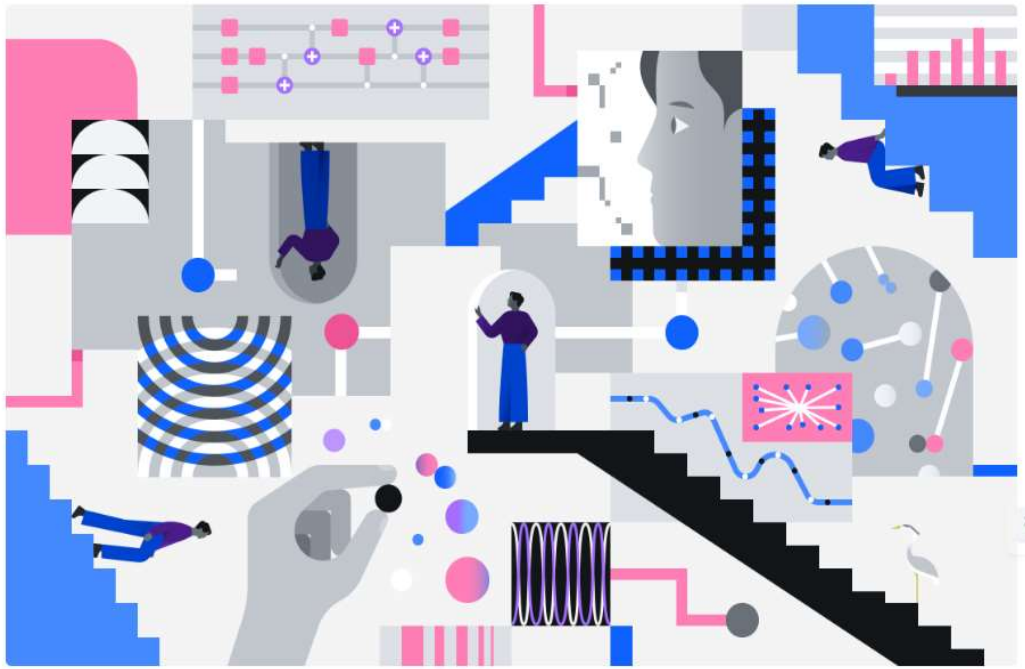


Ressourcen: https://learning.quantum.ibm.com/course/basics-of-quantum-information

IBM Quantum **Learning** | Home | Catalog | Composer | Sign in

Basics of Quantum Information

Created by John Watrous



This is the first unit of the *Understanding Quantum Information and Computation* series, which explains quantum information and computation at a detailed mathematical level.

The first unit, *Basics of Quantum Information*, begins with a mathematical description of quantum information for both single and multiple systems, then moves on to quantum circuits, and finally covers three fundamentally important examples — quantum teleportation, superdense coding, and the CHSH game — all of which are connected to the phenomenon of entanglement.

The series includes both written content and videos featuring John Watrous.

Sign in to track progress | Start from beginning →

Kontakt

Prof. Dr. Gerhard Hellstern

Center of Finance
Duale Hochschule Baden-Württemberg,
Stuttgart

gerhard.hellstern@dhbw-stuttgart.de

Prof. Dr. Gerhard Hellstern
Banking, Data Science and Quantum Computing
(Qiskit Advocat)