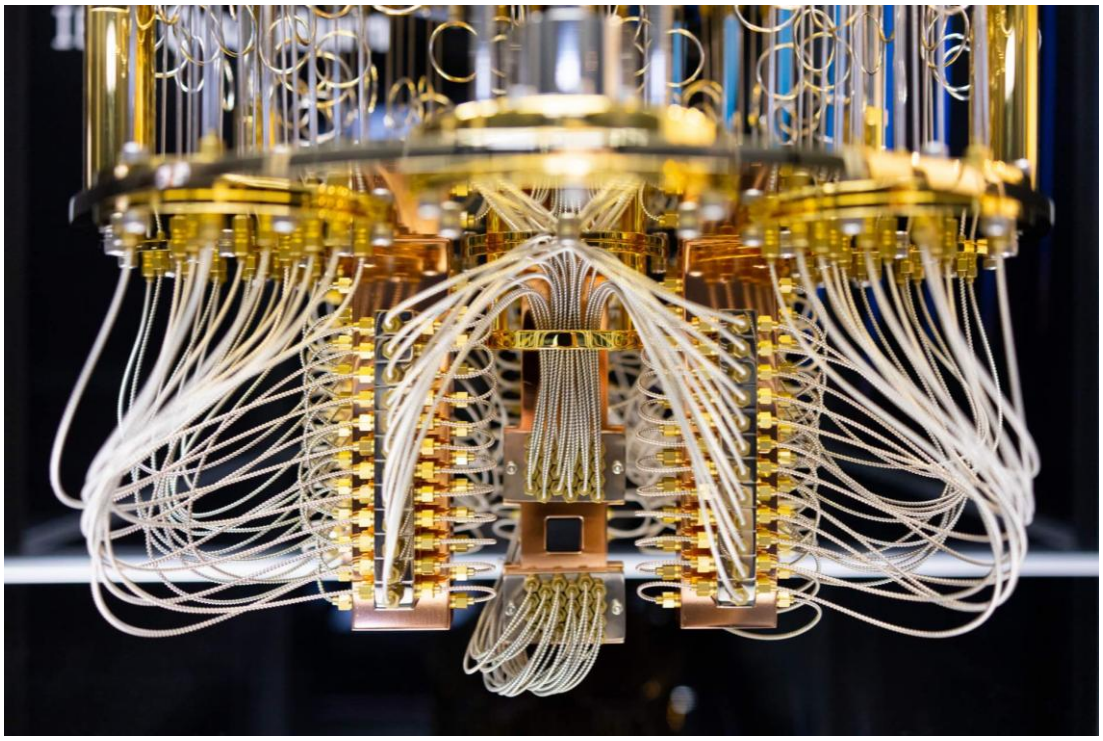
CHALMERS



Kvantfelskorrektion av bitfel i repetitionskoder med maskininlärning

Undersökning av maskininlärningens roll i utvecklingen av kvantdatorer

Kandidatarbete TIFX11-VT25-93

Hugo Modin Asklid
Nils Kalmnäs Drakenfors
Caroline Paulis
Tova Hedmar
Jacob Sandström
Emil Olofsson

INSTITUTIONEN FÖR FYSIK

CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2025
www.chalmers.se

Kandidatarbete 2025

Kvantfelskorrektion av bitfel i repetitionskoder med maskininlärning

Undersökning av maskininlärningens roll i utvecklingen av kvantdatorer

Quantum Error Correction of Bit-Flip Errors in Repetition Codes Using Machine Learning

Investigation of the Role of Machine Learning in the Development of Quantum Computers

HUGO MODIN ASKLID
NILS KALMNÄS DRAKENFORS
CAROLINE PAULIS
TOVA HEDMAR
JACOB SANDSTRÖM
EMIL OLOFSSON

CHALMERS

Institutionen för Fysik
Chalmers Tekniska Högskola
Göteborg, Sverige 2025

Kvantfelskorrektion av bitfel i repetitionskoder med maskininlärning
Undersökning av maskininlärningens roll i utvecklingen av kvantdatorer
HUGO MODIN ASKLID (asklid@student.chalmers.se)
NILS KALMNÄS DRAKENFORS (nilskal@student.chalmers.se)
CAROLINE PAULIS (paulisc@student.chalmers.se)
TOVA HEDMAR (tovah@student.chalmers.se)
JACOB SANDSTRÖM (jacsands@student.chalmers.se)
EMIL OLOFSSON (emilolof@student.chalmers.se)

Handledare: Mats Granath (mats.granath@physics.gu.se)
Examinator: Jan Swenson (jan.swenson@chalmers.se)

Kandidatarbete 2025
Institutionen för Fysik
Chalmers Tekniska Högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Omslagsbild:
Kawase, S. för IBM. (2022). *Photo of a model of IBM Quantum System One, as installed at Shin-Kawaski for the University of Tokyo.*

Skriven i LaTeX
Göteborg, Sverige 2025

Förord

Vi vill först och främst tacka vår handledare Mats Granath som har väglett oss i arbetet. Mats har bistått med relevant litteraturmaterial samt förklarat svårtydda begrepp och koncept, vilket har bidragit till fördjupade kunskaper inom området av kvantfelskorrektion. Mats har också varit en stabil kontaktperson under projektets gång och diverse mötesdiskussioner har varit till stor hjälp i arbetet.

Ytterligare vill vi tacka Moritz Lange och Jacob Olsson för att ha bidragit med grundläggande kod för avkodningsalgoritmerna som använts i projektet, vilket har underlättat och effektiviserat arbetet.

Detta arbete uppskattar stöd från WACQT:s testbädd för kvantteknologi som drivs av Chalmers Next Labs och finansieras av Knut och Alice Wallenbergs stiftelse.

Beräkningarna möjliggjordes av resurser tillhandahållna av Nationell Akademisk Infrastruktur för Superdatorer i Sverige (NAISS), delvis finansierat av Vetenskapsrådet genom bidragsavtal nr 2022-06725.

Under arbetet har AI-baserade verktyg använts för att utveckla förståelse för området och kod samt förbättra formuleringar i text.

Till sist poängteras att denna rapport förutsätter grundläggande kunskaper inom kvantfysik, vilka huvudsakligen innefattar generell förståelse och särskilda kvantfysikaliska begrepp.

Hugo Modin Asklid, Nils Kalmnäs Drakenfors, Caroline Paulis,
Tova Hedmar, Jacob Sandström, Emil Olofsson,
Göteborg, maj 2025.

Kvantfelskorrektion av bitfel i repetitionskoder med maskininlärning
Undersökning av maskininlärningens roll i utvecklingen av kvantdatorer
HUGO MODIN ASKLID
NILS KALMNÄS DRAKENFORS
CAROLINE PAULIS
TOVA HEDMAR
JACOB SANDSTRÖM
EMIL OLOFSSON
Institutionen för Fysik
Chalmers Tekniska Högskola

Sammanfattning

En av de största utmaningarna med kvantdatorer är att minimera och korrigera fel som uppstår på grund av kvantsystemets instabilitet. Kvantbitarna i systemet är känsliga för yttre faktorer och störningar, vilka kan ge upphov till olika typer av kvantfel, bland annat bitfel. För att lösa detta finns olika metoder för att förutsäga och korrigera kvantfel. Två sådana metoder är Minimum Weight Perfect Matching (MWPM) och maskininlärning. I denna rapport undersöks effektiviteten av ett tränat graf-neuralt nätverk (GNN) som avkodare genom att jämföra andelen uppkomna bitfel i repetitionskoder gentemot MWPM på IBM:s kvantdatorer.

Metoden bygger på att, med hjälp av Qiskit, konstruera en kvantkrets som kan upptäcka bitfel utan att kollapsa kvantbitarnas tillstånd, och således erhålla andelen uppstådda fel i form av syndrom. Data som insamlats från körningar på kvantdatorn används för att träna ett GNN och tillämpas på MWPM för koddistanserna $d \in [3, 21]$. Prestandan hos avkodarna jämförs därefter.

I resultaten framkom det att ett större GNN med sju graf-faltningslager generellt presterade bättre än både MWPM och ett mindre nätverk med tre graf-faltningslager för alla undersökta koddistanser, med undantag för $d = 7$. Även MWPM presterade bättre än det mindre nätverket, vilket i sin tur generellt hade lägst prestanda av de undersökta avkodarna. Metod och resultat analyseras utifrån teorin och problemställning för att ge en bredare förståelse för hur ett tränat GNN och MWPM beter sig för olika koddistanser. Även relevanta felkällor som kan ha påverkat resultaten presenteras samt samhälleliga och etiska aspekter diskuteras.

Sammanfattningsvis indikerar slutsatserna att maskininlärning är en effektiv metod att tillämpa för avkodning vid kvantfelskorrektion för tillräckligt stora och välträinade GNN.

Quantum Error Correction of Bit-Flip Errors in Repetition Codes Using Machine Learning
Investigation of the Role of Machine Learning in the Development of Quantum Computers
HUGO MODIN ASKLID
NILS KALMNÄS DRAKENFORS
CAROLINE PAULIS
TOVA HEDMAR
JACOB SANDSTRÖM
EMIL OLOFSSON
Department of Physics
Chalmers University of Technology

Abstract

One of the biggest challenges with quantum computers is minimizing and correcting errors that arise due to the instability of quantum systems. The qubits in the system are sensitive to external factors and disturbances, which can lead to various types of quantum errors, including bit-flip errors. To address this, there are different methods for predicting and correcting quantum errors. Two such methods are Minimum Weight Perfect Matching (MWPM) and machine learning. This report investigates the effectiveness of a trained graph neural network (GNN) as a decoder by comparing the rate of bit-flip errors in repetition codes to MWPM on IBM's quantum computers.

The method is based on using Qiskit to construct a quantum circuit capable of detecting bit-flip errors without collapsing the state of the qubits, thereby obtaining the error rate in the form of syndromes. Data collected from runs on the quantum computer is used to train a GNN and is also applied to MWPM for code distances $d \in [3, 21]$. The performance of the decoders is then compared.

The results showed that a larger GNN with seven graph convolutional layers generally performed better than both MWPM and a smaller network with three graph convolutional layers for all investigated code distances, except for $d = 7$. MWPM also outperformed the smaller network, which generally had the lowest performance among the decoders studied. The method and results are analyzed based on theory and the research question to provide a broader understanding of how a trained GNN and MWPM behave for different code distances. Relevant sources of error that may have affected the results are also presented, along with a discussion of societal and ethical aspects.

In conclusion, the findings indicate that machine learning is an effective method to apply for decoding in quantum error correction, given sufficiently large and well-trained GNNs.

Ordlista

Nedan följer begrepp som är relevanta i rapporten.

Alvis	Datorkluster med GPU:er anpassat för maskininlärningsforskning.
Ancillabit	Kvantbit som en stabiliseringsmätning utförs på.
Avkodning	Processen att analysera syndrom för att avgöra typen av korrektion.
Batch	En datamängd som nätverket testas på innan parametrarna uppdateras.
Bias	Beskriver hur väl en modell kan förutsäga resultat.
Blochsfär	Geometrisk representation av tillståndet hos en kvantbit.
Dekoherens	Att ett kvantsystem förlorar sitt kvanttillstånd på grund av störningar för omgivningen.
Epok	En omgång av att nätverket tränas på all träningsdata.
Feature-vektor	Vektor innehållande en nods egenskaper i en graf.
GNN	(Graf-neuralt nätverk) Neuralt nätverk vars input är grafer.
Graf-faltningslager	Del av neuralt nätverk som uppdaterar en nods egenskaper utifrån information hos grannoder.
Klassificeringsgren	Del av ett neuralt nätverk som klassificerar datan.
Koddistans	Minsta antalet fel som ändrar en logisk bit till en annan.
Kollaps	Kvantfysikaliskt begrepp som innebär att informationen i ett kvanttillstånd går förlorad vid mätning av det.
Kvantbit	Kvanttillstånd med två komponenttillstånd i superposition.
Kvantchip	Processorchip med kvantbitar i en kvantdator.
Kvantdator	Beräkningsenhet bestående av kvantbitar som utnyttjar kvantfysikaliska fenomen.
Kvantfel	Oavsiktlig förändring av ett kvanttillstånd.
Kvantfelskorrektion	Metod för att korrigera kvantfel i kvantdatorer.
Kvantkrets	Krets innehållande kvantlogiska kretskomponenter och kvantbitar.
Kvantlogisk grind	Grind som används som operatorer på kvantbitar i kvantkretsar.
Kvanttillstånd	Kvantmekanisk beskrivning av tillståndet hos ett fysikaliskt system.
Logisk kvantbit	Kvantbitarna i en repetitionskod som tillsammans utgör en informationsbit.

MWPM	(Minimum Weight Perfect Matching) En algoritm för avkodning.
Neuralt nätverk	Maskininlärningsmodell där information färdas genom lager bestående av noder.
Paulimatris	Operator som kan tillämpas på ett kvanttillstånd för att matematiskt beskriva en förändring av det.
Qiskit	Program av IBM Quantum för konstruktion och körning av kvantkretsar.
Repetitionskod	Bit som har expanderats till flera i syfte att skapa redundans.
Sammanflätning	Kvantfysikaliskt fenomen där kvanttillståndet för sammanflätade fysikaliska system ej kan beskrivas oberoende av varandra.
Sannolikhetsamplitud	Kvantfysikalisk sannolikhet vars storlek i kvadrat motsvarar observerad sannolikhet.
Sparse Blossoms	En vanlig implementering av MWPM.
Stabiliseringsmätning	Extraktion av information hos kvantbitar utan att mäta dem och således kollapsa deras kvanttillstånd.
Superposition	Att ett kvantsystem kan befinna sig i flera tillstånd samtidigt.
Syndromextraktion	Mätning av ancillabitar för att extrahera information.
Ytkod	Kvantkod som kan korrigera för både bit- och fasfel.
Överanpassning	När ett nätverk anpassar sig för bra till träningsdata så att det underpresterar för ny data.

Innehåll

1 Inledning

Det moderna samhället präglas av accelererande teknisk utveckling som kräver allt mer effektiva och omfattande beräkningsmetoder. För att möta dessa behov behövs således allt snabbare datorer. Dagens mest kraftfulla klassiska superdatorer erbjuder mycket stor beräkningskraft och används för att lösa de flesta stora tekniska utmaningar som samhället står inför. Men det finns komplexa problem som inte ens dessa datorer kan lösa, problem som i praktiken hade tagit väldigt lång tid med den teknik som finns tillgänglig idag. Några exempel är modellering av molekylära processer inom medicin och läkemedel, lösningar av komplexa optimeringsproblem med ekonomisk betydelse samt kryptografi och datasäkerhet [1]. Det är i frågeställningarna av hur man löser sådana problem som kvantdatorer kan komma till användning.

Likt hur klassiska datorer är uppbyggda av bitar, utgörs kvantdatorer av kvantbitar [2]. Klassiska bitar kan endast befinna sig i ett 0- eller 1-tillstånd, medan kvantbitar befinner sig i en superposition av dessa. Det är denna egenskap som ligger till grund för kvantdatorers beräkningspotential. Precis som i det klassiska fallet krävs flera bitar för att ett faktiskt problem ska kunna lösas, men på grund av kvanttillståndets superpositionsegenskaper behövs betydligt färre kvantbitar för att bygga upp en effektiv kvantdator [2].

Det huvudsakliga problemet med kvantbitar är att de utgörs av kvantmekaniska system som är instabila och påverkas lätt av sin omgivning [3]. På grund av detta uppstår ofta fel i kvantbitarna, det vill säga att den ursprungliga lagrade informationen i systemet inte nödvändigtvis är densamma över tid. För att skapa redundans mot sådana fel expanderar man bitarna till flera, vilka utgör så kallade repetitionskoder. Däremot går det inte direkt att lista ut var eventuella fel har skett, då mätning av systemet skulle kollapsa kvanttillståndet och informationen gå förlorad. För att kvantdatorer ska vara praktiskt användbara behövs det därmed en metod för att upptäcka och korrigera dessa fel utan att mäta kvantsystemet. Metoden att göra just detta kallas för kvantfelskorrektion [3].

Huvudprincipen med kvantfelskorrektion är att extrahera information om tillståndet hos kvantsystemet för att göra en förutsägelse om vilka kvantbitar som måste korrigeras samt på vilket sätt, och slutligen utföra själva korrektionen. Steget att förutsäga vilken korrigering som ska ske kallas avkodning och kan utföras på olika sätt [4], där den mest använda klassiska avkodningsalgoritmen är en metod som kallas Minimum Weight Perfect Matching (MWPM) [5]. Denna metod är inte optimal och därför är det relevant att försöka hitta ännu mer effektiva avkodare. På grund av maskininlärningens alltmer betydande roll i samhället studerar projektet om denna teknik är rimlig att utnyttja vid avkodning, i detta fall i form av graf-neurala nätverk (GNN).

1.1 Mål

Denna rapport ämnar att undersöka hur effektivt ett tränat graf-neuralt nätverk är som avkodare för kvantfelskorrektion baserat på repetitionskoder. Detta görs genom att jämföra andelen uppkomna bitfel i kvantmekaniska repetitionskoder av olika längd hos IBM:s kvantdatorer, utan och med det graf-neurala nätverket som avkodare. För att avgöra det graf-neurala nätverkets effektivitet jämförs det med metoden Minimum Weight Perfect Matching.

1.2 Syfte

Syftet med denna undersökning är att få en inblick i hur väl maskininlärning kan utnyttjas i utvecklingen av kvantdatorer. För att kvantdatorer överhuvudtaget ska vara praktiskt användbara behövs effektiv kvantfelskorrektion. Genom att undersöka hur väl maskininlärningsalgoritmer står sig i förhållande till klassiska metoder för att upptäcka och korrigera fel i kvantbitar, går det att dra slutsatser kring vilka metoder som är relevanta att vidareutveckla i den fortsatta forskningen inom området. Ytterligare ett syfte med projektet är att samla in relevant mätdata från kvantdatorer som förhoppningsvis kan användas i framtida studier, både för att utvärdera kvantdatorers utveckling och maskininlärning som verktyg vid kvantfelskorrektion.

1.3 Avgränsningar

Inledningsvis gjordes avgränsningen att enbart studera repetitionskoder. Ytkoder tar däremot hänsyn till att både korrigera både bit- och fasfel samtidigt, vilket gör dem mer användbara i verkligheten eftersom de generellt ger upphov till en lägre felsannolikhet. Dessa tillför dock ett extra lager av komplexitet, då det krävs fler kvantbitar för att avkoda mätdatan. Detta ger upphov till längre simuleringstider och flera olika stabiliseringsmätningar för att korrigera båda typerna av fel. Inom ramen för arbetet ansågs detta vara för tids- och resursmässigt komplicerat, och därför studeras endast repetitionskoder. En mer utökad undersökning skulle kunna vidareutveckla de metoder som presenteras i denna rapport för att inkorporera ytkoder.

Vidare syftar denna rapport till att undersöka specifikt bitfel i repetitionskoder. Då både bitfel och fasfel kan uppstå, är det rimligt att studera deras inverkan i repetitionskoder i förhållande till varandra. Anledningen till att detta inte görs är på grund av att ett liknande kandidatarbete har som syfte att träna ett graf-neuralt nätverk som avkodare för repetitionskoder, men med korrigering av just fasfel. På så sätt kan resultaten i båda undersökningarna jämföras i vidare forskning för att dra slutsatser kring hur effektiva graf-neurala nätverk är på att korrigera för båda typer av fel i repetitionskoder.

I detta arbete studeras även repetitionskoder med enbart vissa koddistanser, nämligen udda från och med 3 till och med 21. Koddistanser kortare än 3 har ingen praktisk tillämpbarhet, då dessa inte kan korrigera för uppstådda fel. Den övre begränsningen på 21 beror både på arbetets tidsbegränsning och att körningar vid dessa koddistanser var tids- och resurskrävande. Syftet med projektet uppnås utan att behöva studera väldigt långa koddistanser, och man kan ändå dra slutsatser kring maskininlärningens betydelse utifrån resultaten. Därför studeras endast detta intervall av längder på repetitionskoder.

Ytterligare finns det många olika företag som forskar inom utvecklingen av kvantdatorer och specifikt kvantfelskorrektion. Dessa använder sig av olika sätt att konstruera och extrahera information ur kvantbitar, samt skild programvara och dokumentation. Att generalisera denna undersökning för att kunna anpassa metoderna och resultaten till utvecklingen av kvantdatorer skulle i allmänhet vara väldigt svårt och krävande. I detta projekt fick projektgruppen tillgång till att göra mätningar på IBM:s kvantdatorer, och därför utgår projektet från detta företags programvara och dokumentation, samt hur deras kvantchip är konstruerade. De framtagna resultaten överensstämmer därmed inte nödvändigtvis för kvantfelskorrektion i allmänhet.

Till sist utvärderas hur väl det graf-neurala nätverket kan användas för kvantfelskorrektion genom att jämföra med MWPM. Denna jämförelse är inte nödvändigtvis det enda sättet att avgöra dess effektivitet, men utförs av den anledningen att MWPM är en allmänt tillämpad avkodningsalgoritm som inte använder sig av maskininlärning. Däremot är avkodaren inte optimal, vilket bland annat är varför man försöker använda sig av maskininlärning i förhoppning om att kunna utföra ännu mer effektiv kvantfelskorrektion. Detta är en avgränsning för tolkningen av resultatet, eftersom slutsatserna kring det graf-neurala nätverkets effektivitet rimligtvis är beroende av vad man jämför med.

2 Teori

I detta avsnitt presenteras den bakomliggande teori som ligger till grund för undersökningen.

2.1 Kvantteori

I detta delavsnitt redogörs de kvantfysikaliska fenomen och den grundläggande matematik som beskriver områdena som studeras i denna undersökning. Kvantbitar samt kvantfel introduceras och kvantfysiken bakom de steg som behövs för kvantfelskorrektion beskrivs.

2.1.1 Kvantbitar

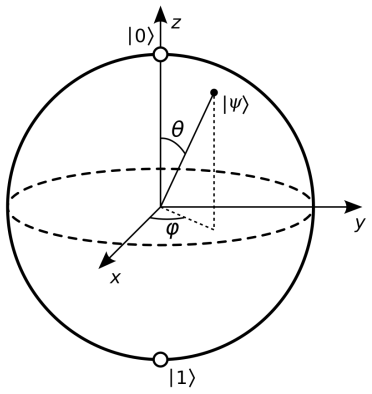
En klassisk bit beskrivs av att den kan vara i tillstånd 0 eller 1. En *kvantbit* befinner sig istället i en superposition av dessa och beskrivs kvantmekaniskt av tillståndet

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

där α och β är sannolikhetsamplituder som uppfyller $|\alpha|^2 + |\beta|^2 = 1$ [6]. Detta kvanttillstånd kan geometriskt representeras av en punkt på den så kallade Blochsfären i Figur 1 och beskrivs i så fall enligt

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle\,, \tag{2}$$

där θ och ϕ är sfäriska vinklar, det vill säga att $\alpha = \cos\frac{\theta}{2}$ och $\beta = e^{i\phi}\sin\frac{\theta}{2}$ i ekvation (1) [7].



Figur 1: Blochsfären: en geometrisk representation av ett kvantbitssystem [7].

Matematiskt beskrivs alltså ett kvantbitsystem av en superposition av komponenttillstånden $|0\rangle$ och $|1\rangle$, men rent fysikaliskt kan det representeras av exempelvis fotoner, elektroner eller supraledande kretsar [2].

2.1.2 Kvantfel

Kvantsystem är instabila och påverkas lätt av sin omgivning, bland annat genom värme- och elektriska fluktuationer, vilket kan orsaka fel i kvanttillståndet. För att minimera felen kyls bitarna ned till temperaturer nära absoluta nollpunkten, men avvikelser uppstår ändå [2]. Teoretiskt kan fel i kvantbitstillståndet tolkas som rotationer på Blochsfären, vilket

matematiskt beskrivs genom att tillämpa en unitär operation $U(\delta\theta, \delta\phi)$ på tillståndet [4] enligt

$$U(\delta\theta, \delta\phi)\,|\Psi\rangle = \alpha_I \mathbb{1}\,|\psi\rangle + \alpha_X X\,|\psi\rangle + \alpha_Z Z\,|\psi\rangle + \alpha_Y Y\,|\psi\rangle\,, \tag{3}$$

där $\alpha_I, \alpha_X, \alpha_Z$ samt α_Y är konstanter och $\mathbb{1}, X, Y$ samt Z är Paulimatriser som ges av

$$\mathbb{1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},\ Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix},\ Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Genom att notera att $Y = XZ$ upp till en fasskillnad kan ekvation (3) skrivas om till

$$U(\delta\theta, \delta\phi)\,|\Psi\rangle = \alpha_I \mathbb{1}\,|\psi\rangle + \alpha_X X\,|\psi\rangle + \alpha_Z Z\,|\psi\rangle + \alpha_{XZ} XZ\,|\psi\rangle\,, \tag{4}$$

där även α_{XZ} är en konstant. Eftersom en förändring av ett kvantbitstillstånd kan beskrivas enbart med X och Z, räcker det att ta hänsyn till två typer av kvantfel [4], nämligen bit- och fasfel. X-fel kan klassiskt tolkas som bitflips och påverkar ett kvantbitstillstånd enligt

$$X\,|\psi\rangle = \alpha X\,|0\rangle + \beta X\,|1\rangle = \alpha\,|1\rangle + \beta\,|0\rangle\,, \tag{5}$$

medan Z-fel kan tolkas som fasflips och istället påverkar tillståndet enligt

$$Z\,|\psi\rangle = \alpha Z\,|0\rangle + \beta Z\,|1\rangle = \alpha\,|0\rangle - \beta\,|1\rangle\,. \tag{6}$$

Denna rapport fokuserar enbart på att korrigera bitfel. Fasfel kommer därmed framöver inte att tas hänsyn till i samma utsträckning, vilket beror på att ett annat kandidatarbete inom samma område har till syfte att undersöka just fasfel.

2.1.3 Repetitionskoder

Det går inte att avgöra huruvida ett fel har uppstått genom att studera tillståndet hos endast en bit då man inte vet dess ursprungliga tillstånd, utan det behövs någon typ av redundans. Klassiskt uppnås detta genom att expandera en bit till flera enligt

$$\{0, 1\} \xrightarrow{\text{bitkodare}} \{00\ldots, 11\ldots\}, \tag{7}$$

vilka tillsammans bildar en så kallad logisk bit [4]. Detta kallas för en *repetitionskod*. Under antagandet att ett fel uppstår med relativt låg sannolikhet, går det att avgöra vad det ursprungliga tillståndet troligtvis var om ett bitfel skulle uppstå. Det görs genom att helt enkelt notera vilket tillstånd i repetitionskoden som är mest förekommande. I fallet med en trebit-repetitionskod ($\{000, 111\}$), kan endast ett bitfel upptäckas för att korrigera den logiska biten [4]. Längre repetitionskoder tillåter högre redundans, men kräver också fler faktiska bitar. Generellt kan en repetitionskod betecknas $[n, k, d]$, där n är antalet bitar per logisk bit, k är den ursprungliga bitens längd och $d = 2t + 1$ är koddistansen som beskriver det minsta antalet fel som ändrar en logisk bit till en annan, varvid t är antalet fel som koden kan korrigera [4]. En trebit-repetitionskod betecknas alltså $[3, 1, 3]$.

Likt det klassiska fallet kan även kvantbitar expanderas i form av repetitionskoder till en logisk kvantbit genom sammanflätning enligt

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \xrightarrow{\text{kvantbitkodare}} |\psi\rangle_L = \alpha\,|0\rangle_L + \beta\,|1\rangle_L = \alpha\,|00\ldots\rangle + \beta\,|11\ldots\rangle\,, \tag{8}$$

vilket kan utnyttjas för att skapa redundans [4]. För att differentiera kvantrepetitionskoder från klassiska betecknas här dessa $[[n, k, d]]$, där n är antalet kvantbitar, k är antalet logiska kvantbitar och d är koddistansen [4].

2.1.4 Tröskelvärde för repetitionskoder

Även om repetitionskoder kan användas för att minska effekten av bitfel genom redundans, är deras praktiska användbarhet begränsad av felfrekvensen i det underliggande fysiska systemet. För att felkorrigering ska vara effektiv, säger tröskelsatsen (threshold theorem) att den individuella felsannolikheten per kvantoperation p måste vara lägre än ett visst tröskelvärde p_{th} [8]. Om $p < p_{th}$ kan logiska fel bättre korrigeras genom att öka koddistansen d. För det omvända fallet $p > p_{th}$ växer antalet fel snabbare än vad koden kan korrigera i takt med ökad koddistans, vilket i sin tur ökar felfrekvensen för den logiska kvantbiten. Ett teoretiskt idealt tröskelvärde är $p_{th} = 50\,\%$, men i praktiken är tröskelvärdet för kvantrepetitionskoder lägre — mellan 11 % och 17,2 % [9]. Det är alltså viktigt att felsannolikheten per individuell kvantbit är relativt låg för att längre repetitionskoder ska kunna bidra till bättre redundans.

2.1.5 Stabiliseringsmätningar och syndromextraktion

Problemet med kvantbitar är att man helt enkelt inte kan studera varje enskild bit för att avgöra vad den ursprungliga logiska kvantbiten bör vara. När en mätning utförs på ett kvantsystem, kollapsar tillståndet till ett av flera möjliga utfall och den ursprungliga informationen går förlorad [6]. Det krävs en annan typ av mätning för att avgöra vilken typ av korrigering som behövs, nämligen så kallade *stabiliseringsmätningar* [4]. Detta går ut på att applicera Z-operatorer enligt (6) till vardera av två närliggande kvantbitar i en logisk kvantbit. Om en logisk kvantbit $|\psi\rangle_L$ utgörs av två kvantbitar (1 och 2) och inte utsätts för bitfel, appliceras en Z_1Z_2-operator på tillståndet enligt

$$Z_1Z_2 |\psi\rangle_L = Z_1Z_2(\alpha |00\rangle + \beta |11\rangle) = (+1) |\psi\rangle_L , \tag{9}$$

vilket ger ett egenvärde +1, det vill säga att den logiska kvantbiten lämnas oförändrad [4]. Låt säga att den andra kvantbiten utsätts för ett bitfel X_2. I så fall ger Z_1Z_2-operatorn upphov till följande:

$$Z_1Z_2X_2 |\psi\rangle_L = Z_1Z_2(\alpha |01\rangle + \beta |10\rangle) = (-1)X_2 |\psi\rangle_L , \tag{10}$$

det vill säga ett egenvärde -1 som representerar att någon av kvantbitarna utsätts för ett bitfel. För att upptäcka om ett bitfel har skett, introduceras en så kallad *ancillabit* $|0\rangle_A$ som används för att utföra stabiliseringsmätningen [4]. Denna process kallas *syndromextraktion* och transformerar kvanttillståndet som eventuellt utsatts för något bitfel E enligt

$$\begin{aligned} E |\psi\rangle_L \otimes |0\rangle_A \xrightarrow{\text{syndromextraktion}} & \frac{1}{2}(\mathbb{1}_1\mathbb{1}_2 + Z_1Z_2)E |\psi\rangle_L \otimes |0\rangle_A \\ & + \frac{1}{2}(\mathbb{1}_1\mathbb{1}_2 - Z_1Z_2)E |\psi\rangle_L \otimes |1\rangle_A . \end{aligned} \tag{11}$$

Ur syndromextraktionen i (11), noteras att en mätning av ancillabiten, vilket kallas syndromet, deterministiskt ger 0 eller 1 om stabiliseringsmätningen gav egenvärdet +1 respektive -1. Syndromet beror alltså på vilken typ av bitfel som kvanttillståndet utsattes för enligt Tabell 1.

Bitfel	Syndrom
$\mathbb{1}_1\mathbb{1}_2$	0
$\mathbb{1}_1 X_2$	1
$X_1\mathbb{1}_2$	1
$X_1 X_2$	0

Tabell 1: Syndrom beroende på typ av bitfel för en tvåkvantbit-repetitionskod.

Syndromextraktionen kan generaliseras på en repetitionskod med godtycklig längd n, där en ancillabit $|0\rangle_{A_{i,i+1}}$ introduceras för varje par av två närliggande kodande kvantbitar (i och $i+1$) i en logisk kvantbit [4]. I så fall uttrycks syndromextraktionen enligt

$$\begin{aligned} E|\psi\rangle_L \otimes |0\rangle_{A_{i,i+1}} \xrightarrow{\text{syndromextraktion}} & \frac{1}{2}(\mathbb{1}^{\otimes n} + Z_i Z_{i+1}) E|\psi\rangle_L \otimes |0\rangle_{A_{i,i+1}} \\ & + \frac{1}{2}(\mathbb{1}^{\otimes n} - Z_i Z_{i+1}) E|\psi\rangle_L \otimes |1\rangle_{A_{i,i+1}}, \end{aligned} \tag{12}$$

för $1 \leq i \leq n-1$. För en n-kvantbit-repetitionskod behövs alltså totalt $2n-1$ faktiska kvantbitar för att utgöra en logisk kvantbit (n kodande kvantbitar och $n-1$ ancillabitar). För exempelvis en trekvantbit-repetitionskod ($\{|000\rangle, |111\rangle\}$) behövs totalt fem kvantbitar och syndromet beror på typen av bitfel enligt Tabell 2. De syndrom som bara utgörs av nollor och där inga bitfel har uppstått kallas triviala.

Bitfel	Syndrom
$\mathbb{1}_1\mathbb{1}_2\mathbb{1}_3$	00
$\mathbb{1}_1\mathbb{1}_2 X_3$	01
$\mathbb{1}_1 X_2\mathbb{1}_3$	11
$X_1\mathbb{1}_2\mathbb{1}_3$	10
$\mathbb{1}_1 X_2 X_3$	10
$X_1\mathbb{1}_2 X_3$	11
$X_1 X_2\mathbb{1}_3$	01
$X_1 X_2 X_3$	00

Tabell 2: Syndrom beroende på typ av bitfel för en trekvantbit-repetitionskod.

2.1.6 Avkodning

Syndrommätningarna används för att slutligen utföra själva kvantfelskorrektionen. Processen att analysera syndromet för att bestämma vilken typ av korrektion som måste utföras kallas för *avkodning* [4]. Målet med korrektionen är att återskapa det ursprungliga logiska tillståndet $|\psi\rangle_L$ innan det utsattes för fel E genom att tillämpa en unitär operation R [4]. För en lyckad korrektion ska alltså avkodningen erhålla ett R som uppfyller

$$RE|\psi\rangle_L = (+1)|\psi\rangle_L. \tag{13}$$

Ekvation (13) uppfylls exempelvis om $R = E^\dagger$, eftersom $E^\dagger E = \mathbb{1}$, vilket innebär ett egenvärde $+1$ [4]. Själva avkodningen kan ske på flera sätt, men i detta arbete jämförs två olika typer, nämligen genom att använda en metod som kallas *Minimum Weight*

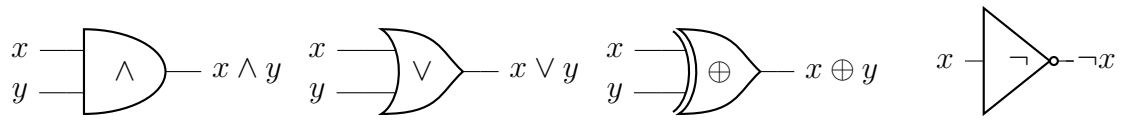
Perfect Matching (MWPM) och genom att träna ett *graf-neuralt nätverk* (GNN), vilka presenteras i delavsnitt 2.3 respektive 2.4.

2.2 Kvantkretsar

I detta delavsnitt presenteras de operationer och komponenter som används i kvantkretsar för att illustrera och utföra kvantfelskorrektion. Först introduceras klassiska booleska kretsar, som består av logiska grindar, följt av kvantkretsar som byggs upp med kvantlogiska grindar och mätoperationer.

2.2.1 Booleska operationer

Kretsar är modeller som tillämpas inom datavetenskap för att skicka information genom en serie av grindar, där vardera grind utgör en funktion på ingångsvärdena [10][11]. Så kallade *booleska kretsar* är vanligt förekommande och innehåller logiska grindar som utgör booleska operationer [10]. Exempel på sådana är AND- ($\wedge$), OR- ($\vee$), XOR- ($\oplus$) och NOT-grindar ($\neg$), vilka illustreras i Figur 2.



(a) AND-grind. **(b)** OR-grind. **(c)** XOR-grind. **(d)** NOT-grind.

Figur 2: Representation av de logiska grindarna AND, OR, XOR och NOT tillämpade på ingångsbitar x och y.

Grindarna i Figur 2 ger en viss output givet värdet på ingångsbitarna enligt Tabell 3 [12].

xy	**x ∧ y**
00	0
10	0
01	0
11	1

(a) AND.

xy	**x ∨ y**
00	0
10	1
01	1
11	1

(b) OR.

xy	**x ⊕ y**
00	0
10	1
01	1
11	0

(c) XOR.

x	**¬x**
0	1
1	0

(d) NOT.

Tabell 3: Output av de booleska operationerna AND, OR, XOR och NOT tillämpade på ingångsbitar x och y.

2.2.2 Kvantlogiska grindar och kretskomponenter

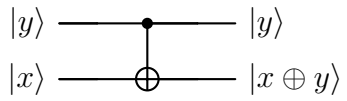
I kvantmekaniska kretsar, till skillnad från klassiska, används *kvantlogiska grindar* för att påverka tillståndet hos kvantbitar [11]. De fungerar som kvantmekaniska motsvarigheter till klassiska logiska grindar, men är alltid unitära, till skillnad från de klassiska. Det innebär att de är reversibla och bevarar sannolikhetsamplituden i kvanttillståndet [13].

Exempel på kvantlogiska grindar är Paulimatriserna som presenteras i delavsnitt 2.1.2, men även CNOT (Controlled NOT)-grinden som illustreras i Figur 3. Denna grind påverkar en så kallad målkvantbit beroende på tillståndet hos en kontrollkvantbit och används ofta

för att skapa kopplingar mellan kvantbitar. En $\mathbb{1}$- respektive X-Paulimatris tillämpas på målkvantbiten $|x\rangle$ beroende på tillståndet hos kontrollkvantbiten $|y\rangle$ enligt

$$|x\rangle \otimes |y\rangle \xrightarrow{\text{CNOT}} |x \oplus y\rangle \otimes |y\rangle = \begin{cases} \mathbb{1}\,|x\rangle \otimes |y\rangle \;\; (|y\rangle = |0\rangle) \\ X\,|x\rangle \otimes |y\rangle \;\; (|y\rangle = |1\rangle) \end{cases}, \tag{14}$$

det vill säga att målkvantbitens tillstånd flippas om kontrollkvantbiten är $|1\rangle$ och annars inte.



Figur 3: Kvantkrets med CNOT-grind tillämpad på målkvantbit $|x\rangle$ och kontrollkvantbit $|y\rangle$, vilket ändrar målkvantbitens tillstånd till $|x \oplus y\rangle$.

För att kunna avläsa resultatet från en kvantberäkning behövs ett sätt att mäta kvantbitarna. För detta används mätoperationen som illustreras i Figur 4. Den kollapsar ett kvanttillstånd $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ på en klassisk bit med hjälp av mätning i Z-basen enligt

$$|\psi\rangle \xrightarrow{\text{mätoperation}} \begin{cases} 0 \;\; (P_0 = |\alpha|^2 = |\,\langle 0|\psi\rangle\,|^2) \\ 1 \;\; (P_1 = |\beta|^2 = |\,\langle 1|\psi\rangle\,|^2) \end{cases},$$

där P_0 är sannolikheten att tillståndet kollapsar till 0 och P_1 är sannolikheten att tillståndet kollapsar till 1 [14]. Kvanttillståndet förintas efter mätningen. Eftersom mätningarna inte är unitära och inte kan reverseras räknas de inte som kvantlogiska grindar i strikt mening [13].

Figur 4: Kvantkrets med mätoperation som kollapsar ett kvanttillstånd $|\psi\rangle$ på en klassisk bit.

I kvantkretsar är det också vanligt att man tillämpar en barriär,  , för att undvika att grindar oavsiktligen kombineras [14]. Kombinationen av kvantlogiska grindar gör det möjligt att konstruera kvantkretsar som kan detektera och hantera fel i kvanttillstånden.

2.2.3 Qiskit

För att utföra kvantfelskorrektion konstrueras en krets som kan verkställa de operationer som presenteras i delavsnitt 2.1. I detta arbete används *Qiskit*, en mjukvara med öppen källkod utvecklad av IBM Quantum som möjliggör konstruktion och körning av kvantkretsar i både simulerade miljöer och på verkliga kvantdatorer [15]. Qiskits indexeringskonvention är att kvantbiten längst upp i kretsen har index 0 och fortsätter nedåt med stigande index [11]. I exempelvis tensorprodukter eller tupler hamnar däremot kvantbiten med index 0 längst till höger av tuplen. Med andra ord indexeras kvantbitar av ordning n som $(q_{n-1}, \ldots, q_0)$, där kvantbit q_0 hamnar högst upp i kretsen medan q_{n-1} hamnar lägst

ned som tidigare beskrivet. Man kan även tillämpa klassiska kretsar som representeras av dubbla linjer. Detta är användbart vid mätning av kvanttillstånd och generera svar i form av klassiska bitar. I Figur 5 illustreras ett exempel på en kvantkrets i Qiskit som utför en stabiliseringsmätning och syndromextraktion av en tvåkvantbit-repetitionskod, där den horisontella axeln representerar tidsflödet genom kretsen.

Figur 5: Kvantkrets över stabiliseringsmätning och syndromextraktion av en tvåkvantbit-repetitionskod. Före första barriären sammanflätas två kodande kvantbitar $q[0]$ och $q[1]$ med hjälp av en CNOT-grind för att bilda en logisk kvantbit (tvåkvantbit-repetitionskod). Efter första barriären tillämpas två CNOT-grindar på en ancillabit $a[0]$ som målkvantbit med respektive kodande kvantbit som kontrollkvantbit. Efter sista barriären mäts slutligen ancillabiten på en klassisk bit $c1$ för att extrahera syndromet. Notera att endast ancillabiten mäts, vilket innebär att tillståndet för de kodande kvantbitarna ej kollapsar.

Samma princip kan tillämpas på längre repetitionskoder, men det kräver fler stabiliseringsmätningar och således fler ancillabitar.

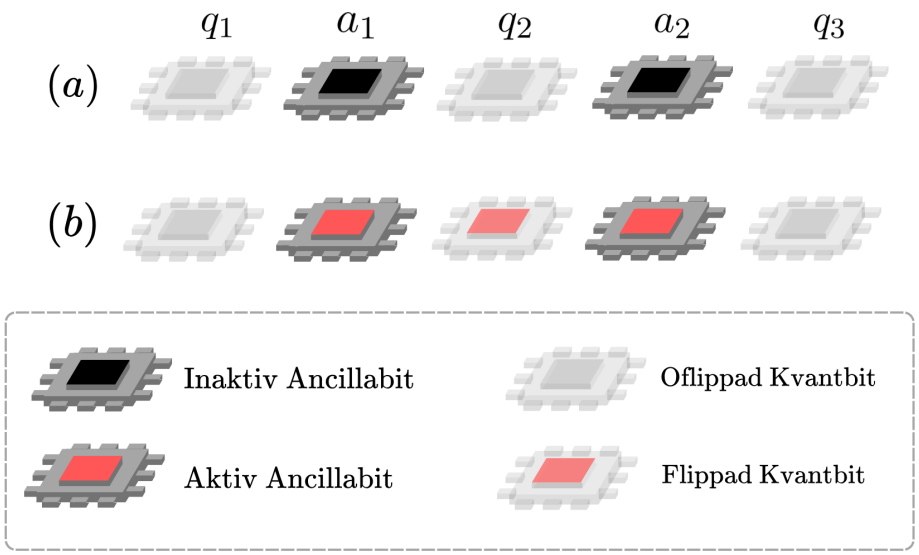
2.3 Minimum-Weight Perfect Matching

I detta avsnitt presenteras en av metoderna som används för syndromavkodning: Minimum-Weight Perfect Matching.

2.3.1 Bitfel och mätfel

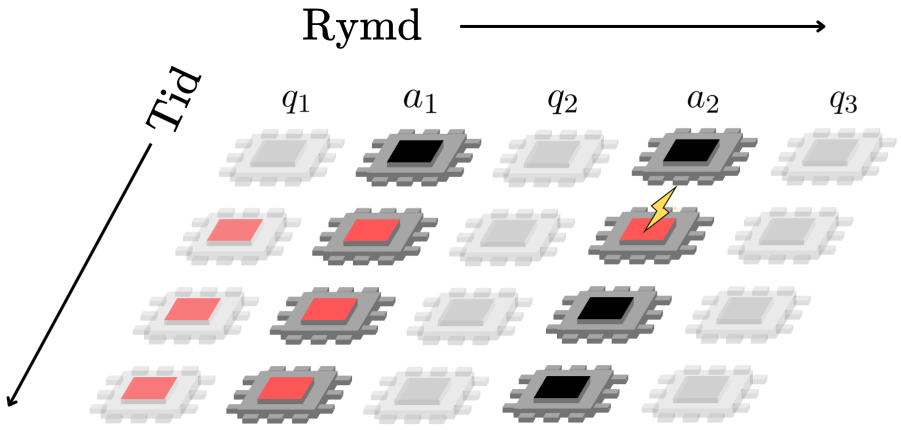
Den idag mest använda algoritmen för syndromavkodning inom kvantfelskorrektion kallas för Minimum-Weight Perfect Matching (MWPM) [5]. Algoritmens funktion är densamma som hos alla syndromavkodare: att givet en uppsättning syndrommätningar hitta den mest sannolika uppsättningen fel hos kvantbitarna som ger upphov till syndromet. Att göra detta helt optimalt kräver dock mycket beräkningsresurser. MWPM har fördelen att den kan implementeras för snabb körning samtidigt som den har en mycket god korrektionsgrad. Med MWPM kan alltså kvantfelskorrektion utföras i en takt som matchar genereringen av syndromdata. [5].

För att underlätta beskrivningen av MWPM betraktas först en trekvantbit-repetitionskod, med tre kodande kvantbitar och två ancillabitar. En schematisk illustration av repetitionskoden ges i Figur 6 nedan.



Figur 6: Schematisk illustration av en trekvantbit-repetitionskod, där två ancillabitar a_1 och a_2 används för att mäta syndrom på närliggande kodande kvantbitar q_1, q_2 och q_3. I (a) visas ett fall utan några bitfel, där båda ancilla-kvantbitarna är inaktiva. I (b) har kvantbit q_2 utsatts för ett bitfel, vilket gör att båda intilliggande ancillabitar a_1, a_2 registrerar ett fel.

I repetitionskoden används oftast ett antal tidssteg, där ancillabitarna mäts flera gånger på rad. Detta gör det möjligt att skilja på faktiska bitfel och rena mätfel som innebär att ancillabitarna visar syndrom trots att inget fel uppstått i kvantbitarna. I Figur 7 visas ett exempel på en trekvantbit-repetitionskod med tre tidssteg.

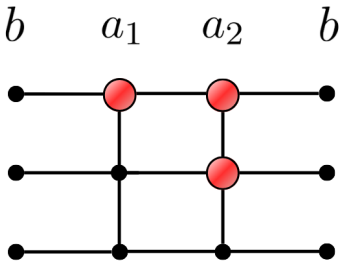


Figur 7: Illustration av en trekvantbit-repetitionskod med tre tidsrepetitioner. I den första tidsrepetitionen uppstår två fel: dels ett *bitfel* på q_1, vilket gör att ancillabit a_1 aktiveras; dels ett *mätfel* på a_2, som aktiveras trots att intilligande kvantbitar är felfria.

Ur denna synvinkel har MWPM en dubbel funktion: dels att upptäcka och rätta bitfel, dels att kunna skilja dessa från rena mätfel.

2.3.2 Grafrepresentation och matchning

För att identifiera och rätta fel använder MWPM en grafrepresentation av repetitionskoden och dess tidssteg. Grafen representerar detektormätningar (ancillabitar) som noder och kvantbitar som kanter mellan noderna i grafen. En punkt markerar en förändring hos en detektor, alltså en ancillabit som går från inaktiv till aktiv eller vice versa. Motsvarande detektorgraf för kretsen i Figur 7 kan ses i Figur 8. Notera de två noderna, motsvarande ancillabitarna a_1 och a_2, och tre kanter, motsvarande de tre kvantbitarna, i varje rad.

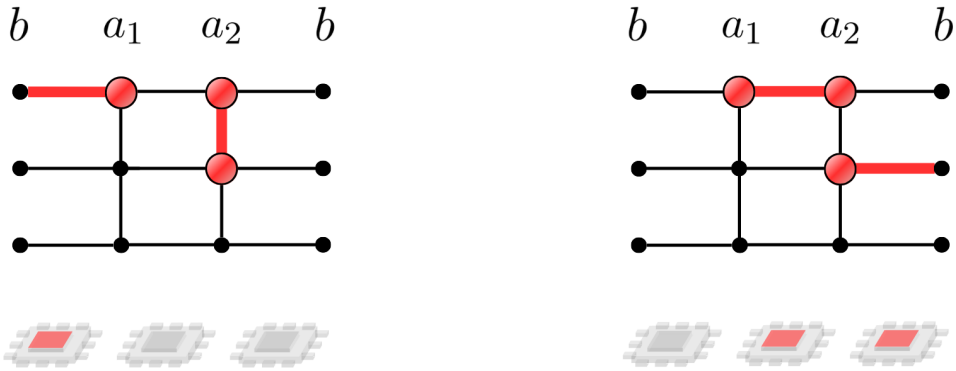


Figur 8: Grafrepresentation av kretsen i Figur 7 och dess tidssteg. Punkter representerar ancillabitar a_1 och a_2 samt virtuella så kallade randnoder b. Horisontella kanter representerar kvantbitar och vertikala representerar tidssteg. Jämfört med Figur 7 är detektorhändelserna (röda punkter) placerade annorlunda. Detta beror på att grafrepresentationen visar *förändringar* i detektorvärden; när ett mätfel sker på a_2 aktiveras och deaktiveras ancillabiten efter varandra, vilket ger två händelser.

MWPM utgår från denna graf för att hitta de mest sannolika felen. Algoritmens mål är att koppla ihop de aktiverade noderna två och två med kanter. Om två noder är kopplade horisontellt med en kant innebär det att det har uppstått ett fel i den mellanliggande kvantbiten. Om två noder är kopplade vertikalt med en kant innebär det ett mätfel i motsvarande detektor. Genom att hitta ett sätt att matcha alla aktiverade ancillabitar med antingen en randnod eller en annan ancillabit, ges en förklaring för alla detektorhändelser. På så sätt har problemet att tolka syndromet reducerats till att hitta en matchning av alla aktiverade ancillabitar.

2.3.3 Minimal-vikt-matchning

Att hitta en matchning av alla noder är dock inte ett entydigt problem. Det finns flera möjliga sätt att para ihop en given mängd av detektorhändelser, ta till exempel grafen i Figur 8. Den producerades av motsvarande fel i Figur 7, men denna information finns inte tillgänglig i grafen. Det bästa att göra är att konstatera ett antal möjliga fall av fel. Två exempel på möjliga matchningar av grafen ges i Figur 9.



(a) I detta fall stämmer matchningen med verkligheten: syndromet på a_1 förutsägs korrekt bero på ett bitfel hos q_1 (kanten till b), medan syndromen på a_2 förutsägs bero på ett mätfel.

(b) I detta fall förutsäger matchningen att syndromet beror på bitfel hos både q_2 och q_3, vilket inte stämmer med verkligheten. Denna matchning bör därför väljas bort av MWPM.

Figur 9: Två möjliga matchningar av grafen i Figur 8. Matchningarna ger upphov till olika förutsägelser och därför olika korrektioner.

Målet med MWPM är därmed inte bara att hitta en matchning, utan att hitta den mest sannolika matchningen för en uppsättning av detektorhändelser. För att åstadkomma detta betraktas istället en viktad graf. Varje kant får en vikt som motsvarar sannolikheten för motsvarande fel, där större vikter innebär lägre sannolikheter. Dessa vikter kan exempelvis tas fram genom korrelationsmatriser eller sättas till ett konstant värde. På så sätt omformuleras problemet till följande: givet en graf av detektorhändelser och vikter för alla kanter, hitta den parvisa matchningen av alla punkter som ger lägst total vikt.

2.3.4 Sparse Blossoms

Slutligen krävs det även en effektiv implementering för att hitta en matchning som löser MWPM-problemet. Detta är ett problem som har arbetats med under en lång tid, med många potentiella lösningar. Svårigheten ligger i att många lösningar använder grafsökningsalgoritmer som kräver stora beräkningsresurser, något som är oundvikligt om målet är att garantera en helt optimal lösning. Samtidigt måste implementeringen vara tillräckligt effektiv för att kunna avkoda syndrommätningar i realtid allteftersom de produceras. Denna avvägning mellan träffsäkerhet och effektivitet är central i designen av MWPM-avkodare.

Den idag mest använda implementeringen av MWPM kallas för Sparse Blossoms [5]. Med både hög träffsäkerhet och snabb körningstid är den ett utmärkt val för avkodare. I programmeringsspråket Python finns algoritmen tillgänglig via paketet `pymatching`, där funktioner för att konstruera en detektorgraf, välja vikter och hitta själva matchningen har skapats av författarna. På så sätt kan hela algoritmen beskriven i detta avsnitt enkelt köras med ett fåtal kodrader i Python.

2.4 Maskininlärning

I detta avsnitt presenteras den andra metoden för syndromavkodning som undersöks, nämligen maskininlärning. Grunden av maskininlärning är att använda data och optimeringsmetoder för att förutspå framtida data [16]. I detta projekt används övervakad inlärning, vilket typiskt är en beslutsprocess som utifrån given data framställer ett resultat (i detta fall en felfunktion som beskriver hur fel resultatet är) samt en optimeringsmetod som försöker minska felet av framtida beslut. Hur stora justeringar som görs i modellen vid varje iteration styrs av en inlärningshastighet, vilket är en central parameter i träningsprocessen. En vanlig beslutsprocess för maskininlärning är neurala nätverk. Dessa härmar uppbyggnaden av neuroner i en mänsklig hjärna genom kopplade lager av viktade noder som datan skickas genom för att erhålla ett slutgiltigt resultat [17].

2.4.1 Graf-neurala nätverk

Den specifika typen av neurala nätverk som används i denna undersökning är så kallade graf-neurala nätverk (GNN). Som namnet antyder är GNN anpassade för att tillämpas på grafer, vilka är vanligt förekommande modeller inom flera områden, exempelvis molekylära strukturer, relationer i sociala nätverk, trafikflöden och rekommendationssystem [18]. GNN utnyttjar maskininlärning för att dra slutsatser och förutsäga egenskaper hos grafer. I detta projekt bearbetas de grafer som genereras av syndromdata från ancillabitarna i repetitionskoden (se delavsnitt 2.3.2). Funktionen hos ett GNN i detta sammanhang blir densamma som hos MWPM; att givet en graf av syndrommätningar, hitta den mest sannolika uppsättningen fel hos kvantbitarna som ger upphov till syndromet.

Till skillnad från andra neurala nätverk, som oftast förutsätter att datan har en fast och regelbunden struktur, är graf-neurala nätverk anpassade för att arbeta med grafer där kopplingarna mellan datapunkterna varierar [18]. Den inlärningsmekanism som GNN bygger på kallas ofta för *message passing*, där varje nod uppdaterar sitt tillstånd genom att ta emot information från sina grannar i grafen [18]. På så sätt kan nätverket uppfatta lokala mönster i strukturen, vilket gör dem särskilt användbara för analys av syndromdata [18]. Genom denna process konstruerar nätverket successivt vektorrepresentationer för noderna baserat på både egna och omgivande egenskaper [18].

2.4.2 GNN-arkitektur

För att kunna förutsäga egenskaper hos grafer använder GNN en speciell arkitektur som är baserad på graf-uppbyggnaden. Arkitekturen kan variera beroende på tillämpningen, men i denna undersökning används en arkitektur utvecklad av Lange et al. [19]. Den använder följande lager:

- **Inputlager**: Detta lager innehåller alla noder och kanter hos den ursprungliga grafen som ska processeras. Grafens noder har vissa givna egenskaper, representerade med en så kallad feature-vektor och kanter med vikter. I fallet med syndromavkodning är egenskaperna hos en nod exempelvis huruvida motsvarande ancillabit är aktiverad eller inte. Kanterna motsvarar felsannolikheten hos kvantbitar.

- **Graf-faltningslager** (graf-konvolutionslager): Efter inputlagret följer ett antal av dessa lager, vars funktion är att uppdatera egenskaperna hos noderna genom en aggregering av egenskaperna hos dess närliggande noder (message passing). Namnet

kommer från att aggregeringen är baserad på graf-faltning, en analogi av den inom analysen förekommande faltningsoperationen, men tillämpat på grafer. Beräkning av graf-faltningen brukar förenklas på olika sätt när GNN implementeras i kod. I den implementering som detta projekt använder (PyTorch GraphConv) utnyttjas följande aggregeringsformel:

$$\mathbf{x}_i' = \mathrm{ReLU}\left(\mathbf{W}_1\mathbf{x}_i + \mathbf{W}_2 \sum_{j\in\mathcal{N}(i)} e_{i,j}\mathbf{x}_j\right), \tag{15}$$

där $\mathbf{x}_i$ samt $\mathbf{x}_i'$ representerar nodens gamla respektive nya vektor-värde, $\mathcal{N}(i)$ anger grannarna till nod i, $e_{i,j}$ anger kant-vikterna mellan nod i respektive j och $\mathbf{W}_1$ samt $\mathbf{W}_2$ anger de två vikt-matriserna som kan tränas av nätverket [20]. Aktiveringsfunktionen ReLU (Rectified Linear Unit) sätter negativa värden till noll för att modellen ska lära sig icke-linjära samband.

- **Poolinglager**: Eftersom graf-faltningslagrena endast förändrar feature-vektorerna hos grafens noder, kommer output från dessa lager fortfarande att vara en graf. För att reducera antalet frihetsgrader och samla information om hela grafen används därför så kallade *poolinglager*. I detta projekt används en global medelvärdespooling som ger medelvärdet av alla nodernas feature-vektorer [21]. För en graf med N noder som alla har respektive feature-vektorer $\mathbf{x}_i$, ges alltså output från pooling-lagret av feature-vektorn

$$\mathbf{x}' = \frac{1}{N}\sum_{i=0}^{N}\mathbf{x}_i. \tag{16}$$

- **Klassificeringsgren**: Efter poolinglagret, som sammanfattar hela grafens information i en vektor, används *klassificeringsgrenar*, en för varje logisk felklass. En logisk felklass representerar en grupp av fysiska fel som påverkar den logiska informationen på samma sätt. För repetitionskoder finns två felklasser; bit- och fasfel, där endast den förstnämnda felklassen studeras i detta projekt. Varje gren består av ett flerskiktat perceptronnätverk, även kallat Multi-Layer Perceptron (MLP), vilket är en typ av neuralt nätverk som består av flera klassificeringslager, även kallat dense layers [22]. Ett klassificeringslager innebär att varje nod är kopplad till samtliga noder i föregående lager, vilket gör att all information från ett lager kan påverka beräkningarna i nästa [22]. Storleken på lagret definieras som antalet noder. Precis som i fallet för graf-faltningslagret används en ReLU-aktiveringsfunktion mellan klassificerings-lagren. Syftet med dessa grenar är att att tolka grafens samlade information och göra binära klassificeringar, det vill säga avgöra om ett visst logiskt fel har inträffat eller inte.

- **Outputlager**: Det sista lagret i varje klassificeringsgren är ett klassificeringslager som ger ett logitvärde, vilket är ett numeriskt mått som modellen använder för att avgöra sannolikheten att ett visst logiskt fel har inträffat. För att tolka dessa logitvärden används en sigmoidfunktion σ definierad enligt

$$\sigma(z) = \frac{1}{1+e^{-z}}, \tag{17}$$

där z är logitvärdet som modellen producerar för en viss klass. Sigmoidfunktionen omvandlar detta till en sannolikhet mellan 0 och 1. Den slutliga binära klassificeringen för varje logisk felklass görs genom att jämföra sannolikheten med ett tröskelvärde, i detta fall 0,5:

$$\hat{y}_i = \begin{cases} 1 \text{ om } \sigma(z_i) > 0{,}5 \\ 0 \text{ annars} \end{cases}, \tag{18}$$

där $\hat{y}_i$ är den predikterade etiketten för logisk felklass i och z_i är logitvärdet för just den klassen. Om $\hat{y}_i = 1$ förutspår modellen att ett fel i klass i har inträffat, medan $\hat{y}_i = 0$ betyder att inget sådant fel förväntas ha inträffat.

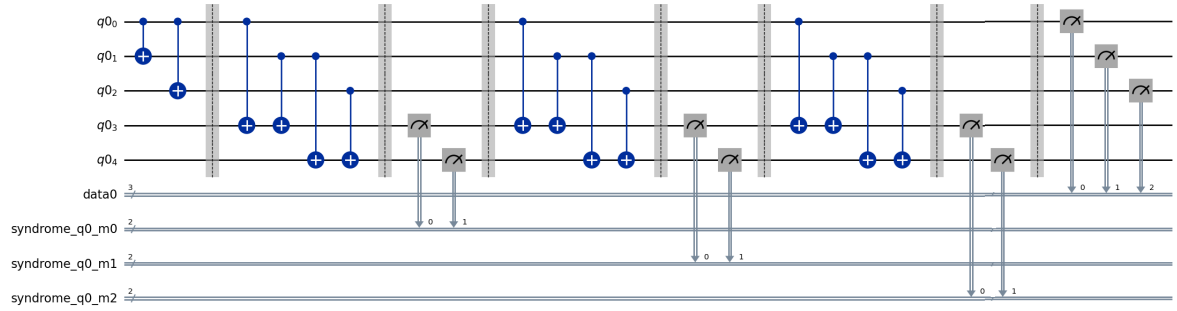
Sammanfattningsvis bygger den GNN-arkitektur som används i detta projekt på att först ta in syndromdata i form av grafer, där nodernas egenskaper uppdateras genom flera lager av graffaltning. Därefter sammanfattas hela grafen till en vektor med hjälp av pooling som sedan används för att avgöra vilka logiska fel som har inträffat [23].

3 Metod

I detta avsnitt presenteras den metod som användes för att konstruera kvantkretsen, erhålla och lagra data samt metoden för felkorrigering med MWPM och GNN. För fullständig kod med kommentarer, se GitHub [24].

3.1 Konstruktion av kvantkretsar

Med hjälp av Python och Qiskit konstruerades kvantkretsar med olika koddistanser och antal tidssteg. Dessa kretsar kördes på IBM:s kvantdatorer för att generera data till GNN och MWPM. I Figur 10 illustreras en kvantkrets med koddistans $d = 3$.



Figur 10: Kvantkrets med koddistans $d = 3$ och tre tidssteg (antal upprepade syndromextraktioner). De första tre bitarna ($q0_0$, $q0_1$ och $q0_2$) är de kodande kvantbitarna medan de två följande ($q0_3$ och $q0_4$) är ancillabitar för syndrommätningar. Mätning av kvantbitarna och syndrom avkodas på klassiska register (data0 respektive syndrome_q0_m0, syndrome_q0_m1 och syndrome_q0_m2).

Med samma huvudprincip som i Figur 5 sammanflätades tre kvantbitar $q0_0$, $q0_1$ och $q0_2$ innan första barriären med två CNOT-grindar för att bilda den logiska kvantbiten, i detta fall en trekvantbit-repetitionskod ($\alpha |000\rangle + \beta |111\rangle$). Efter den första barriären ansattes CNOT-grindar för vardera närliggande kvantbitpar på de två ancillabitarna $q0_3$ och $q0_4$, där respektive kodande kvantbit motsvarade kontrollkvantbitarna medan ancillabitarna var målkvantbitarna. Detta utgjorde en stabiliseringsmätning. Kretsen konstruerades så att stabiliseringsmätningar kunde utföras utan att de kodande kvantbitarnas tillstånd kollapsar och informationen går förlorad.

Syndromextraktion skedde efter andra barriären genom att mäta ancillabitarna och lagra informationen på en klassisk bit. De kvantbitar som utsattes för bitfel uppvisas i syndromen enligt Tabell 2. Syndromextraktionen upprepades tre gånger för att upptäcka eventuella mätfel, där värdena därefter användes som indata för MWPM- och GNN-avkodarna. Deras prediktioner jämfördes med resultatet från slutgiltiga mätningar av de kodande kvantbitarna efter sista barriären för att träna avkodarna.

Efter att kvantkretsen hade konstruerats, optimerades kretsen för att göras lämplig att köras på en kvantdator, det vill säga att den omskrevs (transpilerades) på ett sätt som minimerar exempelvis dekoherens i kvantsystemet. Detta gjordes genom att hämta en `PassManager` som genererats genom `generate_preset_pass_manager()` [25]. Transpileringen kördes på kvantkretsen för att därefter implementeras på kvantdatorn `ibm_torino`.

Denna process upprepades för kretsar motsvarande den i Figur 10 för udda koddistanser upp till och med 21.

3.2 Lagring av data

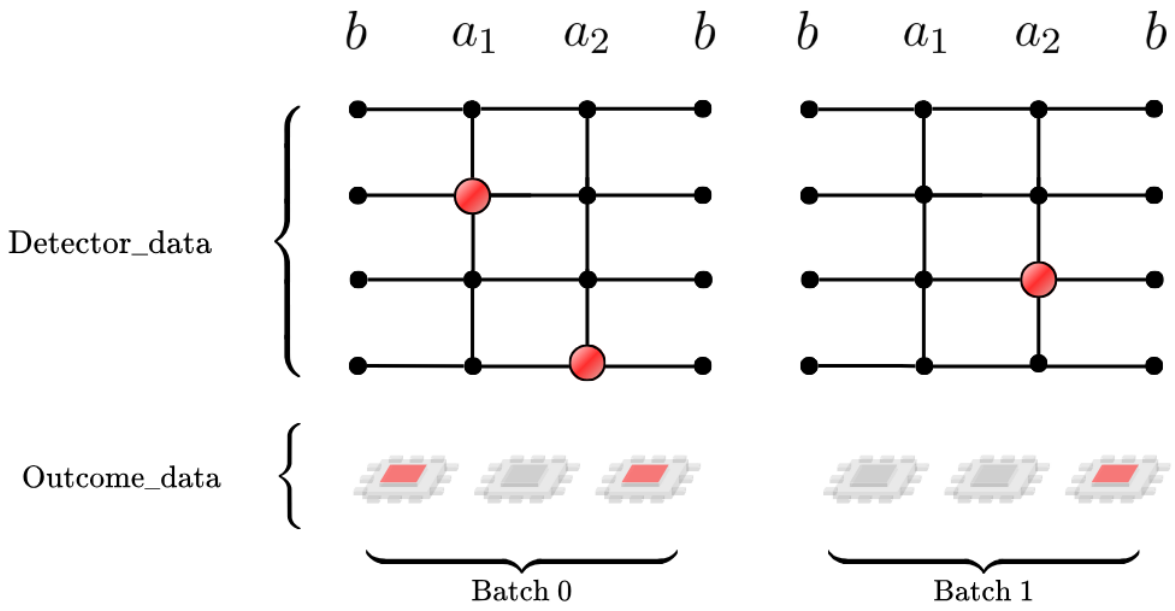
Efter att data hade genererats från kvantdatorn, extraherades den från körningen för att därefter sparas och struktureras med tidigare erhållen kod [26]. Detta gjordes med hjälp av programmet `repetition_code_data.py`, vilket delade upp och sparade datan i json-filer enligt följande struktur:

`Detector_data`	Skillnader i syndrom mellan mätningar (tidssteg).
`Outcome_data`	Formaterad slutmätningsdata.

Datan som användes för maskininlärningen delades upp i två olika typer filer, där valideringsdatan innehöll triviala syndrom medan de ej inkluderades i träningsdatan. För att visualisera datalagringen generellt visas ett exempel efter två körningar (batch 0 och 1) med kvantkretsen i Figur 10, där data har extraherats och slutligen sparats samt formaterats:

```
Detector_data   {"0": [[0, 0], [1, 0], [0, 0], [0, 1]],
                 "1": [[0, 0], [0, 0], [0, 1], [0, 0]]}
Outcome_data    {"0": [1, 0, 1], "1": [0, 0, 1]}
```

I Figur 11 illustreras exempeldatan ovan via grafrepresentationen i delavsnitt 2.3.2.



Figur 11: Illustrativ grafrepresentation av exempeldata från körningar på en kvantdator. Datan kommer från mätning på en repetitionskod med tre kvantbitar och två ancillabitar, uppdelat i två batcher.

Den formaterade datan sparades slutligen i mappar för att därefter tillämpas av MWPM- och GNN-avkodarna.

3.3 Grafrepresentation

Grafrepresentationen i delavsnitt 2.3.2 användes både för GNN och MWPM för att utföra syndromavkodning. Utifrån detektordatan skapades en graf för varje körning på kvantdatorn, med noder i varje förändring av ancillabitarna och kanter mellan dessa noder, som användes som indata till MWPM- och GNN-avkodarna. Grafrepresentationen för GNN genererades i Python och baserades på ett tidigare projekt [19]. Grafen representerades med följande variabler:

`node_features`	Koordinater för syndromförändringar.
`edge_index`	Kanter mellan `node_features`.
`batch_labels`	Vilka `node_features` som tillhör vilken körning.
`edge_attr`	Beskriver vikter (avstånd) för varje kant.

3.4 Minimum Weight Perfect Matching

För att utföra MWPM-avkodning av syndromdata användes `pymatching`, en Python-baserad implementering av MWPM-algoritmen Sparse Blossoms (se delavsnitt 2.3.4). För att formatera datan, konstruera grafrepresentationen och köra MWPM-algoritmen användes tidigare erhållen kod [26]. Eftersom inputdatan är densamma som i GNN-metoden kunde den formaterade datan i `Detector_data` och `Outcome_data` direkt återanvändas för MWPM. Utifrån `Detector_data` genererades grafrepresentationen via `pymatching`-funktionerna `Matching.add_edge` och `Matching.add_boundary`, och grafens kantvikter sattes till 1. Själva MWPM-algoritmen utfördes sedan med funktionen `Matching.decode_batch` och rättfrekvensen togs fram genom att jämföra MWPM-förutsägelserna med `Outcome_data`.

3.5 GNN

Även koden för GNN-avkodaren skrevs i Python och baserades på ett tidigare projekt [19] som använde maskininlärning i avkodning av ytkoder, till skillnad från detta arbete som behandlar repetitionskoder. Projektet kunde dock användas som mall till denna implementering med fördefinierade bibliotek, GNN-modeller och generell struktur.

Ytkoder behandlar tredimensionella grafer (för att korrigera för bitfel, fasfel och mätfel) medan repetitionskoder behandlar tvådimensionella (för att korrigera för antingen bitfel eller fasfel, samt mätfel). För detta projekt anpassades alltså modellen till repetitionskoder genom att omstrukturera nätverket från tredimensionella grafer som indata till tvådimensionella. Den grafrepresentation som tidigare skapats användes då även som indata till GNN-avkodaren.

Avkodaren tränades därefter med data genererad från 6×10^6 körningar på kvantdatorn och testades med data från 500 000 körningar. Avkodaren kunde dock endast tränas med data från syndrom som inte är triviala, det vill säga `Detector_data` där minst ett fel inträffat. Triviala syndrom saknar händelser (fel), vilket innebär att de resulterande graferna som skulle använts till träningen blir tomma och därför inte bidrar till inlärningen. På grund av detta beror mängden användbar träningsdata på koddistansen, eftersom sannolikheten för icke-triviala syndrom ökar med längre koddistans. Träningen och testningen upprepades för udda koddistanser upp till och med 21, där parametrar som nätverkets inlärningshastighet, storlek och antalet graf-faltningslager justerades för

att optimera inlärningen, medan de tre klassificeringslagren hölls konstanta med storlekarna `[64, 128, 256]`. Djupare beskrivning av nätverkets arkitektur finns i avsnitt 2.4.2.

Inledningsvis tränades ett mindre GNN med en inlärningshastighet på $0,01$ och tre graf-faltningslager med storlekarna `[32, 128, 256]`. GNN med denna arkitektur kunde tränas och testas på en personlig dator. Därefter tränades ett större GNN med en inlärningshastighet på cirka $0,002$ och sju graf-faltningslager med storlekarna `[32, 128, 256, 512, 512, 256, 256]`. För det större nätverket utnyttjades Alvis, vilket presenteras i delavsnitt 3.6. Prestandan hos båda nätverken jämfördes för att analysera skillnader.

3.6 Träning av GNN med Alvis

För att möjliggöra den mer omfattande träningen som krävdes för det större graf-neurala nätverket användes Alvis, ett GPU-baserat datakluster som är en del av NAISS (Nationell Akademisk Infrastruktur för Superdatorer i Sverige). Det tidigare utvecklade nätverket samt den genererade träningsdatan importerades till Alvis, där de anpassades för att köras på dess GPU-arkitektur.

För att fullständigt träna de större nätverken behövdes långa träningstider. Träningen utfördes därför under flera epoker, där datans ordning slumpades inför varje ny epok för att skapa mer varierad inlärning och minska risken för att nätverket fastnar i ett lokalt minimum eller överanpassas.

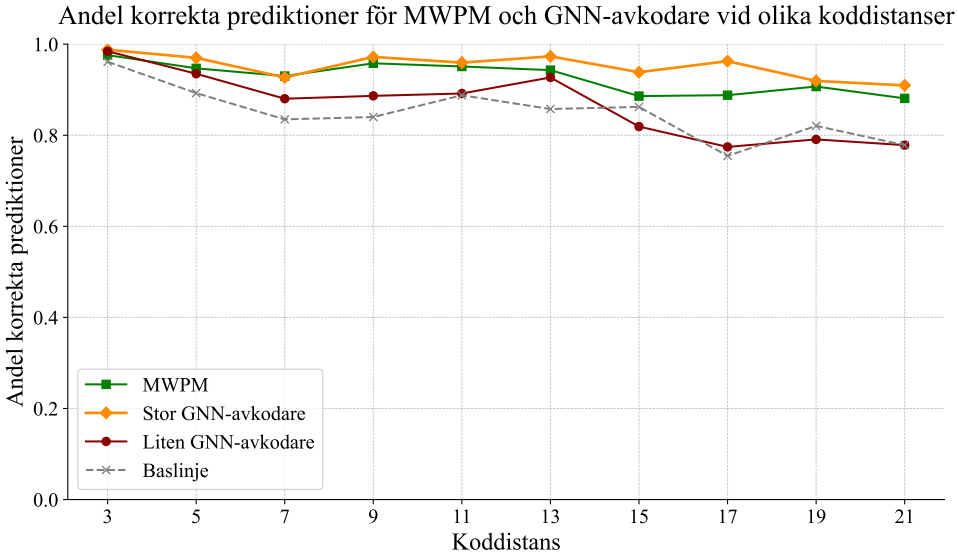
Under träningen undersöktes nätverkets prestanda kontinuerligt genom att rättfrekvensen på valideringsdatan utvärderades. Detta gjorde det möjligt att kontrollera att nätverket tränades fullständigt utan att överanpassas till träningsdatan.

4 Resultat

I detta avsnitt presenteras resultaten på hur GNN- och MWPM-avkodarna presterade för olika koddistanser. Andelen triviala syndrom illustreras även för att betona skillnaden i prestanda vid koddistanserna och inlärningsprocessen för det större GNN:et presenteras.

4.1 GNN och MWPM

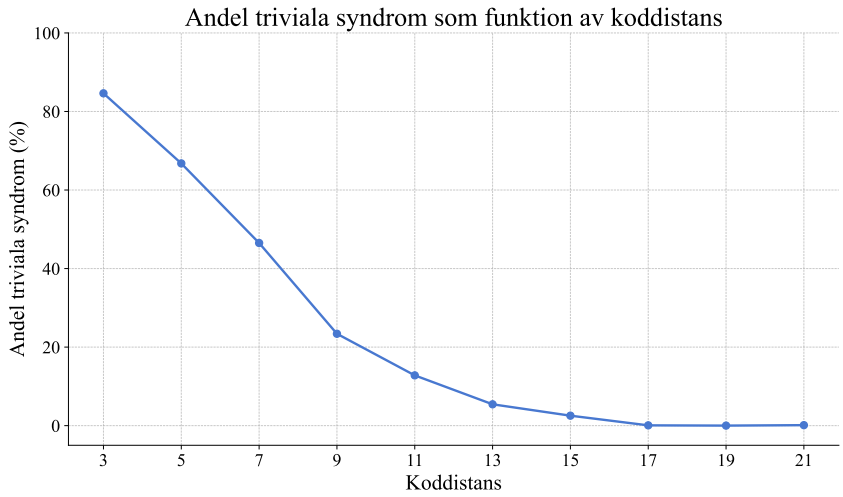
I Figur 12 presenteras prestandan hos GNN och MWPM som avkodare genom andelen korrekta prediktioner (rättfrekvensen) för respektive avkodare vid olika koddistanser. Det noteras att samtliga avkodare presterade bättre än om ingen felprediktion hade utförts för lägre koddistanser ($d \leq 11$), men för högre koddistanser varierar prestandan för den mindre GNN-avkodaren. Exempelvis presterade den sämre än baslinjen vid koddistans 15 och 19. Både MWPM- och den större GNN-avkodaren presterade dock bättre än fallet utan felprediktion, och den mindre GNN-avkodaren för samtliga undersökta koddistanser. Den större GNN-avkodaren presterade även i allmänhet bättre än MWPM.



Figur 12: Andel korrekta prediktioner vid olika koddistanser med MWPM samt ett mindre och större GNN som avkodare. Baslinjen representerar andelen körningar utan uppstådda bitfel och utan att någon felkorrektion utförs.

4.2 Triviala syndrom

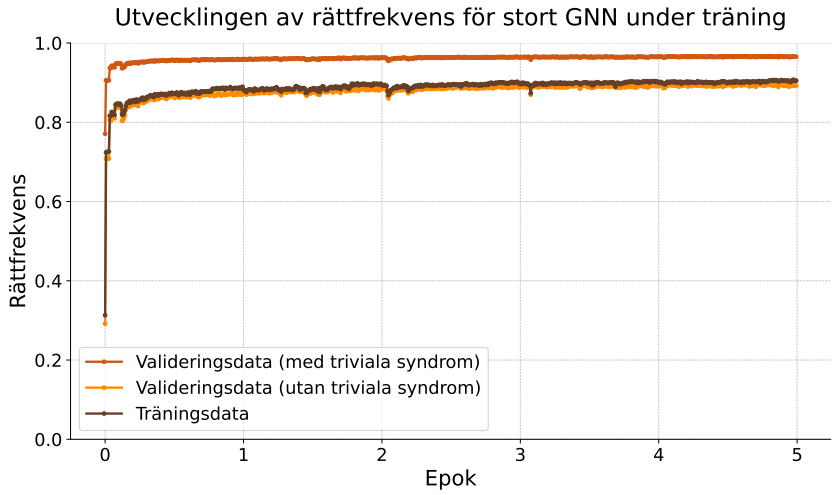
I Figur 13 presenteras andelen triviala syndrom för respektive koddistans, vilket överlag tyder på hur stor risken är för att fel uppstår för olika koddistanser. Ur figuren observeras att andelen triviala syndrom konvergerar mot noll med ökad koddistans.



Figur 13: Andel triviala syndrom som funktion av koddistans. Den triviala andelen beräknas utifrån skillnaden i antal data före och efter borttagning av triviala syndrom.

4.3 Inlärningskurva för GNN

I Figur 14 redovisas inlärningskurvan för den större GNN-avkodaren vid koddistans 5 genom att plotta rättfrekvensen för träning- och valideringsdata för varje träningsepok. Inlärningskurvorna för övriga koddistanser, förutom 19 och 21, hade samma utseende som för koddistans 5. Gällande fallet för koddistans 19 och 21, visade inlärningskurvan en stor skillnad mellan tränings- och valideringsdatan. Figuren demonstrerar att inlärningskurvan för träningsdatan ger något högre rättfrekvens än valideringsdatan utan triviala syndrom, vilket är väntat eftersom detta är den datamängd som nätverket tränades på. Inlärningskurvan för valideringsdatan med triviala syndrom skiljer sig således, på grund av att nätverket tränades på icke-trivala syndrom. Rättfrekvensen stiger snabbt i början av träningen och stabiliseras därefter. Att rättfrekvensen för valideringsdatan ej minskar i förhållande till träningsdatan tyder på att nätverket inte överanpassas.



Figur 14: Inlärningskurvan för den större GNN-avkodaren som funktion av epok vid koddistans 5. Graferna motsvarar rättfrekvens vid varje epok för träningsdata samt valideringsdata med och utan triviala syndrom. Notera att träningsdatan inte inkluderar triviala syndrom.

5 Diskussion

I detta avsnitt analyseras de erhållna resultaten utifrån metod och teori. Även relevanta felkällor samt samhälleliga och etiska aspekter presenteras och redogörs för.

5.1 Analys av resultat

I teorin bör rättfrekvensen öka med koddistansen eftersom längre distanser innebär högre redundans mot uppstådda fel i kvantbitarna. Däremot noteras det i Figur 12 att rättfrekvensen, med undantag för fluktuationer, har en negativ trendlinje, det vill säga att rättfrekvensen generellt minskar för ökad koddistans. Detta gäller för både GNN och MWPM. Detta kan bland annat bero på tröskelvärdet som beskrivs i delavsnitt 2.1.4, där risken för uppstådda fel kan ha överstigit tröskelvärdet för repetitionskoden, vilket i så fall ej bidrar till högre redundans. Vad gäller GNN skulle trenden också kunna ha påverkats av arkitekturen och storleken hos nätverket samt tiden för träning.

I och med hur rättfrekvensen varierar över koddistansen, visas att MWPM presterar relativt bra vid låga koddistanser, men att rättfrekvensen sjunker vid högre distanser. Efter koddistans $d \geq 13$ börjar prestandan för MWPM märkbart försämras, vilket står i kontrast till det stora GNN:et som trots nedgång i prestanda vid dessa koddistanser ändå uppvisar högre rättfrekvens än MWPM. Detta kan dels bero på att modellerna får tillgång till mer data som överensstämmer med det verkliga utfallet som trivial data i viss mån representerar, dels på att sannolikheten för triviala syndrom är högre för de kortare koddistanserna, vilket vid inkludering medför att andelen bitfel i datan minskar i stor grad. Det gör att modellerna vid dessa koddistanser oftare möter fall som är lättare att klassificera korrekt. Som tidigare beskrivet uppvisar resultaten att det större nätverket uppnår högre rättfrekvens för längre koddistanser i förhållande till MWPM. En orsak är att maskininlärning har en större flexibilitet att anpassa sig till ändringar, medan MWPM begränsas av en fördefinierad algoritm som inte bygger på verkliga utfall. Detta kan ha påverkat resultaten om datan avviker markant från den algoritm som MWPM följer, vilket maskininlärningen inte begränsas av på samma sätt. Resultaten i Figur 12 tyder även på att ett mer välträ nat större GNN ger en större andel korrekta prediktioner än ett mindre GNN. En möjlig faktor är att maskininlärning har förmågan att identifiera komplexa felmönster [27], där det är rimligt att ett större GNN lär sig dessa mönster bättre. Skillnaden mellan det större och mindre nätverket är tydlig för alla koddistanser $d > 3$.

Trots att ett välträ nat GNN kan överstiga prestandan hos MWPM, noteras en försämring i resultatet för koddistanser $d \geq 19$. Detta pekar på att även det större nätverket saknar kapaciteten att hantera den ökade felkomplexiteten eller systemfel som eventuellt har uppstått vid träning av nätverket. För att dra djupare slutsatser kring hur olika GNN-strukturer påverkar rättfrekvensen krävs vidare studier, men i allmänhet bidrar större nätverk och mer träning till högre rättfrekvens, vilket är rimligt då nätverket lär sig mer komplexa samband som kan korrigeras för.

Vad gäller träningen av GNN, tyder inlärningskurvan i Figur 13 på att nätverken har tränats bra. Rättfrekvensen för valideringsdatan var aldrig högre än träningsdatan som nätverken tränades med och rättfrekvensen konvergerade väl. Detta berodde bland annat på att träningen skedde över flertalet epoker, vilket möjliggjorde en reducerad inlärningsgrad

som ledde till en mjukare konvergens och ett mer stabilt tränat nätverk. Detta visade sig vara avgörande för nätverkets prestanda på valideringsdata, särskilt för de större koddistanserna som generellt krävde längre träningssessioner. Dessa träningar må ha varit gynnsamma för prestandan, men tog lång tid, vilket minskar effektiviteten [27]. Det finns alltså en avvägning mellan prestanda och effektivitet som är relevant att ta hänsyn till, vilket bland annat kan påverkas av valet av kvantdator, den information som ska skickas samt vilken typ av kvantfelskorrektion som ska utföras.

5.2 Felkällor

En möjlig orsak till resultaten för MWPM är att algoritmen har en del begränsningar, framför allt vid uppkomsten av korrelerade kvantfel som kan ge upphov till dekoherens, antingen på grund av exempelvis fasfel eller magnetiskt flöde [28]. När flera kvantbitar påverkas samtidigt blir det svårare för MWPM att korrigera dessa fel, vilket kan ha påverkat resultaten för projektet. En möjlig lösning är att kombinera maskininlärning parallellt med MWPM vid kvantfelskorrektion. Således kan noggrannheten i felkorrigeringen förbättras, då det ofta uppstår fel i kvantbitar intill varandra istället för enskilda bitfel. Ett förslag [28] beskriver hur man kan tillämpa en hybridmetod som kombinerar GNN och konvolutionella (convolutional) neurala nätverk (CNN) med MWPM. Denna metod förbättrar felkorrigeringen av enskilda och korrelerade kvantfel, i detta fall upp till en 50 % minskning av logiska bitfel kontra tillämpning av enbart MWPM [28]. En annan möjlighet är att träna ett neuralt nätverk för att hitta de troligaste vikterna som varje kant viktas mot, istället för att utnyttja korrelationsmatriser eller förutbestämda vikter. Därefter kan MWPM tillämpas som vanligt, men med sannolikheter som överensstämmer med kvantfelet bättre. Denna metod skulle vidare kunna undersökas i framtida projekt. En annan metod är att använda experimentellt uppmätta felfrekvenser som baseras på kvantdatorns felmodell, vilket är något som tidigare har undersökts [29]. Således kan man prediktera två syndrom i ytkoden, p_{ij}, genom att relatera detta till felmodellen och erhålla vikter som bättre stämmer överens med det verkliga utfallet [29].

Som tidigare beskrivet blir felfrekvensen lägre vid längre koddistanser om den fysiska felsannolikheten p för varje enskild kvantbit är lägre än kodens tröskelvärde p_{th}. Men i takt med att distansen blir längre krävs det mer resurser då fler operationer och kvantbitar är inblandade [30]. Exempelvis orsakar detta längre körningstider, vilket i sin tur kan orsaka dekoherens som ett resultat av att kvantbitarnas tillstånd blir instabila vid längre inaktivitet [31][32]. Med andra ord ökar sannolikheten för att ett kvantfel inträffar när koddistansen blir längre och ger därmed en ökad risk att felsannolikheten överskrider tröskelvärdet. Det kan i sin tur vara en möjlig orsak till den generella minskningen av rättfrekvensen för de större koddistanserna.

Ytterligare en felkälla är kvantkretsens struktur. När en grind tillämpas på flera kvantbitar parallellt kan också närliggande kvantbitar påverkas [33]. Detta kallas för *crosstalk* och bygger på att kvantbitar styrs av mikrovågor och lasrar som eventuellt kan påverka omgivningen [33]. Om kvantbitarna befinner sig fysiskt nära varandra kan crosstalk bli påtaglig, vilket ger upphov till oönskade fel i vissa kvantbitar. Transpilatorn på IBM ska optimera för att bland annat minimera crosstalk men bör ej uteslutas som felkälla. Likaledes kan tillämpning av *error mitigation* minimera brus och således andelen uppstådda fel i kvantkretsen, exempelvis genom att minska dekoherensen [34]. I projektet implemen-

terades inte error mitigation av den anledningen att skillnaden i rättfrekvens före och efter implementering var ytterst liten, men kan vara tillämpbar i en större utsträckning för längre koddistanser än de som har undersökts i detta arbete.

I och med mätfelen är valet av kvantdator som körs en begränsning för hur bra resultaten blir, där vissa kvantdatorer överlag presterar bättre än andra. Hur hög rättfrekvensen kan bli varierar alltså för olika kvantdatorer. För kvantdatorn `ibm_torino` noterades att vissa kvantbitar har högre felfrekvens än andra, vilket ökar sannolikheten för mätfel. Anledningen till detta har troligtvis att göra med att grindarna som tillämpas på dessa kvantbitar inte utför operationerna korrekt, i detta fall SX-grinden (en en-kvantbitsgrind som motsvarar $\sqrt{X}$ [14]), eller att kvantbitarna i kvantdatorn inte är i tillräckligt bra skick. Det genomsnittliga mätfelet på kvantbitarna med hög felfrekvens låg mellan 17 %-31 %, till skillnad från övriga kvantbitar som hade en median av mätfel på 2,637 % [35]. Som tidigare beskrivet bör detta inte vara problematiskt för GNN, eftersom nätverket till viss del kan lära sig att ta hänsyn till variationer i felfrekvens som uppstår då vissa kvantbitar underpresterar. I kontrast fungerar detta inte för MWPM med förutbestämda vikter, eftersom algoritmen inte tar hänsyn till sådana mönster eller lär sig från tidigare data. Därmed påverkas MWPM mer negativt av denna typ av systematiska mätfel, särskilt om vissa kvantbitar är betydligt mer felbenägna än andra.

En begränsning för GNN-avkodaren är mängden data som användes för träning. Data genererades från 6×10^6 körningar inklusive triviala fall, annars färre då dessa tas bort. Andelen triviala fall varierar med koddistansen, där mängden träningsdata minskar för lägre distanser på grund av att andelen triviala syndrom ökar. Däremot genererar mer data bättre träningsresultat och därmed mindre bias, eftersom det neurala nätverket blir exponerat för mer komplexa mönster [36]. Detta blev särskilt tydligt för det större nätverket, där antalet parametrar är högt i förhållande till mängden träningsdata. Inledningsvis tenderade nätverket att överanpassas innan det lyckades nå en bra rättfrekvens, men genom att sänka inlärningsgraden kunde en stabil träning uppnås utan tydliga tecken på överanpassning. Även om detta förbättrade inlärningen väsentligt för alla koddistanser, observerades fortfarande en tydlig skillnad mellan tränings- och valideringsdatan för distans 19 och 21, där rättfrekvensen på valideringsdatan var betydligt lägre. Detta tyder på att nätverket troligen stötte på fel i valideringsdatan som inte täcktes av träningsdatan, vilket i sin tur pekar på att mängden träningsdata inte var tillräcklig. Det är därför troligt att en större mängd träningsdata, särskilt för längre koddistanser med större felkomplexitet, hade lett till ännu bättre generaliseringsförmåga och lägre felfrekvens. Detta är ytterligare en förklaring till att nätverkens felfrekvens ökade för högre koddistanser.

Slutligen upprepades stabiliseringsmätningarna i kvantkretsen tre gånger för varje koddistans. Däremot ger längre koddistanser generellt upphov till högre sannolikhet för uppstådda fel. Därför hade det varit rimligt att utföra ett antal mätningar proportionellt mot den undersökta koddistansen för att korrigera för den ökade risken att fel uppstår. Å ena sidan skulle fler upprepade stabiliseringsmätningar bidra till att GNN-avkodaren kan identifiera ett större felmönster och därmed tränas på mer data för att således maximera rättfrekvensen, å andra sidan hade detta bidragit till fler datapunkter att ta hänsyn till, vilket komplicerar datalagring och gör träningen av GNN mer ineffektiv.

5.3 Samhälleliga och etiska aspekter

Effektiv kvantfelskorrektion är nödvändig för att garantera funktionaliteten hos kvantdatorer. Dessa har potential att effektivisera flera olika samhällsområden och därmed möjliggöra nya upptäckter och framsteg både inom forskning och industri som beskrivits i inledningen. Utvecklingen av kvantteknologi sker snabbt och Sverige är en av de ledande aktörerna inom detta område [37]. Delsing [37] förklarar att användningsområdena för kvantteknologi är på väg att kunna omsättas i praktiska sammanhang i samhället.

Kvantdatorers samhällspåverkan medför dock i sin tur betydande etiska utmaningar, framförallt i takt med utvecklingen av kryptering och informationssäkerhet. På grund av hur kvantdatorer är uppbyggda kan de relativt snabbt lösa problem som hade tagit klassiska datorer väldigt lång tid. Sådana typer av problem ligger bland annat till grund för dagens krypteringsmetoder. Alltså finns en överhängande risk att nuvarande skydd inte räcker till för att garantera säkerheten för känslig information [38]. Forskning pågår för att utveckla kvantsäkra krypteringsmetoder, men det är fortfarande osäkert när dessa kommer att vara helt implementerade [39]. Detta skapar en potentiell säkerhetsrisk där lagrad data kan bli oskyddad i framtiden, vilket gör det viktigt att redan nu arbeta med nya skyddsmekanismer [40]. Med dessa aspekter i åtanke är en praktisk generalisering av kvantdatorer i samhället fortfarande värd att tillämpa på grund av kvantteknologins positiva effekter i flertalet samhällsområden.

Utöver de etiska aspekterna kring utveckling av kvantdatorer, krävs det mycket resurser för att konstruera, köra och underhålla ett kvantsystem. En kvantbit uppskattas kosta mellan \$10 000 och \$50 000, vilket innebär att framtida kvantdatorer, som förväntas innehålla 1000 kvantbitar, kan kosta uppemot \$100 miljoner [41]. I detta projekt användes cirka 8 timmar på körningar på kvantdatorn, vilket motsvarar en kostnad på drygt 200 000 kr enligt IBM Quantum Platforms Premium Plan, vars prislista ligger på \$48 per minut [42]. Å ena sidan kan detta anses vara dyrt då projektet endast befattar en liten del av kvantfelskorrektion i kvantdatorer, å andra sidan har det erhållits tydliga resultat för maskininlärningens roll, vilket kan påverka framtida och större forskningsprojekt som främjar utvecklingen av kvantdatorer. Med tanke på de positiva effekterna som en optimal kvantdator medför, kan de initiala kostnaderna betraktas som ekonomiskt lönsamma på lång sikt.

Tidseffektivitet är även en väsentlig faktor inom kvantfelskorrektion. MWPM är generellt sett snabbare än maskininlärningstillämpningar, där den större GNN-avkodaren kunde ta upp till 20 timmar att genomföra en träningssession för att minimera felfrekvensen. Ur ett samhällsperspektiv är det av stor betydelse att tillämpa effektiva metoder för felkorrigering, då det minskar ekonomiska kostnader och arbetsbelastning. Dessutom möjliggör kortare beräkningstider snabbare tillgång till färdigt material och därmed främjar utvecklingen av kvantdatorer.

Användningen av maskininlärning är också en central del i detta projekt och är relevant att studera. Etiken kring maskininlärning bygger enligt framför allt på fyra pelare [43]:

- **Rättvisa**: Det är viktigt att säkerställa att maskininlärningsalgoritmer agerar rättvist mot alla och inte diskriminerar individer eller folkgrupper baserat på särskilda egenskaper.

- **Transparens**: Det ska erbjudas tydliga och förståeliga förklaringar kring hur maskininlärningsalgoritmer tar beslut och på vilka faktorer.
- **Integritet**: Personlig integritet ska skyddas och robusta säkerhetsåtgärder ska etableras vid insamling, lagring och användning av data för att hindra missbruk och obehörig åtkomst.
- **Ansvarsskyldighet**: Det är nödvändigt att utvecklare och användare av maskininlärningssystem hålls ansvariga för eventuella negativa konsekvenser som kan orsakas av systemet för att upprätthålla etiska principer.

Det är i huvudsak dessa principer som ligger till grund för en moraliskt godtagbar användning av maskininlärning. För att upprätthålla rättvisa är exempelvis transparens viktigt, eftersom man ska kunna spåra hur eventuellt diskriminerande beslut har fattats av maskininlärningssystemet. Men det är också viktigt att se till att systemet även kan användas exempelvis av personer med nedsatt språk-, tal-, syn- respektive hörselförmåga [43]. Transparens kan uppnås genom att exempelvis erbjuda tillgång till den underliggande koden och träningsdata eller genom att dokumentera inlärningsprocessen [43]. Det finns idag stora krav på integritet av allmänheten och denna upprätthålls genom bland annat dataskydd som styrs av föreskrifter som GDPR, HIPAA och CCPA. Utöver detta måste företag också skydda de interna processerna som styr databehandlingen [43]. Ansvarsskyldigheten uppstår ur exempelvis tydliga riktlinjer kring hur man använder maskininlärningssystemet. Sammanlagt ger användning av maskininlärning alltså upphov till ett flertal etiska aspekter som måste tas hänsyn till.

En viktig miljöaspekt kring användningen av maskininlärning är dess energikonsumtion och de utsläpp som medföljer. Stora GPU-kluster som används för att träna olika modeller kan i vissa fall ha lika stora koldioxidutsläpp som en 4000 km lång flygresa [44]. Alltså är det relevant att utvärdera den miljöpåverkan särskilda beräkningar har för att avgöra deras ändamålsenlighet. Det är möjligt att kvantdatorer kan minska energikraven för maskininlärning, men eftersom det ännu inte finns något överenskommet system för att utvärdera energieffektiviteten hos kvantdatorer går det inte att säga med säkerhet [45].

Trots att maskininlärning är centralt i detta arbete, används den i en relativt låg utsträckning kontra de algoritmer som tillämpas av exempelvis stora företag. Således är de samhälleliga och etiska implikationerna av maskininlärning i detta projekt förhållandevis låga. Däremot ökar relevansen av sådana aspekter allt eftersom kvantfelskorrektion kan generaliseras och tillämpas inom vidare forskning. Det är alltså av hög betydelse att ha dessa perspektiv i åtanke.

6 Slutsatser

Sammanfattningsvis har detta arbete belyst betydelsen av maskininlärning i utvecklingen av kvantdatorer. Rapporten har redogjort för teorin bakom kvantfelskorrektion och dess roll i kvantdatorer. Även metoder för syndromavkodning som använder sig av både GNN och MWPM har beskrivits och jämförts för att dra rimliga slutsatser i förhållande till undersökningens syfte.

De erhållna resultaten tyder på att mer omfattande träning av graf-neurala nätverk generellt ger högre rättfrekvens för samtliga undersökta koddistanser. Resultaten pekar även på att ett tillräckligt stort och välträinat GNN kan korrigera för uppstådda bitfel i repetitionskoder bättre än MWPM. Då MWPM är den mest väletablerade avkodningsalgoritmen som inte använder maskininlärning, indikerar detta att maskininlärning har potential för att vara en effektiv avkodningsmetod i kvantfelskorrektion och att den har en ansenlig roll i utvecklingen av kvantdatorer.

Vidare kan resultaten användas för att dra slutsatser kring huruvida maskininlärning är en rimlig avkodningsmetod i kvantdatorer som använder sig av ytkoder, det vill säga koder som kan korrigera för både bitfel och fasfel. För att göra de erhållna resultaten mer praktiskt tillämpbara, är det alltså högst relevant att vidare studera de områden som har presenterats i detta arbete i framtida undersökningar som rör kvantfelskorrektion.

Referenser

[1] "Quantum Computing and Its Potential Impact". (13 sept. 2024), [Online]. URL: https://medium.com/@MakeComputerScienceGreatAgain/quantum-computing-and-its-potential-impact-9ff292a9f172 (hämtad 2025-05-01).

[2] "What is quantum computing?" Hämtad från IBM Quantum. (aug. 2024), [Online]. URL: https://www.ibm.com/think/topics/quantum-computing (hämtad 2025-03-05).

[3] "Correcting quantum errors". Hämtad från IBM Quantum. (u.å.), [Online]. URL: https://learning.quantum.ibm.com/course/foundations-of-quantum-error-correction/correcting-quantum-errors (hämtad 2025-05-06).

[4] J. Roffe, "Quantum error correction: an introductory guide", *Contemporary Physics*, årg. 60, nr 3, s. 226–245, 2019. DOI: 10.1080/00107514.2019.1667078.

[5] O. Higgott och C. Gidney, "Sparse Blossom: correcting a million errors per core second with minimum-weight matching", *Quantum*, årg. 9, 2025. DOI: https://doi.org/10.48550/arXiv.2303.15933.

[6] "Quantum state". Hämtad från Wikipedia, the free encyclopedia. (febr. 2025), [Online]. URL: https://en.wikipedia.org/wiki/Quantum_state (hämtad 2025-05-08).

[7] "Bloch sphere". Hämtad från Wikipedia, the free encyclopedia. (juli 2024), [Online]. URL: https://en.wikipedia.org/wiki/Bloch_sphere (hämtad 2025-03-04).

[8] E. Knill, R. Laflamme och W. H. Zurek, "Resilient quantum computation", *Science*, årg. 279, nr 5349, s. 342–345, jan. 1998. DOI: 10.1126/science.279.5349.342.

[9] "Quantum codes with other thresholds", Error Correction Zoo. (u.å.), [Online]. URL: https://errorcorrectionzoo.org/list/quantum_threshold (hämtad 2025-05-14).

[10] "Circuit (datavetenskap)". Hämtad från Wikipedia, den fria encyklopedin. (jan. 2025), [Online]. URL: https://en-m-wikipedia-org.translate.goog/wiki/Circuit_(computer_science)?_x_tr_sl=en&_x_tr_tl=sv&_x_tr_hl=sv&_x_tr_pto=rq (hämtad 2025-03-11).

[11] "Quantum circuits". Hämtad från IBM Quantum. (u.å.), [Online]. URL: https://learning.quantum.ibm.com/course/basics-of-quantum-information/quantum-circuits#quantum-circuits (hämtad 2025-03-11).

[12] "Boolean circuit". Hämtad från Wikipedia, den fria encyklopedin. (dec. 2024), [Online]. URL: https://en.wikipedia.org/wiki/Boolean_circuit (hämtad 2025-03-11).

[13] "Quantum logic gate". Hämtad från Wikipedia, the free encyclopedia. (april 2025), [Online]. URL: https://en.wikipedia.org/wiki/Quantum_logic_gate (hämtad 2025-04-16).

[14] "Explore gates and circuits with IBM Quantum Composer". Hämtad från IBM Quantum. (), [Online]. URL: https://learning.quantum.ibm.com/tutorial/explore-gates-and-circuits-with-the-quantum-composer (hämtad 2025-04-16).

[15] "Qiskit". Hämtad från Wikipedia, the free encyclopedia. (febr. 2025), [Online]. URL: https://en-m-wikipedia-org.translate.goog/wiki/Qiskit?_x_tr_sl=en&_x_tr_tl=sv&_x_tr_hl=sv&_x_tr_pto=rq (hämtad 2025-03-05).

[16] "Machine learning". (juni 2020), [Online]. URL: https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/ (hämtad 2025-04-14).

[17] "Neural network". Hämtad från IBM. (okt. 2021), [Online]. URL: https://www.ibm.com/think/topics/neural-networks (hämtad 2025-04-18).

[18] "Graph neural network". Hämtad från Wikipedia, the free encyclopedia. (april 2025), [Online]. URL: https://en.wikipedia.org/wiki/Graph_neural_network (hämtad 2025-05-02).

[19] M. Lange, P. Havström, B. Srivastava *et al.*, "Data-driven decoding of quantum error correcting codes using graph neural networks", *Phys. Rev. Res.*, årg. 7, s. 023 181, 2 maj 2025. DOI: 10.1103/PhysRevResearch.7.023181. [Online]. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.7.023181.

[20] PyG Team. "conv.GraphConv". (2024), [Online]. URL: https://pytorch-geometric.readthedocs.io/en/2.5.3/generated/torch_geometric.nn.conv.GraphConv.html.

[21] PyG Team. "pool.global_mean_pool". (2025), [Online]. URL: https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.pool.global_mean_pool.html.

[22] "Layer (deep learning)". Hämtad från Wikipedia, the free encyclopedia. (dec. 2024), [Online]. URL: https://en.wikipedia.org/wiki/Layer_(deep_learning) (hämtad 2025-05-04).

[23] J. Zhou, G. Cui, S. Hu *et al.*, "Graph neural networks: A review of methods and applications", *AI Open*, årg. 1, s. 57–81, 2020, ISSN: 2666-6510. DOI: https://doi.org/10.1016/j.aiopen.2021.01.001. [Online]. URL: https://www.sciencedirect.com/science/article/pii/S2666651021000012.

[24] J. Sandström, N. Kalmnäs Drakenfors och E. Olofsson. "QuantumErrorCorrection", GitHub. (2025), [Online]. URL: https://github.com/Jacob-sandstrom/QuantumErrorCorrection (hämtad 2025-05-14).

[25] "Transpile with pass managers". Hämtad från IBM Quantum. (u.å.), [Online]. URL: https://docs.quantum.ibm.com/guides/transpile-with-pass-managers (hämtad 2025-05-07).

[26] J. Olsson, *Privat kommunikation*, 2025.

[27] Rowanrothe. "The Magic of Hidden Layers in Neural Networks", Medium. (), [Online]. URL: https://medium.com/demistify/the-magic-of-hidden-layers-in-neural-networks-989b05791dc7.

[28] Y. Li. "Hybrid CNN-GNN Decoding for Enhanced Quantum Error Correction under Correlated Noise with MWPM". (u.å.), [Online]. URL: https://ncur.secure-platform.com/2025/gallery/rounds/30/details/28725? (hämtad 2025-05-02).

[29] Google Quantum AI, "Exponential suppression of bit or phase errors with cyclic error correction", *Nature*, årg. 595, nr 7866, s. 383–387, juli 2021, Open access. DOI: 10.1038/s41586-021-03588-y.

[30] "Quantum Error Correction (QEC)". (u.å.), [Online]. URL: https://www.quera.com/glossary/quantum-error-correction (hämtad 2025-05-02).

[31] M. Ivezic. "Quantum Errors and Quantum Error Correction (QEC) Methods", Postquantum. (10 maj 2023), [Online]. URL: https://postquantum.com/quantum-computing/quantum-error-correction/ (hämtad 2025-05-02).

[32] A. Davour. "Så testades den svenska kvantdatorn med ett realistiskt problem", Forskning Framsteg. (12 jan. 2021), [Online]. URL: https://fof.se/artikel/2021/2/sa-testades-den-svenska-kvantdatorn-med-ett-realistiskt-problem/ (hämtad 2025-05-02).

[33] M. T. D. Mura. "Quantum Noise: Overcoming This Obstacle is Crucial for the Evolution of Quantum Computing", Tech4Future. (29 jan. 2024), [Online]. URL: https://tech4future.info/en/quantum-noise-quantum-computing/ (hämtad 2025-05-04).

[34] "Configure error mitigation". Hämtad från IBM Quantum. (u.å.), [Online]. URL: https://docs.quantum.ibm.com/guides/configure-error-mitigation.

[35] "Quantum processing units". Hämtad från IBM Quantum. (u.å.), [Online]. URL: https://quantum.ibm.com/services/resources?system=ibm_torino.

[36] "Why Does More Data Increase Accuracy?", Digital Products. (18 nov. 2024), [Online]. URL: https://www.digitalproductsdp.com/blog/why-does-more-data-increase-accuracy#viewer-gf9n5111667 (hämtad 2025-05-06).

[37] Vinnova, "Sverige kan stärka sin position inom kvantteknologi", *Vinnova Rapport*, 22 mars 2023. [Online]. URL: https://www.vinnova.se/nyheter/2023/03/sverige-kan-starka-sin-position-inom-kvantteknologi/.

[38] "Shor's algorithm". Hämtad från Wikipedia, the free encyclopedia. (nov. 2024), [Online]. URL: https://en.wikipedia.org/wiki/Shor%27s_algorithm (hämtad 2025-01-30).

[39] "NIST Drops New Deadline for PQC Transition". Hämtad från Keyfactor. (nov. 2024), [Online]. URL: https://www.keyfactor.com/blog/nist-drops-new-deadline-for-pqc-transition/?utm_source=chatgpt.com (hämtad 2025-02-11).

[40] Vinnova, "Viktiga tekniker för Sverige – Kvantsäker kryptering och framtida utmaningar", *Vinnova Rapport*, s. 15, okt. 2024. [Online]. URL: https://www.vinnova.se/contentassets/f7d33a184cde48e6917e888d10c2836c/2024-01501-slutrapport-viktiga-tekniker-for-sverige.pdf.

[41] B. Tran. "The Cost Of Quantum Computing: How Expensive Is It To Run A Quantum System? (Stats Inside)", PatentPC. (1 maj 2025), [Online]. URL: https://patentpc.com/blog/the-cost-of-quantum-computing-how-expensive-is-it-to-run-a-quantum-system-stats-inside.

[42] "Explore access options". Hämtad från IBM. (u.å.), [Online]. URL: https://www.ibm.com/quantum/pricing.

[43] E. Canorea. "Ethics and Machine Learning: Present and Future Challenges", Plain Concepts. (3 sept. 2024), [Online]. URL: https://www.plainconcepts.com/ethics-machine-learning-challenges/ (hämtad 2025-05-04).

[44] E. Strubell, A. Ganesh och A. McCallum, "Energy and Policy Considerations for Deep Learning in NLP", i *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum och L. Màrquez, utg., Florence, Italy: Association for Computational Linguistics, juli 2019, s. 3645–3650. DOI: 10.18653/v1/P19-1355. [Online]. URL: https://aclanthology.org/P19-1355/.

[45] S. Chen, "Are quantum computers really energy efficient?", *Nature Computational Science*, nr 3, s. 457–460, juni 2023. [Online]. URL: https://doi.org/10.1038/s43588-023-00459-6.

INSTITUTIONEN FÖR FYSIK
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2025
www.chalmers.se

CHALMERS