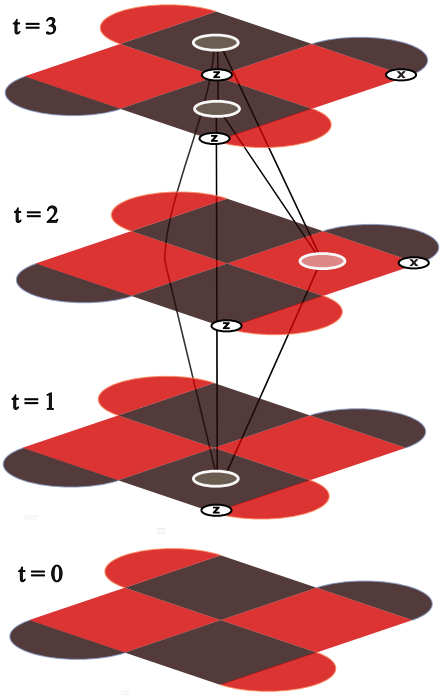
CHALMERS




# Avkodning av ytkoder för kvantfelskorrektion med grafneurala nätverk

Undersökning av hur grafneurala nätverk presterar vid avkodning med data från IBM Nighthawk

Kandidatarbete vid Chalmers tekniska högskola

NOA GRANATH, FELICIA JOHANSON, HUGO LARSSON,
EMANUEL PETERSON, DAVID RIVERON, SIMON SHAFIQ AHLGREN

INSTITUTIONEN FÖR FYSIK OCH ASTRONOMI

CHALMERS TEKNISKA HÖGSKOLA

Göteborg 2026

www.chalmers.se

Kandidatarbete 2026

# Avkodning av ytkoder för kvantfelskorrektion med grafneurala nätverk

Undersökning av hur grafneurala nätverk presterar vid avkodning med data från IBM Nighthawk

# Decoding Surface Codes for Quantum Error Correction using Graph Neural Networks

An Investigation into the Performance of Graph Neural Networks for Decoding Data from IBM Nighthawk

NOA GRANATH, FELICIA JOHANSON, HUGO LARSSON,
EMANUEL PETERSON, DAVID RIVERON, SIMON SHAFIQ AHLGREN


CHALMERS

Institutionen för fysik och astronomi
Chalmers tekniska högskola
Göteborg 2026

Avkodning av ytkoder för kvantfelskorrektion med grafneurala nätverk
Undersökning av hur grafneurala nätverk presterar vid avkodning med data från IBM Nighthawk
NOA GRANATH, FELICIA JOHANSON, HUGO LARSSON,
EMANUEL PETERSON, DAVID RIVERON, SIMON SHAFIQ AHLGREN



Handledare: Mats Granath, Institutionen för fysik vid Göteborgs universitet
Examinator: Jan Swenson, Institutionen för fysik och astronomi vid Chalmers tekniska högskola

Kandidatarbete 2026
Institutionen för fysik och astronomi
Chalmers tekniska högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Omslagsbild: Exempel på grafkonstruktion från detektionshändelser på ytkoden i fyra tidssteg.

Skriven i LaTeX
Göteborg 2026

## Sammandrag

Kvantdatorer har potential att lösa beräkningsproblem som är svåra eller praktiskt omöjliga för klassiska datorer, men deras utveckling begränsas av att kvantbitar är mycket känsliga för brus och fel. För att möjliggöra storskalig kvantberäkning krävs därför effektiva metoder för kvantfelskorrektion. I detta arbete undersöks om grafneurala nätverk (GNN) kan användas som avkodare för ytkoder och prestera bättre än den etablerade metoden Minimum Weight Perfect Matching (MWPM). Projektet fokuserar på ytkoder med kodavstånd $d \in 3, 5$ implementerade på kvantdatorn IBM Miami med Nighthawk r1-processorn. Data genererades först från verkliga körningar, varefter data från en Detector Error Model (DEM) genererades utifrån den experimentella datan. Den utvecklade modellen består av en kombination av ett grafneuralt nätverk och ett rekurrent neuralt nätverk (GNN-RNN), där syndromdata representeras som grafer för att fånga både rumsliga och tidsmässiga samband mellan detektionshändelser. Modellen tränades på NAISS GPU-kluster Alvis; först på simulerad data och finjusterades därefter ytterligare med experimentell IBM-data. Resultaten visar att den färdigtränade GNN-RNN-avkodaren uppnår lägre logisk felfrekvens än MWPM för samtliga undersökta kombinationer av kodavstånd och antal rundor $T \in 10, 20$.



## Abstract

Quantum computers have the potential to solve computational problems that are difficult or practically impossible for classical computers. However, their development is limited by the fact that qubits are highly sensitive to noise and errors. Effective methods for quantum error correction are therefore required to enable large-scale quantum computation. This work investigates whether graph neural networks (GNNs) can be used as decoders for surface codes and outperform the established method Minimum Weight Perfect Matching (MWPM). The project focuses on surface codes with code distances $d \in 3, 5$ implemented on the IBM Miami quantum computer equipped with the Nighthawk r1-QPU. Data was generated from experimental runs, after which data from a detector error model (DEM) was generated based on the experimental data. The developed model consists of a combination of a graph neural network and a recurrent neural network (GNN-RNN), where syndrome data is represented as graphs in order to capture both spatial and temporal correlations between detection events. The model was trained on NAISS GPU cluster Alvis; first using simulated data and subsequently fine-tuned further on experimental IBM data. The results show that the fully trained GNN-RNN decoder achieves a lower logical error rate than MWPM for all investigated combinations of code distance and number of rounds $T \in 10, 20$.

# Förord

Vi vill först och främst tacka vår handledare Mats Granath för all hjälp och vägledning vi fått under arbetets gång. Mats har varit ett stort stöd i att förstå den bakomliggande teorin i arbetet och lett oss i rätt riktning. Ytterligare vill vi rikta ett stort tack till Moritz Lange för allt stöd och hjälp, bland annat med att felsöka vår kod.

Beräkningarna möjliggjordes av resurser tillhandahållna av National Academic Infrastructure for Supercomputing in Sweden (NAISS), delvis finansierat av Swedish Research Council via bidragsavtal nr. 2022-06725.

Mätningarna på IBM:s kvantdator möjliggjordes av stöd från WACQT Quantum Technology Testbed som drivs av Chalmers Next Labs, vilket är finansierat av Knut och Alice Wallenbergs stiftelse.

Under arbetet har AI-baserade verktyg använts för att hitta information, ge förslag till kod, förbättra formuleringar i text samt rendera skissade figurer för tydlighet.

Noa Granath, Felicia Johanson, Hugo Larsson, Emanuel Peterson,
David Riveron, Simon Shafiq Ahlgren
Göteborg, maj 2026

# Beteckningar

Nedan följer en sammanställning av de akronymer och centrala begrepp som används i rapporten.

## Akronymer

| | |
|---|---|
| DEM | Detector Error Model |
| GNN | Grafneuralt nätverk (eng. Graph Neural Network) |
| GRU | Grindad rekurrent enhet (eng. Gated Recurrent Unit) |
| MWPM | Minimum Weight Perfect Matching |
| QEC | Kvantfelskorrektion (eng. Quantum Error Correction) |
| QPU | Kvantprocessor (eng. Quantum Processor Unit) |
| RNN | Rekurrent neuralt nätverk (eng. Recurrent Neural Network) |

## Begrepp

| | |
|---|---|
| Ancillabit | Fysisk kvantbit som används för att detektera fel |
| Databit | Fysisk kvantbit som håller information |
| Detektionshändelse | Händelse som sker då en ancillabit byter tillstånd |
| Hyperparameter | Förutbestämd parameter för träning av neurala nätverk |
| Incident (Grafer) | Benämning för att en nod och en kant är anslutna |
| Kodavstånd | Minsta antalet kvantfel som kan orsaka ett logiskt fel i en QEC-kod |
| Syndrom | Mätresultat som används för att identifiera fel |
| Ytkod | QEC-kod där kvantbitar organiseras på ett tvådimensionellt rutnät |

# Innehåll


**1 Inledning** **1**
- 1.1 Syfte och mål . . . 1
- 1.2 Avgränsningar . . . 2

**2 Teori** **3**
- 2.1 Kvantberäkning . . . 3
  - 2.1.1 Kvantbiten . . . 3
  - 2.1.2 System av flera kvantbitar . . . 4
  - 2.1.3 Kvantlogiska grindar . . . 5
- 2.2 Kvantfelskorrektion . . . 6
  - 2.2.1 Stabilisatorkoder . . . 6
  - 2.2.2 Ytkod . . . 7
  - 2.2.3 Syndrommätning från ytkoden . . . 8
  - 2.2.4 Minimum Weight Perfect Matching . . . 9
- 2.3 Artificiella neurala nätverk . . . 10
  - 2.3.1 Rekurrenta neurala nätverk . . . 11
  - 2.3.2 Grindad rekurrent enhet . . . 11
  - 2.3.3 Grafneurala nätverk . . . 12
  - 2.3.4 Global medelpooling . . . 12
  - 2.3.5 Överföringsinlärning . . . 12

**3 Metod** **14**
- 3.1 Konstruktion av ytkodskretsen . . . 14
- 3.2 Data . . . 15
  - 3.2.1 IBM-hårdvarudata . . . 15
  - 3.2.2 DEM-data . . . 16
- 3.3 GNN-RNN-modellen . . . 16
- 3.4 Träning av modellerna . . . 17
- 3.5 MWPM-modellen . . . 18
- 3.6 Jämförelse . . . 18

**4 Resultat** **19**

**5 Diskussion** **23**
- 5.1 Samhälleliga och etiska aspekter . . . 24
- 5.2 Slutsats . . . 24

**Referenser** **25**

# Kapitel 1

# Inledning

Under de senaste decennierna har kvantdatorer vuxit fram som ett av de mest lovande forskningsområdena inom modern fysik och datavetenskap. Genom att utnyttja kvantmekaniska fenomen för informationsbearbetning erbjuder de en möjlig lösning på vissa beräkningsproblem som annars är orealistiska att hantera med klassiska datorer. Exempelvis kan kvantdatorer potentiellt ge betydande hastighetsfördelar vid optimeringsproblem, upptäcka nya läkemedel samt åstadkomma stora framsteg inom kryptografi [1], [2], [3]. Detta är möjligt eftersom kvantdatorer använder kvantbitar som kan utnyttja kvantfenomen som superposition och sammanflätning [4]. Däremot är kvantbitar mycket mottagliga för störningar från omgivningen, vilket leder till brus och fel i beräkningar [5]. Hanteringen av kvantfel är därför en central utmaning för utvecklingen av kvantdatorer i större skala.

För att hantera kvantfelen har teorin om kvantfelskorrektion utvecklats [6]. Kvantfel skiljer sig från fel i klassiska informationssystem, där fel ofta modelleras som bitflips. I kvantsystem kan fel istället delas upp i två huvudkategorier, nämligen bitfel ($X$-fel) och fasfel ($Z$-fel) samt kombinationer av dessa ($Y$-fel) [7]. För att kunna hantera de båda typerna av fel har så kallade ytkoder vuxit fram som en lovande metod. Ytkoder delar in kvantbitarna i databitar och ancillabitar, där ancillabitar i sin tur är uppdelade för att kunna detektera antingen $X$-fel eller $Z$-fel. För att ingen kvantinformation ska förstöras utförs inga direkta mätningar på databitarna. Istället görs mätningar på ancillabitarna vars utfall är beroende av om fel uppkommer i deras sammankopplade databitar, utan att avslöja det individuella tillståndet hos databitarna [8]. Utifrån detektionshändelserna som uppkommer från ancillabitarna kan olika metoder appliceras för att försöka avkoda de uppkomna felen.

Den vanligaste metoden för avkodning av kvantfel är Minimum Weight Perfect Matching (MWPM) [9]. MWPM är en algoritm som räknar ut det mest sannolika felet som har uppstått utifrån detektionshändelser. Däremot antar algoritmen att bitfel och fasfel bidrar oberoende av varandra och därmed utnyttjas inte all information som kan erhållas från syndromen [10]. Med de senaste framstegen inom maskininlärning har en metod utvecklats där grafneurala nätverk används och tränas på syndromdatan för att försöka avkoda de fel som observerats. Detta har visat sig kunna prestera bättre än MWPM i fall där endast simulerad data används, vilket visades i Fjeldså och Jonasson Johanssons examensarbete [11], [12]. Förhoppningarna är att denna metod ska kunna överträffa MWPM även på experimentell data och att detta kan bli en ny standard för användning vid kvantfelskorrektion.

## 1.1 Syfte och mål

Syftet med projektet är att träna ett grafneuralt nätverk för avkodning av ytkoder med kodavstånd 3 och 5 med både experimentell och simulerad data. Det tränade nätverket ska sedan jämföras med MWPM för att undersöka vilken metod som avkodar felen mest träffsäkert. Målet med projektet är att kunna bidra med värdefulla insikter inom kvantfelskorrektion genom att jämföra hur GNN presterar gentemot MWPM. Resultatet av projektet kan därmed bli viktigt, då framtida beslut ska tas om vilken kvantfelskorrigeringsmetod för avkodning man bör använda i praktiska beräkningar med kvantdatorer.

## 1.2 Avgränsningar

Inom projektet har endast kvantdatorn IBM Miami med en Nighthawk r1-processor använts för körning av ytkoder. Processorn har 120 kvantbitar i ett $12 \times 10$ gitter, vilket begränsar den maximala storleken på ytkoder som kan användas. Ytkoder med kodavstånd 7 eller mer är för stora och därmed användes endast ytkoder med kodavstånd 3 och 5 i projektet.

Då endast en och samma kvantdator användes under hela arbetet kommer all insamlad data att influeras av detta. På chippet förekommer kvantbitar som är mer instabila än andra, samt att kopplingarna mellan dessa också kan variera i stabilitet. Detta innebär att den tränade modellen relativt väl lär sig de imperfektioner som finns på den specifika processorn den är tränad för. Om samma modell skulle appliceras på en annan processor skulle resultaten troligtvis bli betydligt osäkrare. Därmed kommer resultatet inte nödvändigtvis att gälla för samtliga kvantdatorer.

Körtiden på kvantdatorn har också varit begränsad till en timme per månad. Eftersom de flesta körningarna genomfördes i slutet av projektet, innebar detta en betydande begränsning av mängden experimentell data som kunde samlas in. Därmed behövde en stor del av datan simuleras för träningen av modellerna. Den simulerade datan baserades på den experimentellt uppmätta datan utifrån en Detector Error Model (DEM) för att efterlikna den experimentella datan så mycket som möjligt.

För att utvärdera hur väl det tränade grafneurala nätverket presterar jämförs det med MWPM. MWPM är den nuvarande standarden för avkodning av ytkoder och är därmed en lämplig kandidat för att utvärdera hur väl en ny metod presterar. Däremot är detta inte nödvändigtvis den enda metoden som kan jämföras med och därmed är det en avgränsning i tolkningen av resultatet.

# Kapitel 2

# Teori

I kapitlet presenteras teorin som ligger till grund för projektet. Kapitlet innehåller en introduktion till grundläggande kvantberäkning, kvantfelskorrektion och artificiella neurala nätverk.

## 2.1 Kvantberäkning

I detta avsnitt introduceras centrala koncept inom kvantfysik som ligger till grund för kvantberäkning.

### 2.1.1 Kvantbiten

Beräkning på klassiska datorer bygger på konceptet bitar, där varje bit kan befinna sig i ett av två tillstånd, 0 eller 1. Kvantbitar är motsvarigheten till bitar i klassiska datorer men skiljer sig åt då de kan befinna sig i kvanttillstånd, vilket medför egenskaper som superposition och sammanflätning [13]. Superpositionsegenskapen innebär att en kvantbit utöver tillstånden 0 och 1, även kan befinna sig i en superposition av dessa. Tillståndet $|\psi\rangle$ för en kvantbit kan då skrivas som en linjärkombination av bastillstånden $|0\rangle$ och $|1\rangle$ enligt

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.1}$$

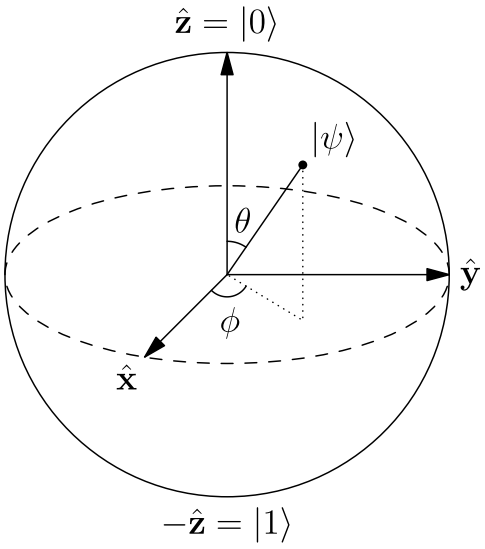
där $\alpha$ och $\beta$ är motsvarande sannolikhetsamplitud för tillstånden, vilka är komplexa tal som uppfyller $|\alpha|^2 + |\beta|^2 = 1$ [14].

Tillståndet hos en kvantbit kan representeras som en vektor i ett tvådimensionellt komplext Hilbertrum $\mathcal{H} = \mathbb{C}^2$, där tillstånden $|0\rangle = [1,0]^T$ och $|1\rangle = [0,1]^T$ utgör en ortonormal bas som brukar benämnas beräkningsbasen eller $Z$-basen, $\{|0\rangle, |1\rangle\}$ [14]. Ett generellt tillstånd $|\psi\rangle$ för en kvantbit kan då enligt ekvation (2.1) uttryckas som $|\psi\rangle = [\alpha, \beta]^T$ där $|\alpha|^2$ och $|\beta|^2$ beskriver sannolikheten för kvantbiten att kollapsa till tillståndet $|0\rangle$ respektive $|1\rangle$ vid mätning.

Tillståndet hos en kvantbit kan även tolkas geometriskt genom representation på Blochsfären, se figur 2.1, där varje tillstånd motsvarar en punkt på ytan av en enhetssfär. Eftersom $|\alpha|^2 + |\beta|^2 = 1$ kan vi uttrycka ekvation (2.1) som

$$|\psi\rangle = cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}sin\frac{\theta}{2}\,|1\rangle \tag{2.2}$$

där $\theta \in [0, \pi],\ \phi \in [0, 2\pi)$ är de sfäriska koordinater som bestämmer tillståndets position på Blochsfären [6].




**Figur 2.1:** Blochsfärsrepresentation av en kvantbit. Bloch Sphere.svg, [15], CC BY-SA.

### 2.1.2 System av flera kvantbitar

I ett system av flera kvantbitar ges det gemensamma tillståndet av tensorprodukten av de enskilda kvantbitarnas tillstånd [14]. Det gemensamma tillståndet för $n$ kvantbitar representeras då som en vektor i Hilbertrummet

$$\mathcal{H} = \bigotimes_{i=1}^{n} \mathcal{H}_i$$

där $\dim \mathcal{H}_i = 2$ vilket medför att $\dim \mathcal{H} = 2^n$.

Tillståndet av ett system med två kvantbitar representeras exempelvis av ett Hilbertrum med beräkningsbasen

$$|00\rangle = |0\rangle \otimes |0\rangle = [1,0,0,0]^T, \quad |01\rangle = |0\rangle \otimes |1\rangle = [0,1,0,0]^T,$$
$$|10\rangle = |1\rangle \otimes |0\rangle = [0,0,1,0]^T, \quad |11\rangle = |1\rangle \otimes |1\rangle = [0,0,0,1]^T,$$

som är de fyra möjliga kombinationerna av bastillstånden för de enskilda kvantbitarna [14]. Ett allmänt tillstånd för ett två-kvantbitarssystem kan då skrivas

$$|\psi\rangle = \alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle + \alpha_{10}\,|10\rangle + \alpha_{11}\,|11\rangle \tag{2.3}$$

där det gäller att $\sum |\alpha_{ij}|^2 = 1$. Sannolikheten att den första kvantbiten kollapsar till 0 vid en mätning ges då av $|\alpha_{00}|^2 + |\alpha_{01}|^2$. Efter en sådan mätning blir det efterföljande tillståndet för systemet

$$|\psi'\rangle = \frac{\alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}} \tag{2.4}$$

eftersom tillståndet fortfarande måste uppfylla normaliseringskriteriet [6].

Ett viktigt tillstånd för två kvantbitar är Bell-tillståndet

$$|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}.$$

Om en mätning görs på en av kvantbitarna så följer av ekvation (2.4) att efterföljande mätning av den andra kvantbiten ger samma utfall [6]. Detta innebär att tillståndet för de två kvantbitarna är sammanflätade.

### 2.1.3 Kvantlogiska grindar

Kvantlogiska grindar beskrivs med unitära operatorer. För enkubitssystem kan dessa uttryckas som $2 \times 2$ matriser, där de mest fundamentala är Paulimatriserna [6]

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

eftersom alla typer av kvantfel kan beskrivas som en kombination av dessa [16]. Pauli-$X$ operatorn motsvarar en bitflip och verkar på ett tillstånd enligt

$$X|\psi\rangle = \alpha X|0\rangle + \beta X|1\rangle = \alpha|1\rangle + \beta|0\rangle, \tag{2.5}$$

medan Pauli-$Z$ operatorn motsvar en fasflip och verkar på ett tillstånd enligt

$$Z|\psi\rangle = \alpha Z|0\rangle + \beta Z|1\rangle = \alpha|0\rangle - \beta|1\rangle. \tag{2.6}$$

Oönskad applicering av dessa operatorer brukar benämnas $X$-fel respektive $Z$-fel och kan visualiseras som rotationer på Blochsfären. $X$-fel motsvarar en rotation med $\pi$ radianer kring $\hat{x}$-axeln och $Z$-fel en rotation med $\pi$ radianer kring $\hat{z}$-axeln [16].

En annan central kvantlogisk grind är Hadarmardgrinden [6]

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.7}$$

eftersom den omvandlar ett bestämt bastillstånd $|0\rangle$ eller $|1\rangle$ till en superposition av båda bastillstånden enligt

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle, \qquad H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle \tag{2.8}$$

där basen $\{|+\rangle, |-\rangle\}$ benämns $X$-basen då den består av egentillstånd till Pauli-$X$ operatorn.

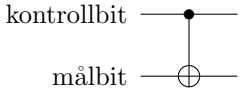
CNOT-grinden är en kvantlogisk grind som använder två kvantbitar, en kontrollbit $|k\rangle$ och en målbit $|m\rangle$. Grinden verkar genom att

$$|k\rangle|m\rangle \to |k\rangle|m \oplus k\rangle \tag{2.9}$$

[6], där $\oplus$ är addition i modulo 2, vilket motsvarar den klassiska XOR-grinden när kvantbitarna befinner sig i beräkningsbasen. CNOT-grinden kan uttryckas i matrisform som

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.10}$$

och markeras i en kvantkrets enligt figur 2.2.




**Figur 2.2:** CNOT-grind representerad i kvantkrets.

CNOT-grinden är central för skapandet av sammanflätade tillstånd. Exempelvis kan Bell-tillståndet skapas genom att applicera en Hadamardgrind på kontrollbiten och sedan tillämpa en CNOT-grind

[6]. För bastillståndet $|00\rangle = |0\rangle \otimes |0\rangle$ följer det av ekvation (2.8) att tillståndet efter tillämpning av Hadarmardgrind på kontrollbiten ges av

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |0\rangle = \frac{|00\rangle + |10\rangle}{\sqrt{2}}. \tag{2.11}$$

Termvis applicering av CNOT ger enligt ekvation (2.9)

$$CNOT\left(\frac{|00\rangle + |10\rangle}{\sqrt{2}}\right) = \frac{|00\rangle + |11\rangle}{\sqrt{2}}, \tag{2.12}$$

vilket är ett Bell-tillstånd.

## 2.2 Kvantfelskorrektion

För att skydda kvantinformation från brus kan den kodas i ett kodrum hos en kvantfelskorrektionsskod. I detta avsnitt introduceras sådana koder.

### 2.2.1 Stabilisatorkoder

En stabilisator $S$ är en operator som vid applicering på ett kvanttillstånd lämnar tillståndet oförändrat [6]

$$S|\psi\rangle = |\psi\rangle. \tag{2.13}$$

Stabilisatorkoder är en klass av kvantfelskorrektionskod där kodrummet utgörs av alla tillstånd som är simultana egentillstånd med egenvärde $+1$ till en uppsättning kommuterande stabilisatorer [16]. I många kvantfelskorrigeringskoder utgörs stabilisatorerna av produkter av Paulioperatorer. Stabilisatorer bestående av produkter av Pauli-$X$ operatorer benämns $X$-stabilisatorer medan stabilisatorer bestående av produkter av Pauli-$Z$ operatorer benämns $Z$-stabilisatorer [6]. Som exempel kan $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ kodas till trebitsrepitionskoden $|\psi\rangle_L = \alpha|000\rangle + \beta|111\rangle$ genom sammanflätning. Den stabiliseras exempelvis av operatorn $Z_1Z_2$

$$Z_1Z_2|\psi\rangle_L = Z_1Z_2(\alpha|000\rangle + \beta|111\rangle) = (+1)(+1)(\alpha|000\rangle + (-1)(-1)\beta|111\rangle) = +1|\psi\rangle_L. \tag{2.14}$$

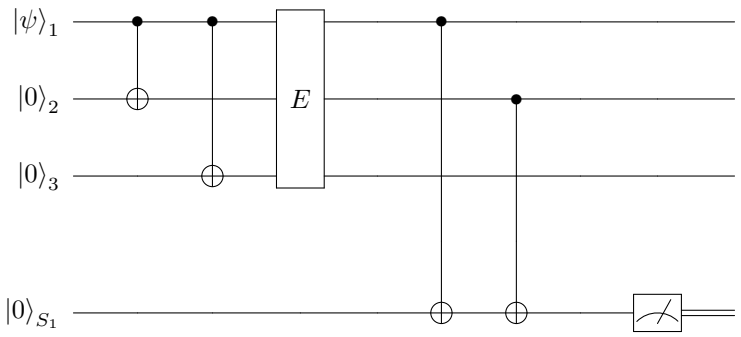
Pauli-$X$ och Pauli-$Z$ operatorerna antikommuterar med varandra, det vill säga att $XZ = -ZX$. Vilket innebär att om ett kvanttillstånd är ett egentillstånd för en $Z$-stabilisator med egenvärde $+1$ kommer applicering av Pauli-$X$ operatorn byta paritet på egenvärdet till $-1$. Antikommutationsegenskapen för Pauli-operatorerna medför därför att ett uppkommet $X$-fel resulterar i ett förändrat mätresultat för en $Z$-stabilisator och vice versa [17]. På grund av att Pauli-operatorerna är egeninversa gäller det dock att en jämn paritet av uppkomna $X$-fel återställer egenvärdet för en $Z$-stabilisator till $+1$ vilket innebär att det vid en stabilisatormätning endast går att utläsa pariteten på antalet uppkomna fel. Betraktas återigen $|\psi_L\rangle$ leder ett $X_1$-fel (udda paritet i relation till $Z_1Z_2$) på $|\psi\rangle_L$ till egenvärdet $-1$ enligt

$$Z_1Z_2(X_1|\psi\rangle_L) = Z_1Z_2(\alpha|100\rangle + \beta|011\rangle) = (-1)(+1)\alpha|100\rangle + (+1)(-1)\beta|011\rangle) = (-1)X_1|\psi\rangle_L, \tag{2.15}$$

medan ett $X_1X_2$-fel (jämn paritet) leder till $+1$ som egenvärde enligt

$$Z_1Z_2(X_1X_2|\psi\rangle_L) = Z_1Z_2(\alpha|110\rangle + \beta|001\rangle) = (-1)^2\alpha|110\rangle + (1)^2\beta|001\rangle) = +1(X_1X_2)|\psi\rangle_L. \tag{2.16}$$

Pariteter är även viktiga inom kvantkretsar. Kretsen i figur 2.3 visar hur $|\psi\rangle_1$ kodas till $|\psi\rangle_L$ efter sammanflätning med kvantbitarna $|0\rangle_2$ och $|0\rangle_3$ med en serie CNOT-grindar, och sedan utsätts för felet som tillhör mängden $E = \{X_1, X_2\}$. Målkvantbiten $|0\rangle_{S_1}$ kommer projiceras till $|1\rangle_{S_1}$ om ett $X$-fel uppkommer på någon av de två övre kvantbitarna.



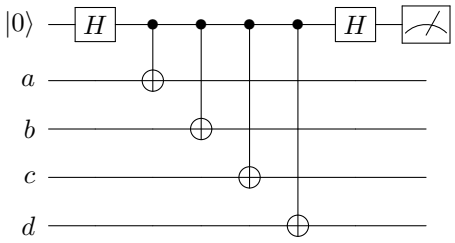

**Figur 2.3:** Stabilisatorkrets. Om en udda paritet av $X$-fel uppkommer, exempelvis $X_1$, byter ancillabiten från tillståndet $|0\rangle_{S_1}$ till $|1\rangle_{S_1}$.

Kretsen i figur 2.3 kallas stabilisatorkrets och den nedre kvantbiten $|0\rangle_{S_1}$ som mäts i slutet är en så kallad ancillabit. Om ancillabiten kvarstår i tillståndet $|0\rangle_{S_1}$, (exempelvis om det inte sker något $X$-fel), säger man att syndromet $s$ antar värdet 0. På motsatt sätt innebär en projektion till $|1\rangle_{S_1}$ ett syndrom $s = 1$.

### 2.2.2 Ytkod

Ytkod är en typ av stabilisatorkod som implementeras på ett tvådimensionellt gitter och bygger enbart på lokala växelverkningar mellan närliggande kvantbitar, vilket gör den särskilt lämplig för kvantsystem med begränsad koppling mellan kvantbitar [18].

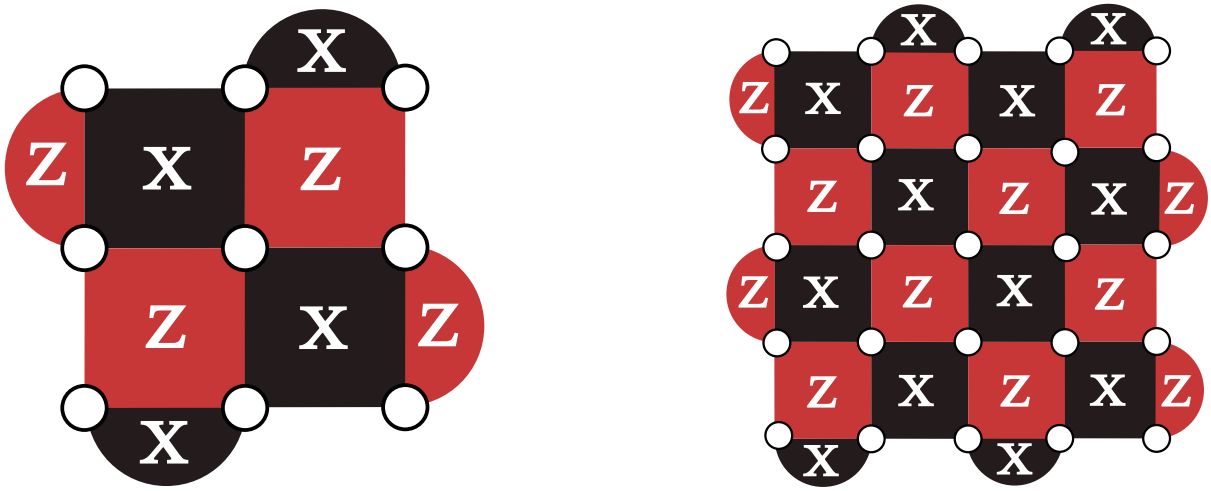
Ytkoden konstrueras genom att de fysiska kvantbitarna delas upp i databitar och ancillabitar och arrangeras så att varje databit angränsar till 4 (eller 2) ancillabitar. Ancillabitarna är i sin tur uppdelade i två typer, $X$-ancillabitar och $Z$-ancillabitar, se figur 2.5. För att mäta stabilisatorerna sammanflätas ancillabitarna med sina angränsande databitar genom en sekvens av CNOT-grindar, se figur 2.4. Denna sammanflätning överför information om pariteten hos databitarna till ancillabiten, vilket innebär att en mätning av en ancillabit ger ett egenvärde av motsvarande stabilisator som utfall, utan att databitarna direkt mäts. $X$-ancillabitar används därmed för att projicera sina angränsande databitar $a, b, c, d$ till ett egentillstånd av $X$-stabilisatorn $X_aX_bX_cX_d$ och $Z$-ancillabitar projicerar sina angränsande databitar till ett egentillstånd av $Z$-stabilisatorn $Z_aZ_bZ_cZ_d$ [17].




**Figur 2.4:** Schema över kvantkrets som visar hur en $X$-ancillabit initialiserad i tillståndet $|0\rangle$ sammanflätas med databitarna $a, b, c, d$ genom tillämpning av Hadamardgrind och en sekvens av CNOT operationer. Se avsnitt 2.1.3 och ekvation (2.11, 2.12).

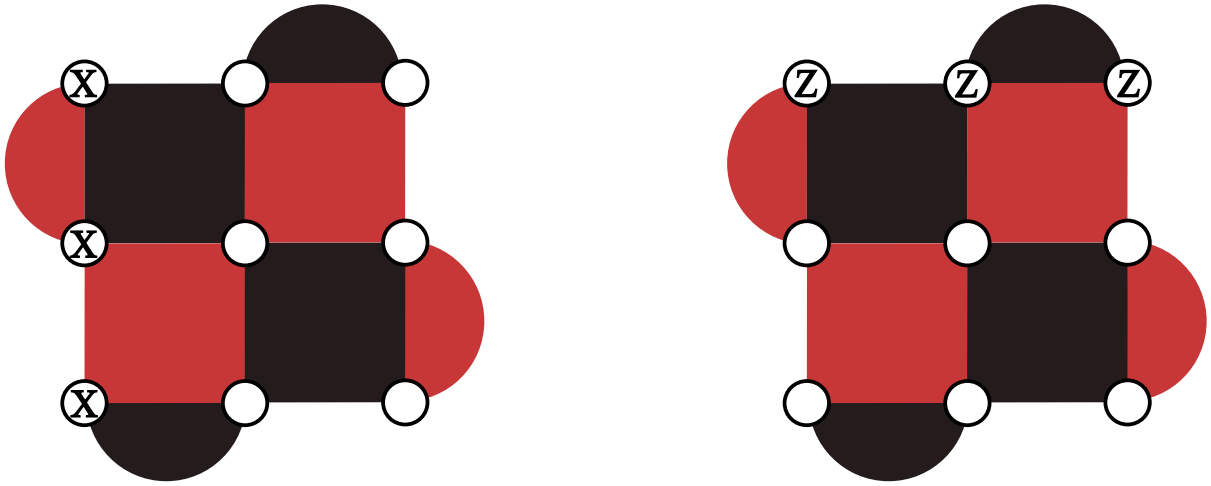
De logiska operatorerna $X_L, Z_L$ i en ytkod måste väljas så att de kommuterar med stabilisatorerna men antikommuterar med varandra [16]. Detta uppnås genom att definiera dem som kedjor av Pauli-operatorer mellan motsatta kanter, se figur 2.6.

De logiska $X$ respektive $Z$-felen definieras som felen som har samma effekt på det logiska tillståndet som den motsvarande logiska operatorn. I ytkoden innebär det att felkedjor som bildar en sammanhängande väg mellan motsatta kanter orsakar logiska fel. Kodavståndet $d$ definieras som det minsta antalet fysiska fel som krävs för att orsaka ett logiskt fel [16]. I ytkoden innebär detta att $d = L$ där $L^2$ är antalet databitar.



**(a)** Ytkod med kodavstånd 3. **(b)** Ytkod med kodavstånd 5.

**Figur 2.5:** Ytkoder med olika kodavstånd. De vita punkterna representerar datakvantbitar medan de svarta och röda kvadraterna samt halvcirklarna representerar ancillabitar.



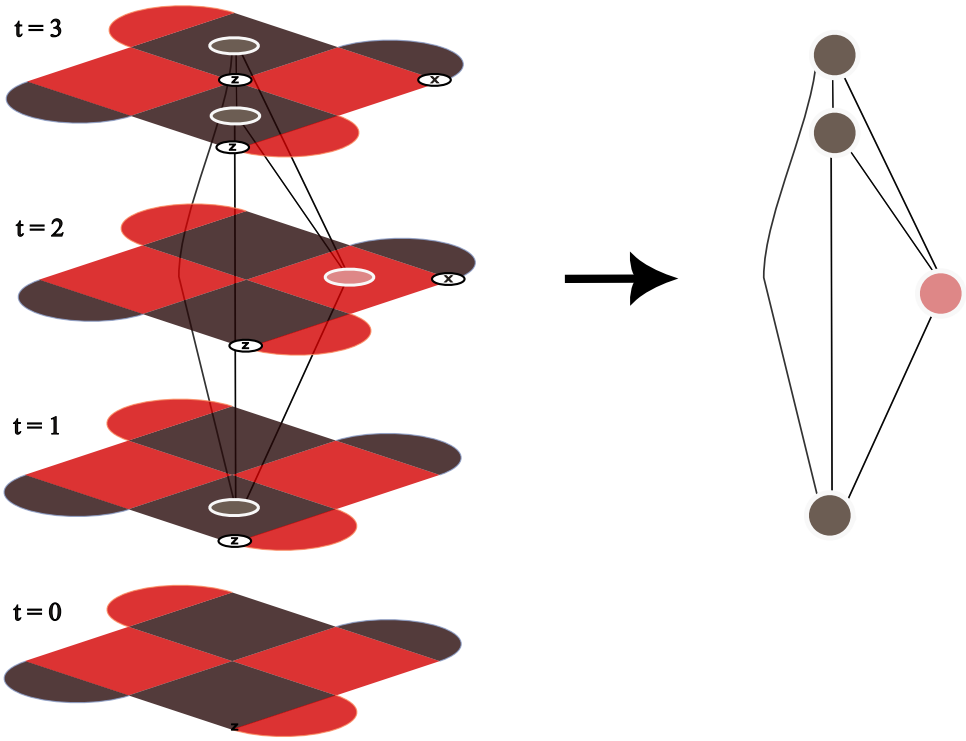
**(a)** Logiska X-operatorn **(b)** Logiska Z-operatorn

**Figur 2.6:** Logiska $X$ respektive $Z$-operatorn på en ytkod med kodavstånd 3.

### 2.2.3 Syndrommätning från ytkoden

För att konstruera ett syndrom från ytkoden mäts ancillabitarna. Eftersom en mätning av en ancillabit motsvarar en mätning av en stabilisator, kan mätutfallet tolkas som stabilisatorns egenvärde. Ett mätresultat på $-1$ på en $Z$-kontrollbit motsvarar därmed en udda paritet av $X$-fel på angränsande databitar och därmed syndromet $s = 1$. På motsatt sätt motsvarar en jämn paritet $X$-fel syndromet $s = 0$. Mätresultatet för samtliga ancillabitar benämns syndromvektorn $S \in \{0, 1\}^{n_S}$ där $s_i$ är mätresultatet för ancillabit $i$ och där $n_S$ är antalet ancillabitar. Genom att göra upprepade syndrommätningar över tid kan förändringarna i mätresultatet användas för att lokalisera var fel har inträffat. En detektionshändelse $\mathbf{d}_{it}$ på ancillabit $i$ under tidssteg $t$ skapas vid en förändring av syndromet mellan två tidssteg, det vill säga om $s_{it} \oplus s_{i(t-1)} = 1$. En avkodare används sedan för att utifrån detektionshändelserna bestämma

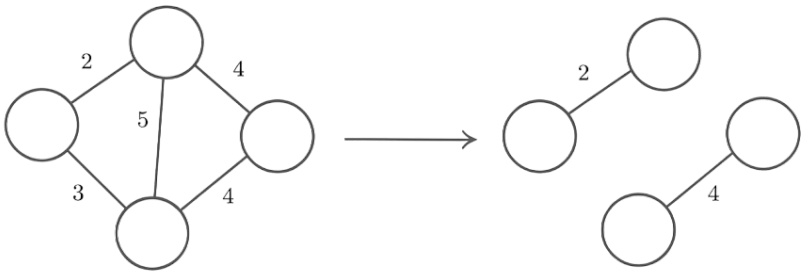
vilka felkedjor som mest sannolikt har inträffat. Avkodarna som används i arbetet utnyttjar sig av att en oriktad graf $G(V, E)$ definieras, där noderna $v_{it} = \mathbf{d}_{it}$ och sammankopplas av kanterna $e$, se figur 2.7.




**Figur 2.7:** Visualisering av grafkonstruktionen från detektionshändelser över fyra tidssteg. Figuren visar hur uppkomna $X$- respektive $Z$-fel på databitarna ger upphov till detektionshändelser på angränsande stabilisatorer (markerade som grå respektive rosa cirklar för $X$ respektive $Z$-stabilisatorer). Detektionshändelserna representeras sedan som noder i en graf som matas till avkodaren.

### 2.2.4 Minimum Weight Perfect Matching

Minimum Weight Perfect Matching (MWPM) är ett optimeringsproblem inom grafteori. Givet en viktad graf $G(V, E)$ där varje kant $e \in E$ har en vikt $w(e) \geq 0$, definieras problemet genom att hitta en perfekt matchning $M \subseteq E$ som minimerar $\sum_{e \in M} w(e)$. En perfekt matchning definieras som en matchning där varje nod $v \in V$ är incident med exakt en kant $e \in E$. MWPM-problemet för grafen kan lösas genom exempelvis Blossom-algoritmen [19], [20].




**Figur 2.8:** MWPM-matchning av en graf bestående av fyra noder och fem kanter med tillhörande kantvikter. Figuren visar matchningen som minimerar summan av kantvikterna.

MWPM är en vanligt förekommande metod för avkodning av ytkoden. Varje detektionshändelse repre-

senteras som en nod och vikterna på kanterna mellan noderna ges av sannolikheten för att en felkedja har orsakat defekten. Avkodningsproblemet reduceras därför till ett MWPM-problem där de minimerade vikterna motsvarar den mest sannolika felkonfigurationen givet syndromet [17].

## 2.3 Artificiella neurala nätverk

Artificiella neurala nätverk är uppbyggda av perceptroner, vilka tar en indatavektor $\mathbf{x} = [x_1, x_2, .., x_n]^T$ och returnerar ett utdatavärde $y$ [21]. Perceptronens utdata är beroende av parametrarna $\mathbf{w} = [w_1, w_2, .., w_n]^T$ vilket definierar vikterna för indatan, samt ett bias $b$. Utdatan ges enligt

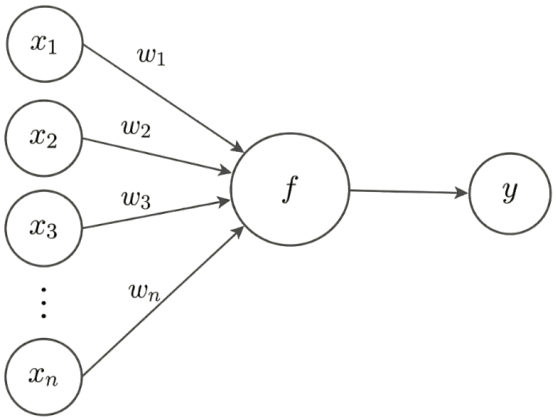
$$y = f(\mathbf{w}^T\mathbf{x} + b) \tag{2.17}$$

där $f$ är en aktiveringsfunktion [22]. Vanligt förekommande aktiveringsfunktioner är Sigmoidfunktionen [21]

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \tag{2.18}$$

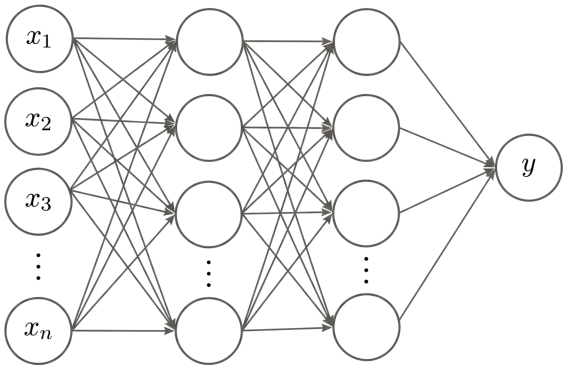
samt Rectified linear unit (ReLU) [22]

$$g(z) = max(0, z). \tag{2.19}$$




**Figur 2.9:** Schematisk bild av perceptron. Figuren visar hur indata vektorn $\mathbf{x} = [x_1, x_2, .., x_n]^T$ viktas med motsvarande vikter $\mathbf{w} = [w_1, w_2, .., w_n]^T$ och bearbetas av aktiveringsfunktionen $f$ för att producera utdata $y$.

Framåtriktade neurala nätverk är uppbyggda av flera lager av perceptroner, där utdatan från första lagrets perceptroner agerar som indata för andra lagret och så vidare [21].




**Figur 2.10:** Schematisk bild av framåtriktat neuralt nätverk med två lager av perceptroner.

Neurala nätverk tränas genom att uppdatera vikterna så att de minimerar en kostnadsfunktion $J$ beroende av nätverkets modelparametrar $\theta$ och ges av korsentropin mellan träningsdatan och nätverkets prediktioner [22]

$$J(\theta) = -\mathbf{E}_{x,y\sim\hat{p}_{data}} \log(\mathbf{p}_{modell}(\mathbf{y} \mid \mathbf{x})) \tag{2.20}$$

där $\mathbf{E}_{x,y\sim\hat{p}_{data}}$ är väntevärdet av träningsdatan. Minimeringen görs vanligtvis med gradientnedstigningsbaserade metoder där parametrarna $w_i$ och $b_i$ uppdateras iterativt enligt

$$w_i' = w_i - \eta\frac{\partial J}{\partial w_i}, \qquad b_i' = b_i - \eta\frac{\partial J}{\partial b_i} \tag{2.21}$$

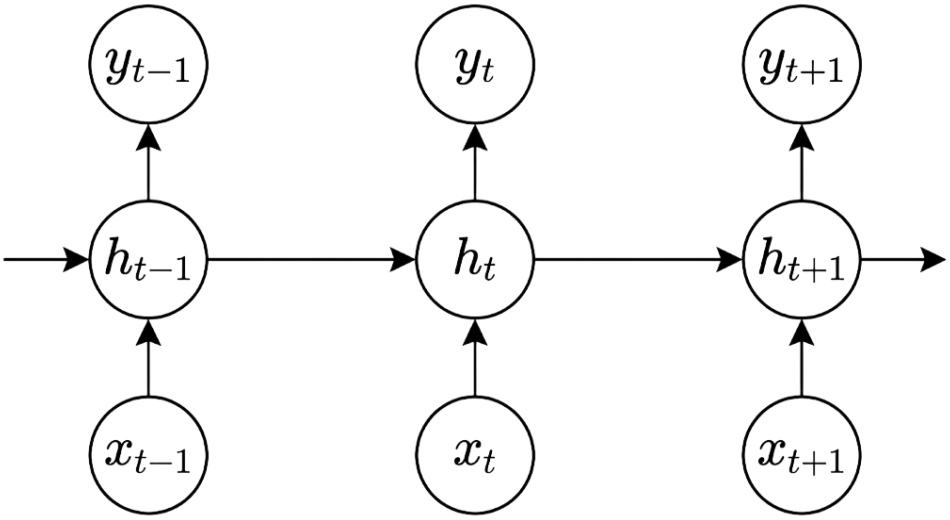
där $\eta$ är en hyperparameter som benämns inlärningshastigheten [21]. För att beräkna de partiella derivatorna av kostnadsfunktionen används bakåtpropagering, vilket är en algoritm som gör detta effektivt genom applicering av kedjeregeln baklänges genom nätverkets lager [23]. I detta arbete används optimeringsalgoritmen Adam som är en variant av stokastisk gradientnedstigning [24].

### 2.3.1 Rekurrenta neurala nätverk

Rekurrenta neurala nätverk (RNN) är en undergrupp av neurala nätverk specialiserade på att hantera sekventiell data, där beroende mellan observationer över tid är av betydelse. Detta hanteras genom införandet av ett dolt tillstånd $\mathbf{h}$ som uppdateras vid varje tidssteg och agerar som ett minne av tidigare indata

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t). \tag{2.22}$$

Utifrån det dolda tillståndet genereras därefter utdata $y_t$, vilket gör det möjligt att modellera tidssekvenser av varierande längd [22].




**Figur 2.11:** Arkitekturen för ett RNN över tre tidssteg. Figuren visar hur det dolda tillståndet $\mathbf{h}_t$ beror på det föregående dolda tillståndet $\mathbf{h}_{t-1}$ och den nuvarande indatan $x_t$.

### 2.3.2 Grindad rekurrent enhet

Grindade rekurrenta enheter (eng. Gated Recurrent Unit) är en variant av ett rekurrent neuralt nätverk (RNN) som används för att hantera sekventiell data och långsiktiga beroenden [25]. Till skillnad från traditionella RNN använder GRU specifika grindar för att kontrollera informationsflödet och reducera problem såsom vanishing gradients. En GRU cell består huvudsakligen av två grindar, en återställningsgrind (eng. Reset Gate) $r_t$ och en uppdateringsgrind (eng. Update Gate) $z_t$. Dessa definieras enligt

$$r_t = \sigma(W_r x_t \ + \ U_r h_{t-1} + b_r), \tag{2.23}$$

$$z_t = \sigma(W_z x_t \ + \ U_z h_{t-1} + b_z), \tag{2.24}$$

där $x_t$ är indata vid tidssteg $t$, $h_{t-1}$ är föregående dolda tillstånd och $\sigma$ är sigmoidfunktionen.

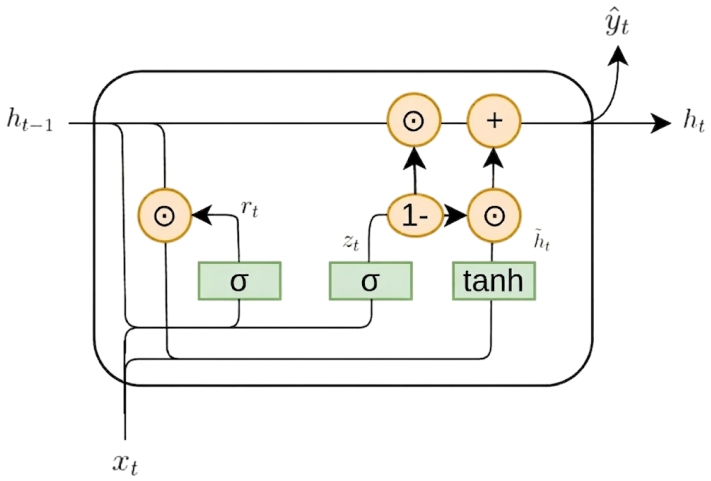
Kandidatvärdet $\tilde{h}_t$ definierat som

$$\tilde{h}_t = tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{2.25}$$

representerar en potentiell uppdatering baserat på indata samt filtrerad tidigare information från den tidigare nämnda återställningsgrinden. Detta värde fungerar som en ny föreslagen representation, vilket sedan kombineras med det tidigare tillståndet genom uppdateringsgrinden för att bestämma det nya dolda tillståndet. Det nya dolda tillståndet ges av

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{2.26}$$

där $\odot$ betecknar elementvis multiplikation (Hadamardprodukt).




**Figur 2.12:** Illustration av informationsflödet i en GRU cell där $r_t$, $z_t$, $\tilde{h}_t$ och $h_t$ beräknas enligt ekvation (2.23 - 2.26). $\hat{y}_t$ representerar modellens prediktion och $h_t$ det dolda tillstånd som skickas vidare till nästa cell vid nästa tidssteg.

### 2.3.3 Grafneurala nätverk

Grafneurala nätverk är en generalisering av neurala nätverk till data med grafstruktur, där indata består av noder och kanter. Varje nod i grafen representeras av ett dolt tillstånd som successivt uppdateras genom att kombinera information från närliggande noder. Detta gör att modellen kan fånga strukturen i grafen oberoende av vilken ordning noderna ges som indata till modellen. Nodernas attribut $g(v)$ uppdateras i varje steg enligt

$$g_t(v) = f\left(W_{1,t}\ g_{t-1}(v) + \sum_{n \in N(v)} W_{2,t}\ g_{t-1}(n)\right) \tag{2.27}$$

där $n \in N(v)$ är de närliggande noderna till $v$ och $f$ är en aktiveringsfunktion [26].

### 2.3.4 Global medelpooling

Den globala medelpoolingen av en graf ges av medelvärdet av alla noders attribut enligt [27]

$$\text{gmp}(G) = \frac{1}{|V|} \sum_{v \in V} g(v). \tag{2.28}$$

Global medelpooling är en sammanfattning av grafen i form av en enda vektor som sedan skickas vidare i nätverket.

### 2.3.5 Överföringsinlärning

Överföringsinlärning är en maskininlärningsmetod som syftar till att förbättra inlärningen av en måluppgift genom att utnyttja inlärda modellparametrar från en relaterad källuppgift. Praktiskt genomförs detta

genom att istället för en slumpmässig initialisering, återanvända det neurala nätverket från källuppgiften som utgångspunkt för vidare optimering. Överföringsinlärning är särskilt användbart om måluppgiften har begränsad tillgång till träningsdata [28].

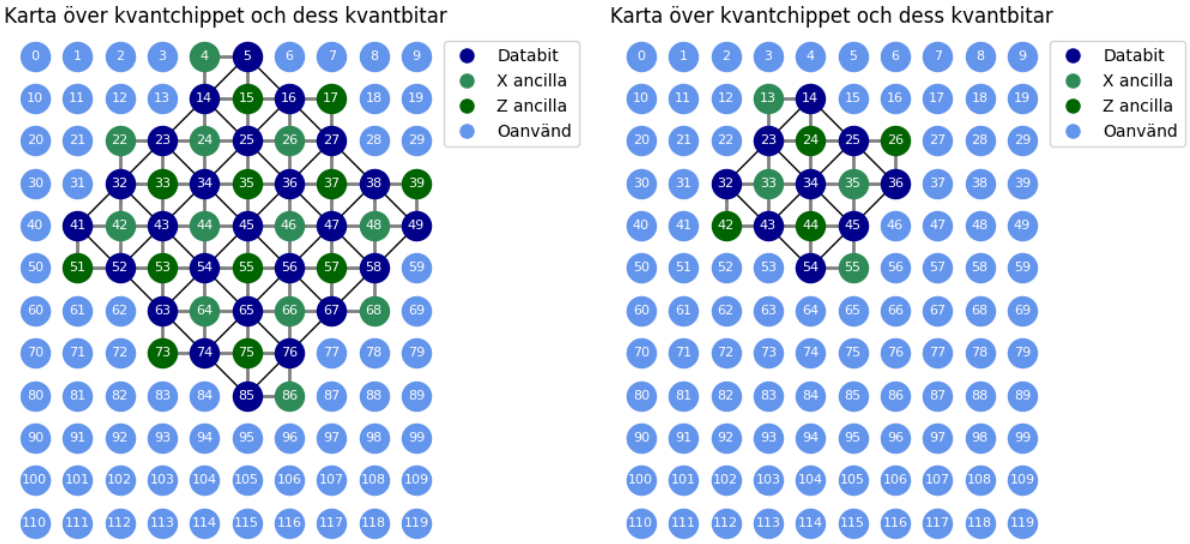
# Kapitel 3

# Metod

I det här kapitlet presenteras den metod som användes för att erhålla resultaten. De resurser som använts såsom IBM Miami kvantdatorn, Qiskit och ytkodskretsen introduceras, följt av hur datan genererats och slutligen hur modellen är uppbyggd och tränad samt hur avkodarna utvärderas.

## 3.1 Konstruktion av ytkodskretsen

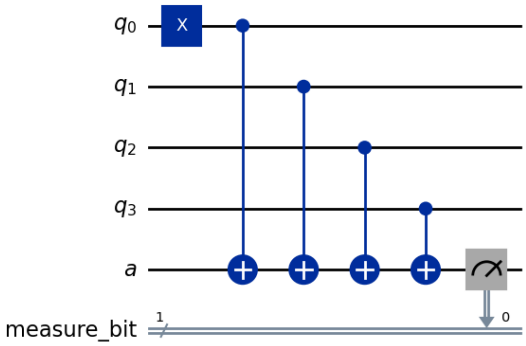
Kvantdatorn som använts för att genomföra projektet heter IBM Miami och har en Nighthawk r1-processor. Kvantdatorn IBM Miami består av ett gitter av $10 \times 12 = 120$ kvantbitar som kan ses i figur 3.1 och det finns tillgänglig felstatistik rörande varje enskild kvantbit och kopplingar mellan kvantbitar. Exempelvis har kvantbit 85 haft ett relativt stort mätfel genom hela projektet, runt 17% jämfört med runt 2% för de bättre bitarna. Kopplingen mellan vissa kvantbitar är också sämre än för andra och några få kopplingar fungerar inte. Denna statistik ändras över tid i takt med att sannolikheterna ändras. Ytkoden för $d = 5$ är placerad enligt figur 3.1a och enligt figur 3.1b för $d = 3$. De har placerats på chippet för att undvika de kvantbitar och kopplingar med högst fel i den mån det går, för ytkoden med kodavstånd 5 har implementationen varit särskilt utmanande. Av denna anledning behövde kvantbit 85 användas eftersom alternativa placeringar skulle inkludera fler kvantbitar och kopplingar av sämre kvalitet.




**(a)** Ytkod med kodavstånd 5 placerad på kvantchippet. **(b)** Ytkod med kodavstånd 3 placerad på kvantchippet.

**Figur 3.1:** Kvantbitarna för de implementerade ytkoderna på kvantchippet som använts under projektets gång. Blå noder representerar databitar, medan gröna noder utgör ancillabitar för mätning av stabilisatorer: ljusgröna för $X$-stabilisatorer och mörkgröna för $Z$-stabilisatorer. Linjerna visar kopplingar mellan kvantbitarna som används vid stabilisatormätningar. Ljusblå kvantbitar är fysiskt tillgängliga men används inte. Figurerna illustrerar ytkoder med kodavstånd 3 och 5.

Mjukvarubiblioteket Qiskit av IBM Quantum användes för att konstruera kvantkretsen med de metoder som beskrivs i avsnitt 2.1. Biblioteket möjliggör körning av kvantkretsar på såväl simulerad som experimentell data och användes för att skapa ytkoderna för projektet. Databitar representeras med ett kvantregister, där mätningarna av dem lagras med klassiska bitar som i sin tur representeras med ett klassiskt register. Se figur 3.2 för ett exempel på en kvantkrets i Qiskit.




**Figur 3.2:** $Z$-stabilisator gjord med Qiskit. En CNOT-grind är kopplad mellan varje datakvantbit $q$ och ancillabiten $a$. Ett bitfel appliceras på den första kvantbiten $q_0$, varpå alla kvantbitarna sammanflätas med hjälp av CNOT-grindarna. Ancillabiten mäts i slutändan och lagras i den klassiska biten measure_bit, vilket ger de klassiska utfallen 0 eller 1.

## 3.2 Data

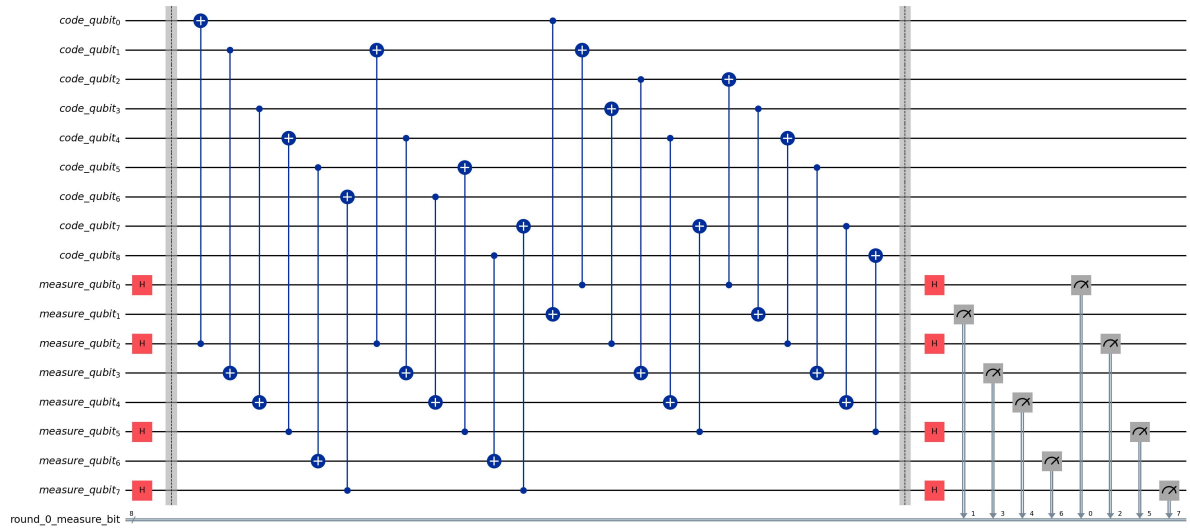
Modellen tränas och utvärderas på data från tre olika källor. Först används simulerad data genererad med Stim, som är högpresterande mjukvara för att simulera stabilisatorkretsar [29]. För den simulerade datan i denna fas används konstant brusnivå. Den resulterande modellen benämns hädanefter den förtränade modellen. Därefter tränas modellen på simulerad DEM-data som återspeglar brus likt data från experimentella körningar på IBM:s kvantprocessor Nighthawk r1. Slutligen används experimentell IBM-data för finjustering av modellen.

### 3.2.1 IBM-hårdvarudata

Experimentell data erhölls genom molntjänsten IBM Cloud vilket gav oss tillgång till hårdvaran tidigare beskrivet i avsnitt 3.1. Ytkoder för de två kodavstånden implementerades som en explicit kvantkrets i Qiskit. Implementationen inleds genom att definiera två kvantregister, ett för databitar som lagrar den logiska informationen och ett för ancillabitar som används för att extrahera felinformation.

För varje runda konstrueras kretsen och avslutas med mätning av samtliga ancillabitar. Ancillabitarna initialiseras inledningsvis i $|0\rangle$ medan databitarna initialiseras slumpmässigt i $Z$-basen. På ancillabitar som tillhör $X$-stabilisatorer appliceras en Hadamardgrind, därefter konstrueras stabilisatorerna med en sekvens av CNOT-grindar mellan ancillabitarna och deras grannar i fyra iterationer, vilket sammanflätar bitarna. Efter sammanflätningen appliceras ytterligare en Hadamardgrind på ancillabitarna som tillhör $X$-stabilisatorerna och slutligen mäts samtliga ancillabitar och resultatet sparas i klassiska databitar.

Denna process upprepas $T$ gånger, vilket ger en tidsserie av syndrom som möjliggör identifiering av fel över rum och tid. Efter mätning av den sista rundan $T$ mäts alla datakvantbitar direkt och resultatet används för att jämföra det slutgiltiga logiska tillståndet med det första. En komplett process med $T$ rundor kallas för sampling (eng. shot).




**Figur 3.3:** Illustration av ytkodskretsen ($d = 3$) med Qiskit vid $T = 0$. Datakvantbitarna är `code_qubit_0`,..., `code_qubit_8` medan ancillabitarna är `measure_qubit_0`,..., `measure_qubit_7`. Ancillabitarna mäts och lagras sedan i den klassiska biten `round0_measure_bit`.

### 3.2.2 DEM-data

En Detector Error Model (DEM) används för att beskriva hur fel på kretsnivå ger upphov till detektionshändelser och eventuella logiska fel. I modellen representeras varje fel som en stokastisk händelse kopplad till ett specifikt antal detektorer samt huruvida felen påverkar det logiska tillståndet.

Experimentell data som är till grund för modellen omvandlas till detektionshändelser för att uppskatta felsannolikheter genom empiriska korrelationer mellan detektorer. Grafstrukturen från modellen extraheras från en fördefinierad roterad ytkodskrets i Stim. De uppskattade sannolikheterna används därefter för att uppdatera vikterna för kanter i grafstrukturen. Uppskattningen av felsannolikheter utförs enligt de slutna uttrycken enligt Appendix E i [30]. Den resulterande modellen används sedan för att generera träningsdata till det neurala nätverket, med en brusprofil som efterliknar det brus som observerats i den experimentella datan.

## 3.3 GNN-RNN-modellen

Den modell som används är baserad på Fjeldså och Jonasson Johanssons modell [11], som är utformad för att kunna identifiera fasfel och bitfel, där vi endast undersöker bitfel. Den består av ett grafneuralt nätverk och ett antal grindade rekurrenta enheter som bildar en typ av rekurrent neuralt nätverk. Syndromgrafer som konstrueras från datan behandlas därefter av det grafneurala nätverket där de passerar genom flera grafkonvolutionella lager med ReLU-aktiveringsfunktion beskriven i avsnitt 2.3. Genom denna process genereras högdimensionella representationsvektorer för varje nod.

Därefter sammanfattas varje graf till en inbäddningsvektor genom global medelpooling, se avsnitt 2.3.4. Dessa matas sekventiellt in i det rekurrenta neurala nätverket, som modellerar tidsberoenden mellan mätningar. Slutligen används det sista dolda tillståndet från GRU-nätverket som indata till ett fullt kopplat lager som omvandlar tillståndet till en skalär. En sigmoidfunktion appliceras därefter för att erhålla ett värde mellan 0 och 1, detta värde tolkas som sannolikheten att ett fel inträffat. Värdet avrundas sedan till närmaste heltal, där 1 indikerar att ett fel har inträffat och 0 att inget fel har inträffat.

## 3.4 Träning av modellerna

Innan finjusteringen kunde genomföras behövde IBM-datan först bearbetas till samma typ av indata som modellerna använder. IBM-jobben lästes in från JSON-filer och bitsträngarna omvandlades till slutliga databitsmätningar och syndrom för varje runda. Eftersom ancillabitarna inte återställs fysiskt mellan rundorna beräknades de effektiva syndromen genom XOR mellan efterföljande mätningar. Detektionshändelser skapades sedan genom att jämföra syndromet mellan intilliggande tidssteg. En sista detektionshändelse konstruerades även genom att jämföra syndromet från den sista mätrundan med ett rekonstruerat slutligt syndrom, där ancillabitarnas värden beräknades från pariteten hos de slutliga databitsmätningarna.

IBM-datan hämtades från flera separata IBM-jobb som kördes vid olika tillfällen. Eftersom bruset på hårdvaran kan variera mellan sådana körningar behandlades varje jobb separat vid uppdelningen av datan. Varje jobb delades först slumpmässigt upp i tränings-, valideringss- och testdata. Därefter slogs motsvarande delar från de olika jobben ihop till en gemensam träningsmängd, valideringsmängd och testmängd. På så sätt innehöll varje del data från samtliga IBM-jobb.

För $d = 3$ användes en tvåfasig finjustering. I den första fasen tränades modellen på samplingar från en kalibrerad DEM, byggd från detektionshändelser i IBM-datan. I den andra fasen tränades modellen vidare på experimentell IBM-data. För $d = 5$ användes endast direkt finjustering på IBM-data, eftersom den DEM-baserade fasen inte gav en användbar anpassning för den brusigare $d = 5$-datan.

Under träningen användes endast samplingar där minst en detektionshändelse inträffade. Samplingar utan detektionshändelser innehåller ingen grafstruktur för modellen att behandla och motsvarar i praktiken det triviala fallet där modellen predikterar inget logiskt fel. Träningsbatcharna valdes slumpmässigt från de samplingar som hade detektionshändelser. För IBM-finjusteringen användes 2500 batcher med batchstorlek 64 per epok, vilket ger 160000 träningsexempel per epok. Detta är i samma storleksordning som antalet tillgängliga samplingar. Eftersom urvalet är slumpmässigt kan vissa samplingar förekomma flera gånger medan andra inte används under en viss epok.

Vid finjusteringen tränades modellen med Adam som optimerare och binär korsentropi som förlustfunktion. Modellens utdata tolkades som sannolikheten för ett logiskt flip. Inlärningshastigheten minskades exponentiellt mellan epoker enligt

$$\eta_t = \max(\eta_{\min}, \eta_0 \cdot 0.95^t),$$

där $t$ är epoknumret, $\eta_0$ är den initiala inlärningshastigheten och $\eta_{\min}$ är den minsta tillåtna inlärningshastigheten.

Valideringsdatan användes för tidig avbrytning (eng. early stopping), vilket innebär att träningen avbröts när valideringsnoggrannheten inte förbättrades. Under träningen sparades den modellversion som hade högst valideringsnoggrannhet, och efter avslutad träning återställdes modellen till denna version. Hyperparametrarna i tabell 3.1 är de slutliga inställningarna efter flera inledande tester med bland annat inlärningshastighet, regularisering och olika träningsscheman. I tabellen avser träningssteg antalet optimeringssteg fram till dess att den bästa valideringsnoggranheten uppnåddes. Extra epoker kunde därför köras under tidig avbrytning. Däremot återställdes den slutliga modellen alltid till modellversionen med bäst valideringsnoggrannhet. Antalet träningsexempel beräknas som träningssteg gånger batchstorlek. För finjusteringen på experimentell IBM-data avser detta antalet samplade träningsexempel, inte antalet unika samplingar i träningsdatan. Eftersom batcherna valdes slumpmässigt kunde samma samplingar förekomma flera gånger.

**Tabell 3.1:** Hyperparametrar för de förtränade d3- och d5-modellerna från Fjeldså och Jonasson Johansson [11] och för finjusteringen i detta arbete. De modeller som används här har samma GNN-RNN-arkitektur med 532 385 träningsbara parametrar, grafembeddingdimension 256, dold GRU-dimension 128 och fyra GRU-lager.

| | **Förträning** | | **Finjustering (DEM)** | | **Finjustering (IBM)** | | | |
|---|---|---|---|---|---|---|---|---|
| | $d = 3, T = 99$ | $d = 5, T = 49$ | $d = 3, T = 10$ | $d = 3, T = 20$ | $d = 3, T = 10$ | $d = 3, T = 20$ | $d = 5, T = 10$ | $d = 5, T = 20$ |
| Träningssteg | 38,4k | 76,8k | 16,8k | 20,8k | 87,5k | 32,5k | 157,5k | 302,5k |
| Batchstorlek | 2048 | 2048 | 256 | 256 | 64 | 64 | 64 | 64 |
| Träningsexempel | 78,6M | 157M | 4,3M | 5,3M | 5,6M | 2,1M | 10,1M | 19,4M |
| Initial $\eta_0$ | 1e-3 | 1e-3 | 1e-4 | 1e-4 | 1,5e-5 | 1,5e-5 | 1e-5 | 1e-5 |
| Minsta $\eta_{\min}$ | 1e-4 | 1e-4 | 1e-6 | 1e-6 | 1e-6 | 1e-6 | 5e-7 | 5e-7 |

## 3.5 MWPM-modellen

Som jämförelse till GNN-RNN-modellen implementerades en MWPM-avkodare med PyMatching. Utgångspunkten var en MWPM-kod för repetitionskod på IBM-hårdvara, där kantvikter uppskattades från observerade korrelationer i detektionshändelser. I detta arbete anpassades samma idé till den roterade ytkoden på IBM Miami.

MWPM-grafen byggdes endast från detektionshändelser på $Z$-stabilisatorer. Varje möjlig $Z$-detektor i alla tidslager representerades som en nod. Kanter lades till mellan $Z$-stabilisatorer som delar en databit, mellan samma stabilisator i två efterföljande tidslager samt mellan en stabilisator och en grannstabilisator i nästa tidslager. Stabilisatorer vid kodens kant kopplades även till en virtuell kant, och de kantkopplingar som motsvarade den logiska operatorn markerades som logiska.

Kantvikterna bestämdes från korrelationer i tränings- och valideringsdatan. För två detektorer $i$ och $j$ uppskattades en effektiv felsannolikhet $p_{ij}$ från hur ofta de gav utslag var för sig och hur ofta de gav utslag samtidigt. Kantvikten sattes till $w_{ij} = -\log(p_{ij})$, så att mer sannolika felkedjor fick lägre vikt.

## 3.6 Jämförelse

GNN-RNN-avkodaren och MWPM-avkodaren utvärderades på samma testdata för varje kombination av kodavstånd $d$ och antal syndromrundor $T$. Testdatan hölls separat från tränings- och valideringsdatan. För GNN-RNN-modellen användes valideringsdatan för att avbryta träningen i förtid om prestationen försämrades, medan MWPM använde tränings- och valideringsdatan för att uppskatta kantvikter.

Vid den slutliga utvärderingen användes hela testmängden, vilket innebär att även samplingar utan detektionshändelser ingick i testresultatet. Samplingar utan detektionshändelser räknades som prediktion av inget logiskt fel, medan samplingar med detektionshändelser avkodades av modellen.

För avkodarna MWPM, förtränat GNN-RNN, DEM-tränat GNN-RNN och färdigtränat GNN-RNN rapporteras noggrannhet, logisk felfrekvens och logisk felfrekvens per runda. Noggrannheten definieras här som andelen korrekt predikterade logiska utfall. Vi rapporterar den totala logiska felfrekvensen som

$$P_L = 1 - \text{noggrannhet}.$$

Den logiska felfrekvensen per runda $p_r$ definieras genom antagandet att oberoende och identiskt fördelade logiska fel kan uppstå i varje runda, och att pariteten av dessa fel avgör det slutliga logiska mätresultatet

$$P_L = \frac{1}{2}\left[1 - (1 - 2p_r)^T\right].$$

Löser man ut $p_r$ fås

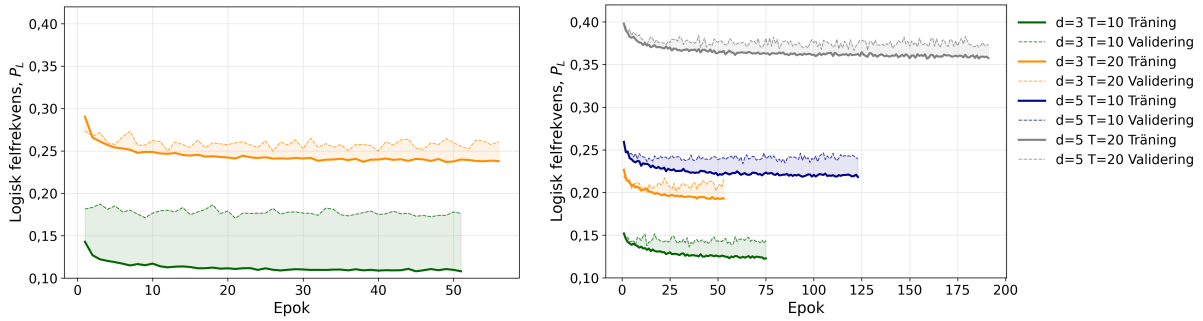
$$p_r = \frac{1}{2}\left[1 - (1 - 2P_L)^{1/T}\right].$$

För små $P_L$ reduceras detta ungefär till $P_L/T$, men för större $P_L$ blir värdet högre eftersom ett jämnt antal logiska fel tar ut varandra i den slutliga pariteten.

# Kapitel 4

# Resultat

Den konstruerade ytkoden för kodavstånd 3 respektive 5 har framgångsrikt implementerats och körts på kvantdatorn IBM Miami samt avkodats. Figur 4.1 visar den logiska felfrekvensen som funktion av epok under träningen av respektive GNN-RNN-modell, där heldragna linjer representerar träningsdata och streckade linjer representerar valideringsdata. Delfigur 4.1a visar träning av det förtränade nätverket på DEM-genererad data medan delfigur 4.1b visar efterföljande träning på experimentell IBM-data. I båda fallen skedde valideringen med IBM-data. Notera att modellerna för $d = 5$ endast tränats på experimentell data och inte på DEM-data, till skillnad från modellerna för $d = 3$.




**(a)** Logisk felfrekvens som funktion av epok under träningen på simulerad DEM-data för GNN-RNN-avkodaren.

**(b)** Logisk felfrekvens som funktion av epok under träningen på experimentell IBM-data för GNN-RNN-avkodaren.

**Figur 4.1:** Logisk felfrekvens som funktion av epok under träningen av GNN-RNN-avkodaren. Valideringen har skett med IBM-data. Heldragna linjer visar träningsdata och streckade linjer visar valideringsdata. Resultat visas för $d = 3$ och $d = 5$ samt $T = 10$ och $T = 20$.

Detektortätheten för den data som använts för att träna, validera och utvärdera GNN-RNN-modellerna redovisas i tabell 4.1. Detektortätheten definieras som andelen möjliga detektorpositioner som gav upphov till en detektionshändelse. Tabellen visar att data för $d = 5$ innehöll betydligt fler detektionshändelser per sampling än data för $d = 3$. Skillnaden är särskilt tydlig för $T = 20$, där detektortätheten var högst.

**Tabell 4.1:** Detektortäthet i IBM-datan som användes för träning och utvärdering. Tätheten definieras som andelen detektionshändelser av alla möjliga detektorpositioner.

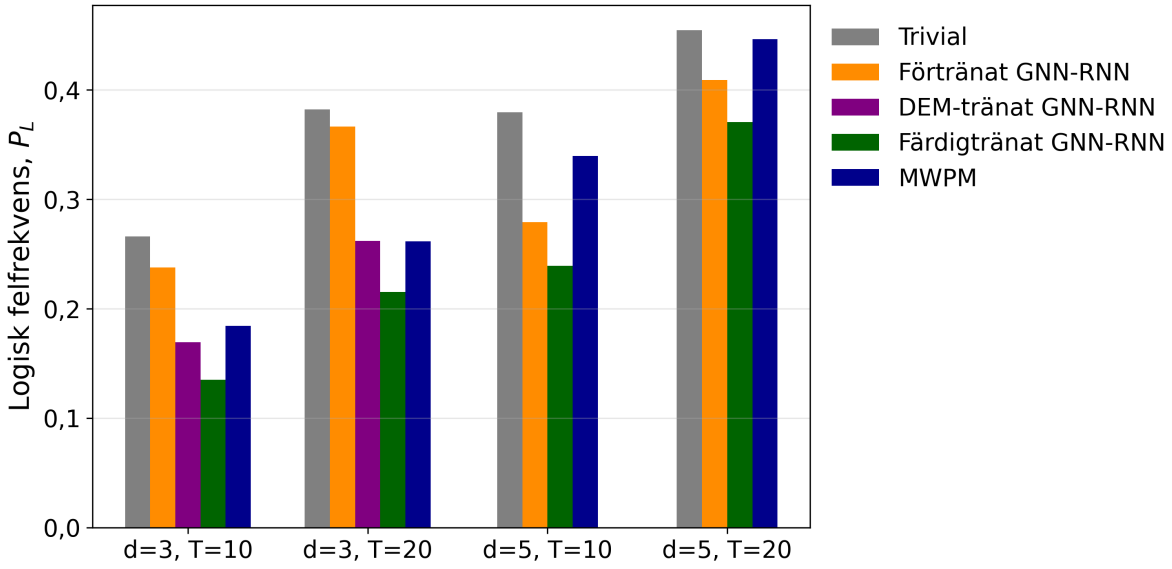
| $d$ | $T$ | Antal jobb | Antal samplingar | Detektioner/sampling | Detektortäthet |
|---|---|---|---|---|---|
| 3 | 10 | 2 | 200000 | 8,85 | 10,06% |
| 3 | 20 | 3 | 200000 | 22,90 | 13,63% |
| 5 | 10 | 2 | 200000 | 48,96 | 18,55% |
| 5 | 20 | 2 | 200000 | 115,52 | 22,92% |

Det genomsnittliga antalet detektionshändelser per sampling för experimentell IBM-data och simulerad DEM data jämförs i tabell 4.2. För $d = 5$ är både den logiska felfrekvensen och antalet detektionshändelser per sampling högre för DEM-datan än för den experimentella datan. För $d = 3$ är dessa värden däremot relativt lika.

**Tabell 4.2:** Jämförelse mellan experimentell IBM-data och simulerad DEM-data. Tabellen visar det genomsnittliga antalet detektionshändelser per sampling för $d = 3$ och $d = 5$ med $T = 10$.

| $d$ | T | Datatyp | Detektionshändelser per sampling |
|---|---|---|---|
| 3 | 10 | Experimentell | 8,85 |
| 3 | 10 | Simulerad | 11,63 |
| 5 | 10 | Experimentell | 48,96 |
| 5 | 10 | Simulerad | 78,66 |

Den färdigtränade GNN-RNN-avkodaren uppvisar lägre logisk felfrekvens än MWPM och samtliga andra avkodningsalternativ för alla undersökta värden på $d$ och $T$. Detta visas i figur 4.2 där den logiska felfrekvensen för de olika avkodarna som testats jämförs. Resultaten gäller vid avkodning av ett testjobb med 20 000 samplingar experimentell IBM-data. Här presenteras statistik för MWPM-avkodaren samt GNN-RNN-avkodaren i olika stadier av dess träning och även resultat för en trivial avkodare, det vill säga en avkodare som alltid gissar på det mest förekommande resultatet.



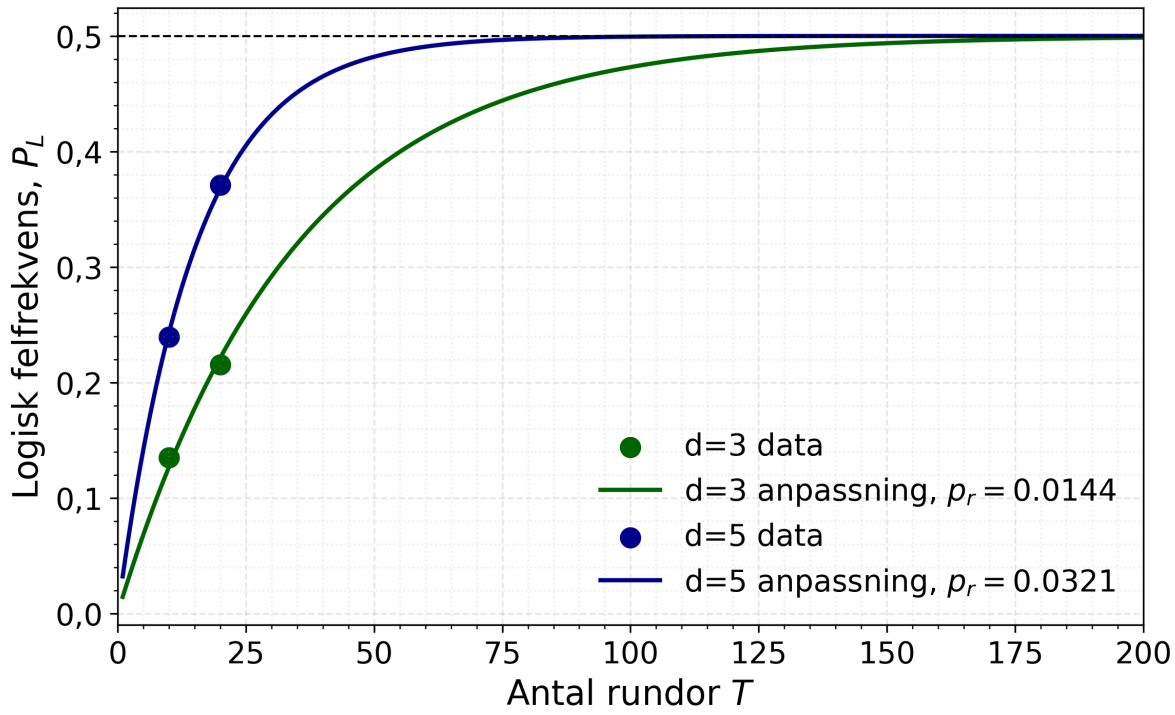

**Figur 4.2:** Slutlig avkodningsprestanda för den förtränade GNN-RNN-avkodaren, DEM-tränade GNN-RNN-avkodaren, färdigtränade GNN-RNN-avkodaren, MWPM och den triviala avkodaren för $d = 3$ och $d = 5$ samt $T = 10$ och $T = 20$.

Den logiska felfrekvensen för samtliga avkodningsalternativ återfinns även i tabell 4.3, avkodarnas noggrannhet och logiska felfrekvens per runda för samma testjobb på 20 000 samplingar.

**Tabell 4.3:** Avkodningsprestanda på experimentell IBM-testdata. Tabellen visar noggrannhet, logisk felfrekvens $P_L$ samt logisk felfrekvens per runda $p_r$. Noggrannheten definieras här som andelen korrekt predikterade logiska utfall. Osäkerheterna har beräknats som binomiala standardfel med $N = 20\,000$ samplingar. Det bästa värdet för varje kombination av $d$ och $T$ är markerat i fetstil.

| $d$ | $T$ | Avkodare | Noggrannhet | $P_L$ | $p_r$ |
|---|---|---|---|---|---|
| 3 | 10 | Ingen | $0,734 \pm 0,003$ | $0,266 \pm 0,003$ | 0,0365 |
| 3 | 10 | MWPM | $0,816 \pm 0,003$ | $0,184 \pm 0,003$ | 0,0225 |
| 3 | 10 | Förtränat GNN-RNN | $0,762 \pm 0,003$ | $0,238 \pm 0,003$ | 0,0313 |
| 3 | 10 | DEM-tränat GNN-RNN | $0,831 \pm 0,003$ | $0,169 \pm 0,003$ | 0,0203 |
| 3 | 10 | Färdigtränat GNN-RNN | $\mathbf{0,865 \pm 0,002}$ | $\mathbf{0,135 \pm 0,002}$ | **0,0144** |
| 3 | 20 | Ingen | $0,618 \pm 0,003$ | $0,382 \pm 0,003$ | 0,0349 |
| 3 | 20 | MWPM | $0,738 \pm 0,003$ | $0,262 \pm 0,003$ | 0,0182 |
| 3 | 20 | Förtränat GNN-RNN | $0,633 \pm 0,003$ | $0,367 \pm 0,003$ | 0,0320 |
| 3 | 20 | DEM-tränat GNN-RNN | $0,738 \pm 0,003$ | $0,262 \pm 0,003$ | 0,0182 |
| 3 | 20 | Färdigtränat GNN-RNN | $\mathbf{0,785 \pm 0,003}$ | $\mathbf{0,215 \pm 0,003}$ | **0,0139** |
| 5 | 10 | Ingen | $0,620 \pm 0,003$ | $0,380 \pm 0,003$ | 0,0663 |
| 5 | 10 | MWPM | $0,660 \pm 0,003$ | $0,340 \pm 0,003$ | 0,0538 |
| 5 | 10 | Förtränat GNN-RNN | $0,721 \pm 0,003$ | $0,279 \pm 0,003$ | 0,0392 |
| 5 | 10 | Färdigtränat GNN-RNN | $\mathbf{0,761 \pm 0,003}$ | $\mathbf{0,239 \pm 0,003}$ | **0,0315** |
| 5 | 20 | Ingen | $0,545 \pm 0,004$ | $0,455 \pm 0,004$ | 0,0566 |
| 5 | 20 | MWPM | $0,553 \pm 0,004$ | $0,447 \pm 0,004$ | 0,0529 |
| 5 | 20 | Förtränat GNN-RNN | $0,591 \pm 0,003$ | $0,409 \pm 0,003$ | 0,0408 |
| 5 | 20 | Färdigtränat GNN-RNN | $\mathbf{0,629 \pm 0,003}$ | $\mathbf{0,371 \pm 0,003}$ | **0,0327** |

I figur 4.3 visas en extrapolering av den logiska felfrekvensen som funktion av antalet rundor $T$, baserad på den uppmätta logiska felfrekvensen per runda för den färdigtränade GNN-RNN-avkodaren. Extrapoleringen har utförts separat för $d = 3$ och $d = 5$ enligt $P_L = \frac{1}{2}(1 - (1 - 2p_r)^T)$ där $p_r$ representerar den logiska felfrekvensen per runda. Extrapoleringen antyder att modellen för $d = 3$ fungerar för högre $T$ än modellen för $d = 5$.




**Figur 4.3:** Extrapolering av den logiska felfrekvensen $P_L$ som funktion av antalet rundor $T$ för den färdigtränade GNN-RNN-avkodaren. Anpassningarna baseras på den uppmätta logiska felfrekvensen per runda för $d = 3$ respektive $d = 5$. Den streckade linjen markerar $P_L = 0.5$.

Vid förflyttning av ytkoden på kvantchippet observerades även ett beroende av den fysiska placeringen. För förflyttningar ett jämnt antal steg i sidled från positionen i figur 3.1 erhölls jämförbara resultat, med variationer beroende på de lokala felparametrarna för kvantbitar och kopplingar. För förflyttningar ett ojämnt antal steg i sidled observerades däremot genomsnittliga detektionsfrekvenser omkring 0,5 per runda för i stort sett samtliga ancillabitar och rundor. Detta observerades för både $d = 3$ och $d = 5$.

# Kapitel 5

# Diskussion

Syftet med projektet var att jämföra och undersöka om den färdigtränade GNN-RNN-avkodaren kunde prestera bättre än MWPM. Resultatet i tabell 4.3 påvisar att detta är fallet för alla testade kodavstånd och $T$. Dessutom påvisar resultatet att GNN-RNN-avkodaren överträffar MWPM för $d = 5$ även då det var relativt begränsad data för finjusteringen eftersom endast experimentellt uppmätt data användes för det. Däremot så förekom det en hel del störningar och brus på kvantprocessorn under samplingarna vilket försämrar felkorrigeringsförmågan överlag, men potentiellt sätt kan detta påverka MWPM värre än för den tränade modellen. MWPM har inte samma möjlighet att anpassa sig efter bruset då den bygger på en förbestämd algoritm oberoende av kvantprocessor. Därmed hade resultatet förmodligen påverkats något till MWPM:s fördel om mindre störningar och brus hade förekommit på kvantprocessorn. Rimligtvis hade däremot även GNN-RNN-avkodaren gynnats av en mer koherent och störningsfri kvantprocessor och dessutom bygger MWPM på vissa förenklade antaganden som den tränade modellen inte antar. Ytterligare är det heller inte så ovanligt att det förekommer mycket brus och störningar vid körning med kvantdatorer. Det som resultatet säkert visar är att GNN-RNN-avkodaren presterade bättre än MWPM på kvantdatorn IBM Miami, och även om det inte går att påstå säkert, tyder ändå resultatet på att en vältränad GNN-RNN-baserad avkodare förmodligen överträffar MWPM mer generellt också.

Som redovisas i tabell 4.2 blev detektorhändelserna per sampling högre för den simulerade datan än för den experimentella trots att den simulerade var baserad på den experimentella. Denna ökning var främst märkbar för $d = 5$ och innebar att den simulerade datan blev betydligt mer brusig än den experimentella och därav inte speciellt representativ. Det är på grund av detta $d = 5$-modellerna inte tränades på någon simulerad DEM-data utan endast experimentell IBM-data.

Detektortätheten i den experimentella IBM-datan, se tabell 4.2, var även betydligt högre än de detektionssannolikheter som rapporterats för motsvarande ytkodsexperiment på andra kvantprocessorer, exempelvis Googles Sycamore-arkitektur [31]. Måtten är inte identiskt definierade men beskriver båda andelen detektionshändelser per möjlig detektorposition. Resultatet tyder på att IBM Miami uppvisade en relativt hög fysisk felnivå under de körningar som användes i projektet, vilket sannolikt försvårade felkorrigeringen för samtliga avkodare.

Extrapoleringen av den logiska felfrekvensen i figur 4.3 indikerar att avkodningen för $d = 3$ förblir effektiv upp till ett större antal rundor än för $d = 5$. För $d = 3$ når den extrapolerade logiska felfrekvensen $P_L = 0,5$ först vid ungefär $T \approx 160$, medan motsvarande värde för $d = 5$ uppnås redan vid ungefär $T \approx 75$. Extrapoleringen baseras dock endast på två datapunkter för respektive kodavstånd och bör därför tolkas med försiktighet. För att kunna dra säkrare slutsatser skulle ytterligare mätningar för fler värden på $T$ behöva genomföras.

Resultaten visar att den färdigtränade GNN-RNN-avkodaren presterar bättre än MWPM vid korrigering av kvantfel. Samtidigt är modellens prestanda starkt beroende av det kvantchip som den har tränats på, eftersom de enskilda kvantbitarna uppvisar olika typer av instabiliteter. Kvantbitarnas kvalitet kan dessutom variera över tid då de tenderar att försämras till följd av driftsrelaterade effekter, men kan även förbättras genom kalibrering och underhåll. Detta innebär att den tränade modellen endast är giltig för det specifika chippet den tränats på och under en begränsad tidsperiod efter träningen. I praktiken betyder detta att ett grafneuralt nätverk sannolikt endast ger ett prestandaövertag gentemot MWPM om det nyligen är tränat på den kvantprocessor där det tillämpas. Följaktligen kan den modell som tagits

fram i detta arbete inte betraktas som en generell lösning för kvantfelskorrektion på olika kvantdatorer. Resultaten indikerar dock att om en modell tränas i nära anslutning till en sampling, kan den användas effektivt för kvantfelskorrektion och sannolikt uppnå bättre prestanda än MWPM.

Det finns flera intressanta aspekter till detta arbete som går att undersöka vidare. Som nämns i avsnitt 4 gav vissa placeringar av ytkoden detektionsfrekvenser runt 0,5 per runda för nästan alla ancillabitar och rundor vilket tyder på att dessa stabilisatormätningar i princip ger slumpmässiga resultat. Orsaken till detta kunde inte fastställas inom ramen för projektet men skulle vara intressant att undersöka vidare.

Det skulle även vara av intresse att testa modellen på andra kvantdatorer eller med ytkoder placerade på andra delar av kvantchippet. Vidare skulle avkodaren kunna utvärderas för fler värden på $T$ för att ge en bättre förståelse av hur prestandan utvecklas vid längre körningar.

## 5.1 Samhälleliga och etiska aspekter

Kvantdatorer, och maskininlärning är två av de mest lovande teknologiska innovationer som uppkommit under de senaste decennierna [32], [33]. Som tidigare nämns i inledningen har kvantdatorer, som är beroende av kvantfelskorrektion, potential att lösa vissa problem snabbare än traditionella datorer. Trots dessa möjligheter medför teknikutvecklingen även etiska och samhälleliga utmaningar.

En central aspekt är säkerhet. Storskaliga kvantdatorer skulle potentiellt kunna bryta delar av dagens krypteringssystem [34]. Exempelvis finns det en risk att delar av krypteringssystemet RSA bryts med tillräckligt många fysiska kvantbitar och effektiv kvantfelskorrektion. I sin tur innebär detta att all lagrad data skyddad med RSA kan dekrypteras och göras tillgänglig. För den enskilda personen kan detta inkludera lösenord, bankuppgifter och personliga meddelanden. Detta understryker behovet av parallell utveckling av kvantsäker kryptografi och internationell samverkan kring forskning om kvantdatorer.

Dessutom medför användningen av maskininlärning inom kvantfelskorrektion flera etiska och samhälleliga aspekter. Stora GPU-kluster, som utnyttjades i vårt projekt, kräver stora mängder energi och i sin tur leder till koldioxidutsläpp. Studier har visat att träning av stora språkmodeller kan ge upphov till utsläpp på omkring 300 000 kg koldioxid, vilket motsvarar ungefär 125 tur- och returresor med flyg mellan New York och Peking [35]. Det går heller inte att undkomma att träning av stora AI-modeller är dyra, i synnerhet på grund av hårdvaran och elektriciteten som krävs [36]. Maskininlärningsmodellen som tränas i detta projekt är jämförelsevis med stora språkmodeller mycket liten, vilket innebär avsevärt mycket lägre energiförbrukning och koldioxidutsläpp. Det kan däremot vara av intresse att betrakta energiförbrukningen för träning av maskininlärningsmodellen när den ska jämföras med MWPM, då den sämre prestandan för MWPM skulle kunna vägas upp för av mindre kostnader och utsläpp. Utöver de aspekter som är direkt kopplade till detta arbete finns även mer generella etiska frågor kring maskininlärning, såsom som diskriminering [37], en maktkoncentration till ett fåtal forskningsinstitutioner och företag [38] samt en avsaknad av transparens vid beslutsfattning [39]. Samtidigt har maskininlärning potential att bidra till betydande samhällsnytta genom att möjliggöra mer robusta kvantdatorer, vilket på sikt kan användas inom flertalet områden som nämnts i inledningen och potentiellt minska koldioxidutsläpp [35].

Sammanfattningsvis är de etiska och samhälleliga aspekterna av kvantdatorer och maskininlärning både mångfacetterade och invecklade. Den snabba teknologiska utvecklingen inom dessa områden medför konsekvenser som sträcker sig bortom de tekniska tillämpningarna, vilket understryker behovet av ett ansvarsfullt, långsiktigt och medvetet förhållningssätt.

## 5.2 Slutsats

I projektet har det visats att GNN-RNN-avkodaren överträffade MWPM på kvanttorn IBM Miami, vilket också visades i Fjeldså och Jonasson Johanssons examensarbete [11], fast i fallet där endast simulerad data användes. Därmed stärks detta resultat nu ytterligare och en GNN-RNN-baserad avkodare står därmed som ett starkt alternativ för kvantfelskorrektion gentemot MWPM. Däremot, som diskuterats tidigare, kan det inte helt fastslås att GNN-RNN-avkodaren överträffar MWPM generellt eftersom förutsättningarna på kvantprocessorn kan ha gynnat den tränade modellen gentemot MWPM. Ytterligare har MWPM även fördelen att den fungerar relativt väl för alla olika kvantdatorer, medan en tränad modell förmodligen skulle behöva tränas om på en ny kvantprocessor.

Några framtida problemställningar som därmed kan tas för att bygga vidare på projektet är att träna flera olika modeller på andra kvantdatorer för att undersöka om GNN-RNN-avkodaren fortfarande överträffar MWPM. Modellerna skulle förmodligen behöva tränas om på de nya kvantdatorerna eftersom fördelningen av felbenägna kvantbitar på processorn kan variera. Däremot hade det även varit intressant att testa hur väl den tränade modellen för detta projekt skulle prestera på en annan kvantdator. Dessutom skulle även en ny modell kunna skapas i syfte att försöka träna en modell som presterar så bra som möjligt på flera olika kvantdatorer.

Slutligen visar resultatet att GNN-RNN-avkodaren presterade bättre än MWPM och därmed kan slutsatsen dras att denna metod är bättre för kvantfelskorrektion på kvantdatorn IBM Miami. Resultatet är däremot inte säkert nog för att bekräfta att detta gäller generellt för samtliga kvantdatorer och fler tester skulle behöva genomföras för att bekräfta detta. Däremot går det att säga att framtiden för maskininlärning inom kvantfelskorrektion troligen har stor potential.

# Referenser


[1] A. Abbas m. fl., "Challenges and opportunities in quantum optimization," *Nature Reviews Physics*, årg. 6, nr 12, s. 718–735, okt. 2024. DOI: 10.1038/s42254-024-00770-9.

[2] R. Ur Rasool, H. F. Ahmad, W. Rafique, A. Qayyum, J. Qadir och Z. Anwar, "Quantum computing for healthcare: A review," *Future Internet*, årg. 15, nr 3, s. 94, febr. 2023. DOI: 10.3390/fi15030094.

[3] V. Raseena, "Quantum computing: foundations, algorithms, and emerging applications," *Frontiers in Quantum Science and Technology*, årg. 4, nr 1723319, dec. 2025, Section: Basic Science for Quantum Technologies. DOI: 10.3389/frqst.2025.1723319. URL: https://doi.org/10.3389/frqst.2025.1723319.

[4] T. D. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe och J. L. O'Brien, "Quantum computers," *nature*, årg. 464, nr 7285, s. 45–53, mars 2010. DOI: 10.1038/nature08812.

[5] J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, årg. 2, s. 79, aug. 2018, ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79. URL: https://doi.org/10.22331/q-2018-08-06-79.

[6] M. A. Nielsen och I. L. Chuang, *Quantum Computation and Quantum Information 10th Anniversary Edition*. Cambridge, UK: Cambridge University Press, 2000, ISBN: 9780521635035. hämtad 15 maj 2026. URL: https://doi.org/10.1017/CBO9780511976667.

[7] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Physical review A*, årg. 52, nr 4, R2493, okt. 1995. DOI: 10.1103/PhysRevA.52.R2493.

[8] W. P. Livingston, M. S. Blok, E. Flurin, J. Dressel, A. N. Jordan och I. Siddiqi, "Experimental demonstration of continuous quantum error correction," *Nature communications*, årg. 13, nr 1, s. 2307, april 2022. DOI: 10.1038/s41467-022-29906-0.

[9] O. Higgott och C. Gidney, "Sparse Blossom: correcting a million errors per core second with minimum-weight matching," *Quantum*, årg. 9, s. 1600, jan. 2025, ISSN: 2521-327X. DOI: 10.22331/q-2025-01-20-1600. URL: https://doi.org/10.22331/q-2025-01-20-1600.

[10] D. K. Tuckett, S. D. Bartlett och S. T. Flammia, "Ultrahigh Error Threshold for Surface Codes with Biased Noise," *Phys. Rev. Lett.*, årg. 120, s. 050 505, 5 jan. 2018. DOI: 10.1103/PhysRevLett.120.050505. URL: https://link.aps.org/doi/10.1103/PhysRevLett.120.050505.

[11] O. Fjeldså och G. Jonasson Johansson, "Sequential Graph-Based Decoding of the Surface Code using a Hybrid Graph and Recurrent Neural Network Model," Examensarbete för masterexamen, Instutitionen för fysik, Chalmers tekniska högskola, Göteborg, Sverige, 2025. URL: https://hdl.handle.net/20.500.12380/309551.

[12] M. Lange, "Decoding the surface code using graph neural networks," examensarb., Institutionen för fysik, Göteborgs universitet, Göteborg, Sverige, 2023. URL: https://gu-se-primo.hosted.exlibrisgroup.com/permalink/f/rmbr1s/46GUB_GUPEA2077/78810.

[13] E. Chae, J. Choi och J. Kim, "An elementary review on basic principles and developments of qubits for quantum computing," *Nano Convergence*, årg. 11, s. 11, mars 2024. DOI: 10.1186/s40580-024-00418-5. URL: https://doi.org/10.1186/s40580-024-00418-5.

[14] H. A. Bhat, F. A. Khanday, B. K. Kaushik, F. Bashir och K. A. Shah, "Quantum Computing: Fundamentals, Implementations and Applications," *IEEE Open Journal of Nanotechnology*, årg. 3, s. 61–77, maj 2022. DOI: 10.1109/OJNANO.2022.3178545.

[15] Glosser.ca, *Bloch Sphere.svg*. hämtad 14 maj 2026. URL: https://commons.wikimedia.org/wiki/File:Bloch_Sphere.svg.

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

[17] A. G. Fowler, M. Mariantoni, J. M. Martinis och A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Phys. Rev. A*, årg. 86, s. 032 324, 3 sept. 2012. DOI: 10.1103/PhysRevA.86.032324. URL: https://link.aps.org/doi/10.1103/PhysRevA.86.032324.

[18] A. M. Dalzell m. fl., "Quantum error correction with the surface code," i *Quantum Algorithms: A Survey of Applications and End-to-end Complexities*, Cambridge: Cambridge University Press, 2025, s. 320–324.

[19] V. Kolmogorov, "Blossom V: A new implementation of a minimum cost perfect matching algorithm," *Mathematical Programming Computation*, årg. 1, nr 1, s. 43–67, juli 2009. DOI: 10.1007/s12532-009-0002-8.

[20] J. Edmonds, "Paths, trees, and flowers," *Canadian Journal of Mathematics*, årg. 17, nr 3, s. 449–467, nov. 1965. DOI: 10.4153/CJM-1965-045-4.

[21] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press San Francisco, CA, USA, 2015. hämtad 4 maj 2026. URL: http://neuralnetworksanddeeplearning.com/.

[22] I. Goodfellow, Y. Bengio och A. Courville, *Deep Learning*. MIT Press Cambridge, MA, USA: 2016, http://www.deeplearningbook.org. hämtad 4 maj 2026. URL: https://doi.org/10.4258/hir.2016.22.4.351.

[23] D. E. Rumelhart, G. E. Hinton och R. J. Williams, "Learning representations by back-propagating errors," *Nature*, årg. 323, s. 533–536, okt. 1986. DOI: 10.1038/323533a0.

[24] D. P. Kingma och J. Ba, "Adam: A Method for Stochastic Optimization," i *The Thirteenth International Conference on Learning Representations*, Singapore, 2015. hämtad 5 maj 2026. URL: https://arxiv.org/abs/1412.6980.

[25] K. Cho m. fl., "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," i *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, okt. 2014, s. 1724–1734. DOI: 10.3115/v1/D14-1179. URL: https://aclanthology.org/D14-1179/.

[26] C. Morris m. fl., "Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks," *CoRR*, årg. abs/1810.02244, 2018. arXiv: 1810.02244. URL: http://arxiv.org/abs/1810.02244.

[27] Y. Dogan, "A new global pooling method for deep neural networks: Global average of top-k max-pooling," *Traitement du signal*, årg. 40, nr 2, s. 577–587, april 2023. DOI: https://doi.org/10.18280/ts.400216.

[28] K. Weiss, T. M. Khoshgoftaar och D. Wang, "A survey of transfer learning," *Journal of Big Data*, årg. 3, nr 1, s. 9, maj 2016. DOI: 10.1186/s40537-016-0043-6.

[29] C. Gidney, "Stim: a fast stabilizer circuit simulator," *Quantum*, årg. 5, s. 497, juli 2021, ISSN: 2521-327X. DOI: 10.22331/q-2021-07-06-497. URL: https://doi.org/10.22331/q-2021-07-06-497.

[30] A. Remm m. fl., *Experimentally Informed Decoding of Stabilizer Codes Based on Syndrome Correlations*, febr. 2025. arXiv: 2502.17722 [quant-ph]. URL: https://arxiv.org/abs/2502.17722.

[31] Google Quantum AI and Collaborators, "Quantum error correction below the surface code threshold," *Nature*, årg. 638, s. 920–926, 2025. DOI: 10.1038/s41586-024-08449-y. URL: https://doi.org/10.1038/s41586-024-08449-y.

[32] Q. A. Memon, M. Al Ahmad och M. Pecht, "Quantum Computing: Navigating the Future of Computation, Challenges, and Technological Breakthroughs," *Quantum Reports*, årg. 6, nr 4, s. 627–663, 2024, ISSN: 2624-960X. DOI: 10.3390/quantum6040039.

[33] A. Mayr m. fl., "Machine Learning in Production – Potentials, Challenges and Exemplary Applications," *Procedia CIRP*, årg. 86, s. 49–54, 2019, ISSN: 2212-8271. DOI: https://doi.org/10.1016/j.procir.2020.01.035. URL: https://www.sciencedirect.com/science/article/pii/S2212827120300445.

[34] M. Mosca, "Cybersecurity in an Era with Quantum Computers: Will We Be Ready?" *IEEE Security Privacy*, årg. 16, nr 5, s. 38–41, sept. 2018. DOI: 10.1109/MSP.2018.3761723.

[35] P. Dhar, "The carbon impact of artificial intelligence," *Nature Machine Intelligence*, årg. 2, nr 8, s. 423–425, aug. 2020. DOI: 10.1038/s42256-020-0219-9.

[36] 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*, Florence, Italy: Association for Computational Linguistics, juli 2019, s. 3645–3650. DOI: 10.18653/v1/P19-1355. URL: https://aclanthology.org/P19-1355/.

[37] J. Zou och L. Schiebinger, "AI can be sexist and racist—it's time to make it fair," *Nature*, årg. 559, nr 7714, s. 324–326, juli 2018. DOI: 10.1038/d41586-018-05707-8.

[38] W. K. Lau. “Between Concentration and Decentralization: The Changing Landscape of AI Power.” 3 maj, 2026, hämtad 15 maj 2026. URL: https://students.bowdoin.edu/bowdoin-science-journal/csci-tech/between-concentration-and-decentralization-the-changing-landscape-of-ai-power/.

[39] M. Kosinski. “What is black box AI?” 19 okt. 2024, hämtad 15 maj 2026. URL: https://www.ibm.com/think/topics/black-box-ai.