KU LEUVEN

FACULTY OF ENGINEERING SCIENCE

Quantum Computing for Earth Observation

Ciem Cornelissen

Promotor	Prof. Dr. Matthew B. Blaschko (KULeuven)
Supervisor	Dr. Tanja Van Achteren (VITO)

June 8, 2023

Abstract

Because of the promises of the growing and evolving field of quantum computation, research is done on the applicability of near-term quantum computation in the field of earth observation. From a literature study of the current state-of-the-art in AI for earth observation, a conclusion can be made that hybrid classic-quantum neural networks have the potential to improve the convergence speed and accuracy of neural networks. In this master thesis, two use cases have been implemented, a land use classification model and a semantic segmentation model for water detection. For both new hybrid networks, the performance is investigated by looking at the convergence of the network and its accuracy in classifying the image or the pixel. Depending on the shift used for backpropagation the user can choose between a faster convergence of the networks, and a higher eventual accuracy when classifying, than classical versions of the networks.

Nederlandse Samenvatting

Deze thesis gaat over het gebruik van kwantumcomputers en toepassen van concepten van kwantum computing op het gebied van aardobservatie. Omdat kwantumcomputers grote computationele versnellingen beloven kijken veel gebieden in de wetenschap of zij zich al moeten bezighouden met deze nieuwe revolutie. Dus ook het gebied van aardobservatie, om te kijken of kwantumcomputers inderdaad al relevant zijn voor hen. In deze thesis wordt eerste gekeken naar het veld van kwantum computing en wat de predicties zijn voor de komende jaren. Daarna worden experimenten uitgevoerd die al toepasbare concepten toepassen die komen van het veld van kwantumcomputers om te kijken hoe deze concepten bestaande algoritmes beïnvloeden die gebruikt worden in het veld van aardobservatie.

Het tweede hoofdstuk is een review van recente en basisliteratuur. Het hoofdstuk start met een introductie van basisconcepten uit de kwantummechanica, bijv. qubits, entanglement en qubit gates. Hierna worden problemen met de huidige kwantumcomputers aangekaart. Verstoringen in de interacties van de mechanismen om qubits te controleren, interacties van qubits en onzuiverheden in hoe de qubits gemaakt zijn, blijken een groot probleem te zijn voor huidige kwantumcomputers. Daarna worden basis kwantumalgoritmen samengevat, algoritmen zoals het bekende Shor's algoritme en de rol van superpositie en entanglement worden uitgelegd. Ook worden nog andere algoritmen worden besproken en uitgelegd. Verschillende types kwantumcomputers, samen met verschillende types qubits worden uitgelegd. Met een focus op Transmon qubits, deze worden meer in detail uitgelegd. Een visualisatie van de bedrijven die aan kwantum computers werken met de hoeveelheid qubits in hun kwantumcomputers wordt gegeven. Een belangrijk concept dat hoort bij de visualisatie, is het concept van een kwantum error gecorrigeerde qubit. Dit is een qubit die een voldoende lange levenstijd heeft zodat men er voldoende toepassingen op kan doen zonder dat er informatie verloren gaat. De hoeveelheden die in de visualisatie staan zijn allemaal niet error gecorrigeerde, dit is omdat error gecorrigeerde qubits nog niet bestaan. Maar de denkwijze van de meeste bedrijven is om voornamelijk de hoeveelheid qubits in hun kwantumcomputers te vermeerderen

zodat ze later door te werken met een grote hoeveelheid qubits kwantum error correctie kunnen toepassen. Zodat ze met meerdere qubits, 1000+, enkele error gecorrigeerde qubits kunnen maken. Ook wordt er een visualisatie gegeven van de predicties van de bedrijven voor de komende jaren en daar uit blijkt dat voor de komende 10 jaar er nog geen zekerheid is voor werkende kwantumcomputers. Maar bedrijven als IBM en Google willen wel tegen 2030 een vorm van kwantumcomputer hebben met error gecorrigeerde qubits maar de hoeveelheid error gecorrigeerde qubits dat deze bedrijven zullen gaan verwezenlijken moet nog blijken.

Na het gedeelte over kwantumcomputers worden machine learning concepten ingeleid. Na deze inleiding worden de twee velden samengevoegd en wordt er naar papers gekeken die gaan over hybride machine learning netwerken. In zo een netwerk wordt een vorm van kwantum computatie toegepast om te proberen de prestatie van de netwerken te verbeteren. Uiteindelijk worden wat caveats bovengehaald voor kwantumcomputers en wordt er wat specifieker ingegaan op toepassingen voor aardobservatie.

Het derde hoofdstuk gaat over de methodologie van de thesis. Dit hoofdstuk gaat eerst over de gebruikte data in de experimenten, kijkt naar voorbeelden in de data en verschillen tussen verschillende datasets. Daarna worden de kwantum circuits beter bekeken, want deze worden gebruikt om de hybride netwerken te maken die later in de experimenten gebruikt worden. Hoe deze werken wordt uitgelegd en hoe ze de netwerken eventueel kunnen verbeteren word ook aangekaart. Uiteindelijke worden de basisnetwerken geïntroduceerd. Dit zijn het convolutionele neurale netwerk en het U-Net. Het eerste netwerk wordt gebruikt voor het classificeren van afbeeldingen en het tweede netwerk word gebruikt voor het classificeren van pixels in afbeeldingen (segmentatie). Hierna wordt uitgelegd waar in deze netwerken deze nieuwe lagen komen.

Het vierde hoofdstuk gaat over de resultaten van de experimenten. Eerst wordt zowel voor het classificatienetwerk als het segmentatie netwerk een basis gelegd door een klassiek netwerk te trainen. Dit wordt gedaan om te zien welke performantie men kan verwachten van de netwerken, zodat men later kan zien of de kwantum laag de performantie verbeterd of verslecht. Na de gelegde basis worden de hybride netwerken getraind. Voor het classificatie gedeelte bleek dat de kwantum laag een positieve impact had op de performantie. Zo bleek dat de convergentie sneller was mits het gebruik van de hoge shift voor de training. Als men koos voor een kleinere shift bleek dat de uiteindelijke classificatie accuratie verbeterde. Uit dit experiment blijkt dat naargelang men zijn shift klein of groot kiest, voor het trainen van het classificatienetwerk, men kan kiezen tussen een snelle convergentie of een wat trage convergentie maar een uiteindelijk betere accuratie. Ook bleek uit het gebruik van verschillende lagen dat de lagen met entanglement tussen de qubits beter presteerden dan de lagen zonder deze entanglement. Voor het segmentatie gedeelte werd opnieuw deze snellere convergentie waargenomen. Enkel was hier door verandering van de shift de uiteindelijke accuracy niet significant verbeterd.

Acknowledgement

First and foremost I want to thank my daily supervisor Dr. Tanja Van Achteren, for her time during the many meetings, for guiding me this year, and for the countless feedback and creative ideas.

I also want to thank Xenia Ivashkovych and Lisa Landuyt for providing and explaining the data.

Contents

List of Figures

Introduction 1

If life were predictable it would cease to be life, and be without flavor.

— **Eleanor Roosevelt**

Quantum computation promises computational advantages in many areas. One of the most famous examples of a computational speed-up promised by quantum computers is through an algorithm called Shor's algorithm, which promises to change the prime factoring problem from exponential to polynomial time. Because of the big promises many fields of research start to explore how quantum computing could benefit their field. One of these fields is the field of earth observation.

The purpose of this thesis is to learn more about quantum computation and how it could benefit the field of earth observation. This is done by looking at the field of quantum computers and summarizing what the potential is for the foreseeable future, i.e. the next 10 years. After taking a look at quantum computers there is a dive into the field of machine learning, as machine learning is a big part of earth observation. Some recent developments of the merging of the two fields are looked at. In the end, experiments are done to get some experimental proof of if there are already things from the field of quantum computation that are able to be implemented and how these benefit the algorithms used by the field of earth observation. These experiments aim to be further proof of near-future quantum computational advantages and broaden the field of this kind of computation.

This is done by applying quantum circuits inside already existing machine learning techniques, e.g. convolutional neural networks and segmentation networks like the U-Net. The question is then how these quantum layers inside these networks affect their performance in both convergence speed and ability to generalize their classification performance to unseen data.

If these kinds of algorithms perform well this work can be a cornerstone for further research. These layers are easy to use and could be implemented easily in other kinds of networks and with an increase in the abilities of quantum computers, these layers can only become more powerful and efficient. organization of the study

Chapter 2 is a review of related literature and research and starts with a small introduction to quantum qubits and some basics of quantum mechanics. After this, some problems with today's quantum computers, e.g. decoherence and error correction, are discussed. Basic quantum algorithms, and quantum computers are introduced a further explained. An overall view of the field with future predictions is also given. After the quantum mechanical aspect, machine learning gets introduced and some basic networks are introduced. These fields are then merged and recent papers about hybrid machine learning, i.e. a combination of quantum computation and machine learning, are summed up. This chapter then ends with some caveats and some conclusions on the applications to the field of earth observation.

Chapter 3 goes over the methodology of the thesis. The chapter first goes over the datasets used for the experimental part and the difference between them. After this, a closer look at the quantum layer is taken. The quantum circuits and how they work are explained, also some reasoning is given on how these quantum layers could benefit the already existing techniques. Backpropagation is explained as this is different from classical layers. The basic techniques for how the quantum layers are inserted are introduced and explained and also the evolved models with the quantum layers are visualized.

Chapter 4 goes over the results from the experiments and the main conclusions from the main experiments. The thesis then finishes with the last chapter summarizing the main findings from the chapters and giving some possibilities for further research.

Review of Literature and Research

2

Everything has beauty, but not everyone can see.

— **Confucius**

This chapter goes over some of the basics of quantum mechanics and machine learning, the merging of both fields, and recent papers that go over hybrid networks. The power of quantum computers comes from how a quantum computer operates differently from a classical computer. The non-determinism of a qubit makes it that a qubit has a continuum of probabilities of being in either of its two states. This is also called superposition. One can say the qubit is in its two states at the same time. One will only know the state and its probabilities after multiple measurements. Another way quantum computers are different is because of entanglement. Because of entanglement, qubits become related to one another. These two properties make it so that a quantum computer can parallelize operations that on a normal computer would have to be done separately. Machine learning is a field of research that tries to, with special algorithms, make machines learn something about the input data. The term machine learning originates from A.L. Samuel, who in 1959 published a paper on machine learning for the game of checkers [68]. Today, there are many different types of machine learning. Some examples are unsupervised learning [25], supervised learning [74], reinforcement learning [61], semi-supervised learning [84], and deep learning [31].

2.1 Qubit

The qubit was first introduced by Benjamin Schumacher in April 1995 [71], as a fundamental unit of quantum information. In his paper, he also gave a practical realization of a quantum system with two states, i.e. the electron spin. The electron spin has been looked at for quantum computation [77, 16] and is in some cases still used as a qubit, e.g. quantum dots.

A quantum bit is the name for a bit that has quantum mechanical properties. A standard bit is something that can be in either two states, e.g. the transistor, which can close a circuit and let current flow or be open so that no current can flow through the circuit. A qubit has quantum mechanical properties which come with non-determinism, i.e. one does not know in what state the bit is. Because of this non-determinism, the qubit has a probability of being in one of two states. To be able to use this property one has to be able to tune this probability or, after computation, be able to measure the probability. In the latter case, one can only know the state of the qubit by measuring it. For regular bits, this is not a thing. One can always check without looking at the bit in what state the bit is in. The down and up states of a qubit are often written as $|0\rangle$ and $|1\rangle$ respectively. That way, a one-qubit system can be written as a linear combination of these two states:

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

With α and β the square root of the probability of the corresponding state. This is because mathematically, one gets the probabilities of the states of the system by multiplying a $|\rangle$ with a $\langle|$. $|0\rangle$ is the same as $\left(\begin{smallmatrix}1\\0\end{smallmatrix}\right)$ and $|1\rangle = \left(\begin{smallmatrix}0\\1\end{smallmatrix}\right)$. The $|\rangle$ is called a ket and comes from the Dirac notations, and they represent vectors. The reason Dirac notation is often used is that of the exponential growth of vectors that represent multiple qubits states, i.e. a state of n qubits needs a vector of size 2^n but only a $|\rangle$ of size n. For example, a quantum system of only 4 qubits would need state vectors of 16 and $|\rangle$ with 4 bits in it.

A way to visualize a one-qubit system is by using the Bloch sphere (2.1). This is a 3-dimensional sphere with a one-qubit system only being able to have states on the edge of the sphere. The mathematical relation between the Bloch sphere and the quantum system is as follows:

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

Fig. (2.1) is a representation of this one qubit system. One can use this visualization to visualize the effect of certain gates. However, this only works for a one-qubit system. The $|0\rangle$ and $|1\rangle$ states lay on the z-axis and a superposition can be thought of as the system laying in the xy-plain. The θ relates to probabilities of system being in $|0\rangle$ and $|1\rangle$. The ϕ relates to the phase of the qubit.

To change the state of a qubit, one uses operations. Mathematically, these are represented by matrices. Practical realization depends on the type of practical qubit,

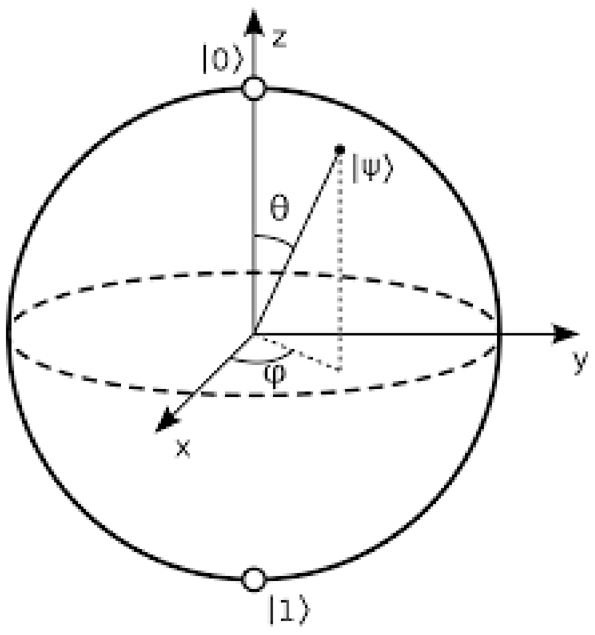


Fig. 2.1: A visualization of the Bloch sphere for a one qubit system.

more on that later. To go from a determined state being $|0\rangle$ or $|1\rangle$ to a superposition of these two states, one can perform a Hadamard gate also written as:

$$H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.3}$$

If H acts on state $|0\rangle$ one gets a $|+\rangle$ state, which is equal to $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ a superposition of both states with equal probability. With the use of the Bloch sphere, one can see this as changing the system from the $|0\rangle$ with 90 degrees such that the system lays in the XY-plain. Another one-qubit operation is the rotation operation or the rotation gates with e.g.:

$$R_y(\theta) = \begin{pmatrix} cos(\frac{\theta}{2}) & -sin(\frac{\theta}{2}) \\ sin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{pmatrix} \tag{2.4}$$

y refers to the axis of the Bloch sphere around which one is rotating. This gate can be used to encode classical data to the quantum world [3], e.g. one can set θ equal to some classical data and then apply this gate to an easy-to-initialize qubit.

There exist practical implementations of qubits. Some of these implementations will be talked about later, to get an idea of how these mathematical concepts are made practical.

2.2 Entanglement

Apart from superposition, another quantum property that makes quantum computing powerful is entanglement. Entanglement is a quantum mechanical relation between two qubits, i.e. the two qubits depend on each other without noticeable communication between the qubits. One can mathematically entangle two qubits through two qubits gates. The controlled-NOT gate(CNOT-gate) is often used:

$$U = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.5}$$

Applying this gate to a two-qubit system makes the second qubit depend on the first one. How this is implemented practically depends on the practical qubit and will be mentioned later in the thesis.

Combining the Hadamard gate with the CNOT gate, one can get the bell states. The circuit implementing these two gates working on the $|00\rangle$ can be seen in Fig. (2.2). The blue symbol that connects the two lines is the CNOT gate. The last symbol on each line stands for doing a measurement on the qubit. The symbol in the figure shows that this is done with respect to the z-axis of the Bloch sphere. These bell states are maximally entangled, i.e. if one measures the state of one qubit and knows the initial state of the system, one knows what state the other qubit is in. The bell states are given by:

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{2.6}$$

$$|\Phi^-\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}} \tag{2.7}$$

$$|\Psi^+\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}} \tag{2.8}$$

$$|\Psi^-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}} \tag{2.9}$$

The circuit in Fig. (2.2) gives an outcome (2.6). The other ones can be created by changing the initial values of the qubits.

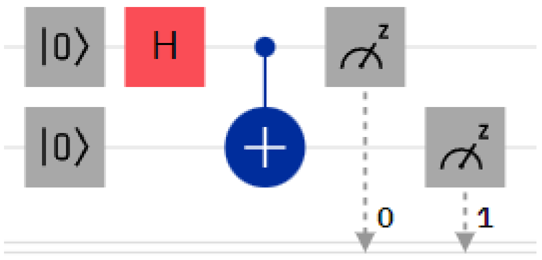


Fig. 2.2: Quantum circuit applying the Hadamard gate and CNOT gate.

2.3 Decoherence and error correction

Superposition and entanglement are fundamental properties at the atom level and are thus very sensitive to disturbances. Qubits on which one operates are not fully disconnected from their environment. One always has a connection to the operand but also a connection to the rest of the environment. Also, defects in how a qubit is made can be a source of noise [58].

This coupling to the sources of noise results in a loss of information from the system. This is one of the major problems with quantum computers at the moment. This results in that not all qubits having the same performance, and also the induced error because operations can be different from operation to operation. Some qubits hold their information better and thus have longer lifetimes. Which is important for longer quantum computations. One can increase the lifetime of information in their quantum system by using error correction. But for this one needs more qubits to hold the same amount of information. The worse the decoherence, the more qubits one needs to hold the same amount of information. The amount of qubits and quality of the qubits is thus both important.

To sustain computation, one has to be able to correct errors faster than they accumulate. This point is the threshold from which error-correcting can contain the information in the system. This threshold depends on the algorithm used for the error-correcting. Error-correcting algorithms are optimized for the information that can be held with the least amount of qubits while taking into account the possible architectures for quantum chips that are in quantum computers. Error correction works by sharing the information normally stored on one bit or qubits to multiple qubits. Even though one cannot just clone qubits to copy information. It is possible to spread the information over a highly entangled network of qubits. Through multi-qubit gates, one is then able to catch the error and correct it by applying another gate. An example of a quantum error correcting code is the algorithm devised by R. Raussendorf, J. Harrington, and K. Goyal [64]. This algorithm makes use of a 2D nearest neighbor array of qubits. Their calculated threshold for all sources of error (gates operations, storage, measurements) is 0.75%. This threshold has recently been achieved by X. Xue et al. [82]. Assuming an error probability on operations of $10^{-3}\%$, so below the threshold, one would need 10000 physical qubits [14] to have one good error-corrected qubit. However if one can decrease the error probability during operations one needs fewer qubits than this.

These errors induced by both noise in the system and operations is one of the main reason that current quantum computers do not yet outperform classical ones. As will be later shown, many companies have already existing quantum computers, with some having hundreds of qubits. However, because of the loss of information, these qubits can not be used for long computations. To goal to combat this is to scale the number of qubits such that one has enough qubits to use these error-correcting codes mentioned here to be able to make small amounts of error-corrected qubits.

2.4 Quantum volume and CLOPS

Because of the many types of ways a qubit can be made and even within a certain type of building materials, operational procedures can affect the quality of the qubits. Because the lifetime of qubits and the quality of operation can change the quality of computations so much, it's necessary to not only look at a quantum computer by the number of qubits it uses.

A better measure to assess the quality of computations is by looking at a quantum computer's quantum volume [8]. It not only takes the number of qubits into account but also the depth of the circuits that can be run, i.e. the number of operations that can be done before the errors get so large that the system essentially behaves classically. Quantum volume depends on the minimum of both depth and amount of qubits. This is necessary because otherwise, one could achieve a high quantum volume with, e.g. a two qubits quantum system that could be simulated.

Even though it's better to look at a combination of both qubit count and depth of available operation, a universal measure used by everyone is still emerging, and not all companies publicly disclose their quantum volume or different metrics of the same kind.

Another important way of comparing quantum computers is by the speeds of operations it is capable of. In 2021, such a measure was introduced by IBM [80]. Their measure is called CLOPS(Circuit Layer Operations Per Second) and it tells something about how many quantum circuits can be run on the computer. To make the measure more universal, it uses multiple kinds of circuits to be run sequentially and also takes into account parameter updating times:

$$CLOPS = \frac{M \times K \times S \times D}{timetaken} \tag{2.10}$$

With M and K, the number of templates for quantum circuits and the number of parameter updates, respectively. A quantum circuit template is a quantum circuit from a set of quantum circuits chosen to be used for measurements when one wants to calculate the CLOPS. S is the number of shots to run the circuit. This is taken into account to give weight to the execution time. D is the number of layers, i.e. the number of permutations followed by paired 2-qubit gates.

The best way of comparing quantum computers takes into account both measures discussed in this section, as one then takes into account quality and speed. However, with a high number of qubit quantum computers just emerging, not many companies use both such measures.

2.5 Quantum algorithms

A way of understanding how a quantum computer speeds up calculations is by looking at algorithms that promise a speed-up over the classical versions, by using superposition and entanglement between qubits. Even though these algorithms can not be run yet because qubits lose their information too fast to do interesting calculations. Many algorithms have been thought of that if, one day, quantum computers will work, these algorithms will give an advantage over running them on classical computers.

An example of an algorithm that uses both the superposition part and entanglement part of a quantum computer to achieve faster computation is Shor's algorithm [73] for finding the prime factorization of numbers. It speeds up the need for exponential time to only polynomial time. The algorithm uses a quantum version of fast Fourier transform(QFT) [17] to find p of the equation:

$$g^p = mN + 1 \tag{2.11}$$

With g a random guess, N is the number one wants to factor, and m is the number such that 1 is the residual. Shor's algorithm works by making a superposition of possible exponents. Taking this superposition through two processes, one that applies the exponent to the guess and one that calculates the residual. One can find p by applying QFT to the remaining superposition that one gets after doing a measurement on the residual part of the superposition. The superposition then partly collapses such that only the exponents with this residual stay in the superposition. For this algorithm, superposition helps because one parallelizes calculations that on a classical computer would have to be done sequentially. Because of the periodic properties of p in this remaining superposition, QFT can find this period. By using a found p one can get a better guess that has around 37.8% chance of being a correct guess. This probability has to do with some problems that can occur with p, e.g. if p is odd, then p divided by two is not an integer. If p gives a wrong result, one keeps on sampling p's till a correct p is found. It has also been proven that entanglement is necessary for this algorithm by R. Jozsa, N. Linden [45]. Entanglement is necessary for two parts of this algorithm. First, one needs entanglement for the preparations of the input to the QFT algorithm. One can see these two registers getting entangled as the guess that gets entangled with the output $g^p(mod)N$. QFT requires entanglement along the outputs, as it requires the use of controlled rotational gates. These gates are applied so that during this step, the output gets fully entangled to find p.

Another example of an algorithm promising a speed-up is a search algorithm invented by L.K.Grover [29], promising the search speed to go from $\mathcal{O}(\frac{n}{2})$ to $\mathcal{O}(\sqrt{n})$ with n the size of the database. The algorithm works by increasing the probability of observing the item one searches for. This is done by flipping the phase of the wanted item in the database. After one inverts the amplitudes of the qubits around the mean. This decreases the probability of other items and increases the probability of the correct item. For this algorithm, it has also been proven that entanglement is necessary for operating [11].

Other algorithms that might benefit are algorithms that can make use of these algorithms and have problem sizes big enough such that quantum computer preparation, and the delay caused by communication between the quantum computer and the classical computer is not the limiting factor of the algorithm's speed. Or algorithms that can apply some of the parts of the algorithms. There are also other algorithms that promise a speed-up. These will be mentioned later in the machine learning section.

2.6 Types of quantum computers

There are multiple kinds of quantum computers. The most used and best understood one is the gate model. Which acts just like a classical computer in the sense that it acts with gates on the input to get an output. It is said that the gate model quantum computer is a universal quantum computer, i.e. it can simulate a quantum Turing machine. A gate model quantum computer works by initializing the input to the qubit and then working with a gate to locally edit the qubits. After the computation is done, the qubits get measured to get the answer to the computation. Such a quantum computer can do all computations a classical computer can. But because of the difficulty of making a quantum computer work, simple calculations should be left to classical computers. It is only for special cases, e.g. the two algorithms that have been discussed, a quantum computer would show a better performance than a classical computer.

Another type is the adiabatic quantum computer [26]. It has been proven that, just like the gate model quantum computers, adiabatic quantum computers are also universal [4]. An adiabatic quantum computer works by minimizing the energy of the system. Over time, it evolves from an easy-to-initialize system to the assignment, to find the minimal energy of the final system, i.e. the answer to the problem. A practical implementation of this is quantum annealing. This implementation also uses the minimal energy state of a system to find the answer to a problem. However, this practical implementation is not a universal quantum computer. This is because, as a practical implementation, it operates at a finite temperature and is not closed off from its surroundings. Even though this practical implementation is not a universal quantum computer, it has some practical use cases. But more on this later.

Measurement-based quantum computers [12], also called one-way quantum computers, is also a universal quantum computer scheme [79]. This scheme works by first initializing the system to a wanted state and doing the measurement on the qubits one by one. The order and basis in which the qubits are measured specifies the algorithm to be run on the computer.

A more theoretical scheme is the topological quantum computer [48]. It is more theoretical because of the qubits that it uses. The qubits are particles named non-Abelian anyons, e.g. Majorana zero modes. This scheme is also proven to be universal [51]. These quantum computers should be more stable because of the separation in how the qubits are made, that is if these particles can be found and used.

2.7 Physical qubits

There are many physical ways to make a qubit. In Fig. (2.3) one can see 8 different realizations of the qubit. The figure shows the companies working on the qubit type. The numbers next to the company names are the already practical forms of quantum computers made by that company. Companies without a number are researching the qubit type or have not yet publicly disclosed the number of qubits. The figure is not exhaustive, but it shows a general overview and the focus of the field on certain types. One of the most researched and used variants is the superconducting qubit and, from all superconducting qubit types, the transmon qubit [50] is the most used. Because this is one of the most used and promising qubits, this thesis will go into more detail on the transmon qubit.

In Fig. (2.3) one can see that DWAVE has a different color. This is because this is not a universal quantum computer scheme. The quantum computer of DWAVE is a quantum annealing quantum computer [43] using superconducting flux qubits. Because one does not have to control the qubits with gates, it is easier to make a quantum computer with a higher amount, but with the drawback that not all computations can be performed on the computer. This computer is, however, still good at combinatorial optimization problems [85]. The problem gets translated to a logical interaction graph, which is again translated to a graph suited for the physical device. The problem is then further programmed by tuning the bias of the qubits. This is done by controlling the magnetic field acting on the qubit. Then the spins are initiated into an easy-to-initiate lowest energy configuration. After this, the annealing starts, and now the system is guided from the initiated state to a problem state. After the annealing, the spins of the individual qubits are read out and stored and the process is done again. The process has to be done multiple times because the system is not promised to be in a true ground state at the end of the annealing.

Photonic quantum computers use photons as qubit type [78]. For example, the quantum computers of Xanadu [57] use a combination of light squeezers to reduce quantum uncertainty and create the qubit, followed by an interferometer to apply the quantum circuit. The polarisations of the photon are used as the state of the qubit. In the end, the photons get counted by photon detectors. Trapped ion quantum computers [13] use charged atoms as qubit type. They get their name from trapping single atoms close to each other. The atoms are trapped by an oscillating electron field coming from the trapping arms. Gates can be applied with lasers, or an interacting electrode, and the qubits can be read out by detecting the emitted photons from the charged atoms. Color center qubits [69] can be made

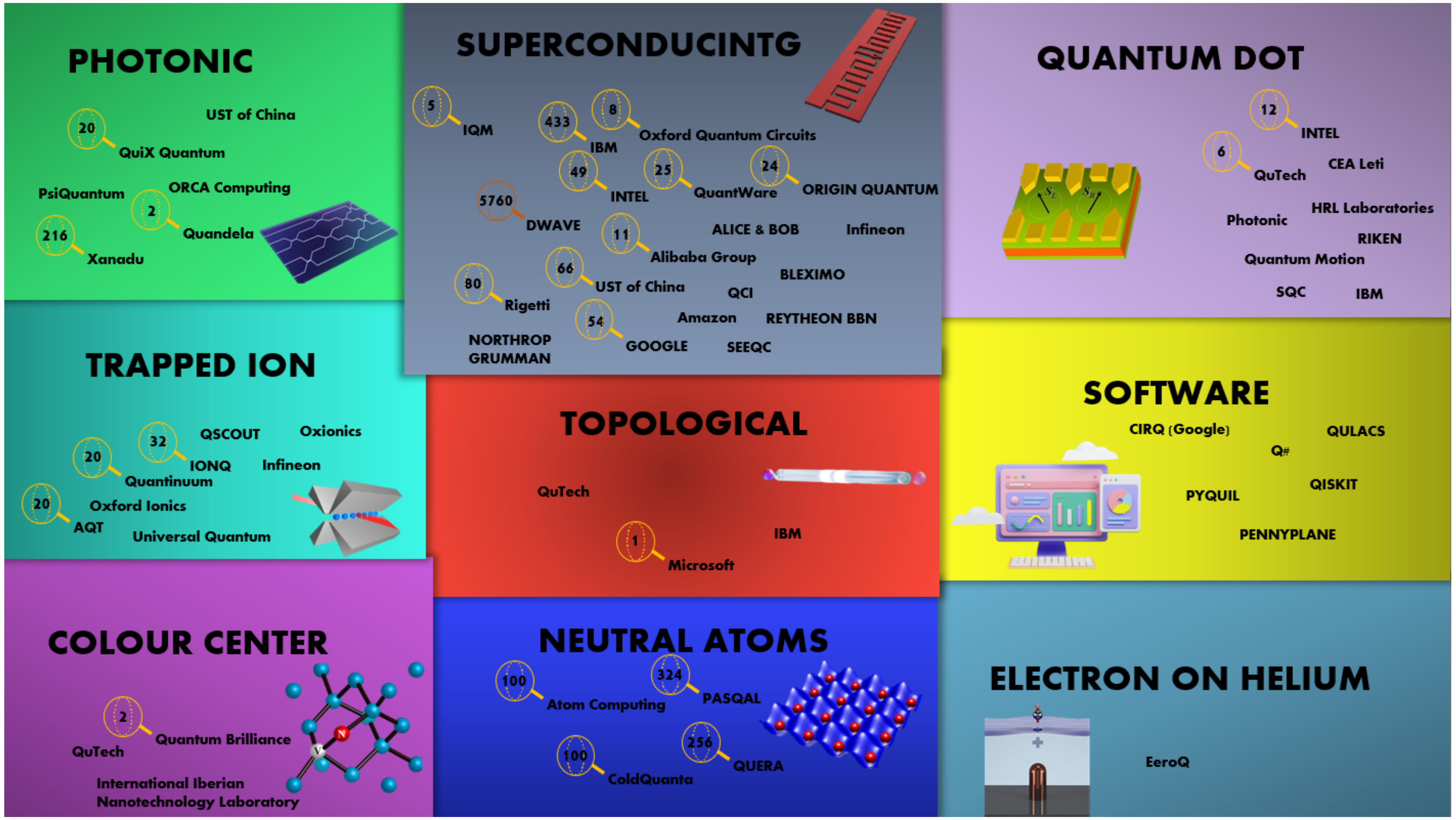


Fig. 2.3: Companies with Physically realized Quantum computers and their qubits. Companies without a number are researching the qubit type or have not disclosed the working amount of qubits.

with nitrogen vacancies in diamonds. The nitrogen supplies an extra electron to the structure of the diamond, which can be worked with. An advantage is that the quantum state can survive at room temperature. However, due to limitations and technical reasons, quantum computers of multiple qubits are relatively hard to make. Neutral atom quantum computers [36] use neutral atoms as qubits, which can be used for simulating materials and interaction between the atoms and also for gate-based calculations. The neutral atoms are cooled, trapped, and controlled using lasers. Topological quantum computers [48] are still the most theoretical quantum computers. Now and then some people claim they have found the qubits needed to build these quantum computers, i.e. Majorana zero modes. However, after reanalyzing the data, it was shown that this was incorrect. In 2022, Microsoft claimed for the second time they found the qubit, but the paper still is going through the peer review process. If they have indeed found this elusive physical phenomenon, it would be the first step to a topological quantum computer. Eeroq is one of the few companies trying to build a quantum computer using electron helium [19]. This type of qubit is still in the early stages of research and development, however, experiments have shown some progress [52, 41]. Intel is one of the front runners of the quantum dot qubits. They use a quantum dot qubit [33] type called silicon spin qubits. The spin states of an isolated electron are the qubit states of the quantum computer. Entanglement and interactions on the electrons can be handled by physical gates which are placed on the silicon substrate.

Fig. (2.3) also shows some software frameworks that are made to program quantum computers or to run quantum simulations. For example, Cirq, which comes from Google, is a Python software library to simulate quantum circuits. It is also able to run the circuits on quantum computers. Q# is an open-source programming language, from Microsoft, to write and run quantum algorithms. Qiskit is an optimization package that can be used to work with the IBM quantum and also the IonQ quantum computer. One that will become more important later on is the QULACS software, which is optimized for fast simulations. Especially for smaller quantum circuits, this software can do certain simulations an order of magnitude faster than, for example, Qiskit.

2.7.1 Transmon qubit

The transmon qubit [50] is one of the most popular superconducting qubits [40]. It is a charge-type superconducting qubit [62], i.e. the state is determined by the absence or presence of Cooper pairs. An example of a cooper pair is a pair of electrons that are bound together and have because of this binding special properties. Other superconducting types are the phase [9] and flux qubits [24].

The transmon qubit is derived from an electronic circuit with parallel a capacitor and an inductor. This type of circuit is called an LC oscillator. The current through the circuits oscillates back and forth. The dynamics of the circuit can be mapped to a harmonic oscillator [22]. The quantized energy spectrum of the harmonic oscillator has periodical energy levels. This is, however, not good for containing the dynamics of the system to just two levels. Therefore, an inductor, a Josephson junction [44] is used. This introduces non-linear behavior. This makes it so the energy levels are not spaced evenly anymore and the dynamics of the system can be contained to two levels. The transmon qubit uses two Josephson junctions in parallel. By changing the magnetic flux through the parallel Josephson junctions, the energy levels can be fine-tuned for better separation, making the qubit less sensitive to noise. A picture of a physical example of a 4 transmon qubit processor can be seen in Fig. (2.4)

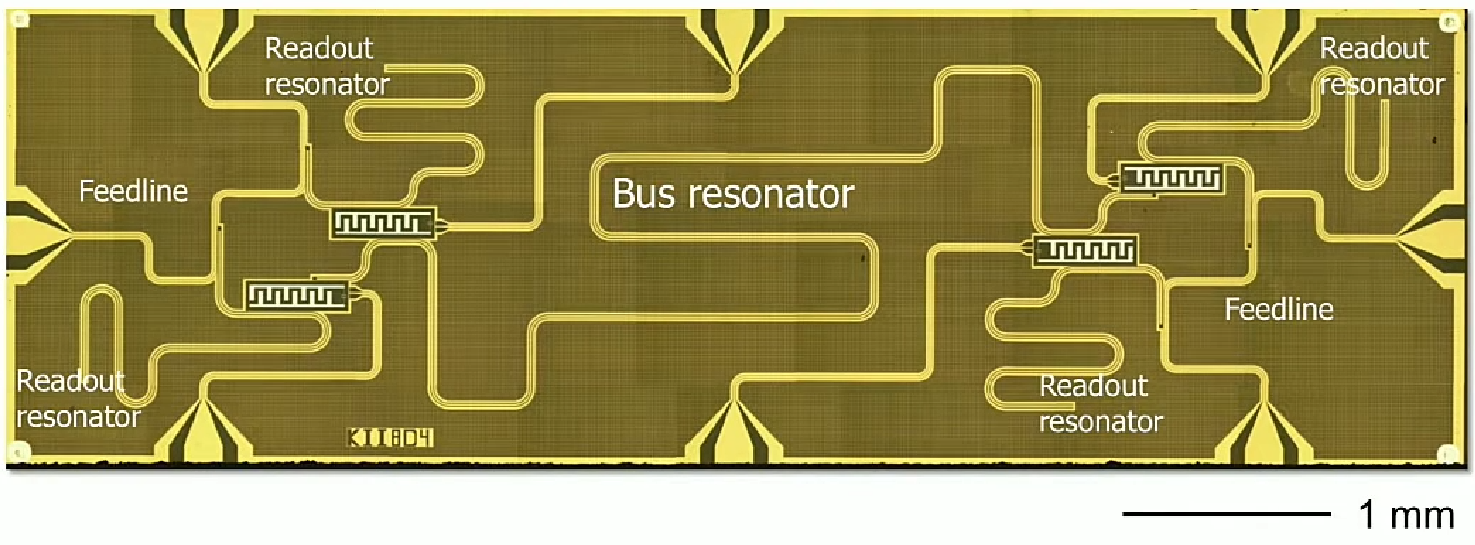
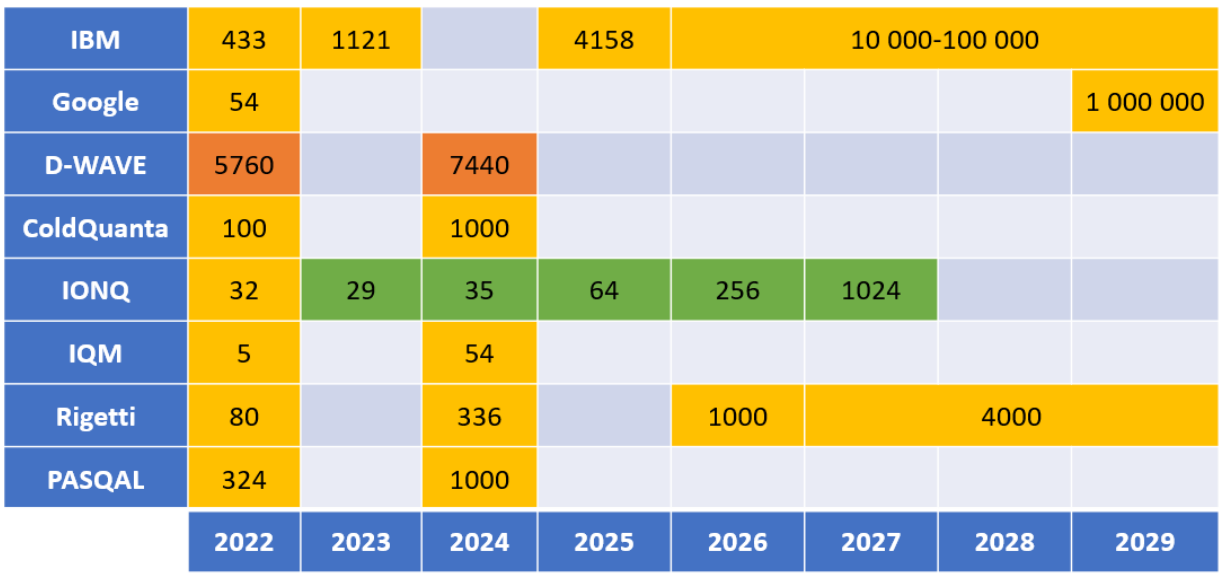


Fig. 2.4: Processor with 4 transmon qubits on it, Each transmon qubit has its own transmission-line resonator and one can also see the transmission line for entangling qubits. [67]

The readout and single qubit gates on a transmon qubit happen through a transmission-line resonator that is coupled to the circuit. The resonating frequency of the transmission line depends on its length and, because of the vacuum Rabi splitting, also on the state of the qubit. By measuring the frequency of the resonator, one can thus measure the state of the qubit. Single qubit gates happen by applying an oscillating electric field to the circuit. The amplitude of the electric field determines the speed of the transition and phase of the qubit. Thus, to change the qubit state, one sends a pulse, i.e. an oscillating electric field with a certain period and amplitude, through the qubit.

The entangling of two separate transmon qubits happens through a transmission line just like the readout resonator, but now the resonator couples to two qubits. To make the two qubits interact, the resonating frequencies must be changed. This happens as mentioned before through the magnetic flux through the parallel Josephson junctions. The time the two qubits are coupled through the change in their frequency determines the amount, e.g. the state of seconds qubits depends on the first qubit. Making the qubits resonant with one another for just long enough thus entangles the two qubits.

2.7.2 Future predictions



	2022	2023	2024	2025	2026	2027	2028	2029
IBM	433	1121		4158	10 000-100 000			
Google	54							1 000 000
D-WAVE	5760		7440					
ColdQuanta	100		1000					
IONQ	32	29	35	64	256	1024		
IQM	5		54					
Rigetti	80		336		1000	4000		
PASQAL	324		1000					

Fig. 2.5: Companies and their predictions for the future amount of qubits of their future quantum computers. With the green cells having some form of error-correcting and the orange being quantum annealers.

The quantum computational field is an active and evolving field. It is not easy to predict the future, but some companies have a road map of what they want to achieve in the future. Some more near term like IBM and some more long term like Google.

Fig. (2.5) shows some companies that can produce quantum computers with their qubit count close to the company name. The horizontal axis depicts time, which ranges from 2022 to 2029. One of the companies that is the clearest and most open about their future is IBM [1]. At the moment of the release of this thesis, the quantum chips in their quantum computer can have up to 433 qubits. As qubit, IBM uses the transmon qubit. If they stay on their trajectory as they have done for the past 3 years, they will double their qubits in the coming 4 years. This will be done by making the number of qubits on one chip larger, but also by coupling multiple chips together. Coupling on short ranges, i.e. chips next to each other, and coupling on longer ranges, i.e. between chips in different cryogenic chambers. Through these 3 innovations, IBM plans to build quantum computers with 10000+ qubit quantum computers. This is necessary to build fault-tolerant quantum computers because, depending on the error rate of the operations on the qubits, thousands of qubits are needed to make one fault-tolerant qubit. Some companies are less clear about how they will move closer to a good working quantum computer. Google wants to build a fault-tolerant quantum computer before the end of the decade but is unclear on how they will do that. The predictions for IONQ [2] are green-colored because they define their qubits as the effective number of qubits for typical algorithms. They also predict that they will have error-corrected qubits in 2025. With a ratio of 16 qubits for 1 error-corrected qubit. D-WAVE [43] also has a different color as they use a different kind of quantum computer as previously mentioned and thus can not be directly compared to the other predictions.

2.8 Quantum Simulation

Current quantum computers are only able to run with a couple of non-error-corrected qubits, which can only achieve a shallow depth. Limited by the number of qubits and possible depth of circuits, fully Error-corrected quantum computers with unlimited depth are still a thing of the future. Today, one can simulate the circuits that will be able to be run on quantum computers. Because these are simulated, one can not expect the speed-ups promised by quantum computers, however, one can get a glimpse of what the output of the quantum computers will look like. Because these are simulated, only a certain depth(amount of operations) and width (amount of parallel qubits) can be achieved. This section goes over how these simulations work. Multiple software packages give access to these simulations, e.g. qiskit [5], cirq [21], qulacs [76], but there will be a focus on the latter. Apart from playing with quantum circuits and seeing how they work, quantum circuits also have other purposes which make them necessary. The simulators can be used to get the ideal results expected from real quantum computers if the width and depth are small enough. Some things cannot be measured but can be made available with simulation, e.g. the probabilities of having certain qubit sequence outcomes. To see whether quantum error correction works, it is useful to compare them to simulations.

There are multiple ways to simulate the quantum circuit, but there will be a focus on the so-called Schrodinger's method [7], used by qulacs [76]. Other methods can also be used for simulating a quantum, e.g. the Feynman path method [10]. This method works by looking at the contribution of Feynman paths. This method uses more time but less space than Schrodinger's method. This is because with this method every gate that connects qubits a decision point is created at which a new branch of possible outcomes is created. All these branches or paths give their contribution to the output quantum state. For larger-sized quantum circuits, even a combination of methods can be used. For Schrodinger's method [7], the simulations work by updating the quantum state of the qubits by applying quantum gates. The initial state of the quantum circuit can be represented as a vector. On this vector, one can then apply the quantum gates by applying the matrix that represents this gate. After applying all the necessary gates, one can do measurements by sampling from the probability distribution. This probability distribution is calculated from the amplitude of the vector after applying all the gates. This is because the square of the amplitude is the probability of finding a qubit in its 0 or 1 state.

Qulacs has some optimizations to run the software faster. This is especially necessary when one wants to run many quantum circuits, e.g. when training a network that has a quantum circuit built into it. Qulacs supports multi-threading and SIMD, which makes it so that the same operation can be applied to multiple data points in parallel. Because of these two optimizations, the time processing arithmetic operations is smaller than memory operations. To balance this and to take more advantage of fast arithmetic operations, circuits are optimized such that memory operations are reduced. Qulacs also enables GPU-accelerated execution, however, this is only available on Linux and not on Windows. This is slower for smaller circuits, but gives a constant speed up for larger circuits. With the crossing point for circuits of around 14 qubits. All these things make this software one of the fastest for simulating quantum circuits. In Fig. (2.6) one can see the execution time for executing random quantum circuits. One can see that for all amounts of qubits, the Qulacs simulations outperform everybody else. Especially for the smaller quantum circuits. From the experimental part of the thesis during the training of the CNN, it became clear that the qulacs packages are around 20 times faster than the Qiskit packages.

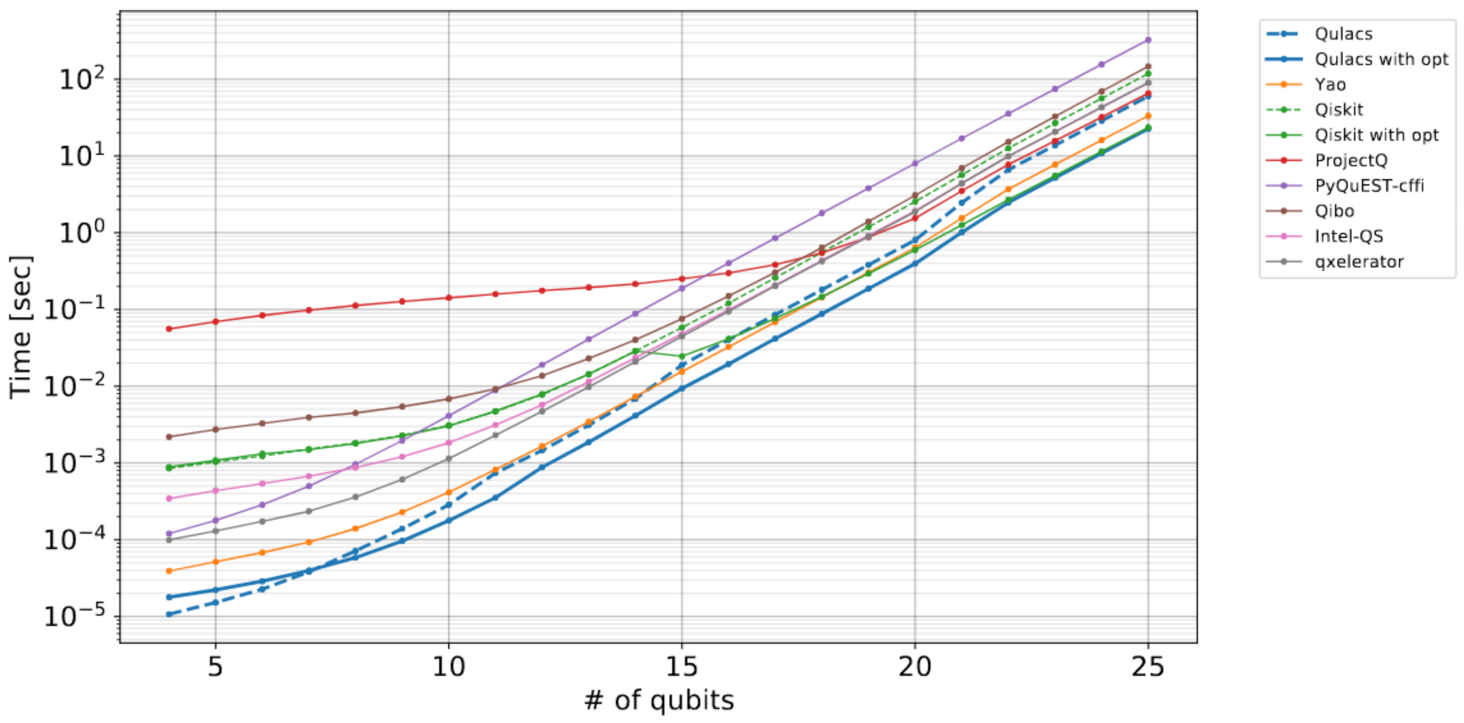


Fig. 2.6: "Times for simulating random quantum circuits with a single thread using several libraries." [76]

2.9 Machine learning

There are many types of machine learning that can be further divided into distinct domains and different algorithms to solve problems. Examples of problems that can be solved by unsupervised machine learning are clustering, dimension reduction, and seriation. Each of these problems has multiple models that can solve it. Each model has their strong and weak points. For these kinds of problems, something must be optimized as without labeled data, it can not learn from examples.

Learning with labeled data is called supervised learning, and examples of problems that are solved with this are classification and regression. Both problems can be trained on labeled data with different models. A combination of labeled and unlabeled data can also be used. This is then called semi-supervised learning. Both types can also be solved by models which have a certain depth to them. Models with large depths fall also under the type of deep learning. The depth of these models refers to the number of layers a model is made of.

Reinforcement learning is a type of machine learning that works with a function that has to be maximized, i.e. the reward function. The agent, i.e. the algorithm that acts on an environment, uses this function to know if the agent gets a positive or negative reward because of an action the agent has chosen. The goal of the agent is to maximize its rewards before finishing a task.

2.9.1 Neural network

The idea of using a computational model to model a neural network originates from Warren S. McCulloch and Walter Pitts [59]. A neural network can be of different types with different purposes, e.g. an auto-encoder [37] is a form of unsupervised learning that reduces the dimension and size of the input and can also decode it back. Long short-term memory(LSTM) [38] is a recurrent neural network that can be used, for example, for language translation. Its name comes from the way the algorithm is designed, i.e. the nodes that pass along information can store it in an information stream that retains information for longer periods and also gives its information to the next node in line. This comes in helpful for translation as the context of what is said can be important for translation in the future.

An artificial neural network works by taking an input and feeding it through a network of artificial neurons with, in the end, an output to say something about the input. An artificial neuron works by taking multiple inputs, adding a bias, and pulling all of this through a function:

$$y = \sigma(w_0 + \sum_i w_i h_i) \tag{2.12}$$

The multiple inputs are multiplied by their weight w_i, with w_0 being the bias. To make the NN perform well, non-linear functions should be used. An example of such a function that can be used is the sigmoid function:

$$S(x) = \frac{1}{1 + e^{-x}} \tag{2.13}$$

This function is S-shaped and has an output range]0,1[. The output of this node can be used as the output of the network or as input to the next node. The higher the weight associated with an input, the more important the value next to the weight, thus the better that nodes listen to the node supplying that input. Nodes can be put into layers that are connected to the previous and the next layer to form an artificial neural network.

To make the neural network learn, one has to optimize the weight and biases of the network to get the desired output. The being right or wrong happens through the loss function, which takes as input the outcome of the neural network and gives back if the output is good or bad. This can be, for example, for an auto-encoder, the difference between input and output. This can also be supervised with inputs that have to be classified to a certain label. During training, the labels have to be added to the inputs. An example of a loss function is the Mean Squared Error(MSE):

$$MSE = \frac{1}{M}\sum_{i}^{M}(y_i - Y_i)^2 \tag{2.14}$$

With Y the ground truth. The optimization of the network happens through back-propagation. This uses the error output given by the loss function. One needs to change the weights and biases in such a way the loss function gets minimized. By calculating the derivative of the loss function regarding the weight or the biases. The network can be optimized such that the cost or the sum of all losses over the training data can be minimized. An often-used optimizer is adaptive moment estimation (ADAM) [47], it is efficient and uses little memory. It is a gradient optimizer that bases its learning rates on estimations of the first and second-order moments of the gradient.

2.9.2 Convolutional Neural Network

Convolutional neural networks (CNNs) were first used by Y. LeChun et al. [53]. CNNs differ from NNs by the way they handle the input to the network. Some layers of CNNs are convolutions of the layer input with different filters. An example of such a convolution can be seen in Fig. (2.7). In this figure, one can see an example of a filter that extracts vertical features from the input.

A filter is a matrix with trainable values. The values of the filters are shared over the input, i.e. the values of individual filters do not change when moving over the input. The matrix moves over the input and gives back the sum of the multiplication of the values of the matrix with the values of the input as the output of the layer. This is done for many filters and because of this, the convolutional layer gives back an output with an extra dimension based on the number of filters. Each filter looks for features in the input, e.g. horizontal and vertical lines in an image. These convolutional layers are often used together with pulling layers. These layers move

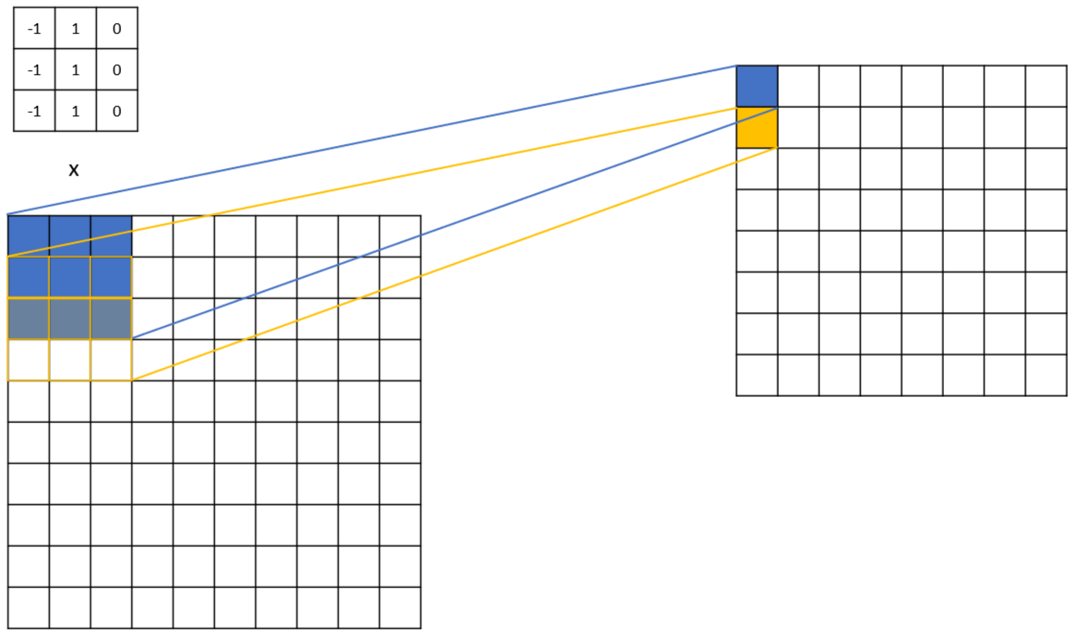


Fig. 2.7: Example of convolution with a filter that extracts vertical lines out of the input.

just like the convolutional layer with matrices over the input, but now the step size of the matrices is the size of the matrices, instead of 1 that is depicted in Fig. (2.7), and the output of matrix is also different. Now the matrix reduces the size of the input. This can be done by taking the average of the input and giving that as output, or taking the max values in the matrix and giving that as output. The combination of the convolutional layer together with the pooling layer is used to reduce the size of the input. Instead of one having a large vector of values, one now has a vector of present features in the input that, because of the local correlation in the input, are detected through the used filters. After the convolutional and pooling layers, the network has normal node layers that try to predict the outcome depending on the features present in the input.

A CNN has multiple parameters to be chosen by the user. For example, padding can be used to keep the same size of the output and input and make edge inputs used as frequently as center inputs. Padding is done by adding edge inputs of 0, the amount of padding is chosen by the user. The stride size is also a parameter and says how many inputs the matrices jump over between evaluations, e.g. in Fig. (2.7) the stride size is 1. The amount of filters a convolutional layer has is also a parameter to be chosen.

2.10 Hybrid machine learning

Because of the limitations of quantum computers at the moment, only small quantum circuits can be used. Because the size of these smaller quantum circuits is not big enough to run full-fledged machine learning schemes like quantum neural networks or quantum variational auto-encoders, hybrid machine learning schemes are used. In hybrid machine learning schemes, some part of the previous full classical algorithm is made quantum. This can be done in several ways.

A quantum variational auto-encoder(QVAE) can be realized for the generative part of the scheme. Amir Khoshaman et al. [46] used a Quantum Boltzmann Machine (QBM) for the latent generative process of the VAE. The sampling from the QBM can be done with both a gate-based QC or an annealing QC [6], which is positive as an annealing QC can be more easily scaled. The QBM implements the prior distribution in the latent space, this makes it so that the gradient for training can be estimated by sampling from the QBM. This QVAE gives state-of-the-art results on the MNIST dataset [20].

CNN can also be made quantum, as done so by M. Henderson et al. [35], by making the filtering in the convolution layers quantum. These new layers are called quanvolutional layers. The difference with normal convolutional layers is that multiplication with a matrix is changed into using the input for a quantum circuit. The initial states of the quantum circuit are determined by a threshold on the input values, i.e. inputs lower than this threshold result in a qubit in state 0, and inputs with a value higher than this threshold result in a qubit in state 1. After the initialization, a random quantum circuit is applied. In the end, the states of the qubits are read. The quantum circuit gets sampled a given amount of times, then the most likely output vector is taken. From this vector, the qubits in state 1 are counted and returned as the output of the quantum circuit. This scheme, however, found no improvement in either accuracy or convergence speed over the classical algorithm yet. The performance is on par with the classical version.

NNs and CNNs can be made hybrid by exchanging one of the normal layers of the network for a quantum layer. A. Sebastianelli et al. [72] made a hybrid CNN network by changing one of the latter layers of a CNN with a quantum circuit. This quantum circuit encodes the data from the previous classical layer to the quantum world by rotating operations. The angle of rotation is the input from the previous layer before the quantum circuit. After the encoding, the qubits get entangled. After this entangling, another layer of inputs can be encoded or the qubits can be read out. The readouts from the quantum layer get then passed on to the next layer.

The gradient for backpropagation before the quantum layer gets calculated through small shifts in the input of the quantum layer. The change between the outcomes from the shifts gets passed back as a gradient for backpropagation. A. Sebastianelli et al. found that the network with the quantum layer performed better than the classical CNN not only in accuracy but also in convergence speed to an optimal set of weights.

2.11 Quantum machine learning

In the future, when quantum computers become error corrected and can run larger algorithms, there will be an effect on the field of machine learning. There are some machine learning procedures that, when run on quantum computers, have a reduction in the operation needed to run the algorithm. This section will look better at which algorithm will improve and by how much.

2.11.1 PCA

Principal Component Analyses (PCA) is a way to look at a dataset with a large number of dimensions. By calculating the covariance matrix of the data and its eigenvalues, one can visualize the data with the most important components, i.e. the new dimensions with the highest amount of variance in the data. Quantum PCA [56] works by translating normal data into a quantum state through quantum random access memory (QRAM) [28]. By sampling from these vectors, one can analyze the properties of the unknown density matrix through density matrix exponentiation [55]. Then, with the help of a quantum phase estimation algorithm [49] that can calculate the eigenvectors and eigenvalues of matrices, one can get the eigenvectors and eigenvalues of the density matrix, which is the same as the covariance matrix up to a constant factor. This quantum version of the algorithm reduces the complexity from $\mathcal{O}(d^2)$ to $\mathcal{O}(Rlog(d))$, with d the dimensionality of the data and $R << d$ the dimension of a subspace used for the projection of the encoded quantum state.

2.11.2 SVM

Support Vector Machine(SVM) is a supervised machine learning technique that finds a hyperplane to separate labeled vectors with maximum margins. Through the use of kernels, this boundary can be found in a higher dimensional space than the dimension of the original data. This is done through the use of kernel functions, which is a function that takes as input the vector of the data and can transform it into a space where the data could be more easily separable. This boundary can then be used to classify new vectors. Quantum SVM [65] works because of the increase in the speed of the quantum evaluation of inner products. Another key take is the use of least-squares SVM [75] that has a classical complexity of $\mathcal{O}(M^3)$ with M the number of training samples. QSVM gets an advantage through exponentiation of the least squares approximation and doing a phase estimation. This results in the optimal Karush-Kuhn-Tucker multipliers that minimize the Lagrangian of the problem. The complexity of QSVM is $\mathcal{O}(k_{eff}^3 c^{-3} log(MN))$ [65] with N the dimension of the data. c is the accuracy of the algorithm and the k_{eff} is the number of repetitions to a high success probability.

2.11.3 Quantum deep learning

The word deep in deep learning refers to the depth of the architecture or algorithm used. The depth of an architecture points to the number of layers or amount of abstraction the algorithm can do. Through these higher forms of abstraction, the architecture can combine abstraction such that real-world things can be put together, e.g. the edge features in CNNs are put together to form boxes, which again be put together with wheels to recognize things like bikes or cars. For quantum deep learning, the training time could be reduced through better gradient estimations. The Gradient Estimation Quantum Amplitude Estimation (GEQAE) [81] reduces the variance of the estimated gradient through the use of the training data in superposition form instead of getting it sequentially. Because of the better gradient estimations, the number of times the training data needs to be taken through the network will be smaller and, thus, if this can be done efficiently, the training will go faster. However, this algorithm would also need more qubits to work because the data must be stored on the quantum computer.

2.12 Caveats

Even though quantum computers have come a long way, from purely theoretical to many companies building their first or second quantum computers, there are still no real well-working quantum computers that hold information and can process the amount of information that would be necessary. Apart from this, some other things are also unclear about quantum computing and can hold back quantum computing advantages.

It's not clear yet how fast one will be able to input information into a future quantum computer. If inputting information will turn out to be slow because of some physical law, then algorithms that require large inputs will be disadvantaged. This is often where a quantum computer should get a quantum advantage. The output could also be a problem. Just like the input problem, the output could also turn out to be a bottleneck, especially when the full solution of the system read out as a string of bits would be exponential. For many algorithms, it is uncertain how many operations one would need to make the algorithm quantum if the depth of a circuit becomes deep. To see where there are practical advantages, one should be able to simulate or use the algorithms, but that stage of the field is still many years away.

The quantum linear system algorithm (QLSA) is generalized by B. D. Clader et al. [15] to a wider range of problems. It comes from a quantum algorithm designed by Harrow et al. [32], used by many quantum machine learning algorithms, to solve a linear system of equations:

$$A\,|x\rangle = |b\rangle \tag{2.15}$$

With A defining the system of linear equations. To give an idea of how deep and wide a circuit would need to be to achieve a quantum advantage Scherer, A., Valiron, B., Mau, SC. et al. [70] calculated this for QLSA. They found that a quantum advantage would only become possible for a problem size of 332 020 680 and to achieve a quantum circuit that can handle such a problem, it would need a depth of 10^{25} with 340 qubits. With no optimization, that would result in a shallower depth and fewer qubits. For factoring a 1024-bit integer with Shor's algorithm, it is estimated that a depth of 2.24×10^{11} [83] is needed. To factor such integers on a classical computer, one would need a year of computation on a 10M$ device [54]. achieving these depths and amount of qubits will take many years to achieve, knowing that the standard is 0 error-corrected qubits and a depth of around 10 for non-error-corrected qubits [63].

2.13 QC4EO

Earth observation(EO) is the field that applies information like satellite images or local ground sensors to, for example, study the evolution of the environment. Through this gathered information, research can be conducted and businesses or institutions can be helped with certain problems. There are multiple areas where quantum computers could become helpful for Earth Observations in the future. This field requires a lot of computational processing for data processing, monitoring of the climate, and making forecasts of the weather. The computational promises of quantum computers could push the field forward.

In section 2.10 a paper is mentioned [72] that goes over an application for earth observation. Some of the processing happens with the aid of neural networks, e.g. in the paper, a neural network was trained to classify images of the ground. Quantum layers or quantum neural networks could provide an increase in accuracy or convergence speed as shown in [72] or an improvement of computational intensity, i.e. less power and complexity with respect to the neural network architecture, is needed to achieve the same performance. These improvements could intuitively be understood by looking at how a quantum circuit works and what it does. This reasoning will be explained later on in the thesis. In section 2.10 also another paper is mentioned [35], one that also goes over something that could benefit image processing, i.e. quanvolutional layers. Combining these two things, one could make a fully quantum convolutional network.

Another way to see if quantum machine learning will ever be helpful for the computational needs of the field of earth observation is by finding data that shows a computational advantage for quantum machine learning. This is done by Hsin-Yuan Huang et al. [39] and Manish K. Gupta et al. [30]. First, one does a principal component analysis so that the dimensionality of the data is low. One then keeps the most important features of the data. To transform the data, one can use Projected Quantum Kernels(PQKs). The data is transformed by taking the data through a quantum circuit. An example of how this is done is by applying 3 rotations, one for each axis, to each qubit of the circuit. Then the data is put into the circuit by applying a unitary transformation to the qubits of the form:

$$V(x_i) = exp(-i\sum_{j=1}^{N} x_{ij}(X_jX_{j+1} + Y_jY_{j+1} + Z_jZ_{j+1})) \tag{2.16}$$

With N, the number of features or the number of qubits, x_i a data point, and the X, Y, and Z the Pauli operators. After this layer of the circuit, the circuit is read out and the X, Y, and Z values of the qubits are stored as new features and the Gaussian kernel is calculated:

$$k^{PQ}(x_i, x_j) = exp(-\gamma^{PQ}\sum_{k} ||\rho_k(x_i) - \rho_k(x_j)||^2) \tag{2.17}$$

γ can be fine-tuned so that the neural networks perform better. This kernel matrix is used to relabel the new data points based on the largest eigenvector of the kernel. To check the advantages of the quantum data to the classical data. A neural network is trained on the classical data and one is trained on the new features. Both neural networks are trained with the new labels. To compare the two networks, the accuracy of the validation data can be used. Both papers found that for certain new labels and features, there was a quantum advantage. Manish K. Gupta et al. [30] found almost a doubling of their accuracy on the validation data, from 40 to 80, with these PQKs.

2.14 Summary

This chapter looked at both the quantum computing and machine learning aspects of this thesis. Quantum computing is introduced and some fundamental concepts are explained. One of the biggest things holding back quantum computing at the moment is explained, i.e. error correction. Some examples of popular algorithms that promise a computational advantage have been looked at and the role of superposition and entanglement in these algorithms are explained. The most used type of quantum qubit has been further explained and some further predictions of companies on their qubit and quantum computers are summarized. The conclusion is that error-corrected quantum computers are not here yet and the companies predict that in around 10 years error corrected qubits will be available, but the amount of computational improvements that they will bring is still uncertain.

As machine learning is half of this thesis, it is also introduced in this chapter and machine learning algorithms like neural networks and convolutional networks are explained, because they are necessary for the experimental part of the thesis. After the basic machine learning, there is a deeper dive into the combination of quantum computing in combination with machine learning, and some hybrid QNNs are looked at and explained. Some caveats are brought up to not get too optimistic by all the promises normally related to quantum computers. A conclusion from this chapter is that the problem sizes that promise a computational advantage require quantum circuits with a large number of corrected qubits and available depth, which will require big breakthroughs in different fields before these requirements will be able to get achieved. In the end, the chapter gets closed with a section on quantum machine learning for earth observation. As this is what the thesis is eventually about. A hybrid between quantum computation and a CNN will be used as a basis and will be closely looked at and expanded upon in the next chapters.

Methodology 3

Imagination is everything. It is the preview of life's coming attractions.

— Albert Einstein

This chapter goes over the why and how of the thesis, i.e. look at the effect of quantum computation on artificial intelligence for the field of earth observation. This is done by looking at the effect of putting quantum circuits into neural networks relevant to the field of earth observation, e.g. classifying neural networks and segmentation neural networks. These networks are then used on typical earth observation data, i.e. airborne and satellite images. By looking at the accuracy and convergence speed of the used models, it will be determined if these quantum circuits can be beneficial for their performance. In the review of the literature, many ways of using quantum computing for NNs have been brought up. Here the paper of A. Sebastianelli et al. [72] and their use of quantum circuits in classifying convolutional neural networks will be used as an inspiration to learn how to work with these types of CNNs and to further improve and expand the field. This is done by improving the circuits, taking a closer look at them, and extending their use for segmentation.

3.1 Data

3.1.1 EuroSat and Vito

Classification data consists of images and their corresponding labels. To get a reference of the performance of the classic CNN and to see if it works well, the EuroSat dataset is used as a reference point. The EuroSat dataset [34] contains 10 classes, with each class containing 3000 images. The data has a resolution of $10 \times 10m^2$ and patches of 64×64 pixels. The network gets trained so that the network can classify the images into one of the 10 different classes. The dataset supplied by Vito has 6 classes and is also different because of the land area that is viewable in each image. The data is of resolution $25 \times 25cm^2$, with a pixel size of 64×64 pixels. The data is extracted from the yearly airborne remote sensing flights conducted above Flanders, Belgium [23]. The labels are generated by Vito in an automated way by randomly extracting patches from larger polygons corresponding to one of six classes (agriculture, forest, water, dense urban, sparse urban, industrial).

In Fig. (3.1) one can see some examples of images from both the EuroSat and Vito datasets. One of the main distinctions between the two datasets is the land area in the images. From the example, one can see that the Vito images cover less ground area in each image. Because these pictures are more zoomed in, it is harder to see what is in the picture. One could thus expect that a CNN will have a harder time on the Vito dataset. Another difference is the amount of data in the Vito dataset, i.e. the Vito dataset has 1200 images for every class. The last difference is that the Vito dataset only has 6 classes instead of 10.

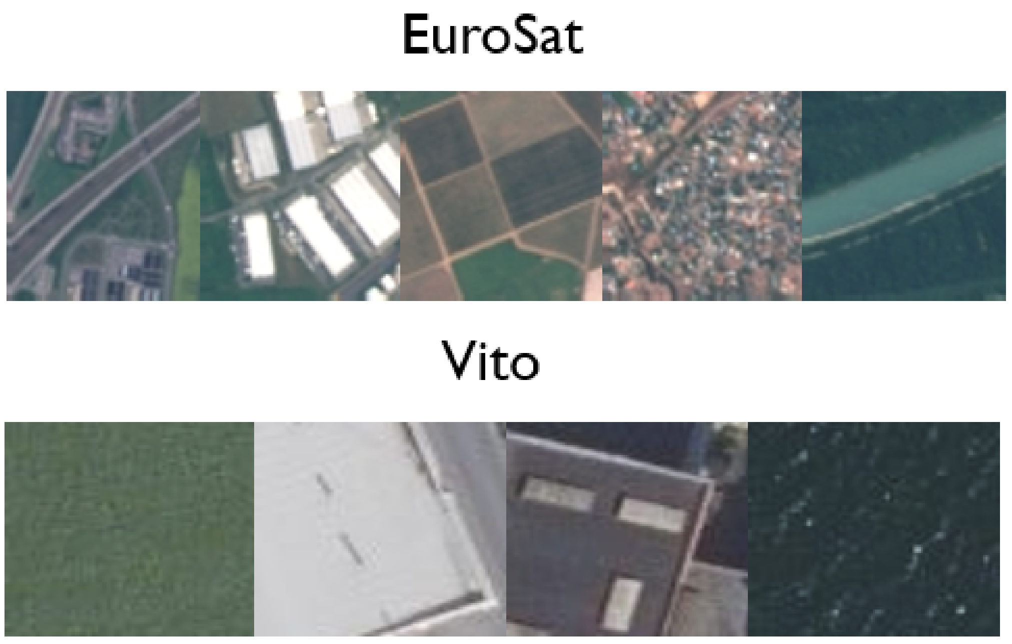


Fig. 3.1: Examples of the data provided by EuroSat and Vito.

3.1.2 Cloud and water segmentation

Segmentation data is different from classification data in the sense that the labels happen on a pixel level and not the image level. For segmentation, one needs a ground truth for training, which has the segmentation of the image. Because this is labeling at a pixel level, one can expect to have less data, as the label of the images is much more expensive than the previous classification labels. For this experiment, two datasets are used. There is one dataset that contains cloud detection [60]. This dataset contains only two classes: clouds, and land. The images in this dataset come from the Landsat 8 satellite and have a resolution of 30m. So the network has to label the pixels that are part of clouds differently from the other pixels. The dataset contains 4 bands, i.e. red, blue, green, and near-infrared. Each of the bands contains 8400 images, which is a part of the full band images. The other dataset is provided by Vito [42] and is about water detection. This data comes from the Sentinel 2 satellite, which has a resolution of 10m and contains 2 classes: open water, and land. The shallow and covered water pixels are included in the land class. This application is related to water detection in the Geo.Informed project [27]. In the ground-truth images, the water is labeled as 1 and the rest is labeled as 0. This dataset also contains 4 bands and 2100 images for every band, i.e. their part of the original full band image.

Fig. (3.2) shows some examples of both datasets. The left examples are of the GeoInformed dataset and the right examples are of the Cloud segmentation dataset. Both examples also show their ground truth to the right of the example. Looking closer, one can also see that at the bottom of the 4th example of the cloud dataset, there is a black area. In the dataset, there are black images or images that are partly black, this is because the images are taken from a big image, which is rotated. The empty space created by the rotation is filled by black pixels.

3.2 Models

3.2.1 Classification

As a first experiment, and to get familiar with these types of hybrid algorithms, an already existing algorithm is used. This algorithm is made by [72]. The Con-

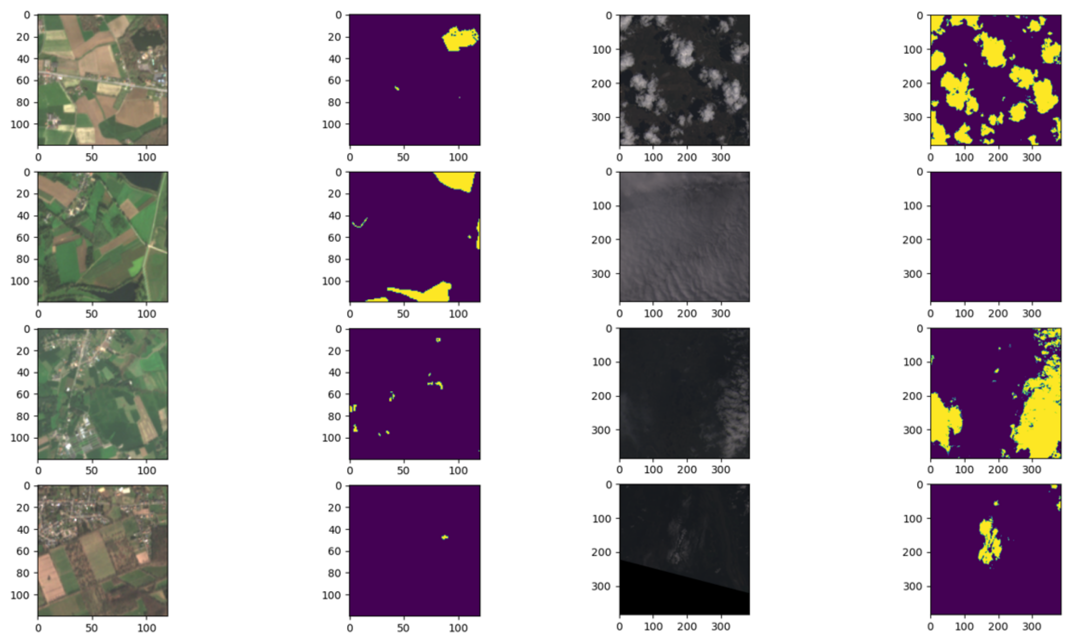


Fig. 3.2: Examples of both segmentation datasets. The left examples are of the GeoInformed dataset and the right examples are of the Cloud segmentation dataset. Both examples also show their ground truth right of the example.

volutional Neural Network(CNN) can be seen in 3.3. In this figure, one can see the algorithm is made of 3 convolutional layers to extract features and reduce the feature space. These layers are followed by densely connected layers to further handle the classifications from the features. In the middle of the dense layers, part one can see the quantum circuit. This Quantum circuit takes as input the values of the previous layers. These values get encoded in the quantum circuit by applying a rotational gate around the Y-axis onto the qubits. After this, different things can be done to the circuit, e.g. entangling the qubits or/and doing another rotation with more values from the previous layer. In the experiment, 3 different quantum circuits are compared.

3.2.2 QCNN and different quantum circuits

A quantum circuit is implemented into the CNN, to make a QCNN. First, the effect of the different kinds of quantum circuits will be looked at, see Fig. (3.4). The first circuit has no entangling between the qubits and just encodes the previous features into the qubits by applying rotational gates on the qubits. The second

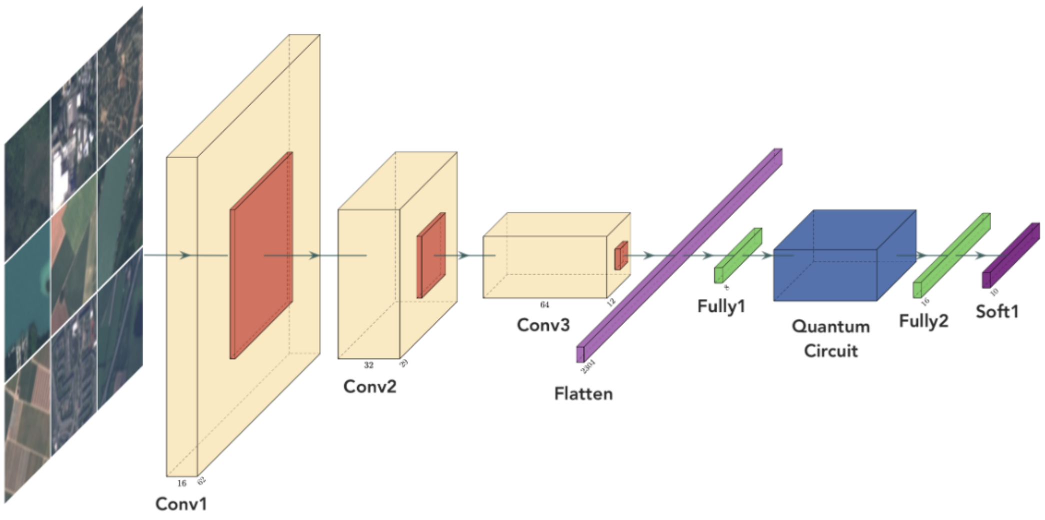


Fig. 3.3: "Proposed hybrid Quantum Neural Network Classifier. The network is a modified version of LeNet-5, where the blue box indicates the Quantum Circuit layer." [72]

circuit works by first pairwise entangling all qubits next to each other, then encoding the previous features, and then again applying the same entanglement. The last circuit makes use of more features from the previous layer. This layer begins with applying the rotational gates and then entangling all qubits, such that the circuit is fully entangled. After this, additional rotational gates are applied to encode even more features. When using 4 qubits, this last layer is thus able to encode 8 features from the previous layer. The output of these circuits is a probability distribution over potential qubits sequences, e.g. for 4 qubits, one has 16 outputs 0000, 0001, 0010... . One can sample from the circuit and divide the counts of each sequence by the total amount of samples, or one can directly calculate the probability of achieving a certain sequence.

The three circuits can be seen in Fig. (3.4).The circuits in order are the RY circuit, the Real Amplitude circuit, and the Bell circuit. The squares with RY in them are the rotational gates around the Y-axis and the connected circles are the entanglements between qubits, i.e. the CNOT gates. The small circle is the control qubit and the large circles are the target qubits. The previous layers get pulled through a tanh function and multiplied with pi to better translate the features to the quantum circuit.

By looking at the first of the three circuits, one can start to understand what these quantum circuits do. One starts with all the qubits in the 0 state, such that all qubits

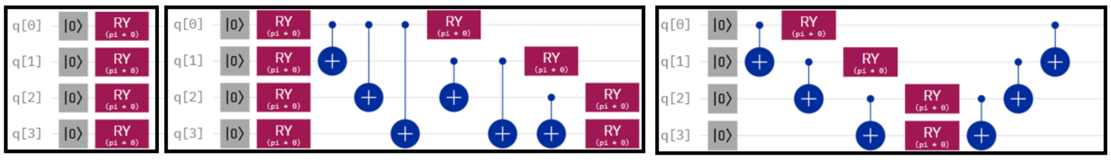


Fig. 3.4: Visualisation of the different applied circuits. The first circuit is the RY circuit, the second one is the Real Amplitude circuit, and the last one is the Bell circuit.

have a 100 percent chance of being in the 0 state. Then the feature of the previous layer gets encoded in the qubit using a rotational gate. The maximum that the feature can rotate the qubit is 1 pi, and the minimum is -1 pi. Rotating a qubit by pi or - pi has the same probabilistic outcome apart from a phase shift, which is not noticeable by looking at the probability. Rotating the qubit by pi makes it so that the qubit has a probability of 1 of being in the 1 state. Thus features get mapped to the probability of qubits being either 0 or 1. The higher the output of the node in the previous layer, the higher the probability of finding the corresponding qubit in the 1 state. The output of this layer is not just the probabilities of the qubits but the marginal probabilities of qubit combinations, i.e. the output of this layer is the probability of every configuration of qubits. So in addition to this map from features to probabilities of qubits, there is another map happening, i.e. from single probabilities to the probabilities of all the combinations of qubits. One can add additional gates. However, often these do not change the mapping, e.g. placing a Hadamard gate at the beginning of the circuit would be the same as adding 0.5 pi to the feature.

CNOT gates make the qubits or features depend on each other, this makes it so that they shift the probabilities of the qubit combinations around. When one fully entangles the circuit, the probabilities do not change in amplitude, but the location of probabilities changes. Without entanglement, the state of the last qubit in the 1001 state will always mean the same, i.e. the feature has very high activation and thus is probably present in the input. If one entangles the circuit, then the 1001 state does not mean that the last feature is present. This output now means that the first feature is present and the last feature is not. However, the 0001 state has remained the same, i.e. the last feature is present. This can work counterintuitively and one can expect the output to be less interpretable for the next layer because of this switching of meaning. From this, one could assume entangling features on a quantum circuit lead to worse performance in both convergence and accuracy. However, one can also look at it differently. When one has two similar pictures, one will get similar extracted features. Because of the similar features, these images are hard to separate for classical networks. When the features go through the quantum circuits, because of the entangling of qubits, small differences in features can result in a big separation in the output space. Because of the bigger distance in the output space, the other layers have an easier time operating the two similar images. This will be looked at further in the result chapter.

The second shown circuit in Fig.3.4 encodes the first 4 features on the qubits with RY gates, then applies CNOT gates between all qubits so that the circuits become fully entangled. After the CNOT gates, the last 4 features get encoded on qubits again. Because of the periodic nature of the encoding, one could think that this layout would do worse. Because of the two encodings on one qubit, there is some loss of information as one looks only at the probabilities of the qubits and not the shifts. In the last circuit, the qubits get entangled twice. The first entangling constantly shifts the rotations of the qubits. So this is expected not to have a big effect. The second entangling after the encoding also constantly changes the outcome of the output, so this is also not expected to change the performance.

3.2.3 Back propagation

The goal of back propagation [66] is to make a neural network learn from labeled data. By adjusting the weights and biases of each neuron, the network can learn to correctly map inputs to the expected outputs. Weights and biases are adjusted through their effect on the loss function. By using the chain rule, gradients are passed backwards from the output layer to the input layer. The weights and biases follow the negative gradient so that the loss function decreases. This gradient gets multiplied with a learning rate so one can choose how fast the weights and biases change. This is done over and over until the network converges, after which the network is no longer able to learn more from the data and starts to overfit [86] and starts to remember the input data.

Because the quantum circuit layer is a self-made layer, one has to define the propagation themselves. To make the QCNN learn, one has to implement the right algorithm that calculates the gradient of the quantum layer efficiently. This can be done by saving the input vector of the circuit and adding a shift and subtracting a shift to one of the inputs. Then one recalculates the outputs of both shifts and subtracts the two outputs from each other to get the gradient of the circuit. This has to be done for all input features of the input vector. The shift one uses can be chosen and can be seen as an intermediate learning rate. From a bigger shift, one can expect that the NN learns faster but has worse performance at the end, and from a smaller shift, one can expect a smaller learning speed, but the network might be able to learn more in the end.

To get a better understanding of these intermediate layers, the shift parameter will be studied. By looking at two different shifts, it will be determined what the effect of small and big shifts will be. The effect will be studied by looking at the convergence speed and the accuracy. Because of bigger gradients, one can expect that for bigger shifts the network will learn faster, and for smaller shifts, the network will converge slower but achieve a higher accuracy. To find two shifts to compare for the big one, $\frac{\pi}{4}$ will be used. As this is also used as the shift in [72]. To find a small shift that results in good networks with good accuracies, a grid search will be done.

3.2.4 Coarse To Fine

To see how good the QCNN can be, a coarse-to-fine classifier is made. This is a classifier made out of 3 QCNNs. One that classifies all the data into subsets of classes and then two more models, each trained on a specific subset. These subsets are chosen so that the classes in the subset are hard to distinguish, such that another classifier only trained on the subset can do a better job at classifying the classes in the subset. This is done for both a normal CNN and the QCNN to see the difference between the two. First, to choose the subset, one looks at the confusion matrix, i.e. a matrix from which the accuracies can be derived and also the confusion present between the classes. By looking at this confusion matrix, one can then subdivide all the classes into two subsets. In this case, that is a hard-to-distinguish subset and an easy-to-distinguish subset. In Fig. (3.5), one can see a visualization of how the coarse-to-fine classifier works. At the bottom of the figure, examples of the data are given for the two fine classifiers.

3.2.5 Segmentation

As a second part of the experimental part of the thesis, a segmentation algorithm is made hybrid. This is done to see the effect of the quantum circuit on the performance of segmentation and to see if there is, just like the classification part, an increase in performance. The classical network is called a U-Net, i.e. because of its skipping layers, it can be visualized as a U. Fig. (3.6) shows a visualization of the classical UNet. This classical UNet will first be used to get a baseline of the performance, just

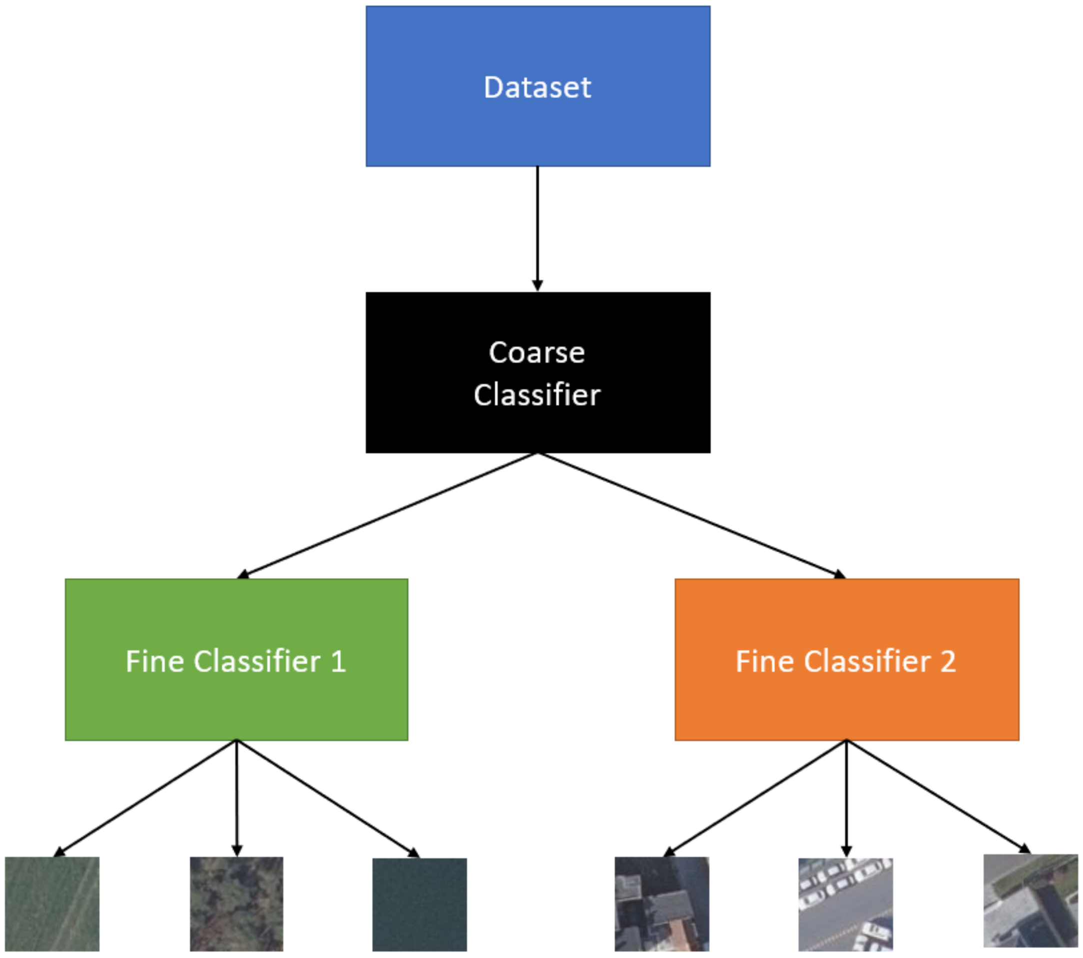


Fig. 3.5: A visualization of the coarse-to-fine classifier.

like part 1 of the experiment. This will again be done on two datasets, to be able to compare the performance.

3.2.6 Classical UNet

Just as for the CNN part of the experiments, an experiment is done to get a baseline of how well a classical neural network will do on the datasets. This then later gives an idea of how well the QUNet is doing. The UNet is made of contracting blocks which are the gray blocks in Fig. (3.6) and expanding blocks which are the green blocks. The contracting block is made of 2 convolutional layers which apply a given amount of filters of kernel size 3 with stride 1 on the input. These layers are depicted

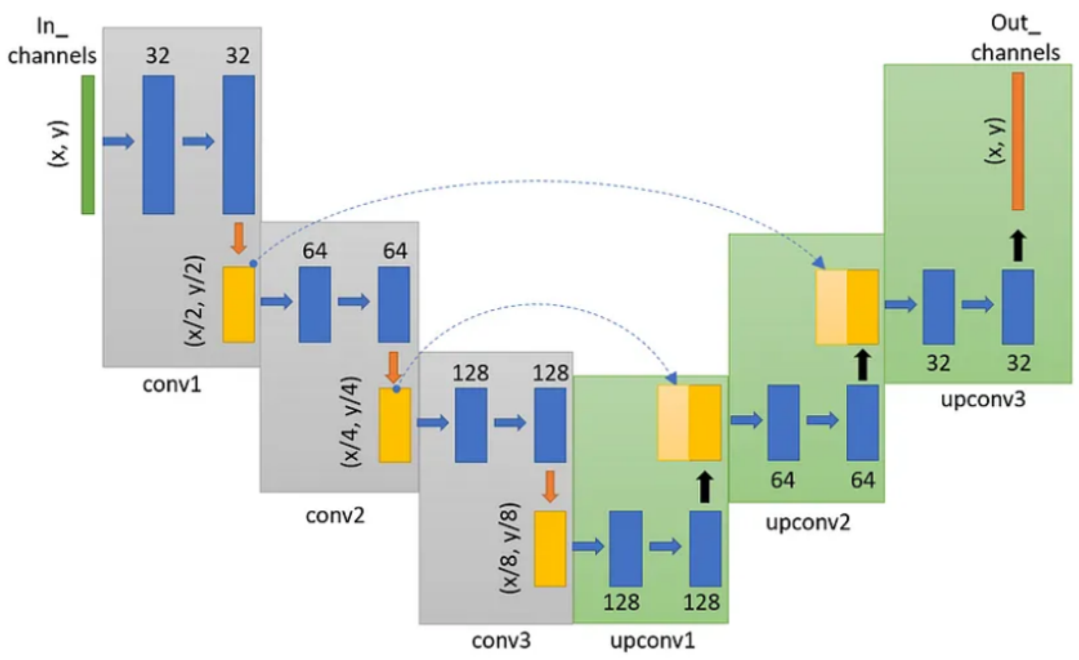


Fig. 3.6: Visualization of the U-Net. [18]

by the blue arrows in the figure. At the end of this block, there is a max pool layer of kernel size 3 and stride 2, indicated by the orange arrow pointing downwards. To make this a UNet and not just an encoder/decoder, one needs to add skipping layers. These are indicated by the blue dashed arrows. To make it so one does not have too many quantum layers, the feature space needs to be reduced. Here this is done by reducing the number of filters. Instead of a typical 120 filters at the deepest point of the UNet, only 12 filters are used. If one would have too big of a feature space, the simulation of too many quantum circuits would be needed and the training would take too long, as the quantum layer is the most time-consuming computation and makes it so the network is not fully able to run on a GPU. At the deepest point, because of these contracting blocks, one has a feature space of size 2700. This is around 11 times smaller than the previous feature space of size 28800. Then another change is made by changing the padding of one of the contracting blocks, i.e. the padding of the second padding block is set to 0 instead of 1. This is done to get a 14 by 14 output at the deepest point, such that a quantum layer with 7 qubits can be used instead of a quantum layer with 8 qubits. This reduces the training time by a factor of 2 and also reduces the output of the quantum circuit by half. The size of the feature space of this final UNet architecture is 2352. To know what the impact is of this reduction, the performance of both architectures will be looked at.

3.2.7 QUNet

The quantum layers used in the QUNet are the same quantum layers used for the QCNN section. From the three options, the quantum layer that can encode the most features is chosen, as the feature space that needs to be encoded is still large compared to the QCNN. This Real Amplitude quantum layer is also one of the best-performing layers, as one will see in the experimental section. Because the output of the quantum circuit with 7 qubits is $2^7 = 128$, a dense layer is used to get the size back to the original size so that it can be used by the next layers in the network. As for the classical UNet, both the performance of the network without and with skipping layers will be looked at. The performance of the network without skipping layers is looked at to get a better idea of the effect of this quantum layer on the performance of segmentation networks. In the network, without skipping layers, all features in the later layers have been through the quantum circuits. In the network with skipping layers, one also has the features coming from the section before the quantum circuit, so one could expect the quantum layers to have less of an impact on the overall performance. In Fig. (3.7) one can see a visualization of the QUNet. The green array that comes after the quantum circuit represents the output of the dense layer, which is again 14 features. From this figure, one can see where precisely the quantum circuit is applied. The quantum circuit is applied to all features of the $12 \times 14 \times 14$ matrix. The output of the dense layers again forms a $12 \times 14 \times 14$ matrix.

3.3 Summary

This chapter went over the methods of investigating how quantum computation can affect the performance of CNN or segmentation networks. The data used for the experiments is explained and examples of data points are given. Multiple types of quantum circuits are looked at, how they work is explained, and the reasons why they might benefit CNNs and segmentation networks have been explored. The main takeaway from this is that the features in the middle of the network get mapped to a vector with the probabilities of all the possible combinations of present features. The role and effects of entanglement in these circuits have been explained. The architecture of the classical versions of the convolutional neural network and the UNet are visualized and explained. The hybrid or quantum versions of the networks

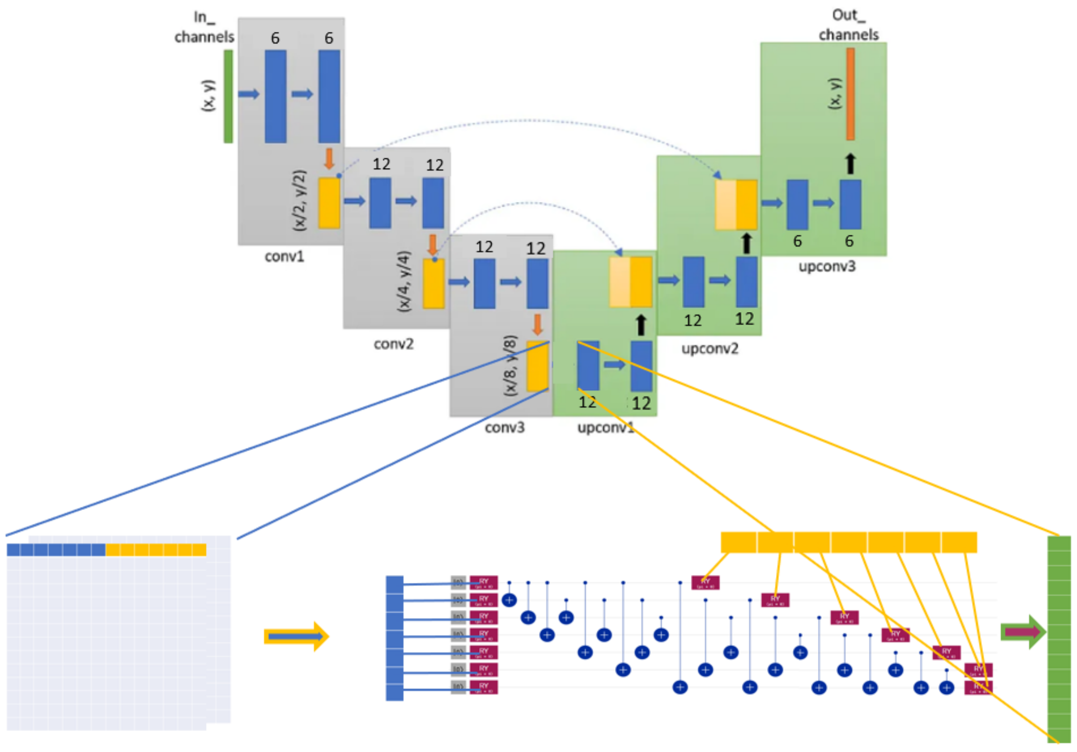


Fig. 3.7: Visualization of the QUNet.

are also visualized. The location of where the quantum layer gets deployed is also explained. The next chapter goes over the results coming from the mentioned experiments to see if these quantum circuits affect the performance of both CNN and segmentation algorithms.

4 Results

Peace begins with a smile.

— **Mother Teresa**

The experimental part of the thesis is made of two parts. The first part takes a classification algorithm used by A. Sebastianelli et al. [72] for an earth observations task. This algorithm gets re-used and updated. It is trained on data provided by Vito to see how it will do on their dataset and if similar performance as promised by [72] is achieved. The algorithm gets updated by updating the quantum circuit. Because of this update, the circuit can run 20 times faster and with greater accuracy. This is achieved by switching from Qiskit to Qulac drivers and using different methods of calculating the probability distribution, such that probability distributions over the qubits output space are calculated more efficiently. This is necessary to be able to run more and bigger circuits. Without this, part two of the experiments would be unrunnable through long runtimes. There is also a closer look at a shift parameter that is used for backpropagation and the role of entanglement is also discussed. The role of the number of qubits in the quantum circuit also gets looked at. A classic CNN is trained to be able to benchmark the modified one. By looking at the accuracy and the convergence speed of both networks, it will be determined if the quantum circuits and thus quantum computing can improve CNNs in this way.

The second part goes into new frontiers by using these quantum simulations and designing a hybrid U-Net. This is a segmentation algorithm that uses quantum circuits to transform its feature space. This is done to see the effects of quantum circuits on the convergence speed of segmentation networks and their pixel classification accuracy.

	Validation accuracy
EuroSat	0.813 ± 0.018
Vito	0.673 ± 0.014
EuroSat limited data	0.705 ± 0.007
EuroSat Limited data/classes	0.793 ± 0.010

Tab. 4.1: Validation accuracies of classical CNNs.

4.1 Classification

4.1.1 CNN

To have a baseline of how a classic CNN would perform on these datasets, a classic CNN is trained on both datasets. The classic CNN is mostly the same as the QCNN in Fig. (3.3) but without the quantum circuit layer and dense layers of the size and order 128, 64,32,10. The results can be seen in Tab. (4.1). One can see that the CNN performs worse on the Vito dataset. To see why this is first, some other experiments are done. A CNN is run on EuroSat with the same amount of data as the Vito data, to see if it is just the amount of data that makes the CNN perform better. However, one can see that even with this limited amount of data, the EuroSat performs better, even though this is not by much. This test is unfair as the EuroSat dataset contains 10 labels instead of 6, one could thus expect the CNN to do worse. Another CNN is trained on the EuroSat dataset, now with the same amount of data as the Vito data set and the same amount of classes. To make it fair, a combination of easy and difficult-to-classify classes is chosen from the EuroSat dataset. The results show that, with these restrictions, the CNN performs almost as well as on the full dataset. One can thus assume that this is because the land area per image is smaller for the Vito dataset, which results in a hard time for the CNN to detect good features. Table (4.1) shows that the baseline accuracy of a normal CNN on the Vito dataset is 67.3 present.

4.1.2 QCNN

Accuracy

Tab. (4.2) gives the results from the QCNN experiments with different quantum circuits, which are visualized in Fig. (3.4). The error flags come from the last 10 trained networks, by taking the mean and looking and the statistical deviation of

	Amount of qubits	Validation accuracy
CNN	0	0.673 ± 0.014
RY	4	0.571 ± 0.018
RY	8	0.635 ± 0.026
Bell	4	0.612 ± 0.023
Bell	8	0.634 ± 0.027
Real amplitude	4	0.655 ± 0.016

Tab. 4.2: Validation accuracies of classical CNN and all the different adaptations of the quantum circuit.

the data with respect to that mean. It would be better to look at the variance in the accuracy between models each trained from scratch with different validation sets. However, to get enough data for each instance, this would take too much computing time. After doing a test by looking at this variance, coming from different models trained from scratch, for two different types of models. This variance looked similar to the variance coming from taking the last 10 trained networks, i.e. the difference between the two was both positive and negative and not greater than 20 percent. Thus, it will be assumed that these two variances are the same. From looking at this table, one can conclude some things. By increasing the qubits of the quantum circuits, one can get better performance, especially for the RY circuit. From the different quantum circuits and amount of qubits, one can see that the real amplitude circuit performs the best. The performance of the RA CNN is still worse than the normal CNN. Even though the error flags are not exclusive regarding the next two best-performing quantum circuits, the real amplitude circuit will be used for the coarse-to-fine classifier.

CNN	Validation accuracy
RY	0.663 ± 0.010
Bell	0.700 ± 0.014
RA	0.700 ± 0.018

Tab. 4.3: Validation accuracies of all the different adaptations of the quantum circuits, with a small shift.

Tab. (4.3) shows the results for the different types of quantum circuits, but now with a small shift. By doing a grid search for a small shift, it is found that $\frac{\pi}{30}$ achieves good accuracies. By comparing this table to the Tab. (4.2) one can see all the accuracies of all the circuits are improved. The RY circuit is still the worst-performing circuit and the other two have the same performance. Because it is the worst-performing circuit and the only one without entangling, one could say that entangling has a positive impact on the performance of the hybrid CNN. From this one conclude that from the two viewpoints discussed in the methodology section, the latter thus outweighs the first viewpoint. The increase in distance between images with similar features in the output space thus positively impacts the performance of the network. All accuracies have increased by around 5 percent and the error bars have also decreased, which shows that the network is more stable during training. By comparing the accuracy with its error bars of the circuit types to the same type in the other Tab. (4.2), one can conclude that the smaller shift does result in better performance. As the error bars are not overlapping. The results in Tab. (4.3) for the Bell and RA circuit are also better than the classic CNN, the error bars still overlap but not by much.

Convergence

From the explanation of the quantum circuits, one could suspect that the convergence of a CNN could differ from a CNN with a quantum circuit layer. Because of the two mappings of the quantum circuits, the next layer gets the probabilities of all feature combinations. As this output is easy to interpret, the next layers have an easier time inferring the label of the image from the features that are present.

In Fig. (4.1) one can see the smoothed validation accuracies from the different QCNNs and the CNN. These curves are smoothed by taking the mean of the last 10 accuracies to be able to better see the convergence. One can see from the figure that the QCNN with a big shift is converged around epoch 30, i.e. around this epoch, they achieve their highest accuracies. From the CNN curve, it is clear that the normal CNN trains slower and thus converges slower to the optimal parameters. The CNN convergence around epoch 60, which is double that of the QCNN with a big shift. So although the accuracies have not changed too much, one can conclude that a QCNN trains faster and thus convergences faster. In the figure, one can also see the accuracy curve of the real amplitude quantum circuit with a small shift. This curve converges faster, however, it is able to achieve higher accuracies. This QCNN converges around 70 epochs, which is later than the CNN. One can conclude from this that this shift to calculate the derivative of this layer works as a dial that one can use to choose between convergence and accuracy.

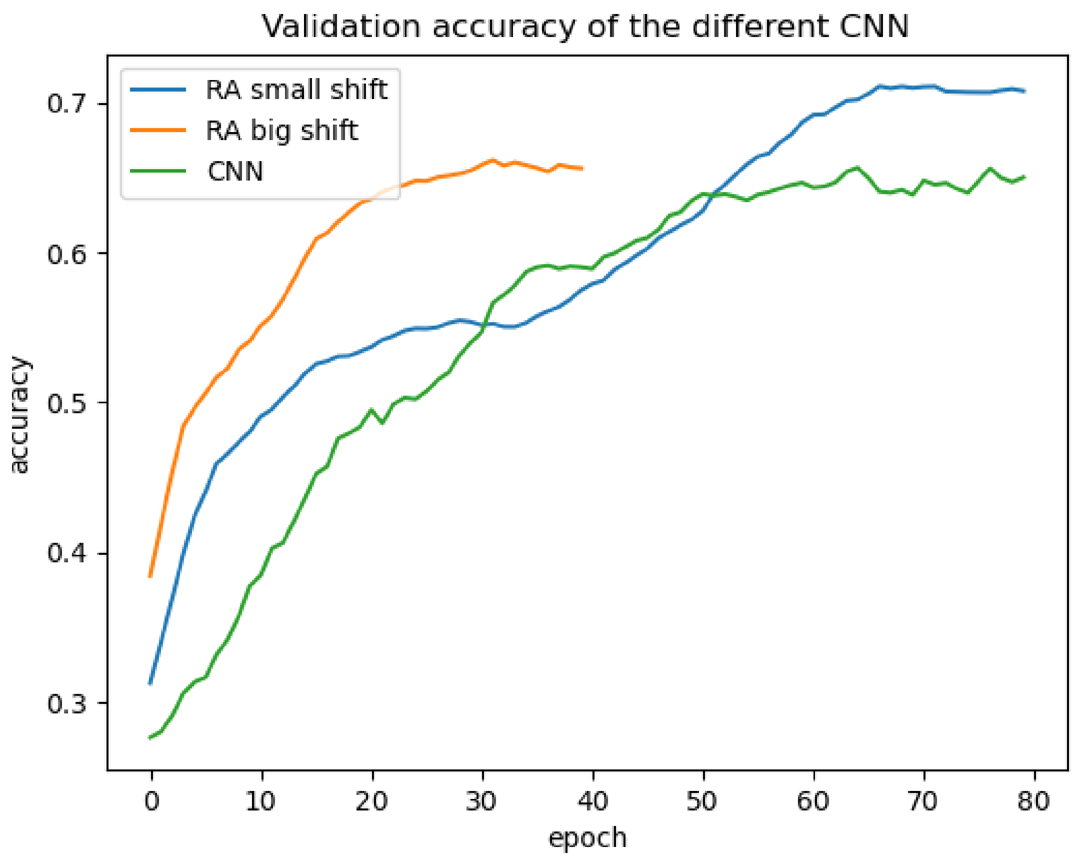


Fig. 4.1: The smoothed validation accuracies of different QCNNs and the classic CNN.

4.1.3 Coarse-to-fine classifier

In Fig. (4.2) one can see the confusion matrix of a QCNN with the real amplitude circuit. In this figure, one can see the accuracy of the classification of each class but also the percentage of the class labeled as another class. Looking at this figure, one can see that dense urban, industry and sparse urban classes are confused with each other a lot. Therefore, these 3 are put in a subclass and the other three are also put in a subclass. From the amount of confusion between the sets, one can see this as an easy-to-distinguish subset and a hard-to-distinguish subset. The relative confusion between the different classes are same for the CNN and QCNN therefore, the subsets are the same for the two.

Tab. (4.4) and Tab. (4.5) give the results for the coarse-to-fine classifiers. By comparing the two, one can see if the QCNN is superior to the normal CNN. One can

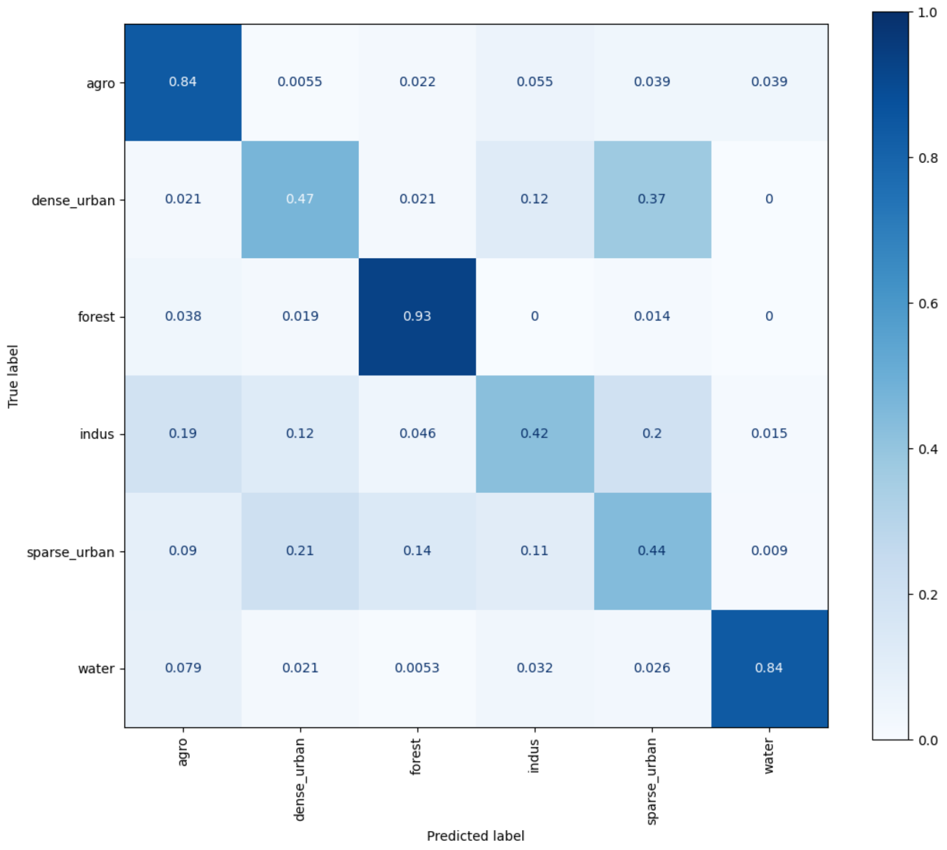


Fig. 4.2: Confusion matrix of the quantum convolution neural network with the real amplitude circuit.

CNN	Validation accuracy
Average	0.666 ± 0.014
Coarse	0.890 ± 0.012
Fine easy	0.869 ± 0.022
Fine hard	0.626 ± 0.026

Tab. 4.4: Validation accuracies of classical CNNs in the coarse to fine experiment.

see by looking at the Coarse and Fine classifiers for both the CNN and QCNN that the QCNN scores best on all three of them. Not only is there an increase of around 4 present on every classifier, but they are also more stable. The error bars for the three classifiers of the QCNN are considerably smaller than those of the CNN, suggesting that the training of QCNNs is more stable. This could be because of the feature space mapping of the quantum circuits, giving the following layers a more stable and complete input. The average accuracy also shows that overall, the QCNN scores

QCNN	Validation accuracy
Average	0.709 ± 0.012
Coarse	0.902 ± 0.007
Fine easy	0.910 ± 0.016
Fine hard	0.650 ± 0.013

Tab. 4.5: Validation accuracies of classical QCNNs in the coarse-to-fine experiment with a big shift.

QCNN	Validation accuracy
Average	0.718 ± 0.009
Coarse	0.902 ± 0.007
Fine easy	0.929 ± 0.009
Fine hard	0.658 ± 0.013

Tab. 4.6: Validation accuracies of classical QCNNs in the coarse-to-fine experiment with a small shift.

4 present better than the CNN. By comparing Tab. (4.5) to Tab. (4.2) one can also see that the coarse-to-fine way of classifying paid off for the QCNN but not for the CNN. The latter seems not to be affected by a coarse-to-fine classifier, as the accuracy stayed the same. From all this, one can thus conclude that the coarse-to-fine QCNN classifier is the best-performing classifier on this dataset even when looking at the error bars, i.e. the error bars do not overlap with each other.

Tab. (4.6) shows the results for the coarse-to-fine classifier but now with a small shift in the quantum circuit. Comparing this table to Tab. (4.5) one can see that the accuracy has increased and some of the errors have decreased but not by much, i.e. the results for both training stability and accuracy are comparable. Comparing all variations of the QCNNs and the classic CNN. One can see from the results that the coarse-to-fine model with a small shift is the best preforming model on this dataset. This could be expected as from the previous section, it was clear that the quantum circuit with a small shift performed the best and this model is also more complex than the model made of only one network.

4.2 Segmentation

4.2.1 Classical UNet

Tab. (4.7) displays the validation accuracies of different kinds of UNet and different kinds of datasets. One can see that the UNet has an easier time detecting water than clouds. This could be because of the black parts of images and total black images spread throughout the dataset, as mentioned before when going over the examples. Or this can also be because clouds are harder to segment than water, because of the transparent property of clouds. The data of GeoInformed WSL is from a UNet trained on the GeoInformed segmentation dataset but Without Skipping Layers(WSL). Removing the skipping layers seems to have an impact, although not a very big impact. The accuracy is lower by around 1 present and it seems less stable as the deviation is higher. The next two accuracies come from the limited UNet, i.e. the UNet, with a different number of filters and padding to be better suited for the quantum layers. Comparing the last two with each other, one can see that there is no difference with or without skipping layers. When comparing the performance with the not limited UNet, one can see that there is a decrease in performance when reducing the number of filters and padding. This reduction has the biggest effect as it causes the accuracy to drop by 3 percent.

	Validation accuracy
Cloud segmentation	0.970 ± 0.004
GeoInformed	0.9967 ± 0.0002
GeoInformed WSL	0.983 ± 0.005
GeoInformed limited	0.965 ± 0.005
GeoInformed limited WSL	0.964 ± 0.005

Tab. 4.7: Validation accuracies of the classical UNets.

4.2.2 Quantum UNet

To make the quantum UNet work, some experiments are done, as a comparable performance to the classical UNet was not as straightforward as for the QCNN in the previous sections. One can make an individual dense layer for each quantum circuit, this dense layer can then focus to optimally map the circuit outputs back to their original size and keep as much information as possible. However, when doing this, the QUNet was not able to learn anything and converged to a state where it only predicted the same label for each pixel. This could be because of the large number of extra weights added because of all the dense layers. One needs 168 dense layers if one wants a unique dense layer for each circuit, this can make it such that the network has too many weights to optimize efficiently and thus fails. This problem is fixed by only using one dense layer for all quantum circuits. The network is also more sensitive to the shift of the quantum circuit for backpropagation. A shift of 1/4 resulted in a bad performance. 1/30 worked better but was still not optimal. After some experimentation, 1/120 came out as an optimal shift for both accuracy and convergence speed.

Accuracy

In Tab. (4.8) one can see the validation accuracies of the quantum UNets and their deviations during training. Comparing the two to each other, one can here conclude that skipping layers did improve the performance of the network significantly, i.e. the error bars are not overlapping. This could be explained by using the first viewpoint on using entanglement in the quantum circuit. Because the network now needs to go from the output space of the quantum circuit and the dense layer back to a classified image, the changes in the output space can negatively affect the network to do so. However, the skipping layer can combat this by adding more primitive features with more accurate locational information. The combination of the two different output spaces of the quantum circuit and the skipping layer can then be a more clear space regarding the classical network making for speed up in convergence. Comparing the performance of the quantum UNets to the performances of the classical UNets in Tab. (4.7) one can see that the quantum layer did not increase the performance enough for the error bars to be exclusive however, the accuracy of the QUNet is higher with the skipping layer. Without the skipping layer, the QUNet performed significantly worse than the classical QUNets.

	Validation accuracy
GeoInformed QUNet	0.968 ± 0.004
GeoInformed QUNet WSL	0.956 ± 0.003

Tab. 4.8: Validation accuracies of the quantum UNets.

Convergence

In Fig. (4.3) one can again see the effect of the skipping layers as can be seen in the performance tables (4.7, 4.8). One can again see the accuracy differences. This figure is to compare the convergence speed for the different variations of the UNet. The figure is just like the convergence figure for the QCNNs, with an average accuracy of the last 10 scores to better see the convergence. One can see that the QUNet with skipping layers, convergence is significantly faster than all the other models. The classic UNet takes around 3 to 4 times longer to achieve optimal results. One can get a faster convergence speed for the classic UNet by using a higher learning rate but after doing this, it resulted in a worse overall classifying performance. From this, one can conclude that the quantum layer improves the convergence speed of the UNet significantly, but accuracy stays around the same.

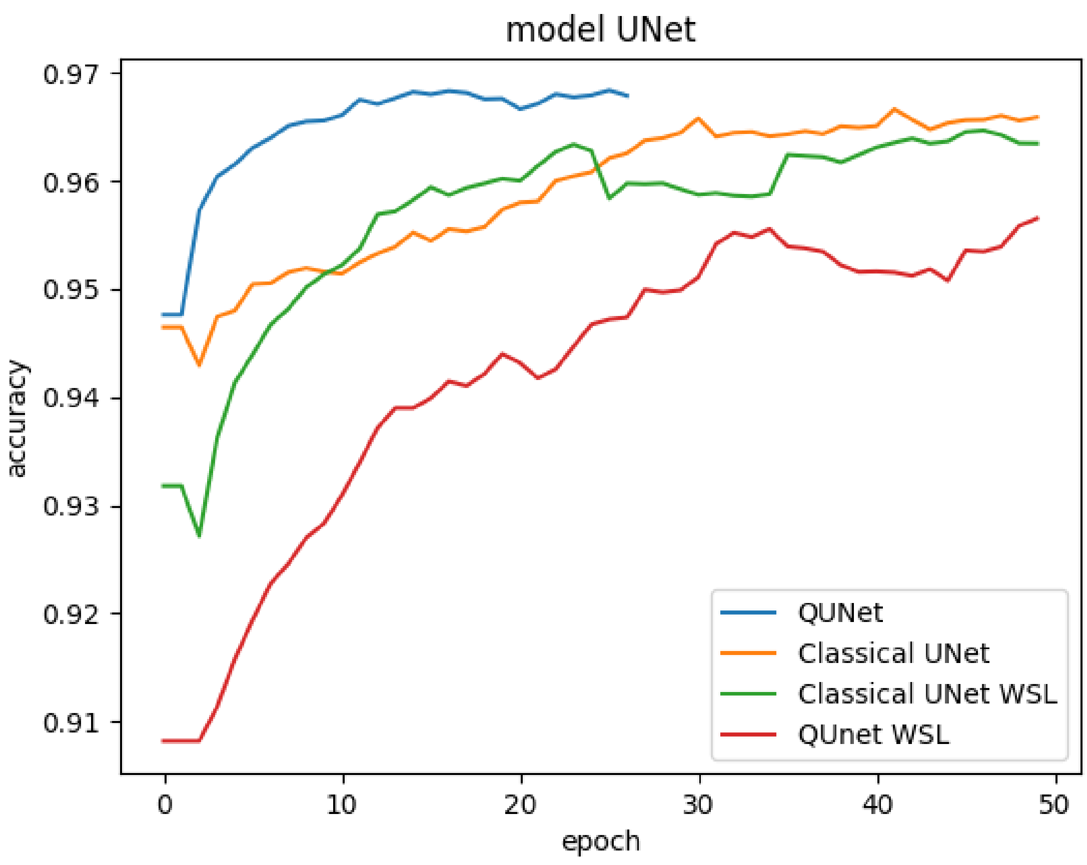


Fig. 4.3: Running validation accuracy of the different segmentation models.

For the segmentation network, the quantum layer did not show similar behavior as for the QCNN. The shift can not as easily be controlled to get either a better convergence speed or optimal classifying accuracy. Experiments are done with even smaller shifts than 1/120, up to 1/1000, but this did not result in better-classifying abilities. Also, to make it work, the feature space needs to be reduced. This results in a significant decrease in classifying accuracy regarding the classical UNet with no modifications. However, the quantum layer did show a significant improvement in convergence speed and a small accuracy increase. Again, showing that also for segmentation, these new layers can benefit their performance. In the future, when quantum computers can run bigger qubits circuits or when the computing pipeline for using these circuits gets improved, these circuits could be used in bigger networks, which might also benefit from this found behavior.

4.3 Summary

The experimental section of the thesis looked at the experimental performance of quantum layers in both CNNs and UNets. From this section, some conclusions can be made. Depending on the shift of the quantum layers, the QCNN performance is better at either convergence or accuracy. A large shift resulted in a fast convergence of the network and a small shift resulted in a significantly higher accuracy on the validation set. By comparing the different quantum circuits, one can conclude that entanglement in the quantum circuit gives the best results. For segmentation, the quantum layers showed a smaller performance increase. Faster convergence can be observed, but the accuracy stayed the same.

5 Conclusion

This thesis is a bridge between two very active and evolving fields: quantum computing and artificial intelligence. More specifically, a bridge is built for the field of earth observation. Quantum computing is still in its early development phase, i.e. the field is still working hard on making working prototypes and fabricating practical applications. A.I. is already further, with already many practical working applications. Both fields have the possibility to change the world. One can only dream (for now) of what would come from combining the two. The first aim of the thesis is to introduce both fields. The first chapter goes over some important basics from quantum mechanics and machine learning. It looks at recent attempts to bridge the gap between the fields by looking at recent papers that go over hybrid networks. Different types of quantum computers are explained, together with different types of qubits. The field of companies actively working on making working quantum computers is visualized and their future predictions have been summarized. Caveats are brought up and applications for earth observation are explored. The second chapter goes over the data and models that are used for the experiments and the quantum circuits used in the quantum layers. Chapter 3 presents the results obtained from the experiments.

As this thesis goes over both the quantum computational aspect and machine learning aspect, two types of conclusions can be drawn. There are the conclusions from the literature review of the field of quantum computation and the conclusions from the experiments. The literature review showed that are no fully working quantum computers with error-corrected qubits yet. The field is beyond the phase of making individual qubits. There are many companies with quantum computers with tens of working qubits, which already can be used for experiments. The field is in a phase in which many companies have a vision of how to achieve a fully working error-corrected quantum computer. These companies are working on scaling their already working architecture to thousands of qubits by putting more qubits in each quantum computer and connecting quantum computers. Companies like IBM and Google predict that they will have a working error-corrected quantum computer by 2030. However, to outperform classical computers on important problems, one will need many error-corrected quantum computers, which can take many more years after 2030. It is thus hard to say when practical well working practical quantum computers will be here.

From the experiments, one can conclude a couple of things. The QCNN showed an increased performance compared to the classical CNN. Depending on the shift parameter for the backpropagation of the quantum circuit in the quantum layer, there is either a significant increase in convergence or an increase in accuracy on the validation set. The convergence speed is two times as fast and the accuracy increase is 7.8 percent. The best-performing model on the Vito dataset is the coarse-to-fine QCNN. Entanglement in the circuit showed also an increase in performance from the circuits without the entanglement. The performance of the UNet for segmenting did show less of an improvement. No significant increase in accuracy is found when changing the shift parameter. However, the QUNet did show a faster convergence with the quantum layer, i.e. the QUNet converged 4 times faster.

From the discussion on the workings of the quantum circuits, it became clear that presenting the data in a more interpretable way to the next layer in the network was one of the benefits of the circuit. The quantum circuit gives a more spread-out map of the input features. Another change the circuit brings to a layer is the ability of features to interact with one another by bringing entanglement into the circuit. This makes it so that the presence of one feature can affect the state of another feature, separating more similar inputs. The faster convergence and higher accuracies point towards the fact that these layers do indeed present that data in a better way.

Even though quantum computers might be a long time off, this thesis showed that using some concepts from the quantum computational field can already benefit the performance of classical networks. This work only looked at a small set of possibilities for how these quantum layers can be applied. To get a good understanding of how these layers can benefit machine learning, further research should be done. One could look at more types of quantum circuits, try to improve the efficiency of their computation more so that they can be used in bigger models, try to apply them to different models, ...

References

[1] https://www.ibm.com/quantum/roadmap. May 2022 (cit. on p. 17).

[2] https://ionq.com/posts/december-09-2020-scaling-quantum-computer-roadmap (cit. on p. 17).

[3] Amira Abbas, David Sutter, Christa Zoufal, et al. “The power of quantum neural networks”. In: *Nature Computational Science* 1.6 (2021), pp. 403–409 (cit. on p. 5).

[4] Tameem Albash and Daniel A. Lidar. “Adiabatic quantum computation”. In: *Reviews of Modern Physics* 90.1 (2018) (cit. on p. 11).

[5] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, et al. “Qiskit: An Open-source Framework for Quantum Computing”. In: (Jan. 2019) (cit. on p. 17).

[6] Mohammad H. Amin, Evgeny Andriyash, Jason Rolfe, Bohdan Kulchytskyy, and Roger Melko. “Quantum Boltzmann Machine”. In: *Physical Review X* 8.2 (2018) (cit. on p. 23).

[7] Frank Arute, Kunal Arya, Ryan Babbush, et al. “Quantum supremacy using a programmable superconducting processor”. In: *Nature* 574.7779 (2019), pp. 505–510 (cit. on p. 18).

[8] Lev Bishop, Sergey Bravyi, Andrew W. Cross, et al. “Quantum Volume”. In: 2017 (cit. on p. 8).

[9] G. Blatter, V. B. Geshkenbein, and L. Ioffe. *Engineering Superconducting Phase Qubits*. 1999. arXiv: cond-mat/9912163 [cond-mat.mes-hall] (cit. on p. 15).

[10] Sergio Boixo, Sergei V. Isakov, Vadim N. Smelyanskiy, and Hartmut Neven. *Simulation of low-depth quantum circuits as complex undirected graphical models*. 2018. arXiv: 1712.05384 [quant-ph] (cit. on p. 18).

[11] Samuel L Braunstein and Arun K Pati. “Speed-up and entanglement in quantum searching”. In: *arXiv preprint quant-ph/0008018* (2000) (cit. on p. 10).

[12] H. J. Briegel, D. E. Browne, W. Dür, R. Raussendorf, and M. Van Den Nest. “Measurement-based quantum computation”. In: *Nature Physics* 5.1 (2009), pp. 19–26 (cit. on p. 11).

[13] Colin D. Bruzewicz, John Chiaverini, Robert McConnell, and Jeremy M. Sage. “Trapped-ion quantum computing: Progress and challenges”. In: *Applied Physics Reviews* 6.2 (June 2019), p. 021314 (cit. on p. 12).

[14] Earl T. Campbell, Barbara M. Terhal, and Christophe Vuillot. “Roads towards fault-tolerant universal quantum computation”. In: *Nature* 549.7671 (2017), pp. 172–179 (cit. on p. 8).

[15] B. D. Clader, B. C. Jacobs, and C. R. Sprouse. “Preconditioned Quantum Linear System Algorithm”. In: *Physical Review Letters* 110.25 (2013) (cit. on p. 26).

[16] Susan M. Clark, Kai-Mei C. Fu, Thaddeus D. Ladd, and Yoshihisa Yamamoto. “Quantum computers based on electron spins controlled by ultrafast off-resonant single optical pulses”. In: *PHYSICAL REVIEW LETTERS* 99.4 (July 2007) (cit. on p. 3).

[17] Don Coppersmith. “An approximate Fourier transform useful in quantum factoring”. In: *arXiv preprint quant-ph/0201067* (2002) (cit. on p. 9).

[18] Maurício Cordeiro. *Creating a very simple U-net model with pytorch for semantic segmentation of satellite images*. `https://medium.com/analytics-vidhya/creating-a-very-simple-u-net-model-with-pytorch-for-semantic-segmentation-of-satellite-images-223aa216e705`. Dec. 2022 (cit. on p. 38).

[19] A. J. Dahm, J. M. Goodkind, I. Karakurt, and S. Pilla. *Using Electrons on Liquid Helium for Quantum Computing*. 2001. arXiv: `quant-ph/0111029 [quant-ph]` (cit. on p. 14).

[20] Li Deng. “The mnist database of handwritten digit images for machine learning research”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142 (cit. on p. 23).

[21] Cirq Developers. “Cirq”. In: (Dec. 2022). See full list of authors on Github: `https://github.com/quantumlib/Cirq/graphs/contributors` (cit. on p. 17).

[22] Michel H Devoret et al. “Quantum fluctuations in electrical circuits”. In: *Les Houches, Session LXIII* 7.8 (1995), pp. 133–135 (cit. on p. 15).

[23] *Digitaal Vlaanderen*. `https://www.geopunt.be/` (cit. on p. 30).

[24] A. Yu. Dmitriev and O. V. Astafiev. “A perspective on superconducting flux qubits”. In: *Applied Physics Letters* 119.8 (Aug. 2021). 080501. eprint: `https://pubs.aip.org/aip/apl/article-pdf/doi/10.1063/5.0047690/14551579/080501\_1\_online.pdf` (cit. on p. 15).

[25] Salim Dridi. “Unsupervised Learning - A Systematic Literature Review”. In: (Dec. 2021) (cit. on p. 3).

[26] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. “Quantum computation by adiabatic evolution”. In: *arXiv preprint quant-ph/0001106* (2000) (cit. on p. 11).

[27] *GEO.INFORMED – Remote Sensing amp; Deep Learning for Environmental policy* (cit. on p. 31).

[28] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. “Quantum Random Access Memory”. In: *Physical Review Letters* 100.16 (2008) (cit. on p. 24).

[29] Lov K. Grover. “Quantum Computers Can Search Arbitrarily Large Databases by a Single Query”. In: *Physical Review Letters* 79.23 (1997), pp. 4709–4712 (cit. on p. 10).

[30] Manish K. Gupta, Martin Beseda, and Piotr Gawron. “How Quantum Computing-Friendly Multispectral Data can be?” In: 2022 (cit. on pp. 27, 28).

[31] Zhiying Hao. “Deep learning review and discussion of its future development”. In: *MATEC Web of Conferences* 277 (Jan. 2019), p. 02035 (cit. on p. 3).

[32] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. “Quantum Algorithm for Linear Systems of Equations”. In: *Physical Review Letters* 103.15 (2009) (cit. on p. 26).

[33] Shannon P. Harvey. *Quantum Dots/Spin Qubits*. Feb. 2022 (cit. on p. 14).

[34] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. “EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 12.7 (2019), pp. 2217–2226 (cit. on p. 29).

[35] Maxwell Henderson, Samriddhi Shakya, Shashindra Pradhan, and Tristan Cook. “Quanvolutional neural networks: powering image recognition with quantum circuits”. In: *Quantum Machine Intelligence* 2.1 (2020), pp. 1–9 (cit. on pp. 23, 27).

[36] Loïc Henriet, Lucas Beguin, Adrien Signoles, et al. “Quantum computing with neutral atoms”. In: *Quantum* 4 (2020), p. 327 (cit. on p. 14).

[37] Geoffrey E. Hinton and Richard S. Zemel. “Autoencoders, Minimum Description Length and Helmholtz Free Energy”. In: NIPS’93 (1993), pp. 3–10 (cit. on p. 20).

[38] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. eprint: `https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf` (cit. on p. 20).

[39] Hsin-Yuan Huang, Michael Broughton, Masoud Mohseni, et al. “Power of data in quantum machine learning”. In: *Nature Communications* 12.1 (2021) (cit. on p. 27).

[40] He-Liang Huang, Dachao Wu, Daojin Fan, and Xiaobo Zhu. “Superconducting quantum computing: a review”. In: *Science China Information Sciences* 63.8 (2020) (cit. on p. 14).

[41] Hiroki Ikegami, Hikota Akimoto, David G. Rees, and Kimitoshi Kono. “Evidence for Reentrant Melting in a Quasi-One-Dimensional Wigner Crystal”. In: *Phys. Rev. Lett.* 109 (23 Dec. 2012), p. 236802 (cit. on p. 14).

[42] Xenia Ivashkovych, Lisa Landuyt, and Tanja Van Achteren. “CORSA Deep Earth Observation Semantic Compression applied to Flood Detection”. In: Sept. 2022 (cit. on p. 31).

[43] Mark W Johnson, Mohammad HS Amin, Suzanne Gildert, et al. “Quantum annealing with manufactured spins”. In: *Nature* 473.7346 (2011), pp. 194–198 (cit. on pp. 12, 17).

[44] B.D. Josephson. “Possible new effects in superconductive tunnelling”. In: *Physics Letters* 1.7 (1962), pp. 251–253 (cit. on p. 15).

[45] Richard Jozsa and Noah Linden. “On the role of entanglement in quantum-computational speed-up”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 459.2036 (2003), pp. 2011–2032 (cit. on p. 10).

[46] Amir Khoshaman, Walter Vinci, Brandon Denis, et al. “Quantum variational autoencoder”. In: *Quantum Science and Technology* 4.1 (Sept. 2018), p. 014001 (cit. on p. 23).

[47] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on p. 21).

[48] A Yu Kitaev. “Fault-tolerant quantum computation by anyons”. In: *Annals of Physics* 303.1 (2003), pp. 2–30 (cit. on pp. 11, 14).

[49]A Yu Kitaev. “Quantum measurements and the Abelian stabilizer problem”. In: *arXiv preprint quant-ph/9511026* (1995) (cit. on p. 24).

[50]Jens Koch, M Yu Terri, Jay Gambetta, et al. “Charge-insensitive qubit design derived from the Cooper pair box”. In: *Physical Review A* 76.4 (2007), p. 042319 (cit. on pp. 12, 14).

[51]Nikita Kolganov, Sergey Mironov, and Andrey Morozov. “Large k topological quantum computer”. In: *arXiv preprint arXiv:2105.03980* (2021) (cit. on p. 11).

[52]Gerwin Koolstra, Ge Yang, and David I. Schuster. “Coupling a single electron on superfluid helium to a superconducting resonator”. In: *Nature Communications* 10.1 (2019) (cit. on p. 14).

[53]Y. LeCun, B. Boser, J. S. Denker, et al. “Backpropagation Applied to Handwritten Zip Code Recognition”. In: *Neural Computation* 1.4 (1989), pp. 541–551 (cit. on p. 21).

[54]Arjen Lenstra, Eran Tromer, Adi Shamir, et al. “Factoring Estimates for a 1024-Bit RSA Modulus”. In: *STACS 98*. STACS 98, 2003, pp. 55–74 (cit. on p. 26).

[55]Seth Lloyd. “Universal Quantum Simulators”. In: *Science* 273.5278 (1996), pp. 1073–1078. eprint: `https://www.science.org/doi/pdf/10.1126/science.273.5278.1073` (cit. on p. 24).

[56]Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. “Quantum principal component analysis”. In: *Nature Physics* 10.9 (2014), pp. 631–633 (cit. on p. 24).

[57]Lars S. Madsen, Fabian Laudenbach, Mohsen Falamarzi. Askarani, et al. “Quantum computational advantage with a programmable photonic processor”. In: *Nature* 606.7912 (2022), pp. 75–81 (cit. on p. 12).

[58]Alejandro Matos. “Overview of Prominent Decoherence Mechanisms in Superconducting Qubits”. In: () (cit. on p. 7).

[59]Warren S. McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (Dec. 1943), pp. 115–133 (cit. on p. 20).

[60]S. Mohajerani and P. Saeedi. “Cloud-Net: An End-To-End Cloud Detection Algorithm for Landsat 8 Imagery”. In: *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*. July 2019, pp. 1029–1032 (cit. on p. 31).

[61]Rui Nian, Jinfeng Liu, and Biao Huang. “A review On reinforcement learning: Introduction and applications in industrial process control”. In: *Computers Chemical Engineering* 139 (2020), p. 106886 (cit. on p. 3).

[62]Yu. A. Pashkin, O. Astafiev, T. Yamamoto, Y. Nakamura, and J. S. Tsai. “Josephson charge qubits: a brief review”. In: *Quantum Information Processing* 8.2-3 (2009), pp. 55–80 (cit. on p. 14).

[63]Elijah Pelofske, Andreas Bartschi, and Stephan Eidenbenz. “Quantum Volume in Practice: What Users Can Expect From NISQ Devices”. In: *IEEE Transactions on Quantum Engineering* 3 (2022), pp. 1–19 (cit. on p. 26).

[64]Robert Raussendorf, Jim Harrington, and Kovid Goyal. “Topological fault-tolerance in cluster state quantum computation”. In: *New Journal of Physics* 9.6 (2007), p. 199 (cit. on p. 8).

[65]Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. “Quantum Support Vector Machine for Big Data Classification”. In: *Physical Review Letters* 113.13 (2014) (cit. on p. 25).

[66]David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323.6088 (1986), pp. 533–536 (cit. on p. 35).

[67]O.-P. Saira, J. P. Groen, J. Cramer, et al. “Entanglement Genesis by Ancilla-Based Parity Measurement in 2D Circuit QED”. In: *Physical Review Letters* 112.7 (2014) (cit. on p. 15).

[68]A. L. Samuel. “Some Studies in Machine Learning Using the Game of Checkers”. In: *IBM Journal of Research and Development* 3.3 (1959), pp. 210–229 (cit. on p. 3).

[69]Thomas Schenkel, Walid Redjem, Arun Persaud, et al. “Exploration of Defect Dynamics and Color Center Qubit Synthesis with Pulsed Ion Beams”. In: *Quantum Beam Science* 6.1 (2022), p. 13 (cit. on p. 12).

[70]Artur Scherer, Benoît Valiron, Siun-Chuon Mau, et al. “Concrete resource analysis of the quantum linear-system algorithm used to compute the electromagnetic scattering cross section of a 2D target”. In: *Quantum Information Processing* 16.3 (Jan. 2017), p. 60 (cit. on p. 26).

[71]Benjamin Schumacher. “Quantum coding”. In: *Phys. Rev. A* 51 (4 Apr. 1995), pp. 2738–2747 (cit. on p. 3).

[72]Alessandro Sebastianelli, Daniela Alessandra Zaidenberg, Dario Spiller, Bertrand Le Saux, and Silvia Liberata Ullo. “On circuit-based hybrid quantum neural networks for remote sensing imagery classification”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 15 (2021), pp. 565–580 (cit. on pp. 23, 27, 29, 31, 33, 36, 41).

[73]Peter W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM Journal on Computing* 26.5 (1997), pp. 1484–1509 (cit. on p. 9).

[74]Amanpreet Singh, Narina Thakur, and Aakanksha Sharma. “A review of supervised machine learning algorithms”. In: *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. 2016, pp. 1310–1315 (cit. on p. 3).

[75]J.A.K. Suykens and J. Vandewalle. “None”. In: *Neural Processing Letters* 9.3 (1999), pp. 293–300 (cit. on p. 25).

[76]Yasunari Suzuki, Yoshiaki Kawase, Yuya Masumura, et al. “Qulacs: a fast and versatile quantum circuit simulator for research purpose”. In: *Quantum* 5 (2021), p. 559 (cit. on pp. 17–19).

[77]C Tahan, M Friesen, and R Joynt. “Decoherence of electron spin qubits in Si-based quantum computers”. In: *PHYSICAL REVIEW B* 66.3 (July 2002) (cit. on p. 3).

[78] Shuntaro Takeda and Akira Furusawa. “Toward large-scale fault-tolerant universal photonic quantum computing”. In: *APL Photonics* 4.6 (2019), p. 060902 (cit. on p. 12).

[79] Maarten Van den Nest, Wolfgang Dür, Akimasa Miyake, and Hans J Briegel. “Fundamentals of universality in one-way quantum computation”. In: *New Journal of Physics* 9.6 (2007), p. 204 (cit. on p. 11).

[80] Andrew Wack, Hanhee Paik, Ali Javadi-Abhari, et al. “Quality, Speed, and Scale: three key attributes to measure the performance of near-term quantum computers”. In: *arXiv preprint arXiv:2110.14108* (2021) (cit. on p. 9).

[81] Nathan Wiebe, Ashish Kapoor, and Krysta M Svore. “Quantum deep learning”. In: *arXiv preprint arXiv:1412.3489* (2014) (cit. on p. 25).

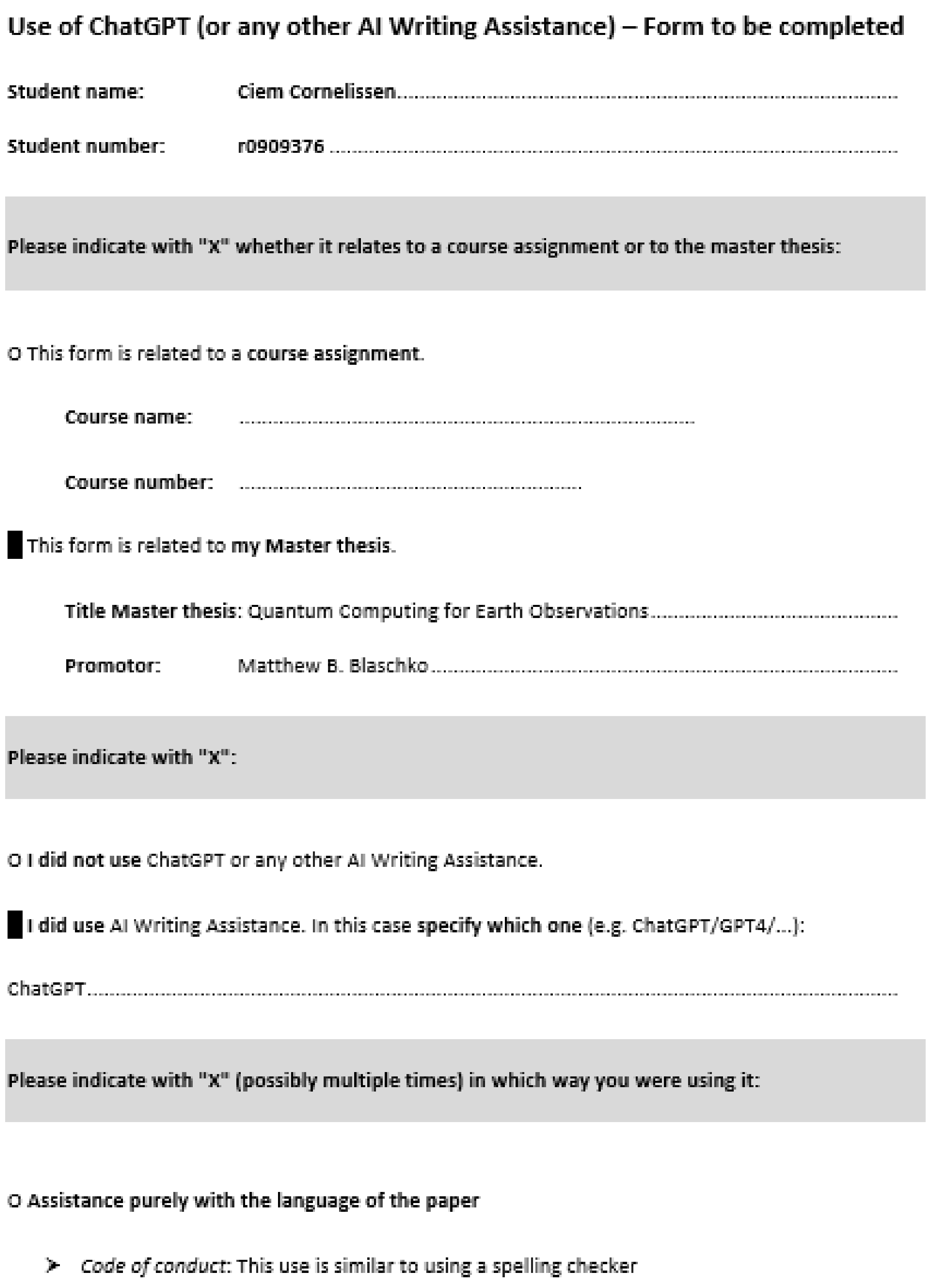
[82] Xiao Xue, Maximilian Russ, Nodar Samkharadze, et al. “Quantum logic with spin qubits crossing the surface code threshold”. In: *Nature* 601.7893 (2022), pp. 343–347 (cit. on p. 8).

[83] Junpei Yamaguchi, Masafumi Yamazaki, Akihiro Tabuchi, et al. *Estimation of Shor’s Circuit for 2048-bit Integers based on Quantum Simulator*. Cryptology ePrint Archive, Paper 2023/092. `https://eprint.iacr.org/2023/092`. 2023 (cit. on p. 26).

[84] Xiangli Yang, Zixing Song, Irwin King, and Zenglin Xu. *A Survey on Deep Semi-supervised Learning*. 2021. arXiv: `2103.00550 [cs.LG]` (cit. on p. 3).

[85] Sheir Yarkoni, Elena Raponi, Thomas Bäck, and Sebastian Schmitt. “Quantum annealing for industry applications: introduction and review”. In: *Reports on Progress in Physics* 85.10 (2022), p. 104001 (cit. on p. 12).

[86] Xue Ying. “An Overview of Overfitting and its Solutions”. In: *Journal of Physics: Conference Series* 1168 (2019), p. 022022 (cit. on p. 35).



Use of ChatGPT (or any other AI Writing Assistance) – Form to be completed

Student name:	Ciem Cornelissen...
Student number:	r0909376 ....

Please indicate with "X" whether it relates to a course assignment or to the master thesis:

O This form is related to a **course assignment**.

Course name: ....

Course number: ....

[✓ This form is related to my **Master thesis**.]

Title Master thesis: Quantum Computing for Earth Observations...

Promotor: Matthew B. Blaschko..

Please indicate with "X":

O **I did not use** ChatGPT or any other AI Writing Assistance.

[✓ **I did use** AI Writing Assistance. In this case **specify which one** (e.g. ChatGPT/GPT4/...):]

ChatGPT...

Please indicate with "X" (possibly multiple times) in which way you were using it:

O **Assistance purely with the language of the paper**

- *Code of conduct*: This use is similar to using a spelling checker

■ **As a search engine to learn on a particular topic**

- ➢ *Code of conduct*: This use is similar to e.g. a google search or checking Wikipedia. Be aware that the output of Chatbot evolves and may change over time.

O **For literature search**

- ➢ *Code of conduct*: This use is comparable to e.g. a google scholar search. However, be aware that ChatGPT may output no or wrong references. As a student you are responsible for further checking and verifying the absence or correctness of references.

O **For short-form input assistance**

- ➢ *Code of conduct*: This use is similar to e.g. google docs powered by generative language models

O **To let generate programming code**

- ➢ *Code of conduct*: Correctly mention the use of ChatGPT and cite it. You can also ask ChatGPT how to cite it.

O **To let generate new research ideas**

- ➢ *Code of conduct*: Further verify in this case whether the idea is novel or not. It is likely that it is related to existing work, which should be referenced then.

O **To let generate blocks of text**

- ➢ *Code of conduct*: Inserting blocks of text without quotes from ChatGPT to your report or thesis is not allowed. According to Article 84 of the exam regulations in evaluating your work one should be able to correctly judge on your own knowledge. In case it is really needed to insert a block of text from ChatGPT, mention it as a citation by using quotes. But this should be kept to an absolute minimum.

O **Other**

- ➢ *Code of conduct*: Contact the professor of the course or the promotor of the thesis. Inform also the program director. Motivate how you comply with Article 84 of the exam regulations. Explain the use and the added value of ChatGPT or other AI tool: ....

Further important guidelines and remarks

- ChatGPT cannot be used related **to data or subjects under NDA agreement.**
- ChatGPT cannot be used related **to sensitive or personal data due to privacy issues**.
- **Take a scientific and critical attitude** when interacting with ChatGPT and interpreting its output. Don't become emotionally connected to AI tools.
- As a student you are responsible to comply with Article 84 of the exam regulations: your report or thesis should reflect your own knowledge. Be aware that plagiarism rules also apply to the use of ChatGPT or any other AI tools.
- **Exam regulations Article 84**: "Every conduct individual students display with which they (partially) inhibit or attempt to inhibit a correct judgement of their own knowledge, understanding and/or skills or those of other students, is considered an irregularity which may result in a suitable penalty. A special type of irregularity is plagiarism, i.e. copying the work (ideas, texts, structures, designs, images, plans, codes , ...) of others or prior personal work in an exact or slightly modified way without adequately acknowledging the sources. Every possession of prohibited resources during an examination (see article 65) is considered an irregularity."
- **ChatGPT suggestion about citation**: "Citing and referencing ChatGPT output is essential to maintain academic integrity and avoid plagiarism. Here are some guidelines on how to correctly cite and reference ChatGPT in your Master's thesis: 1. Citing ChatGPT: Whenever you use a direct quote or paraphrase from ChatGPT, you should include an in-text citation that indicates the source. For example: (ChatGPT, 2023). 2. Referencing ChatGPT: In the reference list at the end of your thesis, you should include a full citation for ChatGPT. This should include the title of the AI language model, the year it was published or trained, the name of the institution or organization that developed it, and the URL or DOI (if available). For example: OpenAI. (2021). GPT-3 Language Model. https://openai.com/blog/gpt-3-apps/ 3. Describing the use of ChatGPT: You may also want to describe how you used ChatGPT in your research methodology section. This could include details on how you accessed ChatGPT, the specific parameters you used, and any other relevant information related to your use of the AI language model. Remember, it is important to adhere to your institution's specific guidelines for citing and referencing sources in your Master's thesis. If you are unsure about how to correctly cite and reference ChatGPT or any other source, consult with your thesis advisor or a librarian for guidance."

Additional reading

ACL 2023 Policy on AI Writing Assistance: https://2023.aclweb.org/blog/ACL-2023-policy/

KU Leuven guidelines on citing and referencing Generative AI tools, and other information:https://www.kuleuven.be/english/education/student/educational-tools/generative-artificial-intelligence

Dit formulier werd opgesteld voor studenten in de Master of Artificial intelligence. Ze bevat een code of conduct, die we bij universiteitsbrede communicatie rond onderwijs verder wensen te hanteren.

Deze template samen met de code of conduct zal in de toekomst nog verdere aanpassingen behoeven. Het schept alvast een kader voor de 2de en de 3de examenperiode van 2022-2023.