Karlsruhe Institute of Technology
Communications Engineering Lab
Prof. Dr.-Ing. Laurent Schmalen

Quantum Computing Assisted Speech Processing

Masterthesis

Melvin Strobl

Advisor	:	Prof. Dr.-Ing. Laurent Schmalen
Supervisors	:	Prof. Dr.-Ing. Achim Streit
		Dr.-Ing. Eileen Kühn

Start date	:	01.05.2021
End date	:	02.11.2021

Declaration

With this statement I declare that I have independently completed the above master's thesis. The thoughts taken directly or indirectly from external sources are properly marked as such. This thesis was not previously submitted to another academic institution and has also not yet been published.

Karlsruhe, 02.11.2021

Melvin Strobl

„Computer programming is an art form, like the creation of poetry or music.“

(Donald Knuth)

Abstract

Human-machine interaction in general and speech processing (SP) in particular are key-disciplines in today's consumer electronics. While the computational power of smart devices heavily increased within the last couple of years, tasks such as speech recognition (SR) still mainly rely on cloud based solutions. In such architectures, not only high accuracy, but also quick response time and fast inference are essential for providing a pleasant user experience. State of the art approaches use machine learning (ML) based solutions for recognition of spoken words making use of high-performance hardware and huge datasets. Besides the actual training and inference of such ML models, SR requires the extraction of acoustic features from the recorded speech. Spectrograms have proven to be a well-suited feature space and are broadly being used in many ML based frameworks.

An application of quantum computing (QC) in a SR pipeline was previously proposed in [YQC+20b] where a neural network (NN) trained on quantum manipulated spectrogram slightly surpassed the validation accuracy of the classical approach.

However, quantum computers are foremost known for their potential speedup in certain applications and given that the quantum Fourier transform (QFT), the quantum equivalent of the Fourier transform (FT), is just such an algorithm, the natural question and thus topic of this thesis arises, if there are options or preferably advantages, utilizing the QFT for spectrogram generation.

This requires construction of an adequate framework where a short time quantum Fourier transform (STQFT) is developed, optimization of the STQFT and post processing involving noise mitigation applied and finally evaluated based on the performance of the NN. Since speech synthesis (SS), as another subcategory of SP, requires a completely different framework and comes with another set of challenges, although many of the insights of SR can be transferred, this thesis focuses exclusively on SR.

By using a modular approach, different signal types as well as transformations can be quickly interchanged and tested either in simulation or on real quantum computers. For evaluating the performance of the NN given features from variant STQFT configurations, the architecture proposed in [YQC+20b] is used as a starting point and their accuracy of 95.12 % as a baseline value.

Experiments show that although quantum computers in the noisy intermediate scale quantum (NISQ) era are capable of processing the QFT of band-limited harmonic oscillations, waiting queues and the restricted access to complex quantum systems forbid an application of more complex QFTs necessary to meet the requirements for the sampling frequency of speech signals w.r.t. time and frequency resolution in a real SR application.

However, by using a simulation environment and the noise model of a real quantum device, the spectrograms generated with the STQFT enables the NN to achieve a competitive test accuracy of 89.92 %, whilst forfeit the speedup gained on real devices. Although the accuracy did not surpass the baseline and noise and circuit capacity of NISQ devices restricts the applicability of SR with quantum advantage, the results motivate further investigations in practical applications of the QFT and STQFT for speech processing.

Zusammenfassung

Mensch-Maschine-Interaktion im Allgemeinen und Sprachverarbeitung im Besonderen sind Schlüsseldisziplinen in der heutigen Unterhaltungselektronik. Obwohl die Rechenleistung mobiler Geräte in den letzten Jahren stark zugenommen hat, sind Aufgaben wie Spracherkennung immernoch hauptsächlich auf cloudbasierte Lösungen angewiesen. Bei solchen Architekturen is nicht nur eine hohe Genauigkeit, sondern auch eine schnelle Reaktionszeit für eine reale und nutzerfreundliche Anwendung unerlässlich. Moderne Ansätze verwenden maschinelles Lernen für die Erkennung der Sprache, die hoch performante Hardware und umfassende Datensätze erfordert. Neben dem eigentlichen Training und der Inferenz solcher Modelle für das maschinelle Lernen erfordert Spracherkennung die Extraktion von akustischen Merkmalen aus der aufgenommenen Sprache. Spektrogramme haben sich hierbei als gut geeigneter Merkmalsraum erwiesen und sich in heutigen Systemen etabliert.

Eine Anwendung von Quantencomputern in der Spracherkennung wurde zuvor in der Arbeit von [YQC+20b] vorgeschlagen, in welcher ein Neuronales Netzes, das auf mittels von einem Quantencomputers manipulierten Spektrogrammen trainiert wurde, die Validierungsgenauigkeit des klassischen Ansatzes übertraf. Quantencomputer sind jedoch vor allem für ihre Überlegenheit gegenüber klassischen Computern im Berechnen bestimmter Algorithmen bekannt. Da die Quanten-Fourier-Transformation, das Äquivalent der klassischen Fourier-Transformation auf einem Quantencomputer, ein solcher Algorithmus ist, stellt sich die natürliche Frage und somit das Thema dieser Arbeit, ob es Möglichkeiten oder sogar Vorteile gibt, die Quanten-Fourier-Transformation für die Spektrogrammerzeugung zu nutzen.

Die Untersuchung dieser Frage erfordert den Aufbau eines geeigneten Frameworks, in dem eine kurzzeit-Quanten-Fourier-Transformation entwickelt, optimiert und ggf. Rauschunterdrückung angewandt wird. Anschließend wird die Genauigkeit eines Neuronalen Netzes, trainiert auf den mittels der kurzzeit-Quanten-Fourier-Transformation erzeugten Merkmalen, evaluiert und diskutiert. Da die Sprachsynthese, als eine weitere Unterkategorie der Sprachverarbeitung, ein völlig anderes Framework erfordert und ein ganzes Set an weiteren Herausfoderungen beherbergt, wenngleich viele aus der Spracherkennung gewonnenen Erkenntnisse darin übertragen werden können, konzentriert sich diese Arbeit ausschließlich auf die Spracherkennung.

Durch die Verwendung eines modularen Ansatzes können verschiedene Signaltypen sowie Transformationen schnell ausgetauscht und entweder in der Simulation oder auf realen Quantencomputern getestet werden. Für die Bewertung der Genauigkeit des Neuronalen Netzwerks, gegebenen den Merkmale aus verschiedenen Konfigurationen der kurzzeit-Quanten-Fourier-Transformation, wird die in [YQC+20b] vorgeschlagene Architektur als Ausgangspunkt verwendet und mit ihrer Genauigkeit von 95.12 % als Referenzwert verglichen.

Experimente zeigen, dass Quantencomputer der “Noisy Intermediate Scale Quantum”-Ära zwar in der Lage sind, die Quanten-Fourier-Transformation von stark bandbegrenzten harmonischen Schwingungen zu verarbeiten. Jedoch verbietet der beschränkte Zugang zu komplexeren Quantencomputern, die notwendig sind um den Anforderungen an die Abtastfrequenz von Sprachsignalen in Bezug auf Zeit- und Frequenzauflösung zu erfüllen,

eine Anwendung in praktischen Spracherkennungsszenarien.

Durch die Verwendung einer Simulationsumgebung mit dem Rauschmodell eines Quantencomputers in Kombination mit den in dieser Arbeit entwickelten Ansätze, ermöglicht das mit dem kurzzeit-Quanten-Fourier-Transformation erzeugte Spektrogramm dem Neuronalen Netzwerk eine Testgenauigkeit von 89.92 %, während jedoch die auf realen Geräten potentielle Geschwindigkeitssteigerung verloren geht. Obwohl die Genauigkeit nicht über der Referenz liegt und das Rauschen und die Kapazität von "Noisy Intermediate Scale Quantum"-Geräten die Anwendbarkeit von Spracherkennung mit Quantenvorteil einschränkt, motivieren die Ergebnisse zu weiteren Untersuchungen in praktischen Anwendungen der Quanten-Fourier-Transformation für die Sprachverarbeitung.

Contents

1. Introduction **3**

2. Foundations **5**
2.1. Speech Recognition . . . 5
2.1.1. Signal Processing . . . 5
2.1.2. Speech Recognition Neural Networks . . . 10
2.2. Quantum Computing . . . 14
2.2.1. Qubits and Qubit States . . . 14
2.2.2. Operations . . . 18
2.2.3. Measurement . . . 22
2.2.4. Entanglement . . . 23
2.2.5. Realization of Circuits . . . 25
2.2.6. Noise . . . 25
2.2.7. Encoding . . . 33
2.2.8. Quantum Fourier Transform . . . 33
2.2.9. Quantum Machine Learning . . . 39
2.3. Related Research . . . 39

3. Approach **43**
3.1. Quantum Fourier Transform for Signal Processing . . . 43
3.1.1. Encoding . . . 43
3.1.2. Transformation Accuracy . . . 43
3.1.3. Noise . . . 45
3.2. Short Time Quantum Fourier Transform . . . 47
3.3. Hybrid Quantum Speech Processing . . . 48

4. Validation **51**
4.1. Signal Generation and Data Acquisition . . . 53
4.2. Transformations . . . 54
4.2.1. Discrete Fourier Transform . . . 55
4.2.2. Fast Fourier Transform . . . 55
4.2.3. Quantum Fourier Transform . . . 55
4.3. Short Time Quantum Fourier Transform . . . 62
4.3.1. Noise Mitigation . . . 63
4.3.2. Angular Filter . . . 65
4.3.3. Speech Signals . . . 66
4.4. Connection to the Neural Network . . . 71
4.5. Discussion . . . 77

5. Conclusion **81**

List of Figures **83**

Symbols **87**

A. Additional Figures **89**

B. Abbreviations **95**

Bibliography **97**

1. Introduction

Cloud-based speech processing solutions, incorporating speech recognition and speech synthesis, have become an essential part of consumer goods and entertainment products in recent years. This has been made possible by continuous improvements in processing hardware, but also by advances in machine learning research. machine learning-based systems require a significant amount of data to solve a given learning problem. In the case of speech recognition, for example, a dataset contains speech samples and their associated labels, which an ML model like a neural network is then supposed to classify correctly.

Choosing a characteristic data representation is crucial for the success of the machine learning task. The *frequency composition* or more precisely, the *spectrogram* of a speech signal has been shown to meet this requirement and can be obtained from the time domain of the speech signal by a Fourier transform.

Driven by the importance of Fourier transform for feature generation in speech processing tasks, this work investigates the applicability of the quantum equivalent, the quantum Fourier transform, in this area. Quantum computing is a relatively new technology that differs from classical computational hardware in that quantum mechanical effects are used to build basic computational logic. More complex systems are then built by combining these logic blocks. Furthermore, quantum algorithms can achieve a computational advantage over classical equivalents, as they solve certain complex problems more efficiently. In combination with machine learning-based approaches, quantum computing promises attractive possibilities in terms of data processing and data generation.

The quantum Fourier transform, as an algorithm representative for the potential speedup gained through appliance of quantum computing is evaluated as an option for speech recognition within the scope of this thesis. While speech processing encompasses both speech recognition and speech synthesis, this work focuses primarily on speech recognition, as many of the lessons learned in this area can also be applied to speech synthesis, but would require a completely different framework. This, in turn, is considered as a subject of separate investigation.

Since quantum computing hardware is still in early development, noise and errors are inherent with calculations on such machines. Furthermore, the number of quantum bits, the quantum equivalent of classical bits, is not sufficiently large to allow for applications of quantum computing comparable to those on classical systems. The small number of available quantum bits and error prone computations led to the coining of the noisy intermediate scale quantum term commonly used to describe nowadays quantum hardware. These circumstances only allow the evaluation of the quantum Fourier transform for speech recognition in simulation environments although less demanding type of signals will also be processed on real quantum devices.

This thesis covers the implementation and integration of the quantum Fourier transform in a common speech recognition architecture. Furthermore the option of noise mitigation, as an important aspect of quantum computing in the noisy intermediate scale quantum era, post processing and performance investigations in union with a neural network architecture are evaluated.

The foundations of speech recognition including signal theory and neural network architecture are covered in chapter 2. Furthermore, basic concepts of quantum computing are introduced, advanced topics such as the quantum Fourier transform are presented and potential advantages discussed. Finally, related research is presented in chapter 2.3.

The main concepts and ideas of this work are introduced in chapter 3, where short time quantum Fourier transform is also introduced as an approach to generate spectrograms with quantum computing. Since quantum computing is subject to noise in the noisy intermediate scale quantum era, additional methods to reduce these noise effects on the measurement results are explained in chapter 3 as well.

Chapter 4 then shows the suitability of such methods in combination with the neural network architecture presented in 2.3. Furthermore, the framework used and proposed in this thesis to evaluate the performance of quantum Fourier transform and short time quantum Fourier transform is explained in detail and discussed at the end.

In the final chapter 5, the future perspectives of quantum computing in speech recognition are discussed, important results summarized and possible research topics derived.

2. Foundations

In recent years, quantum computing (QC) evolved from a theoretical concept, introduced by [Fey82], to a rapidly growing research field, sparking many new application ideas and gaining broad interest. QC promises a potential computational speedup of certain algorithms compared to their classical counterpart, but although Quantum supremacy has been proven [AAB+19], nowadays noisy intermediate scale quantum (NISQ) hardware limits the possibilities of bringing this advantage to application. Nevertheless hardware is already capable of solving certain algorithms more efficiently than classical computers, but noise and limited quantum resources restrict the exploitation of the theoretical potential of QC.

As mentioned in the introduction 1, speech recognition (SR) will be the primary focus in this thesis since many concepts can be applied on speech synthesis (SS) as well, but require a different framework. This chapter covers common basics and approaches in SR tasks and introduces target-oriented foundations of quantum computing and related research used within this work.

2.1. Speech Recognition

SR describes the task of automatic conversion of speech into a sequence of words [HD10]. This field of research has gained enormous popularity in recent years and advances in neural network (NN) gave rise to almost natural conversations between human beings and computers [FTYA21]. Despite the increasing computing power of mobile devices, SR is still mostly run in the cloud using machine learning (ML) techniques, enabling continuous improvements and a consistent user experience across all devices. Fundamental to almost all SR methods is the signal processing for input preparation and feature generation. The following sections cover the foundations of signal processing required for appliance in SR tasks.

2.1.1. Signal Processing

In this section, mathematical methods from the field of signal processing relevant to speech recognition tasks are presented.

Sampling

Digital information processing of a continuous speech $x(t)$ over time t, $t \in \mathbb{R}$ necessitates sampling of the latter. Mathematically, this can be expressed by the convolution of a continuous signal with a *Dirac* comb $\delta(t)$ [FPL19]

$$x_* = x(t) * \delta(nt_s) = \sum_{n=-\infty}^{\infty} x_n \delta(t - nt_s) \ , \qquad n \in \mathbb{Z} \tag{2.1}$$

where x_* is the sampled signal and x_n describes a single sample with frequency $f_s = \frac{1}{t_s}$ at index n. The signal is represented with 0 in between sampling points.

The minimum sampling frequency depends on the spectral components contained in the signal by the definition of the *sampling theorem* [FPL19]:

Theorem 1 *Let $x(t)$ be a band-limited signal such that there is no frequency information for $|f| \geq B$ and $B \geq 0$, then the signal $x(t)$ can be reconstructed from its sampled variant $x_n = x(nt_s)$ without any loss in information if $t_s \leq \frac{1}{2B}$ and therefore $f_s \geq 2B$ holds true.*

As a consequence, if spectral information of a speech signal up to e.g. $f_{\max} = 8$ kHz should be extracted, then the signal needs to be band-limited to 8 kHz and the sampling frequency must be $f_s \geq 16$ kHz. A signal is said to be within the *Nyquist Band* [Sha98] if it fulfills above condition or in general:

$$f_{\max} < B \leq \frac{f_s}{2} \quad \rightarrow \quad X(f) = 0 \quad \forall \quad |f| \geq B \tag{2.2}$$

where B is called the *bandwidth* of a signal and $X(f)$ the fourier transform of a signal $x(t)$ covered in the following section 2.1.1.

Fourier Transform

For applications in SR, the Fourier transform (FT) defined by [FPL19]

$$X(f) = \int_{-\infty}^{\infty} x(t) \mathrm{e}^{-\imath 2\pi f t} \mathrm{d}t \tag{2.3}$$

with $X(f)$ being the spectrum of $x(t)$ needs to be adapted to the discrete case such that

$$X_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n \mathrm{e}^{-\imath 2\pi \frac{kn}{N}} \tag{2.4}$$

where N denotes the total number of samples and $k \in [0 \dots K-1]$ with $K \in \mathbb{N}$ indicates the frequency bin resulting from the sampled signal x_n. K indicates the number of frequencies being considered and is conveniently chosen $K = N$ [Mü21].

The equation above consists of N complex multiplications for each single frequency bin out of $K = N$ bins in total. Therefore the complexity of this algorithm is $\mathcal{O}(N^2)$. By substituting $\omega_N := \mathrm{e}^{-\imath 2\pi \frac{1}{N}}$ the formula becomes:

$$X_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n \omega_N^{kn} \tag{2.5}$$

By using the vector representation of x_n, X_k can be calculated using a matrix equivalent to the discrete Fourier transform (DFT) as follows:

$$\begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_k \end{pmatrix} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_N^1 & \omega_N^2 & \cdots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \omega_N^4 & \cdots & \omega_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \cdots & \omega_N^{(N-1)(N-1)} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \tag{2.6}$$

where the DFT matrix F is $\in \mathbb{C}^{N \times N}$ and $N = 2^n$, thus $X = Fx$ in short.

Leakage Effect Theorem 1 inherently requires a band-limited signal to avoid *leakage effect*. This effect arises from the fact that real signals are always time-finite thus never band-limited due to the implicit filtering with a rectangular window function (rect) the time domain. This in turn corresponds to a convolution with a sinc function in the frequency domain [FPL19] as indicated in the following equation.

$$\mathcal{F}(\mathsf{rect}_T(t)) = T\mathsf{sinc}(fT) = T\frac{\sin(\pi fT)}{\pi fT}, \quad t, f \in \mathbb{R} \tag{2.7}$$

The rect function is defined by

$$\mathsf{rect}_T(t) = \begin{cases} 1 & \forall \quad |t| \leq \frac{T}{2} \\ 0 & \forall \quad |t| > \frac{T}{2} \end{cases} \qquad T \in \mathbb{R} \tag{2.8}$$

with T being the period length of a signal.

Due to the properties of the sinc function, this leads to an infinitely wide frequency band. This disallows the Nyquist theorem 1 to be fulfilled and finally results in *smearing* of the actual signal spectrum [FPL19]. Nonetheless, the leakage effect can be efficiently suppressed even with a finite observation window, if the oscillating frequency of the signal is an integer multiple of the frequency resolution $\Delta f = \frac{f_s}{N}$ [FPL19].

Fast Fourier Transform [CT65] developed the fast Fourier transform (FFT) algorithm enabling efficient application of the FT in digital devices. Their algorithm is based on the idea that calculating half of the points of a DFT requires only a quarter of computations since $\mathcal{O}((\frac{N}{2})^2) = \mathcal{O}(N^2)/4$.

Using the representation in equation (2.6), this idea comes clear when writing down a 8-point DFT matrix and omitting half of the values in each direction thus resulting in a 4-point DFT:

$$F = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & o & o & o & o \\ 1 & \omega & \omega_N^2 & \omega_N^3 & o & o & o & o \\ 1 & \omega_N^2 & \omega_N^4 & \omega_N^6 & o & o & o & o \\ 1 & \omega_N^3 & \omega_N^6 & \omega_N^9 & o & o & o & o \\ o & o & o & o & o & o & o & o \\ o & o & o & o & o & o & o & o \\ o & o & o & o & o & o & o & o \\ o & o & o & o & o & o & o & o \end{bmatrix} \tag{2.9}$$

Therefore, the signal $x(n)$ is being split up into odd and even parts recursively:

$$\begin{aligned} X_k &= \sum_{r=0}^{\frac{N}{2}-1} x_{2r} \mathrm{e}^{-\imath 2\pi \frac{k2r}{N}} + \sum_{r=0}^{\frac{N}{2}-1} x_{2r+1} \mathrm{e}^{-\imath 2\pi \frac{k(2r+1)}{N}} \\ X_k &= \sum_{r=0}^{\frac{N}{2}-1} x_{2r} \mathrm{e}^{-\imath 2\pi \frac{k2r}{N}} + \mathrm{e}^{\frac{-\imath 2\pi k}{N}} \sum_{r=0}^{\frac{N}{2}-1} x_{2r+1} \mathrm{e}^{-\imath 2\pi \frac{k2r}{N}} \\ X_k &= \sum_{r=0}^{\frac{N}{2}-1} x_{2r} \mathrm{e}^{-\imath 2\pi \frac{kr}{N/2}} + \mathrm{e}^{\frac{-\imath 2\pi k}{N}} \sum_{r=0}^{\frac{N}{2}-1} x_{2r+1} \mathrm{e}^{-\imath 2\pi \frac{kr}{N/2}} \\ X_k &= X_{\text{even}} + \mathrm{e}^{-i2\pi \frac{kn}{N}} X_{\text{odd}} \end{aligned} \tag{2.10}$$

A single split therefore yields a remaining complexity of $\mathcal{O}\frac{N^2}{2} + N$, consisting of 2 DFTs on $({}^{N}/_{2})^2$ elements and additional summation of N parts afterwards.

This iterative approach can be repeated $p = \log_2 N$ times (assuming N being a power of 2) and finally yielding

$$\begin{aligned} \frac{N}{2^p} &\to \frac{N^2}{2^p} + Np \\ &= \frac{N^2}{N} + N \log_2 N \\ &= N + N \log_2 N \end{aligned} \tag{2.11}$$

which results in overall $\mathcal{O}(N \log_2 N)$. The assumption that N is a power of 2, might not always be fulfilled given a fixed signal. For this reason, *Zero Padding* is commonly used to match the signal length.

Short Time Fourier Transform

In SR, one is usually interested in the frequency content of a signal over time. The Short Time Fourier transform (STFT) is a common method for computing such a spectrogram, approaching the frequency change in the input signal by multiple subsequent FT applied to windowed cut-outs. Ideally, a rectangular window (filter) would be applied, such that

neighboring signal components cannot influence the transformation result. The definition of the STFT using a filter γ applied on a continuous signal $x(t)$ is defined by [FPL19]

$$F_x^\gamma(\tau, f) = \mathcal{F}_t \{x(t)\gamma^*(t-\tau)\} = \int_{-\infty}^{\infty} x(t)\gamma^*(t-\tau)\mathrm{e}^{-\imath 2\pi f t}\ \mathrm{d}t \tag{2.12}$$

whereas the spectrogram is calculated by the square of the absolute value of the STFT:

$$S_x^\gamma(\tau, f) = |F_x^\gamma(\tau, f)|^2 = \left| \int_{-\infty}^{\infty} x(t)\gamma^*(t-\tau)\mathrm{e}^{-\imath 2\pi f t}\ \mathrm{d}t \right|^2 \tag{2.13}$$

In the case of discrete time and frequency, this becomes:

$$S_x^\gamma(m, k) = \sum_{n=-\infty}^{\infty} x_n \gamma_{n-m} \mathrm{e}^{-\imath 2\pi \frac{k}{W} n} \tag{2.14}$$

where $m \in [0 \dots M]$ with $M := \left\lfloor \frac{N-W}{H} \right\rfloor$ is the shift of the discrete window function γ with length $W \in \mathbb{N}$ applied on x with length $N \in \mathbb{N}$ and $k \in [0 \dots \frac{N}{2}]$ indicates the frequency bin of the m-th time frame. H refers to the hop size describing the distance by which the window is shifted over x.

Windowing Functions The STFT comes with the issue of finding a trade-off between time and frequency resolution. This can be seen by considering the edge cases of the number of samples used in the FT:

- $n \to 1$ (one sample of the signal): no frequency information, but maximum time resolution
- $n \to N$ (all samples of the signal): maximum frequency resolution, but minimum time resolution

Additionally, in case of the STFT, where time limited signals are explicitly desired, the Nyquist theorem 1 can only be approximately fulfilled, thus leading to the leakage effect as explained in paragraph 2.1.1

For this reason, windowing functions with a *softer* shape than the rectangular filter to reduce the impact of the leakage effect. Some of these filters and their properties from [GP21] are:

- Kaiser: lowest artefacts of filtering among all filters
- Blackman: slightly worse than a kaiser window
- Hamming: optimized to minimize the nearest side lobe
- Hanning: removes the discontinuities at the beginning and end of the sampled signal
- Bartlett: similar to triangular window, but with less ripple

Usually, the *Hanning* window provides a sufficient frequency resolution with small computational effort. However, the exact differences are not relevant to the subject of this thesis thus are not regarded any further.

Mel-Spectrogram

For the reason that human beings do not perceive a linear distance between distinct frequencies [SVN37], it is worth reconsidering the suitability of the signal representation introduced with the STFT from section 2.1.1.

[Sha87] therefore proposes the *Mel-scale* which represents the frequencies on a quasi-logarithmic scale defined by

$$f_{\text{mel}} = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{2.15}$$

where the constants in this term are given by definition and result from evaluations of the perceptual distance between tones perceived by human hearings. Equation (2.15) then maps a frequency f to its corresponding Mel-frequency f_{mel}.

The Mel-scale is usually applied after calculating the spectrogram using a filterbank (matrix) mapping parameterized by the number of desired Mel-bins and the sampling rate given a specific spectrogram. Example 2.1.1 provides such an calculation on the basis of typical values of a speech signal.

Example 2.1.1 *Converting a* 1024*-point STFT with a matrix size of e.g.* (512×1024) *into a* 60*-bin Mel-spectrogram, would yield a* (512×60) *matrix.*

2.1.2. Speech Recognition Neural Networks

While the field of research around NNs has gained enormous popularity within the last several decades, this section aims to cover only the necessary basics and specific advanced topics mandatory for the application of NNs within this thesis.

Convolutional Neural Networks

In a convolutional neural network (CNN), instead of matrix multiplications, convolutions are applied at least once per layer on $(n \times n \times c)$ -dimensional input data [GBC16] with $n, c \in \mathbb{Z}$. The following discussion will focus on the case of one channel $c = 1$ for simplicity, thus $(n \times n \times 1)$ is abbreviated with $(n \times n)$.

This works by shifting a $(k \times k), \quad k \in \mathbb{Z}$ matrix (filter) W consisting of k^2 weights $(w_0 \dots w_k, \quad w_i \in \mathbb{R})$ over an input matrix of size $(n \times n)$, summing up the pointwise multiplied entries and thus retrieving the value o_i of the output matrix. Naturally, this resulting matrix size would decrease, if $n \geq 1$ which is usually addressed by introducing zero-padding around the border of the input matrix. Furthermore, the distance, by which the filter matrix shifts over the input can be controlled by the stride [DV16].

A general expression for the resulting matrix size $(o \times o)$ after applying a filter of size $(k \times k)$ on an input of size $(n \times n)$ is stated in the following equation derived from [DV16]:

$$o = \left\lfloor \frac{n + 2 * p - k}{s} \right\rfloor + 1 \tag{2.16}$$

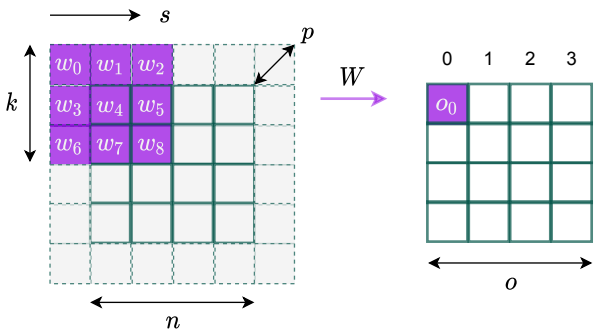


Figure 2.1: Convolution of a $(k \times k)$ $(k = 3)$ filter kernel over a $(n \times 5)$ $(n = 5)$ input padded with $p = 1$ border of zeros padded around the image and unit stride $s = 1$ resulting in an output image of size $o \times o$ $(o = 4)$

where p denotes the padding, s the stride and k the filter size as illustrated in figure 2.1.

Applying c' filters with distinct weights but equal shape will yield a resulting output depth of c', also referred to as *channels*. In conjunction with equation (2.16) the overall output then becomes $(o \times o \times c')$. The filter values (weights) stay constant within a single channel thus enable *location invariance* of a CNN but can vary across different channels. This yields the more general dimensionality of the filter $(k \times k \times c)$.

Optimizers In general, stochastic gradient descent (SGD) is used in a backpropagation [GBC16] manner to update the weights. However, evaluating a single training example and then updating all weights accordingly is computationally very expensive and only gives a stochastic approximation of the true gradient.

The batch gradient descent (BGD) approach, often just referred to as gradient descent (GD), is on the other side of the extreme were all training samples are evaluated and the mean gradient is then used to update the parameters of the network. It is conceivable that although this method results in a smooth loss curve, it is not suited well for large datasets since one single gradient step would require the NN output calculation of all samples [GBC16]. Furthermore BGD could converge in local minimums which is less likely to happen with the high frequent and stochastic gradient updates in SGD [Sra12].

For this reason, *mini batch GD* is commonly used, where multiple training samples (mini batch) specified by the *batch size*, are passed through the NN and contribute to the loss together. Afterwards, backpropagation is applied once for those samples [GBC16]. This approach requires less memory than BGD while resulting in a smoother loss curve compared to SGD. Increasing the batch size is therefore an approach to address strong fluctuations in the training history [Sra12]. A full iteration of all samples, independently of the number of batches required, is referred to as an *epoch*. Both, the batch size and the number of epochs are hyperparameters whose typical values vary depending on the NN architecture and the learning problem (LP).

The *Adam* [KB17] optimizer provides an additional approach to tackle the problem of noisy or sparse gradients by considering the first and second order moments of the gradient. This stochastic gradient-based optimization can stabilize convergence and also improves speed and memory consumption compared to BGD [KB17]. Despite the learning rate is being adapted based on the first and second order moments of the gradient, Adam requires a step-size hyperparameter α as well as exponential decay rates β_1 and β_2 for both moment estimates. Decreasing the learning rate also reduces the gradients and can

therefore, depending on the learning problem, help to reduce fluctuations in the loss curve. However, a reduction of the learning rate also reduces the overall training progress per epoch which is why usually the number of epochs need to be increased simultaneously.

Dataset Splitting Commonly, a given dataset for a ML task is split into three subsets [Rus15]. The *training-dataset* is used in the learning process to fit a model (i.e. adjust weights and biases in case of a NN). Hyperparameters of the network are then adjusted based on the performance of the model on the validation set. With each adjustment of the hyperparameters, the *validation-dataset* weakens the unbiased evaluation property. Therefore, after the training and validation accuracy is sufficient, a final completely unbiased evaluation of the network on the unseen *test-dataset* is conducted. Splitting the dataset into these three subsets (70 % $\rightarrow$ training-, 20 % $\rightarrow$ validation- and 10 % $\rightarrow$ test-dataset) is a common approach in ML to ensure realistic final performance measurement [Rus15].

Over- and Underfitting Given a dataset with weak expressiveness and a network architecture with high capacity, the NN would likely tends to *overfit*. This negative example describes a common issue in ML where the network does not solve the LP by gaining assumptions necessary for generalization. Instead of learning this *inductive bias* [Mit07], it memorizes the dataset which leads to a weak performance on previously unseen data as in the validation- or test-dataset. Besides architectural changes, regularization techniques such as *dropout* [SHK+14] or *batch normalization* [IS15] can be applied. Contrary, *underfitting* occurs, when the capacity of a NN is not sufficiently large to capture the complexity of a LP.

Recurrent Neural Networks

If the input matrix contains temporal information, plain CNNs cannot always meet the requirements, though exceptions exist [Wai89]. This problem can be solved by using recurrent neural networks (RNNs) as they are specialized for processing sequential data $x = [x^{(1)} \dots x^{(n)}]$ [GBC16], about which a CNN can hardly generalize.

Figure 2.2 shows the basic structure of a vanilla RNN. The input data, denoted by x is fed into the network using a weight matrix U. The hidden layer then stores the state across several iterations while learning the inherent structure of x in W which in turn are updated using backpropagation. The output vector o is attained by applying a third matrix V on the hidden state. This result, together with the label vector y then contributes to the loss.

From the figure 2.2 it is conceivable, that with increasing length of the sequential data, number of matrix multiplications also increase. This in turn often leads to the *vanishing gradient problem* where updates of the weight matrices are effectively prevented by small gradients. In such a case, the network is unable to continue learning [KK01]. Similar can happen, if the gradients become very large, thus causing heavy fluctuations weights as well as in the learning curve and therefore preventing the network from learning. This effect is referred to as *exploding gradients*. These effects particularly occur in large RNN architectures, which however are required to model *long-term* dependencies in the input data [KK01].

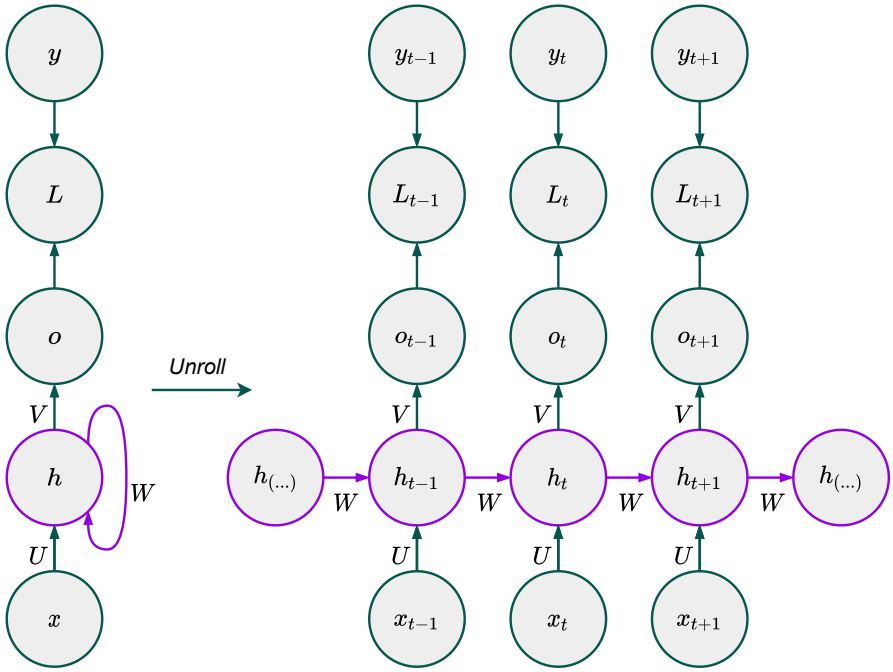


Figure 2.2: Schematic of a vanillaRNNarchitecture derived from [GBC16]. Input data x is learned across several iterations in W, resulting in a time dependent output vector o. Comparison with the label vector y then contributes to the loss function L.

Long Short-Term Memory Networks The authors [HS97] addressed these effects by introducing the long short-term memory (LSTM), specifically designed to avoid problems of RNNs regarding long-term dependencies. LSTMs have, similar to RNNs, a chain-like structure but a different setup in each cell. Such a LSTM is showed in figure 2.3. The dashed lines on the left and right of the indicate previous and following cells respectively. Circular shaped blocks depict matrix operations and rectangular shaped blocks the activation function applied on the sum of a weighted input and bias.

The exact internal structure is not explained in detail here. In difference to the RNN, the LSTM uses a *cell state* C_t and a *hidden state* h_t for storing information over time. Based on the last hidden state and current input, the LSTM decides in $f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$ if information is added updated to the cell state or discarded. Hereby the sigmoid activation function is defined by $\sigma = \frac{1}{1+e^{-x}}$. The actual adding of information is then done in $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$ where i_t and $\tilde{C}_t$ are calculated $i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$ and $\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right)$ respectively. The summation instead of multiplication of the cell state is essential for the ability of the LSTM to store information over time without the problem of vanishing or exploding gradients as in RNNs. Finally, the hidden state, acting as output of the LSTM, is calculated based on the cell state with $h_t = o_t * \tanh\left(C_t\right)$ and $o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$.

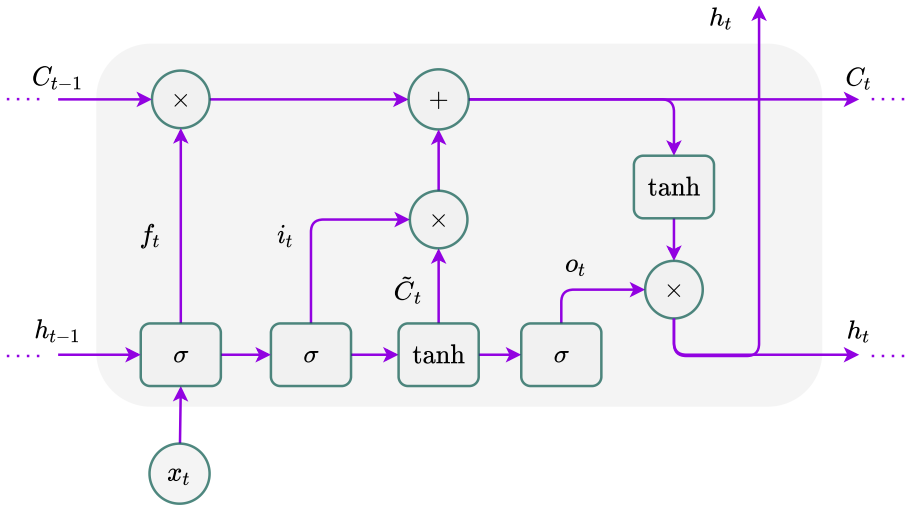


Figure 2.3: Schematic of a LSTM cell architecture derived from [HS97]. Enabled by additive processing of cell states C this architecture does not have the problem of vanishing or exploding gradients as in casae of the vanilla RNN in figure 2.2.

2.2. Quantum Computing

The following section introduces the fundamental terms and tools of QC required in the context of this thesis. In general quantum algorithms are build by the structure shown in figure 2.4 derived from [LB20].

After a preprocessing (e.g. normalization) of the input data, quantum states are prepared (encoding circuit) and the data then manipulated by unitary transformations ('functional' circuit). The result is measured and then passed on to the classical environment, where the output is evaluated and, if necessary, post-processed. The components of this structure are explained in detail in the following sections, but the figure 2.4 may be useful as a general roadmap.

2.2.1. Qubits and Qubit States

In classical computing, we represent the smallest unit of information by a *bit*, the quantum equivalent is called quantum bit (qubit).

The evolution of a quantum state ψ in time is, according to the *Time Evolution Postulate* [Sch19], described by Schrödingers equation $i\hbar\frac{d|\psi\rangle}{dt} = H|\psi\rangle$ where H is a fixed *Hamiltonian* and $\hbar = \frac{h}{2\pi}$ with h being the *Planck's constant* commonly absorbed in H [IH06].

The Hamiltonian has the spectral decomposition $H = \sum_E E|E\rangle\langle E|$ with $|E\rangle$ being the discrete *energy eigenstates* and the state with lowest energy being referred to as *ground state* [IH06]. In contrast, the *excited state*, represents a qubit with a higher energy level. Naturally, systems tend towards a state with the lowest possible energy, which holds true for qubits as well.

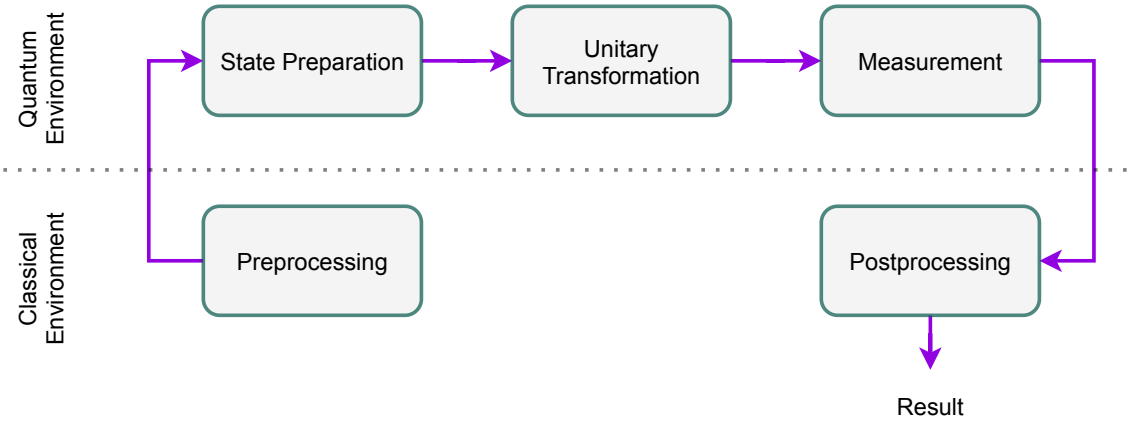


Figure 2.4: Quantum Algorithms structure derived from [LB20]. Input data is preprocessed classically and then encoded into quantum states (state preparation). These states are then manipulated by unitary transformations ('functional' circuit) and the result is measured. In the classical environment, this output is evaluated and, if necessary, post-processed.

Just like classical bits, qubits are described by their state, which, in contrast to classical bits, is not limited to 1 and 0. The representation of such a qubit can be derived by first introducing a vector notation of a classical bit and its equivalent in *Dirac notation*:

$$0 \to \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle \quad \text{and} \quad 1 \to \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle \tag{2.17}$$

The state of a qubit can be described by equation (2.18), thus allowing qubits to take any state in a two-dimensional complex vector space.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \quad \alpha, \beta \in \mathbb{C} \tag{2.18}$$

More precisely this space is called a *Hilbert Space* $\mathtt{H}$ and implies the definition of an hermitian inner product [IH06]

$$\langle , \rangle : \mathcal{H} \times \mathcal{H} \longrightarrow \mathbb{C} \tag{2.19}$$

and a norm [IH06]

$$\|u\| = \langle u, u \rangle^{\frac{1}{2}}, \quad u \in \mathcal{H}. \tag{2.20}$$

For a qubit's state in Dirac notation, the inner product is defined as

$$\langle \psi | \phi \rangle := \langle \psi, \phi \rangle = (\langle \psi |) \cdot (| \phi \rangle) \tag{2.21}$$

where $\langle \psi |$ is the complex conjugate transpose of $|\psi\rangle$ (dual to $|\psi\rangle$), so that $|\psi\rangle^\dagger = \langle \psi |$ [JJS20]. Furthermore it is $\langle \psi | \phi \rangle = \langle \phi | \psi \rangle^\dagger$.

In addition to the representation in equation (2.18), qubits can form a superposition (linear combination) of the $|0\rangle$- and $|1\rangle$-states, resulting in the general representation of a qubit's state as depicted in (2.18). If not stated otherwise, the qubit is always initially in the $|0\rangle$ state. To introduce some more language, an exactly known state ψ of a qubit is called a *pure* state [IH06]. Contrastingly, a *mixed state* is the mixture of different pure states [IH06].

Nevertheless, if we measure a qubit we will never measure a superposition, but either $|0\rangle$ or $|1\rangle$ with a certain probability $|\alpha|^2$ or $|\beta|^2$ respectively instead. This means that although it is possible to create an infinite amount of possible states, at the time a qubit is measured, its superposition collapses into a final value, either 0 or 1.

For this reason, QC only allows statistical reasoning about a qubit's state by evaluating several thousand measurements. Therefore for any state vector ψ it is required that

$$\|\psi\|^2 = 1 \tag{2.22}$$

must hold true [Sch19].

A superposition of multiple states obtained in a single qubit in state $|\psi\rangle$ is commonly visualized by the so called *Bloch sphere* [Blo46]:

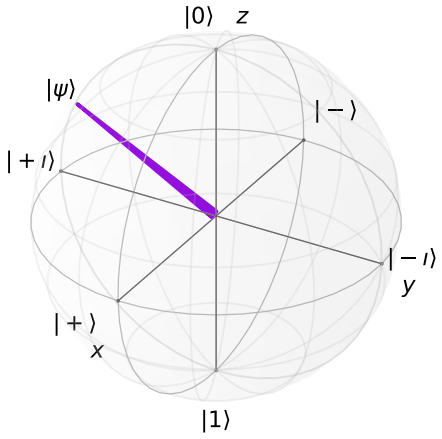


Figure 2.5: Bloch sphere displaying a state $|\psi\rangle$, generated with [JNN13]. $|0\rangle$ and $|1\rangle$ state along the Z-axis, $|+\rangle$ and $|-\rangle$ state along the Z-axis and $|-\imath\rangle$ and $|+\imath\rangle$ state along the Y-axis.

The z axis, in figure 2.5 represents the ground ($|0\rangle$) and excited ($|1\rangle$) state. Furthermore, states along the y axis are referred to as $|-\imath\rangle$ and $|+\imath\rangle$ and along the x axis as $|+\rangle$ and $|-\rangle$. At this point it should be noted that the naming of the states in this thesis may differ from the ones in the literature in some cases.

Utilizing the geometric representation of states within the Bloch sphere, we can derive another notation of the quantum state in equation (2.18) as an expression of the angles θ (between and z axis and the state) and ϕ (between the x axis and the state in the XY plane).

If we rewrite a qubit's state using Euler's form, we obtain

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = r_0 e^{\imath\gamma_0} + r_1 e^{\imath\gamma_1} = e^{\imath\gamma_0}(r_0|0\rangle + r_1 e^{\imath(\gamma_1-\gamma_0)}|1\rangle) \tag{2.23}$$

Measurement of a quantum state gives the amplitude of the latter, which will be discussed

in 2.2.3. However, calculating the amplitude of the term in equation (2.23) will eliminate the first factor $\exp(\imath\gamma_0)$ [Wil11]. This term also referred to as the *global phase* of a quantum state and is a non-measurable value as shown in the following equation (2.24).

$$\langle\psi|\psi\rangle = e^{-\imath\gamma_0}\langle\phi|e^{\imath\gamma_0}|\phi\rangle = \langle\phi|\phi\rangle, \quad |\psi\rangle = e^{\imath\gamma_0}|\phi\rangle \tag{2.24}$$

where it has been used that $e^{-\imath\gamma+\imath\gamma} = 1$. Contrary, the *relative phase* $\gamma_1 - \gamma_0$ in equation (2.23) is an observable quantity.

Equation (2.23) and the fact that it is not possible to measure the global phase, allows to confine $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ to real numbers $\in \mathbb{R}$ and add a relative phase term [ACB+20]:

$$|\psi\rangle = \boldsymbol{\alpha}|0\rangle + \mathrm{e}^{\imath\phi}\boldsymbol{\beta}|1\rangle \tag{2.25}$$

The normalization introduced in (2.22) can be written as

$$\sqrt{\boldsymbol{\alpha}^2 + \boldsymbol{\beta}^2} = 1 \tag{2.26}$$

Furthermore the trigonometric identity can be used to rewrite $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ as an expression of the a single variable θ:

$$\begin{aligned} &\sqrt{\sin^2 x + \cos^2 x} = 1 \quad x \in \mathbb{R} \\ \Rightarrow \qquad &\boldsymbol{\alpha} = \cos\frac{\theta}{2} \qquad \boldsymbol{\beta} = \sin\frac{\theta}{2} \quad \boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R} \end{aligned} \tag{2.27}$$

This yields the equation of a qubit's state expressed by the angles θ and ϕ:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + \mathrm{e}^{\imath\phi}\sin\frac{\theta}{2}|1\rangle \quad \theta \in [0, \pi], \phi \in [0, 2\pi] \tag{2.28}$$

where θ and ϕ rotate around the x and y axis respectively.

The two computational basis states introduced in equation (2.17) for a single qubit extend to $N = 2^n$ computational basis states (superposition of theses states) in the n-qubit case. Therefore, a quantum state consisting of e.g. 2 qubits can be written as:

$$|\psi\rangle = \boldsymbol{\alpha}_{00}|00\rangle + \boldsymbol{\alpha}_{01}|01\rangle + \boldsymbol{\alpha}_{10}|10\rangle + \boldsymbol{\alpha}_{11}|11\rangle \tag{2.29}$$

Where e.g. $|00\rangle$ denotes the tensor product $|0\rangle\otimes|0\rangle$ as defined in equation (2.31). It is important to note that the normalization introduced in equation (2.22) also needs to hold for the n-qubit case:

$$\sum_{\iota\in\{0,1\}^n} |\boldsymbol{\alpha}_\iota|^2 = 1 \tag{2.30}$$

where the notation in [IH06] was extended to the n-qubit case, such that the set $\{0,1\}^n$ represents an arbitrary string of 0s and 1s indicating the computational basis.

Referring to the vector notation in equation (2.17) it is useful to note that a quantum state composed of n qubits is a $N = 2^n$ dimensional vector, calculated by the *Tensor product* [Sch19]:

$$|\psi\rangle = \bigotimes_{i}^{N} |\psi_i\rangle = \begin{pmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_n \end{pmatrix} \quad |\psi\rangle \in \mathbb{C}^N \tag{2.31}$$

where $|\psi_i\rangle = \begin{pmatrix} \boldsymbol{\alpha}_{x_i} \\ \boldsymbol{\beta}_{x_i} \end{pmatrix}$ refers to the ith qubits state.

Furthermore the *tensor product* between a $|\psi\rangle$ and $\langle\phi|$ is denoted by

$$|\psi\rangle\langle\phi| \tag{2.32}$$

from [JJS20].

The Kronecker product as a special case of the *tensor product* for matrices is defined as in [IH06] by:

$$A \otimes B \equiv \overbrace{\left.\begin{bmatrix} A_{11}B & A_{12}B & \dots & A_{1n}B \\ A_{21}B & A_{22}B & \dots & A_{2n}B \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1}B & A_{m2}B & \dots & A_{mn}B \end{bmatrix}\right\}}^{=nq} = mp \tag{2.33}$$

where the matrices A and B can have different dimensions such that $A \in \mathbb{C}^{m \times n}$ and $B \in \mathbb{C}^{p \times q}$.

2.2.2. Operations

Given a set of qubits, their state can be changed by applying quantum gates which are further referred to as "operations". These operations are represented by their matrix, which must be unitary due to the *Time Evolution Postulate* [Sch19]. It describes, assuming measurements are deferred, the evolution of a state $|\psi(t_0)\rangle$ in a Hilbert Space $\mathcal{H}$, to another state $|\psi(t)\rangle$. This transition can be described by the time evolution operator $U(t, t_0)$ using the matrix-vector product:

$$|\psi(t)\rangle = U(t, t_0)|\psi(t_0)\rangle \quad |\psi(t)\rangle, |\psi(t_0)\rangle \in \mathbb{C}^N \quad U \in \mathbb{C}^{N \times N} \tag{2.34}$$

where $U(t, t_0)$ is the solution of the following initial value problem, namely the operator equivalent of the *Schrödinger equation* [Sch19]:

$$\begin{aligned} i\frac{d}{dt}U(t, t_0) &= H(t)\,U(t, t_0) \\ U(t_0, t_0) &= \mathbf{1} \end{aligned} \tag{2.35}$$

and U must hold $U^\dagger U = 1$

Constrained by equation (2.22), operations on a qubit will always *move* on the surface of the Bloch sphere. In Dirac notation, subsequent operations can be expressed as following:

$$|\psi(t)\rangle = U_k \cdot U_{k-1} \cdot \ldots \cdot U_0 |0\rangle \quad k \in \mathbb{N} \tag{2.36}$$

An illustrative representation of this expression can be obtained by the *Circuit Notation* where subsequent gates applied on qubit(s) are written on horizontal lines.

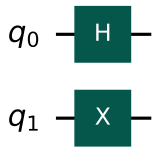


Figure 2.6: Quantum circuit showing a Hadamard (`H`) and Negation (`X`) gate applied on two separate qubits in the $|0\rangle$ state.

The exemplary circuit in figure 2.6 contains a Hadamard gate `H` applied on the first and a negation gate (`X` gate) applied on the second qubit. The matrix representation of these gates are stated in the following equations (2.37) and (2.38) respectively. It should be noted, that `H` denotes the Hadamard gate and is not to be confused with the Hilbert Space $\mathcal{H}$ or hamiltonian operator H.

$$\mathtt{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.37}$$

$$\mathtt{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2.38}$$

Applying these gates on the $|0\rangle$ state (left side in (2.17)), yields the following qubit's state in case of the Hadamard gate applied on the initial state $|\psi_0\rangle$ of the first qubit q_0:

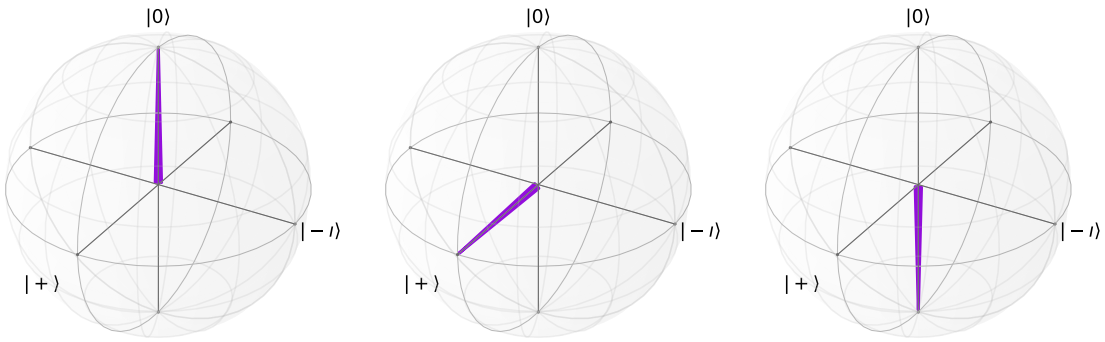
$$\mathtt{H}|\psi_0\rangle = \mathtt{H}|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle \tag{2.39}$$

and in case of the negation gate applied on the initial state $|\psi_1\rangle$ of the second qubit q_1:

$$\mathtt{X}|\psi_1\rangle = \mathtt{X}|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle \tag{2.40}$$

These states can be visualized using Bloch spheres shown in 2.7 where the first one depicts the $|0\rangle$ state of a qubit followed by the representation of the $|1\rangle$ and $|+\rangle$ state.

As the `X` gate rotates the qubit's state by 180 deg around the x axis, other gates such as the `Y` and `Z` gate have equivalent effect on the y and z axis respectively. The Hadamard gate creates a superposition of the $|0\rangle$ and $|1\rangle$ state such that $\mathtt{H}|0\rangle = |+\rangle$ and $\mathtt{H}|1\rangle = |-\rangle$.



(a) **(b)** **(c)**

Figure 2.7: Bloch spheres describing the states in the circuit from figure 2.6 consisting of an `X` resulting in the $|1\rangle$ state (Figure 2.7c) and `H` resulting in the $|+\rangle$ state (Figure 2.7b) gate applied to the $|0\rangle$ state (Figure 2.7a).

Pauli Gates The `X` gate is one of the well-known *Pauli matrices*. The remaining ones are `Y` and `Z` gates and defined as follows:

$$\mathtt{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \mathtt{Y} = \begin{bmatrix} 0 & -\imath \\ \imath & 0 \end{bmatrix} \qquad \mathtt{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.41}$$

The purpose of these matrices is similar to the discussed `X` gate, as `Y` and `Z` gates perform rotations by π rad around the y and z axis respectively. However, these rotations of Φ can also be precisely controlled by the $\mathtt{R_X}$, $\mathtt{R_Y}$ and $\mathtt{R_Z}$ gates as follows [IH06]:

$$\begin{aligned} \mathtt{R_X}(\Phi) &= \begin{bmatrix} \cos(\frac{\Phi}{2}) & -i\sin(\frac{\Phi}{2}) \\ -i\sin(\frac{\Phi}{2}) & \cos(\frac{\Phi}{2}) \end{bmatrix} \\ \mathtt{R_Y}(\Phi) &= \begin{bmatrix} \cos(\frac{\Phi}{2}) & -\sin(\frac{\Phi}{2}) \\ -\sin(\frac{\Phi}{2}) & \cos(\frac{\Phi}{2}) \end{bmatrix} \\ \mathtt{R_Z}(\Phi) &= \begin{bmatrix} \mathrm{e}^{-i\frac{\Phi}{2}} & 0 \\ 0 & \mathrm{e}^{i\frac{\Phi}{2}} \end{bmatrix} \xrightarrow{\cdot \mathrm{e}^{i\frac{\Phi}{2}}} \begin{bmatrix} 1 & 0 \\ 0 & \mathrm{e}^{i\Phi} \end{bmatrix} = \mathtt{P}(\Phi) \end{aligned} \tag{2.42}$$

The $\mathtt{R_Z}$ is often also referred to as *phase gate* `P`.

$\mathtt{C_X}$ Gate The $\mathtt{C_X}$ gate, often referred as $\mathtt{C_X}$ gate, extends the `X` by a control input. Depending on the state of the control qubit, the negation operation is applied on the target qubit. Equation (2.43) shows both cases of a $\mathtt{C_X}$ matrix, with the first qubit acting as the control- and the second qubit as the target input ($C = 0, T = 1$) and vice-versa ($C = 1, T = 0$).

$$\mathtt{C_X}^{C=0,T=1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad \mathtt{C_X}^{C=1,T=0} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.43}$$

Assuming $|0\rangle$ and $|1\rangle$ as the only values on the control qubit, the truth table of the $\texttt{C}_\texttt{X}$ would be identical to a classical exclusive or-gate:

Input		Output	
Control	Target	Control	Target
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|0\rangle$	$\|1\rangle$	$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|1\rangle$	$\|0\rangle$

Table 2.1: Truth table of the $\texttt{C}_\texttt{X}$ gate

Further more quantum mechanical effects of the $\texttt{C}_\texttt{X}$ gate will be discussed in section 2.2.4.

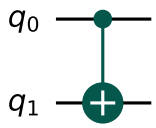
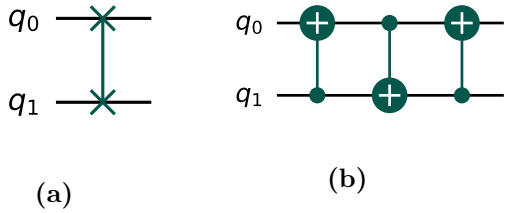


Figure 2.8: $\texttt{C}_\texttt{X}$ gate with q_0 as the control qubit applied on the target qubit q_1

SWAP Gate Certain applications, e.g. the quantum Fourier transform (QFT), require to swap the state of two qubits within a circuit. This operation can be accomplished by the *Swap Gate* shown in figure 2.9. While the representation on the left of this figure is commonly used, it can be decomposed into three single $\texttt{C}_\texttt{X}$ gates.



(a) (b)

Figure 2.9: Swap gate in circuit notation (Figure 2.9a) and its decomposition into three $\texttt{C}_\texttt{X}$ gates (Figure 2.9b)

The matrix representation of the swap gate is given by equation (2.44).

$$\texttt{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.44}$$

CP(Φ) Gate In context of the QFT, another important gate is the controlled-phase gate CP(Φ) parameterized by the rotation angle Φ. This gate induces a predefined phase Φ in the state of the target qubit depending on the state of the control qubit [AAA+21]. Its

matrix representation can be derived from the $\mathtt{C_X}$ and the $\mathtt{P}$ as stated in equation (2.45). In contrast to the $\mathtt{C_X}$ gate, there is no difference between the control and target qubit.

$$\mathtt{CP}(\Phi) = |0\rangle\langle 0| \otimes \mathtt{I} + |1\rangle\langle 1| \otimes \mathtt{P} = \mathtt{I} \otimes |0\rangle\langle 0| + \mathtt{P} \otimes |1\rangle\langle 1| = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\Phi} \end{pmatrix} \tag{2.45}$$

Figure 2.10 shows the representation of this gate in circuit notation with an exemplary rotation of $\Phi = \frac{\pi}{4}$.

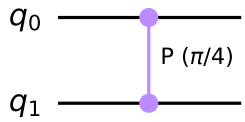


Figure 2.10: Phase gate $\mathtt{CP}(\Phi)$ with $\Phi = \frac{\pi}{4}$. Depending on the state of the control qubit this phase is induced in the state of the target.

2.2.3. Measurement

The probability of measuring a state $|\psi\rangle$ in any arbitrary state $|\phi\rangle$ is the absolute squared of the inner product between theses states $|\langle\phi|\psi\rangle|^2$ as explained for example in [ACB+20].

Measuring in different states allows to measure along various axes as represented by the Bloch sphere. If not stated otherwise, measurements will always be applied along the z axis, thus either in $|0\rangle$ or $|1\rangle$ state in case of a single qubit.

To see this in an example, the Hadamard gate from circuit 2.6 shall now be regarded isolated as shown in figure 2.11. The measurement of a qubit within a quantum circuit is depicted by the measurement block on the very right in figure 2.11. This block usually represents the end of a circuit since applying the measurement operation collapses a quantum state into a final value.

Figure 2.11: Circuit containing a single Hadamard Gate $\mathtt{H}$ and a measurement operator.

Applying the Hadamard gate on a qubit $|q\rangle$ in the $|0\rangle$ state results in a superposition state $|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ as shown in equation (2.39) The measurement of a qubit in this state leads to the final probability of $1/2$ as shown in equation (2.46) [ACB+20].

$$
\begin{aligned}
\langle 0|q\rangle = \langle 0|+\rangle &= \frac{1}{\sqrt{2}}\langle 0|0\rangle + \frac{1}{\sqrt{2}}\langle 0|1\rangle \\
&= \frac{1}{\sqrt{2}}1 + \frac{1}{\sqrt{2}}0 \\
&= \frac{1}{\sqrt{2}} \\
|\langle 0|q\rangle|^2 &= \frac{1}{2}
\end{aligned}
\tag{2.46}
$$

Ideally, the measurement results for the $|0\rangle$ and $|1\rangle$ state would be identical in this specific example. However, as discussed in 2.2.1, real measurement in QC only allows for statistical reasoning. This means, that the ideal probability of $1/2$ in both states would only be obtained approximately by running a few thousand measurements, also referred to as *shots*.

The resulting probabilities of a quantum circuit are usually visualized in a histogram. This histogram results by counting the number of times, a specific state is measured in a series of shots. It should be noted that for obtaining the probabilities from a histogram, a normalization is implicitly carried out. For aboves example, the corresponding histogram is given in figure 2.12. As expected, small fluctuations around the ideal values are visible.

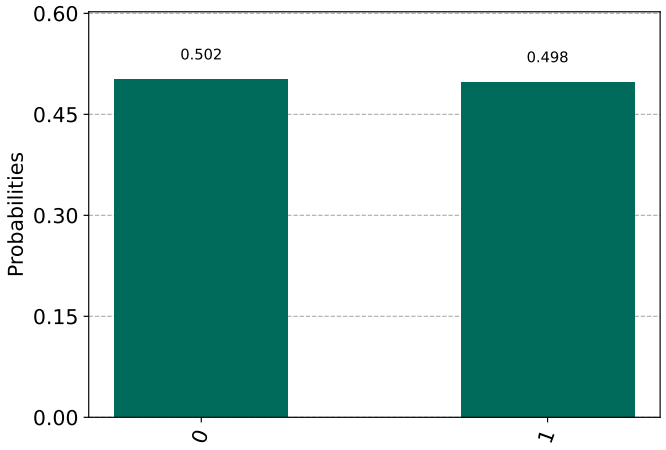


Figure 2.12: Histogram of a `H` gate applied on a qubit in the $|0\rangle$ state as in the circuit from figure 2.11. Small fluctuations around the ideal value (0.5) are visible.

2.2.4. Entanglement

Entanglement is a unique property of QC and is fundamental for computational advantages compared to classical computing [ACB+20]. It describes the effect, that two qubits can share a state of superposition [Zic21]. This „spooky-action-at-a-distance“, as Albert Einstein described it, also affects measurement in a way, that if one of two entangled qubits is measured, the other one will automatically collapse as well.

This section shows the effect of entanglement using an example depicted in figure 2.13. In this circuit, a Hadamard gate is applied on the first qubit whose output is the control

input for the *Controlled-Not* gate or `CX` gate for short, applied on the second qubit. An additional 2-bit register is used to store the measurement results of both qubits.

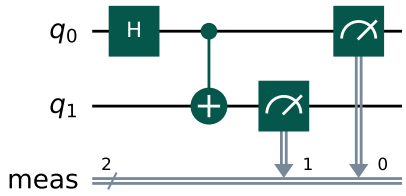


Figure 2.13: Circuit containing a Hadamard (`H`) and a Controlled-Not Gate (`CX`) as well as measurements on both qubits.

On a classical system, a set control bit will cause the target bit to be flipped as discussed while introducing the `CX` gate in section 2.2.2. However, in this case as well as in QC in general, it is more convenient to say that the value of the control bit qubit controls the *amount* of probabilities at which the target qubit is flipped.

The Hadamard gate in figure 2.13 causes the control qubit to be in a superposition of $|0\rangle$ and $|1\rangle$ which is then directly transferred to the target qubit. Measurements of this circuit yield the histogram presented in figure 2.14 which shows two important effects; First, the probabilities for $|01\rangle$ and $|10\rangle$ are exactly zero (in case of simulation without noise) since two entangled qubits will never distinguish in their final state [Zic21]. Second, the effect of entanglement becomes visible, when considering that the measurement on the target qubit q_1 is evaluated *before* the measurement on the control qubit q_0. Classically this would mean, that the effect is seen before the cause. In QC the `CX` gate is therefore said to entangle two qubits with each other [Zic21].

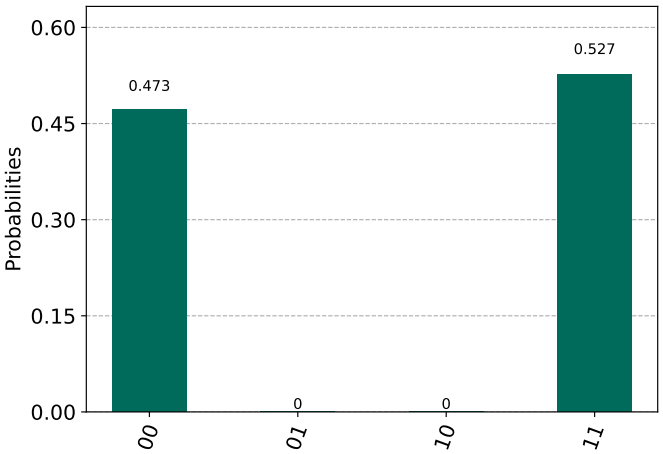


Figure 2.14: Histogram of a `CX` gate applied on a target qubit in the $|+\rangle$ state yielding a Bell state as in the circuit from figure 2.13. The states $|01\rangle$ and $|10\rangle$ have zero probability.

The resulting state (one of four possible so called *Bell states* [IH06]) can also be obtained analytically, by evaluating the matrix operations applied on the initial state $|00\rangle$ of both qubits as depicted in equation (2.48). Since the Hadamard gate is a 1-qubit operator only applied on the first qubit, but needs to match the size of the 2-qubit `CX` matrix, it needs to be tensored with the identity operator `I` yielding equation (2.47).

$$\texttt{H}\otimes\texttt{I} = \frac{1}{\sqrt{2}}\begin{bmatrix} I & I \\ I & -I \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \tag{2.47}$$

$$\begin{aligned} \texttt{C}_\texttt{X}(\texttt{H}\otimes\texttt{I})|00\rangle &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} \end{aligned} \tag{2.48}$$

It should be noticed, that the state of the overall system of qubits, the *Bell state*, is well known and thus a pure state, although the first qubit is in a mixed state because it is entangled with the second qubit [IH06]. This finding can be derived from the trace of the density operator introduced in equation (2.53).

2.2.5. Realization of Circuits

In QC the topology of the device plays an important role. Connections between qubits (e.g. as required in the $\texttt{C}_\texttt{X}$ gate) must be realized on the actual qubit layout of the specific device chosen. However, in the NISQ era, not all qubits are necessarily connected with all others which requires a transpiler to find the most optimal realization of a given circuit such that the least amount of additional gates is required to *swap* qubits state across the quantum computer [Pre18]. There exist techniques which are able to tackle this problem but are not regarded in detail within this thesis.

It should be noted, that these *transpilers* are not always deterministic due to the high-dimensional optimization problem that must be solved on complex circuits. Furthermore, transpilers generally also try to implement the circuit on a quantum device such that noisy qubits or their interconnections are omitted. Therefore, resulting circuit layouts may vary between identical experiments and which is why transpilers are regarded as another source of stochasticity to be accounted in the results. For the reason that the QFT, discussed in section 2.2.8 in particular requires a connection between every qubit, causing additional `SWAP` operations during transpilation, resulting circuits can become very large which in turn increases the overall noisiness.

2.2.6. Noise

In section 2.2.3 it was already mentioned, that real measurement results cannot be expected to be equal to the expectation value obtained from solving the circuit's matrices. In addition to that, QC especially in the NISQ era is always noisy, whereby this noise can be separated

into *coherent* and *incoherent* noise [Min21]. These types of noise are discussed in the upcoming sections and finally lead to the conclusion that noisy quantum systems (gates and measurement devices) actually cause a shrinkage of the applicable space within the Bloch sphere.

Coherent Noise

Coherent noise comes from systematic errors and „differ from the desired operation by a unitary expression“ [BEH+04] in gate operators. These are caused by e.g. inaccurate rotations of a $\mathtt{R_X}$ gate which cause a qubit's state to be rotated by $\phi - \tau$ instead of ϕ. Coherent errors can also be caused by faulty measurement devices which will be covered later in this section.

Gate Errors For the following explanation of the effect of coherent noise, a circuit with d subsequent `X` gates as depicted in figure 2.15 is being regarded. One would expect a toggling behavior when increasing the number d of `X` gates applied and tracking the qubit's state over the number of theses gates as shown on the left in figure 2.16. It should be noted, that such a measurement procedure would require evaluations of d different circuits. This is due to the fact that a single measurement in between a quantum circuit would cause subsequent states to collapse thus making it impossible to observe the desired effect as discussed in section 2.2.3.

Figure 2.15: Circuit consisting of d subsequent `X` gates.

However, it turns out that even for a very small error, the actual output shows an oscillating behavior as shown on the right in figure 2.16 for increasing circuit depth d.

It should be noted, that the the errors being discussed are iteratively added to the graphs in following related figures. These figures also show the previous graph as a grey hull for reference.

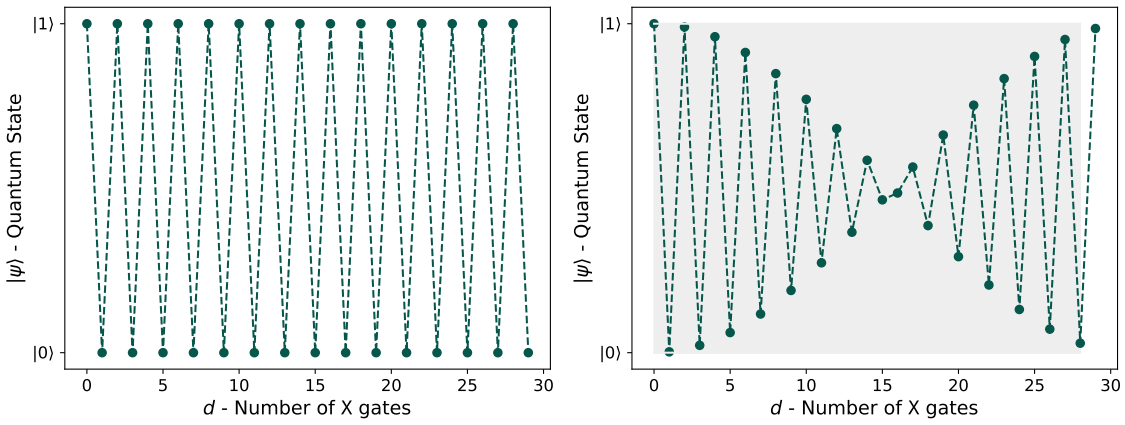
This oscillating behavior can be explained by considering the noisy $\tilde{\mathtt{X}}$ gate being a parameterized rotation gate combined with an ideal `X` gate:

$$\tilde{\mathtt{X}} = \tilde{\mathtt{R}}_\mathtt{X}(\tau)\mathtt{R}_\mathtt{X}(\pi) \qquad \text{with} \quad \mathtt{R}_\mathtt{X}(\pi) = \mathtt{X} \tag{2.49}$$

which yields the overall unitary of the circuit:

$$\tilde{\mathtt{U}} = \tilde{\mathtt{X}} = [\tilde{\mathtt{R}}_\mathtt{X}(\tau)\mathtt{R}_\mathtt{X}(\pi)]^d = \mathtt{R}_\mathtt{X}(d\tau)\mathtt{R}_\mathtt{X}(d\pi) = \mathtt{R}_\mathtt{X}(d\tau)\mathtt{U} \tag{2.50}$$

Measurement Errors Measurement errors are due to imperfect measurement devices, which, with a probability ν and ϵ return 0 instead of 1 and 1 instead of 0 respectively. This yields the transition graph shown in figure 2.17a.



(a) **(b)**

Figure 2.16: 2.16a: Measurement of circuit 2.15 with ideal gates
2.16b: Measurement of circuit 2.15 with gates errors $\tau = 0.1$ rad resulting in an oscillating behavior. Figure 2.16a is shown as a grey hull for reference

Such a noisy measurement can be modeled by introducing a stochastic measurement variable:

$$\begin{aligned} \mathcal{P}(\tilde{\mathtt{M}} = 0) &= \mathcal{P}(\tilde{\mathtt{M}} = 0|\mathtt{M} = 0)\mathcal{P}(\mathtt{M} = 0) + \mathcal{P}(\tilde{\mathtt{M}} = 0|\mathtt{M} = 1)\mathcal{P}(\mathtt{M} = 1) \\ \mathcal{P}(\tilde{\mathtt{M}} = 1) &= \mathcal{P}(\tilde{\mathtt{M}} = 1|\mathtt{M} = 1)\mathcal{P}(\mathtt{M} = 1) + \mathcal{P}(\tilde{\mathtt{M}} = 1|\mathtt{M} = 0)\mathcal{P}(\mathtt{M} = 0) \end{aligned} \tag{2.51}$$

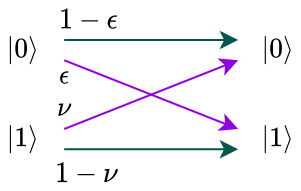
where $\tilde{\mathtt{M}}$ depicts a noisy measurement device $\mathtt{M}$. This equation can also be interpreted as a mapping of the ideally expected probability p of a measurement $\mathtt{M}$ onto the noisy probability $\tilde{p}$ of a measurement $\tilde{\mathtt{M}}$ which is shown in figure 2.17b.

In the ideal case $\epsilon = 0$ and $\nu = 0$ holds true, thus results in an identity between p and $\tilde{p}$ depicted by the green line. The other line represents the case of a noisy measurement, where the probability of the measurement $M = 1$ will neither be fully 0 or 1 but 0.08 and 0.8 instead.

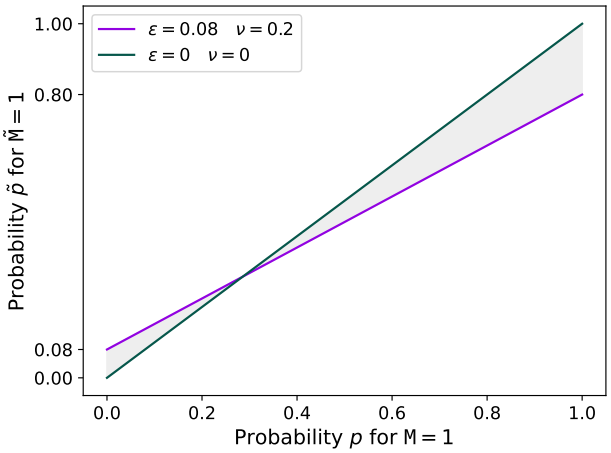
Applying this behavior in addition to the results from the right graph in figure 2.16 leads to a shift towards the $|0\rangle$ state which is shown in figure 2.18a. Again, the hull from the right graph in figure 2.16 is added as reference.

Besides the gate error and the measurement error, the underlying stochastic evaluation method of the whole system introduced in 2.2.3 needs to be considered This in turn makes an evaluation of values besides 0 and 1 possible in the first place.

The graph in figure 2.18b shows this property using a binomial distribution $\mathcal{B}(n = 100, p = 0.5)$, resulting in small fluctuations. These become visible when being compared with the gray hull representing the graph in figure 2.18a.



(a)



(b)

Figure 2.17: 2.17b: Measurement state transition graph modeling the ideal and noisy behavior with a transition probability of ϵ and ν respectively.
2.17a: Resulting probability mapping between the expected measurement of $M = 1$ and the noisy result for $\tilde{M} = 1$. The green represents the ideal and the other line the noisy case.

Incoherent Noise

This section covers aspects of incoherent noise on quantum gates and how this affects the stochastic measurement results. First a more general approach is considered, where the behavior of a noisy gate is replaced by an ideal gate `G` cascaded with either an identity `I` or a `X` gate as depicted in figure 2.19. Here, the identity gate represents the ideal case, the `X` gate models a bit flip. This case decision is just another way of modeling a noisy gate chosen for the discussion in the remainder of this section.

If we assume `G` being a `X` gate as in the example from the previous section, the total unitary become

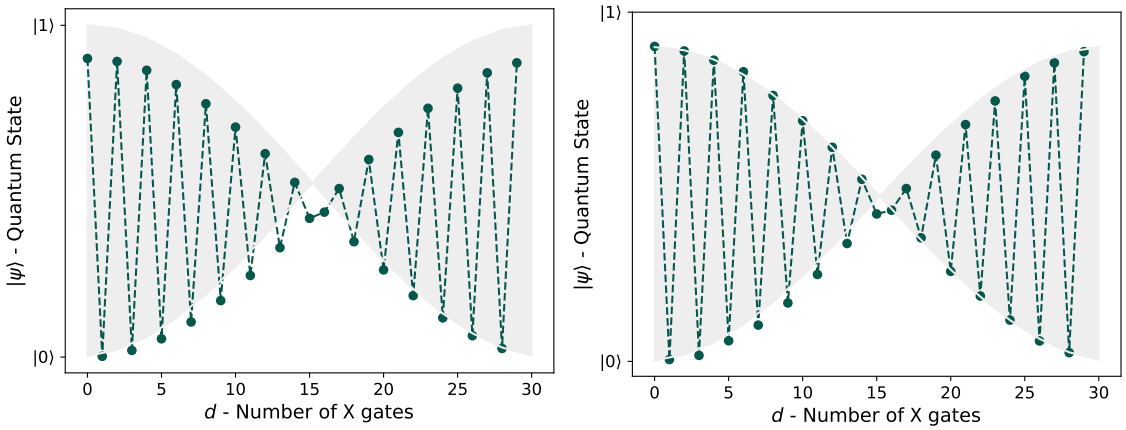
$$\begin{aligned} \text{Ideal case (A)}: \quad & \mathtt{U_A} = \mathtt{GI} = \mathtt{X} \\ \text{Flipped case (B)}: \quad & \mathtt{U_B} = \mathtt{GX} = \mathtt{XX} = \mathtt{I} \end{aligned} \tag{2.52}$$

where case A and case B occur with the probability $1 - p$ and p respectively. These probabilities are at the same time the expectation values for the case the circuit was stochastically evaluated as discussed in section 2.2.3.

Further discussions require the introduction of the *density operator* ρ defined in equation (2.53) as another approach for describing quantum systems. This operator, or matrix in general, comes useful if the state of a system is not completely known [IH06].

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i| \tag{2.53}$$

where p_i describes the probability of a system being in the state $|\psi\rangle$. A state ψ which



(a) **(b)**

Figure 2.18: 2.18a: Measurement of circuit 2.15 with gates errors $\tau = 0.1$ rad and additional measurement error as depicted in figure 2.17b resulting in a shift towards $|0\rangle$.
Gray hull indicates results from figure 2.16b
2.18b: Measurement of circuit 2.15 with gates errors $\tau = 0.1$ rad, measurement errors and general stochastic properties of quantum measurement modeled as a binomial distribution with $\mathcal{B}(n = 100, p = 0.5)$ resulting in fluctuations around the grey hull from figure 2.18a.

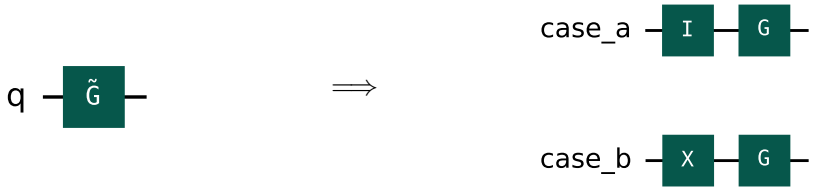


Figure 2.19: Noisy gate `G̃` on the left and its decomposition on the right after introducing a case decision where a bit flip occurs in *case_b* indicated by the `X` gate.

density matrix has a trace $\mathrm{Tr}(\rho) \leq 1$ is said to be in a mixed state. Contrastingly, density matrix with a trace $\mathrm{Tr}(\rho) = 1$ indicates a pure state [IH06]. If we track multiple states of a system each with different density matrices, the overall density matrix can be calculated by [IH06]:

$$\rho = \sum_i p_i \rho_i \tag{2.54}$$

Using equation (2.54), the mixture of circuits on the right in figure 2.19 can be used to describe the density matrix as follows:

$$\rho = p_A \rho_A + p_B \rho_B = (1-p) \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} + p \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} p & 0 \\ 0 & 1-p \end{bmatrix} \tag{2.55}$$

where the density operators for each case can be calculated using the unitary matrices

U_A and U_B as defined in equation (2.52) for the ideal and flipped case respectively:

$$\begin{aligned} \text{Case A} \quad & |\psi_A\rangle = U_A|1\rangle = |1\rangle \rightarrow \rho_A = |\psi_A\rangle\langle\psi_A| = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \\ \text{Case B} \quad & |\psi_B\rangle = U_B|0\rangle = |0\rangle \rightarrow \rho_B = |\psi_B\rangle\langle\psi_B| = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \end{aligned} \tag{2.56}$$

This means that the representation using the density matrix is ambiguous as shown in [IH06], so that a pair of quantum states $|\psi\rangle$ and $|\phi\rangle$ will result in the same density matrix if $|\psi_i\rangle = \sum_j u_{ij}|\phi_j\rangle$ holds true. It should be noted, that *ambiguous* refers to the ensemble of operations that led to a specific density matrix and should not be confused with the description of a system by its density matrix, which in turn is unique [IH06].

The resulting value corresponding to a measurement along one of the three axes within the Bloch sphere of a particular state can be obtained from the product of Pauli matrices and the density matrix using the *Trace* operator $\mathrm{Tr}(\ )$:

$$\begin{aligned} \hat{x} &= \mathrm{Tr}\left(\mathtt{X}\rho\right) = 0 \\ \hat{y} &= \mathrm{Tr}\left(\mathtt{Y}\rho\right) = 0 \\ \hat{z} &= \mathrm{Tr}\left(\mathtt{Z}\rho\right) = \mathrm{Tr}\left(\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} p & 0 \\ 0 & 1-p \end{bmatrix}\right) = 2p-1 \end{aligned} \tag{2.57}$$

The values along the x and y dimensions are trivial since this particular density matrix is orthogonal to these axis. Using the geometric representation introduced in equation (2.27) it is possible to derive a general expression for the density matrix described by the angles ϕ and θ:

$$\begin{aligned} \rho = |\psi\rangle\langle\psi| &= \begin{pmatrix} \cos\frac{\theta}{2} \\ \sin\frac{\theta}{2}\mathrm{e}^{\imath\phi} \end{pmatrix}\begin{pmatrix} \cos\frac{\theta}{2} & \sin\frac{\theta}{2}\mathrm{e}^{\imath\phi} \end{pmatrix} \\ &= \begin{bmatrix} \cos^2\frac{\theta}{2} & \cos\frac{\theta}{2}\sin\frac{\theta}{2}\mathrm{e}^{-\imath\phi} \\ cos\frac{\theta}{2}\sin\frac{\theta}{2}\mathrm{e}^{+\imath\phi} & \sin^2\frac{\theta}{2} \end{bmatrix} \\ &= \begin{bmatrix} \cos^2\frac{\theta}{2} & \frac{1}{2}\sin\theta(\cos\phi - \imath\sin\phi) \\ \frac{1}{2}\sin\theta(\cos\phi + \imath\sin\phi) & \sin^2\frac{\theta}{2} \end{bmatrix} \\ &= \frac{1}{2}(\mathtt{I} + x\mathtt{X} + y\mathtt{Y} + z\mathtt{Z}) \end{aligned} \tag{2.58}$$

The identity gate is hereby denoted by $\mathtt{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Assuming ρ_{in} being a quantum state as defined in (2.58) and $\tilde{\mathtt{G}}$ a noisy quantum gate as shown in figure 2.19, the resulting density matrix can then be calculated as follows:

$$\begin{aligned}\rho_{out} = \tilde{\mathtt{G}}(\rho_{in}) &= (1-p)I\rho_{in}\mathtt{I} + p\mathtt{X}\rho_{in}\mathtt{X} \\ &= (1-p)\begin{pmatrix} +x_{in} \\ +y_{in} \\ +z_{in} \end{pmatrix} + p\begin{pmatrix} +x_{in} \\ -y_{in} \\ -z_{in} \end{pmatrix}\end{aligned} \tag{2.59}$$

where $X\rho X = \frac{1}{2}(\hat{I} + x\mathtt{X} - y\mathtt{Y} - z\mathtt{Z})$ has been used resulting from the definition of the $\mathtt{X}$ Pauli matrix. This then results in

$$\rho_{out} = (1-2p)\begin{pmatrix} +x_{in} \\ (1-2p)y_{in} \\ (1-2p)z_{in} \end{pmatrix} \tag{2.60}$$

which finally leads to the conclusion that the applicable space has been shrunk from the full Bloch sphere to an elliptical sphere as shown in figure 2.20. This can be interpreted as the *reachable* space retained after applying a noisy gate.

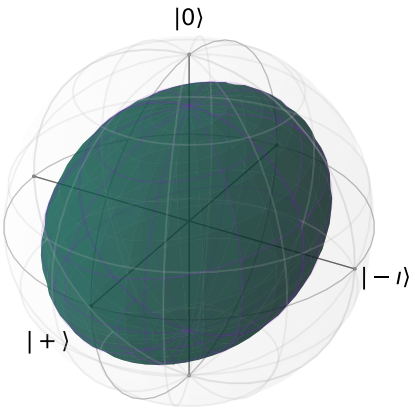


Figure 2.20: In y and z dimension shrunk applicable Bloch sphere space (green) resulting from bit flip errors by measuring a noisy gate $\tilde{\mathtt{G}}$ gate based on the equation (2.61).

For deep circuits, this turns out to have an enormous effect on the measured amplitudes, as any subsequent noisy gate will have multiplicative cause on the reduction of the overall Bloch sphere, thus decreasing the measured amplitudes. This effect can be seen in the following equation:

$$\rho_{total} = \tilde{\mathtt{G}}^d(\rho_{in}) = \begin{pmatrix} x_{in} \\ (2p-1)^d y_{in} \\ (2p-1)^d z_{in} \end{pmatrix} \tag{2.61}$$

where d depicts the circuit depth. Applying equation (2.61) with $p < 0.5$ on the initial problem from section 2.2.6, leads to the graph shown in figure 2.21. This also represents the overall result after applying all previously discussed noise effects (coherent and incoherent).

To summarize, the following derivations compared to the ideal case on the left in figure 2.16 are found:

- Rotation error ϵ leads to oscillation

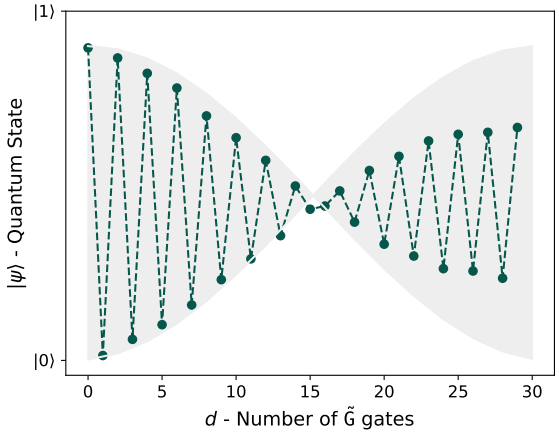


Figure 2.21: Decay of the measurable amplitude over the number of gates d, resulting from bit flip errors by measuring a noisy gate $\tilde{\mathtt{G}}$ based on the equation (2.61).

- Measurement error yields a shift towards the $|0\rangle$ state
- Stochastic nature of QC adds fluctuations modelled by a binomial distribution
- Bit flip error (Incoherent noise) results in decaying amplitude over d

Noise Mitigation

Since QC in the NISQ era is dominated by noise as discussed in previous chapters, there are noise mitigation techniques which can either be integrated directly into the circuit or classically applied in post-processing. Noise mitigation techniques (also referred as *quantum error correction*) which involves changes in the circuit often comes at the cost of additional gates and qubits, required for the correction algorithm.

The width (number of qubits) and depth (longest path of gates in a circuit between the data input and the output) [SP18] of a QFT circuit grows rapidly with increasing sampling rate as discussed in section 2.2.8. Exactly this problem needs to be faced in the application of the QFT algorithm in SR tasks, thus applying error correction using additional gates is considered to be not practical in this thesis, as each additional gate and qubit would increase the resulting noise. Therefore, the following section is restricted to noise mitigation using classical computation.

The most intuitive and yet effective approach is to measure a circuit's output for all possible input combinations and then mitigate noise by multiplying the inverse of the recorded matrix on the circuit output [ACB+20]:

$$|\psi\rangle_{\text{noisy}} = M|\psi\rangle_{\text{ideal}} \quad \overset{mitigate}{\longrightarrow} \quad |\psi\rangle_{\text{ideal}} = M^{-1}|\psi\rangle_{\text{noisy}} \tag{2.62}$$

where M is the recorded matrix. Besides this linear mitigation approach, there also exist more advanced mitigation methods using e.g. ML methods as done in [LMK+21]. However, such methods require a more computing time and therefore are less suitable for an application in SR.

2.2.7. Encoding

An important component in QC represents the information encoding into the quantum computer. In the context of this thesis this includes the encoding of the speech signals to corresponding qubits in the QFT circuit. The encoding often trades robustness against noise and performance regarding the information density. The following three encoding approaches are commonly used:

- Basis (Binary) Encoding,
- Amplitude Encoding and
- Angle Encoding.

By using only the $|0\rangle$ and $|1\rangle$ state of a qubit, *binary encoding* is very robust against noise, but also provides the lowest information density equal to information encoding in classical computers (n qubits are required to encode n bits of information). Given a feature vector $x \in \mathbb{R}^N$, *angle encoding* one can express feature x_i in a quantum state by $|\psi\rangle = \cos(x_i)|0\rangle + \sin(x_i)|1\rangle$. This yields a (nonlinear) cosine kernel [SP18] which can be useful for quantum machine learning (QML) tasks but is also computationally very expensive [LB20] In context of the QFT discussed in section 2.2.8, *amplitude encoding* as explained in the remainder of this section, is commonly used [Zam20]. This approach does not require nonlinear encoding and implementation in quantum hardware is more efficient compared to angle encoding [LB20].

The Amplitude encoding of a normalized vector $x \in \mathbb{R}^N$, $|x|^2 = 1$ describes, according to [LB20] the quantum state of $n = \log_2(N)$ qubits

$$|\psi\rangle = \sum_{i}^{N} x_i |\iota_i\rangle \qquad \iota \in \{0,1\}^n \tag{2.63}$$

where x must be normalized such that [SP18] and $|\iota\rangle$ indicates the computational basis states in similar notation as introduced in equation (2.30). The implementation of amplitude encoding is well explained in [PB11] and should not be described in detail here.

2.2.8. Quantum Fourier Transform

The QFT algorithm describes a set of gates forming the quantum equivalent to the classical FT. It finds application in many popular algorithms, including *Shor's Algorithm* [Sho97]. This section derives the QFT algorithm from its classical counterpart and proofs the speedup gained in the context of QC even against the FFT algorithm using the concept proposed in [CBY20]. For this purpose, the FFT algorithm is derived from the DFT matrix. Afterwards it will be showed, that essential steps of this transformation can be realized on a quantum computer very efficiently.

Following examples are restricted to input vectors of size 2^n, $n \in \mathbb{N}$ due to the constraints of the encoding discussed in section 2.2.7. Starting from the matrix notation of the DFT introduced in equation (2.6) it is useful to analyze the examples for $N = 1, 2, 4$:

$$F_1 = 1 \begin{bmatrix} 1 \end{bmatrix} \quad F_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = H \quad F_4 = \frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \tag{2.64}$$

where one might note that the case $N = 2$ is equivalent to the definition of the Hadamard gate introduced in equation (2.37). For simplicity, it is useful to consider the exponents of the DFT matrix. Matrices using this representation are donated by *tilde* operator, e.g. $\tilde{M}$, in following equations. The examples in the remainder of this section cover the case of $N = 3$, starting with the exponent representation of a $2^3 = 8$-point DFT:

$$\tilde{F}_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 2 & 4 & 6 & 8 & 10 & 12 & 14 \\ 0 & 3 & 6 & 9 & 12 & 15 & 18 & 21 \\ 0 & 4 & 8 & 12 & 16 & 20 & 24 & 28 \\ 0 & 5 & 10 & 15 & 20 & 25 & 30 & 35 \\ 0 & 6 & 12 & 18 & 24 & 30 & 36 & 42 \\ 0 & 7 & 14 & 21 & 28 & 35 & 42 & 49 \end{bmatrix} \tag{2.65}$$

Next, we apply $\mod N$ with $N = 8$ on each of the entries, write the remainder in the matrix and introduce binary enumerated rows. The latter will become important in following steps.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 2 & 4 & 6 & 0 & 2 & 4 & 6 \\ 0 & 3 & 6 & 1 & 4 & 7 & 2 & 5 \\ 0 & 4 & 0 & 4 & 0 & 4 & 0 & 4 \\ 0 & 5 & 2 & 7 & 4 & 1 & 6 & 3 \\ 0 & 6 & 4 & 2 & 0 & 6 & 4 & 2 \\ 0 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \end{bmatrix} \parallel \begin{bmatrix} 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{bmatrix} \tag{2.66}$$

As mentioned in the beginning of this section, the first goal is to derive concept of the FFT from aboves DFT matrix. Therefore an inherent structure must be found, that represents the divide-and-conquer approach, discussed in section 2.1.1. Next, the enumerating bits on the right are resorted in such a way, that they are counted from left to right. Matrices on which this operation is applied, will be denoted with M' in the following equations. The Fourier matrix for $N = 8$ then becomes:

$$\tilde{F}'_8 = \left[\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 4 & 0 & 4 & 0 & 4 \\ 0 & 2 & 4 & 6 & 0 & 2 & 4 & 6 \\ 0 & 6 & 4 & 2 & 0 & 6 & 4 & 2 \\ \hline 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 5 & 2 & 7 & 4 & 1 & 6 & 3 \\ 0 & 3 & 6 & 1 & 4 & 7 & 2 & 5 \\ 0 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \end{array}\right] \parallel \left[\begin{array}{c} 000 \\ 100 \\ 010 \\ 110 \\ 001 \\ 101 \\ 011 \\ 111 \end{array}\right] \tag{2.67}$$

Each quadrant of the matrix can be rewritten using the 2-pointDFT matrix in exponent notation and a new matrix $\tilde{\Omega}$.

$$\left[\begin{array}{c|c} \tilde{F}'_4 & \tilde{F}'_4 \\ \hline \tilde{F}'_4 + \widetilde{\Omega}_4 & \tilde{F}'_4 + \widetilde{\Omega}_4 + 4 \end{array}\right] \tag{2.68}$$

with

$$\tilde{F}'_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 2 \\ 0 & 1 & 2 & 3 \\ 0 & 3 & 2 & 1 \end{bmatrix} \qquad \tilde{\Omega}_4 = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \end{bmatrix} \tag{2.69}$$

where it must be noted, that the top left quadrant in $\tilde{F}'_8$ differs from $\tilde{F}'_4$ by twice of their values due to the normalization by $1/\sqrt{N}$. The ability to represent a DFT matrix by smaller DFT matrices with exactly half of the original size is essential for the FFT algorithm. This allows the original DFT matrix being reformulated as

$$F'_8 = \frac{1}{\sqrt{2}} \left[\begin{array}{c|c} F'_4 & F'_4 \\ \hline F'_4\Omega_4 & -F'_4\Omega_4 \end{array}\right] \qquad \Omega_4 = \operatorname{diag}\left(1, \omega_8^1, \omega_8^2, \omega_8^3\right) \tag{2.70}$$

after applying the exponents accordingly. In the following equations, bold matrices $\boldsymbol{M}$ will indicate, similar to the notation in [CBY20], matrices with exponential dimension 2^n, meaning that $\mathbf{M}_n = M_{2^n}$ would denote a $(2^n \times 2^n)$ matrix.

This allows to generalize the DFT matrix:

$$\mathbf{F}'_n = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{F}'_{n-1} & \mathbf{F}'_{n-1} \\ \mathbf{F}'_{n-1}\boldsymbol{\Omega}_{n-1} & -\mathbf{F}'_{n-1}\boldsymbol{\Omega}_{n-1} \end{bmatrix} = \mathbf{P}_n\mathbf{F}_n \tag{2.71}$$

where P_n is called the *bit reversal permutation matrix* and $\boldsymbol{\Omega}_n$ in the general form of the diagonal matrix:

$$\boldsymbol{\Omega}_n = \Omega_{2^n} := \begin{bmatrix} \omega_{2^{n+1}}^0 & 0 & \dots & 0 \\ 0 & \omega_{2^{n+1}}^1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \omega_{2^{n+1}}^{2^n-1} \end{bmatrix} \tag{2.72}$$

with $\omega_{2^{n+1}} := \exp\left(\frac{-2\pi i}{2^{n+1}}\right)$.

The equation (2.71) is basically the FFT introduced in section 2.1.1 written in matrix form, where the iterative split is hidden in $\mathbf{F}'_{n-1}$ since $2^{n-1} = 2^n \frac{1}{2}$.

The authors in [CBY20] proof that F_n can be factored as:

$$\mathbf{F}_n = \mathbf{P}_n \mathbf{F}'_n = \mathbf{P}_n \mathbf{A}_n^{(0)} \mathbf{A}_n^{(1)} \cdots \mathbf{A}_n^{(n-1)} \tag{2.73}$$

where for $k \in [0 \dots n-1]$, $\mathbf{A}_n^{(k)} = \mathbf{I}_{n-k-1} \otimes \mathbf{B}_{k+1}$ and $\mathbf{B}_{k+1} = \frac{1}{\sqrt{2}} \begin{bmatrix} I_k & I_k \\ \mathbf{\Omega}_k & -\mathbf{\Omega}_k \end{bmatrix} \mathbf{A}_n$ only consists of two non-zero elements per row, so that its matrix-vector product can be computed on $\mathcal{O}(2^n) = \mathcal{O}(N)$.

Combined with the permutations introduced in equation (2.67), the overall complexity becomes $\mathcal{O}(N \log N)$, which corresponds to the complexity of the FFT. Nevertheless the matrix

$$\mathbf{B}_{k+1} = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{I}_k & \mathbf{I}_k \\ \mathbf{\Omega}_k & -\mathbf{\Omega}_k \end{bmatrix} = (\mathbf{I}_k \oplus \mathbf{\Omega}_k)\,(H \otimes \mathbf{I}_k) \tag{2.74}$$

hides a complexity of $\mathcal{O}(N)$ in the direct sum $\mathbf{I}_k \oplus \mathbf{\Omega}_k \in \mathbb{C}^{2^{k+1} \times 2^{k+1}}$ since „the multiplications of a diagonal matrix with a vector has to be performed in $\mathcal{O}(N)$ operations“ [CBY20]. The direct sum between two matrices $A \in \mathbb{C}^{n \times n}$ and $B \in \mathbb{C}^{m \times m}$ is defined as the following $(n+m) \times (n+m)$ diagonal block matrix [CBY20]:

$$A \oplus B := \begin{bmatrix} A & \\ & B \end{bmatrix} \tag{2.75}$$

It turns out that $\mathbf{I}_k \oplus \mathbf{\Omega}_k$ can be written as the Kronecker product of k simpler matrices. For that reason, another matrix is introduced

$$R_n := \begin{bmatrix} \omega_{2^n}^0 & 0 \\ 0 & \omega_{2^n}^1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \omega_{2^n} \end{bmatrix} \tag{2.76}$$

which can be used to decompose Ω_{2^n} such that with

$$(R_n)^{2^j} = \begin{bmatrix} 1 & 0 \\ 0 & \omega_{2^n}^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \omega_{2^{n-j}} \end{bmatrix} = R_{n-j} \tag{2.77}$$

we get

$$\mathbf{\Omega}_n = R_2 \otimes R_3 \otimes \cdots \otimes R_n \otimes R_{n+1} \tag{2.78}$$

Inserting this into $\mathbf{I}_k \oplus \mathbf{\Omega}_k$ yields

$$\mathbf{I}_k \oplus \mathbf{\Omega}_k = \prod_{i=1}^{k} \left[E_1 \otimes \mathbf{I}_{i-1} \otimes \mathbf{I}_2 \otimes \mathbf{I}_{k-i} + E_2 \otimes \mathbf{I}_{i-1} \otimes R_{i+1} \otimes \mathbf{I}_{k-i} \right] \tag{2.79}$$

where

$$E_1 := e_1 e_1^\top = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \qquad E_2 := e_2 e_2^\top = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.80}$$

While the initial idea behind this decomposition was, to improve the complexity of $\mathcal{O}(n)$ from $\mathbf{I}_k \oplus \mathbf{\Omega}_k$, equation (2.79) increased the computational effort. This can be seen by taking a rank-1 tensor $x_n = x_1 \otimes x_2 \otimes \ldots \otimes x_n$ with $x_i \in \mathbb{C}^2$ and multiplying it with each entry in equation (2.79), requiring $\mathcal{O}(n)$ operations. However, the product of this tensor with a diagonal matrix $\mathbf{I}_k \oplus \mathbf{\Omega}_k$ would require $\mathcal{O}(2^n)$ operations.

Nonetheless, the authors in [CBY20] show that the expression in equation (2.79) enables an efficient implementation on quantum computers by utilizing fundamental operations although being computationally more expensive due to redundant terms on a classical computer. To see this we rewrite equation (2.79) into

$$\mathbf{I}_k \oplus \mathbf{\Omega}_k = \prod_{i=1}^{k} I_2 \otimes \mathbf{I}_{i-1} \otimes E_1 \otimes \mathbf{I}_{k-i} + R_{i+1} \otimes \mathbf{I}_{i-1} \otimes E_2 \otimes \mathbf{I}_{k-i} \tag{2.81}$$

where it is conceivable that every matrix except R_n is trivial. However, R_n as stated in equation (2.77) is a well known operation on quantum computers, namely the $\mathtt{R_Z}$ gate introduced in equation (2.42):

$$\mathtt{R_Z}(\Phi) = \begin{bmatrix} 1 & 0 \\ 0 & \mathrm{e}^{i\Phi} \end{bmatrix} = \mathtt{P}(\Phi) \tag{2.82}$$

where $\omega_{2^n} = \mathrm{e}^{\frac{-2\pi i}{2^n}}$ has been used. This allows $\mathbf{I}_k \oplus \mathbf{\Omega}_k$ with $k \in [0 \ldots N-1]$ to be written as a circuit consisting of exactly $\mathcal{O}(N) = \mathcal{O}(n^2)$ rotation gates. For this purpose, $\mathbf{I}_k \oplus \mathbf{\Omega}_k$ is expressed by a controlled $\mathbf{\Omega}_k$ gate, which further decomposes into k controlled phase gates $\mathtt{CP}(\phi)$ introduced in equation (2.45). The last step results from the combination of the $\mathtt{R_Z}$ gate included in $\mathbf{\Omega}_k$ with a control operation. For simplicity, this gate will be displayed without its parametrization angle Φ but only indexed as R_k in subsequent circuit diagrams. This result can be seen in figure 2.22.

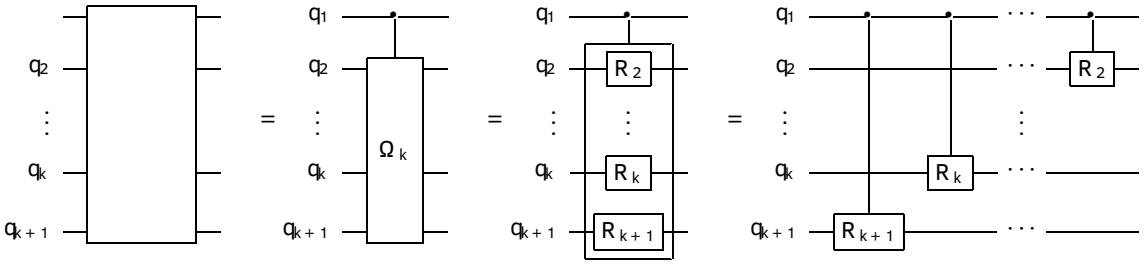


Figure 2.22: Decomposition of $\mathbf{I}_k \oplus \mathbf{\Omega}_k$ first into a controlled $\mathbf{\Omega}_k$ gate, which equals k controlled phase gates $\mathtt{CP}(\phi)$. For simplicity, the parametrization angle Φ is omitted and replaced by the index k, yielding R_k.

We further see that $\mathbf{I}_k \oplus \mathbf{\Omega}_k$ came from the matrix $\mathbf{B}_{k+1}$ in equation (2.74). Thus, the equations (2.73) and $\mathbf{A}_n^{(k)} = I_{n-k-1} \otimes \mathbf{B}_{k+1}$ can be written as a general circuit shown on

the left in figure 2.23.

The permutation matrix $\mathbf{P}_n$ in $\mathbf{F}_n = \mathbf{P}_n\mathbf{F}'_n$ is equivalent to a composition of *swap gates* introduced in section 2.2.2. The right circuit in figure 2.23 shows the swap gate as equality to the permutation matrix $\mathbf{P}_n$.

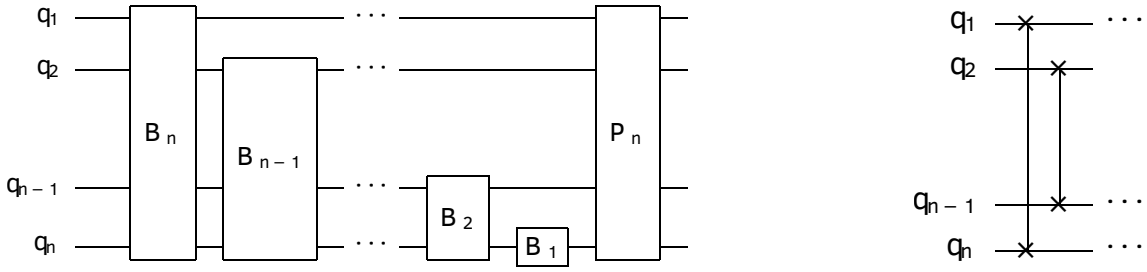


Figure 2.23: Permutation matrix $\mathbf{P}_n$ written in circuit notation from [CBY20].

The gate $\mathbf{B}_{k+1}$ in the left circuit in figure 2.23 can be resolved by the circuit in figure 2.22 and a single Hadamard gate, according to equation (2.74).

Combining the previous steps and circuits, the overall QFT results in a circuit as visualized in figure 2.24.

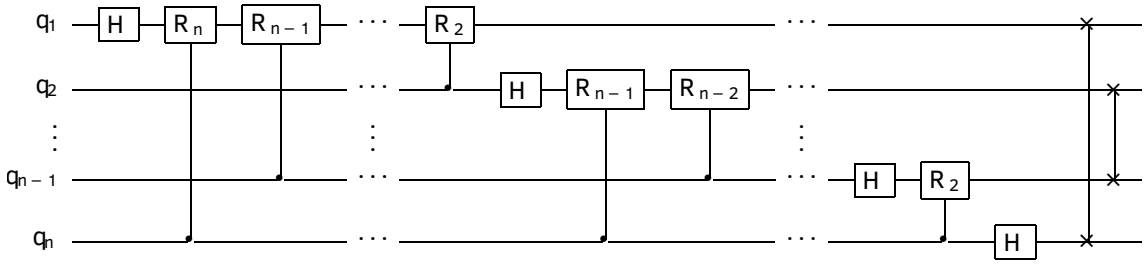


Figure 2.24: Overall QFT circuit from [CBY20]

The complexity of the QFT circuit depends on its overall depth, given by the number of qubits, which in turn constrains the required number of gates for each operation (matrix). This depicted in table 2.2.8 which shows the count of each type of matrix in the QFT. It should be noted that on common NISQ devices, the number of qubits is typically in the order of $1 \dots 65$ [Qua21].

Matrix	Gate	Count
$\mathbf{P}_n$	CNOT	$\lfloor 3n/2 \rfloor$
$\mathbf{A}_n^{(0)} \cdots \mathbf{A}_n^{(n-1)}$	Hadamard	n
$\mathbf{A}_n^{(0)} \cdots \mathbf{A}_n^{(n-1)}$	$\texttt{CP}(\Phi)$	$n(n-1)/2$

This shows that the prevailing complexity is hidden in the controlled `CP` gates with $\mathcal{O}(n^2) = \mathcal{O}(\log N^2)$ which is better than $\mathcal{O}(N \log N)$ of the FFT. The QFT applied on an input signal x will be represented by $\mathcal{F}_{\text{QFT}}$ in following equation, similar to the classical FT. It should be noted, that in contrast to the classical equivalent, the QFT inherently requires the adequate encoding.

2.2.9. Quantum Machine Learning

„*Quantum machine learning* summarizes approaches that use synergies between machine learning and quantum information“ [SP18]. QML as a relatively new discipline is, according to [Hid19], characterized by the type of data and the type of algorithm utilized in a specific scenario. These types are either of classical (C) or quantum (Q) nature and lead to four possible approaches of how to combine QC and ML as depicted in figure 2.25.

The topic of this thesis can, on the one hand be categorized as a *Quantum-Classical* approach where quantum systems are utilized to generate data and processed and interpreted by classical algorithms afterwards. On the other hand, the data origin is classical and processing it with QFT would rather fall into the area of *Classical-Quantum*, leading to a mixture of *CQ* and *QC*. An example for *pure* Classical-Quantum, would be the generation of quantum circuits using machine learning algorithms running on classical systems. In the case of *CC*, classical data is being processed classically, whereas *QQ* „looks at *quantum data* being processed by a quantum computer“ [SP18].

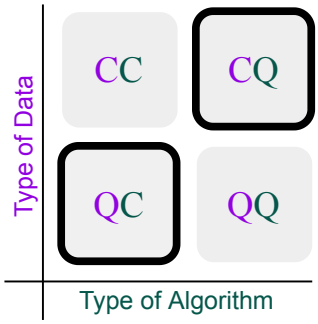


Figure 2.25: QML approaches characterized by the type of data and type of algorithm each either **C**lassical or **Q**uantum. Image derived from [SKX+19]. Approaches which match the topic of this thesis are highlighted with a bold border.

Developing algorithms in the field of QC also yields two strategies. The *translational approach* aims to „reproduce the results of a given model [...] but to outsource the computation to a quantum device“ [SP18]. However, the *exploratory approach* rather exploits the processing on quantum devices in general striving for innovative methods. In this work, an existing approach, namely speech recognition based on spectrograms of a signal, is partially outsourced to a quantum computer. Therefore, this thesis investigates a translational approach.

2.3. Related Research

In [YQC+20b], the authors propose a "novel decentralized feature extraction approach [...] to address privacy preservation issues for speech recognition" [YQC+20b]. Utilizing Google's speech commands dataset [Tea17] consisting of "65.000 one-second long utterances of 30 short words" [Tea17], [YQC+20b] first generate a 60-band Mel-Spectrogram using a classical 1024-point STFT and afterwards process the data using a *quantum convolutional (quanv) layer* as shown in figure 2.26.

The results are fed into a local *Attention-RCNN* model, for classification in a supervised learning setup [YQC+20b]. Figure 2.26 shows the overall system.

These authors also introduce the *self-attention U-Net* architecture [YQC+20a] which is

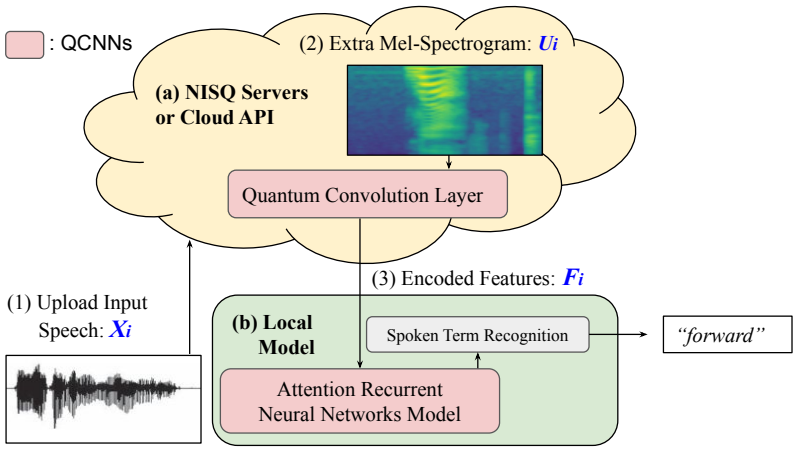


Figure 2.26: "QML-AM" architecture from [YQC+20b]. The Mel-spectrogram of a speech signal is classically generated and afterwards post-processed by a NISQ device. The result is then classified by a local ML model.

applied prior to the *RCNN* and is shown in figure 2.27.

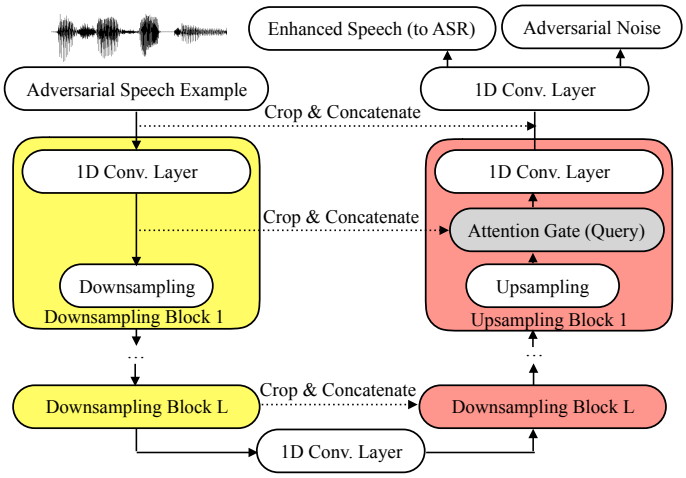


Figure 2.27: Self-attention U-Net architecture from [YQC+20a]

The quantum convolution neural network (QCNN) architecture used in [YQC+20b], including the U-Net and quanv layer, is shown in figure 2.28. After these blocks, a 2-D convolutional layer followed by a two-layered LSTM attached to an attention mechanism perform the location invariant classification in the spectrogram input. The final prediction is processed by two dense layers and assessed by a categorical cross-entropy loss. However, since this work does not seek to make significant improvements to the above architecture, the individual layers are not discussed in detail here.

Quantum convolutional (quanv) filters have a similar functionality to classical convolutional filters, in the sense that they are also shifting over the input data. While classical filters usually map a region of $(n \times n)$ pixels to a single value by a calculating a weighted sum of theses pixels, the circuit in [YQC+20b] returns c channels ((2×2) proofed as the best choice out of $c \in [1, 2, 3]$) with the input data processed by a random constitution of quantum gates. This filtering process is depicted in figure 2.29.

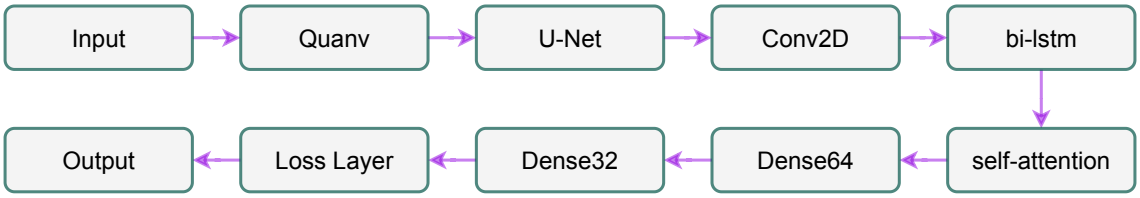


Figure 2.28: Original QCNN structure proposed in [YQC+20b]. The input is processed by a quanv layer and then passed to a U-Net from figure 2.27 and 2-D convolutional layer. A two-layered LSTM in conjunction with an attention mechanism and two final dense layers performs the location invariant classification.

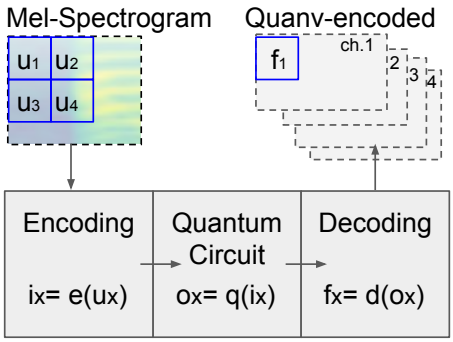


Figure 2.29: Quantum convolutional (quanv) layer from [YQC+20b]. A 2×2 region of a spectrogram is encoded and processed by a random quantum circuit. The output is then distributed over 4 channels.

Their quantum algorithm is small enough to fulfill the restrictions of nowadays NISQ devices while enhancing the NN accuracy by 0.4 % compared to the plain UNet-Attention-RCNN (94.72 %) as shown in table 2.2.

RNN_{UAtt}	94.72 ± 0.23	$176,535$	(32-bits)
$\text{Conv} + \text{RNN}_{\text{UAtt}}$	94.74 ± 0.25	$180,595$	(32-bits)
$\text{Quanv} + \text{RNN}_{\text{UAtt}}$	95.12 ± 0.18	$180,575$	(32-bits + 4-qubits)

Table 2.2: Benchmark results of the SR pipeline from [YQC+20b] ($\text{Quanv} + \text{RNN}_{\text{UAtt}}$) in comparison with classical architectures.

Reducing the number of gates, more specifically the number of `CP` gates of the QFT is a common approach to alleviate the effects of noise on measurement results, compare section 2.2.6. In [NB13] the authors examine the minimum number of rotation gates needed, to achieve similar performance with a banded n-qubit N-QFT compared to a streamlined version as used in Shor's algorithm. The results found in their work showed that a bandlimited QFT can perform well in this specific application which motivates further investigations of their approach in the context of SR tasks.

3. Approach

This chapter presents the main idea of the thesis, the application of the quantum Fourier transform (QFT) in a speech recognition (SR) pipeline, and explains the chosen architecture in detail.

In section 3.1, the application of the QFT algorithm on classical data is discussed and potential problems are pointed out. Section 3.2 introduces the short time quantum Fourier transform (STQFT), the quantum equivalent of the Short Time Fourier transform (STFT) necessary for generating spectrograms. These spectrograms are then used as features for a neural network in a SR pipeline presented in section 3.3.

3.1. Quantum Fourier Transform for Signal Processing

This section describes the application of the QFT, introduced in section 2.2.8. From a classical perspective, the Fourier transform (FT) is expected to not only represent the frequencies of a signal (assuming an infinite observation window), but also to correlate directly with its energy. However, in quantum computing (QC) these assumptions can not always be taken as granted as discussed in the following sections.

3.1.1. Encoding

Starting from the very beginning of constructing a quantum circuit, the first task is to convert classical data into quantum states. In section 2.2.7, amplitude encoding is presented as a suitable encoding scheme in conjunction with the QFT. This encoding approach, as well as others, requires the *normalization* of the input data to fulfill the property from equation (2.22) of a quantum state which in turn vanishes information regarding the signals absolute energy. Additionally a second normalization is required for obtaining the probabilities from the histogram as discussed in section 2.2.3.

Another point of consideration is that, similar to the fast Fourier transform (FFT), the QFT requires the number of samples to match 2^n, where n depicts the number of quantum bits (qubits). In general it cannot be expected that the speech samples fulfill this criterion, thus zero-padding as introduced in paragraph 2.1.1 is required.

3.1.2. Transformation Accuracy

Amplitude Accuracy The nature of QC only allows stochastic reasoning about the measurement amplitudes as discussed in section 2.2.1. This holds true as well for the case of the QFT, thus accurate frequency amplitudes as achievable with the FT will not be possible. In addition to the stochasticity, gate- and measurement errors explained in section 2.2.3 will distort the frequency amplitudes even further.

Frequency Accuracy In contrast to the amplitude accuracy, the QFT can achieve the same frequency accuracy as the FFT if noise is neglected. This means, that if sampling rate and frequency resolution fulfill the requirements stated in section 2.1.1, a signal could be fully reconstructed from a spectrum generated by the QFT. This is conceivable, since the calculations in the QFT are identical to the FFT as shown in section 2.2.8 and only differ in the realization on the device itself. However, actually running the QFT on a quantum device, would inevitably lead to noise influences in the measurement results. As discussed in section 2.2.6, this noise, either caused by the measurement device or quantum gates, could lead to states which deviate from the expected ones and in case of the QFT this would then result in an inaccurate spectrum.

Low-Frequency Tendency The quantum systems used within this thesis map the ground- and excited state on the $|0\rangle$ and $|1\rangle$ state respectively [1]. As stated in section 2.2.1, a qubit always tends towards its *ground state*. In the QFT case where each qubit is measured to have the $|0\rangle$ state corresponds to the 0 Hz frequency. The mapping between frequency and a qubit's index correlates in such a way that qubits with high indices are mapped to high frequencies. Measuring a qubit accidentally to be in the $|0\rangle$ state therefore results in a much lower frequency since the overall state has a completely different binary representation. This yields the effect, that the frequency spectrum measured, has a general tendency towards low frequencies. Due to the nonlinear mapping between an index in the binary state representation and a frequency, higher frequencies are less likely to be measured, compared to lower frequencies. This behavior enhances the low frequency tendency and is demonstrated in example 3.1.1.

Example 3.1.1 *Case A depicts the ideal measurement, where q_1 and q_3 of a 16-QFT (4-qubits in total) are measured to be in the $|1\rangle$ state. This results in the overall state $|1010\rangle \underset{int}{\hat{=}} 10 = f_k$ where the binary representation of the state has been transferred to an integer index. Assuming a sampling rate f_s, the actual frequency can be obtained from the frequency index f_k by $f = f_k \frac{f_s}{2^4}$*

$$\begin{aligned} &\textit{Case A:} \quad q_0 = |0\rangle \quad q_1 = |1\rangle \quad q_2 = |0\rangle \quad q_3 = |1\rangle \longrightarrow |1010\rangle \longrightarrow f_k = 10 \\ &\textit{Case B:} \quad q_0 = |0\rangle \quad q_1 = |1\rangle \quad q_2 = |0\rangle \quad q_3 = |0\rangle \longrightarrow |0010\rangle \longrightarrow f_k = 2 \end{aligned}$$

In case B, qubit q_3 has been erroneously measured to in the $|0\rangle$ state. Although only a single bit has flipped, the resulting frequency index $f_k = 2$ is completely different from case A.

On real quantum hardware, each qubit has an individual error rate, thus above effects on the spectrogram might not fully follow the expected behavior. However, the mitigation approach discussed in the following section 3.1.3 is well suited to address the low-frequency tendency problem.

[1] The mapping of the ground- and excited state to the $|0\rangle$ and $|1\rangle$ state respectively must not necessarily be the case for all quantum systems. The relation can also be inverted.

3.1.3. Noise

The following section describes two approaches to reduce noise in the measurement results. While the *angular filter* method achieves this by reducing the overall gate count of the QFT, the *noise subtraction* approach aims to reduce the noise from the histogram output directly.

Angular Filter Section 2.2.6 clearly shows that gate errors and noise in general increases consistently with the number of gates and qubits involved in a circuit. Considering the QFT the number of rotation gates `CP` is directly constrained by the number of qubits involved following $n(n-1)/2$ as stated in table 2.2.8.

However, the number of rotation gates is defined by the size of the transformation matrix ($N \times N = 2^n \times 2^n$) which is derived from the number of samples n. Taking e.g. a $n = 10$-qubit QFT, thus $N = 2^n = 1024$ samples, then the number of rotation gates is $10(10-1)/2 = 45$. From the discussion in section 2.2.6 it is conceivable that gates with small angles of rotation ϕ in particular are prone to error.

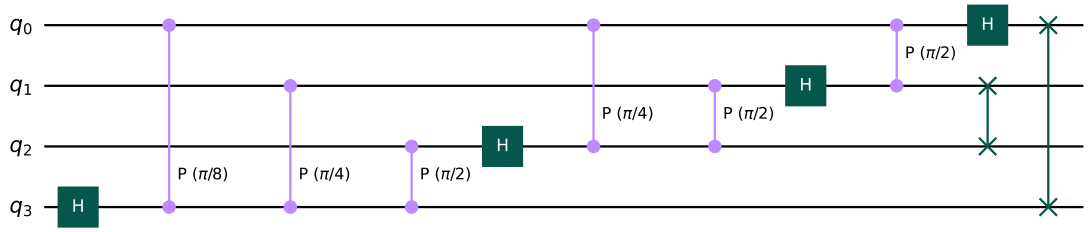
The authors in [NB13] therefore investigated a *minimum angle* for rotation gates used in QFT. Omitting certain rotation gates implies a banding on the QFT, since each rotation gate represents a sinusoidal function of a certain frequency which is being controlled by b neighboring qubits. Naturally, b is limited by the number of neighboring qubits of the specific gate and the number of qubits in the QFT as an upper bound. However, if b is restricted to a fixed number $\leq n$, high frequencies are considered in the transformation anymore, thus resulting in a bandlimited QFT.

While the research in [NB13] focuses on the performance of the QFT in *Shor's Algorithm*, it motivates similar investigations in case of SR. Therefore an *angular filter* is introduced and parameterized by an *minimum rotation angle* allowed for the qubits.

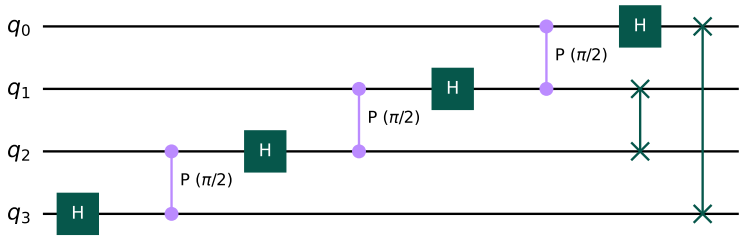
Example 3.1.2 *Figure 3.1 shows a* 16*-QFT as full implementation with* $b = 1$ *and as bandlimited variant with* $b = 3$*. In the latter case,* 3 *rotation gates with an angle smaller than* $\pi/4$ *were omitted which reduced the overall rotation gate count by* 50 %*.*

Instead of a bandwidth parameter as introduced in [NB13], a minimum rotation angle mr as mentioned above is used to parameterize the filter instead. The rotation angle can be obtained from the bandwidth b by $\mathsf{mr} = \pi/2^{n-b}$ where n depicts the number of qubits in the 2^n-QFT. This approach was chosen, since the bandwidth parameter directly constraints the number of rotation gates omitted, whereas a specific minimum rotation angle is invariant in regards of the size of the QFT. Therefore, the resulting impact of the angular filter on the spectrum is expected to be independent from the chosen window size, i.e. the size of the QFT.

Noise Subtraction The noise mitigation approach mentioned in section 2.2.6 can also be extended, such that instead of all possible input combinations on an empty circuit, a constant signal $\bar{x}$ is fed into the QFT. The transformation of a constant signal ideally results in a single peak at $f = 0$ thus any deviation can be regarded as *background noise*.



(a)



(b)

Figure 3.1: 3.1a: 16-QFT Circuit, full implementation ($b = 1$)
3.1b: 16-QFT Circuit, bandlimited ($b = 3$)
Controlled phase gates displayed by `CP`(Φ.)

This background noise $\bar{X}$ is then subtracted from the noisy measurement counts $\tilde{X}$ which is depicted in equation (3.2):

$$\begin{aligned} X &= \mathcal{F}_{\mathrm{QFT}}(x) \\ \bar{X} &= \mathcal{F}_{\mathrm{QFT}}(\bar{x}) \end{aligned} \tag{3.1}$$

where $\mathcal{F}_{\mathrm{QFT}}$ describes the QFT introduced in section 2.2.8, applied on the input signal x and a constant signal $\bar{x} = [1, ..., 1]$ matching the length of x.

$$X_k = \begin{cases} \tilde{X}_k - \bar{X}_k & \forall \quad \tilde{X}_k \geq \bar{X}_k \\ 0 & \text{else} \end{cases}, \quad k \in [1 \dots K] \tag{3.2}$$

The noise mitigated output is then obtained in X where negative values are discarded and set to zero for each frequency bin $\tilde{X}_k$.

It should be noted, that due to the normalization requirement, a zero signal is not a valid choice for a constant signal. The QFT of $\bar{x}$ $\mathcal{F}_{\mathrm{QFT}}(\bar{x})$ is then expected to return the deviation from an all-zero frequency domain signal except for a single peak at $f = 0$. These deviations can result from gate- or measurement errors which have a unique probability on each quantum device. By using the same circuit for the measurement of the background noise and the computation of the actual signal's spectrum, this *Noise Subtraction* approach can therefore tackle device specific noise characteristic.

This mitigation strategy simultaneously counteracts the effect of the low-frequency tendency as mentioned in the previous section in paragraph 3.1.2, as it efficiently mitigates the 0 hertz frequency.

3.2. Short Time Quantum Fourier Transform

Similar to the classical case, the processing of time-variant signals requires subsequent application of the FT. While approaches such as the wavelet transform (WT) or wigner ville distribution (WVD) would be conceivable (see discussion in chapter 5), the STFT is well-established in the SR domain, fulfills the required time-frequency resolution, and enables an efficient implementation at the same time. Therefore, by investigating similar options on a quantum computer, this thesis introduces the quantum equivalent of the STFT, the STQFT.

Similar to the classical case, the STQFT consists of multiple QFTs, each of which is applied to a windowed signal. Windowing however can be done classically using common filter methods as mentioned in section 4.3. Besides this additional step, application of the QFT does not differ from 3.1.

Signal Threshold Filter The noise, which is introduced in section 2.2.6, also exists when the QFT is applied on a constant signal, as discussed in the previous chapter. Since the length of each sample of the dataset used within this thesis and presented in section 4.1 lasts for exactly 1 second and most utterances are significantly shorter, some QFTs within the STQFT will transform zero-like parts of the speech signal.

The normalization, inherent with the amplitude encoding as discussed in section 2.2.7 used in the QFT, will, applied on a zero-like signal, result in very large values. This extremely noisy signal is then fed into the QFT whose output in turn is normalized again before being handed over to the neural network (NN). While noisy input data can improve the generalization capability of a NN [Bis95], this only holds true for a certain amount of noise, applied homogeneously on the whole signal. However, it is to be expected that the artifacts mentioned above occur only in parts of the signal that are normally also zero in the spectrum. Therefore, the NN would be forced to learn a separation between the actual relevant signal and the noise which might be challenging in regards of the given architecture from [YQC+20b].

This problem can be addressed by introducing a *signal threshold filter* described in equation (3.3). If a signal's maximum amplitude is below a fixed threshold, the a zero spectrum is returned instead of the actual QFT output. It is conceivable that this may not only improve the overall speed of STQFT, since many QFTs can simply be omitted, but also effectively suppress the spectral artifacts mentioned above.

$$X = \begin{cases} \mathcal{F}_{\mathrm{QFT}}(x) & \text{if} \quad max(x) \geq \mathsf{sr} \\ 0 & \text{else} \end{cases} \tag{3.3}$$

Normalization As discussed in section 3.1.1, the QFT requires normalization of the signal. In context of the STQFT, where multiple QFTs are used to build up a spectrogram, this

results in the effect, that e.g. raising or lowering voice in a speech signal is not reflected by the STQFT. This in turn could yield an issue in the classification task with a NN where changes in the amplitude between neighboring QFT windows might be missing features.

3.3. Hybrid Quantum Speech Processing

The approach of a quantum-enhanced SR framework is visualized in figure 3.2. Typical building blocks fundamental for SR, the speech signal, STFT and Mel-spectrogram, are depicted by grey blocks with a dashed border. The authors from [YQC+20b], whose approach is explained in section 2.3, extended this pipeline by a quantum convolutional (quanv) layer and a modified attention recurrent neural network (RNN), displayed as grey blocks with solid border. Furthermore, they also tested a learnable convolutional layer as comparison to the quanv layer. Last but not least, the approaches developed in this thesis are visualized by the green blocks. These include the STQFT involving the angular filter and noise subtraction method and the Pixel-Channel-Mapping as replacement for the quanv layer. Vertically lined up blocks represent interchangeable components of the pipeline. This means that the existing STFT can be replaced by the STQFT independent from the methods applied after the Mel-Spectrogram. This also shows that the STQFT presented in this thesis and methods optimized specifically for speech recognition can also be used flexibly in other frameworks.

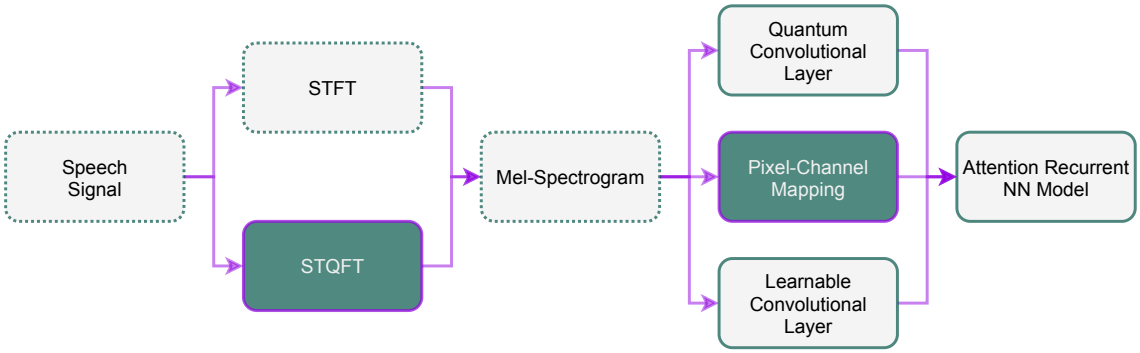


Figure 3.2: Approach to integrate the STQFT in an existing SR pipeline. Typical blocks for such a pipeline are depicted by a dashed border and blocks which are extended or modified by the authors in [YQC+20b] by a solid border. The STQFT, as a replacement for the STFT and the Pixel-Channel-Mapping developed in this work are displayed with a green background.

Quantum Convolutional Alternatives The approach of using a quanv layer directly after the Mel-spectrogram as depicted in figure 3.2 improved the STFT based feature recognition compared to a classical convolutional layer in [YQC+20b]. They found that their network learned „much more correlated and richer acoustic features“ [YQC+20b] by using a randomized quantum circuit as filter. However, it is conceivable, that this finding might not hold true in case where the spectrogram is generated using the STQFT, since the spectrogram obtained from this transformation was already computed by a quantum processing algorithm.

The most intuitive way to test this assumption is by using an identity circuit (1:1 mapping between input and output) represented by the *Pixel-Channel-Mapping* component in figure 3.2. This Pixel-Channel-Mapping is then used as a replacement for the random circuit from [YQC+20b]. For the sake of simplicity the identity circuit can be realized by a *classical* mapping of a $(k \times k)$ image (spectrogram) region directly onto $c = kk$ channels.

Figure 3.3 shows this filter approach. By using a stride of $s = 2$, as in [YQC+20b], the output size results in $(n/s \times n/s \times c)$.

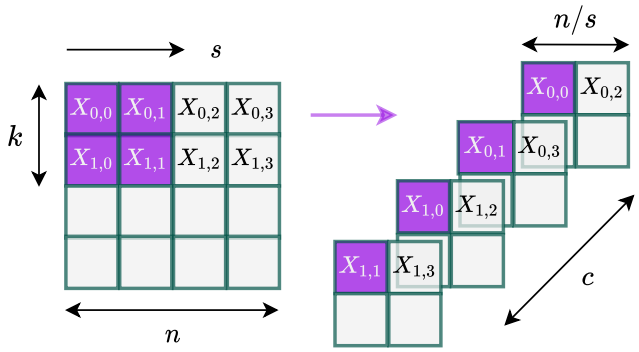


Figure 3.3: Pixel-Channel-Mapping Filter as replacement for the quanv layer. The values of an area $(k \times k)$ inside the $(n \times n)$ spectrogram X are directly copied into separate channels $c = k$ of size $(n/s \times n/s)$ with $s = 2$.

Another conceivable approach, is to implement the quanv layer as a *classical convolutional layer* with learnable weights stored in a matrix W as in figure 2.1. This matrix will then map the image region highlighted on the left in figure 3.3 to the channel values on the right. This layer is then integrated into the NN as suggested in [YQC+20b]. However, this might not yield better results, since the perceptive field of this layer (a 2×2 kernel required to match the output shape to the quanv layer) is relatively small, thus no obvious feature extraction is expected.

4. Validation

This chapter presents the implementation results and evaluations of the quantum Fourier transform (QFT) and short time quantum Fourier transform (STQFT) in comparison with the fast Fourier transform (FFT) and Short Time Fourier transform (STFT) for reference. Furthermore, performance evaluations of the speech recognition (SR) pipeline including the STQFT for spectrogram generation are shown and parameters of the neural network (NN) as well as the STQFT adapted. Finally, the resulting test-accuracy of the NN is presented and the findings from this section are discussed.

The generation of test signals as well as the data for training the NN are described in section 4.1. Afterwards, section 4.2 describes the implementation of discrete Fourier transform (DFT), FFT and QFT followed by the two short time transformations (STTs), the STFT and the STQFT in section 4.3. The final connection to the NN and corresponding test results are presented in section 4.4.

Test Framework Besides the implementation of the QFT in a SR pipeline, this thesis also evaluates the performance of the NN based on certain configurations of hyperparameters. These hyperparameters involve for example the minimum rotation angle mr of the angular filter, the signal threshold st and general parameters of the NN such as the batch size or epochs described in paragraph 2.1.2. Consistent and comprehensible test results are crucial for further investigations. To ensure this, a flexible *test framework* was designed and implemented in which various signals can be applied to either the DFT, FFT, QFT, STFT or STQFT.

For the purpose of archiving and reproducibility, a *versioning mechanism* has been designed which, together with the source control management (SCM) software *Git*, allows the experimental results and generating code to be restored retrospectively. Exported results of experiments can then be viewed afterwards via a *viewer*. An objective comparison between the results of e.g. the FFT and QFT is mandatory, however, there were no suitable comparative methods for achieving this, which is why a *grader* was designed and developed. This test framework is shown in figure 4.1, where in the most abstract form, a signal x is fed into a transformation $\mathcal{F}$ and the result is evaluated by a grader $\mathcal{L}$. This signal can either be synthetic (harmonic or chirp) or real speech sample. The DFT, FFT and QFT can then be applied on the harmonic signal, or are provided to the STT framework if a time-variant signal (chirp or speech sample) should be transformed. In case of the QFT, noise mitigation can be applied. The STFT and STQFT are optionally post processed to yield e.g. the Mel-spectrogram. If the transformation was time-invariant, the spectrum can be evaluated with the Grader by comparison with a reference spectrum. Certain manifestations of these signals and transformations are discussed in following sections.

Simulation Variations Key component of running any quantum circuit is the choice of either simulating the circuit or evaluating it on a real device. However, in case of simulation, either *mock backends* (including the approximated noise and structure of the real device), *noise models* (only gate and readout errors of a specific device but with ideal qubit layout),

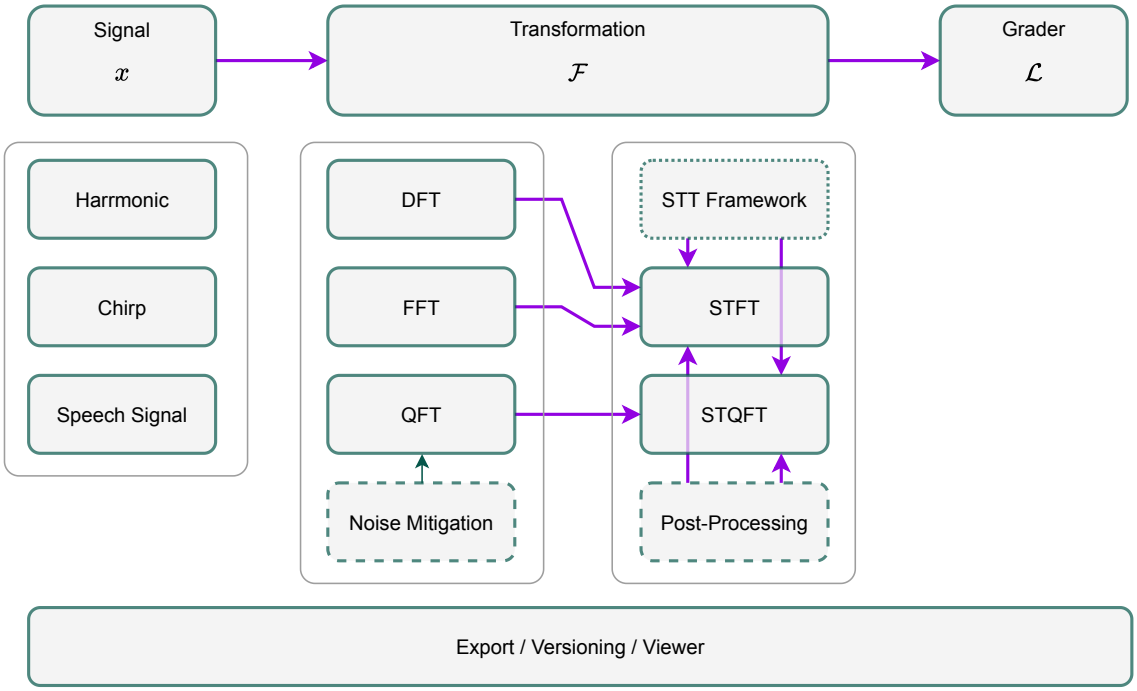


Figure 4.1: Framework for validation of individual transformations. In the most general form, a transformation $\mathcal{F}$ is applied on a signal x and the result is evaluated by a grader $\mathcal{L}$. The signal can either be synthetic (harmonic or chirp) or a real speech sample. If the signal is harmonic, then a DFT, FFT or QFT can be applied. In case of a time variant signal, these transformations are carried out as STFT or STQFT and yield a spectrogram. In case of the QFT and STQFT noise mitigation can be applied.

or *plain simulation* (no noise, ideal qubit layout) can be used.

In many cases, results obtained from the simulated devices are sufficient and using them instead of the real devices is more convenient since the latter has a very limited number of publicly available quantum bits (qubits) and often also involve long waiting times. Especially in the context of the STQFT, where several computations of the QFT are required for a single signal, this can become an issue. The chosen backend device (either simulated or real) is then used to perform the QFT and also serves for the noise mitigation approach as discussed in section 3.1.3. The usage of the exact device either in simulation or as a real quantum device will be stated and justified together with the experiments in later sections.

Tools All quantum specific implementations were done using the *python* 3.8 [VRD09] framework *qiskit* [AAA+21], except for the existing implementations from [YQC+20b] using *pennylane* [BIS+20].

For numerical methods, *numpy* [HMvdW+20] and *scipy* [VGO+20] were used in addition to python's built-in *math* library. Plotting was done with *mathplotlib* [Hun07] and *qutip* [JNN13].

IBM Quantum [Qua21] devices were used to perform experiments on real quantum

computers which are accessed via IBM's Python application program interface (API). These devices include:

- ibmq_quito
- ibmq_guadalupe
- ibmq_melbourne
- ibmq_casablanca

The specific device properties are stated together with the experiments later in this chapter. However, the views expressed are my personal ones and do not reflect the official policy or position of IBM or the IBM Quantum team.

Simulation of quantum circuits was performed in a multiprocessing manner on two "Intel(R) Xeon(R) Gold 5118" central processing units (CPUs) at 2.30 GHz with 376 GB RAM while training tasks of the NN were executed on "NVIDIA V100" graphic processing units (GPUs).

Source Code The code for the experiments as well as their results is available on Github and is split into the following repositories:

- Main Repository [Str21a]: Code for section 4.4.
- STQFT Repository [Str21c]: Code for transformations in section 4.2 and explicitly section 4.3 as well as for the synthetic signals in section 4.1.
- STQFT-Data Repository [Str21d]: Experiment results including plots and parameters.
- QCNN Repository [Str21b]: Original QCNN repository from [YQC+20b] with the modifications discussed in this section and abstracting layers.

4.1. Signal Generation and Data Acquisition

At the very beginning of the pipeline shown in figure 4.1 a general signal component must be instantiated. For flexibility, such a signal can be instantiated either as a harmonic oscillation, chirp signal, or directly from an audio file. Choosing an appropriate signal type is not only required, for example, to investigate the QFT separately from the STQFT, but also useful to investigate properties of each transformation based on simple signals instead of e.g. complex speech signals.

In each case, a specific sampling rate has to be specified unless provided from the a speech file directly. Furthermore, the signal duration can be specified by the number of samples or the time in seconds. Reducing or zero-padding the signal to a specific length is particularly important in the context of the STQFT, since the encoding scheme requires the number of samples (window length) processed by each QFT to match 2^n with n being the number of qubits as discussed in section 3.1.1. This in turn requires the overall number samples in the speech signal to be an integer multiple of the window length.

The remainder of this section will describe the types of signals used for evaluation of the QFT and STQFT including their properties. These signals, which consist of synthetic data such as *Harmonic Oscillations* and *Chirps*, as well as real data such as *Speech Signals*, are shown on the left in figure 4.1

Harmonic Oscillation A harmonic signal as shown in figure 4.1 can be composed of multiple harmonic oscillations, each with a amplitude, phase shift and frequency:

$$x = \sum_i a_i * \sin(2\pi f_i * (t - p_i)), \quad a_i, f_i, p_i \in \mathbb{R}, \quad i \in \mathbb{Z} \tag{4.1}$$

where a_i, f_i and p_i are the i-th amplitude, frequency and phase, respectively, that make up the total harmonic signal. The signal x is represented as a continuos signal for simplicity, although the sampled variant is used in the actual time invariant transformation.

Chirp A linear chirp signal is used to validate the functionality of time-variant transformation methods. The framework provides parameters for start and stop frequency, as well as a time offset and utilizes *Scipy*'s built-in chirp method.

Speech Signal The NN of [YQC+20b] is trained on the *Google's speech command dataset*. In order to ensure the comparability of the results, this dataset is also used in the present thesis. It contains 65000 one-second long utterances of 30 short words provided by many different people [Tea17]. The following labels are selected from the dataset following the setup in [YQC+20b]:

- left
- go
- yes
- down
- up
- on
- right
- no
- off
- stop

This subset of labels poses significant challenges for SR in general while representing a realistic problem [FTYA21]. It consists of 23682 speech samples which are further divided into *training*, *validation* and *test* sets as discussed in paragraph 2.1.2. For the latter, 10 % of the dataset were extracted and the training-test-split of 1/5 from [YQC+20b] has been carried over. Thus, the sets include the following number of samples such that:

- Training set: 17048
- Validation set: 4262
- Test set: 2372

The test set was separated from the dataset before any experiments were conducted and only used for the final performance evaluation. In each experiment with the NN, the training and validation sets are drawn randomly from the remainder of the dataset.

4.2. Transformations

Transformations such as the QFT and STQFT can within the presented framework be carried out on the data presented in the previous chapter. According to the framework

in figure 4.1, these transformations are interchangeable thus no additional preparation is necessary for applying e.g. a QFT instead of a FFT on a given signal. In general and for the remainder of this chapter, the DFT, FFT and QFT are referred to as time invariant transformations since the resulting spectrum contains no temporal information. In contrast, the STFT and STQFT are referred to as STTs and provide time variant spectral information.

The initial parametrization, which is only necessary in case of the QFT, is achieved by the instantiation of these transformations as objects. This abstraction seems to be complex on first sight compared to directly setting up the transformation at the time of applying it on the signal, but enables efficient utilization of the QFT within the STQFT later. Furthermore, the interchangeability between the transformations allows for an efficient setup of experiments. In the remainder of this section, the time invariant transformations are presented.

4.2.1. Discrete Fourier Transform

For the sake of completeness, a DFT was implemented. However, due to the high computational complexity of this transformation, see discussion in paragraph 2.1.1, it is not used in further experiments. Nonetheless the DFT has been implemented to prove the correctness of the initial FFT implementation.

4.2.2. Fast Fourier Transform

The FFT was implemented to provide a reference against the QFT. Implementation was adapted to the approach from Cooley and Tukey in [CT65]. No further parameters are required for instantiation. Therefore, the FFT smoothly integrates within the presented framework.

4.2.3. Quantum Fourier Transform

The QFT was implemented following the theoretical derivations from section 2.2.8 and the work in [Zam20] was used as a starting point. Compared to the DFT and FFT, the QFT requires a more complex implementation. A schematic representation of necessary parts for the computation of the QFT is depicted in the diagram 4.2.

To carry out a QFT, three essential tasks must be fulfilled:

- *Load Backend*: Every quantum circuit requires a backend for its execution. This can either be the (*qiskit*) Simulator (optionally parameterized by a mock backend or noise model) or a real (IBM) quantum device accessed through a provider.
- *Instantiate*: Instantiation of a QFT requires the encoding of (classical) information into quantum states. This encoding is stored in an *encoding circuit* after the signal was checked to not be a zero signal as explained in section 3.1.1 and optionally a signal threshold is applied. In addition to the encoding circuit, the actual QFT circuit is generated and the angular filter as discussed in paragraph 3.1.3 is optionally applied. Since the encoding- and QFT circuit can be transpiled separately, a QFT

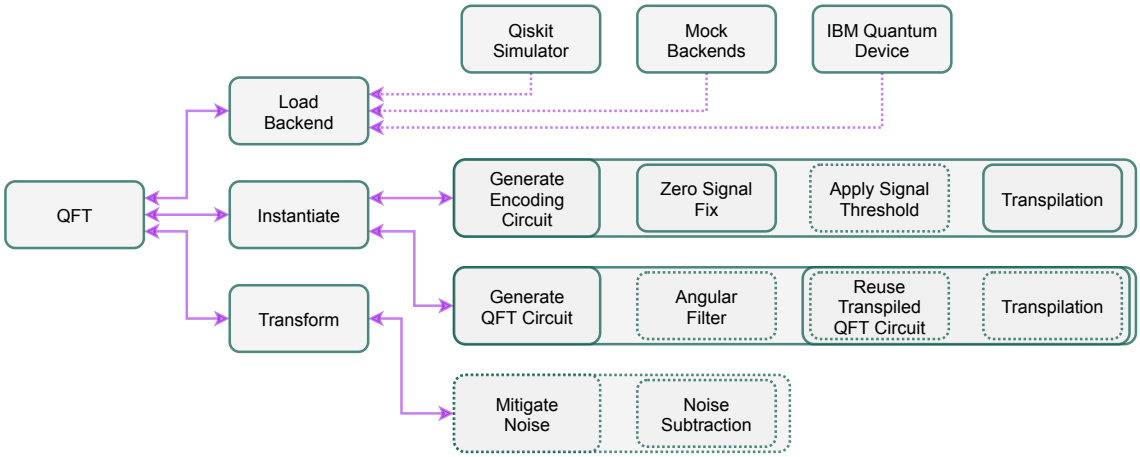


Figure 4.2: Implementation of the QFT with an abstract representation of available methods. The requirements for computing an QFT include loading a desired backend, instantiating the encoding- and QFT-circuit and finally the actual execution of the transformation with optional noise reduction. Details explained in listing 4.2.3.

circuit can be loaded, if existent, to reduce computational overhead. At execution time both, the encoding and actual QFT circuit, will be treated as a single circuit.

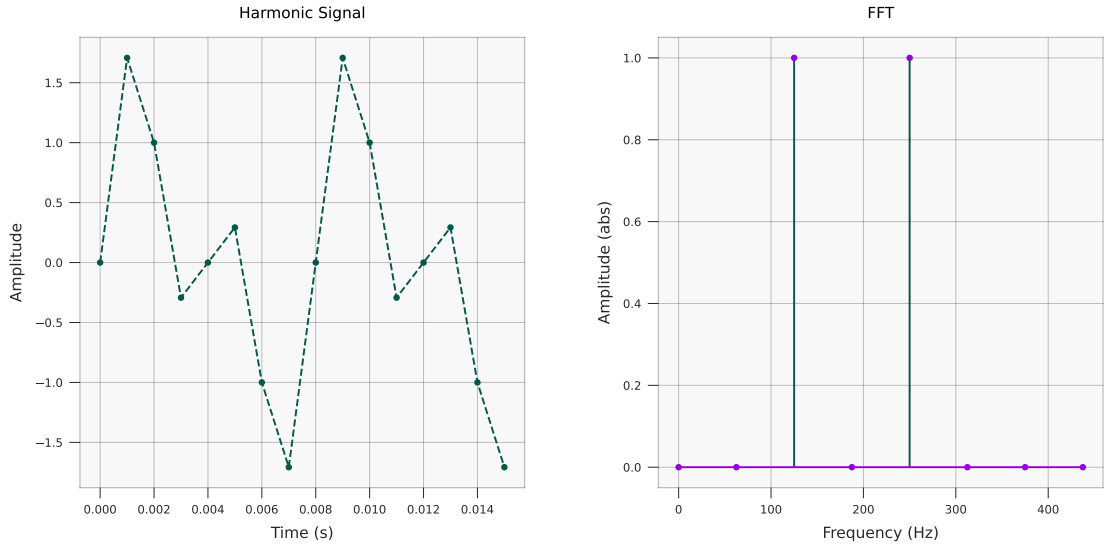
- *Transform*: After generating the overall circuit, the execution can be carried out on a previously loaded backend. This is referred to as the actual *Transformation*. After obtaining the measurement results, the noise subtraction approach can optionally be applied.

Separating the transpilation of the encoding and QFT circuit yields the advantage that in case of the STQFT where multiple QFTs executions are necessary, only the encoding circuit has to be transpiled several times whereas the QFT circuit can be reused once it is transpiled.

Angular Filter With growing width of the QFT the rotations applied by the `CP` gates become smaller. Taking a 16-QFT as an example, the `CP` gates would rotate by π, $\pi/2$, $\pi/4$ and $\pi/8$, respectively. This QFT would then have maximally 3 subsequent rotation gates which can be seen in figure 3.1. Compared with a 1024-QFT this number seems relatively small. However, it turns out that running even a 16-QFT on a real quantum device results in severe noise resulting from gate and measurement errors as discussed in section 2.2.6. This issue motivated the idea of the angular filter approach as introduced in paragraph 3.1.3.

To investigate the effect of this approach, experiments with the QFT applied on a harmonic signal are conducted. This harmonic signal, shown in figure 4.3a, is composed of two harmonic functions with 125 and 250 Hz, respectively. These frequencies are chosen because they are not susceptible to the leakage effect described in section 2.1.1 at a sampling rate of 1000 Hz used within this experiment. The number of samples in the signal was, according to the restrictions of a QFT mentioned in section 3.1, set to $2^4 = 16$ derived from $n = 4$ qubits. For reference, the same signal was transformed with a classical FFT

yielding the spectrum shown on the right in figure 4.3b



(a) **(b)**

Figure 4.3: 4.3a: Harmonic signal with 125 and 250 Hz sampled at 1000 Hz
4.3b: Spectrum of this harmonic signal with two distinct peaks at 125 and 250 Hz. Amplitudes normalized to 1.

Due to the fact that gates and measurements are subject to noise as discussed in section 2.2.6 strong fluctuations in the histogram are expected. Thus each experiment associated with a plot was performed *five times* and the corresponding deviation and mean value were plotted using *box-plots* [Kro21]. These visualization method is commonly used to graphically depict groups of numerical data. Hereby, a *box* spans across the lower and upper quartiles of the data. The median is represented by a line inside this box and *whiskers* visually extend the box to indicate the variability outside the lower and upper quartiles. Outliers are represented by individual points above and below the boxes.

"QFT_sim, mr: mr" indicates that rotation gates smaller than mr were omitted. All amplitudes plotted on the y axis are normalized to the maximum peak. The x axis indicates the frequency bins as follows:

- $1 \rightarrow 0$ Hz
- $2 \rightarrow 62.5$ Hz
- $3 \rightarrow 125$ Hz
- $4 \rightarrow 187.5$ Hz
- $5 \rightarrow 250$ Hz
- $6 \rightarrow 312.5$ Hz
- $7 \rightarrow 375$ Hz
- $8 \rightarrow 437.5$ Hz

The results achieved by utilizing the QFT with different values mr for the angular filter, are discussed in the following and presented in figure 4.4.

In this experiment, an increasing number of CP gates are omitted. This can be seen in the plots from left to right indicated by the angular filter parameter mr. The first and second row show the simulation without (*QFT_sim*) and with a noise model (*QFT_sim_n*),

respectively. It should be noted, that the simulation with a noise model, does assume and ideal layout of the device as explained in paragraph 4, although respecting its inherent noise and errors. In the third row (*QFT_real*) the real quantum device *"ibmq_quito"* was used to carry out the QFT.

In comparison with the reference spectrum in figure 4.3b, the simulations with and without the noise model achieve generally good results. However, for $\mathsf{mr} = 0.79$ and $\mathsf{mr} = 1.57$ an additional peak at frequency index $7 \rightarrow 375$ Hz occurs and the amplitude of frequency index 3 and 5 drops respectively. This effect can be explained by the reduced bandwidth, inherent with an increasing value mr of the angular filter as discussed in paragraph 3.1.3.

Nevertheless, the spectra obtained from the QFT performed on the real device hardly allow the identification of the signal spectrum. The low frequency tendency as discussed in paragraph 3.1.2 is clearly visible and yields a significant peak at frequency index $1 \rightarrow 0$ Hz. If this peak is disregarded, one can argue, that for $\mathsf{mr} = 0.39$ and especially $\mathsf{mr} = 0.79$, the desired frequencies are at least above the median of the noise.

To achieve an objective comparison between the results within each row and thus to allow for statements regarding the performance of the angular filter approach, a grader was introduced as mentioned in figure 4.1. The grader's output is calculated as follows:

$$g = 1 - \frac{1}{N} \sum_{i}^{N} |Y_i^2 - \hat{Y}_i^2| \tag{4.2}$$

where $\hat{Y}_i$ and Y_i denote the i-th frequency bin of the reference signal (FFT) and the QFT result, respectively. A comparison between these two transformations is possible, since the FFT is just like the QFT output, normalized to 1. In the ideal case, the grader result would yield 1 for two identical spectra.

The grader results for all three cases (*QFT_sim*), simulation with noise (*QFT_sim_n*) and real device experiments (*QFT_real*) are shown in figure 4.5. The value of the grader g is denoted on the y axis while the x axis ticks indicate the corresponding parameter mr of the angular filter. This correspondence, depicted in the following listing. For instance $\mathsf{mr} = 0.76$ means that all angles smaller than mr are omitted, which corresponds to 2 out of 4 possible kind of rotation gates in this specific example.

- $1 \rightarrow \mathsf{mr} = 0.00$rad
- $2 \rightarrow \mathsf{mr} = 0.40$rad
- $3 \rightarrow \mathsf{mr} = 0.76$rad
- $4 \rightarrow \mathsf{mr} = 1.57$rad

The grader results presented in figure 4.5 show that in both simulation variants (with and without noise) no improvement was achieved. Quite the opposite was the case since for increasing values of mr, the grader values even decrease. However, one could argue that since the results are already close to optimal (all grader values are above 0.92), omitting rotation gates does not decrease the QFT accuracy significantly. Contrary to the simulation variants, the grader results of the real device show a significant improvement for an increasing value mr. With regard to the scaling of the y axis, this improvement was even greater than the deterioration in the cases of simulation.

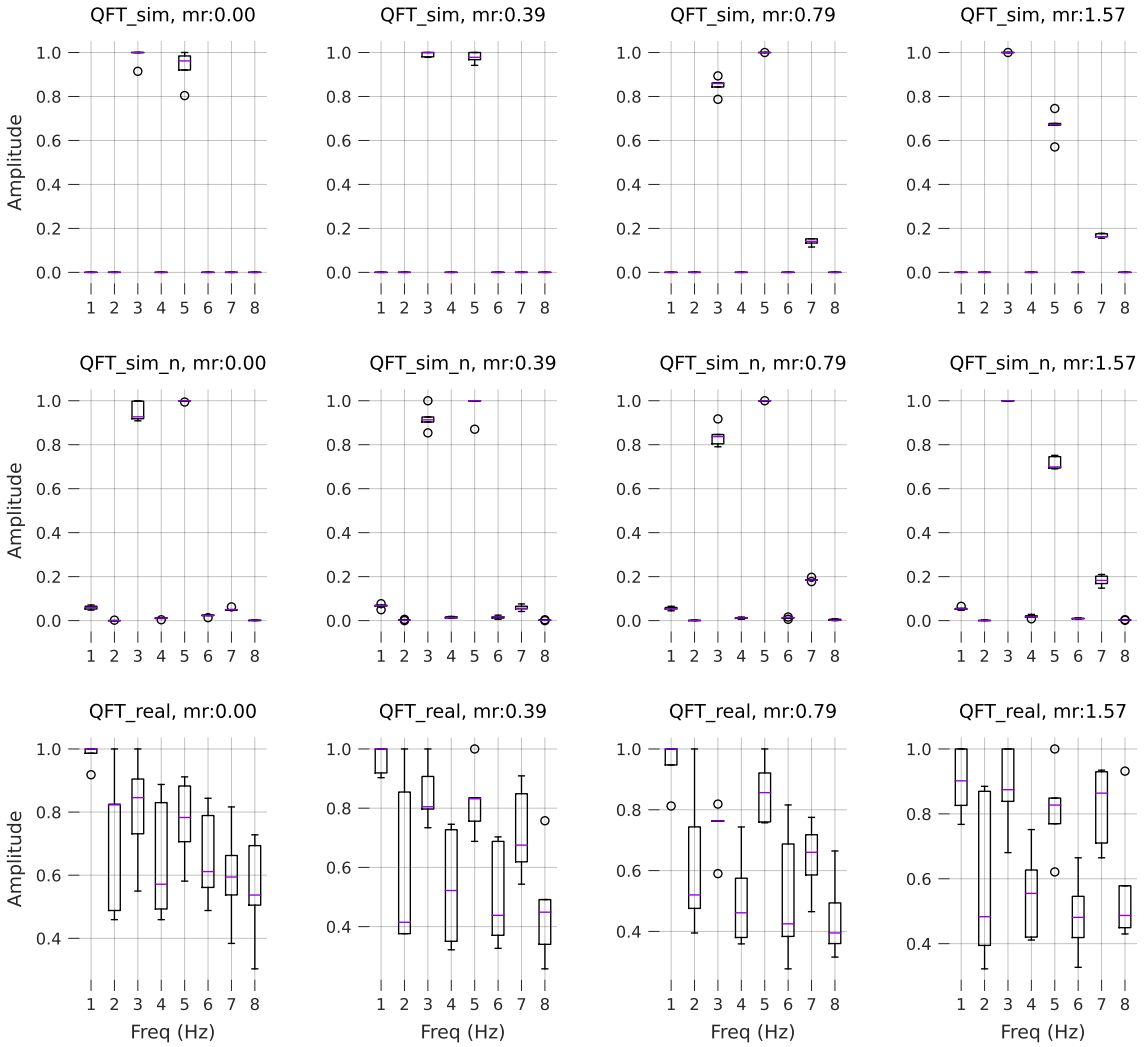
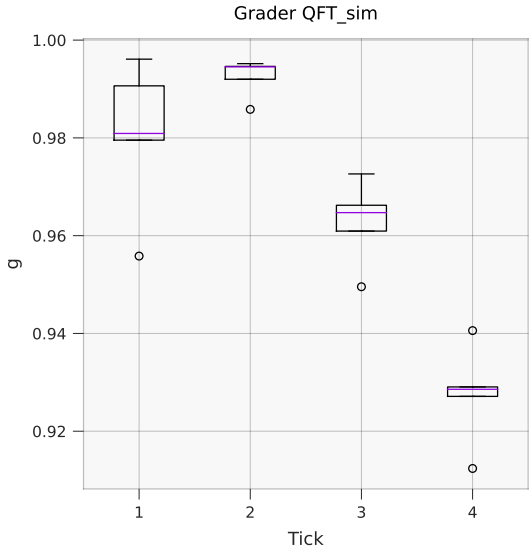
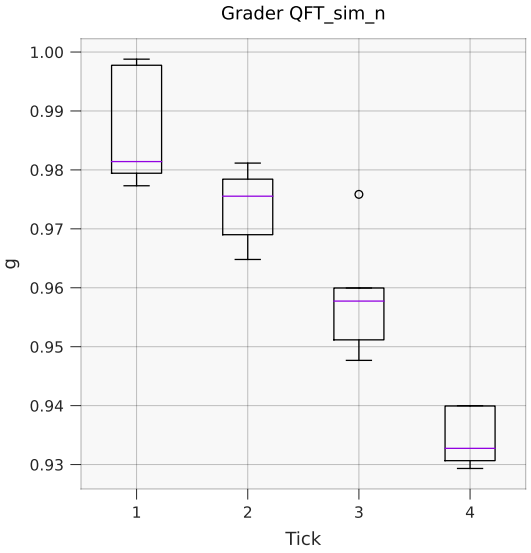


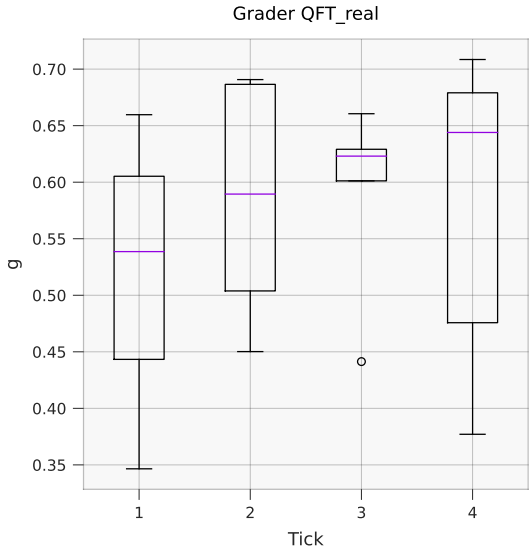
Figure 4.4: Spectra of the harmonic signal in figure 4.3a generated with the QFT and varying parameter mr of the angular filter. The experiments were conducted in simulation, both with a noise model from *"ibmq_quito"* (*QFT_sim_n*) and without noise (*QFT_sim*). The last row shows experiments conducted on the device *"ibmq_quito"* directly. The frequency indexes on the y are according in listing 4.2.3



(a)



(b)



(c)

Figure 4.5: Grader results over the number of omitted rotation gates as indicated in list 4.2.3 for the QFTs showed in figure 4.4.
The FFT on the right in figure 4.3b was used for comparison. Each plot indicates the grader results in
4.5a: simulation (*QFT_sim*),
4.5b: simulation with noise (*QFT_sim_n*) and
4.5c: real device experiments (*QFT_real*).

The 4-qubit quantum device *"ibmq_quito"* with the device layout visualized in figure 4.6 has been used in this experiment. Knowledge about the device structure and error rates of single qubits is useful for assessing which device is suitable for a specific application since not all noisy intermediate scale quantum (NISQ) devices are equally suitable as discussed in section 2.2.5. Useful properties of a device are in general the *readout assignment-* and the $\texttt{C}_\texttt{X}$*-error* of individual qubits. In case of *"ibmq_quito"*, the average readout assignment error is 2.998e−2 and the average $\texttt{C}_\texttt{X}$-error is 1.027e−2 based on the calibrations from [Qua21] on 30.10.2021. This specific device was chosen in favour of the other publicly available 5-qubit devices for further experiments, since it provides the lowest average readout assignment- and $\texttt{C}_\texttt{X}$-error rate. Each circle in figure 4.6 represents a single qubit with its individual readout assignment error corresponding to the line thickness. The $\texttt{C}_\texttt{X}$-error is displayed by the line thickness of the connections between each qubit. In case of interest, the absolute values of the errors can be found in the appendix in table A.1.

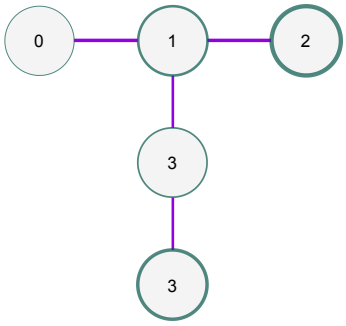


Figure 4.6: Map view of *"ibmq_quito"* with "IBM Quantum Falcon r4T processor". The *Readout assignment error* is indicated by the qubit's line thickness (the thicker the smaller the error) and the $\texttt{C}_\texttt{X}$*-error* is indicated by the connections (the thicker the smaller the error)

Noise Mitigation Taking a closer look at the results in figure 4.4, one can see that there is a significant peek at 0 Hz in measurements from the real device which results from the qubits striving to the ground state as explained in section 3.1.2.

This observation led to the *noise subtraction* approach presented in section 3.1.3 where it was suggested to subtract the counts of a constant signal's QFT from the noisy measurement results. The results of applying the proposed noise mitigation are shown in figure 4.7. The peak at 0 Hz was successfully removed and the noise floor especially for $\mathsf{mr} = 0.20\text{rad}$ was reduced significantly. However, for large mr results become worse, although they stay competitive against the non-mitigated variant.

Figure 4.8 shows the corresponding grader results from this experiment. The grader values improved significantly by up to 1.51 for a single value of mr compared to the case where no noise subtraction was applied, shown in figure 4.5. All ratios of the grader values between these two experiments are listed in the following:

- $\Delta g_1 = \frac{0.675}{0.54} = 1.25$
- $\Delta g_2 = \frac{0.89}{0.59} = 1.51$
- $\Delta g_3 = \frac{0.86}{0.625} = 1.38$
- $\Delta g_4 = \frac{0.74}{0.64} = 1.16$

However, the grader values also indicate a decreasing performance for $\mathsf{mr} \geq 0.4$ rad of the angular filter, although even in the worst case, the improvement still obtains a ratio of 1.16. This indicates two contradicting effects: On the one hand, by reducing the rotation gate count, the noise, inherent with these gates, also decreases, thus improving the QFT

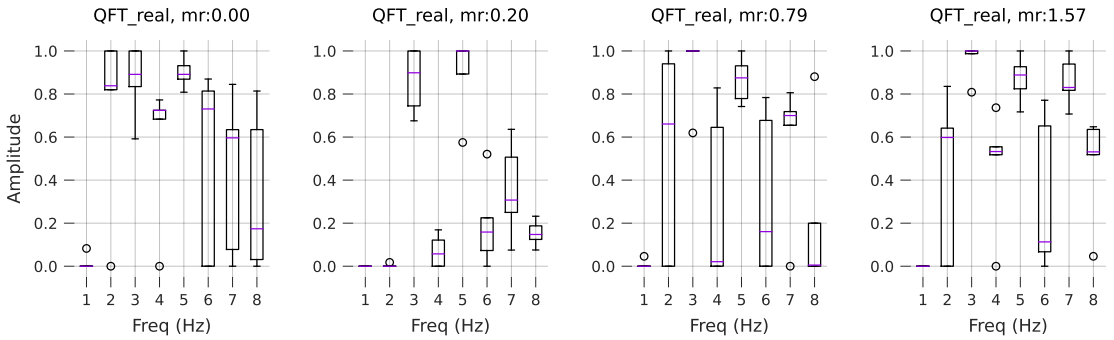


Figure 4.7: QFT on real device *"ibmq_quito"* with noise mitigation technique explained in section 3.1. Angular filter applied, indicated by the minimum rotation angle (mr). Compared to the corresponding spectrograms on the real device showed in figure 4.4, the noise subtraction method significantly improved the results, which is especially visible for $\mathsf{mr} = 0.2$

accuracy. On the other hand, omitting too many rotation gates, also severely limits the QFT bandwidth and thus deteriorates the grader values.

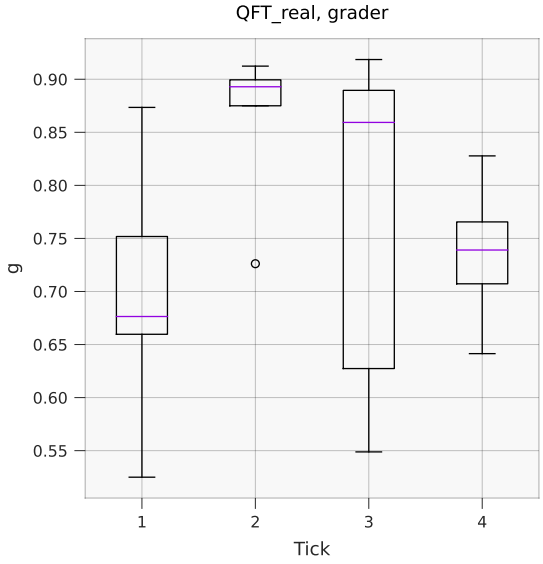


Figure 4.8: Grader results from noise mitigated QFT on the real device *"ibmq_quito"* plotted over angular filter indices declared in listing 4.2.3. Compared to the first tick, where no angular filter is applied, a significant improvement in tick 2 ($\mathsf{mr} = 0.40$) is visible which decays for the remaining ticks ($\mathsf{mr} = [0.76, 1.57]$)

Noise mitigation based on the evaluation of all possible input combinations, as explained in section 2.2.6, was also tested using noise mitigation tools provided by the *qiskit framework*. However, it turned out to be not as effective as above results while being computationally more complex for a large number of QFTs as it is required for the STQFT.

4.3. Short Time Quantum Fourier Transform

As shown already in figure 4.1, a STT framework is used as an abstraction layer covering both the STFT and the STQFT.

To obtain a sufficiently good time and frequency resolution, at least a 7-qubit QFTs has to be used for the following experiments. However, the publicly available devices at IBM Quantum currently cover at most 5-qubit devices thus following experiments are restricted to the noise model of more complex quantum computers and experiments on real devices are not included.

Initial experiments are performed with a chirp, since this is a relatively simple signal and thus the results can be compared well with those to be expected. In the following experiments, two chirps, starting at 500 and 1000 Hz and ending at 2000 and 3000 Hz,respectively, are presented. These signals were chosen because they cover most of the Nyquist band ($[0 \dots 4000]$ Hz at 8 kHz sampling frequency) while having different slopes so that eventual artefacts can easily be associated. To perform experiments as realistic as possible, a relatively high sampling frequency is chosen as it would be conceivable in SR.

During experiments the *Hanning window* 2.1.1 was used since it preserves the signal energy at a window overlap of 0.5. Different windows were tried as well, but did not yield significantly different results, which is why the mentioned window overlap is acceptable as a default.

In contrast to the experiments from previous section, there is no repeated execution of the STQFT: a single STQFT contains a sufficient number of QFTs to reach a low variance thus providing a solid foundation.

4.3.1. Noise Mitigation

The first graph in figure 4.9 shows the STFT as a reference, followed by the STQFT, simulated without any additional noise information *STQFT_sim*. *STQFT_sim_n* adds the noise model from the 7-qubit device *"ibmq_casablanca"* while the rightmost graph *STQFT_sim_n, mitigated* adds the mitigation technique evaluated in the previous experiment with the QFT in paragraph 4.2.3.

The thickness of the lines in these figures indicating the chirp signals spectrogram is the result of the restricted time-frequency resolution inherent with STFT as discussed in paragraph 2.1.1. This restriction leads to the leakage effect described in paragraph 2.1.1 and is visible as two additional stripes on upper and lower side of each chirp in the STFT spectrogram. In case of the STQFT, these three stripes in the resulting chirps are more blurred. This can be explained by the amplitude inaccuracy inherent with the QFT even in the simulated case as discussed in paragraph 3.1.2.

The results of the *STQFT_sim_n* show additional artefacts (crossing chirps with the same inverse slope but lower amplitude) in the plot. Since these additional chirps are not well visible in this experiment, they will be discussed in one of the following experiments. However, it can be noticed that in case of the *STQFT_sim_n, mitigated*, these artefacts are not visible anymore.

For later reference, figure 4.10 shows the architecture of *"ibmq_casablanca"*, from which it is conceivable that the all 7 qubits are required in context of this experiment with a 128-QFT. The average readout assignment error on *"ibmq_casablanca"* is 2.300e−2 and the average $\mathtt{C_X}$-error is 1.035e−2 based on the calibrations from [Qua21] on 30.10.2021. In case of interest, the absolute values of the errors for individual qubits and connections can

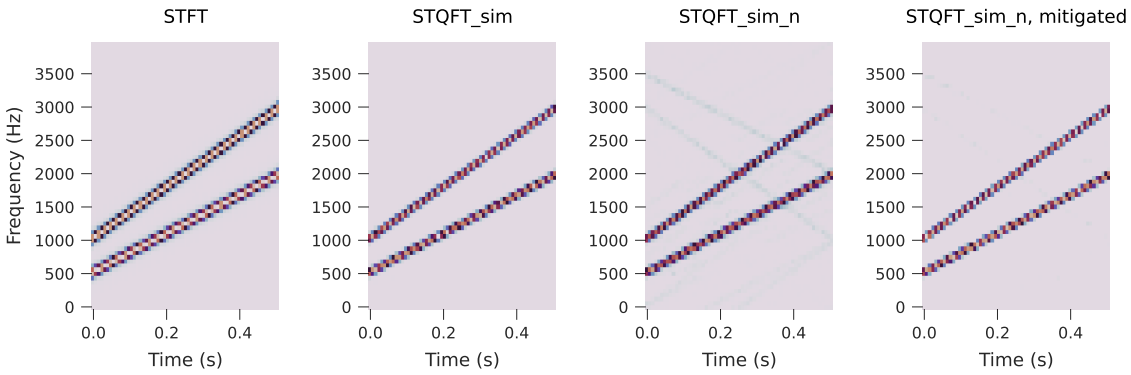


Figure 4.9: STQFTs of a synthetic chirp signal in simulation (*STQFT_sim*), simulation with noise model (*STQFT_sim_n*) from *"ibmq_casablanca"* and applied noise mitigation on the latter (*STQFT_sim_n, mitigated*) besides the STFT as reference. Despite a slightly different amplitude, the results of in *STQFT_sim* appear very similar to the reference. The noise model in *STQFT_sim_n* resulted in artefacts of the chirp signal. However, these are successfully removed by applying the noise subtraction method in *STQFT_sim_n, mitigated*.

be found in the appendix in table A.2.

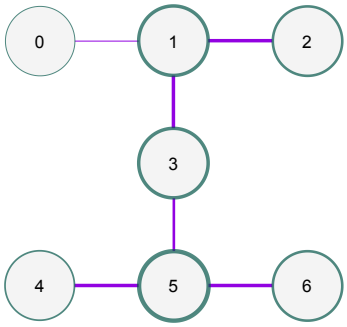


Figure 4.10: Map view of *"ibmq_casablanca"* with "IBM Quantum Falcon r4H processor". *Readout assignment error* indicated by the qubit's line thickness (the thicker the smaller the error) and C_X-*error* indicated by the connections in the same manner.

To draw a conclusion, the results from figure 4.9 give a first indication that STQFT can be successfully applied to synthetic signals on real NISQ devices. In the remainder of this subsection, the STQFT will be tested on another quantum device to show the relevance of the device layout. Finally, the aforementioned artefacts in the *STQFT_sim_n* are further investigated.

STQFT on *"ibmq_guadalupe"* Additional experiments were performed using the noise model from the 16-qubit device *"ibmq_guadalupe"*. The architecture of this device is not shown here due to its high complexity, but can be in the appendix in figure A.1. The device average readout assignment error is 3.165e−2 and the average C_X-error is 1.825e−2 measured on 30.10.2021. In case of interest, the readout assignment- and C_X-errors of individual qubits and connections can be found in the appendix in table A.3. The reason for this additional investigations is the motivation to address the artefacts in the previous experiment without mitigation but with a more complex quantum device having a higher number of qubits available from which the transpiler can choose. As explained in section 2.2.5, the transpiler tries to find an optimal realization of the circuit given the device specific layout [Pre18].

Redundant qubits allow the transpiler to choose the qubits with the lowest readout and $\texttt{C}_\texttt{X}$ error.

The experiments results with this 16-qubit device are shown in figure 4.11. Although still present, the aforementioned artefacts are greatly reduced compared to the spectrogram *STQFT_sim_n* in figure 4.9. The remaining artefacts and their source are subject of the following discussion.

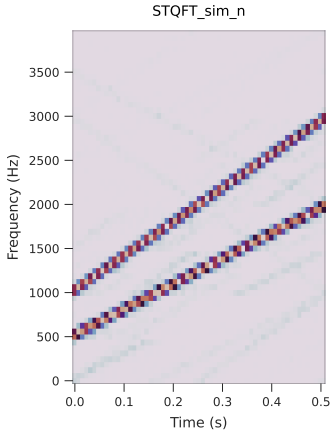


Figure 4.11: STQFT of a chirp signal in simulation with noise model from *"ibmq_guadalupe"*. No noise mitigation applied. Compared to the spectrogram *STQFT_sim_n* in figure 4.9, the artefacts of the chirp signal decreased

STQFT artefacts While the noisiness of a quantum device obviously correlates with the quality of obtained spectrograms, additional experiments were conducted examining the exact source of the geometrically similar artefacts mentioned at the beginning of this section in context of *STQFT_sim_n* in figure 4.9. It turned out that coherent errors as discussed in section 2.2.6, more precisely CP gate rotation errors, are mainly influencing the intensity of such artefacts. This is shown in figure 4.12 where the angular offset is indicated by er stated above the figures and iteratively increased from $\mathsf{er} = [0.12 \dots 0.21]$.

Although for $\mathsf{er} = 0.12$ barely any artefacts are visible, this suddenly changes with $\mathsf{er} = 0.15$ and even more artefacts become visible for increasing er. This finding is confirmed by experiments with the classical STFT, where an identical behavior can be observed for offsets $\epsilon \geq 0$ with $\epsilon \in \mathbb{R}$ in the exponent of $\exp(-\imath 2\pi(\frac{k}{W}n + \epsilon))$ from the equation of the STFT in equation (2.14).

4.3.2. Angular Filter

In paragraph 4.2.3 the angular filter approach based on the results of the QFT of an harmonic signal carried out on *"ibmq_quito"* was evaluated. Figure 4.13 shows this experiment setup with the STQFT applied on the chirp signal and executed on the *"ibmq_casablanca"* device. This device was chosen in favour of *"ibmq_guadalupe"*, evaluated section 4.3.1, for this experiment, since noise reduction was expected to have a more distinct effect on noisier devices.

It turns out that for $\mathsf{mr} \leq 0.20$ (the upper four plots) there are only slight artefacts. This suddenly changes for $\mathsf{mr} = 0.39$ and get worse up to $\mathsf{mr} = 1.57$ where only one rotation gate per qubit is left and thus no meaningful transformation possible. In general it can be said that at a certain point, increasing the mr parameter further will reduce the accuracy of

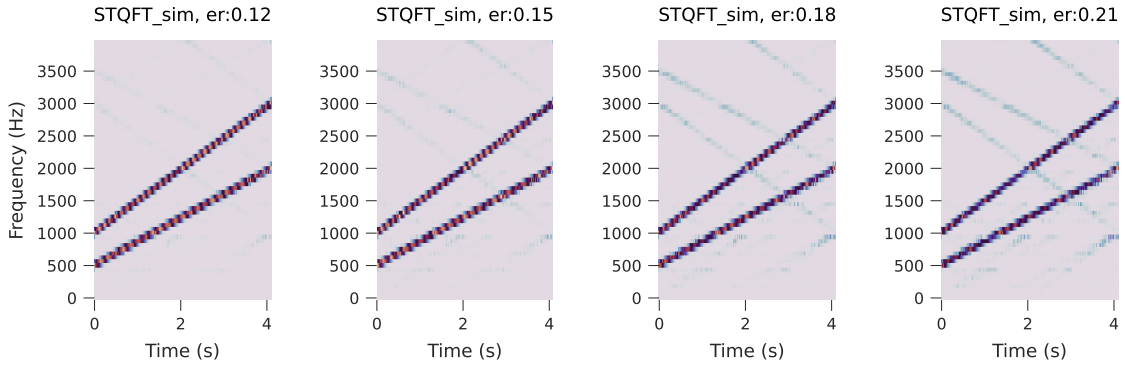


Figure 4.12: STQFTs of a chirp signal in simulation without noise model but with varying rotation gate error er. The artefacts of the chirp signal increase significantly for $\mathsf{er} \geq 0.15$.

the QFT drastically, such that the benefits of reducing noisy rotation gates are negligible compared to the resulting overall noise in the spectrogram. This leads to the suspicion, that an optimal value of this parameter exists and must be evaluated in further experiments.

According to the findings from section 4.3.1, to evaluate the noise subtraction approach in conjunction with the angular filter. Therefore, an experiment was conducted where the results were mitigated and different values for the angular filter evaluated. It turned out that even for the case of $\mathsf{mr} = 0.39$ where severe artefacts were visible in the spectrum in figure 4.13, these significantly reduced after the application of the noise subtraction approach. This allows for the conclusion, that the mentioned mitigation approach is not only capable of reducing device specific noise produced by gate and measurement errors, but also efficiently mitigates artefacts caused by the angular filter approach. Another advantage of omitting CP gates as done with the angular filter, is the decreasing computational effort, not only for simulation on classical machines, but also for quantum computers. Therefore, a combination of both approaches, noise subtraction and angular filter methods seems reasonable for application in SR tasks as they are able to reduce the overall gate count while simultaneously mitigating noise and accompanying artefacts.

4.3.3. Speech Signals

After these experiments the gained knowledge is tested on experiments on the speech signals from the dataset introduced in section 4.1. As in the previous section, experiments have not been performed on real quantum hardware, because of the lack of publicly available and sufficiently large quantum computers. The noise model of the quantum device *"ibmq_guadalupe"* is used instead. This sections aims to find an upper bound for parameters of the signal threshold filter (st) and angular filter (mr). Furthermore the impact of the noise subtraction approach introduced in section 3.1.3 is investigated.

At first an experiment was conducted for reference without any of the aforementioned approaches. This result of the STQFT applied on the spoken sample for the word *"left"* is shown in figure 4.15b. The STFT generated spectrogram is shown in figure 4.15a for

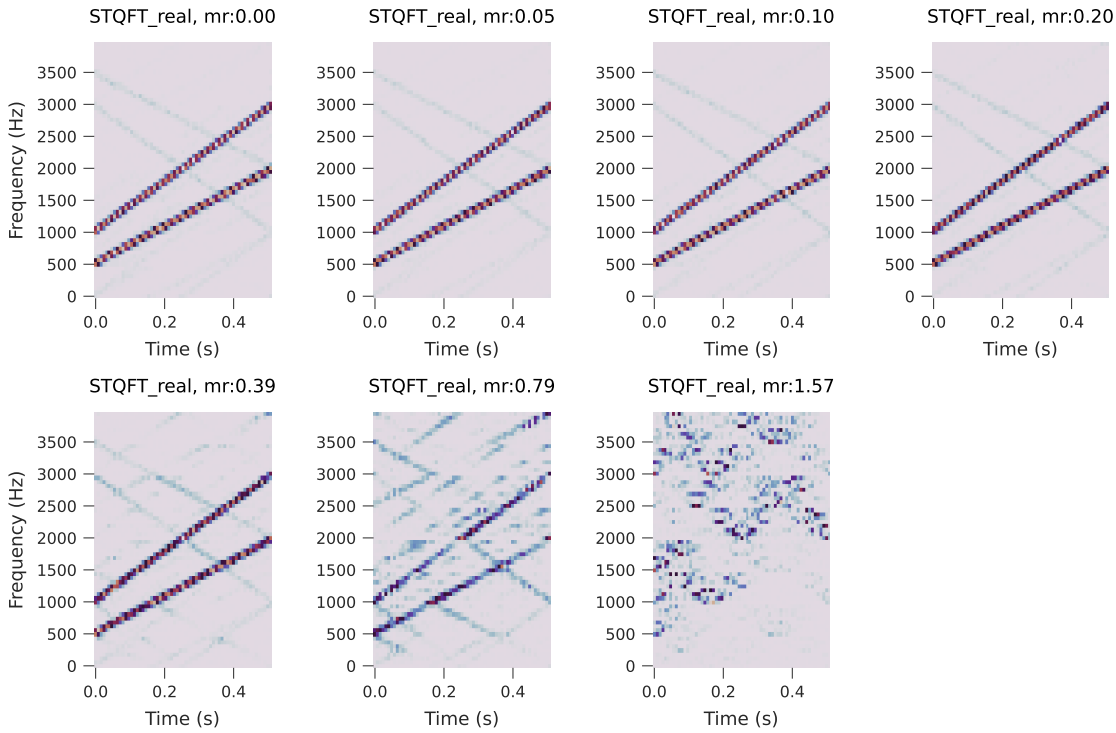


Figure 4.13: STQFTs of a chirp signal in simulation with noise model from *"ibmq_casablanca"*. Noise subtraction *is not* applied. Each spectrogram corresponds to a STQFT with iteratively increased value of the angular filter $\mathsf{mr} = [0.006 \dots 1.571]$ where each increment corresponds to an rotation angle of a set of `CP` gates. For $\mathsf{mr} = [0.00 \dots 0.20]$, there are no visible changes in the spectrogram, which suddenly changes and increases for $\mathsf{mr} = [0.39 \dots 1.57]$. Artefacts of the chirp signal are present even with a disabled angular filter ($\mathsf{mr} = 0.00$) and heavily increase for $\mathsf{mr} \geq 0.39$.

comparison. A 1024 samples *Blackman window* (see common options in section 2.1.1) with an overlap of 0.875 was used in conjunction with a 1024-QFT and 16 kHz sampling rate. These values were chosen according to the STFT of the *librosa framework* [MRL+15] properties used in [YQC+20b]. Other types of windows were evaluated, but did not yield significantly different results, thus are not presented.

Signal Threshold Filter Compared to the reference STFT in figure 4.15a, the output visually appears weaker and artefacts are visible in areas, where no signal is indicated by the STFT. This can be explained by the fact that the QFT output is normalized, thus segments in the STQFT, which are almost zero in the STFT, are forced to sum up to one as explained. To address this issue a *signal threshold filter* was introduced in section 3.2 such that segments in which the input signal is smaller than a predefined threshold are not computed with the QFT, but instead directly set to zero.

Figure 4.16 shows the spectrograms with increasing parameter st of the *signal threshold filter*. As can be seen in figure 4.16a a threshold of $\mathsf{st} = 0.02$ can effectively suppress

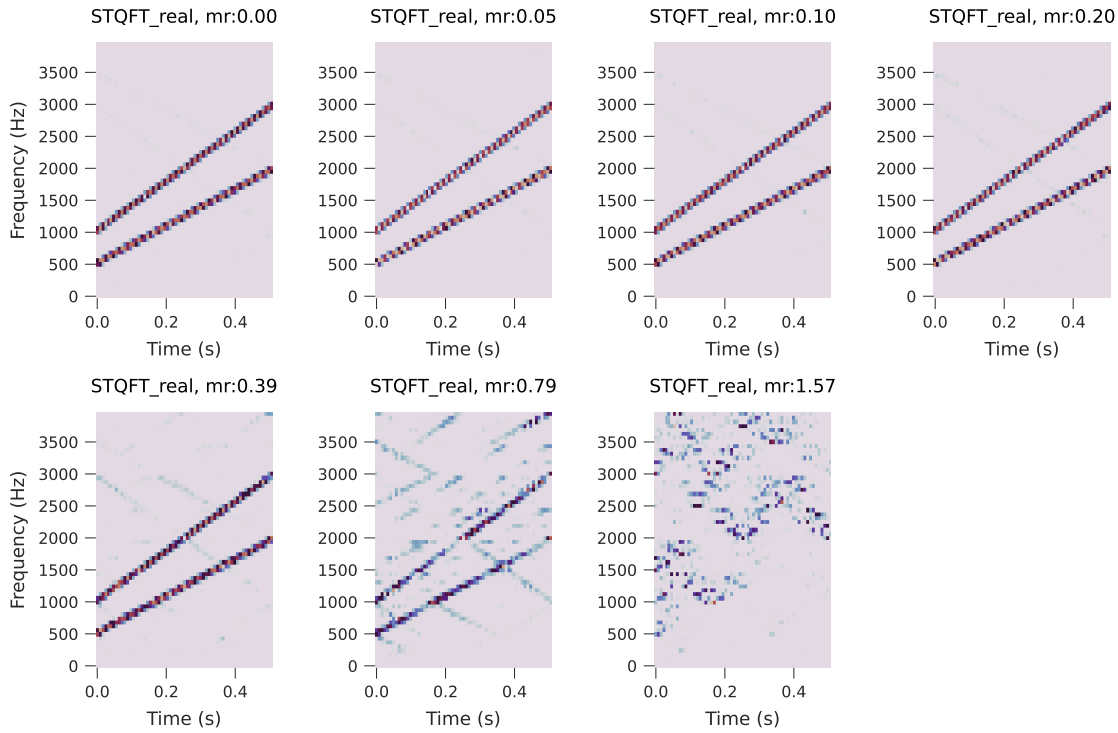
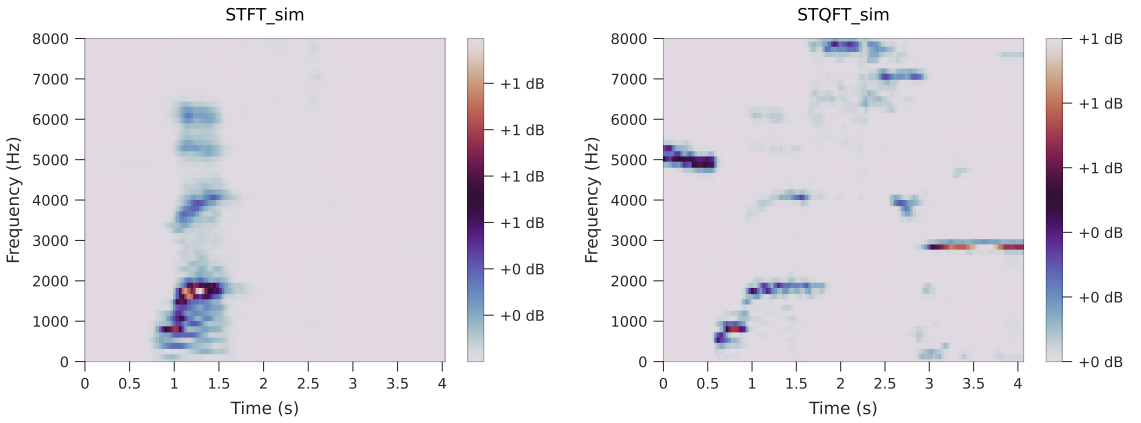


Figure 4.14: STQFTs of a chirp signal in simulation with noise model from *"ibmq_casablanca"*. Noise subtraction *is* applied. Each spectrogram corresponds to a STQFT with iteratively increased value of the angular filter $\mathsf{mr} = [0.006 \dots 1.571]$ where each increment corresponds to an rotation angle of a set of `CP` gates. For $\mathsf{mr} = [0.00 \dots 0.20]$, there are no visible changes in the spectrogram, which suddenly changes and increases for $\mathsf{mr} = [0.39 \dots 1.57]$. Artefacts of the chirp signal are significantly reduced by the noise subtraction method compared to the spectrograms in figure 4.13, but increase as well for $\mathsf{mr} \geq 0.39$.

artifacts at the beginning and end of the signal. Increasing this threshold further, however, yields to a lack of spectrogram parts which are visible in case of the STFT. This in turn removes potentially relevant features for classification with the NN discussed later. Comparison with this reference suggests to choose $\mathsf{st} = 0.06$ as in the spectrogram shown in figure 4.16b for further investigations. Besides the suppression of normalization artefacts, this approach also significantly reduces computational effort and decreases the number of processed QFTs from 127 for $\mathsf{st} = 0$ down to 32 for $\mathsf{st} = 0.06$ in this particular speech sample.

Comparing the spectrograms in figure 4.16 with the reference in figure 4.15a, it can be noticed, that e.g. the spectral components between $\tilde{[}500 \dots 2000]$ Hz for the time period $\tilde{[}0.6 \dots 1.5]$ s appear *wider* in the STQFT as in the reference. This effect can be explained by the normalization inherent to the QFT discarding information about the absolute amplitude as discussed in paragraph 3.1.2. Without this information, raising or lowering amplitudes in the voice are not displayed correctly as neighboring QFTs within a single STQFT will normalize each spectral window. In a classification task with a NN, this

information could be considered as a feature and missing it would thus reduce the accuracy as discussed in paragraph 3.2.



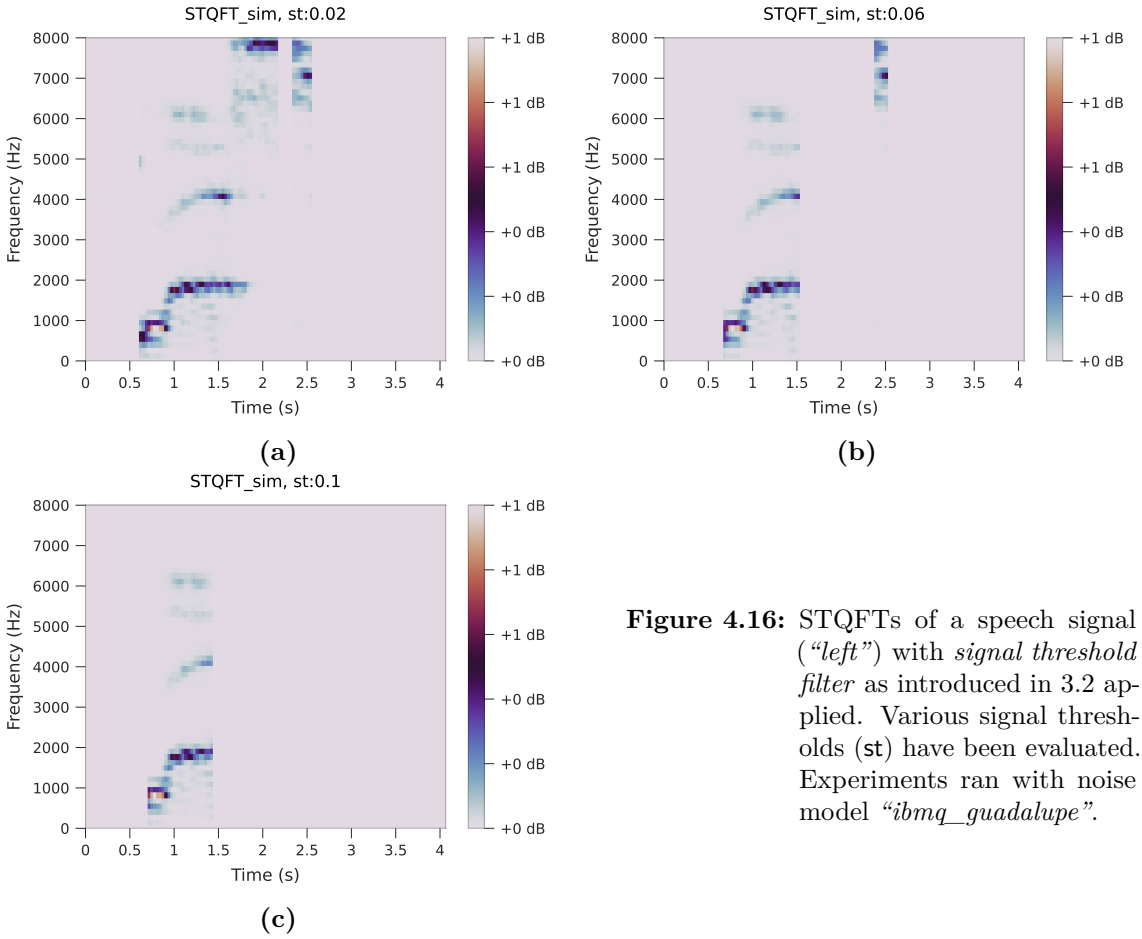
(a) (b)

Figure 4.15: 4.15a: STFTs of a speech signal (*"left"*). 16 kHz sampling rate, *Blackman window* 2.1.1 with overlap of 0.875
4.15b: STQFTs of a speech signal (*"left"*) in simulation with noise model from *"ibmq_guadalupe"*. Compared to spectrogram 4.15a, severe artefacts are visible and relevant spectral areas lack of detail.

Noise Mitigation Section 4.3.1 shows promising opportunities for the noise subtraction approach introduced in section 3.1.3 also for applying the STQFT. To validate this on speech signals experiments were conducted with the noise model of the *"ibmq_melbourne"* quantum device and applied noise subtraction. The 16-qubit device *"ibmq_melbourne"* with "Falcon r4P Processor" was chosen since it is noisier compared to *"ibmq_guadalupe"* thus mitigation results are expected to be more visible in this case. The device average readout assignment error is 5.814e−2 and the average $\mathtt{C_X}$-error is 3.169e−2 measured on 30.10.2021.

The results of this experiment are shown in figure 4.17 where a signal threshold of $\mathsf{st} = 0.06$ is applied in both cases. In the spectrogram shown in figure 4.17b many artefacts that can be seen in the unmitigated spectrogram displayed in figure 4.17a were successfully removed. A comparison with the case of $\mathsf{st} = 0.06$ in figure 4.16 confirms that the application of the noise subtraction method on *"ibmq_melbourne"* led to a spectrogram more similar to the less noisier device *"ibmq_guadalupe"*. However, one might argue that the additional spectrogram components in figure 4.17b at around $[5 \dots 6.5]$ kHz which are not visible in the spectrogram from *"ibmq_guadalupe"*, might also be beneficial for the NN in later experiments. Therefore the performance with the noise models of both devices, *"ibmq_guadalupe"* and *"ibmq_melbourne"*, will be investigated.

Angular Filter As final part of this section, all suitable values for the angular filter $\mathsf{mr} = [0.006 \dots 1.571]$ are evaluated in figure 4.18 where $\mathsf{mr} = 0$ was omitted for the sake of clarity. These experiments were conducted, in addition to the evaluation in section 4.3.2 since a different behavior for synthetic and real speech signals is expected. Since the

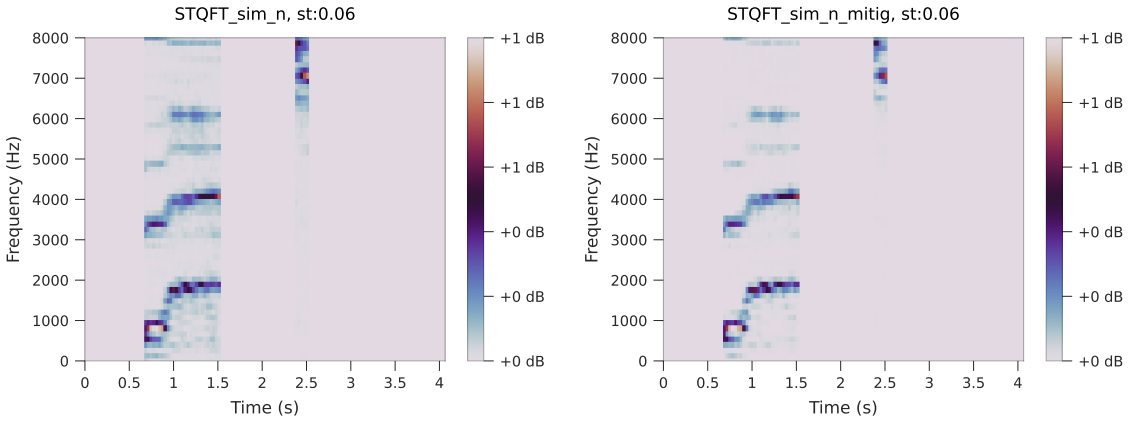


(a) (b) (c)

Figure 4.16: STQFTs of a speech signal (*"left"*) with *signal threshold filter* as introduced in 3.2 applied. Various signal thresholds (st) have been evaluated. Experiments ran with noise model *"ibmq_guadalupe"*.

sampling rate of the speech signal requires a 1024-QFT, the granularity of the angular filter increased, which is why its parameter mr is provided with a higher precision in this experiment. The noise model of *"ibmq_guadalupe"* has been used in this experiment since noise should not be subject to investigations in this case. Based on the evaluations in paragraph 4.3.3, a signal threshold of $\mathsf{st} = 0.06$ was chosen. The noise subtraction method was not applied according to the findings in section 4.3.2, since this would disallow for any conclusion on the effects of the angular filter.

For values of the angular filter of $\mathsf{mr} < 0.2$ there are no visible changes in the spectrograms. However, this changes at values $\mathsf{mr} \geq 0.2$ where artefacts, especially in high frequent parts of the spectrogram become increasingly visible. Due to the applied signal threshold, blank parts of the spectrogram where the maximum amplitude of the signal for a single QFT window is < 0.06, remain unchanged. It should be noted, that the angular filter, removes all rotation gates *smaller than* the given parameter. Therefore, setting this parameter to $\mathsf{mr} = 0.2$ is equivalent to the resulting spectrogram with $\mathsf{mr} = 0.196$ in figure 4.18 and therefore a valid option. The results of this experiment encourage an optimal value of $\mathsf{mr} = 0.2$ (6 out of 10 possible CP gates in a 1024-STQFT) since this provides the QFT with the lowest number of CP gates, while resulting spectrograms are well comparable with those including a full QFT as in figure 4.17a.



(a) (b)

Figure 4.17: 4.17a: STQFTs of a speech signal (*"left"*) with st = 0.06 and noise model of *"ibmq_melbourne"* applied.
4.17b: Mitigated variant with applied noise subtraction approach from section 3.1.3.

4.4. Connection to the Neural Network

This section describes the evaluation of the STQFT as feature generating component in a SR pipeline based on the performance of a NN. The training-, validation-set presented in paragraph 4.1 are used in following experiments. Additionally, the test set is used for a final performance measure of the model at the end of this section. This final model is chosen based on the validation accuracy in subsequent experiments. From the experiments in previous sections, the following properties of the STQFT are found to be well suited for the given setup and also indicate and upper bound:

- mr = 0.2
- st = 0.06
- Include noise mitigation

From the discussion in paragraph 4.3.3 both noise models *"ibmq_guadalupe"* and *"ibmq_melbourne"* will be evaluated to confirm if these will have an impact on the NN learning performance.

Typical learning hyperparameters of the NN model, the *Batch Size* and the number of *Epochs* described in paragraph 2.1.2, have been carried over from [YQC+20b] as follows:

- Batch Size: 16
- Epochs: 30

Figure 4.19 shows the loss (solid line) and accuracy (dashed line) history for the training and validation set, respectively. Three training procedures were performed, yielding a validation accuracy of 0.820 ± 0.003. The resulting history plot shows strong fluctuations in the accuracy, indicating repetitive finding of local minima.

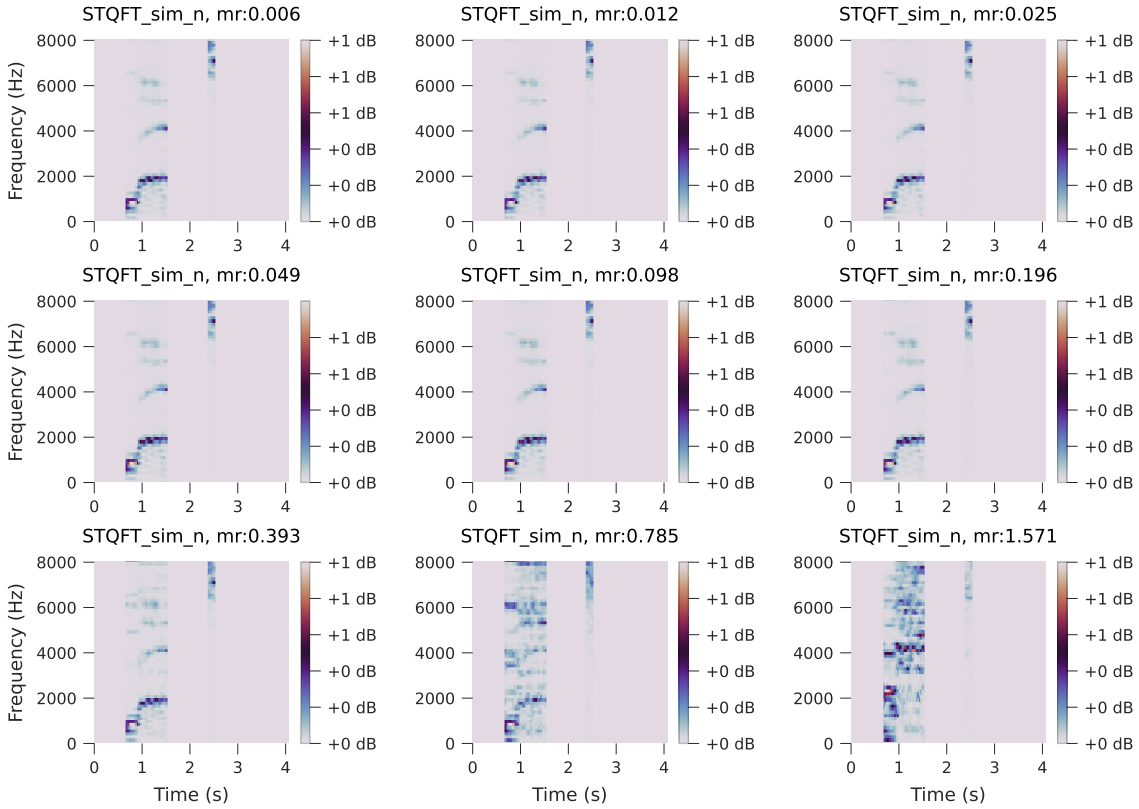


Figure 4.18: STQFTs of a speech signal (*"left"*) with st = 0.06 and noise model of *"ibmq_guadalupe"* applied. Each spectrogram corresponds to a STQFT with iteratively increased value of the angular filter mr = [0.006 . . . 1.571] where each increment corresponds to an rotation angle of a set of CP gates. For mr = [0.006 . . . 0.196], there are no visible artefacts, which suddenly changes and increases for mr = [0.393 . . . 1.571].

Adaptation of NN Hyperparameters As discussed in paragraph 2.1.2, these fluctuation can indicate large gradients. Using the ADAM optimizer instead of SGD is expected to reduce the fluctuations since learning steps are adapted based on the gradients history. Increasing the batch size is also considered appropriate to prevent the above effects, as also shown in paragraph 2.1.2. The increasing validation loss in contrast to the decreasing training loss indicates overfitting which can be addressed by the introduction of dropout as discussed in paragraph 2.1.2. In summary, the following modifications are applied:

- Increase batch size up to 30 (expected to reduce fluctuations)
- Apply dropout as regularization method after each long short-term memory (LSTM) and after the *Dense64* layer
- Use ADAM Optimizer instead of SGD (expected steadier convergence)

These modifications lead to the history plot in figure 4.20. A clear improvement regarding the fluctuations of the validation loss is visible. Although the validation accuracy improves to 0.832 ± 0.003, overfitting is still present despite the applied dropout. This can indicate

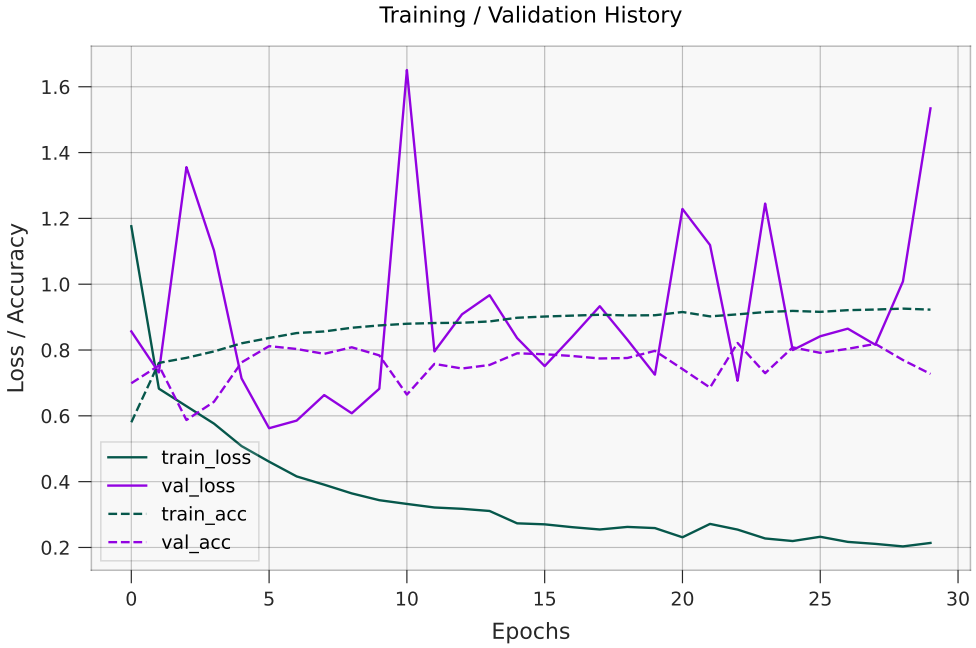


Figure 4.19: Loss & Accuracy plot over epochs for the initial experiment including STQFT in the SR pipeline from [YQC+20b]. Parameters for the STQFT with noise mitigation set to $\mathsf{mr} = 0.2$ and $\mathsf{st} = 0.06$. Strong fluctuations and overfitting is visible. Validation accuracies: 0.820 ± 0.003

a lack of features compared to the original training data in [YQC+20b] on which the NN originally achieved a validation accuracy of 0.9512. In paragraph 4.3.3 a potential reason was given by the fact, that in contrast to the STFT, the STQFT cannot display raising or lowering voices in a speech signal due to the normalization required in every single QFT. In addition, the resulting spectrogram is usually not as detailed as for STFT, further reducing the relevant information for NN in the spectrogram.

Pixel-Channel-Mapping It is suspected that the quantum convolutional (quanv) layer makes the spectrograms of the STQFT unnecessarily noisy. Therefore, a *Pixel-Channel-Mapping* as suggested in paragraph 3.3 was applied which leads to the training results depicted in figure 4.21. By applying this approach, the accuracy increases to 0.845 ± 0.003.

Although still present, overfitting is further reduced. This can be explained by the reduced decoherence inherent with the second quantum circuit (quanv layer from [YQC+20b]) in the processing pipeline. In a further experiment a convolutional layer as suggested in [YQC+20b] and discussed in paragraph 3.3 is used as a replacement for the Pixel-Channel-Mapping. However, this leads to a drop in validation accuracy to 0.824 ± 0.003. Therefore, this approach is not regarded any further.

Noise model of *"ibmq_melbourne"* The findings in section 4.3.3 left the choice of the noise model open to *"ibmq_guadalupe"* and *"ibmq_melbourne"*. While the latter was

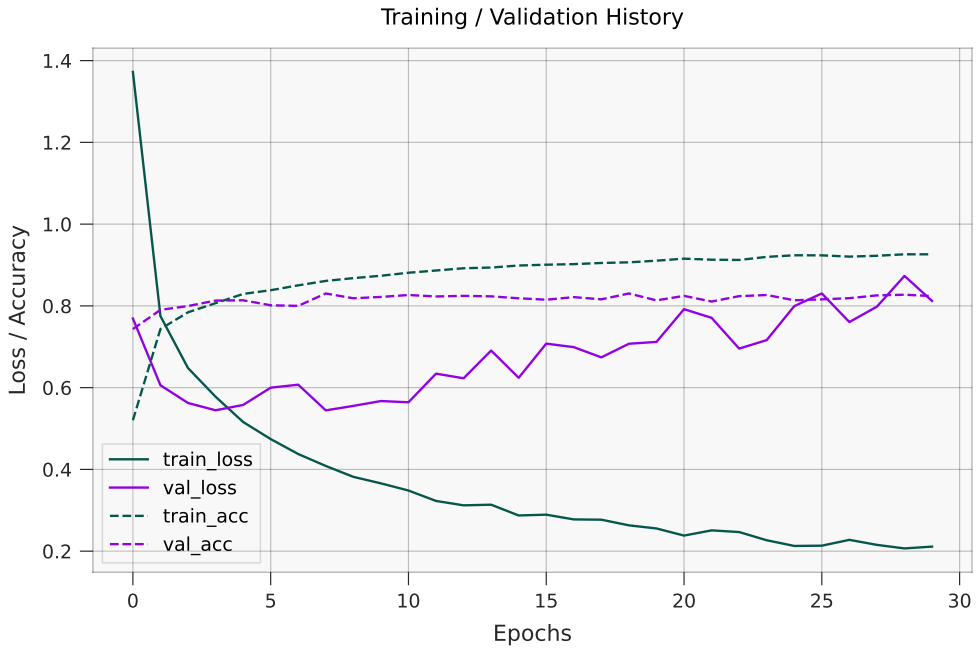


Figure 4.20: Loss & Accuracy plot over epochs after adapting NNhyperparameters. Fluctuations decreased, but severe overfitting still visible. Validation accuracy: 0.832 ± 0.003

found to be noisier, it might also yield additional features needed to address the problem of overfitting. The history plot of the experiment with the noise model *"ibmq_melbourne"* remains similar to the previous ones and is therefore omitted here. In case of interest, the plot can be found in the appendix in figure A.2. With a resulting validation accuracy of 0.825 ± 0.001, the additional noise inherent with the device model do not lead to a better generalization of the NN although the fluctuations decreased. Due to the reduction by 2 % compared to the validation accuracy of 0.845 ± 0.003 achieved with the noise model *"ibmq_guadalupe"*, the latter will be used in further experiments.

Adaptation of STQFT parameters By now, the parameters from listing 4.4 were used for the angular filter ($\mathsf{mr} = 0.2$) and signal threshold ($\mathsf{st} = 0.06$). These parameters were chosen based on previous experiments and can be regarded as an upper bound. Therefore it is reasonable to investigate if a reduction of these parameters would yield an improvement in the NN validation accuracy. These adaptations are made as depicted in the following listing:

- $\mathsf{mr} = 0.1$
- $\mathsf{st} = 0.02$

Figure 4.22 shows the resulting training and validation history. The validation accuracy significantly improved to 0.914 ± 0.002 which is an improvement of 6.9 % compared to the result achieved after applying the *Pixel-Channel-Mapping* from paragraph 4.4.

Compared to the plot in figure 4.21, the overfitting reduced in this experiment. However,

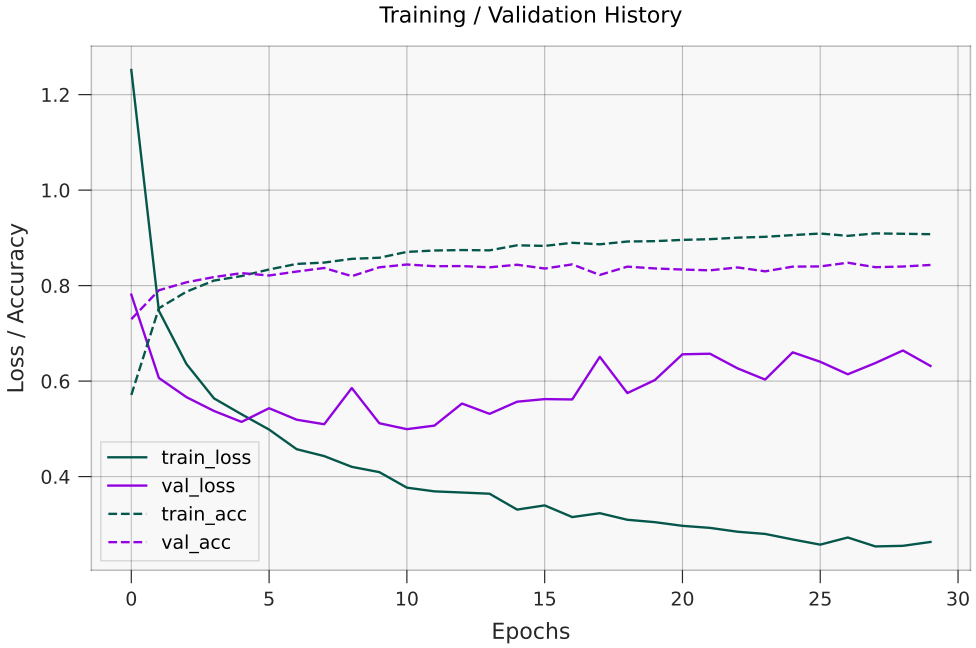


Figure 4.21: Loss & Accuracy plot over epochs after introducing the Pixel-Channel-Mapping from paragraph 3.3. Overfitting reduced and accuracy improved. Validation accuracy: 0.845 ± 0.003

the distinct spikes in the plot in figure 4.22 indicate large gradients. According to the discussion in paragraph 2.1.2, reducing the learning rate can address this problem. In the experiment which generated the plot in figure 4.23 the step size of the ADAM optimizer is as a consequence reduced from 0.001 to 0.0001 At the same time, the number of epochs was increased to 40 to account for the slower training speed as also explained in paragraph 2.1.2.

The resulting plot shows a clearly improved learning curve without spikes and reduced overfitting. However, the validation accuracy reduced to 0.903 ± 0.001. Since the number of epochs is already increased to address the slower training speed and overfitting still remains, training for an even longer period of time is not expected to yield a better validation accuracy.

For the reason that, the adaptation of the hyperparameters in listing 4.4 made up most of the performance improvement, following experiments investigate which parameter exactly achieved this improvement. It showed, that reverting $\mathsf{mr} = 0.2$ as in the initial setup in listing 4.4 yields a similar validation accuracy of 0.902 ± 0.001. Besides this mostly identical validation accuracy, the history plot remained similar to the previous experiment and is therefore omitted here, but included in the appendix in figure A.3. This allows for the conclusion that reducing the signal threshold mainly improved the validation accuracy.

Since it can be assumed that the angular filter parameterized with $\mathsf{mr} = 0.2$ does not cause a relevant reduction in validation accuracy, it is preferable to $\mathsf{mr} = 0.1$ since a larger value results in a smaller circuit and therefore the reduction of possible noise sources. In this respect, it remains to be tested whether a further reduction of the signal threshold

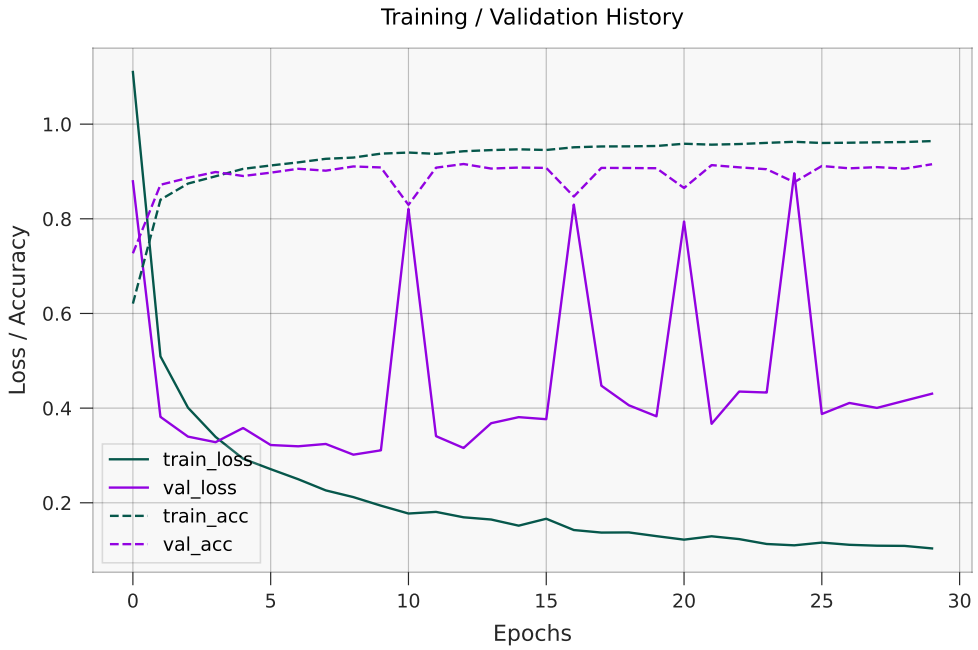


Figure 4.22: Loss & Accuracy plot over epochs after changing QFT parameters to $\mathsf{mr} = 0.1$ and $\mathsf{st} = 0.02$. Accuracy improved, but severe spikes in the validation loss become visible. Validation accuracy: 0.914 ± 0.002

is accompanied by an improvement of the validation accuracy. Figure 4.24 shows the resulting plot for a signal threshold of $\mathsf{st} = 0.01$.

The validation accuracy of this experiment reached 0.916 ± 0.001 which is an improvement of 1.4 */sipercent* compared to the previous experiment. This once more motivated further reduction of the signal threshold down to $\mathsf{st} = 0.001$ which in turn decreased the validation accuracy to 0.914 ± 0.004. The plot of this experiment is not shown here, since no relevant change in the training history occurred, but is included in the appendix in figure A.4. It should also be noted that, the variance in the validation accuracy increased compared to the previous experiment. For this reason and because the reduced signal threshold resulted in an overall longer execution time as more QFTs were performed, the most appropriate parameters for this setup regarding the angular filter and signal threshold can be noted as follows:

- $\mathsf{mr} = 0.2$
- $\mathsf{st} = 0.01$

Further properties of this setup include the noise model *"ibmq_guadalupe"*, applied noise subtraction method and the *Pixel-Channel-Mapping*.

A final performance evaluation of the NN on the test-dataset was conducted which resulted in a test-accuracy of 89.92 %. In comparison with validation accuracy of 0.916 ± 0.001, this result is well acceptable and a minor performance drop was expected due to the still persistent overfitting indicated by the slightly increasing validation loss in figure 4.24.

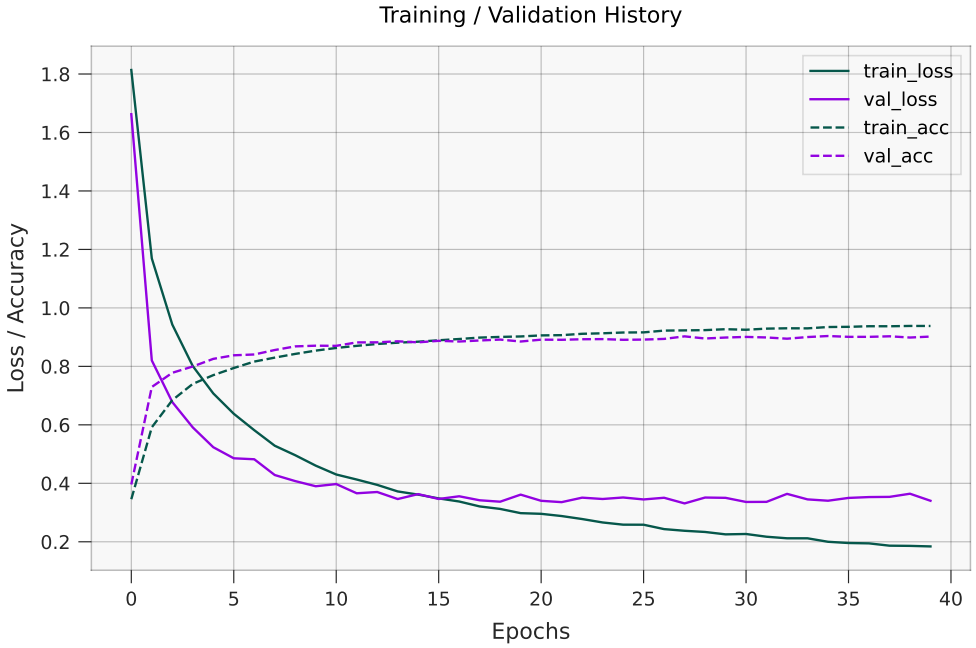


Figure 4.23: Loss & Accuracy plot over epochs after reducing stepsize of the ADAM optimizer. Spikes from the previous experiment vanished and overfitting reduced as well. Validation accuracy: 0.903 ± 0.001

4.5. Discussion

This section summarizes and discusses the findings of preceding experiments.

At first, the angular filter with increasing parameter mr in a QFT applied to a harmonic signal was tested. It turned out that on a real device no significant improvement was noticeable and in simulation with and without noise, the accuracy of the QFT only decreased with increasing mr. However, by combining the angular filter with the noise subtraction approach, it was found that there is an ideal value for the former at which the result of the grader improved by a factor of 1.51.

Subsequently, the STQFT was evaluated using a synthetic chirp signal. Experiments were carried out in the simulator with the noise model from *"ibmq_guadalupe"*. The initial experiment revealed artefacts in the spectrograms which could be attributed to faulty rotation gates. This was confirmed by the manipulation of the classic FFT where similar artefacts were observed. However, these artefacts could be almost completely suppressed by the noise subtraction method. Using the noise model of a larger quantum device with lower qubit error rates was also examined which indeed reduced the artefacts visible in the spectrogram.

The STQFT and aforementioned approaches were then evaluated on speech signals from the dataset used within this thesis. It was shown that due to the low amplitude of the signal and the normalization inherent with the QFT, spectral components appeared in regions which do not exist in the reference spectrogram obtained from the STFT. Therefore, a

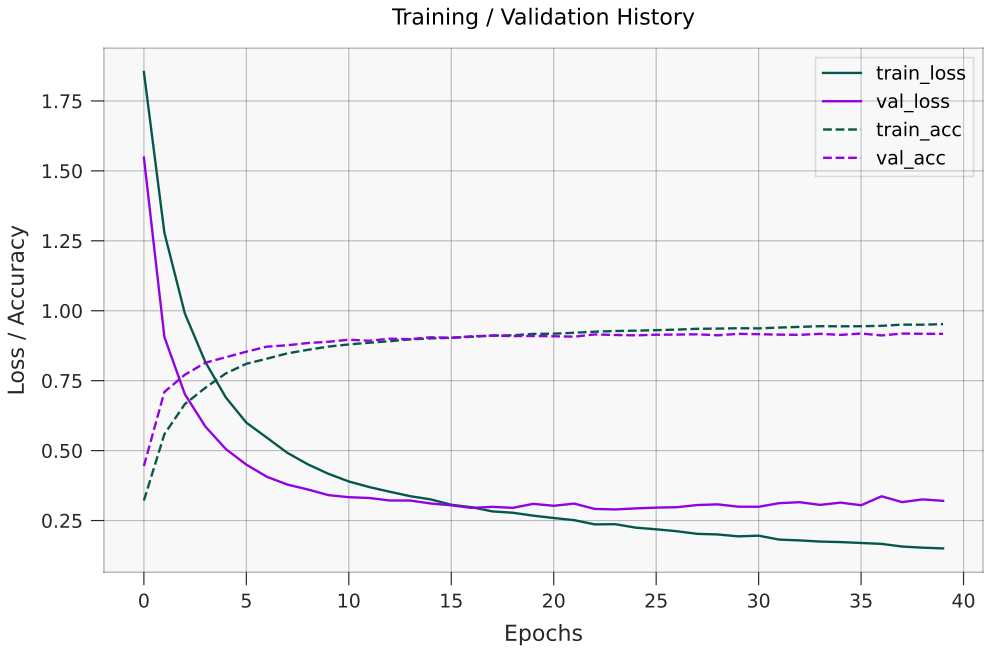


Figure 4.24: Loss & Accuracy plot over epochs after reducing signal threshold further to $\mathsf{st} = 0.01$. Accuracy improved. Validation accuracy: 0.916 ± 0.001

signal threshold filter was introduced which returns an empty spectrum if the amplitude of the signal is smaller than a fixed value within the particular QFT window. This not only reduced the number of QFTs and therefore increased execution speed, but also resulted in a spectrogram similar to that of the STFT. In comparison with this reference, it was found that raising and lowering amplitudes in the voice as at the beginning and end of a speech signal cannot be processed by the STQFT as well as with the STFT due to the normalization of the QFT. Additionally, an evaluation of the angular filter based on the spectrogram of the speech signal was conducted to find an appropriate value for its parameter.

Next, using the previously evaluated signal threshold of $\mathsf{st} = 0.06$ and an angular filter parameter of $\mathsf{mr} = 0.2$, STQFT-based spectrograms of the training and validation set were generated and the NN was trained. Since heavy fluctuations and overfitting was visible in the resulting history plot, the hyperparameters of the NN were adapted. This reduced the overall fluctuations in the training history, although overfitting was still present. By applying a *Pixel-Channel-Mapping* as a replacement for the quanv layer, the overall validation accuracy improved by over 2.45 % compared to the initial training. A learnable convolution layer was tested but decreased the validation accuracy. Using a the noise model from *"ibmq_melbourne"* decreased the validation accuracy as well, which is why the noise model was reverted to *"ibmq_guadalupe"* in further experiments.

Since the initially chosen parameters for the signal threshold and angular filter can be considered as upper limits, their values were reduced. It turned out that decreasing the signal threshold improved validation accuracy significantly to 0.916 ± 0.001, while

decreasing the parameter of the angular filter did not yield further improvements. Another experiment revealed an appropriate value for the signal threshold such that the STQFT parameters were fixed to $\mathsf{st} = 0.002$ and $\mathsf{mr} = 0.2$.

With this setup, an overall performance measure of the NN on the test-dataset was conducted yielding the final test-accuracy of 89.92 %. Given the still persistent overfitting indicated by the training history, this result is well acceptable and provides a solid baseline for further improvements. It is conceivable that adjustments to the NN could reduce overfitting, but in this work the architecture was explicitly fixed except for the hyperparameters to allow direct comparison with the baseline from [YQC+20b].

5. Conclusion

This thesis aims to evaluate the possibility of implementing a quantum Fourier transform (QFT) in a speech processing (SP) pipeline, motivated by the potential speedup of this algorithm compared to its classical counterpart, the fast Fourier transform (FFT). The focus was on speech recognition (SR), as many of the basic insights gained in this topic can also be applied to speech synthesis (SS) as well, but the latter would have required the development of a completely new framework.

Since one of the biggest challenges of quantum computing today is hardware-induced noise, mitigation approaches were developed to deal with this issue. These approaches include the noise subtraction method described in section 3.1.3 where the spectrum of a constant signal, obtained from the QFT, is used to address the low-frequency tendency effect discussed in section 3.1.2 and to mitigate noise resulting from gate and measurement errors on the quantum bits (qubits) themselves. Furthermore, an angular filter presented in paragraph 3.1.3 was implemented to reduce the overall gate count of the QFT and therefore remove their intrinsic noise from the quantum circuit.

The aforementioned approaches were first evaluated using a synthetic harmonic signal. To obtain consistent and traceable measurement results, a test framework was developed which, in addition to evaluation, also enables versioning and identification with the generating source code. Spectra generated by the QFT were objectively measured by a grader against the FFT as a reference. The evaluation of these grader results confirmed the hypothesized improvement obtained from the noise subtraction method and angular filter approach. Depending on the application constraints it is necessary to find a suitable limit for the angular filter, otherwise the bandwidth of the QFT might be too narrow, which in turn could degrade the resulting spectrogram.

Since speech processing is the main focus of this thesis and requires time-variant transformation methods, the short time quantum Fourier transform (STQFT) was derived as a quantum equivalent of the Short Time Fourier transform (STFT) and, similar to the QFT, first evaluated on a synthetic chirp signal. It showed that noise from real devices leads to artefacts in the spectrogram similar to those resulting from violating the Nyquist theorem which, however, was fulfilled. Further investigation revealed that noisy rotation gates were the cause of these artifacts and the noise subtraction approach was found to be able to effectively mitigate them. Increasing the degree of the angular filter also led to an increase in the artifacts, but only above a certain threshold value, which allowed an optimal value for this methods to be derived.

After evaluating synthetic signals, a real speech sample was tested on this setting to validate if the observed effects persisted. For this application, simulated quantum devices with noise models had to be used, since the amount of qubits on publicly available noisy intermediate scale quantum (NISQ) devices is not sufficient. It turned out that, compared to the STFT, the spectrogram from the STQFT contained spectral components (artefacts) in time periods where they should not exist. This effect, discussed in section 3.2, was successfully removed by introducing a signal threshold which disables the QFT in time periods during which the amplitude of the signal is lower than a fixed threshold which in turn was subject of evaluation in subsequent experiments.

Finally, the STQFT was applied together with the noise and artifact suppression methods to a voice command dataset. The spectrograms generated were fed into an existing neural network and the performance on the training and validation data was logged and evaluated. Since the accuracy results clearly indicated overfitting, it was found that further modifications to the network might improve the generalization. Although these modifications improved the generalization to a certain degree, slightly reducing the angular filter and signal threshold finally led to a model which was found to be optimal given the adjustable parameters. The resulting accuracy on the test dataset of 89.92 % motivates further research within this area and although the baseline of 95.12 % is not surpassed, the results obtained in this work suggest that an application of QFT in an SR framework is conceivable, even in the presence of noise. Although the publicly restricted access to quantum computers and the relatively long waiting queues currently forbid an application in scenarios where a quick response time of such a system is required, according to the findings of this thesis, an application even on NISQ devices is generally possible. All source code used in this work is publicly available on Github [Str21a].

For the reason that any quantum circuit is unitary by definition and thus invertible, findings of this thesis may also apply on the inverse quantum Fourier transform (IQFT) necessary for the SS. However, as stated at the beginning of this section, SS requires a different framework and the noise impact on the synthesized signal might be more problematic for consumer applications than it is for speech recognition with a neural network (NN). Therefore, these investigations should be subject of future work where aforementioned framework is implemented and the applicability of the QFT in SS evaluated.

While the QFT was chosen because of its good algorithmic properties regarding its complexity on quantum computers and its direct connection to the FFT used in SR, other transformations such as the wigner ville distribution (WVD) or wavelet transform (WT) are also conceivable. Classically, these transformations can address the time-frequency resolution tradeoff but are due to their additional computational complexity not commonly used in SP where an increased resolution is not required. Nonetheless, it might be worth to investigate in applying these transformations on quantum systems as to the best of my knowledge, there are no approaches in the literature yet.

In this work, in order to improve the results obtained from QFT, different approaches were introduced and their parameters were manually optimized for application in SR. However, training these parameters in an end-to-end architecture, where the NN is supposed to find an optimal configuration for a given dataset, can be another topic of further research. Finally, the encoding of classical information into quantum states is a bottleneck for a potential speedup gained through the QFT and therefore point of consideration for improvement.

Summarizing, quantum computing (QC) is a promising opportunity and although noisy hardware and the relatively small number of qubits currently limit the practical employment of SR or SS applications, a general confirmation of the applicability of the QFT is shown within this thesis and therefore motivates further investigations in this area of research.

List of Figures

2.1. Convolution operation and parameters. 11
2.2. Vanilla recurrent neural network (RNN) schematic. 13
2.3. long short-term memory (LSTM) schematic. 14
2.4. General quantum algorithm structure. 15
2.5. Exemplary bloch sphere 16
2.6. Quantum circuit with a Hadamard gate on the first and `X` on the second qubit. 19
2.7. Bloch spheres of the $|0\rangle$, $|+\rangle$ and $|1\rangle$ states. 20
2.8. Circuit representation of the $\texttt{C}_\texttt{X}$. 21
2.9. Swap gate `SWAP` and its decomposition in circuit notation 21
2.10. Phase gate `P` in circuit notation. 22
2.11. Circuit containing a single Hadamard Gate `H` and a measurement operator. 22
2.12. Histogram of a `H` gate applied on a qubit in the $|0\rangle$ state. 23
2.13. Circuit containing a Hadamard (`H`) and a Controlled-Not Gate ($\texttt{C}_\texttt{X}$) as well as measurements on both qubits. 24
2.14. Histogram of a $\texttt{C}_\texttt{X}$ gate applied on a target qubit in the $|+\rangle$ state yielding a Bell state. 24
2.15. Circuit consisting of d subsequent `X` gates. 26
2.16. Measurement of a circuit with ideal and noisy rotation gates. 27
2.17. State transition and probability mapping modelling a noisy rotation gate. . 28
2.18. Measurements of a circuit with gate errors, measurement errors and additional stochastic fluctuations. 29
2.19. Decomposition of a noisy gate by modeling as a case decision. 29
2.20. Shrunk applicable Bloch sphere 31
2.21. Decay of measurable amplitudes of subsequent noisy gates. 32
2.22. Decomposition of $\mathbf{I}_k \oplus \mathbf{\Omega}_k$ into k controlled phase gates $\texttt{CP}(\phi)$. 37
2.23. Permutation matrix $\mathbf{P}_n$ written in circuit notation. 38
2.24. General QFT circuit. 38
2.25. quantum machine learning (QML) approaches. 39
2.26. "QML-AM" architecture from [YQC+20b]. 40
2.27. Self-attention U-Net architecture from [YQC+20a]. 40
2.28. Original quantum convolution neural network (QCNN) structure proposed in [YQC+20b]. 41

2.29. Quantum convolutional (quantum convolutional (quanv)) layer from [YQC+20b]. 41

3.1. Full banded and bandlimited 16-QFT. 46
3.2. Approach to integrate the STQFT in an existing SR pipeline. 48
3.3. Pixel-Channel-Mapping Filter as replacement for the quanv layer. 49

4.1. Framework for validation of individual transformations. 52
4.2. Implementation of the QFT with an abstract representation of available methods. 56
4.3. Harmonic signal and its spectrum with peaks at 125 and 250 Hz. 57
4.4. QFT generated spectra of a harmonic signal and varying parameter mr. . . 59
4.5. Grader results of the angular filter approach 60
4.6. Map view of *"ibmq_quito"*. 61
4.7. QFT on real device *"ibmq_quito"* with noise subtraction. 62
4.8. Grader results from noise mitigated QFT on the real device *"ibmq_quito"*. . 62
4.9. STQFTs of a synthetic chirp signal with noise mitigation. 64
4.10. Map view of *"ibmq_casablanca"*. 64
4.11. STQFT of a chirp signal in simulation with noise model from *"ibmq_guadalupe"*. 65
4.12. STQFTs of a chirp signal with varying rotation gate error er. 66
4.13. STQFTs of a chirp signal in simulation with noise model from *"ibmq_casablanca"* and varying mr. 67
4.14. STQFTs of a chirp signal in simulation with noise model from *"ibmq_casablanca"* and varying mr and noise mitigation. 68
4.15. STFT and STQFT of a speech signal. 69
4.16. STQFTs of a speech signal (*"left"*) with varying *signal threshold filter*. . . . 70
4.17. STQFTs of a speech signal with and without noise mitigation. 71
4.18. STQFT of a speech signal with varying mr parameter. 72
4.19. Loss & Accuracy plot over epochs for the initial experiment. 73
4.20. Loss & Accuracy plot over epochs after adapting NN hyperparameters. . . . 74
4.21. Loss & Accuracy plot over epochs after introducing the Pixel-Channel-Mapping. 75
4.22. Loss & Accuracy plot over epochs after adapting QFT parameters to mr = 0.1 and st = 0.02. 76
4.23. Loss & Accuracy plot over epochs after reducing stepsize of the ADAM optimizer. 77
4.24. Loss & Accuracy plot over epochs after reducing signal threshold to st = 0.01 78

A.1. Map view of *„ibmq_guadalupe“*. 90

A.2. Loss & Accuracy plot over epochs with noise model from *„ibmq_melbourne“*. 92
A.3. Loss & Accuracy plot over epochs after setting $\mathsf{mr} = 0.2$ and $\mathsf{st} = 0.02$. . . . 92
A.4. Loss & Accuracy plot over epochs after further reducing the signal threshold $\mathsf{st} = 0.001$. 93

Symbols

$\imath$	Imaginary Unit
ψ	Quantum State
ϕ	Alternate Quantum State
Φ	Gate Parametrization Angle
mr	Minimum Rotation
er	Rotation Error
st	Signal Threshold
δ	Dirac Impulse
rect	Rectangular Function
sinc	Sinc Function
t	Time
f	Frequency
x	Signal (continous or discrete)
X	Signal's spectrum
H	Hamilton Operator
$\mathcal{H}$	Hilbert Space
$\mathbb{R}$	Complex Numbers
$\mathbb{C}$	Complex Numbers
$\mathbb{N}$	Natural Numbers
$\mathbb{Z}$	Integer Numbers
$\mathtt{H}$	Hadamard Operator
$\mathtt{X}$	Pauli $\mathtt{X}$ Operator
$\mathtt{Y}$	Pauli $\mathtt{Y}$ Operator
$\mathtt{Z}$	Pauli $\mathtt{Z}$ Operator
$\mathtt{I}$	Identity Operator
$\mathtt{C_X}$	Controlled-Not Operator
$\mathtt{R_X}$	$\mathtt{R_X}$ Gate
$\mathtt{R_Y}$	$\mathtt{R_Y}$ Gate
$\mathtt{R_Z}$	$\mathtt{R_Z}$ Gate
$\mathtt{P}$	Phase Gate ($= \mathtt{R_Z}$)
$\mathtt{CP}$	Controlled Phase Gate
$\mathtt{SWAP}$	SWAP Gate
$\mathcal{O}$	O-Notation
$\dagger$	Complex Conjugate Transpose
$\otimes$	Tensorproduct
$\oplus$	Direct sum
$\mathcal{F}$	Fourier Transform
$\mathcal{F}_{\mathrm{QFT}}$	Quantum Fourier Transform
$\tilde{M}$	Matrix consisting of entries exponents only
M'	Matrix with permutated rows
$\boldsymbol{M}$	Matrix with exponential dimension $2^n, \quad n \in \mathbb{N}$

A. Additional Figures

Qubit	Readout assignment error	`CX`-error (control_target)
q0	4.000e−2	0_1 : 6.062e−3
q1	2.640e−2	1_3 : 1.557e−2 1_2 : 5.816e−3 1_0 : 6.062e−3
q2	2.020e−2	2_1 : 5.816e−3
q3	3.240e−2	3_4 : 1.590e−2 3_1 : 1.557e−2
q4	2.150e−2	4_3 : 1.590e−2

Table A.1: Absolute values of the readout assignment- and `CX` error from the 5-qubit quantum device *"ibmq_quito"* from [Qua21]. The average readout assignment error is 2.998e−2 and the average `CX`-error is 1.027e−2. Calibration date: 30.10.2021

Qubit	Readout assignment error	CX-error (control_target)
q0	4.320e−2	0_1 : 1.484e−2
q1	2.220e−2	1_3 : 6.713e−3 1_2 : 9.569e−3 1_0 : 1.484e−2
q2	2.940e−2	2_1 : 9.569e−3
q3	1.980e−2	3_5 : 1.168e−2 3_1 : 6.713e−3
q4	3.070e−2	4_5 : 1.091e−2
q5	1.160e−2	5_6 : 9.475e−3 5_4 : 1.091e−2 5_3 : 1.168e−2
q6	2.360e−2	6_5 : 9.475e−3

Table A.2: Absolute values of the readout assignment- and CX error from the 7-qubit quantum device *"ibmq_casablanca"* from [Qua21] from [Qua21]. The average readout assignment error on *"ibmq_casablanca"* is 2.300e−2 and the average CX-error is 1.035e−2. Calibration date: 30.10.2021

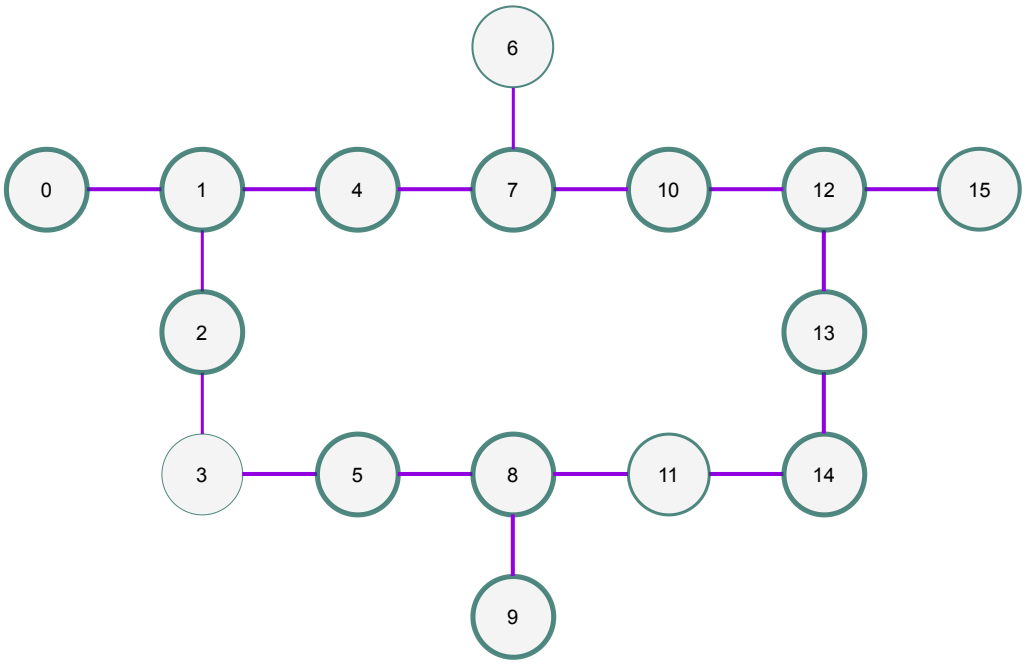


Figure A.1: Map view of *„ibmq_guadalupe"* with „IBM Quantum Falcon r4P processor". *Readout assignment error* indicated by the qubit's line thickness (the thicker the smaller the error) and CX-*error* indicated by the connections (the thicker the smaller the error)

Qubit	Readout assignment error	CX-error (control_target)
q0	1.140e−2	0_1 : 9.788e−3
q1	1.630e−2	1_0 : 9.788e−3 1_4 : 6.241e−3 1_2 : 2.266e−2
q2	1.860e−2	2_3 : 4.723e−2 2_1 : 2.266e−2
q3	1.254e−1	3_2 : 4.723e−2 3_5 : 7.773e−2
q4	1.840e−2	4_7 : 1.228e−2 4_1 : 6.241e−3
q5	1.640e−2	5_3 : 7.773e−2 5_8 : 7.222e−3
q6	1.026e−1	6_7 : 4.265e−2
q7	1.670e−2	7_4 : 1.228e−2 7_6 : 4.265e−2 7_10 : 6.175e−3
q8	2.520e−2	8_9 : 7.206e−3 8_11 : 7.140e−3 8_5 : 7.222e−3
q9	1.150e−2	9_8 : 7.206e−3
q10	1.230e−2	10_12 : 1.399e−2 10_7 : 6.175e−3
q11	6.110e−2	11_8 : 7.140e−3 11_14 : 6.742e−3
q12	1.410e−2	12_15 : 8.269e−3 12_10 : 1.399e−2 12_13 : 8.399e−3
q13	1.320e−2	13_14 : 8.324e−3 13_12 : 8.399e−3
q14	1.900e−2	14_13 : 8.324e−3 14_11 : 6.742e−3
q15	2.420e−2	15_12 : 8.269e−3

Table A.3: Absolute values of the readout assignment- and CX error from the 16-qubit quantum device *"ibmq_guadalupe"* from [Qua21]. The average readout assignment error is 3.165e−2 and the average CX-error is 1.825e−2. Calibration date: 30.10.2021

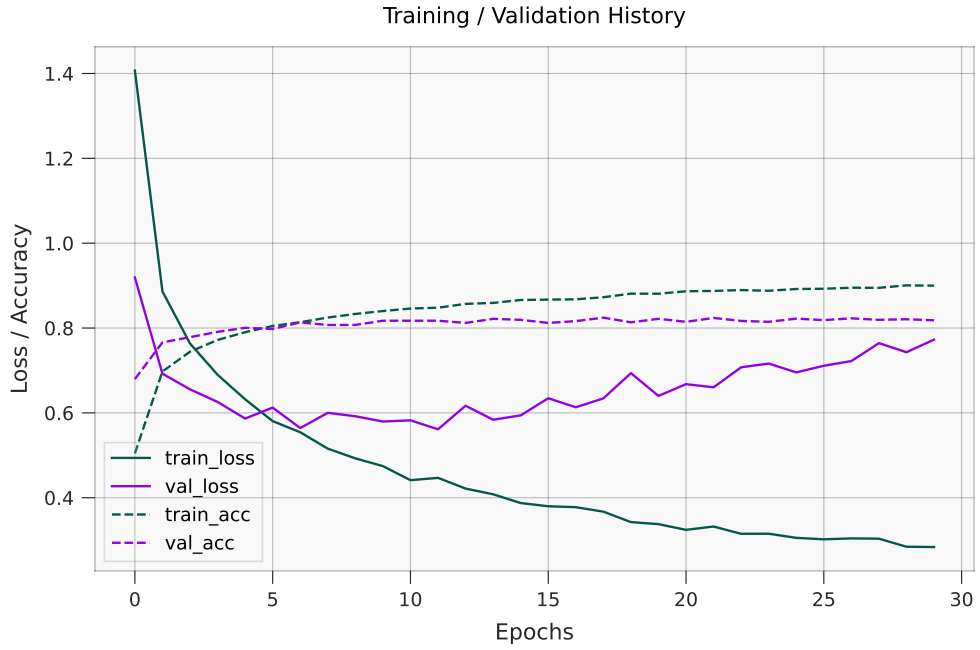


Figure A.2: Loss & Accuracy plot over epochs with noise model from *„ibmq_melbourne“*. Validation accuracy: 0.825 ± 0.001

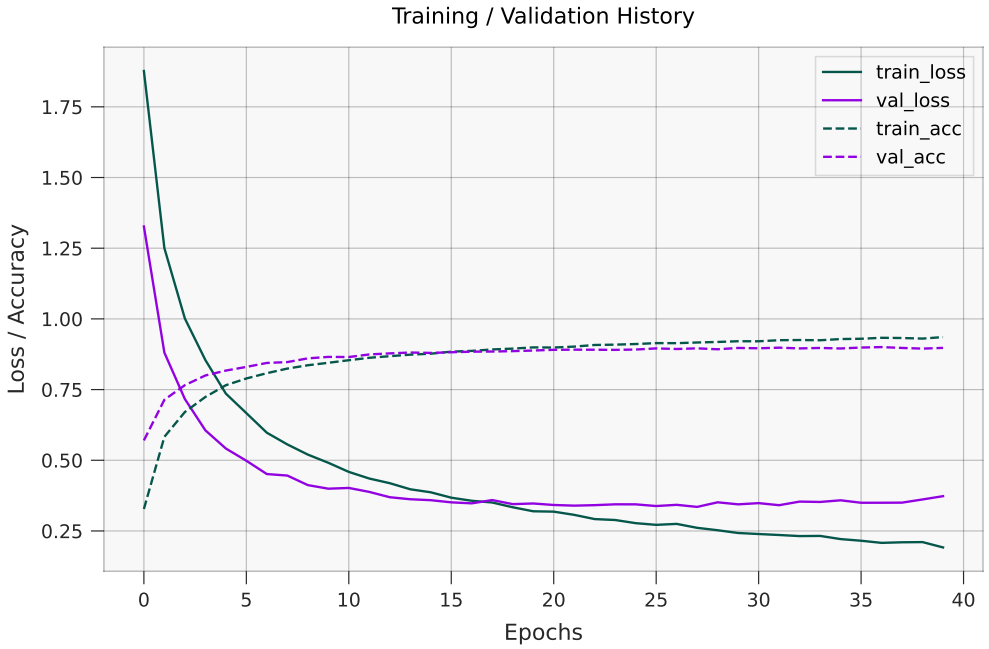


Figure A.3: Loss & Accuracy plot over epochs after setting $\mathsf{mr} = 0.2$ and $\mathsf{st} = 0.02$. Validation accuracy: 0.902 ± 0.001

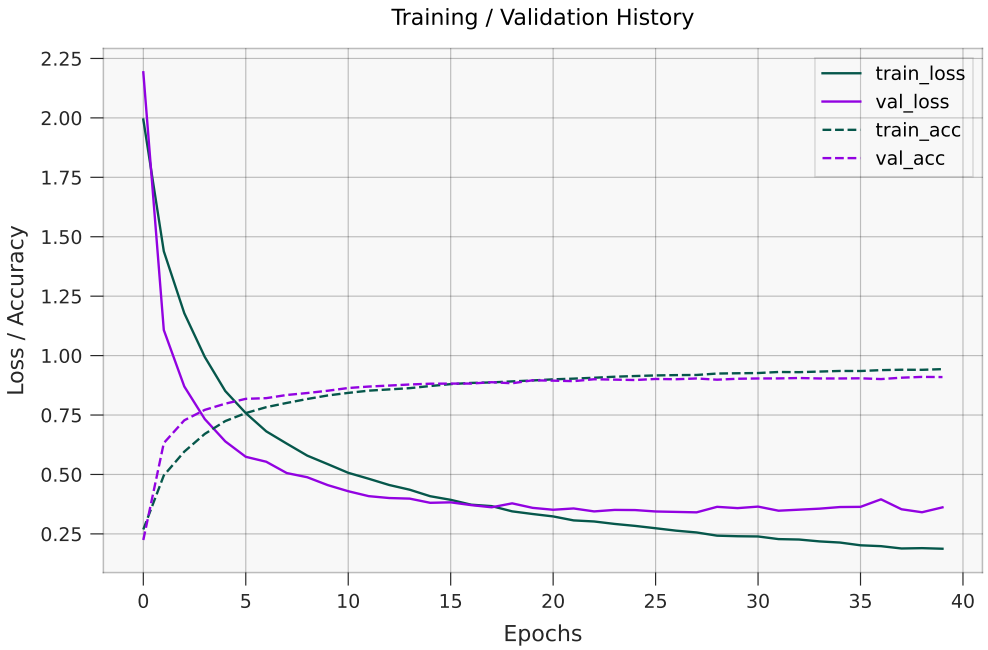


Figure A.4: Loss & Accuracy plot over epochs after further reducing the signal threshold $\mathsf{st} = 0.001$. Angular filter set to $\mathsf{mr} = 0.2$. Validation accuracy: 0.914 ± 0.004

B. Abbreviations

API	application program interface
SR	speech recognition
SP	speech processing
SS	speech synthesis
QC	quantum computing
qubit	quantum bit
quanv	quantum convolutional
QCNN	quantum convolution neural network
NISQ	noisy intermediate scale quantum
QML	quantum machine learning
NN	neural network
ML	machine learning
LP	learning problem
GD	gradient descent
BGD	batch gradient descent
SGD	stochastic gradient descent
LSTM	long short-term memory
RNN	recurrent neural network
CNN	convolutional neural network
FT	Fourier transform
DFT	discrete Fourier transform
FFT	fast Fourier transform
STT	short time transformation
STFT	Short Time Fourier transform
WVD	wigner ville distribution
WT	wavelet transform
QFT	quantum Fourier transform
IQFT	inverse quantum Fourier transform
STQFT	short time quantum Fourier transform
SCM	source control management
CPU	central processing unit
GPU	graphic processing unit

Bibliography

[AAA+21] M. S. ANIS, H. Abraham, AduOffei, R. Agarwal, G. Agliardi, M. Aharoni, I. Y. Akhalwaya, G. Aleksandrowicz, T. Alexander, M. Amy et al., "Qiskit: An open-source framework for quantum computing," 2021.

[AAB+19] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell et al., "Quantum supremacy using a programmable superconducting processor," Nature, vol. 574, no. 7779, pp. 505–510, Oct. 2019. [Online]. Available: https://doi.org/10.1038/s41586-019-1666-5

[ACB+20] A. Asfaw, A. Corcoles, L. Bello, Y. Ben-Haim, M. Bozzo-Rey, S. Bravyi, N. Bronn, L. Capelluto, A. C. Vazquez, J. Ceroni et al. (2020) Learn quantum computation using qiskit. [Online]. Available: http://community.qiskit.org/textbook

[BEH+04] N. Boulant, J. Emerson, T. F. Havel, D. G. Cory, and S. Furuta, "Incoherent noise and quantum information processing," The Journal of Chemical Physics, vol. 121, no. 7, p. 2955–2961, Aug 2004. [Online]. Available: http://dx.doi.org/10.1063/1.1773161

[Bis95] C. M. Bishop, "Training with Noise is Equivalent to Tikhonov Regularization," Neural Computation, vol. 7, no. 1, pp. 108–116, 01 1995. [Online]. Available: https://doi.org/10.1162/neco.1995.7.1.108

[BIS+20] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, M. S. Alam, S. Ahmed, J. M. Arrazola, C. Blank, A. Delgado, S. Jahangiri et al., "Pennylane: Automatic differentiation of hybrid quantum-classical computations," 2020.

[Blo46] F. Bloch, "Nuclear induction," Physical Review, vol. 70, no. 7-8, pp. 460–474, oct 1946.

[CBY20] D. Camps, R. V. Beeumen, and C. Yang, "Quantum fourier transform revisited," Numerical Linear Algebra with Applications, vol. 28, no. 1, sep 2020.

[CT65] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," Mathematics of Computation, vol. 19, no. 90, pp. 297–297, may 1965.

[DV16] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," Mar. 2016.

[Fey82] R. P. Feynman, "Simulating physics with computers," International Journal of Theoretical Physics, vol. 21, no. 6-7, pp. 467–488, jun 1982. [Online]. Available: https://doi.org/10.1007/BF02650179

[FPL19] H. J. Fernando Puente León, Signale und Systeme. Muenchen: Gruyter, Walter de GmbH, Sep. 2019. [Online]. Available: https://www.ebook.de/de/product/36259584/fernando_puente_leon_holger_jaekel_signale_und_systeme.html

[FTYA21] J. L. E. K. Fendji, D. M. Tala, B. O. Yenke, and M. Atemkeng, "Automatic speech recognition using limited vocabulary: A survey," CoRR, vol. abs/2108.10254, 2021. [Online]. Available: https://arxiv.org/abs/2108.10254

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016, http://www.deeplearningbook.org.

[GP21] S. Gupta and A. Panghal, "Performance analysis of fir filter design by using rectangular, hanning and hamming windows methods," 10 2021.

[HD10] X. Huang and L. Deng, An Overview of Modern Speech Recognition, handbook of natural language processing, second edition, chapter 15 ed. Chapman & Hall/CRC, Jan. 2010, pp. 339–366. [Online]. Available: https://www.microsoft.com/en-us/research/publication/an-overview-of-modern-speech-recognition/

[Hid19] J. D. Hidary, Quantum Computing: An Applied Approach. Springer International Publishing, 2019.

[HMvdW+20] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith et al., "Array programming with NumPy," Nature, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[HS97] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, pp. 1735–80, 12 1997.

[Hun07] J. D. Hunter, "Matplotlib: A 2d graphics environment," Computing in Science & Engineering, vol. 9, no. 3, pp. 90–95, 2007.

[IH06] H. Imai and M. Hayashi, Quantum Computation and Information, H. Imai and M. Hayashi, Eds. Springer Berlin Heidelberg, 2006.

[IS15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," CoRR, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[JJS20] J. N. J. J. Sakurai, Modern Quantum Mechanics. Cambridge University Pr., Oct. 2020. [Online]. Available: https://www.ebook.de/de/product/39228511/j_j_sakurai_jim_napolitano_modern_quantum_mechanics.html

[JNN13] J. Johansson, P. Nation, and F. Nori, "QuTiP 2: A python framework for the dynamics of open quantum systems," Computer Physics Communications, vol. 184, no. 4, pp. 1234–1240, Apr. 2013.

[KB17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[KK01] J. F. Kolen and S. C. Kremer, Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies, 2001, pp. 237–243.

[Kro21] F. Kronthaler, Statistik angewandt mit Excel. Springer Berlin Heidelberg, 2021.

[LB20] F. Leymann and J. Barzen, "The bitter truth about gate-based

quantum algorithms in the NISQ era," Quantum Science and Technology, vol. 5, no. 4, p. 044007, Sep. 2020. [Online]. Available: https://doi.org/10.1088/2058-9565/abae7d

[LMK+21] R. LaRose, A. Mari, S. Kaiser, P. J. Karalekas, A. A. Alves, P. Czarnik, M. E. Mandouh, M. H. Gordon, Y. Hindy, A. Robertson et al., "Mitiq: A software package for error mitigation on noisy quantum computers," 2021.

[Min21] Z. K. Minev, "Introduction to quantum noise - qiskit summer school '21," 2021, qiskit Summer School '21 - Introduction To Quantum Noise.

[Mit07] T. M. Mitchell, "The need for biases in learning generalizations," 2007.

[MRL+15] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in Proceedings of the 14th python in science conference, vol. 8, 2015.

[Mü21] M. Müller, Fundamentals of Music Processing. Springer International Publishing, 2021.

[NB13] Y. S. Nam and R. Blümel, "Scaling laws for shors algorithm with a banded quantum fourier transform," Physical Review A, vol. 87, no. 3, p. 032333, mar 2013.

[PB11] M. Plesch and Č. Brukner, "Quantum-state preparation with universal gate decompositions," Physical Review A, vol. 83, no. 3, p. 032302, Mar. 2011.

[Pre18] J. Preskill, "Quantum computing in the NISQ era and beyond," p. 79, Aug. 2018.

[Qua21] I. Quantum, "Ibm quantum," 2021. [Online]. Available: https://quantum-computing.ibm.com/

[Rus15] S. Russell, Artificial intelligence : a modern approach. Pearson India Education Services Pvt. Ltd, 2015.

[Sch19] W. Scherer, "Mathematics of quantum computing." Springer International Publishing, 2019.

[Sha87] D. Shaughnessy, Speech communication : human and machine. Reading, Mass: Addison-Wesley Pub. Co, 1987.

[Sha98] C. E. Shannon, "Communication in the presence of noise," Proceedings of the IEEE, vol. 86, no. 2, pp. 447–457, Feb. 1998.

[SHK+14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," J. Mach. Learn. Res., vol. 15, no. 1, p. 1929–1958, Jan. 2014.

[Sho97] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," vol. 26, no. 5, pp. 1484–1509, 1997.

[SKX+19] M. Schuld, N. Killoran, Xanadu, . R. S. W, Toronto, M. 2L, and Canada, "Quantum machine learning in feature hilbert spaces," Physical Review Letters, vol. 122, no. 4, p. 040504, feb 2019.

[SP18] M. Schuld and F. Petruccione, Supervised Learning with Quantum Computers. Springer International Publishing, 2018, no. summarising,.

[Sra12] S. Sra, Optimization for machine learning. Cambridge, Mass: MIT Press, 2012.

[Str21a] M. Strobl, "Hybrid Quantum Speech Processing - Main Repository," 11 2021. [Online]. Available: https://github.com/stroblme/hqsp-main

[Str21b] ——, "Hybrid Quantum Speech Processing - QCNN Repository," 11 2021. [Online]. Available: https://github.com/stroblme/hqsp-qcnn

[Str21c] ——, "Hybrid Quantum Speech Processing - STQFT Repository," 11 2021. [Online]. Available: https://github.com/stroblme/hqsp-stqft

[Str21d] ——, "Hybrid Quantum Speech Processing - STQFT Repository," 11 2021. [Online]. Available: https://github.com/stroblme/hqsp-stqft-data

[SVN37] S. S. Stevens, J. Volkmann, and E. B. Newman, "A scale for the measurement of the psychological magnitude pitch," The Journal of the Acoustical Society of America, vol. 8, no. 3, pp. 185–190, Jan. 1937.

[Tea17] G. B. Team, "Speech commands dataset," Aug. 2017. [Online]. Available: https://ai.googleblog.com/2017/08/launching-speech-commands-dataset.html

[VGO+20] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright et al., "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," Nature Methods, vol. 17, pp. 261–272, 2020.

[VRD09] G. Van Rossum and F. L. Drake, Python 3 Reference Manual. Scotts Valley, CA: CreateSpace, 2009.

[Wai89] A. Waibel, "Modular construction of time-delay neural networks for speech recognition," Neural Computation, vol. 1, no. 1, pp. 39–46, mar 1989.

[Wil11] M. M. Wilde, "From classical to quantum shannontheory." Cambridge University Press, 2011, pp. xi–xii. [Online]. Available: https://arxiv.org/pdf/1106.1445.pdf

[YQC+20a] C.-H. Yang, J. Qi, P.-Y. Chen, X. Ma, and C.-H. Lee, "Characterizing speech adversarial examples using self-attention u-net enhancement," in ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, May 2020.

[YQC+20b] C.-H. H. Yang, J. Qi, S. Y.-C. Chen, P.-Y. Chen, S. M. Siniscalchi, X. Ma, and C.-H. Lee, "Decentralizing feature extraction with quantum convolutional neural network for automatic speech recognition," 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Oct. 2020. [Online]. Available: https://arxiv.org/abs/2010.13309

[Zam20] S. Zambare, "Quantum computing to find frequencies in an audio file," Jul. 2020. [Online]. Available: https://sarangzambare.github.io/jekyll/update/2020/06/13/quantum-frequencies.html

[Zic21] F. Zickert, Hands-On Quantum Machine Learning With Python : Volume 1: Get Started. City: PyQML, 2021.