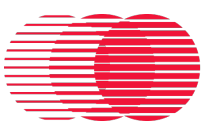
End of Study Internship Report

Quantum Kernel Methods for malware beaconing detection

Maissa BEJI

Industrial Supervisors: Luc ANDREA, Multiverse Computing
Llorenç ESPINOSA, Multiverse Computing

Academic Supervisor: Peter BROWN, Télécom Paris



MULTIVERSE
COMPUTING

بسم الله الرحمن الرحيم

الحمد لله رب العالمين له الحمد و الثناء الجميل، الصلاة والسلام على أشرف الخلق والمرسلين، سيدنا محمد المبعوث رحمة للعالمين.

أهدي هذا العمل المتواضع:

إلى **والديّ** اللذين أحاطاني بحبهما وسخّرا لي كل الوسائل لتعليمي

إلى **إخوتي** وكل **أفراد أسرتي** الذين وثقوا بي

إلى **روح جدّتي** و **روح جدّي** اللذين رسّخا فيّ حب العلم والتعلّم

إلى **أصدقائي** وكل من كان برفقتي في هذه الرحلة

إلى كل **أساتذتي** و **معلّميّ** وكل من علّمني حرفاً

إلى كل من لم يدّخر جهداً في مساعدتي

وإلى كل المتألّمين في **أرض الزيتون والبرتقال**، وإلى جميع البشر المعذّبين حول العالم
لندع إلى عالم بلا حروب ولا مجاعات، فلا معنى لتقدّم التقنيات إذا لم نستطع إنقاذ الأرواح البشرية

I dedicate this humble work:

To **my parents**, who surrounded me with their love and devoted all means to my education.

To **my siblings** and all **members of my family**, who believed in me.

To the **spirit of my grandmother** and the **spirit of my grandfather**, who instilled in me the love of knowledge and learning.

To **my friends** and everyone who accompanied me on this journey.

To all **my teachers** and **mentors**, and to everyone who taught me a single letter.

To all those who never spared any effort in helping me.

And to all those suffering in the **land of olives and oranges**, and to every oppressed human being across the world. Let us call for a world without wars and famines, for technological progress has no meaning if we cannot save human lives.

Acknowledgements

I would like to extend my heartfelt gratitude to the team at Multiverse Computing. This work would not have been possible without this opportunity.

I especially thank my teammate Boaz ATO MICAH for his constant support, technical help, and assistance in shaping this report. I also thank Borja AIZPURUA ALTUNA, whose contributions during meetings were valuable for understanding and discussing scientific challenges, and for helping me find solutions.

I am deeply grateful to my manager Llorenç ESPINOSA, whose approach made me feel like an integrated member of the team and encouraged me to follow a clear scientific path. I also thank my director Luc ANDREA for his support and guidance, as well as the Paris office team for the valuable open discussions.

I would also like to thank my academic supervisor Peter BROWN for his guidance and support throughout this academic year.

Summary

During this internship, I explored quantum kernel methods for anomaly detection in network traffic. I implemented and evaluated different approaches, including Randomized Measurement and Projected Quantum Kernel, in an unsupervised setting. The evaluation focused on the F1-score and provided a benchmark between the different kernel methods. I also learned with the team feature engineering techniques such as variance filtering, clustering, and PCA for preparing the dataset. The results highlight the strengths and limitations of the different kernels and give a baseline for future experiments. Alongside the technical work, this internship allowed me to develop software engineering practices, critical analysis, and communication skills through team collaboration and presentations in consortium meetings.

Contents

1 Internship Context

1.1. Multiverse Computing - Quantum AI revolution

Multiverse Computing is a Spanish quantum software company founded in 2019 and headquartered in San Sebastián, with offices in Paris, Munich, London, Toronto, and Sherbrooke. The company develops AI, quantum, and quantum-inspired algorithms for energy, logistics, manufacturing, mobility, life sciences, finance, cybersecurity, chemistry, materials science, and aerospace.
Its core product, *Singularity*, is a platform that gives non-experts access to quantum algorithms through an intuitive interface. In 2023, it launched *CompactifAI*, a tensor-network tool that reduces the cost and energy of training and running large language models by shrinking their size, memory, and storage footprint.
These platforms expand access to advanced computing while meeting the rising demand for efficient and sustainable AI.

1.2. Team presentation and work methodology

I was part of a collaborative team directed by Luc ANDREA and managed on a daily basis by Llorenç ESPINOSA. The team also included Boaz ATO MICAH, a quantum software engineer, and Borja AIZPURUA ALTUNA, a scientist. Working with them gave me constant opportunities to learn and improve.

Our workflow was structured around different types of meetings.

- Daily team meetings were a space to share progress and solve technical problems together. At first, I felt nervous presenting in front of more experienced researchers, but over time I became more confident. These sessions helped me sharpen my debugging skills, explain my work clearly, and engage more actively in discussions on scientific literature.

- Weekly meetings with the director and CEO focused on planning and preparing for external interactions. Here, I realized how research connects with business strategy. I learned how to prioritize tasks, adapt goals to client needs, and balance scientific depth with practical constraints.

- Bi-weekly meetings with our client and partner IQM were formal and professional. Presenting my results to external experts felt challenging in the

beginning, but it pushed me to structure my work better and anticipate questions. These sessions also gave me direct exposure to real-world perspectives on cybersecurity and quantum hardware.

I also had regular one-on-one meetings with my manager Llorenç and monthly discussions with Luc. These open exchanges gave me a safe space to ask questions, reflect on my progress, and receive constructive feedback.

Overall, this experience taught me how to navigate a professional research environment, combine technical and communication skills, and grow both as a researcher and as a team member.

1.3. Team tools

Our team relied on a set of digital tools that structured both collaboration and project management.

- Microsoft Teams was used for formal communication, while Slack supported quick exchanges. I noticed that this balance kept the team connected without overwhelming us with messages.
- GitLab served as our platform for version control and code sharing. Working in this environment taught me how to manage branches, review code efficiently, and follow professional development workflows.
- Docker containers standardized our coding environments. This eliminated compatibility issues and helped me understand how containerization supports reproducibility in research projects.
- MLflow was employed to track experiments. Using it gave me hands-on experience in monitoring and comparing machine learning runs, which improved how I documented and reproduced results.
- Flyte orchestrated workflows, and AWS instances provided the computational power for large-scale simulations. This exposure showed me how cloud resources and automated pipelines can accelerate research.

This integrated toolset shaped the way I worked: I became more rigorous in documentation, more efficient in collaboration, and more aware of how professional software practices connect with scientific research.

1.4. Context of the project

I contributed to the AQACYB project (Quantum Advantage for Cyber Threat Analysis), a collaboration between a French insurance company, IQM, and Multiverse Computing. The project brings together skills in cybersecurity, quantum computing, and machine learning.

It aims to develop quantum-enhanced anomaly detection algorithms to address practical cybersecurity threats, with a focus on detecting malware beaconing. This attack involves hidden communication between infected systems and command-and-control servers. IQM's 20-qubit quantum processor is used to test quantum methods in real enterprise settings.

The project is structured into several phases: defining the problem, building a quantum proof of concept, testing scalability on hardware, and identifying performance bottlenecks. When I joined, the classical machine learning part was already done, and data exploration was ongoing. My main role during this six-month internship was to focus on quantum algorithm development.

2 Introduction

2.1. High overview

Malware beaconing is a stealth technique where infected machines send periodic signals to a command-and-control server. It allows attackers to exfiltrate data and maintain long-term access while avoiding detection. Beaconing is hard to detect because the traffic often looks normal, uses encryption, and adjusts the timing to remain hidden.

Large organizations with complex digital systems face a growing range of threats and must constantly adapt to stay secure. Missed detection can cause serious damage: financial losses, legal penalties, and trust issues. Traditional rule-based systems don't scale and trigger too many false positives. They slow down analysts and waste time.

This use case targets that problem, making beaconing detection scalable and accurate using quantum-enhanced machine learning.

2.2. Dataset

One of the core challenges in this project was the dataset itself. Throughout the six-month internship, the data evolved significantly due to ongoing feature engineering. The dataset, initially provided by the client, consists of real network communication logs associated with malware beaconing.

Malware beaconing occurs when malware periodically connects to an external server to receive instructions from an attacker. These connections typically use standard protocols like DNS, SSH, SMTP, or HTTPS, making them hard to detect [15]. To avoid triggering alarms, attackers introduce jitter, random variations in timing. Despite these evasion techniques, beaconing patterns leave statistical traces in timing and packet sizes, which we can model.

The original dataset contained 721,899 normal events and only 2,523 attack events. All attack logs originate from the same IP address and headed to the same destination IP address, effectively representing a single attack scenario. This makes the dataset highly imbalanced. Given this, we used an unsupervised learning setup:

models were trained only on normal traffic. At inference, points that deviate from the learned distribution are flagged as anomalies.

The raw logs are based on the TCP protocol for establishing connections. TCP uses a three-way handshake:

1. The client sends a `SYN` packet to the server.
2. The server replies with a `SYN-ACK` packet.
3. The client responds with an `ACK` packet.

When the `SYN` flag is set to 1, it marks the start of a new connection. In our dataset, many attack logs have this flag set, suggesting repeated new connections from the attacker's IP. A correlation analysis between the `SYN` flag and the attack label Figure 2.1 shows a strong dependency. This can cause data leakage: the model may learn to classify based on the SYN flag rather than underlying malicious behavior, leading to inflated performance metrics that will not generalize to real-world traffic.

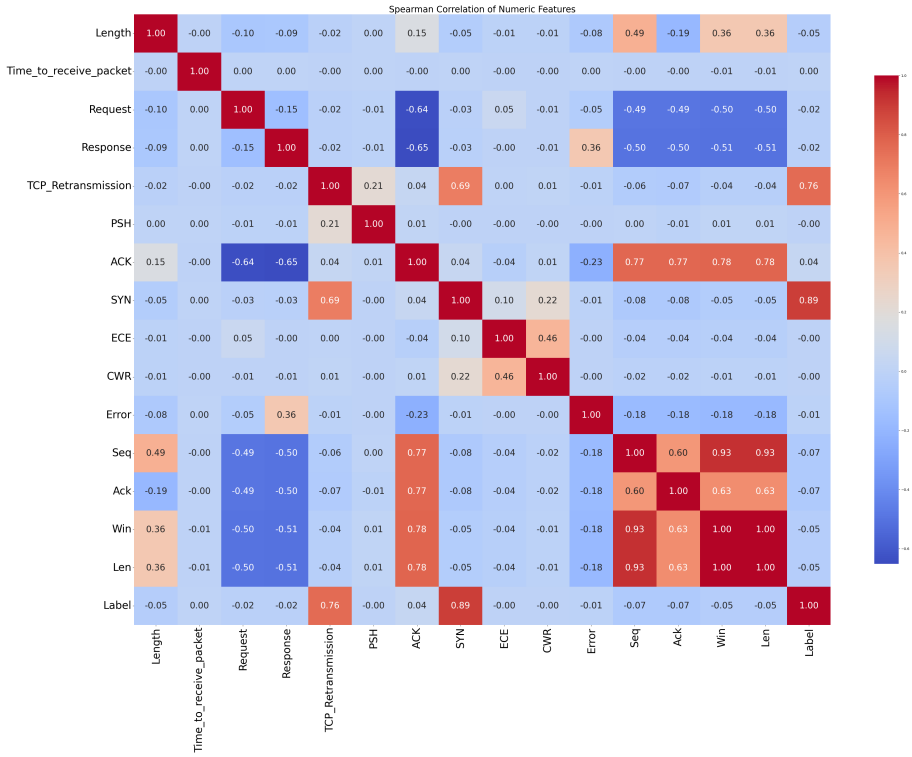
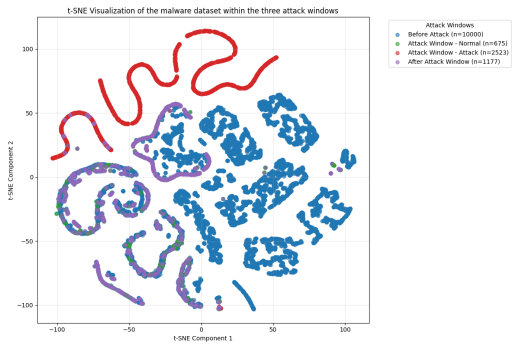


Figure 2.1.: Spearman correlation matrix of numerical features. This correlation matrix shows strong correlations between SYN and TCP flag features with the target label, indicating potential data leakage issues.

The new feature engineering approach focuses on connection timing. For each unique source-destination IP pair, we compute:

- Mean time interval between new connection starts
- Standard deviation of time intervals
- Number of new connections
- Mean packet size
- Standard deviation of packet size
- Number of unique source ports
- Number of unique destination ports



(a) t-SNE visualization showing clear separation between attack and normal traffic.

(b) Correlation matrix showing that the engineered feature has low correlation with the label.

Figure 2.2.: (a) Separation between attack and normal traffic in the feature space. (b) Low correlation between the engineered feature and the label, indicating it does not directly encode the target.

As shown in Figure 2.2, the attack samples are well separated in the t-SNE[1] projection, and the engineered feature shows low correlation with the label.

After applying this transformation, the dataset was reduced to 119 non-attack events and one attack event, giving us a compact, engineered representation for anomaly detection. However, having just one attack in the test set was insufficient to evaluate the performance of the model.

We needed to look into open-source datasets and extract malware beaconing traces to complement our original data. After extensive exploration, we constructed a new dataset by combining multiple sources:

- Network traffic from Internet of Things (IoT) devices

[1]t-SNE (t-Distributed Stochastic Neighbor Embedding) is a dimensionality reduction technique that visualizes high-dimensional data in 2D/3D by preserving local neighborhood structures.

- Normal traffic logs
- Malware dataset provided by the client
- CTU-13 Dataset
- Network traffic from work laptop

The final dataset includes 6,459 training logs and 1,741 testing logs, containing 126 labeled attacks.

The final feature engineering was based on defining network flows using a four-tuple:

- Source IP address
- Destination IP address
- Protocol
- Port number of external IP

We found that using only source and destination IP addresses to define flows was insufficient to capture periodicity in traffic, which is crucial to detect beaconing behavior. This insight informed our final flow construction and improved the relevance of the extracted features.

2.3. One-Class Support Vector Machine (OC-SVM)

The One-Class Support Vector Machine (OC-SVM) is an unsupervised model for anomaly detection [32]. Its objective is to identify data points that deviate from normal behavior. Unlike standard SVMs used in supervised classification, OC-SVM is trained only on a single class of data. It learns a decision boundary in the feature space that encloses the majority of training points, treating those that fall outside as anomalies.
The method relies on kernel functions to compute similarities between data points. This makes OC-SVM directly compatible with quantum kernel methods, where the kernel matrix can be provided as input to the model without modification.
In this work, we use the `OneClassSVM` implementation in `scikit-learn`. Hyperparameters are optimized using an unsupervised strategy inspired by the work of Dai et al. [33], which avoids reliance on labeled data.

2.4. State of the Art in Quantum Kernel Methods

Quantum kernel methods (QKMs) have become a central research direction within quantum machine learning (QML) [17, 18]. A key theoretical result shows that supervised QML models can be formulated as kernel methods [1, 19], embedding QKMs within the well-established framework of classical kernel theory [20].

The core idea is to define kernel functions that map classical inputs into a higher-dimensional Hilbert space, where data can become more easily separable. Quantum computers implement such mappings through parameterized quantum circuits (PQCs), often referred to as data encoding circuits [1, 2, 19, 21].
Two main approaches for evaluating quantum kernels have emerged:

- **Fidelity quantum kernels (FQKs)** [2, 19, 22, 24], which rely on overlaps between encoded quantum states.
- **Projected quantum kernels (PQKs)** [12, 25, 26], which apply a classical projection step to mitigate concentration effects in large Hilbert spaces.

Recent work has also proposed generalized frameworks that unify FQKs and linear PQKs [25].
Beyond foundational studies, QKMs have been tested in diverse application domains: financial classification [27], insurance modeling with QSVMs [28], high-dimensional cosmology data [29], and even solving differential equations [30]. Automated search for encoding architectures has also been investigated [31].
While these advances highlight the potential of QKMs, open challenges remain. Fidelity-based kernels suffer from exponential concentration effects as system size grows [10], and PQKs introduce a classical post-processing step whose practical impact requires careful benchmarking. Addressing these issues is central to the experimental study presented in this work.
It is also important to note that almost all prior work has focused on **supervised classification tasks**. In contrast, the present study investigates QKMs in an **unsupervised anomaly detection setting**, which has received little attention so far and poses different challenges for kernel evaluation and benchmarking.

3 Foundations of Quantum Kernel Methods

3.1. Data Encoding

To apply kernel methods in the quantum setting, we first define how classical data are encoded into quantum states. This step determines the structure of the feature space and shapes the resulting kernel. Throughout this work, the terms data encoding and feature map are used interchangeably.

Suppose we use a quantum circuit $U(x)$ that takes input $x \in \mathcal{X}$ from a classical dataset. The structure of $U(x)$ changes with x. For real-valued vectors, $U(x)$ might apply parameterized rotations where each feature in x sets the rotation angle on a specific qubit. Acting on an initial state $|\psi\rangle$, the circuit produces a new quantum state

$$|\phi(x)\rangle = U(x)|\psi\rangle$$

that reflects the input data. This defines a feature map from the classical input space $\mathcal{X}$ to the Hilbert space of quantum states.

Definition 1 (Data-encoding / feature map). Given an n-qubit quantum system with states $|\psi\rangle$, let $\mathcal{F}$ be the space of complex-valued $2^n \times 2^n$ matrices equipped with the Hilbert–Schmidt inner product

$$\langle \rho, \sigma \rangle_{\mathcal{F}} = \mathrm{tr}(\rho^\dagger \sigma), \quad \rho, \sigma \in \mathcal{F}.$$

The data encoding feature map is the transformation

$$\phi : \mathcal{X} \to \mathcal{F}, \quad \phi(x) = |\phi(x)\rangle\langle\phi(x)| = \rho(x),$$

implemented by a data encoding quantum circuit $U(x)$ [1].

The purpose of data encoding is to transform classical inputs into quantum states and represent them in a space where similarity can be measured. Ideally, the inner product in Hilbert space reflects a similarity metric over the original data. Achieving this may require a feature space dimension on the order of the dataset size. Once this is achieved, comparison and classification can use standard inner products, allowing linear models in the expanded space [2]. The choice of feature map directly determines the kernel function.

Examples from the Literature

Example 1: Basis encoding

[3,4] This method maps binary input strings directly to computational basis states. For instance, the bit string $x = 01001$ becomes the 5-qubit state $|01001\rangle$. Formally,

$$U_\phi : x \in \{0,1\}^n \mapsto |i\rangle,$$

where $|i\rangle$ denotes the standard basis vector indexed by the integer representation of x. This creates an orthonormal feature space, with kernel

$$\kappa(x, x') = \langle i|j\rangle = \delta_{ij},$$

returning 1 if inputs are identical and 0 otherwise.

Example 2: Amplitude encoding

[5] [6]

Here, a real-valued input vector $x = (x_0, \ldots, x_{N-1})^\top \in \mathbb{R}^N$, with $N = 2^n$, is encoded into the amplitudes of an n-qubit quantum state:

$$U_\phi : x \in \mathbb{R}^N \mapsto |\psi_x\rangle = \sum_{i=0}^{N-1} x_i \, |i\rangle \, .$$

This representation leads to a linear kernel:

$$\kappa(x, x') = \langle \psi_x | \psi_{x'} \rangle = x^\top x'.$$

Feature maps used in this project

Example 1 : Z feature map

ZFeatureMap is a first-order Pauli-based quantum feature map that applies parameterized Z-rotations to each qubit individually. It uses only Pauli-Z terms and contains no entangling gates. The state at the end of the circuit can be written as a tensor product of single-qubit basis states, meaning the circuit is fully separable. Each qubit encodes a single classical feature through a sequence of Hadamard and phase gates.
This encoding circuit is taken from Qiskit [6]. An example is given in Figure 3.1.

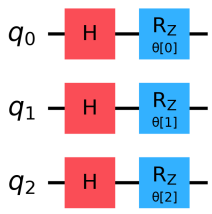


Figure 3.1.: Visualization of a single layer from the Qiskit Z feature map.

Example 2 : CNot feature map

This circuit modifies Qiskit's **ZFeatureMap** by inserting CNOT gates between rotation layers. The **CNOT feature map** introduces entanglement between qubits. It encodes classical data through fixed-parameter rotations. See Figure 3.2 for an example.

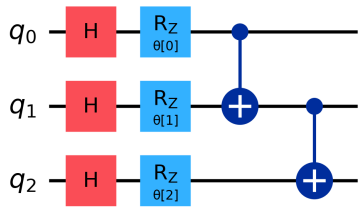


Figure 3.2.: Visualization of a single layer from cnot feature map.

Example 3 : IQP (Instantaneous Quantum Polynomial-time) Like circuit

IQP-like circuits use gates that all commute and are diagonal in the Z-basis. This makes them simpler to simulate on classical computers under certain conditions. The following Figure 3.3 illustrates a simple implementation of it.

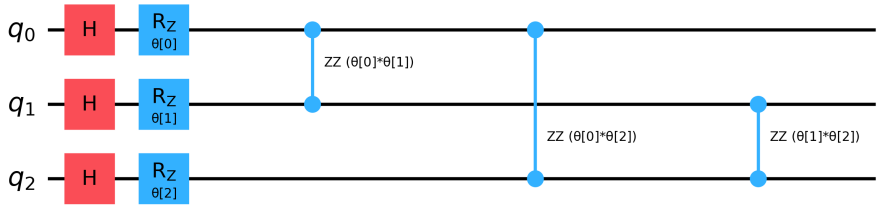


Figure 3.3.: Visualization of a single layer from IQP-like circuit.

Example 4 : Hamiltonian evolution like circuit

This feature map has been developed for many-body problems. It requires $n+1$ qubits to embed n features and is defined as follows:

$$|x_i\rangle = \left(\prod_{j=1}^{n} \exp\left(-i\frac{t}{T}x_{ij}H_j^{XYZ}\right)\right)^T \bigotimes_{j=1}^{n+1} |\psi_j\rangle, \tag{3.1.1}$$

where

$$H_j^{XYZ} = X_jX_{j+1} + Y_jY_{j+1} + Z_jZ_{j+1},$$

with X_j, Y_j, and Z_j being the Pauli operators acting on the j-th qubit. The state $|\psi_j\rangle$ is a Haar-random state, and t and T are hyperparameters representing the total evolution time and the number of Trotter steps, respectively [12] [14]. See Figure 3.4 for an example of the implementation.

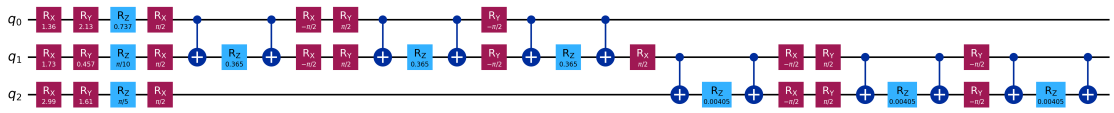


Figure 3.4.: One layer implementation of the Hamiltonian evolution like circuit.

3.2. Kernels in quantum feature spaces

One direct way to define a kernel is by evaluating the similarity between two data points in the feature space. This similarity is quantified through the inner product of their encoded quantum states. Once the Hilbert space is specified by the feature map $\phi : \mathcal{X} \to \mathcal{F}$, the associated kernel is defined as

$$\kappa(x, x') := \langle \phi(x), \phi(x') \rangle_{\mathcal{F}}, \tag{3.2.1}$$

where $\langle \cdot, \cdot \rangle_{\mathcal{F}}$ denotes the inner product in the feature space.
This inner product corresponds to the fidelity between quantum states $|\phi\rangle$ and $|\psi\rangle$:

$$F(|\phi\rangle, |\psi\rangle) = |\langle \phi | \psi \rangle|^2.$$

which naturally reflects the similarity between two encoded states.
However, while fidelity kernels are conceptually straightforward, their computation poses serious challenges when applied to large datasets. The kernel matrix grows quadratically with the number of data points, and each entry requires evaluating a quantum state overlap, which is resource-intensive on near-term devices.

4 Ensemble Fidelity Kernel

Kernel methods fall into two main categories: fidelity-based kernels and projected quantum kernels. This chapter focuses on the former, where kernels are computed as squared inner products between quantum states, directly reflecting state similarity in Hilbert space.

To address the scalability problem of fidelity kernels, recent work has introduced more computationally efficient strategies. Among them, the ensemble fidelity kernel stands out. Its principle is based on the framework of Variable Sampling Ensembles using Inversion Test Kernels mentioned by Kölle et al [16].

Mathematical frame work

Formally, the method defines an ensemble of

$$c = \left\lfloor \frac{n}{100} \right\rfloor \tag{4.0.1}$$

components, where each component is trained on a randomly subsampled dataset D_i of size $n_i \in [n_{\min}, n_{\max}] = [50, 100]$. Each D_i is used to train a quantum one-class SVM with kernel function

$$k(x, x') = \left| \langle 0 | U_\varphi^\dagger(x') U_\varphi(x) | 0 \rangle \right|^2, \tag{4.0.2}$$

where U_φ denotes the unitary feature map. The ensemble prediction is then

$$\text{Score}_{\text{ensemble}}(x_{\text{new}}) = \text{Agg}\left(\{\text{Score}_i(x_{\text{new}})\}_{i=1}^{c} \right), \tag{4.0.3}$$

with aggregation operator $\text{Agg}(\cdot)$ defined as averaging (variance reduction) or maximum (bias reduction).

The training complexity scales as

$$O\left(c \times \left(\frac{n_{\min} + n_{\max}}{2} \right)^2 \right) = O\left(\left\lfloor \frac{n}{100} \right\rfloor \times 75^2 \right), \tag{4.0.4}$$

achieving linear dependence on dataset size n. Testing complexity is

$$O\left(c \times \frac{n_{\min} + n_{\max}}{2} \times n_{\text{test}} \right), \tag{4.0.5}$$

maintaining linear dependence on both the ensemble size and test set size. The final classification is obtained via

$$\hat{y} = \operatorname{sgn}\big(\text{Score}_{\text{ensemble}}(x_{\text{new}})\big). \tag{4.0.6}$$

This method was implemented by the team as part of the benchmarking pipeline. Although I did not directly contribute to the development of the ensemble fidelity kernel, its results serve as a reference point to compare with the projected quantum kernels, which form the central focus of my own contribution.

Numerical Results

To illustrate the behavior of the ensemble fidelity kernel, I tested the method on a subset of an open-source intrusion detection dataset. The training set consisted of 1000 data points with 10 features, encoded into a 10-qubit system. The feature map was implemented as a two-layer CNOT-based circuit. The ensemble size was fixed to $c = 10$.
The test set contained 100 normal samples and 457 attack samples. Each ensemble component was trained on a randomly subsampled set of size $n_i \in [50, 100]$, following the procedure described previously.
The evaluation is summarized in two parts:

	Pred. Normal	Pred. Attack
True Normal	78	22
True Attack	265	192

Table 4.1.: Confusion matrix for the ensemble fidelity kernel.

Metric	Score
Accuracy	0.485
Precision	0.897
Recall	0.420
F1-score	0.572

Table 4.2.: Performance metrics of the ensemble fidelity kernel.

The F1-score is the most relevant result here. It reaches 0.572. Precision is high (0.897) but recall is low (0.420), meaning many attacks are missed. This imbalance limits the method's usefulness. We now turn to randomized measurements as a more efficient way to approximate fidelity kernels and possibly improve detection.

5 Randomized Measurement

A major challenge in applying fidelity kernels to real data is scalability. The kernel matrix grows quadratically with the dataset size: for L datapoints, we need L^2 overlaps. Each overlap requires simulating or measuring quantum states, which becomes prohibitively expensive on near-term devices. This makes the direct use of fidelity kernels impractical for larger datasets.

To address this limitation, we consider *randomized measurement protocols*. Rather than computing overlaps explicitly, these methods estimate kernel entries by sampling measurement outcomes in randomly selected bases. This reduces computational overhead while retaining the kernel structure.

5.1. Mathematical framework

One of the main goals of kernel methods is to assess how similar a data point x_i is to another x_j. In the quantum setting, this translates into comparing their corresponding quantum states. These states are generally unknown and live in high-dimensional Hilbert spaces, making direct comparisons difficult.
As emphasized by Elben et al. [8], the challenge lies in estimating overlaps between quantum states that cannot be directly accessed and may vary across experimental realizations. This difficulty is central to quantum kernel methods, where similarities must be evaluated without explicit knowledge of the states.
To address this, we use randomized measurement protocols as described in [8] and [9]. They provide estimators for state overlaps based only on sampled measurement outcomes, avoiding full state tomography. This section presents the mathematical framework for these estimators.

Once the density matrix is prepared using the chosen encoding, the kernel estimation follows a randomized measurement protocol. We apply M random unitaries, each built from independently sampled single-qubit rotations drawn from the Haar measure over SU(2).
Each transformation is of the form $V^{(m)} = \bigotimes_{k=1}^{N} V_k^{(m)}$, where N is the number of qubits and $V_k^{(m)}$ is a single qubit unitary acting on qubit k, sampled independently for each qubit and each measurement round m.

We then rotate the quantum state $\rho(\theta_i)$ into the new basis:

$$\rho_i^{(m)} = V^{(m)} \rho_i V^{(m)\dagger}.$$

The rotated state is measured s times in the computational basis to estimate the probability $P_i^{(m)}(v)$ of observing the bitstring $v \in \{0,1\}^N$.
This procedure is repeated for all M random transformations and for all L quantum states, where L is the size of the dataset. The kernel value between two encoded states ρ_i and ρ_j is then approximated by:

$$K_b(\theta_i, \theta_j) = \frac{2^N}{M} \sum_{m=1}^{M} \sum_{v,v'} (-2)^{-D(v,v')} P_i^{(m)}(v)\, P_j^{(m)}(v') \tag{5.1.1}$$

where $D(v, v')$ is the Hamming distance between the computational basis strings v and v'.

Algorithm 1 Estimate Fidelity Kernel from Randomized Measurements

Input: Encoded quantum circuits $\{\rho_i\}_{i=1}^{L}$, number of random transformations M, number of shots s
Output: Estimated Gram matrix K
Initialize $K \leftarrow 0$
Initialize empty list *x_logs_statistics*
for each encoded circuit ρ_i **do**
    Generate M random unitaries and measure the circuit s times for each unitary
    Store the resulting measurement statistics in *x_logs_statistics*
end for
for each pair (i, j) of data points **do**
    Retrieve measurement statistics $\log_1, \log_2$ from *x_logs_statistics*
    Compute fidelity:

$$F(\rho_i, \rho_j) = \sum_{s_1,s_2} (-2)^{-d(s_1,s_2)} \cdot \mathrm{avg}_U \left[p_i(s_1|U)\, p_j(s_2|U) \right]$$

    where $d(s_1, s_2)$ is the Hamming distance and the average is taken over all M random unitaries U
    Assign $K_{ij} \leftarrow F(\rho_i, \rho_j)$
end for
return K

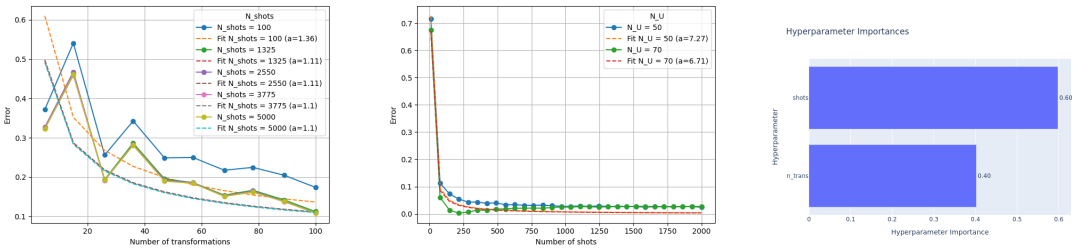
To evaluate all entries of the kernel matrix, a total of $N_R = s\,M\,L$ measurements are required. In practice, estimating the fidelity kernel from randomized measurements introduces a statistical error. This error comes from two sources: the finite number s of projective measurements per random unitary, and the limited number M of sampled unitaries. Both affect the accuracy of the overlap and purity estimates computed using Equation 5.1.1. It has been proven in [8] that the error ΔK of

estimating a single kernel entry scales as $\Delta K \propto \frac{1}{s\sqrt{M}}$.
The statistical error can be quantified as the absolute deviation between the estimated purity and its ideal value for a perfect quantum state, where fidelity equals one. Purity is obtained from the diagonal elements of the Gram matrix, which measure the fidelity of each reconstructed quantum state with itself. The numerical results provided in the next section illustrate how this error scales with the number of shots s and the number of random transformations M.

5.2. Numerical results

Error Scaling

Early in the internship, I obtained results suggesting that the estimation error in the kernel entries roughly followed the expected scaling law: $\Delta K \propto \frac{1}{s\sqrt{M}}$, where s is the number of projective measurements per unitary and M is the number of sampled unitaries. These results were not perfectly aligned with the theory but were reasonably close.
To explore this further, I generated two plots illustrated in Figure 5.1 to study the effect of these parameters individually. These plots were generated using the malware beaconing dataset and a one-layer CNOT feature map.



(a) Statistical error as a function of the number of random transformations

(b) Statistical error as a function of the number of shots

(c) Hyperparameter importance scores from Optuna optimization

Figure 5.1.: Statistical error scaling with number of random transformations (left) and shots (center), and corresponding hyperparameter importance scores from Optuna optimization using statistical error as objective (right).

- **Figure 5.1(a)**: Error decreases as the number of transformations increases across all sample sizes. Smaller sample sizes ($N_{\text{shots}} = 100$) show higher error and greater variance. Larger sample sizes converge to similar error rates ($\sim 0.1-0.12$) by 100 transformations.

- **Figure 5.1(b)**: Fixing the number of transformations and varying the number of shots reveals rapid error decay. Both $N_U = 50$ and $N_U = 70$ show sharp drops from ~ 0.7 to near-zero error by 200 shots.

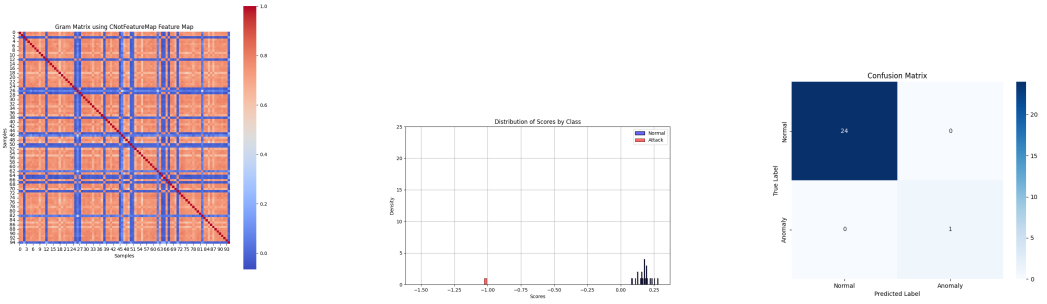
- After $\sim$ 200 shots, additional measurements yield minimal gains. Both settings converge to errors below 0.02.

- **Figure 5.1(c)** In addition, a 300-trial Optuna[2] hyperparameter search indicated that the number of measurement shots contributed approximately 60% to the variance in the objective function, while the number of unitaries accounted for the remaining 40%. This aligns with the empirical observation that increasing shots has a stronger effect on error reduction than increasing the number of unitaries.

At the time of writing, I attempted to reproduce these results to include concrete plots and data. However, the outputs were inconsistent with both the earlier observations and the theoretical expectations. The cause of this discrepancy is still unclear.
Due to time limitations, and since the newer results did not offer reliable insight, I decided not to include them.
Future work could revisit this analysis under more controlled experimental conditions to better confirm or refute the scaling law.

Evaluation on Malware Dataset

The malware dataset, after feature engineering, was reduced to 95 training points and 25 testing points, including one attack. It includes 7 features, with each feature encoded in one qubit, making it a 7-qubit system. This setup was computationally lightweight and easier to manage than the final network dataset.



(a) Kernel Matrix Visualization (b) Test Score Distribution (c) Confusion Matrix

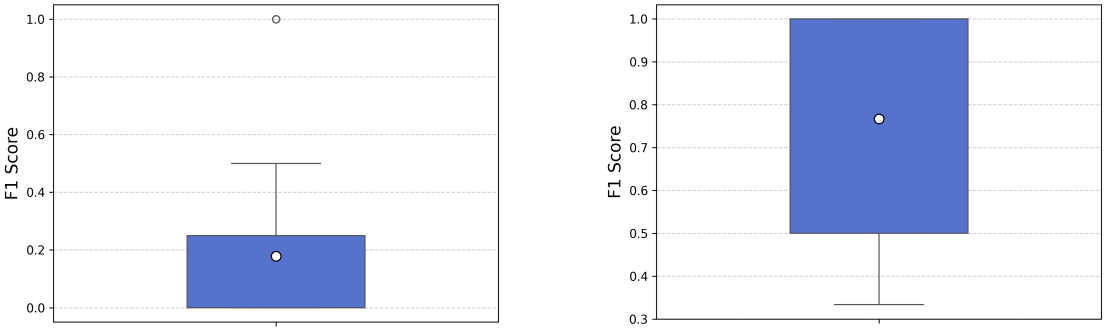
Figure 5.2.: Randomized measurement performance analysis using one layer of CNOT feature map: Gram matrix visualization (left), distribution of test scores (center), and classification confusion matrix (right). Results obtained using 30 random transformations and 400 shots.

[2]Optuna is an open-source hyperparameter optimization framework.

- **Figure 5.2(a)**: The Gram matrix shows a clear block-diagonal structure. This suggests the CNOT feature map might capture natural clusters in the normal training data.
- **Figure 5.2(b)**: The test decision scores show that the attack has a high negative value. It's well separated from the normal test points. This separation is confirmed by the confusion matrix **Figure 5.2(c)**. The model correctly flags the attack while avoiding false positives.

Although the results on the malware dataset were promising, this setup showed high variance in reproducibility. This is likely due to the random unitaries used to rotate the states. The next paragraph illustrates this instability.

F1 Score Variability Across Experimental Repetitions



(a) 30 transformations, 400 shots, 1 layer CNOT (29 experiments)

(b) 10 transformations, 180 shots, 1 layer CNOT (9 experiments)

Figure 5.3.: F1 score distributions for quantum one-class SVM with randomized measurement comparing high-resource and low-resource CNOT feature map configurations.

- **Figure 5.3(a)**: Despite using more quantum resources (30 transformations, 400 shots), this configuration exhibits extreme performance instability. The F1 scores span the entire range from 0 to 1, with most runs clustered around poor performance (mean 0.18), highlighting the unreliability of this high-resource approach.
- **Figure 5.3(b)**: Counterintuitively, the low-resource setup (10 transformations, 180 shots) achieves both superior performance and remarkable consistency. The tight distribution around F1 0.77 demonstrates that fewer quantum operations can lead to more stable and reliable results.

In our numerical experiments, we observed that the statistical error decreases with the number of random bases n_{rand} and the number of measurement shots n_{shots}, but then reaches a plateau. This saturation occurs before any error mitigation is applied and reflects the exponential concentration effect reported in [10]. It highlights

a fundamental challenge for scaling randomized measurement protocols to larger quantum systems.
For the performance evaluation experiments, we included the error mitigation technique introduced in [9]. Depolarizing noise transforms pure states into mixed states and lowers kernel values. By reusing the same randomized measurement data, one can estimate the purity $\mathrm{Tr}(\rho(\theta_i)^2)$, determine the depolarization probability p_i, and mitigate the effect of noise. The corrected kernel is then approximated as

$$K_m(\theta_i, \theta_j) \approx \frac{K_b(\theta_i, \theta_j)}{\mathrm{Tr}(\rho(\theta_i)^2)\,\mathrm{Tr}(\rho(\theta_j)^2)}. \tag{5.2.1}$$

This adjustment improves the quality of the kernel estimates under realistic device conditions. Still, the scalability concerns remain and motivate alternative approaches. In the next chapter, we turn to projected quantum kernels, which approximate the feature space without relying on fidelity estimation.

6 Projected quantum kernels

In the previous chapter, we showed how randomized measurements can estimate fidelities between quantum states and provide a practical way to build kernel matrices. Despite their scalability and the use of error mitigation techniques, the numerical results revealed plateaus in the statistical error and highlighted the exponential concentration effect [10]. This effect becomes more severe as the system size grows, and ultimately limits the usefulness of fidelity-based kernels.

Embedding classical data into quantum states and evaluating their inner products is a central idea in quantum machine learning. The goal is to map inputs from $\mathbb{R}^D$ into a higher-dimensional Hilbert space $\mathcal{H}$ where nonlinear patterns become separable. Yet, as the number of qubits increases, kernel values based on fidelities such as $\mathrm{Tr}(\rho(x_i)\rho(x_j))$ collapse toward a constant [10]. When this happens, the kernel matrix approaches the identity, and the model loses its discriminative power.

To address this limitation, we turn to *projected quantum kernels (PQKs)*. By applying a classical projection to the quantum kernel matrix, PQKs reduce its effective rank and recover meaningful structure that standard fidelity-based kernels fail to capture.

6.1. Mathematical framework

Projected quantum kernels are named after the core idea behind them: projecting quantum states onto approximate classical representations using reduced physical observables or classical shadows [11]. One simple projected quantum kernel is based on the one-particle reduced density matrix (1-RDM). For each encoded quantum state $\rho(x_i)$, we compute:

$$\rho_k(x_i) = \mathrm{Tr}_{j\neq k}[\rho(x_i)]$$

Then define the Gaussian kernel as:

$$k_{\mathrm{PQ}}(x_i, x_j) = \exp\left(-\gamma \sum_k \|\rho_k(x_i) - \rho_k(x_j)\|_F^2\right)$$

This maps the input into a feature space formed by the 1-RDMs. Based on this, different kernel forms have been proposed [12]:

1. Linear kernel on 1-RDMs

$$Q_{1l}(x_i, x_j) = \sum_k \mathrm{Tr}[\rho_k(x_i)\rho_k(x_j)]$$

2. Gaussian kernel on 1-RDMs

$$Q_{1g}(x_i, x_j) = \exp\left(-\gamma \sum_k \|\rho_k(x_i) - \rho_k(x_j)\|_F^2\right)$$

3. Linear kernel using k-RDMs
We generalize to a subsystem K of size k:

$$Q_{kl}(x_i, x_j) = \sum_K \mathrm{Tr}[\rho_K(x_i)\rho_K(x_j)]$$

where $\rho_K(x) = \mathrm{Tr}_{n\notin K}[\rho(x)]$

Our implementation
In practice, we do not compute the kernel as an inner product. Instead, we extract classical features from reduced quantum states, such as k-RDMs, and use them as inputs to classical kernels like the RBF or sigmoid kernel. This avoids estimating quantum fidelities directly, which becomes unreliable in high-dimensional Hilbert spaces.
The kernels used are:
RBF kernel:

$$k_{\mathrm{RBF}}(x_i, x_j) = \exp\left(-\gamma \|f(x_i) - f(x_j)\|^2\right)$$

where:

- $\gamma > 0$ is a hyperparameter that controls the kernel width
- $f(x_i)$ denotes the classical features obtained from the projected quantum state of x_i

Sigmoid kernel:

$$k_{\mathrm{sigmoid}}(x_i, x_j) = \tanh\left(\alpha\langle f(x_i), f(x_j)\rangle + c\right)$$

where:

- $\alpha > 0$ controls the slope of the activation
- $c \in \mathbb{R}$ is a bias term that shifts the decision boundary
- $\langle f(x_i), f(x_j)\rangle$ denotes the standard Euclidean inner product between feature vectors:

$$\langle f(x_i), f(x_j)\rangle = \sum_k f_k(x_i) f_k(x_j)$$

Algorithm 2 Compute k-RDM Features from Quantum Circuits

Input: Circuits $\{C_i\}_{i=1}^{N}$, number of qubits n, integer $k \leq n$
Output: Feature tensor F of shape $(N, \binom{n}{k}, 4^k - 1)$

if $k > n$ **then raise** error
end if
Define Pauli operators $P = \{I, X, Y, Z\}$
Generate all non-identity Pauli strings of length k
Generate all subsets of qubit indices of size k
Initialize feature tensor F with zeros
for $i = 1$ to N **do**
    for each subset S **do**
        for each Pauli string p **do**
            Construct n-qubit Pauli operator O by placing p on S,
                identities elsewhere
            Compute expectation value e of O on circuit C_i
            Store $F[i, S, p] \leftarrow e$
        end for
    end for
end for
return F

6.2. Numerical Results

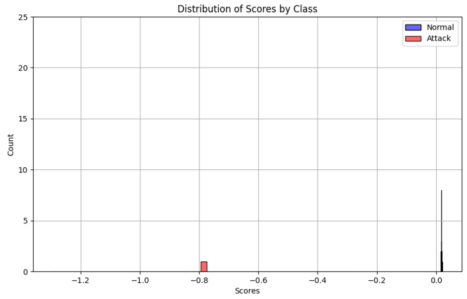
Overall, this approach was more computationally efficient than both the fidelity kernel and the randomized measurements. It also showed greater stability across experimental runs.

In this section, we first evaluate the projected quantum kernel on the malware dataset provided by the client.
We then assess its performance on the final open-source network dataset, comparing it to a classical one-class support vector machine. For experiments with varying dataset sizes, we reduced the number of original features using a clustering technique described in the annex Figure A.1.5. We then applied principal component analysis (PCA) to obtain 10 features. These experiments were tested on the same testing dataset, which contains 100 non-attack samples and 457 attack samples with the same feature selection as the training dataset. We also investigate the effect of different values of k in the k-reduced density matrix.
Performance is measured using the F1-score, recall, precision, and accuracy.

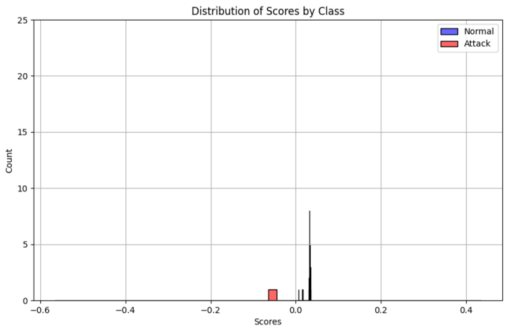
Evaluation on Malware Dataset

As mentioned earlier, the malware dataset is easy to handle but insufficient for generalization, as the test set contains only one attack.

It has seven features, which is optimal for the number of qubits, but the system still has several hyperparameters that can influence results even in this simple case. We therefore focus here on comparing the CNOT and Hamiltonian feature maps using both RBF and sigmoid kernels.



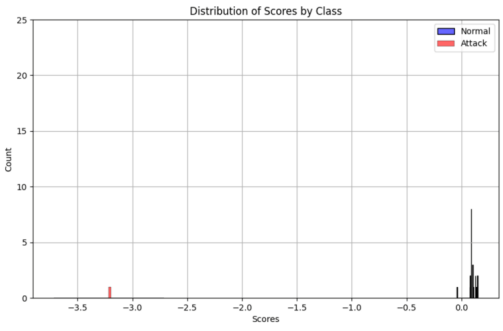
(a) Test score distribution with the Hamiltonian evolution feature map and sigmoid kernel.



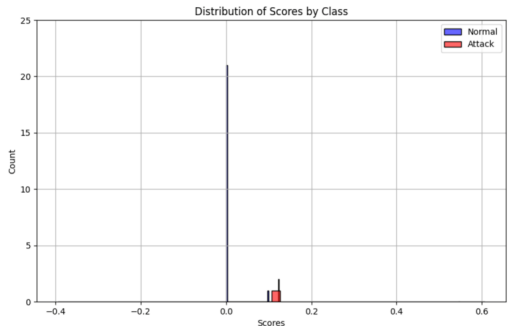
(b) Test score distribution with the CNOT feature map and sigmoid kernel.

Figure 6.1.: Test score distribution comparison for projected quantum kernel with sigmoid kernel showing performance differences between Hamiltonian evolution feature map (left) and CNOT feature map (right).

Figure 6.1 shows that both feature maps, when combined with the sigmoid kernel applied to the reduced density matrix, achieved perfect results. However, the Hamiltonian-like circuit produced better data separation. In these experiments, k was fixed to 1, and the classical kernel hyperparameters in the sigmoid kernel (alpha α and constant coefficient c) were set to $\alpha = 1$ and $c = 0$, respectively.



(a) Test score distribution with the Hamiltonian evolution feature map and RBF kernel.



(b) Test score distribution with the CNOT feature map and RBF kernel.

Figure 6.2.: Test score distribution comparison for projected quantum kernel with RBF kernel showing performance differences between Hamiltonian evolution feature map (left) and CNOT feature map (right).

Switching to the RBF kernel in Figure 6.2 reduced performance. The CNOT + RBF combination gave the lowest scores, with overlapping decision regions. Hamiltonian-

like circuit preserved a clear boundary between attack and non-attack samples, achieving an F_1 score of 0.66. This indicates that the embedding produced by the Hamiltonian-like feature map remains partially separable under the RBF metric, even with the default $\gamma = 1$. These results highlight the sensitivity of the projected quantum kernel to the choice of classical kernel, and suggest that, for the malware dataset, the Hamiltonian-like mapping encodes features more resilient to kernel changes than the CNOT mapping.

Evaluation on Open-Source Network Dataset

In this section, we evaluate the projected quantum kernel on samples of the open-source dataset. Due to computational constraints, we did not process the full dataset. All experiments were conducted on the same test set, which contains all attack samples, as mentioned earlier.

Table 6.1.: Performance comparison for different training set sizes.

Data size	**PQKs**				**Classical**			
	F1	Recall	Precision	Acc.	F1	Recall	Precision	Acc.
200	0.2308	0.1313	0.9524	0.9524	0.7467	0.6936	0.8087	0.6140
1000	0.1193	0.0634	1	0.2316	0.2368	0.1422	0.7065	0.2477

From the analysis of Table 6.1, the experiments were conducted using a CNOT mapping with 2 layers to introduce entanglement, and the reduced density matrix order was fixed to $k = 1$. The classical kernel was set to an RBF kernel with the default $\gamma = 1$. Across all training set sizes, the PQKs showed significantly lower F1-scores compared to the classical model. For instance, at 200 samples, the PQKs reached $F1 = 0.2308$ while the classical model achieved $F1 = 0.7467$. Increasing the data size to 1000 further degraded the PQKs to $F1 = 0.1193$, whereas the classical model still maintained $F1 = 0.2368$. These results indicate that, under this configuration, the PQS failed to capture a balanced trade-off between precision and recall, leading to poor overall detection performance.

We next investigate the effect of varying k while keeping the same configuration: CNOT mapping with 2 layers, RBF kernel with $\gamma = 1$, and the training set fixed to the 200-sample case. This choice of dataset size is motivated by the high computational cost of increasing k. Specifically, the evaluation requires measuring over all $\binom{n}{k}$ possible subsystems, where n is the number of qubits (10 in our case, matching the number of features). For each subsystem, Pauli tomography involves 3^k distinct measurement settings. The number of shots is not considered here since we use the Qiskit statevector simulator for computing expectation values.

The results in Table Table 6.2 show that the F1 score remains within a narrow range from 0.1996 to 0.2308 across all tested k values. The best score is obtained for $k = 1$ (0.2308), followed by a sharp drop for $k = 2$ (0.2031) and $k = 3$ (0.1996), with a slight recovery for $k = 4$ (0.2136). This limited variation (~ 0.031 between best and worst) indicates that, for the CNOT mapping with two layers and an RBF kernel ($\gamma = 1$), the subsystem size has minimal effect on performance. The peak at

Table 6.2.: Performance comparison for varying k in the k-reduced density matrices

k	F1	Recall	Precision	Acc.
1	0.2308	0.1313	0.9524	0.2819
2	0.2031	0.1138	0.9454	0.2675
3	0.1996	0.1116	0.9444	0.2657
4	0.2136	0.1203	0.9482	0.2728

$k = 1$ suggests that higher-order correlations captured at larger k do not improve the anomaly detection, possibly due to insufficient entanglement from this configuration or the RBF kernel overshadowing quantum effects. Considering the computational cost, which grows as $\binom{10}{k} \times 3^k$, the marginal gains do not justify using $k > 1$, making $k = 1$ the most efficient choice for this setup.

Conclusion of Numerical Results

The experiments conducted across both the malware dataset and the open-source network dataset reveal several consistent trends. First, the choice of feature mapping significantly impacts performance, with the Hamiltonian-like mapping showing greater resilience to kernel changes compared to the CNOT mapping. Second, while the classical kernel type influences separability in the embedding space, the projected quantum kernel did not consistently outperform classical baselines across all configurations. In particular, the RBF kernel with default $\gamma = 1$ often led to limited gains, and in some cases masked potential quantum advantages.

The analysis of subsystem size k in the reduced density matrices showed minimal variation in F1 score, with $k = 1$ consistently providing the best trade-off between computational cost and classification performance. Increasing k failed to exploit higher-order correlations effectively, likely due to the limited entanglement generated in the tested configurations.

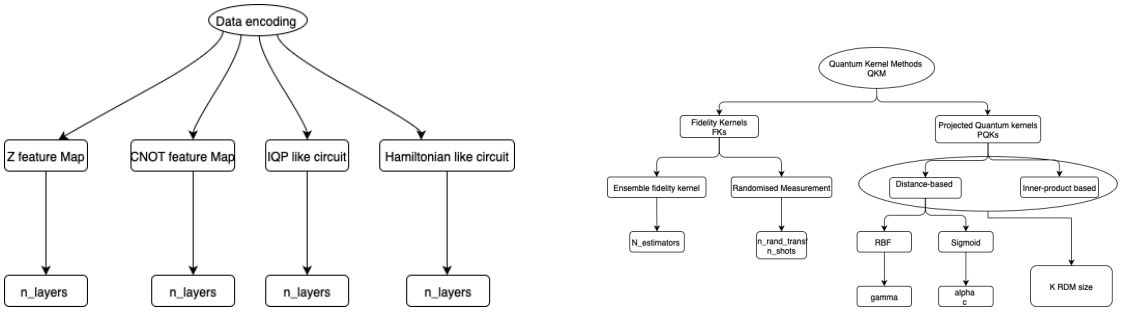
Overall, the numerical results suggest that the projected quantum approach is sensitive to both feature mapping and kernel choice, while subsystem size plays a secondary role under the current setup. These observations motivate a deeper exploration of hyperparameters, such as kernel parameters, circuit depth, and entanglement structure and finally the classical hyperparameters of the selected kernel, which will be the focus of the next chapter.

7 Benchmarking Results

This chapter benchmarks the impact of quantum hyperparameters on the performance of selected quantum kernel methods for anomaly detection. The focus is on the two approaches that are computationally viable for the datasets and objectives considered:

- **Ensemble Fidelity Kernel (EFK)**: a fidelity-based method that, due to the quadratic computation cost of direct fidelity estimation, combines multiple quantum fidelity measurements from different randomly subsampled and dimensionally-reduced data subsets to create a robust, scalable kernel function for anomaly detection with linear time complexity.
- **Projected Quantum Kernel (PQK)**: a kernel computed from reduced density matrices. While PQK supports both distance-based and inner-product-based formulations, the inner-product-based variant is excluded here, as the literature reports [14] that the distance-based approach consistently outperforms it.

The *Randomised Measurement* variant is excluded from this study. Its measurement requirements scale poorly with the number of features, and the number of qubits grows rapidly, making it impractical for the anomaly detection setting targeted here.



(a) Quantum data encoding methods.

(b) Quantum kernel methods and their associated hyperparameters.

Figure 7.1.: Overview of quantum kernel methods: data encoding approaches and hyperparameter configuration.

The benchmarking will evaluate:

- How the *number of estimators* in the EFK impacts F_1 score for anomaly detection.
- How the kernel type (RBF, Sigmoid) and associated parameters (γ, α, C) influence PQK performance.
- The effect of the reduced density matrix size k on PQK computation.
- The **geometric difference** as a measure of closeness between quantum and classical kernels, providing insight into whether the quantum mapping offers a distinct embedding.

The methods and parameters are summarised in Figure 7.1. The subsequent sections present controlled experiments for each configuration, isolating the contribution of individual hyperparameters to overall performance.

7.1. Experimental setup

Models

We combine classical SVMs from the `scikit-learn` library with custom quantum kernels, specifically the Ensemble Fidelity Kernel (EFK) and the distance-based Projected Quantum Kernel (PQK). PQK features are evaluated using either an RBF or a sigmoid kernel, simulated on Qiskit's ideal statevector backend. Classical data points are first embedded using one of the encoding techniques described in section 3.1 before kernel computation.
We perform hyperparameter search for each model and feature map using `Optuna`, selecting the best parameters for kernel configuration and SVM tuning.
The hyperparameter search space included:

- **Feature map:** {Z feature map, CNOT feature map, Hamiltonian-like circuit, IQP-like circuit}, number of layers $\in [1, 5]$
- **EFK:** number of estimators $\in [1, 5]$
- **PQK_RBF:** $k \in [1, 3]$, $\gamma \in [10^{-3}, 10^{3}]$
- **PQK_Sigmoid:** $k \in [1, 5]$, $\alpha \in [-1, 1]$, $c_{\text{sigmoid}} \in [10^{-3}, 10^{3}]$

Metrics

The hyperparameter optimization in `Optuna` uses the F_1 score as the objective function, ensuring that all hyperparameters are selected to maximize this metric from the outset. The F_1 score is chosen as the primary evaluation metric for anomaly detection performance because it provides a balanced assessment of both precision and recall, which is crucial in imbalanced datasets where accuracy alone can be misleading due to the predominance of normal samples over anomalous ones.

In addition to empirical performance evaluation, we introduce the **geometric difference** [12] between quantum and classical kernels as a complementary metric to assess the model's generalization capabilities. This measure quantifies how distinct the quantum feature embedding is from classical approaches, providing insight into whether the quantum mapping offers a meaningfully different representation that could potentially enhance generalization beyond what classical methods achieve.

Geometric Difference (GD)

To assess whether quantum kernels provide a meaningfully distinct embedding from classical approaches, we employ the geometric difference metric introduced by Huang et al. [12]. This metric quantifies the separation between quantum and classical kernel representations, which is theoretically linked to the potential for quantum advantage in machine learning tasks.

The geometric difference is defined as:

$$g_{C\to Q} = ||\sqrt{K_Q}(K_C)^{-1}\sqrt{K_Q}||_\infty \tag{7.1.1}$$

where K_C and K_Q are the classical and quantum kernel matrices respectively, both normalized such that $\mathrm{Tr}(K_Q) = \mathrm{Tr}(K_C) = N$. The $||\cdot||_\infty$ denotes the matrix infinity norm.

This asymmetric measure captures how well the classical kernel can approximate the quantum kernel embedding. Higher GD values indicate greater separation between quantum and classical representations, suggesting that the quantum mapping encodes features that are not easily reproducible by classical methods. Conversely, low GD values suggest that classical models could potentially achieve similar performance with sufficient data, questioning the practical advantage of the quantum approach.
For our analysis, we compute the GD between the PQK and its classical counterpart to evaluate whether the quantum feature mapping provides distinctive embeddings that justify their computational overhead. We do not compute the GD for EFK since it does not utilize a fixed kernel structure.

Dataset

Due to time constraints at the end of the internship and computational limitations, despite having access to company resources on Flyte and AWS EC2 instances, we focused our analysis on a manageable subset of the data. The training dataset consists of 200 datapoints with 5 selected features, resulting in a 5-qubit quantum system that balances computational feasibility with meaningful quantum circuit depth.
For evaluation, we used a test set containing 100 non-attack samples and 457 attack samples, maintaining the same 5-feature selection as the training dataset. This configuration results in an imbalanced test set where attack samples significantly outnumber normal samples, which differs from typical anomaly detection scenarios where normal samples usually dominate. However, this imbalance reflects the

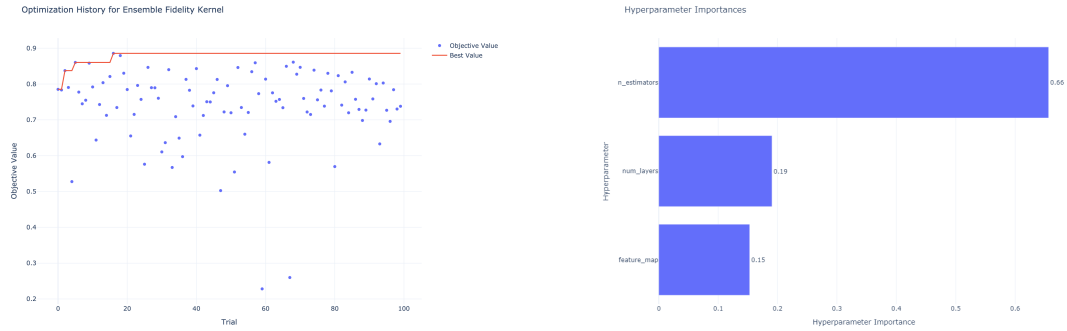
specific characteristics of our dataset subset and provides a challenging evaluation scenario for assessing model performance on minority class detection.
The 5-feature selection was chosen to optimize the trade-off between quantum circuit complexity and computational tractability, allowing for comprehensive hyperparameter exploration within the available computational budget.

7.2. Results

Ensemble fidelity kernel

This section presents the performance analysis of the Ensemble Fidelity Kernel (EFK) through comprehensive hyperparameter optimization and evaluation. The results are organized to provide insights into the optimization process, hyperparameter sensitivity, and overall model performance.

Hyperparameter optimization process



(a) Optimization history showing objective value progression across 100 trials.

(b) Hyperparameter importance scores from Optuna analysis.

Figure 7.2.: Optimization History and Hyperparameter Importance.

Figure 7.2 summarizes the Optuna optimization over 100 trials for the Ensemble Fidelity Kernel.

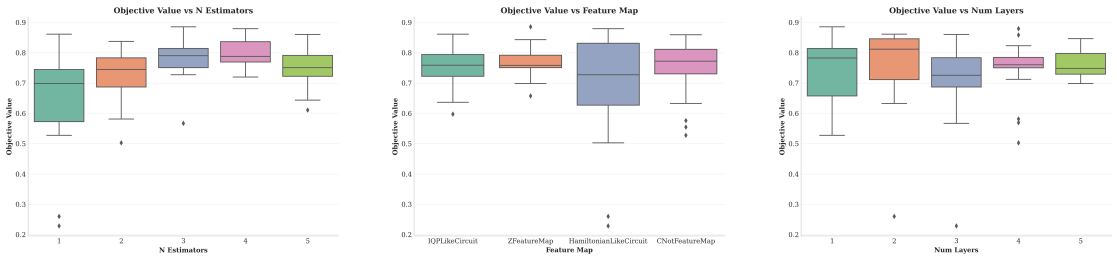
Figure 7.2(a) shows the optimization history. Convergence occurred quickly, within the first 15–20 trials, reaching an objective value close to 0.88. Later trials did not consistently reproduce the highest F1 scores, but values stabilized mostly between 0.7 and 0.8. The spread of results (0.2–0.88) highlights the sensitivity of the model to hyperparameter choices.

Figure 7.2(b) presents the hyperparameter importance analysis. The number of estimators was the dominant parameter, with a 66% importance score, confirming its key role in model performance. The number of layers and the choice of feature

map encoding contributed less, with similar importance values around 15–19%, indicating a secondary but non-negligible influence.

Hyperparameter sensitivity analysis

To gain a deeper understanding of hyperparameter effects, I analyzed the distribution of objective value with respect to each parameter using boxplots.



(a) Boxplot of number of estimators vs objective value

(b) Boxplot of feature map vs objective value

(c) Boxplot of number of layers vs objective value

Figure 7.3.: Impact of individual hyperparameters on EFK performance: (a) number of estimators, (b) feature map type, and (c) number of layers. Box plots show distribution of objective values for different parameter settings across 100 optimization trials.

Figure 7.3 shows how individual hyperparameters shape Ensemble Fidelity Kernel (EFK) performance across 100 optimization trials.

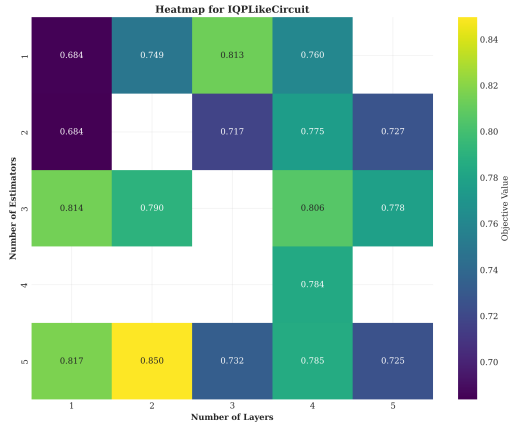
- **Number of Estimators (Figure 7.3(a))**: Performance improves as the number of estimators increases from 1 to 4. The median F1 score reaches its peak at 4 estimators (0.83). Using 5 estimators increases variance, indicating diminishing returns and possible overfitting. Stability improves steadily from 1 to 3 estimators.

- **Feature Map (Figure 7.3(b))**: Most feature maps perform similarly with medians in the 0.75–0.8 range. The Hamiltonian feature map stands out with high variance, spanning 0.2–0.8. This suggests it is highly sensitive to parameter settings and less reliable for consistent outcomes.

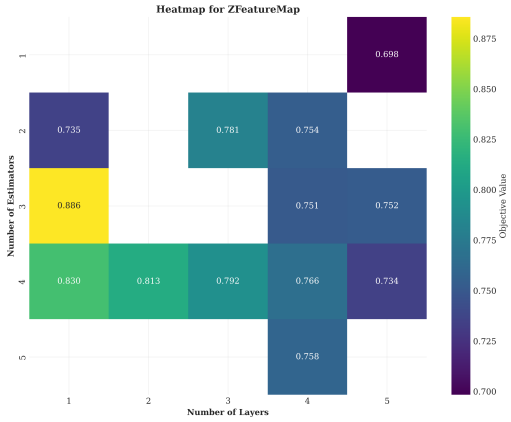
- **Number of Layers (Figure 7.3(c))**: Shallow circuits (1–2 layers) achieve the highest peak scores (0.85) but show larger variance. Deeper circuits (4–5 layers) trade peak performance for stability, yielding lower variance and 0.8 median scores.

Key Insights: EFK optimization requires balancing maximum performance with reliability. The best trade-off appears at 3–4 estimators, stable feature maps (excluding Hamiltonian), and shallow circuits with 1–2 layers.

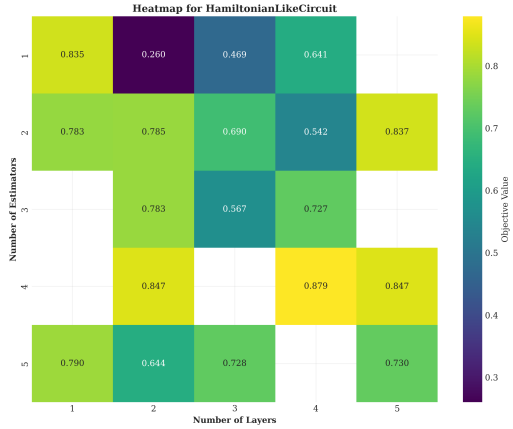
Parameter interaction analysis



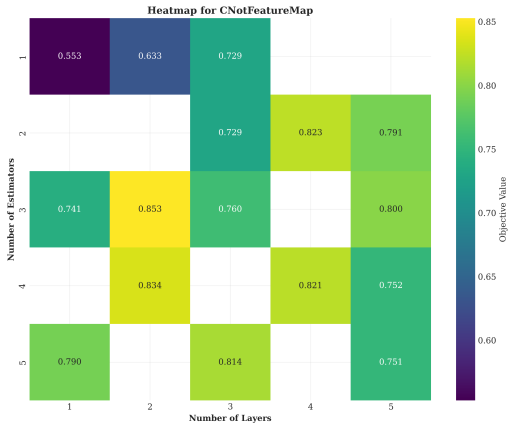
(a) IQP-like circuit: Performance is highly variable, with best scores at shallow depth (1–2 layers) and higher estimator counts.



(b) Z feature map: Scores are more stable, with moderate improvements at 2 layers and medium estimators.



(c) Hamiltonian-like circuit: Displays strong sensitivity, with scattered high values but overall less consistent performance.



(d) CNOT feature map: Shows balanced performance, with multiple regions achieving competitive scores and less sensitivity to depth.

Figure 7.4.: Heatmaps showing the effect of the number of layers and estimators on the objective value (F1 score) across different feature maps. Each subplot fixes a feature map and highlights how performance varies as a function of the other hyperparameters.

Figure 7.4 shows how feature map choice interacts with architectural parameters (number of estimators and layers) to influence EFK performance.

IQP-like Circuit (7.4(a)): Highly unstable with strong variability across parameter settings. A weak gradient appears along the estimator axis: more estimators tend to improve performance, but high variance remains. Both good and poor results appear at all estimator levels, making optimization unreliable.

Z Feature Map (7.4(b)): More structured. Best scores occur with one layer and three estimators. Adding layers reduces performance for all estimator values,

suggesting that shallow encoding is sufficient and extra depth adds unnecessary complexity.

Hamiltonian-like Circuit (7.4(c)): Strongly sensitive to parameters. The heatmap shows scattered high-performance "hot spots" mixed with poor outcomes. This confirms the earlier boxplot trend of high variance, meaning it can reach excellent results but only with precise hyperparameter tuning.

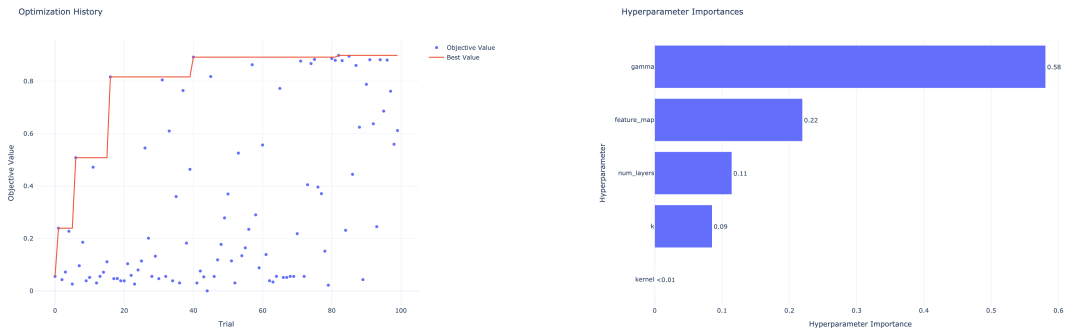
CNOT Feature Map (7.4(d)): Most consistent. Competitive scores appear across several estimator counts and depths with fewer extreme lows. This balanced distribution explains its reliability and shows lower sensitivity to parameter choice.

Overall, CNOT and Z maps are the most reliable for stable performance. IQP and Hamiltonian maps are riskier: they can achieve peaks but demand extensive tuning. The heatmaps confirm that the number of estimators is the dominant hyperparameter, with higher values consistently linked to stronger scores. Circuit depth shows diminishing returns 1–2 layers work best, deeper circuits often reduce accuracy. Feature map choice remains decisive: stable (CNOT, Z) versus sensitive (IQP, Hamiltonian). This sensitivity explains the variance seen during optimization.

Projected quantum kernel using the RBF kernel

This section presents the results for the projected quantum kernel using the RBF kernel as the classical model. The results are organized to provide insights about the optimization process and hyperparameter importance, similar to the EFK analysis, but with an additional analysis of the geometric differences.

Hyperparameter optimization process



(a) Optimization history showing objective value progression across 100 trials.

(b) Hyperparameter importance scores from Optuna analysis.

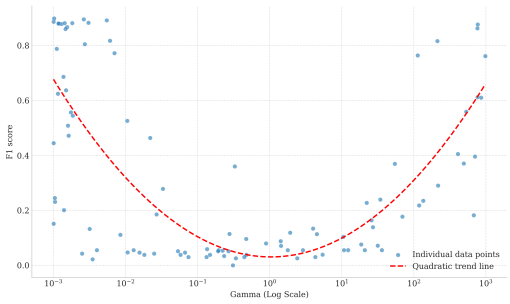
Figure 7.5.: Optimization History and Hyperparameter Importance.

Figure 7.5 shows the optimization history and hyperparameter importance scores for the projected quantum kernel. Compared to the EFK, the PQK exhibits clearer convergence. 7.5(a) shows more consistent improvement and stabilization near the best values, with objective scores ranging from 0 to 0.899. This indicates that PQK performance is more reproducible than EFK.

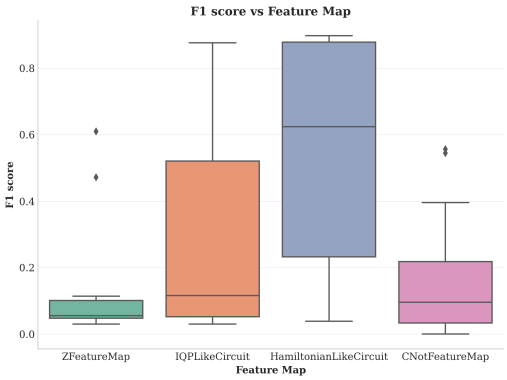
7.5(b) highlights a different hyperparameter hierarchy. Gamma (RBF kernel bandwidth) dominates with 58% importance, confirming that the classical kernel scale strongly shapes PQK performance. Feature map choice ranks second at 22%, followed by the number of layers at 11%. The k parameter, which sets the reduced density matrix size, contributes only 9%. Despite being unique to the PQK, k has less impact compared to the classical kernel bandwidth. This result shows that the quantum projection plays a role, but the classical kernel parameter remains the strongest driver of performance.

Hyperparameter sensitivity analysis

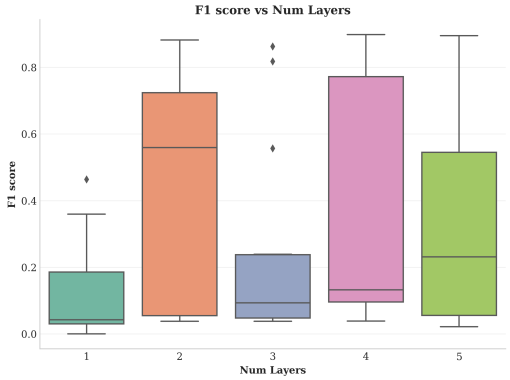
To better understand these optimization patterns, I now turn to a hyperparameter sensitivity analysis. This involves examining the direct relationship between each hyperparameter and the resulting F1 scores using boxplots.



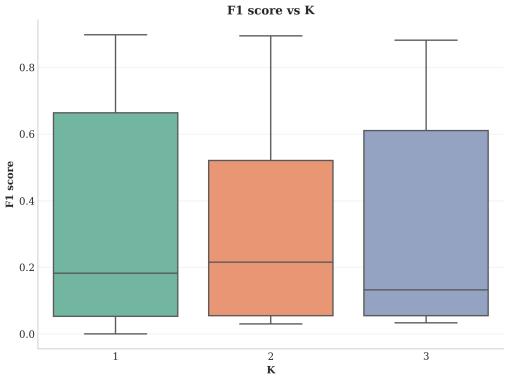
(a) F1 score vs gamma on a logarithmic scale with a quadratic trend line.



(b) Boxplot of feature map vs F1 score.



(c) Boxplot of the number of layers vs F1 score.



(d) Boxplot of k vs F1 score.

Figure 7.6.: Impact of individual hyperparameters on PQK performance: (a) gamma parameter on logarithmic scale, (b) feature map type, (c) number of layers, and (d) k parameter value. Scatter plot shows F1 score variation with gamma; box plots show distribution of F1 scores for different parameter settings across 100 optimization trial.

Figure 7.6 shows the effect of individual hyperparameters on the performance of the projected quantum kernel during the Optuna optimization. Among all parameters, only gamma, the RBF kernel bandwidth, reveals a clear and consistent influence, in line with the importance scores reported earlier. Figure 7.6(a) illustrates a symmetric U-shaped behavior on the logarithmic scale, where the best F1 scores are obtained for very small gamma values (10^{-3} to 10^{-2}) and for large values (10^{2} to 10^{3}). In contrast, the default gamma = 1 yields the lowest scores, around 0.2, which confirms the poor baseline performance previously observed in section 6.2. Figure 7.6(b) compares the encoding feature maps and shows that IQP-like and Hamiltonian-like circuits reach the best performance, with the Hamiltonian circuit giving slightly higher medians but also displaying strong variance that undermines reproducibility. The CNOT and Z feature maps perform the worst, with F1 scores not exceeding 0.4, and while the CNOT map slightly outperforms the Z map, it still suffers from large variability across runs. Figure 7.6(c) examines the number of layers but does not provide a clear trend, most likely because other parameters are not fixed, which suggests that a more controlled study is required to properly assess circuit depth. Finally, Figure 7.6(d) shows that different k values lead to very similar results, consistent with the low importance attributed to this parameter, which confirms that k has little effect on the model's performance within the tested range.

These findings are in line with the results reported by Egginger et al. in [14], where γ was identified as the most impactful hyperparameter, the choice of k showed limited effect on performance, and the number of trotter steps in the Hamiltonian evolution, here represented by the number of layers, was found to influence performance only marginally.

Due to the dominant influence of the gamma parameter on model performance, leaving it unfixed introduces excessive variance and hides potential patterns in the interactions with layers and k. A detailed interaction analysis will be left for future work, where gamma can be systematically controlled. In parallel to this optimization study, we also recorded the geometric difference across all trials as a complementary metric to evaluate the expressivity and separability of the projected quantum kernel feature space.

Geometric difference analysis

This metric complements the F_1 score. While the F_1 score measures task performance, the geometric difference quantifies how much the quantum kernel departs from the classical kernel in terms of data geometry. It allows us to assess whether gains in expressivity come at the cost of larger deviations from the classical model. The following Figure 7.7 shows the scatter plot of the F_1 score against the geometric difference across trials.

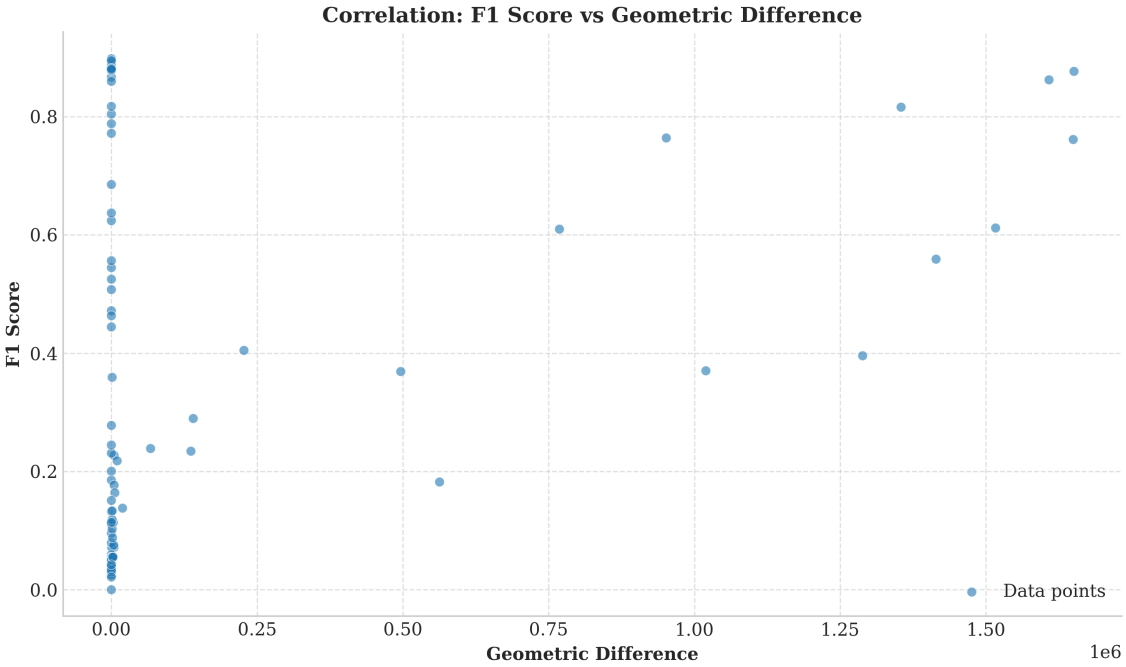


Figure 7.7.: Correlation between F1 Score and Geometric Difference for the projected quantum kernel methods. Each point represents a specific combination of hyperparameters(bandwidth γ, feature map, number of layers, RDM size k)

The correlation plot between F1 score and geometric difference in Figure 7.7 highlights a general inverse trend already reported by Egginger et al. [14]: larger geometric differences, which theoretically indicate stronger separation between quantum and classical kernels, tend to correspond to lower empirical performance. Most trials cluster at low geometric difference values ($< 0.25 \times 10^6$), where F1 scores vary widely between 0.0 and 0.85, confirming that geometric difference alone cannot explain performance since bandwidth γ and the other parameters strongly modulate the outcome. The scarcity of points in the high geometric difference region ($> 0.5 \times 10^6$) further reflects the difficulty of aligning theoretical quantum advantage indicators with practical classification performance.

Yet, a few outliers break this trend. Several configurations with geometric difference above 1.0×10^6 achieve high F1 scores in the 0.6–0.9 range, showing that both metrics can be optimized simultaneously under rare but favorable hyperparameter settings. This nuance supports Egginger et al.' [14]s observation that geometric difference and the empirical performance respond to hyperparameters differently, but also illustrates that the trade-off is not absolute. These cases where both geometric difference and F1 score are high are rare exceptions in the results. Most configurations follow the general inverse trend, which makes these outlier points noteworthy but not representative of the overall behavior.

Out of curiosity, I examined which hyperparameters most strongly affect the geometric difference. I used a Random Forest regression model for this analysis. The

model learns how different hyperparameter choices relate to the geometric difference. It then assigns each parameter an importance score based on how much it helps separate high from low geometric difference values. These scores are normalized to sum to 1. The following Figure 7.8 shows the relative importance of each hyperparameter.

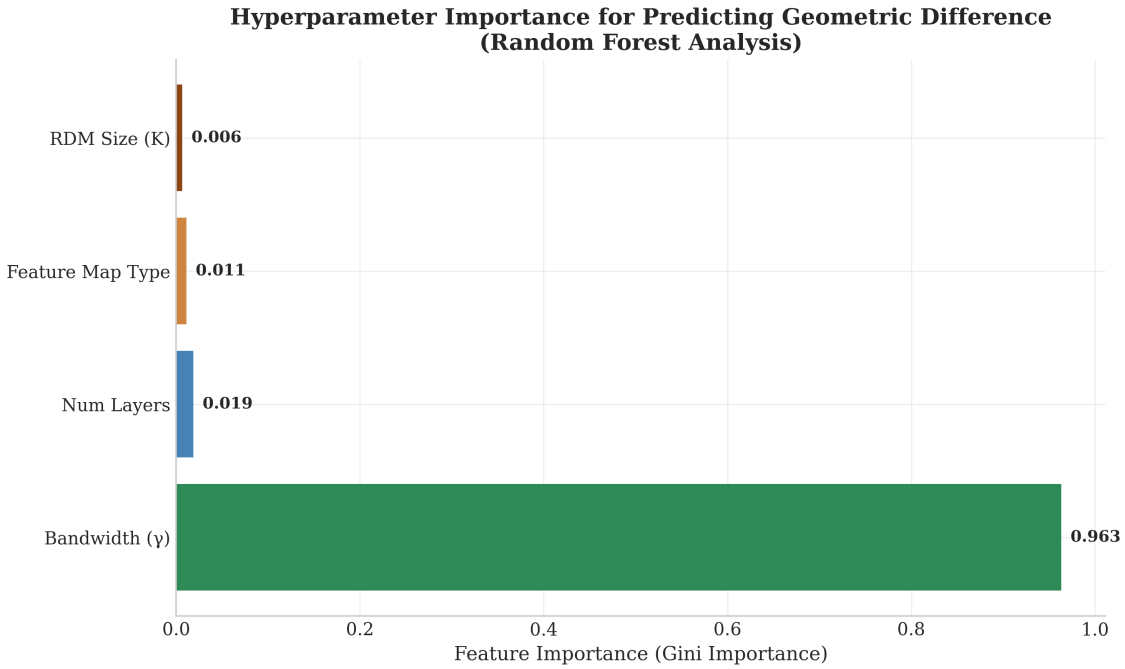
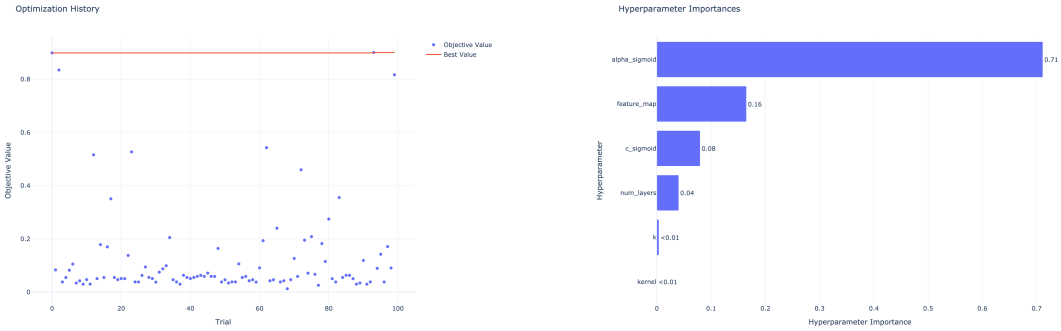


Figure 7.8.: Hyperparameter importance for predicting geometric difference using Random Forest analysis. Feature importance scores (Gini importance) quantify the relative contribution of each hyperparameter to the predictable variance in geometric difference values.

Surprisingly, the results show that γ dominates the influence on geometric difference, with an importance score of 96%. All other quantum parameters contribute negligibly. I find this unexpected because geometric difference is supposed to measure how far the quantum model deviates from the classical one. Yet in practice, it seems almost entirely controlled by γ, a parameter of the classical kernel. In the projected quantum kernel, this makes it look as if the potential quantum advantage is being overshadowed by the classical projection.

Projected quantum kernel using the sigmoid kernel

To complement the analysis with the RBF kernel, I also evaluated the projected quantum kernel using the sigmoid as the underlying classical model. Here I do not delve into the same level of detail, but I present the optimization process and the hyperparameter importance results. The optimization trajectory shows that the overall performance is consistently weaker than in the RBF case, with F_1 scores remaining low across trials.



(a) Optimization history showing objective value progression across 100 trials.

(b) Hyperparameter importance scores from Optuna analysis.

Figure 7.9.: Optimization History and Hyperparameter Importance.

The optimization history (Figure 7.9(a)) displays relatively stable performance, with objective values clustering around the 0–0.2 range and showing less dramatic convergence compared to the RBF-based kernel. A few outliers reach F_1 scores as high as 0.8, but these are exceptions rather than the rule. The hyperparameter importance analysis for the F_1 score confirms the same trend observed before, with α dominating at 75% while the other parameters contribute negligibly. From this insight, the RBF kernel clearly outperforms the sigmoid, which makes it unnecessary to proceed with further detailed analysis for the sigmoid case.

Conclusion Overall, the results demonstrate that the projected quantum kernel does not yield a significant quantum advantage. The optimization history and hyperparameter sensitivity analysis consistently show that performance is dictated by the classical parameters. For the RBF case, γ explains nearly all the variance in both F1 score and geometric difference. For the sigmoid case, α dominates with 75% importance, while all other parameters contribute minimally. Quantum-specific parameters such as the number of layers or the value of k have negligible impact across both setups.

This consistent pattern indicates that the projected quantum kernel is effectively reduced to a classical kernel with only a weak quantum contribution. While the geometric difference metric highlights structural separation between quantum and classical feature maps, it is still governed primarily by γ , further confirming that the quantum projection does not drive the performance. Whether using RBF or sigmoid as the base, the role of the quantum component remains overshadowed by the classical kernel.

These findings suggest that without careful control of the classical projection or the introduction of more expressive quantum circuits, the projected quantum kernel is unlikely to provide a practical advantage over its classical counterparts.

8 Project impact

8.1. Environmental impact

One of the key competencies expected from us as engineering students is the ability to critically assess the environmental impact of our work, alongside its technical outcomes. From the MLOps dashboard, I can see that this project has required substantial computational resources, with about 105,288 CPU hours and 47 GPU hours logged. These figures suggest a significant energy demand, even though the actual carbon footprint depends on the cloud provider's infrastructure and energy mix, which I do not have access to. What stands out to me is the relatively low utilization efficiency: CPU usage averaged 38.2%, and RAM was at 5.34%, which points to a level of over-provisioning. This means the system consumed more capacity than was strictly necessary for the workloads. At the same time, storage use was minimal, so its contribution to the environmental footprint was negligible. In my view, this confirms that the main driver of environmental impact in this project was the compute load itself, while the storage component remained almost irrelevant.

8.2. Social impact

As part of the open discussions about AI ethics with my manager, one of the possible societal angles that caught my attention, is that classical neural networks are often seen as black boxes since their decisions are hard to interpret. Quantum circuits, on the other hand, are reversible by design. This reversibility could, in theory, make them more interpretable and link to ethical concerns around AI transparency. In the context of AI ethics, transparency and accountability are important for building trust in models that affect real decisions. If quantum models can provide clearer reasoning paths or make it easier to trace back how an output was generated, they could address part of the interpretability problem. I don't fully see this advantage in practice yet, but it shows an interesting direction where quantum models might contribute differently than classical ones.

8.3. Impact on company strategy and economic challenges

This project has a direct impact on company strategy and the economic challenges linked to quantum technologies. It is developed with a French insurance company that contributed its expertise in data and cybersecurity, which ensured that the research problem stays connected to a real operational need. The collaboration also involves IQM, where the quantum circuits were tested on real hardware. Multiverse Computing plays a central role in driving this work within the broader French government scientific development programs. The objective is to address cybersecurity issues, which are highly strategic for industry. The methods are designed to be general so they can eventually be deployed as products, making the results relevant beyond a single use case. The project also aims at producing a scientific publication to reinforce credibility and foster customer trust, which is essential for the adoption of such advanced technologies in a competitive market.

9 Conclusion

This report presented an investigation of quantum kernel methods for anomaly detection in network traffic. The work addressed several complementary approaches. First, we studied the Randomized Measurement method and identified the presence of plateaus in performance before applying error mitigation. Next, we implemented and evaluated the Projected Quantum Kernel, considering different variants of the construction. Finally, we benchmarked these approaches against the Ensemble Fidelity Kernel, which served as a competitive baseline.
The experiments also showed that different kernel constructions have distinct trade-offs between stability and sensitivity, which must be considered when designing anomaly detection pipelines.
Beyond the empirical findings, this project demonstrated the feasibility of applying quantum kernels in an unsupervised anomaly detection setting, whereas most previous works focused on supervised classification. This distinction makes the results more relevant to realistic network security scenarios where labeled data are scarce.

Future Work

Several directions arise naturally from this work:

- Experimenting with larger dataset samples to test the robustness of the methods at scale.
- Running the pipelines on noisy simulators and quantum hardware to assess hardware-induced limitations.
- Benchmarking against the quantum autoencoder previously implemented by the team.
- Exploring trainable kernels as a promising alternative to fixed kernel designs.

Personal Reflection

This internship offered a comprehensive learning experience across technical, professional, and personal levels. On the technical side, I deepened my understanding of quantum kernel methods and gained software engineering skills, including version control with Git, code deployment, and adherence to industry-grade workflows with code quality checks. I also conducted a thorough review of the state of the

art. On the professional side, I strengthened my critical analysis, problem-solving, and teamwork abilities. I actively shared progress, iterated on challenges with colleagues, and presented my results during consortium meetings, which improved my communication skills.
Overall, the project not only advanced the study of quantum kernels for anomaly detection but also enriched my academic and professional development.

A Feature Engineering

A.1. Feature Engineering Process

This appendix details the feature engineering steps applied to the network flow dataset prior to training the models. The goal was to reduce redundancy, improve interpretability, and select the most informative features.

A.1.1. Variance Thresholding

We first removed features with very low variance across samples, as they carry little discriminative information.

A.1.2. Clustering-Based Feature Selection

We applied hierarchical clustering-based feature engineering following the methodology described in [34]. The steps are:

1. Compute pairwise correlation distances between features.
2. Build a hierarchical linkage using the average method.
3. Determine the optimal number of clusters by maximizing the silhouette score.
4. For each cluster, select a representative feature (medoid), i.e., the feature with the minimum average distance to all other features in that cluster.
5. Return a dataset containing only the selected representative features.

This method reduces redundancy while retaining interpretability of the selected features.

A.1.3. Dimensionality Reduction

We further applied Principal Component Analysis (PCA) on the selected features to capture the main sources of variance and reduce dimensionality.

A.1.4. Final Feature Set

The final network flow dataset contains the following groups of features:

- **Flow duration**: total duration of the flow.
- **Packet statistics**: total number of forward/backward packets; length of packets (min, max, mean, std); packet rate (packets/s); byte rate (bytes/s).
- **Inter-arrival times**: mean, max, min, std of inter-arrival time between packets (bidirectional, forward, backward).
- **TCP flag counts**: counts of flags such as SYN, ACK, CWR, etc.
- **TCP initialization features**: mean, std, counts related to the start of TCP connections.
- **Unique ports**: number of unique source and destination ports.
- **Protocol counts**: counts of different protocols (TCP, UDP, web protocols, etc.).

A.1.5. Hierarchical Clustering Dendrogram

Figure A.1 shows the hierarchical clustering tree of the features, illustrating the grouping structure that guided representative feature selection.

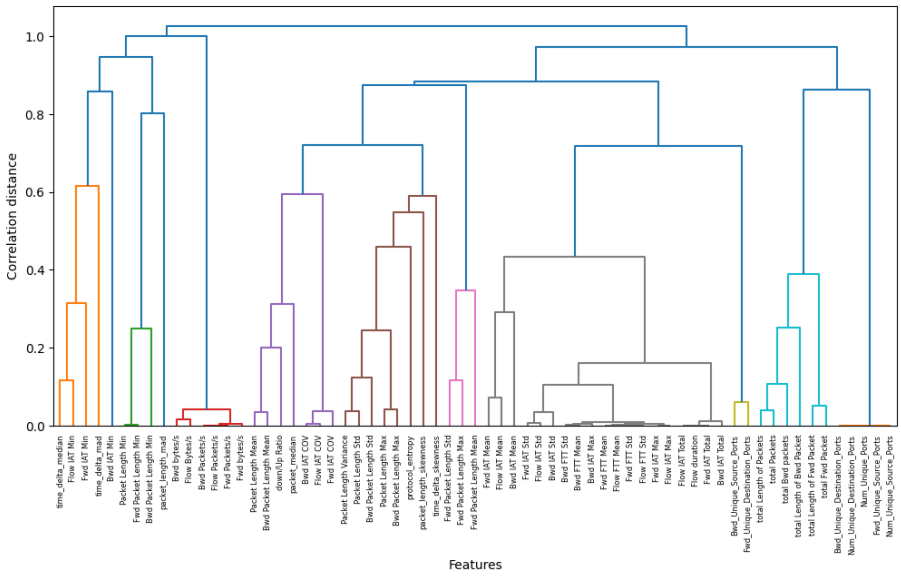


Figure A.1.: Hierarchical clustering tree of the features.

Bibliography

[1] Maria Schuld. "Supervised quantum machine learning models are kernel methods". Xanadu, Toronto, ON, M5G 2C8, Canada,(2021).

[2] Seth Lloyd, Maria Schuld, Aroosa Ijaz, Josh Izaac and Nathan Killoran. "Quantum embeddings for machine learning" (2022).

[3] S. Wang, Journal of Mathematics Research 7, 175 (2015).

[4] E. Farhi and H. Neven, arXiv preprint arXiv:1802.06002 (2018).

[5] N. Wiebe, D. Braun, and S. Lloyd, Physical Review Letters 109, 050505 (2012).

[6] M. Schuld, M. Fingerhuth, and F. Petruccione, Europhysics Letters 119, 60002 (2017).

[7] Seth Lloyd, Maria Schuld, Aroosa Ijaz, Josh Izaac, and Nathan Killoran. *Quantum embeddings for machine learning*. 2022.

[8] Andreas Elben, Benoît Vermersch, Rick van Bijnen, Christian Kokail, Tiff Brydges, Christine Maier, Peter Zoller, and Rainer Blatt. *Cross-platform verification of intermediate scale quantum devices*. Nature, 582(7810):397–400, 2020.

[9] Tobias Haug, Chris N. Self, and M. S. Kim. *Quantum machine learning of large datasets using randomized measurements*. QOLS, Blackett Laboratory, Imperial College London, UK, 2022.

[10] Supanut Thanasilp, Samson Wang, M. Cerezo, and Zoë Holmes. *Exponential concentration in quantum kernel methods*. Nature Communications, vol. 15, article 5200, 2024. https://doi.org/10.1038/s41467-024-49287-w.

[11] H.-Y. Huang, R. Kueng, and J. Preskill. *Predicting many properties of a quantum system from very few measurements*. Nature Physics, vol. 16, pp. 1050–1057, 2020. https://doi.org/10.1038/s41567-020-0932-7.

[12] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, and J. R. McClean. *Power of data in quantum machine learning*. Nature Communications, vol. 12, article 2631, 2021. https://doi.org/10.1038/s41467-021-22539-9.

[13] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta. *Quantum computing with Qiskit*.

ACM Transactions on Quantum Computing, 4(3):1–27, 2023. https://doi.org/10.1145/3564216.

[14] Sebastian Egginger, Alona Sakhnenko, and Jeanette Miriam Lorenz. *A hyperparameter study for quantum kernel methods*,2024.

[15] Y. Borchani. *Advanced malicious beaconing detection through AI*. Network Security, Volume 2020, Issue 3, 2020. `https://doi.org/10.1016/S1353-4858(20)30030-1`

[16] Michael Kölle, Afrae Ahouzi, Pascal Debus, Elif Çetiner, Robert Müller, Daniëlle Schuman, and Claudia Linnhoff-Popien. *Efficient Quantum One-Class Support Vector Machines for Anomaly Detection Using Randomized Measurements and Variable Subsampling*, 2024.

[17] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. *Quantum machine learning*. Nature, Volume 549, pages 195–202, 2017.

[18] M. Cerezo, Guillaume Verdon, Hsin-Yuan Huang, Lukasz Cincio, and Patrick J. Coles. *Challenges and opportunities in quantum machine learning*. Nature Computational Science, Volume 2, pages 567–576, 2022.

[19] Maria Schuld and Nathan Killoran. *Quantum Machine Learning in Feature Hilbert Spaces*. Physical Review Letters, Volume 122, 040504, 2019.

[20] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[21] Sofiene Jerbi, Lukas J. Fideler, Hendrik Poulsen Nautrup, Jonas M. Kübler, Hans J. Briegel, and Vedran Dunjko. *Quantum machine learning beyond kernel methods*. Nature Communications, Volume 14, Article number: 517, 2023.

[22] Vojtěch Havlíček, Antonio D. Córcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta. *Supervised learning with quantum-enhanced feature spaces*. Nature, Volume 567, pages 209–212, 2019.

[23] Carsten Blank, Daniel K. Park, June-Koo Kevin Rhee, and Francesco Petruccione. *Quantum classifier with tailored quantum kernel*. npj Quantum Information, Volume 6, Article number: 41, 2020.

[24] Thomas Hubregtsen, David Wierichs, Elies Gil-Fuster, Peter-Jan H. S. Derks, Paul K. Faehrmann, and Johannes Jakob Meyer. *Training quantum embedding kernels on near-term quantum computers*. Physical Review A, Volume 106, 042431, 2022.

[25] Beng Yee Gan, Daniel Leykam, and Supanut Thanasilp. *A Unified Framework for Trace-induced Quantum Kernels*, 2023.

[26] Yudai Suzuki and Muyuan Li. *Effect of alternating layered ansatzes on trainability of projected quantum kernel*, 2023.

[27] Shungo Miyabe, Brian Quanz, Noriaki Shimada, Abhijit Mitra, Takahiro Yamamoto, Vladimir Rastunkov, Dimitris Alevras, Mekena Metcalf, Daniel J.M. King, Mohammad Mamouei, Matthew D. Jackson, Martin Brown, Philip Intallura, and Jae-Eun Park. *Quantum Multiple Kernel Learning in Financial Classification Tasks*, 2023.

[28] Boualem Djehiche and Björn Löfdahl. *Quantum Support Vector Regression for Disability Insurance*. Risks, Volume 9, Issue 12, 216, 2021. `https://doi.org/10.3390/risks9120216`

[29] Evan Peters, João Caldeira, Alan Ho, Stefan Leichenauer, Masoud Mohseni, Hartmut Neven, Panagiotis Spentzouris, Doug Strain, and Gabriel N. Perdue. *Machine learning of high dimensional data on a noisy quantum processor*. npj Quantum Information, Volume 7, Article number: 161, 2021.

[30] Annie E. Paine, Vincent E. Elfving, and Oleksandr Kyriienko. *Quantum kernel methods for solving regression problems and differential equations*. Physical Review A, Volume 107, 032428, 2023.

[31] Frederic Rapp, David A. Kreplin, Marco F. Huber, and Marco Roth. *Reinforcement learning-based architecture search for quantum machine learning*. Machine Learning: Science and Technology, Volume 6, Number 1, 015041, 2025. `https://doi.org/10.1088/2632-2153/adaf75`

[32] F. Pedregosa et al. *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, Volume 12, pages 2825–2830, 2011.

[33] Wei Dai and Jicong Fan. *AutoUAD: Hyper-parameter Optimization for Unsupervised Anomaly Detection*. Published as a conference paper at ICLR 2025, 2025.

[34] Conor O'Sullivan. *Feature Selection with Hierarchical Clustering for Interpretable Models*. Medium, 2024.