**Project details:**

Implementation from previous classification model (classical computation) to Quantum model and computing using Quantum Machine Learning

Quantum Machine Learning (QML) can be implemented through two ways:

* Qiskit Machine Learning (qiskit library python developed by IBM)
* Pennylane

**Qiskit QSVC Implementation**

**Notebook**: QML\_implementation\_qsvc.ipynb

* Used merged.csv with top 20 selected features; BENIGN labeled as 1, attacks as 0.
* Standardized features using StandardScaler; reduced to 4 dimensions using PCA.
* Selected 500 stratified samples due to kernel matrix size limitations.
* Built a FidelityQuantumKernel using a 4-qubit ZZFeatureMap (2 reps, linear entanglement).
* Trained a QSVC model and evaluated on training set.

**Challenges**:

* Long training time due to quantum kernel evaluation
* Imbalanced kernel-induced decision boundary led to poor attack classification

**Notebook**: qml\_qsvc\_2.ipynb

* Used same top 20 features from merged.csv; BENIGN as 1, attacks as 0.
* Applied StandardScaler and PCA to reduce to 4 features (qubits).
* Selected 1000 stratified samples for training.
* Constructed a 4-qubit FidelityQuantumKernel using ZZFeatureMap (2 reps).

**Challenges**:

* Extremely high training time due to large 1000×1000 quantum kernel matrix.
* Even with more samples, attack classification degraded (Recall: 16% vs 23% earlier).
* Model again overfit to benign class in the quantum feature space.

**Qiskit VQC Implementation (Colab)**

**Notebook**: QML\_VQC.ipynb

* Loaded merged.csv with top 20 features; labeled BENIGN as 1, attacks as 0.
* Balanced the dataset to 400 benign and 400 attack samples.
* Applied MinMaxScaler and reduced to 4 features via PCA.
* Used an 80:20 stratified train-test split (320 train, 80 test per class).
* Built a Variational Quantum Classifier (VQC) using:
  + ZZFeatureMap (2 reps) as feature encoder
  + TwoLocal ansatz (ry, rz, cz, 2 reps)
  + COBYLA optimizer (maxiter=100)
  + Executed on AerSimulator via BackendSampler.

**Observations**:

* VQC gave balanced classification performance compared to QSVC, but at the cost of lower overall accuracy.
* Output variation helped overcome the constant predictions issue seen in earlier PennyLane VQC runs.

**Notebook**: VQC\_2.ipynb

* Loaded merged.csv with top 20 selected features; BENIGN = 1, ATTACK = 0.
* Applied MinMaxScaler and PCA (4 components).
* Stratified train-test split: 2000 training, 400 testing samples.
* VQC setup:
  + Feature Map: ZZFeatureMap (reps=2)
  + Ansatz: TwoLocal with ry, rz, and cz gates (2 reps)
  + Optimizer: SPSA (60 iterations)
  + Backend: AerSimulator via BackendSampler

**Observations**:

* High accuracy but extremely poor detection of attacks (only 30% recall).
* Model leaned heavily toward predicting benign samples.
* Indicates decision boundary imbalance, possibly due to optimizer sensitivity or ansatz expressibility limits.

**PennyLane QSVC Implementation**

**Notebook**: pennylane\_svc.ipynb

* Loaded merged.csv → saved top 20 features into mergedTop20.csv; label encoded (BENIGN = 1, ATTACK = 0).
* Selected top 5 features using Random Forest importance.
* Balanced the data with 500 benign and 500 attack samples.
* Standardized features and reduced to 3 components using PCA.
* Constructed a custom **fidelity kernel** using a qml.qnode:
  + Custom feature\_map using Hadamard, RZ, and CNOT gates.
  + Computed full training (800×800) and test (200×800) kernel matrices.
* Trained SVC(kernel="precomputed") on quantum kernel.
* Evaluated on both training and test sets.

**Issues Encountered**:

* Very slow kernel computation: ~1 min per 13 rows.
* Skewed model output (low recall for attack class, biased toward benign).
* Accuracy plateaued around 72%, despite balanced data and reduced features.

**Notebook**: pennylane\_svc\_2.ipynb

* Used mergedTop20.csv with top 20 features; label-encoded (BENIGN = 1, ATTACK = 0).
* Selected top 5 features via Random Forest importance.
* Balanced subset: 500 benign + 500 attack samples.
* Applied StandardScaler and reduced to 3 PCA components (for qubits).
* 3D PCA scatter plot visualized class separation — partial overlap observed.
* Added t-SNE projection for exploratory analysis.
* Updated quantum kernel:
  + Feature map used RX and RZ rotations followed by CNOT → RZ → CNOT entanglement.
  + Kernel matrix computed via adjoint overlap fidelity method.
* Trained SVC with precomputed quantum kernel matrix (800×800 train, 200×800 test).

**Observations**:

* Class separation was visually evident, but fidelity kernel still struggled with attack classification.
* New feature map structure didn’t yield major accuracy gain compared to previous run.
* Consistent bias toward benign class persisted.

**Notebook**: pennylane\_svc\_3ipynb

* Used mergedTop20.csv → selected top 15 features via Random Forest.
* Balanced dataset: 750 BENIGN + 750 ATTACK samples.
* Scaled using StandardScaler, reduced to 3 components via PCA (mapped to 3 qubits).
* 3D PCA scatter plot showed improved but still overlapping class separation.
* Quantum Kernel Setup:
  + **Feature Map**: AngleEmbedding (rotation='Y') + BasicEntanglerLayers
  + Input normalized to π range for stable training.
  + Kernel computed using adjoint overlap via qml.qnode
* Computed large kernel matrices:
  + Train: 1200×1200 → ~93 mins
  + Test: 300×1200 → ~51 mins
* Trained classical SVM with precomputed quantum kernel.

**Improvements**:

* Most balanced performance so far across both classes.
* Recall values more equitable between attack and benign traffic.
* Embedding structure yielded better separation than earlier custom circuits.

**Notebook**: pennylane\_svc\_4ipynb

* Used mergedTop20.csv and selected top 15 features using Random Forest importance.
* Balanced subset: 750 BENIGN + 750 ATTACK samples.
* Scaled via StandardScaler; reduced to 3 PCA components (→ 3 qubits).
* 3D PCA scatter confirmed reasonable class separation.
* Quantum Kernel Setup:
  + **Embedding**: AngleEmbedding with Y-rotations
  + **Ansatz**: StronglyEntanglingLayers with 2 layers, full connectivity
  + Input normalized to π range
  + Kernel computed using adjoint fidelity
* Large kernel matrices:
  + Train: 1200×1200 → **~2.66 hrs**
  + Test: 300×1200 → **~1 hr**
* Trained SVC(C=10) with precomputed kernel

**Observations**:

* Highest generalization accuracy among all PennyLane-based QSVCs
* Excellent benign classification; attack class improved from prior runs
* Deep entanglement improved fidelity-space separation

**Pennylane VQC Implementation**

**Notebook**: pennylane\_vqc.ipynb

* Used mergedTop20.csv with original multi-class labels.
* Label encoded all 15 classes (BENIGN + 14 attack types).
* PCA reduced features to 4 qubits → normalized in [0, 2π] using MinMaxScaler.
* Labels mapped from {0,1} → {−1,1} for binary hinge loss (but applied to multi-class labels).
* Training subset: 200 samples; full test set retained for real-world generalization.
* **Quantum Model**:
  + Circuit: AngleEmbedding + StronglyEntanglingLayers on 4 qubits
  + Loss: Squared loss; optimizer: Adam
  + Training: 30 epochs using qml.lightning.qubit backend

**Challenges**:

* VQC is **inherently binary**; adapting to multi-class without one-vs-all or hybrid schemes leads to collapsed predictions
* Model outputs **constant decision boundaries**, ignoring minority classes
* Even with high accuracy, **practical utility is poor due to zero attack detection**
* Significant imbalance (BENIGN dominates dataset) → overshadowed smaller attacks

**Notebook**: pennylane\_vqc\_2.ipynb

* Used mergedTop20.csv; binary labels: BENIGN (1), ATTACK (0) → mapped to {−1, 1}.
* PCA reduced features to 4 components; scaled to [0, 2π].
* Balanced dataset: 500 BENIGN + 500 ATTACK → 80:20 train-test split.

**Variations Tested**:

* **Circuit Types**:
  + StronglyEntanglingLayers
  + BasicEntanglerLayers
* **Optimizers**:
  + Adam, Momentum, SGD (learning rate = 0.005)
* **Weight Initialization**:
  + normal\_0.01: small Gaussian
  + uniform\_2pi: full phase spread

**Training**:

* Each configuration trained for 30 epochs.
* Metrics evaluated: Accuracy, F1 Score, AUC, Final Loss.

**Best Result**:

* **BasicEntanglerLayers + Adam + uniform\_2pi**
* **StronglyEntanglingLayers + Momentum + uniform\_2pi**

**Key Observations**:

* uniform\_2pi initialization consistently improved AUC over normal\_0.01.
* BasicEntanglerLayers performed slightly better in terms of stability.
* Most other configs led to stagnated or flat model output (F1 stuck at ~0.66).
* Despite being shallow (3 layers), structure and init scheme impacted performance more than optimizer alone.

**Notebook**: pennylane\_vqc\_3.ipynb

* Input: mergedTop20.csv with top 20 features, label mapping → **BENIGN: +1, ATTACK: -1**
* Class balancing via **upsampling of ATTACK class** using resample
* Data split: 1000 train / 200 test (stratified)
* Feature engineering:
  + Scaled via MinMaxScaler, reduced with PCA (4 components)
  + Normalized to [0, 2π] for angle encoding (RY)
* Quantum Model:
  + Device: default.qubit, 4 qubits
  + Ansatz: StronglyEntanglingLayers with 3 layers
  + Optimizer: GradientDescent, 100 epochs, batch size = 50
  + Loss: Mean squared error (square\_loss)

**Insights**:

* **High recall (91%) on Benign class**, but ATTACK detection recall remains low (37%)
* More balanced performance than earlier trials, but still underfitting on minority class
* Indicates need for further tuning (layers, regularization, more expressive circuits)

**Notebook**: pennylane\_vqc\_4.ipynb

* Used mergedTop20.csv; selected 20 top-ranked features.
* Label mapped to: **BENIGN → +1**, **ATTACK → −1**
* **Upsampled minority class (ATTACK)** to match BENIGN class (≈49k each)
* Feature scaling via MinMaxScaler and **PCA** to 4 dimensions → scaled for RY encoding over [0, 2π]
* Train/test split: 2000 samples train, 400 test (**stratified**)
* Variational classifier:
  + Encoding: RY rotation
  + Ansatz: StronglyEntanglingLayers (3 layers)
  + Device: default.qubit (pennylane simulator)
* Training loop: 100 epochs, batch size = 50, optimizer = Gradient Descent

**Insights**:

* Final model significantly improved **Attack class detection**, with **Precision = 0.86**
* **Recall on Attack** remained limited (42%), likely due to limited data or circuit capacity
* Indicates better generalization than earlier trials—thanks to class balancing, stratified split, and batched training

**Major Challenges Faced**

**1. Qiskit Kernel Deadlocks and Scalability**

* Kernel matrix evaluation for QSVC using **FidelityQuantumKernel** (1000×1000) caused:
  + Extremely high memory and time usage.
  + **Deadlock situations** in AerSimulator on Google Colab.
  + Kernel size forced you to restrict to small training subsets (e.g., 500 samples).
  + Kernel-induced **decision boundary bias toward BENIGN class** — poor recall on attacks.

**2. Constant Outputs in Pennylane VQC**

* Early Pennylane VQC runs yielded **constant predictions**, failing to distinguish classes.
* Root cause: **insufficient expressibility**, improper initialization, and no batch-wise training.
* Solution: Adjusted initialization strategy (uniform\_2π), added batch training and used tanh activation over output.

**3. High Bias Toward Benign Class**

* Even with balanced training data, VQC often favored predicting benign samples:
  + Likely due to **feature map weakness**, **optimizer sensitivity**, or **circuit depth** constraints.
  + Improvements came after **switching to StronglyEntanglingLayers with deeper layers** and larger batch sizes.

**4. Kernel-based Models (QSVC) Plateau**

* Accuracy and recall values **saturated at ~72%**, even with different feature maps and embeddings.
* **Kernel computation was slow** (1 min per 13 rows in some runs).
* Decision surfaces remained limited in complexity even after modifying entanglement structure.

**Future Improvement Scope**

**1. Hybrid Classical-Quantum Model**

* Use quantum layers as **feature extractors** combined with classical classifiers (e.g., Random Forest, XGBoost).
* Helps reduce pressure on quantum circuits while retaining benefits of quantum encoding.

**2. Feature Map Learning**

* Implement **learnable feature maps** using data re-uploading strategies to allow end-to-end learning.
* Explore QML architectures like **Quantum Convolutional Neural Networks (QCNNs)**.

**3. Error Mitigation + Noise-aware Training**

* Explore **noise-aware simulations**
* PennyLane supports noise modeling

**4. Multi-class Attack Detection**

* Current VQC is binary; extend using **one-vs-rest strategy** or build a **hybrid multi-class classifier**.
* Many CICIDS attack classes were dropped in binarization — but detecting specific threats is critical.

**5. Dynamic Circuit Depth and Qubit Allocation**

* Adaptive circuit depth based on complexity of input features or detected overfitting/underfitting.
* Consider **entanglement tuning**: not all layers need full connectivity; investigate locality-aware designs.

**6. Automated Hyperparameter Search**

* Use **Bayesian optimization** or **grid/random search** to automate:
  + Optimizer selection
  + Learning rate
  + Number of layers/qubits
  + Embedding type