IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 11 July 2023, accepted 7 August 2023, date of publication 14 August 2023, date of current version 21 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3304990

RESEARCH ARTICLE

Quantum-Enhanced Support Vector Machine for Sentiment Classification

FARISKA ZAKHRALATIVA RUSKANDA[1,2], (Member, IEEE), MUHAMMAD RIFAT ABIWARDANI[1], RAHMAT MULYAWAN[1,3,4], (Member, IEEE), INFALL SYAFALNI[1,3], (Member, IEEE), AND HARASHTA TATIMMA LARASATI[1,5], (Member, IEEE)

[1]School of Electrical Engineering and Informatics, Bandung Institute of Technology, Bandung 40132, Indonesia
[2]Artificial Intelligence Center, Bandung Institute of Technology, Bandung 40132, Indonesia
[3]University Center of Excellence on Microelectronics, Bandung Institute of Technology, Bandung 40132, Indonesia
[4]Research Collaboration Center for Quantum Technology 2.0, Bandung Institute of Technology, Bandung 40132, Indonesia
[5]School of Computer Science and Engineering, Pusan National University, Busan 46241, South Korea

Corresponding author: Infall Syafalni (infall@ieee.org)

This work was supported and funded by the 2022 Young Researcher Grant from the School of Electrical Engineering and Informatics, Bandung Institute of Technology.

ABSTRACT Quantum computers have potential computational abilities such as speeding up complex computations, parallelism by superpositions, and handling large data sets. Moreover, the field of natural language processing (NLP) is rapidly attracting researchers and engineers in order to build larger model computations of NLP. Thus, the use of quantum technology in NLP tasks, especially sentiment classification, has the potential to be developed. In this research, we investigate the best technique to represent sentiment sentences so that sentiment can be analyzed using the Quantum-Enhanced Support Vector Machine (QE-SVM) algorithm. Investigations were carried out using circuit parameter optimization methods and data transformation. The pipeline of the proposed method consists of sentence-to-circuit conversion, circuit parameter training, state vector formation, and finally the training and testing processes. As a result, we obtained the best classification results with an accuracy of 93.33% using the SPSA optimization method and PCA transformation data. These results have also outperformed the baseline SVM method.

INDEX TERMS Sentiment classification, SVM, quantum-enhanced, quantum representation.

I. INTRODUCTION

Nowadays, opinions can be expressed easily in various online media by anyone. Therefore, this data is an important source that can be used to derive a person's sentiment value for something, such as a product, service, or person. The current sentiment classification process is generally carried out using Natural Language Processing (NLP) technology. Opinions or subjective sentences are automatically labeled as positive or negative sentiment values in sentiment classification [1]. This sentiment value can also be used further to make product profile summaries [2], vote predictions [3], or improve customer service [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Okyay Kaynak.

Sentiment classification is generally solved using a Machine Learning (ML) algorithm that utilizes labeled training data to predict sentiment values. The learning algorithm allows the prediction process to better deal with opinion sentences characteristic of human language or natural sentences. Quantum ML technology can be used to solve this problem. This technology combines quantum computers and artificial intelligence, especially learning algorithms. Quantum computers are used to solve complex problems intractable and solved by classical computers [5]. One of the QML methods is Quantum Enhanced - Support Vector Machine (QE-SVM) proposed in 2018 [6]. This method has advantages over SVM in the form of a quantum kernel, namely kernel functions that can be computed using quantum circuits. This kernel accepts as input a feature map representing a complex vector space. This method outperforms SVM on various structured (numeric) datasets, using feature map transformations and adjustments to the rotation factor [7].

The nature of complex vectors that QE-SVM can handle aligns with the nature of subjective sentences in sentiment classification data. Therefore, using QE-SVM in the sentiment classification task is a potential research area. As one of the tasks in NLP, handling sentiment classification in a quantum environment is carried out using the Quantum NLP (QNLP) methodology [8]. This methodology uses a compositional language structure in the form of grammar and semantics constructed in a quantum way.

The main problem of the research is to find the best data representation for quantum NLP to represent a sentiment in a sentence or called sentiment classification. Moreover, several optimizers such as SPSA and ANN are explored in order to improve the classification performance. Finally, we also expand the improvements of sentiment classifications from the classical SVM method to the Quantum Enhanced-SVM (QE-SVM) method.

Our previous work [9] has formulated a quantum representation for the sentiment classification task [9]. We used a state vector representation and particular negation handling with the Not-box operation. However, the dimensions of the vector representation are large, and the prediction results could be more optimal (81.67% accuracy). The challenge that needs to be solved is building a proper quantum representation of subjective sentences that can be computed quickly and precisely using the QE-SVM learning algorithm.

In this paper, the focus is on exploring how to use quantum natural language processing (QNLP) to represent the sentiment of a sentence. The aim is to come up with an effective and efficient quantum representation of subjective sentences that can be used for quantum sentiment classification. We modified an existing experimental QNLP pipeline (described in [10]) to better suit their needs, particularly during the optimization stage. The methodology involves converting sentences into circuits, training circuit parameters, and reading state vectors, followed by techniques for transforming the state vector data to work with the QE-SVM classifier.

In summary, this paper has two main contributions. The first is developing an effective and efficient quantum representation of subjective sentences. We suggest using the X-gate quantum operation to represent negative sentences in a quantum circuit. In addition, we propose two alternative data transformation methods - double angles and PCA - to make the data compatible with the QE-SVM classifier. The second contribution is being the first to apply QE-SVM to natural language processing tasks, specifically sentiment classification. We demonstrate that using QE-SVM with the appropriate representation leads to better predictive performance than SVM. Moreover, compared to the previous work [9], the proposed method outperforms the accuracy performance up to 93.33% using SPSA circuit parameter training and PCA with $n = 14$ data transformation. This work leads to the potential path for using a quantum kernel in NLP using quantum computing.

The following is the remainder of this paper. Section II covers sentiment analysis in quantum computing, the optimization method, the quantum kernel, and SVM in brief. Section III explains our proposed QE-SVM method. Section IV comprises the findings of our experiments as well as some discussions. The final section brings the paper to some conclusions.

II. RELATED WORKS

1) SENTIMENT ANALYSIS AND QUANTUM NLP

Sentiment analysis, one of the most developed fields of NLP, has been widely researched because of its significant use. One potential approach is to use Quantum Machine Learning. Several methods that try to imitate quantum mechanisms include [11], which examined sentiment analysis on Twitter data using a quantum-inspired representation model. This method uses quantum mechanisms to model semantic and sentiment information on a series of projectors in a probabilistic space. Then this method was developed into a quantum-like multimodal network (QMN). It combines quantum theory with long short-term memory (LSTM) networks for multimodal sentiment analysis on conversations [12]. Quantum algorithms in Variational Quantum Classifiers (VQC) can also be used to solve sentiment analysis problems, in which the work in [13]. carried out one of them using EfficientSU2 and RealAmplitudes, a built-in library from Qiskit quantum computer simulator. Although similar, this method outperforms the classification results of classical ML models.

One of the critical steps in the QNLP methodology is circuit parameter training after changing sentences into circuits. This learning process is carried out using a learning/optimization algorithm. One widely used method is the Simultaneous Perturbation Stochastic Approximation (SPSA) [14]. An essential feature of SPSA is the gradient approximation which requires only two measurements of the objective function regardless of the dimensions of the optimization problem. This feature significantly reduces optimization costs, especially in problems with many variables to optimize. Moreover, this method often outperforms other optimization methods, especially in variational quantum algorithms [15].

2) QUANTUM KERNEL AND OPTIMIZATION

To carry out the classification process, the kernel method for machine learning is one that is widely used. Among them is the Support Vector Machine (SVM) as the most well-known traditional learning method [16]. Combining the advantages of SVM with quantum computing, the authors of [6] proposed the concept of a quantum variational classifier that is run using a quantum variational circuit. Then they also proposed a quantum kernel estimator, which optimizes the SVM classifier by estimating the kernel function. The last method is the

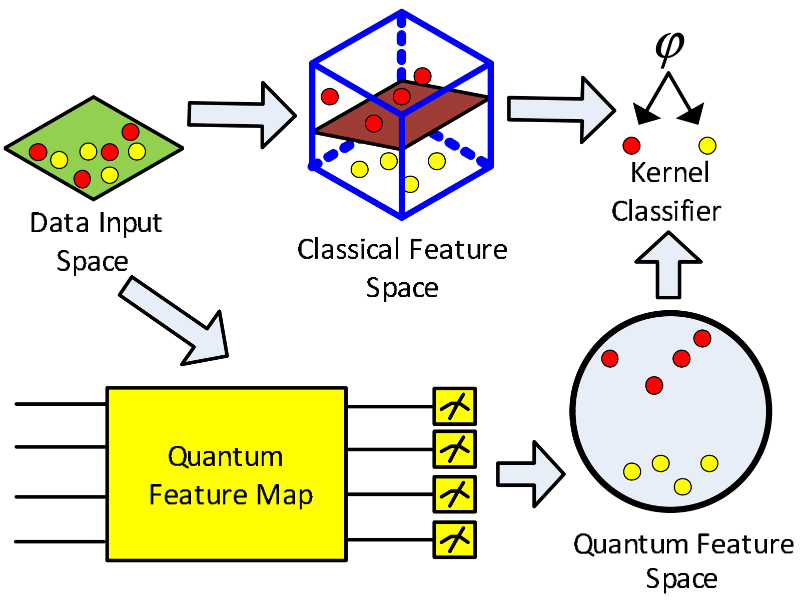

FIGURE 1. Illustration of comparison between quantum and classical kernel.

basis for developing the QSVM module/library on Qiskit so that this method is easily adapted by many parties.

The quantum kernel method utilizes a quantum feature space. Recalling that quantum states exist in Hilbert space [17], one can calculate the inner product between two quantum states. Theoretically, this can be achieved directly on a quantum circuit; the inner product between the state Ψ_1 represented by a set of unitaries U_1 and the state Ψ_2 represented by a set of unitaries U_2 can be calculated by applying the unitaries $U_2^\dagger U_1$ and observing the resulting state [18]. Alternatively, one can measure each state Ψ_1 and Ψ_2 and calculate the inner product classically. In both cases, the value of the inner product is used for further interpretation. Most commonly in machine learning, it is used to find the support vectors of a support vector classifier [6]. The motivation for using quantum kernels is that quantum feature maps are more difficult to calculate classically while potentially partitioning the data/input space in a more distinguishable manner [7].

The QSVM concept that was previously developed was then continued at the application level by [7], using the Noisy Intermediate-Scale Quantum (NISQ) assumption. In their work, the authors use quantum states built from quantum feature maps from structured data. Subsequently, the vector is handled by the quantum kernel to carry out the classification. The dataset used is three standard UCI datasets, namely wine, breast cancer, and handwritten digits, as well as two artificial numeric datasets.

One of the essential stages before the quantum kernel is data transformation which produces feature maps. This process can be done using special functions or rules, such as Principal Component Analysis (PCA) or double angles, or automatically using a learning algorithm. The last category was developed by [19], using a genetic algorithm to minimize circuit parameters. This approach was tested on structured data, namely the Parkinson's dataset, IoT irrigation, and drug classification.

III. PROPOSED METHOD

In this work, we design a sentiment classifier based on a quantum feature map. Figure 1 shows the illustration of the fundamental difference between classical feature maps and quantum feature maps. Basically, the classical feature space is formed by the classical values where the data points are represented by their original features before any kernel is applied. On the other hand, quantum feature space is formed by the quantum states. Thus, the quantum feature map is also formed by the quantum circuits as depicted in Figure 1.

To classify sentiment using quantum representation, we use an experimental QNLP pipeline similar to the one used in [10]. The pipeline involves converting sentences into circuits, optimizing them, and using the resulting circuits to classify sentiment using the QE-SVM method. We made some modifications to the pipeline, particularly during the optimization stage. The general pipeline stages used in this study include: (1) generating circuits from sentences, (2) training circuit parameters, (3) extracting state vectors from the circuits as sentence embeddings, and (4) using these embeddings to train the QE-SVM classifier and predict the sentiment of each sentence. This process is illustrated in Figure 2.

The sentiment classification task used in this study involved the restaurant sentiment dataset and required binary sentiment classification (positive or negative). In the circuit representation, each sentence type $'s'$ was mapped to 1 qubit. The conversion process from sentences to circuits was adapted from previous studies, e.g., [8], [20], and [21], with some modifications. The training process for

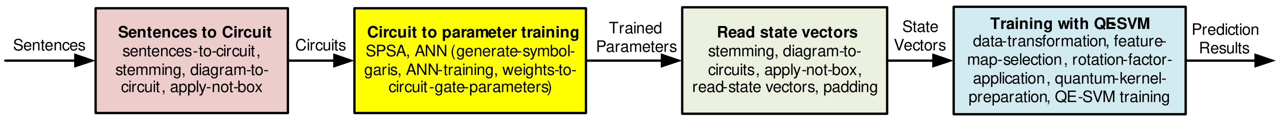
Sentences
Sentences to Circuit
sentences-to-circuit, stemming, diagram-to-circuit, apply-not-box
Circuits
Circuit to parameter training
SPSA, ANN (generate-symbol-garis, ANN-training, weights-to-circuit-gate-parameters)
Trained Parameters
Read state vectors
stemming, diagram-to-circuits, apply-not-box, read-state vectors, padding
State Vectors
Training with QESVM
data-transformation, feature-map-selection, rotation-factor-application, quantum-kernel-preparation, QE-SVM training
Prediction Results

FIGURE 2. General pipeline.

circuit parameters was conducted using SPSA. Each stage is explained further below.

A. SENTENCES-TO-CIRCUIT

1) **Sentences to circuits:** The process of converting sentences into circuits is performed using the Quantum Pipeline with Tket, which can be found in the Lambeq 0.1.2 documentation. First, the sentences are transformed into DisCoCat diagrams through the DepCCG Parser [22], with "not" words being ignored for the time being. A DisCoCat diagram is defined as a model of semantic words interactions in a sentence [22]. A complete explanation of a sentence representation using graphical language or the DisCoCat method is explained in [9]. Then, the diagrams are simplified by rewriting them with the Lambeq Rewrite package, which uses a set of transformation rules to change the strings or boxes of the diagram. The determiner, pre-adverb, and post-adverb rule is used in all experiments, while the auxiliary rule usage varies. The cups in the diagrams are removed using the bigraph method in the Lambeq 0.1.2 documentation. To remove the cups, some restructuring may be necessary, such as moving all the cups below all the word boxes and ordering them such that all the cups on the right of a cup are positioned above it. The algorithm for this conversion is provided in [9].
2) **Stemming:** To reduce the complexity of the words in the diagram, stemming is performed using NLTK's PorterStemmer [23] after the removal of the stop words.
3) **Diagrams to Circuits:** The next step involves converting the diagrams into quantum circuits using the IQP Ansatz, which is available in the Lambeq Ansatz package [10].
4) **Apply Not-Box settings to negative sentences:** To deal with negative sentences, we added the Not-Box settings to the circuit by using the Pauli-X gate on the output qubit since the Not-Box settings were previously removed during the parsing stage. The output qubit refers to the qubit that represents the resulting type 's' after grammatical types reduction. In situations where the output qubit is located in the middle, the Not-Box may be applied there instead. This approach was inspired by [24] which directly captures the negative meaning of negated words. The Pauli-X gate was selected as one of the representations since it flips the probability of measurement on a single qubit, resulting in the sentence's sentiment being flipped when applied to the output $'s'$ qubit. This decision was also supported by the findings of the Not-box experiment in our prior work [9].

B. CIRCUIT-TO-PARAMETER TRAINING

The quantum circuits are transformed into the objective function for SPSA Optimization [25], where the output qubit of type $'s'$ is compared with the sentiment label to determine the cost of the model. During the training process, the free variables of the circuits, which are the gate parameters, are optimized. Once the parameters are optimized, the circuit values are used to predict the sentiment of a sentence by measuring the output qubit.

C. READING STATE VECTORS

1) **Stemming and Diagrams to circuits:** The training process involves using the state vector values of each circuit. These state vectors are obtained from the word boxes of a sentence, which are extracted from a string diagram. The sentence's vector representation is the result of taking the tensor product of the state vectors of each word box in the sentence. In order to measure the states, circuits are created from diagrams that have not had their cups removed.
2) **Applying Not-Box settings to negative sentences:** To handle negative sentences containing the word "not", the Not-Box settings are applied based on the chosen representation.
3) **Reading state vectors and adding the padding:** The circuit parameters are initialized, and the state vector values are read. To ensure uniform feature size across all inputs, the state vectors are padded with the $|0\rangle$ state to fill quantum registers of the same size.

D. TRAINING WITH QE-SVM

Finally, these inputs are trained with QE-SVM using quantum kernel [9]. State vector data trained from the previous section need to be processed to be fit on a QE-SVM. To train the QE-SVM, the pipeline is described in the following section. Note that the output of the training is a feature that is defined as a state vector measurement after a quantum circuit optimizer training. In this case, we use SPSA or ANN as an optimizer.

IV. QE-SVM PIPELINE

This section explains the steps taken to fit the state vector data trained from the previous section on a QE-SVM. In general, the quantum kernel uses a quantum feature map to take data of n-dimension and maps it onto n corresponding qubits. The quantum kernel is then used by a classical SVM to approximate the separating hyperplane. Each data point is supplied to the SVM, where the values are used as the parameters of the feature map to be used as the inputs of the SVM.

A. DATA TRANSFORMATION

First, data is transformed from state vector data with high dimensions into one with lower dimensions. This is done in order to train the QE-SVM in a reasonable time. For this experiment, the data is transformed into 14 columns. The value 14 is chosen because the original state vector data is obtained from reading a quantum circuit with 7 qubits, where each qubit state has a $|0\rangle$ and a $|1\rangle$ component (i.e., in superposition).

This paper investigates two data transformation methods: the double angles method, and PCA with 14 principal components.

1) DOUBLE ANGLES

For a state vector composed of n qubits with states $[\alpha_0, \beta_0,], \ldots, [\alpha_n, \beta_n,]$, each state can be described by two angles θ_1 and θ_2. Given that qubit states are normalized $\alpha^2 + \beta^2 = 1$, we can calculate θ_1 as

$$\tan(\theta_1) = \frac{\beta}{\alpha} \tag{1}$$

$$\theta_1 = \tan^{-1}\left(\frac{\beta}{\alpha}\right) \tag{2}$$

On the other hand, because α and β are complex values, we can find the angle between them in complex space using the cosine rule. Therefore, we can calculate θ_2 as

$$\alpha.\beta = |\alpha||\beta|\cos(\theta_2) \tag{3}$$

$$\theta_2 = \cos\left(\frac{\alpha.\beta}{|\alpha||\beta|}\right) \tag{4}$$

These two angles are chosen because they describe the magnitude and similarity of each component, respectively. Furthermore, this decorrelates a majority of the high dimensional state vector data.

2) PCA (n=14)

Principal Component Analysis is used to obtain the principal components and project the input data onto lower dimensions. PCA takes the input data and projects it onto a set of orthogonal vectors, which describe a p-dimensional ellipsoid fitted onto the reference dataset. The coordinates are ordered such that the components have descending variance (where the projection of the data with the greatest variance is known as the first principal component and lies on the first coordinate, and so on).

PCA takes a data matrix X with n records and p fields (assuming the value of each column has been preprocessed such that the mean of each column is zero) and transforms it by a set of l weight vectors, each with dimension p onto a target vector space known as principal component scores, such that the set of scores t of a data entry has the maximum possible variance of X. It is noted that the weight vectors w have been normalized, and the cardinality of the set of weight vectors l is less than p such that the resulting transformation of X yields data with reduced dimensionality as follows [26]:

$$w_{(k)} = (w_1, \ldots, w_p)_{(k)} \tag{5}$$

$$t_{k(i)} = x_{(i)}.w_{(k)}. \tag{6}$$

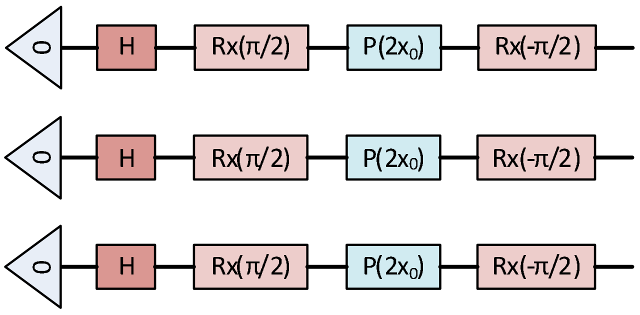

FIGURE 3. Pauli Y feature map circuit.

B. FEATURE MAP SELECTION

The experiments presented in this paper focus on three Pauli feature maps: Pauli Y, Pauli YY, and Pauli Y YY. The decision is inspired by the work in [7], which uses the Pauli Y, Pauli YY, Pauli Y YY and Pauli Z, Pauli ZZ, Pauli Z ZZ feature maps. Preliminary experimentation showed that the results of both the Y and Z counterparts yielded the same results, which is explained by the SVM only reading the real values of the quantum kernel output. Therefore for brevity, the methods listed will cover the Y counterparts of those three feature maps.

A feature map with Pauli Y rotation gates takes input data x and encodes it onto a quantum circuit by the following transformation. The general form can be written as [7]:

$$U_{\phi_Y(x)} = \exp\left(i\left(\sum_{j=1}^{n}\phi_S(x)\prod\sigma_{j\in\{Y\}}\right)\right) \tag{7}$$

The above gate encodes the transformation matrix $\sum_{j=1}^{n}\phi_S(x)\prod\sigma_Y$ as a set of Pauli Y rotations with input $\phi_S(x)$, where S denotes the connectivity between a subset of qubits in the quantum circuit, and $\phi_S(x)$ is x_0 when only a single qubit is concerned and is $\prod_{j\in S}\pi - x$ otherwise.

1) PAULI Y FEATURE MAP

The Pauli Y feature map is a simple feature map with a P gate between a $\pi/2$ X-rotation gate and its inverse. The result is a Y-rotation gate with angle x, which may be repeated multiple times. There is no entanglement in this feature map.

2) PAULI YY FEATURE MAP

The Pauli YY feature map is a second-order Pauli Y evolution circuit with Pauli Y and Pauli YY components. In the YY feature map, binary entanglement is introduced between all pairs of qubits in the circuit, with its input parameter corresponding to the index of qubit pair permutation. As with the

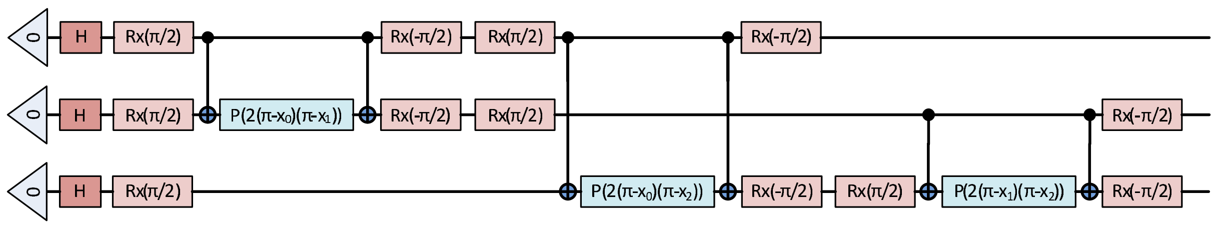

FIGURE 4. Pauli YY feature map circuit.

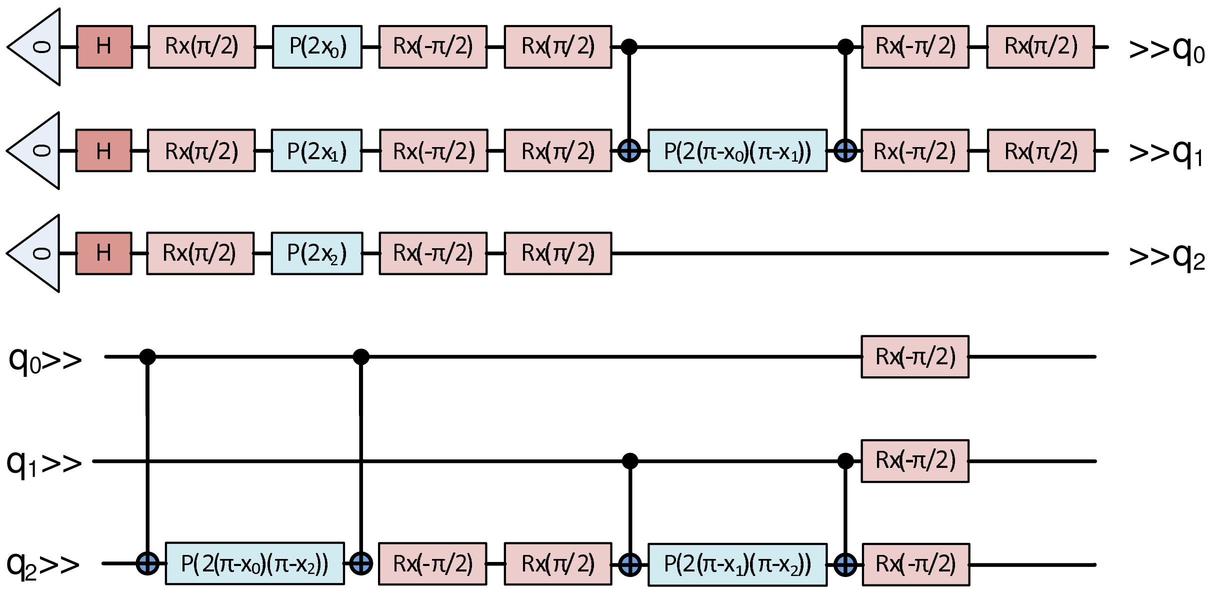

FIGURE 5. Pauli Y YY feature map circuit.

Pauli Y feature map, this Pauli YY circuit may be repeated multiple times.

3) PAULI Y YY FEATURE MAP

The Pauli Y YY feature map is a Pauli Y feature map followed by a Pauli YY feature map. This feature map starts out without entanglement, then has linear entanglement introduced by the second-order Pauli Y evolution circuit component. The Pauli Y YY circuit may be repeated multiple times.

4) PAULI Y Y YY FEATURE MAP

Following the construction pattern of the Pauli Y YY feature map, the Pauli Y Y YY feature map is a Pauli Y feature map, followed by another Pauli Y feature map, followed by a Pauli YY feature map. This feature map prepends an additional Pauli Y encoding circuit to the Pauli Y YY feature map. The Pauli Y Y YY circuit may be repeated multiple times.

C. ROTATION FACTOR APPLICATION

To handle overfitting, a rotation factor is applied to the rotation gate parameter angles $\phi_S(x)$, such that the values are multiplied by the scaling factor, modifying the feature map transformation into the following equation [7].

$$U_{\phi(x)} = \exp\left(i\left(\sum_{j=1}^{n} \alpha\phi_S(x)\prod \sigma_{j\in\{X,Y,Z\}}\right)\right) \quad (8)$$

The values chosen in this paper range from 0.5 to 2.0 with an increment of 0.1, as well as several other interesting values (0.75, 1.25, and 1.75).

1) QUANTUM KERNEL PREPARATION

This step prepares a Qiskit Quantum Instance from a Qiskit backend, then instantiates a Quantum Kernel with the chosen Pauli feature map. The Quantum Instance is a Qiskit object that contains a Qiskit Backend, as well as the configuration for circuit transpilation and execution. It is used to run the Quantum Kernel when called by the SVC during future steps. The Quantum Kernel is a Qiskit object that packages a quantum kernel function by transforming two sets of n-dimensional data, say x and y, onto higher dimensional data (typically of dimension $2n$) through the use of a quantum feature map which takes x as its input parameters, and calculates the dot-product between them. The dot-product result in matrix form can then be used in common machine-learning techniques:

$$K(x, y) = < f(x), f(y) > . \quad (9)$$

2) QE-SVM TRAINING

In this paper, we use the Scikit-learn SVC as the classical basis of the SVM. It takes the previously defined quantum kernel as a hyperparameter and the transformed training set as its input, then classically train them to obtain the separating hyperplane. The transformed testing set is then used to validate the results of the QE-SVM.

D. QE-SVM ALGORITHM IMPLEMENTATIONS

This subsection explains the implementations of the data transformations *i.e.,* double angles and PCA. Finally, the QESVM algorithm is described to predict the testing data.

Algorithm 1 Double Angles Data Trans. (DA)

```
Input: inputData: State vector decomposition of
       training/test data, nQubits: Number of qubits
       used in training/test circuits.
Output: outputData: Double angles of training/test
        data
begin
    DataFrame inputData
    DataFrame outputData
    integer nQubit
    complex $v_0, v_1, \alpha, \beta$
    float norm, zeroComponenet, oneComponent
    integer $N$ ← length of inputData
    statevector $sv$, complex vector of length N/2
    arrayFloat angles
    for $i = 1$ to $N$ do
        angles= $[0.0, \ldots, 0.0]$
        $sv \leftarrow$ inputData$_i$
        $v_0 \leftarrow sv_0$
        for $q = 1$ to nQubits do
            $v_1 \leftarrow sv$ with value $|00\ldots1\ldots00\rangle$, where
            1 at $q$-th position. Note that the index of
            $v_1 = j = 2^{nQubits-q}$.
            norm $\leftarrow \sqrt{(v_0 v_0^T + v_1 v_1^T)}$ for calculating $\alpha$,
            $\beta$, in $\alpha|0\rangle + \beta|1\rangle$.
            $\alpha \leftarrow \frac{v_0}{norm}$
            $\beta \leftarrow \frac{v_1}{norm}$
            angles$_{2q-1} \leftarrow \arctan|\alpha/\beta|$
            angles$_{2q} \leftarrow \arccos(\alpha\beta^T)/(||\alpha||||\beta||)$
        outputData$_i \leftarrow$ angles
    return outputData
```

Algorithm 1 shows the double angles data transformations. The function converts a state vector into double-angle data. For each qubit, we set the sv with the value of $|00\ldots1\ldots00\rangle$, where the q-th is 1. Next, the normalization of v_0 and v_1 is calculated to find the value of α and β. Finally, the double angles are calculated by $\arctan|\alpha/\beta|$ and $\arccos(\alpha\beta^T)/(||\alpha||||\beta||)$. The angle values are stored in the outputData with the corresponding index.

Algorithm 2 PCA Data Trans. (PCA)

```
Input: trainData: state vector decomposition of
       training data. testData: state vector
       decomposition of test data. nQubits: number of
       qubits used in training/test circuits.
Output: outputTrain: PCA of training data.
        outputTest: PCA of test data
begin
    DataFrame trainData
    DataFrame testData
    DataFrame outputTrain
    DataFrame outputTest
    integer nQubit
    initialize TransformerPCA(nQubit)
    fit TransformerPCA(nQubit)
    outputTrain ← TransformerPCA(trainData)
    outputTest ← TransformerPCA(testData)
    return (outputTrain, outputTest)
```

Next, another data transformation is PCA (Algorithm 2). The PCA is formed by the state vector. First, the initialization of the PCA transformer is conducted. The PCA uses a set of *nQubit* weight vectors, where each weight inherits the dimension of X. In this case, X represents a 7-qubit state vector and contains 128 elements. It uses the weights to map each record x in X to the resulting scores t such that t maintains a maximum variance of X. The outputs of the PCA transformation are stored in outputTrain and outputTest.

Finally, Algorithm 3 shows the main QE-SVM procedure. The transformed data (by Double Angles or PCA) is the input of the QE-SVM. First, we apply the rotation factor as expressed in Equation (8) by changing the value of α. Next, we create the feature map with the following properties; type of feature, number of qubits (nQubit), repetitions, and entanglement. After that, we run the simulation using the quantum kernel with n shots (nShots) and stored the result in adhocKernel. Finally, we run the QE-SVM classifier and run the training and testing predictions and accuracy performances. Finally, we have the results by the variables trainAccuracy and testAccuracy.

V. EXPERIMENT AND RESULTS

The experiments in this paper explore and compare three circuit-parameter-training methods and two data transformation methods for identifying the sentiment of a sentence using QE-SVM. These experiments are executed by implementing each step in the general pipeline in Figure 2. The purpose of our experiments can be described as follows:

1) To study the effect of using two data transformation and circuit parameter training methods on the prediction result

TABLE 1. Data transformation and circuit parameter training experiment result.

No.	Circuit Parameter Training Method	Data Transformation	Feature Map	Rotation Factor	Repetitions	Training acc.	Testing acc.	Fit time	Classification time
1	SPSA	Double angles	Pauli Y	0.9	1	86.47%	88.33%	7 min	5 min
2			Pauli Y	0.8	1	87.06%	86.67%	7 min	5 min
3			Pauli Y	1	1	88.24%	83.33%	7 min	5 min
4		PCA (n=14)	Pauli Y Y YY	0.9	1	93.53%	93.33%	20 min	14 min
5			Pauli Y YY	0.9	1	94.12%	90.00%	17 min	12 min
6			Pauli Y Y YY	0.8	1	92.94%	90.00%	20 min	14 min
7	ANN 1	Double angles	Pauli Y	0.5	3	80.00%	85.00%	19 min	14 min
8			Pauli Y	1	1	80.59%	85.00%	7 min	5 min
9			Pauli Y	1	2	81.18%	85.00%	13 min	9 min
10		PCA (n=14)	Pauli Y	2	2	85.29%	80.00%	13 min	9 min
11			Pauli Y	2	3	87.01%	78.33%	19 min	13 min
12			Pauli Y	1	1	78.82%	76.67%	7 min	5 min
13	ANN 2	Double angles	Pauli Y	1	3	79.41%	80.00%	20 min	14 min
14			Pauli Y	1	1	78.82%	76.67%	7 min	5 min
15			Pauli Y YY	0.5	1	84.71%	76.67%	17 min	11 min
16		PCA (n=14)	Pauli Y	1	3	91.18%	85.00%	20 min	15 min
17			Pauli Y	2	3	92.35%	85.00%	20 min	14 min
18			Pauli Y	2	1	90.00%	83.33%	8 min	6 min

*Entanglement = Linear; No of Shots = 16

Algorithm 3 QE-SVM

```
Input: DataFrame trainData, DataFrame testData,
       arrayInteger trainLabels, arrayInteger testLabels,
       integer nQubits, dictionary trainingConfig.
Output: arrayInteger trainPredictions, arrayInteger
        testPredictions, float trainAccuracy, float
        testAccuracy.
begin
    1) Apply rotation factor by ApplyRotationFactor
       (trainData, testData)
    2) Create feature map by FeatureMap(Pauli, nQubits,
       Repetitions, Entanglement)
    3) Run quantum simulation and kernel by
       QuantumSimulationAndKernel (nShots, FeatureMap)
       stored in adhocKernel
    4) Initialize SVM from the QESVM;
       qesvmClassifier(adhocKernel, scaledTrainData)
    5) Call training prediction by
       qesvmClassifier.predict(scaledTrainData) → 
       trainPredictions
    6) Call testing prediction by
       qesvmClassifier.predict(scaledTestData) →
       testPredictions
    return (trainAccuracy, testAccuracy)
```

2) To study the impact of ANN architecture for circuit parameter training on the prediction result
3) To study the effect of rotation factor on the prediction result
4) To perform prediction comparison with SVM baseline

First, the experimental hardware used is a Linux OS with 8 vCPUs and 52 GB of RAM with VM type n1-highmem-8. The hardware is the same as our preliminary work in [9]. The dataset used in the experiment is a collection of simple subjective sentences in the restaurant domain. These sentences are generated from 29 vocabularies, consisting of positive and negative sentences, with each sentence having a length of 4-5 words. This dataset is divided into 170 training, 50 development, and 60 test sentences.

A. DATA TRANSFORMATION AND CIRCUIT PARAMETER TRAINING EXPERIMENTS

This experiment was conducted to determine which combination of methods is most appropriate to improve the sentiment classification results in QE-SVM. The combination of methods is done on: circuit parameter training, data transformation method, feature map, and rotation factor. For the circuit parameter training method, we used three options: SPSA, ANN 1 (3 layers), and ANN 2 (5 layers). We use two alternatives for the data transformation method, namely Double angles and PCA (n=14). As for the feature map, we use four methods: Pauli Y, Pauli YY, Pauli Y YY, and Pauli Y Y YY. The parameter rotation factor varies from 0.5 to 1, and repetitions from 1 to 3, whereas the fixed parameters are using linear entanglement with 16 shots. To simplify the result's presentation, only the three best results for the combination of circuit parameter training method and representation are shown (Table 1). Based on the experimental results, it can be seen that the combination of the SPSA optimization method with the PCA transformation method, the Pauli Y Y YY feature map, and a rotation factor of 0.9 gives the best accuracy results of 93.33%.

B. ANN LAYER EXPERIMENTS

The ANN layer experiment was carried out to determine the effect of the number of ANN layers on the sentiment classification results. The experimental parameters used are the data transformation method, feature map, and rotation factor for ANN architectures: ANN 1 and ANN 2. The number of layers of the two architectures is 3 and 5, respectively. Based on the experimental results in Table 3 and Figure 6, ANN 1 gives better accuracy than ANN 2. The best configuration is

TABLE 2. ANN configurations.

Type	Layer Configurations
ANN1	input: 60, hidden: 2, output:60
ANN2	input: 60, hidden: 60, hidden:60, hidden: 2, output: 60

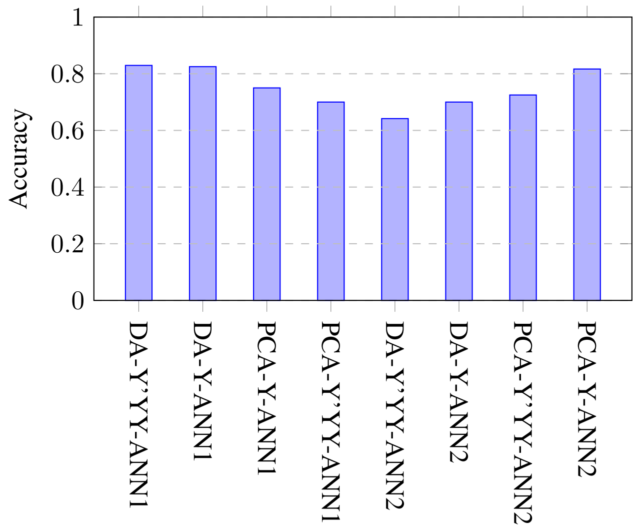

FIGURE 6. Summary of ANN layer experiment.

ANN 1 with Double angles, Pauli Y, rotation factor of 1, and repetition of 1. The detailed configuration is shown in Table 2.

C. ROTATION FACTOR EXPERIMENT

We also conducted rotation factor experiments to find the best values for classification in QE-SVM. We use the PCA transformation method and the best SPSA circuit parameter training method. The experimental results in Table 4 and Figure 7 show that the rotation factor value that gives the best classification results is 0.9, with an accuracy value of 93.33%.

D. COMPARISON WITH BASELINE: SVM

The baseline method used as a comparison in this experiment is (Classical) SVM. In addition, we use the circuit parameter training method SPSA, ANN 1, and ANN 2, as well as the data transformation method Double angles and PCA. As can be inferred from Table 5 and Figure 8, the best accuracy for the baseline method was obtained at 80.00% using ANN 1-Double angles. Meanwhile, for the proposed method, the best accuracy was obtained at 93.33%, and the highest increase was 16.66% using SPSA-PCA. Moreover, compared to the previous work [9], the proposed method outperforms the accuracy performance up to 93.33% using SPSA circuit parameter training and PCA with $n = 14$ data transformation.

E. CASE STUDY

To better understand the sentiment classification capability of our model, we evaluate some of the best models and analyze several cases. In the first evaluation, we compared the three models representing the best circuit parameter training method and representation. The followings are the three models used and their parameter configurations:

1) Model 1: circuit parameter training method SPSA - representation PCA - feature map Pauli Y Y YY - rotation factor 0.9 - repetitions 1
2) Model 2: circuit parameter training method ANN 1 - representation Double angles - feature map Pauli Y - rotation factor 1 - repetitions 2
3) Model 3: circuit parameter training method ANN 2 - representation PCA - feature map Pauli Y - rotation factor 2 - repetitions 3

We take examples of several sentences with representations of sentence types and their properties, along with their prediction results for the three models (Table 6). For example, in negative sentences with the negation word (not) (sentence 3), model 2 (ANN 1) failed to predict correctly. Whereas in negative sentences without negation (sentence 1), model 3 (ANN 2) failed to predict correctly. In positive sentences (sentence 2), model 2 and model 3 fail to predict. On the other hand, model 1 succeeded in predicting all three types of sentences. This fact shows that the SPSA-PCA combination on QE-SVM provides the best performance for various types of positive and negative sentences. However, the

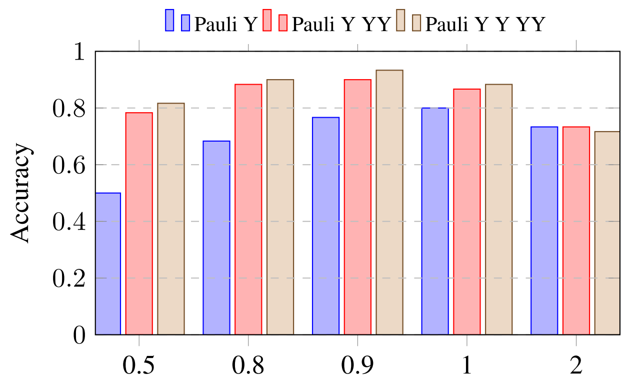

FIGURE 7. Rotation factor parameter.

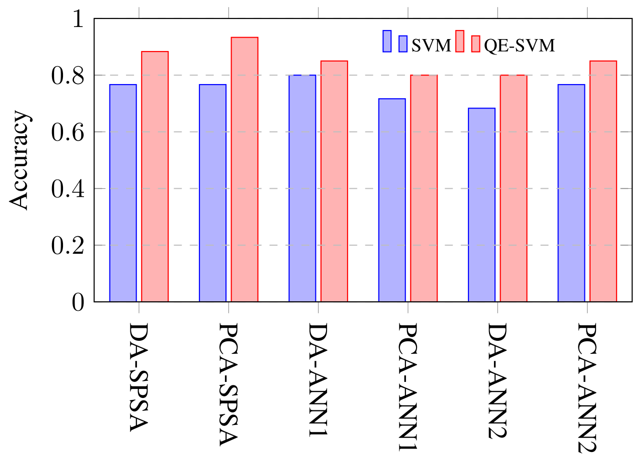

FIGURE 8. CPT and representation method.

TABLE 3. ANN layer experiment result.

No.	Data Transformation	Circuit Parameter Training Method	Feature Map	Rotation Factor	Repetitions	Training Acc	Testing Acc	Fit Time	Classification Time
1	Double angles	ANN 1	Pauli Y	0.5	1	78.82%	80.00%	7 min	5 min
2				1	1	80.59%	**85.00%**	7 min	5 min
3			Pauli Y YY	0.5	1	82.35%	80.00%	17 min	12 min
4				1	1	83.53%	73.33%	17 min	12 min
5		ANN 2	Pauli Y	0.5	1	74.71%	66.67%	7 min	5 min
6				1	1	78.82%	73.33%	7 min	5 min
7			Pauli Y YY	0.5	1	81.76%	65.00%	17 min	12 min
8				1	1	85.29%	63.33%	17 min	12 min
9	PCA (n=14)	ANN 1	Pauli Y	1	1	78.82%	76.67%	7 min	5 min
10				2	1	84.71%	73.33%	7 min	5 min
11			Pauli Y YY	1	1	88.82%	71.67%	17 min	12 min
12				2	1	93.53%	68.33%	17 min	12 min
13		ANN 2	Pauli Y	1	1	87.06%	80.00%	8 min	6 min
14				2	1	90.00%	83.33%	8 min	6 min
15			Pauli Y YY	1	1	91.76%	75.00%	17 min	12 min
16				2	1	95.29%	70.00%	17 min	12 min

TABLE 4. Rotation factor experiment result.

No.	Data Transformation	Circuit Parameter Training Method	Feature Map	Rotation Factor	Repetitions	Training acc.	Testing acc.	Fit time	Classification time
1	PCA (n=14)	SPSA	Pauli Y	0.5	1	63.53%	50.00%	7 min	5 min
2				0.8	1	82.94%	68.33%	7 min	5 min
3				0.9	1	85.29%	76.67%	7 min	5 min
4				1	1	87.05%	80.00%	7 min	5 min
5				2	1	84.71%	73.33%	7 min	5 min
6			Pauli Y YY	0.5	1	80.00%	78.33%	17 min	12 min
7				0.8	1	92.35%	88.33%	17 min	12 min
8				0.9	1	94.12%	90.00%	17 min	12 min
9				1	1	93.53%	86.67%	17 min	12 min
10				2	1	97.65%	73.33%	17 min	12 min
11			Pauli Y Y YY	0.5	1	91.76%	81.67%	20 min	14 min
12				0.8	1	92.94%	90.00%	20 min	14 min
13				**0.9**	**1**	**93.53%**	**93.33%**	**20 min**	**14 min**
14				1	1	93.53%	88.33%	20 min	14 min
15				2	1	96.47%	71.67%	20 min	14 min

TABLE 5. Comparison with SVM baseline.

No.	Circuit Parameter Training	Data Transformation	SVM		QE-SVM	
			Training acc	Testing acc	Training acc	Testing acc
1	SPSA	Double angles	77.65%	76.67%	86.47% (+8.82%)	88.33% (+11.66%)
2		PCA (n=14)	87.65%	76.67%	93.53% (+5.88%)	**93.33% (+16.66%)**
3	ANN 1	Double angles	74.12%	**80.00%**	81.18% (+7.06%)	85% (+5.00%)
4		PCA (n=14)	78.82%	71.67%	85.29% (+6.47%)	80% (+8.33%)
5	ANN 2	Double angles	67.06%	68.33%	79.41% (+12.35%)	80% (+11.67%)
6		PCA (n=14)	77.65%	76.67%	91.18% (+13.53%)	85% (+8.33%)

TABLE 6. Comparison of QE-SVM prediction results with three variations of the circuit parameter training method.

No	Sentence	Label	Model 1	Model 2	Model 3
1	I hated the bland food	0	0	0	$\underline{1}$
2	The restaurant had fast service	1	1	$\underline{0}$	$\underline{0}$
3	The service was not nice	0	0	$\underline{1}$	0

*1 = positive, 0 = negative

combination of ANN with PCA or Double Angles still needs to be optimal in handling positive and negative sentences.

In the second evaluation (Table 5), we tried comparing sentences that could be handled by our method (QE-SVM) to the baseline method (SVM). The followings are the three models used and their parameter configurations:

1) Baseline 1: circuit parameter training method ANN 1 - representation Double angles - classifier SVM

TABLE 7. Comparison of the predicted results of our method and the baselines.

No.	Sentence	Label	Baseline 1	Baseline 2	Ours
Case: a1					
1	The restaurant was bad	0	$\underline{1}$	$\underline{0}$	$\underline{0}$
2	The restaurant had fast service	1	$\underline{0}$	$\underline{1}$	1
3	The meal was not good	0	$\underline{1}$	0	0
Case: a2					
4	I loved the food	1	1	$\underline{0}$	1
5	The restaurant was not nice	0	$\underline{1}$	$\underline{1}$	0
6	The restaurant was not bad	1	1	$\underline{0}$	1
Case: b1					
7	The service was awful	0	0	$\underline{1}$	$\underline{1}$
8	The staff was not great	0	0	$\underline{1}$	$\underline{1}$
Case: b2					
9	The meal was great	1	$\underline{0}$	$\underline{0}$	$\underline{0}$
10	The restaurant had unappetizing meals	0	$\underline{1}$	$\underline{1}$	$\underline{1}$

*1 = positive, 0 = negative

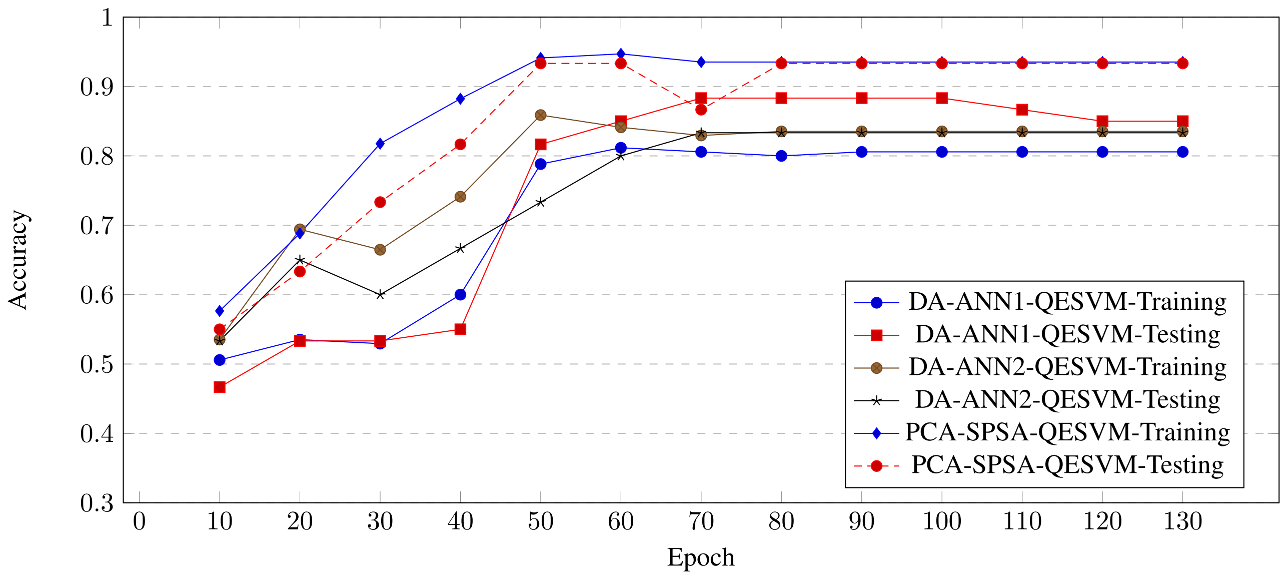

FIGURE 9. Comparisons of several methods of QE-SVM.

2) Baseline 2: circuit parameter training method SPSA - representation PCA - classifier SVM
3) Our method: circuit parameter training method SPSA - representation PCA - classifier QE-SVM

We found some cases when comparing the three models (Table 7).

1) The prediction is correct in our method but wrong in baseline 1 or baseline 2 (cases a1 and a2). Observation of the prediction results shows that baseline 1 is often mistaken as a false positive rather than a false negative. From the test set prediction results, baseline 1 has a minimal tendency to predict positively compared to our method. As additional information, the false positive rate of baseline 1 is 34.62%, while our method's false positive rate is 10.71%, baseline 1 accuracy of 80%, and our method's accuracy of 93.33%). In the case of baseline 2, there is no particular tendency to predict negative or positive, with predicted negative = 31 and predicted positive = 29. Therefore, it cannot be concluded that the model tends to predict positively or negatively, and prediction errors occur due to a lack of models in other aspects. By transforming the state vector, QE-SVM provides more accurate prediction results for both positive and negative sentences. Moreover, classical SVM is unsuitable for transformed data (double angles/PCA) and performs better before transformation because the data is more descriptive and no information is lost. Classical SVM can handle high-dimensional state-vector data because it does not need to simulate a quantum kernel. The dot product between two high-dimensional vectors is only $O(n)$. However, its overall performance is still below QE-SVM.
2) The prediction was wrong in our method but correct in baseline 1 or baseline 2 (Case b1 and b2). The case where our method is wrong and baseline 1 or baseline 2 is correct occurs when the label is negative and best incorrectly predicts it as positive. It is

conjectured that baseline 1 happens to be accurate, given the slight tendency to predict negatively. In addition, our method has a wrong prediction on the positive sentence (9th sentence). This fact is presumably due to the similarity of the sentence with one of the sentences in the train set.

Lastly, the comparison among several methods of QE-SVM in terms of accuracy with respect to epoch is depicted in Figure 9. As shown in the figure for the higher epochs, the combination of PCA and SPSA yields the highest accuracy for both training and testing. This is due to the state vector information being well represented and optimized by the PCA and the SPSA. On the other hand, double angles data (DA) may eliminate some information. For comparison, the PCA (n=14) approximates the distribution of 128-dimensional data with 14 values, whereas the double angle represents 7-dimensional data with only 2 values (the angle and the amplitude).

VI. CONCLUSION

This paper described a study on the implementation of QE-SVM on the NLP task: sentiment classification. The subjective sentence, which contains sentiment value was analyzed by transforming it into a quantum representation that can be used as input to the quantum kernel. The experimental results proved that the combination of sentences-to-circuit steps, the SPSA optimization method, and the data transformation method PCA on QE-SVM provided the best sentiment classification results of 93.33% accuracy, with an increase of 16.6% compared to the baseline SVM. This approach worked both in positive and negative subjective sentences. Our work leads to a potential path for using quantum kernels for NLP using quantum computing.

In the future research, we suggest further development by using Variational Quantum Algorithms and by implementing them on a quantum computer.

ACKNOWLEDGMENT

An earlier version of this paper was presented at the 2022 IEEE International Conference on Quantum Computing and Engineering (QCE), Broomfield, CO, USA [DOI: 10.1109/QCE53715.2022.00025].

REFERENCES

[1] B. Liu, *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. Cambridge, U.K.: Cambridge Univ. Press, 2020.

[2] V. Vyas and V. Uma, "Approaches to sentiment analysis on product reviews," in *Sentiment Analysis and Knowledge Discovery in Contemporary Business*. Hershey, PA, USA: IGI Global, 2019, pp. 15–30.

[3] S. Unankard, X. Li, M. Sharaf, J. Zhong, and X. Li, "Predicting elections from social networks based on sub-event detection and sentiment analysis," in *Proc. 15th Int. Conf. Web Inf. Syst. Eng. (WISE)*, Thessaloniki, Greece. Cham, Switzerland: Springer, Oct. 2014, pp. 1–16.

[4] W. Duan, Q. Cao, Y. Yu, and S. Levy, "Mining online user-generated content: Using sentiment analysis technique to study hotel service quality," in *Proc. 46th Hawaii Int. Conf. Syst. Sci.*, Jan. 2013, pp. 3119–3128.

[5] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[6] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, Mar. 2019.

[7] J.-E. Park, B. Quanz, S. Wood, H. Higgins, and R. Harishankar, "Practical application improvement to quantum SVM: Theory to practice," 2020, *arXiv:2012.07725*.

[8] W. Zeng and B. Coecke, "Quantum algorithms for compositional natural language processing," 2016, *arXiv:1608.01406*.

[9] F. Z. Ruskanda, M. R. Abiwardani, M. A. Al Bari, K. A. Bagaspati, R. Mulyawan, I. Syafalni, and H. T. Larasati, "Quantum representation for sentiment classification," in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, Sep. 2022, pp. 67–78.

[10] D. Kartsaklis, I. Fan, R. Yeung, A. Pearson, R. Lorenz, A. Toumi, G. de Felice, K. Meichanetzidis, S. Clark, and B. Coecke, "Lambeq: An efficient high-level Python library for quantum NLP," 2021, *arXiv:2110.04236*.

[11] Y. Zhang, D. Song, P. Zhang, X. Li, and P. Wang, "A quantum-inspired sentiment representation model for Twitter sentiment analysis," *Appl. Intell.*, vol. 49, no. 8, pp. 3093–3108, 2019.

[12] Y. Zhang, D. Song, X. Li, P. Zhang, P. Wang, L. Rong, G. Yu, and B. Wang, "A quantum-like multimodal network framework for modeling interaction dynamics in multiparty conversational sentiment analysis," *Inf. Fusion*, vol. 62, pp. 14–31, Oct. 2020.

[13] N. Joshi, P. Katyayan, and S. A. Ahmed, "Comparing classical ML models with quantum ML models with parametrized circuits for sentiment analysis task," *J. Phys., Conf. Ser.*, vol. 1854, no. 1, Apr. 2021, Art. no. 012032.

[14] A. Liu, X. Deng, Z. Tong, Y. Luo, and B. Liu, "A simultaneous perturbation stochastic approximation enhanced teaching-learning based optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2016, pp. 3186–3192.

[15] X. Bonet-Monroig, H. Wang, D. Vermetten, B. Senjean, C. Moussa, T. Bäck, V. Dunjko, and T. E. O'Brien, "Performance comparison of optimization methods on variational quantum algorithms," 2021, *arXiv:2111.13454*.

[16] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[17] M. Schuld and N. Killoran, "Quantum machine learning in feature Hilbert spaces," *Phys. Rev. Lett.*, vol. 122, no. 4, Feb. 2019, Art. no. 040504.

[18] M. Schuld, "Supervised quantum machine learning models are kernel methods," 2021, *arXiv:2101.11020*.

[19] S. Altares-López, A. Ribeiro, and J. J. García-Ripoll, "Automatic design of quantum feature maps," *Quantum Sci. Technol.*, vol. 6, no. 4, Oct. 2021, Art. no. 045015.

[20] K. Meichanetzidis, A. Toumi, G. de Felice, and B. Coecke, "Grammar-aware sentence classification on quantum computers," 2020, *arXiv:2012.03756*.

[21] R. Lorenz, A. Pearson, K. Meichanetzidis, D. Kartsaklis, and B. Coecke, "QNLP in practice: Running compositional models of meaning on a quantum computer," 2021, *arXiv:2102.12846*.

[22] R. Yeung and D. Kartsaklis, "A CCG-based version of the DisCoCat framework," 2021, *arXiv:2105.07720*.

[23] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 40, no. 3, pp. 211–218, Jul. 2006.

[24] B. Coecke, M. Sadrzadeh, and S. Clark, "Mathematical foundations for a compositional distributional model of meaning," 2010, *arXiv:1003.4394*.

[25] J. C. Spall, "An overview of the simultaneous perturbation method for efficient optimization," *Johns Hopkins APL Tech. Dig.*, vol. 19, no. 4, pp. 482–492, 1998.

[26] I. T. Jolliffe, *Principal Component Analysis*. Aberdeen, U.K.: Univ. of Aberdeen, 2002.

FARISKA ZAKHRALATIVA RUSKANDA (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the School of Electrical Engineering and Informatics, Bandung Institute of Technology, Bandung, Indonesia. She is currently an Assistant Professor of natural language processing with the Informatics Research Group, Bandung Institute of Technology.

MUHAMMAD RIFAT ABIWARDANI received the B.Sc. degree (cum laude) in informatics from Institut Teknologi Bandung, Indonesia, in 2023. His current research interests include quantum machine learning and quantum NLP.

RAHMAT MULYAWAN (Member, IEEE) received the B.Eng. degree in electrical engineering from ITB, Indonesia, in 2008, and the M.Sc. degree in electrical engineering from TU Delft, The Netherlands, in 2011. He is currently a member of the Microelectronics Centre, ITB. His current research interests include intelligent signal processing, MIMO systems, and transceiver design for optical wireless communications.

INFALL SYAFALNI (Member, IEEE) received the B.Eng. degree in electrical engineering from Institut Teknologi Bandung (ITB), Bandung, Indonesia, in 2008, the M.Sc. degree in electronic engineering from the University of Science Malaysia (USM), Penang, Malaysia, in 2011, and the Dr.Eng. degree in engineering from the Kyushu Institute of Technology (KIT), Iizuka, Fukuoka, Japan, in 2014. From 2014 to 2015, he held a research position with KIT. From 2015 to 2018, he was an ASIC Engineer with the ASIC Development Group, Logic Research Company Ltd., Fukuoka. In 2019, he joined ITB, where he is currently an Assistant Professor with the School of Electrical Engineering and Informatics and a Researcher with the University Center of Excellence on Microelectronics, ITB. His current research interests include logic synthesis, logic design, VLSI design, efficient circuits, and algorithms.

HARASHTA TATIMMA LARASATI (Member, IEEE) received the B.S. and M.S. degrees in telecommunication engineering from Institut Teknologi Bandung (ITB), Bandung, Indonesia, in 2016 and 2017, respectively. She is currently pursuing the Ph.D. degree in computer engineering with Pusan National University, Busan, Republic of Korea. She is a Junior Lecturer with ITB. Her current research interests include quantum computing and cryptanalysis, quantum machine learning, AI security, and networking.