Enhancing Qubit Readout with Autoencoders

Piero Luchi,[1,2,*] Paolo E. Trevisanutto,[3] Alessandro Roggero,[1,2] Jonathan L. DuBois,[4] Yaniv J. Rosen,[4] Francesco Turro,[1,2] Valentina Amitrano,[1,2] and Francesco Pederiva[1,2]

[1] *Dipartimento di Fisica, University of Trento, via Sommarive 14, Povo, Trento I–38123, Italy*
[2] *National Institute for Nuclear Physics (INFN) Trento Institute of Fundamental Physics and Applications (TIFPA), Trento, Italy*
[3] *Scientific Computing Department, Science and Technology Facilities Council (STFC) – UK Research and Innovation (UKRI), Rutherford Appleton Laboratory, Didcot OX11 0QX, United Kingdom*
[4] *Lawrence Livermore National Laboratory, P.O. Box 808, L-414, Livermore, California 94551, USA*

(Received 5 December 2022; revised 17 April 2023; accepted 28 June 2023; published 20 July 2023)

In addition to the need for stable and precisely controllable qubits, quantum computers take advantage of good readout schemes. Superconducting qubit states can be inferred from the readout signal transmitted through a dispersively coupled resonator. This work proposes a readout classification method for superconducting qubits based on a neural network pretrained with an autoencoder approach. A neural network is pretrained with qubit readout signals as autoencoders in order to extract relevant features from the data set. Afterward, the pretrained-network inner-layer values are used to perform a classification of the inputs in a supervised manner. We demonstrate that this method can enhance classification performance, particularly for short- and long-time measurements where more traditional methods present inferior performance.

DOI: 10.1103/PhysRevApplied.20.014045

I. INTRODUCTION

The construction of a computer exploiting quantum—rather than classical—principles represents a formidable scientific and technological challenge. Nowadays, superconducting quantum processors are reaching outstanding results in simulation [1–4] and computational power [5]. However, building a fault-tolerant quantum processor still presents many technical challenges. First, the ability is required to generate high-fidelity gates, exploiting both hardware (e.g., improving the manufacturing process and the design of available qubits [6–8]) and software improvements (e.g., designing precise optimal control protocols [9–11]). Second, one needs the ability to perform a complete quantum error-correction protocol [12–14]. Finally, it is of primary importance to have a high-fidelity qubit readout measurement to extract information from the device, especially for observables that are very sensitive to it (for an extreme case of this, see, e.g., Ref. [15]). In addition to a careful design of the system parameters [16,17] or improvement in fabrication processes extending the qubit coherence time [6,18], the readout fidelity can be enhanced through the use of machine-learning techniques.

The currently most common qubit readout technique is the dispersive readout [in quantum electrodynamics (QED) circuit architecture], which couples the qubit to a readout resonator. In this approach, the state of the qubit is determined by measuring the phase and amplitudes of an electromagnetic field transmitted through the resonator [19–22]. Hardware, random thermal noise, gate error, or qubit decay processes that occur during measurements may reduce the readout fidelity. Machine-learning techniques and classification schemes could help to restore a good fidelity by improving the classification precision of the signal to the correct state of the qubit. The Gaussian mixture model (GMM) [23] is the most commonly used classification method, given its ease of use. It exploits parametric modeling of the probability distribution of the averaged readout signals in terms of a sum of Gaussians to perform a classification of each measurement. In Refs. [24–27], the authors have developed and implemented various classification methods based on neural networks trained on the full dynamics of the measurement, instead of on their averages, obtaining good results. Another approach is the hidden Markov model proposed in Ref. [28], which allows for a detailed classification of the measurement results and detection of the decay processes that the qubit could undergo during the measurement. These schemes help to improve the accuracy of the classification of the qubit readout measurements.

In this work, we propose a semiunsupervised machine-learning classification method based on autoencoder

[*] piero.luchi@unitn.it

pretraining applied to the heterodyne readout signal of a superconducting qubit [19,29]. Autoencoders are a type of artificial neural network designed to encode a set of data efficiently by learning how to regenerate them from a synthetic encoded representation [30,31]. The encoding process automatically isolates the most relevant and representative features of the input data set, i.e., those features that allow for the most faithful reconstruction of input data while neglecting noise and irrelevant details [32,33]. Hence, the main idea of this work is to exploit this characteristic of autoencoders and perform the data classification not on the readout signals or on their time average but on their encoded representation produced by autoencoder training. The model consists of two sections. The first is composed of an autoencoder trained to reconstruct the data set of the qubit readout signals. The second section is a two-layer feed-forward neural network trained to classify the encoded representation of the measurement signals. We demonstrate that this method can enhance the state classification of readout signals—especially for short readout times, where other more traditional methods have worse performance—and, in general, shows a more stable performance for a broad range of measurement-time lengths. We remark on the fact that the most significant improvement occurs with a combination of hardware and software improvements, as obtained by the authors in Ref. [34]. In this paper, the focus is only on software improvement on present machines.

The paper is organized as follows. In Sec. I, the qubit setup and readout and a review of machine-learning models of interest, as well as our proposed method, are presented. In Sec. II, the method is tested on two study cases, based on real data, and the classification results together with considerations of the applicability of the method are presented. Finally, in Sec. III, conclusions are drawn.

II. METHODS

A. Qubit readout

We consider a transmon-type qubit coupled to a detuned resonator (i.e., a quantum harmonic oscillator) in the context of a strong projective dispersive measurement scheme [20,21]. Our device is a qutrit with frequency $\omega_{01} = 2\pi \times 3.44$ GHz, anharmonicity $\alpha = -2\pi \times 208$ MHz, and a Rabi frequency of 5 MHz. It has a relaxation time $T_1 = 220$ µs and a coherence time $T_2 = 20$ µs. The device is based on the work by Place *et al.* [18]. A 100-ns constant π pulse of frequency ω_{01} is used to rotate the qutrit from $|0\rangle$ to $|1\rangle$ and another 100-ns π pulse is used to rotate the qutrit from $|1\rangle$ to $|2\rangle$. The coupling energy between the qubit and the cavity (before the dispersive approximation) is $2\pi \times 107$ MHz. The cavity has a frequency $\omega_r = 2\pi \times 7.24$ GHz. The measurement pulse is a constant pulse with ω_r frequency of arbitrary duration. A Purcell filter is present in the device.

Due to the qubit interaction, the readout resonator undergoes a frequency shift, the value of which depends on the qubit state. This dependency can be exploited to perform measurements of the qubit state in the dispersive regime, i.e., when the detuning of the qubit and resonator is large relative to their mutual coupling strength [35]. Once the resonator is irradiated with a specific microwave pulse, the registered transmitted signal will incorporate different amplitude and phase shifts based on the state of the qubit. The demodulation procedure can extract such information from the signal, discriminating between qubit states.

Our setup consists of a superconducting qubit controlled by the quantum orchestration platform (QOP) programming environment (QM Technologies Ltd.) through the QUA programming language based on PYTHON [36]. In this setup, the measurement pulse is sent into the readout resonator. In interacting with the system, this signal is modulated by the response of the resonator. The output signal is then filtered, amplified, and down-converted to an intermediate frequency $\omega_{\mathrm{IF}} = \omega_r - \omega_{\mathrm{LO}}$ through a signal mixer, where ω_{LO} is the frequency of the local oscillator (an electronic component needed by the mixer to change signal frequency). Finally, it has to be demodulated to extract information about the qubit state that the readout signal acquires in the interaction.

Formally, the demodulation is an integral of the signal multiplied by a sinusoidal function:

$$I = \frac{2}{T_m}\int_0^{T_m} r(\tau)\cos(\omega_{\mathrm{IF}}\tau)d\tau \tag{1}$$

$$Q = -\frac{2}{T_m}\int_0^{T_m} r(\tau)\sin(\omega_{\mathrm{IF}}\tau)d\tau, \tag{2}$$

where the readout signal is denoted by $r(\tau)$ and T_m is the integration time.

In the usual approach, complete demodulation is performed by integrating over time intervals T_m, obtaining a single value for the I and Q components for each qubit readout signal. In this way, each measurement can be represented as a point in the I-Q plane. Due to the state-dependent frequency shift of the qubit, these points will accumulate in different zones of the I-Q plane. An example is displayed in Fig. 1(b), where the points for a three-level qutrit are reported.

However, an alternative approach can be employed, the so-called *sliced demodulation*, which consists of dividing the time interval $[0, T_m]$ into N subintervals and performing the demodulation separately on each chunk of the signal, namely:

$$I(t) = \frac{2}{\Delta t}\int_t^{t+\Delta t} r(\tau)\cos(\omega_{\mathrm{IF}}\tau)d\tau \tag{3}$$

$$Q(t) = -\frac{2}{\Delta t}\int_t^{t+\Delta t} r(\tau)\sin(\omega_{\mathrm{IF}}\tau)d\tau, \tag{4}$$

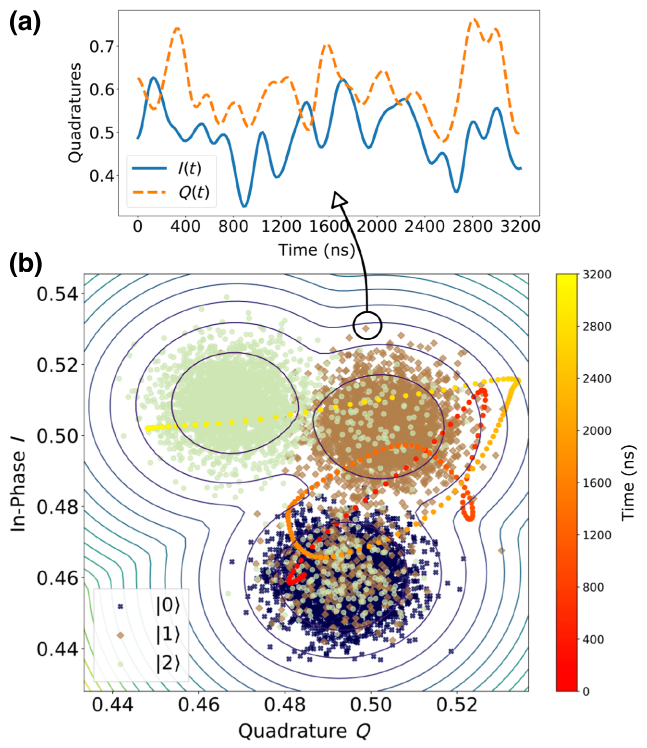

FIG. 1. A pictorial representation of the qubit readout data. (a) An example of the in-phase, $I(t)$, and quadrature, $Q(t)$, components of the heterodyned signal of a single shot obtained via sliced demodulation (as described in Sec. II A). The average of these signals is a single point in the I-Q plane below. (b) An example of the whole data set. Each point is the time average of a measurement represented in the I-Q plane for qubit states 0,1, and 2. The lines represent the two-dimensional (2D) Gaussian contour plot (see Sec. II B 1) for the three-Gaussian distribution. The dotted red-yellow line is an example of a measurement signal represented in the I-Q plane. The colors represent the time evolution (in nanoseconds).

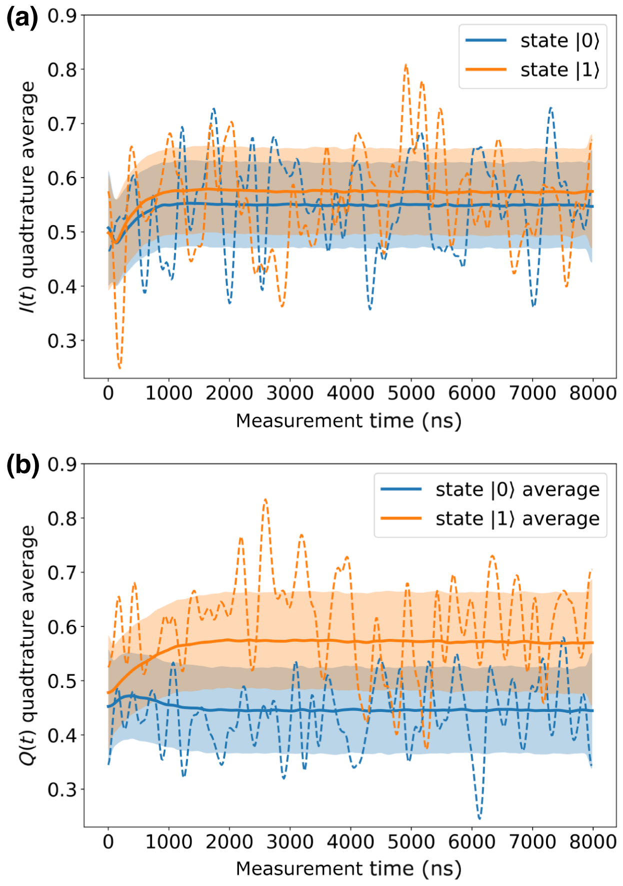

FIG. 2. The average readout trajectories for state |0⟩ and |1⟩ in both quadratures. The solid lines represent the mean of all trajectories in the data set for state |0⟩ (blue) and state |1⟩ (orange). The shaded regions represent the standard deviation of the average for each time step. The dashed line represents an example of a single trajectory. (a) In-phase, $I(t)$, signals. (b) Quadrature, $Q(t)$, signals.

where $\Delta t = T_m/C$ is the subinterval length and C is the number of intervals. In this way, we obtain two time series, $I(t)$ and $Q(t)$, for each measurement. In Fig. 1(a), the $I(t)$ and $Q(t)$ signals of a single readout signal are represented as examples. Averaging these signals, we obtain a single point in the I-Q plane as represented in Fig. 1(b). The red-yellow line in Fig. 1(b) represents the $I(t)$ and $Q(t)$ signals plotted together as a trajectory (state-path trajectory). The color gradient represents time. In Fig. 2, on the other hand, the average signals $\langle I(t)\rangle$ and $\langle Q(t)\rangle$ (solid lines) are reported together with the standard deviation for each time step (the shaded range). The same graph also shows individual readout signals (dashed lines). As can be seen, the noise of the machine is high, as the standard-deviation zones overlap heavily. However, one of the aims of this work is to show how the proposed method can deal with this noise and, in any case, improve the classification of the measures.

In principle, sliced demodulation should retain information that is otherwise lost in the averaging process of complete demodulation. This information is exploited in this work to increase the state-detection accuracy. Usually, in full demodulation, the readout accuracy is adjusted and maximized by tuning the readout length, i.e., the demodulation integration time T_m. The aim is to obtain clouds of points (as in Fig. 1) with a distribution that is as Gaussian as possible, in order to use the GMM to perform the classification (see. Sec. II B). In fact, short integration times produce poorly distinguishable states, while for long times, the qubit states tend to decay during the measurement, which produces a non-Gaussian data distribution and, again, a low classification accuracy. In contrast to full demodulation, sliced demodulation retains more information about the qubit-state measurements and, in principle, allows for increased accuracy of the state classification. Moreover, as is observed in this work, it reduces the dependence of the classification result on T_m, since the data do not need to be Gaussian distributed.

B. Machine-learning models

We briefly review the three machine-learning algorithms used in this work.

1. Gaussian mixture model

Gaussian mixture models (GMMs) approximate the distribution of data [in this case, the clouds of mean demodulation points in Fig. 1(b)] as a weighted superposition of Gaussian distributions [23]. The GMM models the distribution by adjusting the Gaussian parameters through a maximum-likelihood estimation over the data set of points in the I-Q plane. A new point is attributed to one of the classes based on the probability that it belongs to one of the three Gaussians of the GMM.

2. Feed-forward neural network

Feed-forward neural networks (FFNNs) are the simplest class of neural networks. Trained over a labeled data set, they are capable of classifying new inputs. Formally, the neural network implements a closed-form parametrized function, N_ϕ, which maps input in $\mathcal{X} \subseteq \mathbf{R}^m$ into a space $\mathcal{Y} \subseteq \mathbf{R}^n$ that encodes in some way the information on the classes into which the inputs are divided. The inputs are the full qubit readout signals. An optimal classification of data is obtained by adjusting the parameters ϕ making use of optimization algorithms. This is obtained by minimizing some type of loss function l between the correct label $\mathbf{y}^i$ of input $\mathbf{x}^i$ and the neural-network-predicted label $\hat{\mathbf{y}}_i = N_\phi(\mathbf{x}^i)$, namely:

$$\min_\theta \sum_i l\left(\mathbf{y}^i, N_\phi(\mathbf{x}^i)\right) . \tag{5}$$

This optimization is commonly carried out by making use of the well-known *back-propagation* algorithm [30,37].

3. Autoencoders

Autoencoders are neural networks designed to learn, via unsupervised learning procedures, efficient encoding of data [31,38,39]. This encoding is achieved by adjusting the weights and biases of the network to regenerate the input data. It is composed of a first part, the encoder, which learns to map the input data into a lower-dimensional representation (the latent space), ignoring insignificant features or noise, and a second part, the decoder, that, conversely, is trained to reconstruct the original input from the low-dimensional encoding in the latent space. Autoencoders perform dimensionality reduction and feature learning.

Mathematically, the autoencoder is a model composed of two closed-form parametrized functions, the encoder f_{θ^e} and the decoder g_{θ^d}. The parameters $\theta = [\theta^e, \theta^d]$ need to be optimized to perform the correct input reconstruction. These functions are defined as follows:

$$f_{\theta^e} : \mathcal{X} \to \mathcal{L},$$
$$g_{\theta^d} : \mathcal{L} \to \mathcal{X} .$$

The function f_{θ^e} takes an input $\mathbf{x}^i \in \mathcal{X} \subseteq \mathbf{R}^m$ from the data set $\{\mathbf{x}^1, \mathbf{x}^2, \ldots\}$ and maps it into the feature vector $\mathbf{h}^i \in \mathcal{L} \subseteq \mathbf{R}^p$ with $p < m$, i.e., $\mathbf{h}^i = f_{\theta^e}(\mathbf{x}^i)$. Conversely, the decoder function, g_{θ^d}, maps the feature vector $\mathbf{h}^i$ back into the input space, giving a reconstruction $\tilde{\mathbf{x}}^i$ of the input $\mathbf{x}^i$.

The parameters θ of the autoencoder are optimized such that the model minimizes the reconstruction error $l(\mathbf{x}, \tilde{\mathbf{x}})$, i.e., a measure of the discrepancy of the reconstructed input from the original one. The general minimization problem is, therefore,

$$\min_\theta \sum_i l\left(\mathbf{x}^i, g_{\theta^d}(f_{\theta^e}(\mathbf{x}^i))\right) . \tag{6}$$

Again, this is optimized using the already mentioned *back-propagation* algorithm [30,37].

C. Model: Neural network with autoencoder-type pretraining

In this work, we propose a classification model based on a neural network with an autoencoder pretraining, which we denote "*PreTraNN*." It is composed of two sections.

The first section consists of an encoder f_{θ^e} the parameters θ^e of which are pretrained in advance as an autoencoder over the input data set. The encoder consists of two layers with L_1 and L_2 neurons and a third layer, the latent layer, with L_H neurons. The decoder g_{θ^d} necessary for the pretraining has the same structure as the encoder but is in reverse order. Given a input of dimension d, we always set $L_1 = 3/(4)d$, $L_2 = 2/(4)d$ and $L_H = 1/(4)d$. The activation functions are *sigmoid* for the first layer of the encoder (and the last layer of the decoder) and the *tanh* function for all the internal layers. The choice of internal layer size is explained in Appendix A 1, while the complete specifications of the autoencoder are reported in Appendix B.

The second section is a feed-forward neural network, N_ϕ, dependent on a set of parameters ϕ, which works as a classifier taking as inputs the feature vector of the encoder and, as outputs, the exact labels of the readout signals. It is composed of two hidden layers with L_{N_1} and L_{N_2} neurons, respectively, and an output layer with a number of neurons equal to the number of data classes. Given the dimension of the input, d, we set $L_{N_1} = 2d$ and $L_{N_2} = d$. The activation functions are *tanh* for the internal layers and *softmax* for the last layer, commonly employed for classification purposes.

The assignment of the label $\mathbf{y}^i$ to a qubit readout signal $\mathbf{x}^i(t)$ works as follows:

(1) The discrete signal $\mathbf{x}^i$ is flattened by stacking the I and Q components in a single one-dimensional (1D) vector, i.e., $\mathbf{X}^i = [\mathbf{x}^i_I, \mathbf{x}^i_Q]$, so that it can be plugged into the neural network.

(2) The input $\mathbf{X}^i$ is transformed in the feature vector $\mathbf{h}^i$ via the encoder function, i.e., $\mathbf{h}^i = f_{\theta^e}(\mathbf{X}^i)$.

(3) The feature vector $\mathbf{h}^i$ is plugged into the feed-forward neural network N_ϕ to be assigned to one out of the three classes. Formally, $N_\phi(\mathbf{h}^i) = \hat{\mathbf{y}}^i$, where $\hat{\mathbf{y}}^i$ is the predicted label for the input $\mathbf{X}^i$.

A pictorial representation of the PreTraNN classification working principle is displayed in Fig. 3.

1. Training

The training is performed separately for the two sections that comprise the PreTraNN model.

The autoencoder is trained first. The data set is composed by inputs $\mathbf{x}^i$ with $i = 1, 2, \ldots, M$, representing the 2D trajectories in the I-Q plane. The neural-network architecture requires a 1D vector input, so the $\mathbf{x}^i$ need to be flattened, stacking the I and the Q components in a single 1D vector. Therefore, we compose a new data set of $\mathbf{X}^i = [\mathbf{x}^i_I\ \mathbf{x}^i_Q]$. The parameters $\theta = [\theta^e, \theta^d]$ of the autoencoder $A_\theta(\mathbf{x}^i) = g_{\theta^d}(f_{\theta^e}(\mathbf{X}^i))$ are trained by minimizing Eq. (6), where we choose as loss function l, the mean-square error (mse):

$$l = \frac{1}{d}\sum_{t=1}^{d}\left(X^i[t] - \hat{X}^i[t]\right)^2, \tag{7}$$

in which d is the length of the input data $\mathbf{X}^i$ and $\hat{\mathbf{X}}^i = A_\theta(\mathbf{X}^i)$ is the reconstructed input.

In a second step, the neural network N_ϕ is trained taking as input the feature vectors $\mathbf{h}^i$ of the encoder f_{θ^e} and, as output, the real labels $\mathbf{y}^i$ of the corresponding $\mathbf{x}^i(t)$. The optimal network parameters ϕ are obtained by minimizing Eq. (5), where the loss function l is chosen to be the cross-entropy loss function, which is widely used in classification.

A pictorial explanation of the PreTraNN training procedure is depicted in Fig. 4, while a complete specification of the autoencoder structure is reported in Appendix B.

Note that although we use the term "pretraining," PreTraNN is not a pretrained model in the general sense. We do not use a bulk neural network pretrained on a vast quantity of data, attaching to it new layers that are then trained on our specific classification problem. In the PreTraNN model, we take as the "pretrained neural network" the encoder part of an autoencoder that has previously been trained over our specific readout data.

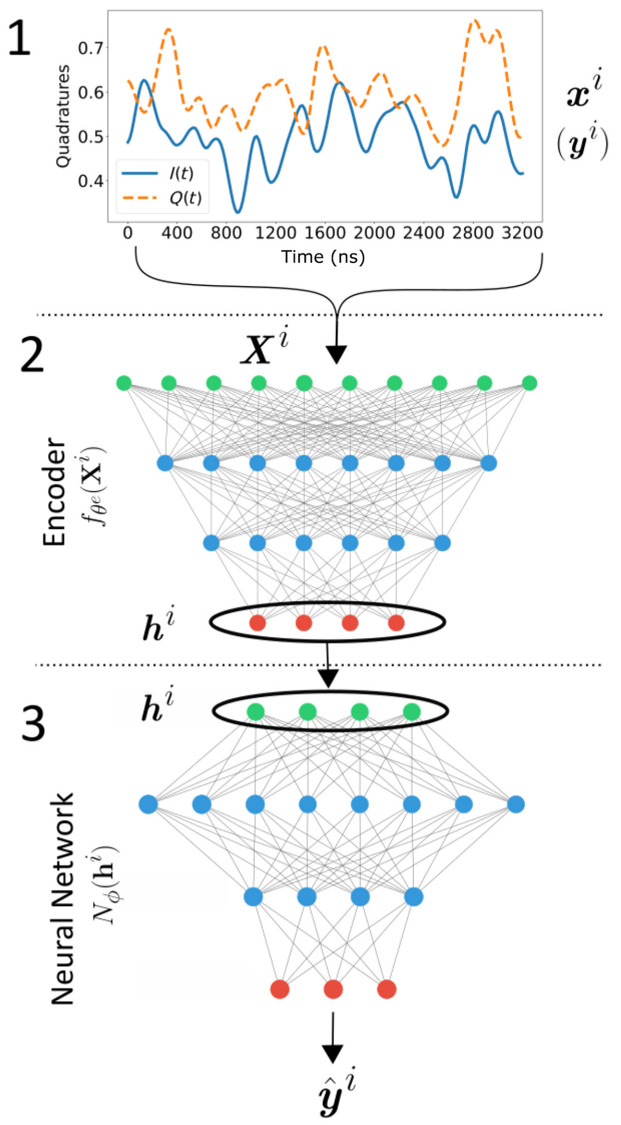

FIG. 3. A pictorial representation of the working principle and the architecture of the PreTraNN method described in Sec. II C. Section 1: an example of the measurement signal $\mathbf{x}(t)$ that we want to classify with PreTraNN. Section 2: the input $\mathbf{x}(t)^i$ is flattened to obtain $\mathbf{X}^i$, plugged into the encoder, previously trained as an autoencoder, and transformed into its encoded representation $\mathbf{h}^i$. Section 3: the latent layer of the encoder, $\mathbf{h}^i$, is passed into a feed-forward neural network trained to assign the label $\hat{\mathbf{y}}^i$.

D. Benchmark methods

We compare the result of the proposed PreTraNN model with two state-of-the-art methods introduced above: the GMM and a simple FFNN.

The GMM is trained directly on I-Q points, averages of the readout signal.

The FFNN is, instead, trained over the readout-signal data set, taking as input the flattened vectors $\mathbf{X}^i = [\mathbf{x}^i_I, \mathbf{x}^i_Q]$ and, as output, their labels $\mathbf{y}^i$. The architecture of the FFNN consists of two inner layers of dimension $L_{FF_1} = 2d$ and

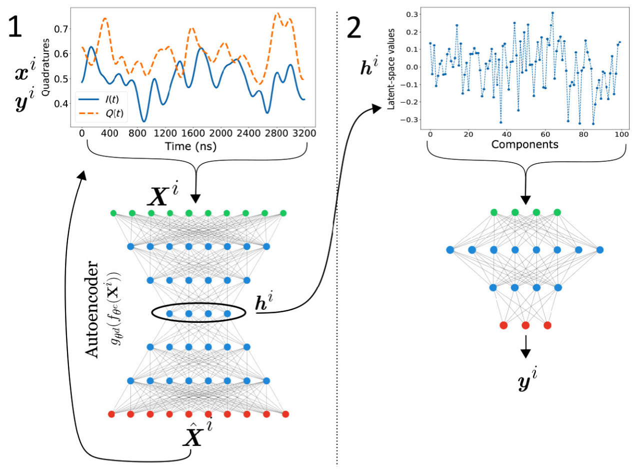

FIG. 4. A pictorial representation of the PreTraNN training described in Sec. II C. Section 1: the autoencoder is trained to reconstruct the measurement signals—this should train the network to extract the relevant features from each temporal chunk. Section 2: after the training, the decoder part of the network is removed and the encoded representation of data (represented in the plot at the top right) is used as the training input data set for the second section of the PreTraNN model, which is trained to classify them into the correct class $\mathbf{y}^i$.

$L_{FF_2} = d$, where d is the input dimension, and an output layer. The activation functions are *tanh* for the internal layer and *softmax* for the output layer. The structure of the FFNN is the same as for the second section of the PreTraNN model. The only difference is that while the PreTraNN neural network takes as input the readout signal encoded in the latent space, the $\mathbf{h}^i$ vector, the FFNN takes the signals $\mathbf{X}^i$ directly.

E. Metrics

To measure the accuracy of the classification systems, we utilize the “classification accuracy,” i.e., the probability that each signal is attributed to the correct label (i.e., the correct state of the qubit). This classification is obtained as a percentage of correctly attributed signals out of their total number (for each state). The global accuracy is the average of the accuracies of each state.

F. Data sets

As already mentioned, two versions of the same data set are used in this work. They are now more clearly defined.

We collect heterodyned readout signals for each qubit state. Each measurement is obtained by preparing the device in a certain state (e.g., $|0\rangle$ or $|1\rangle$) and then by measuring it immediately, storing the obtained signals. The selection of the time windows Δt for sliced demodulation requires careful consideration. The demodulation time step Δt should span an integer number of periods of the readout signal to avoid imprecise demodulation. The frequency of the readout signal is $\omega_{\rm IF} = 60$ MHz, so its period is $1/\omega_{\rm IF} \approx 16$ ns. For this reason, in this work, we take a time window $\Delta t = 16$ ns. Hence, each readout signal $\mathbf{x}^i(t)$ has a point every 16 ns. The length of the measurement, T_m, is also an essential parameter. Here, we choose to consider measurements of increasing length starting from 800 ns up to 8000 ns, corresponding to discrete signals the number of elements of which spans from 50 to 500, to study the efficiency of the classification methods in different configurations. The collection of $I(t)$ and $Q(t)$ signals are then smoothed with a window-smoothing algorithm, with a Hanning window of 50-time-step length to remove some noise.

Each measurement is, therefore, a 2D $\mathbf{x}^i(t) = [I^i(t), Q^i(t)]$ trajectory that, flattened to form the $\mathbf{X}^i$ inputs (see Sec. II C), will form the data set for PreTraNN and the FFNN. The data set for the GMM, on the other hand, is obtained by time averaging each $\mathbf{x}^i(t)$ measurement so as to obtain two values that can be represented in I-Q space [an example of which is shown in Fig. 1(b)]. The data set is then shuffled and split into train and test data sets in a 75%:25% proportion. The size of the data set impacts the accuracy of the method and needs some consideration to avoid under-fitting or unnecessarily long training times. Such considerations are made in Appendix A 2.

It should be mentioned that the data preparation is not error free. In fact, it may happen that the expected state ($|0\rangle$,$|1\rangle$ or $|2\rangle$) is not actually prepared due to control errors or environmental coupling. The $|0\rangle$ state is initialized with an active reset procedure. This procedure works by performing a short measurement on the qubit and applying a π pulse to it if the state is $|1\rangle$ or $|2\rangle$ to push it back to the ground state. However, there will be a residual thermal population to deal with. An estimate of this quantity, for the two-state case, is given by the confusion matrices in Fig. 8, which quantify the percentage of $|1\rangle$-labeled measurements that are actually $|0\rangle$. Due to these errors, the classification will not be 100% accurate even with the model proposed in this paper, because the data set suffers from this inaccuracy.

We also emphasize that the readout data are all from the same device. Although it may be interesting to study a multidevice classification system, in general, different devices may show differences in the average behavior of the readout trajectories, due to design and control differences. This obviously makes training more challenging and could bias the results.

III. RESULTS

The purpose of this work is to demonstrate how the feature-extraction capability of the autoencoder helps to improve the effectiveness of qubit readout: hence, specifically, how the PreTraNN method performs better than other commonly used methods for readout, namely the GMM and a simple FFNN. In this section, PreTraNN and the benchmark methods are compared in terms of classification accuracy and their overall performance is studied.

In addition, to deepen the analysis, the application of the models is extended to two readout configurations. The first is the readout of the usual two-level qubit and the second is the readout of a three-level qutrit. This analysis will give an idea of the good scalability of PreTraNN for multiple-level readout.

A. Two-state qubit readout

In this case, the qubit is prepared and immediately measured in states $|0\rangle$ and $|1\rangle$. The data set consists of 16 000 readout signals (8000 for each state) and it is split into training and test subsets in a 75%:25% proportion. Considerations on the choice of the data set are made in Appendix A 1. The PreTraNN, FFNN, and GMM setup is the one defined in Secs. II C and II E.

1. Classification accuracy

We start by showing our results for the classification accuracy of the three methods for increasing measurement length T_m to compare their performance in different cases. All experiments are carried out in the configuration defined in Sec. II and every experiment is computed 10 times and averaged. We report the state-classification accuracy for each state separately in Fig. 5 and the global classification between states $|0\rangle$ and $|1\rangle$ in Fig. 6.

We start by considering Fig. 5. In Fig. 5(a), the classification accuracy of the $|0\rangle$ state for the three models as a function of measurement length is shown, while in Fig. 5(b), the same information is reported but for the $|1\rangle$ state. First, it can be noted that, for short measurements, the performance of all the models deteriorates. This behavior should be attributed to the fact that, for short measurement times, the data distributions overlap heavily, preventing all methods, and especially the GMM, from fitting them appropriately with two Gaussians (for an illustrative example, see Fig. 7). On the other hand, for middle and long measurement times, the GMM performs, respectively, better and worse for state $|0\rangle$ and state $|1\rangle$ than the other two methods. Moreover, the state-$|0\rangle$ classification accuracy remains high and stable for long measurements, while that of state $|1\rangle$ presents a descending trend at longer times. This behavior has a simple explanation: the qubit excited state (e.g., the $|1\rangle$ state) has leakage to the ground state (the $|0\rangle$ state) at a much higher rate than the opposite direction. As a consequence, there is an asymmetry in the data-point distributions. This results in states prepared as $|1\rangle$ being spotted in the state-$|0\rangle$ distribution due to the decay process, while the reverse is much more unlikely. Therefore the GMM, which fits the distribution with two Gaussians,

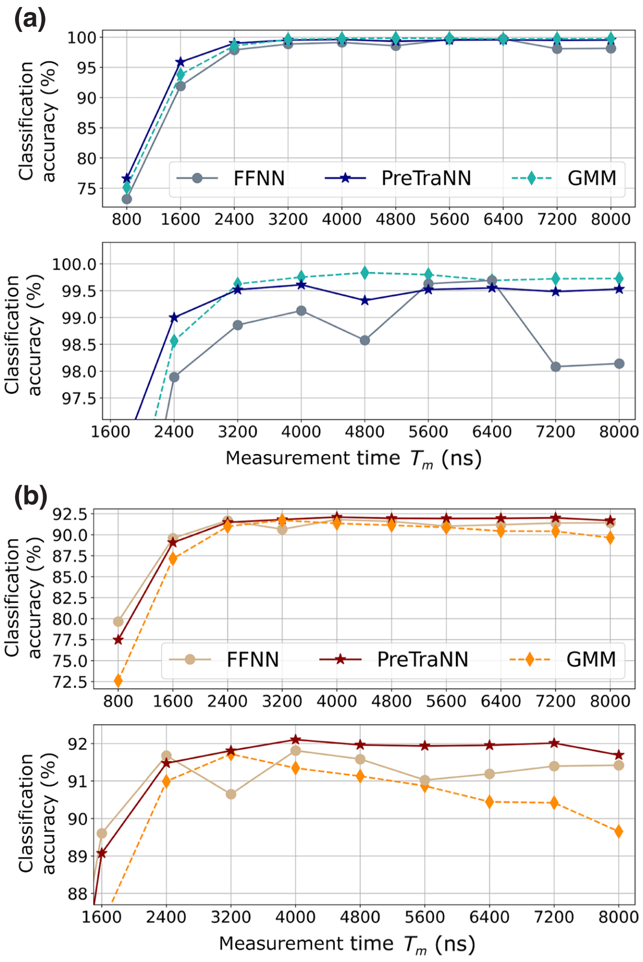

FIG. 5. A comparison of the classification accuracy, for states $|0\rangle$ and $|1\rangle$ separately, among the GMM, the simple FFNN, and the PreTraNN methods. The readout time T_m spans from 800 ns to 8000 ns. (a) Upper panel: the classification accuracy for state $|0\rangle$ by the three methods as a function of the measurement time. Lower panel: an enlargement of the (2400–8000)-ns part of the plot. (b) Upper panel: the classification accuracy for state $|1\rangle$ of the three methods as a function of the measurement time. Lower panel: an enlargement of the (2400–8000)-ns part of the plot.

cannot handle this asymmetry and performs very differently in the two cases. The number of signals decaying during the measurement procedure increases with the measurement time and, in fact, the accuracy of state $|1\rangle$ drops for long times. However, the FFNN has a fluctuating trend and it often performs worse than the GMM. We can speculate that this behavior derives from the fact that, for very large inputs, the training is more difficult and a simple FFNN does not converge adequately. This suggests that the FFNN is not completely adequate for this purpose. On the contrary, the PreTraNN method shows very stable behavior for both states, even for long measurement times. It not only uses all the "history" of the measurements but also exploits the feature extraction of the autoencoder.

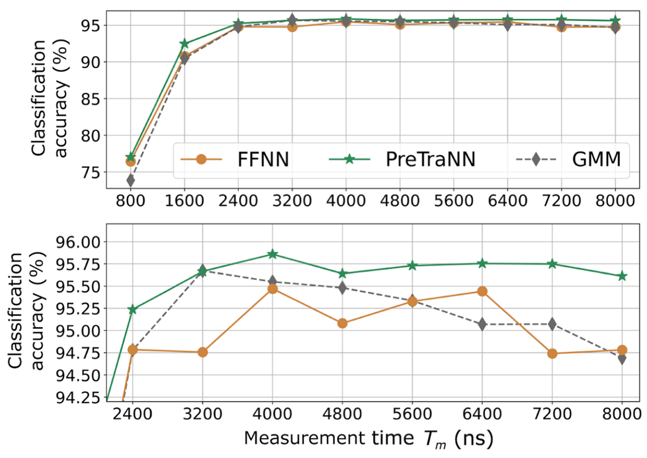

FIG. 6. The global-classification accuracy between states $|0\rangle$ and $|1\rangle$ for increasing measurement time T_m. The accuracy obtained with the PreTraNN method is higher (or at most equal) to those obtained with the GMM and the FFNN.

The global-discrimination accuracy between states $|0\rangle$ and $|1\rangle$ is reported in Fig. 6. It is obtained by averaging the accuracies of the $|0\rangle$ and $|1\rangle$ states. In this global case, the PreTraNN method outperforms the GMM and FFNN methods for every measurement time (except for a measurement time of 3200 ns, where the accuracies of the GMM and PreTraNN coincide). The considerations of the previous case also apply here.

It can also be noted that the GMM accuracy has a global maximum at 3200 ns. As mentioned before, for the GMM to work well, the distribution of I-Q points for each qubit state must be as "Gaussian" and distinguishable as possible. It happens that, for short measurement times, the point distributions overlap, since the qubit-resonator response is still in a transient state, while, for long times, decay processes come into play, which makes the distribution skewed. Therefore, we can deduce that the length of 3200 ns produces the least-overlapping distributions that allow the GMM to reach the greatest accuracy. This measurement time is therefore the one that should be set for the readout in the event of use of the GMM. The PreTraNN method makes the need for this adjustment less strict, since it works well for a larger interval of the experimental parameters T_m. In general, it can be seen that, in the PreTraNN method, the classification accuracy is only increasing or constant. As a consequence, the trimming is faster and easier, since the need to find the maximum accuracy is obviated.

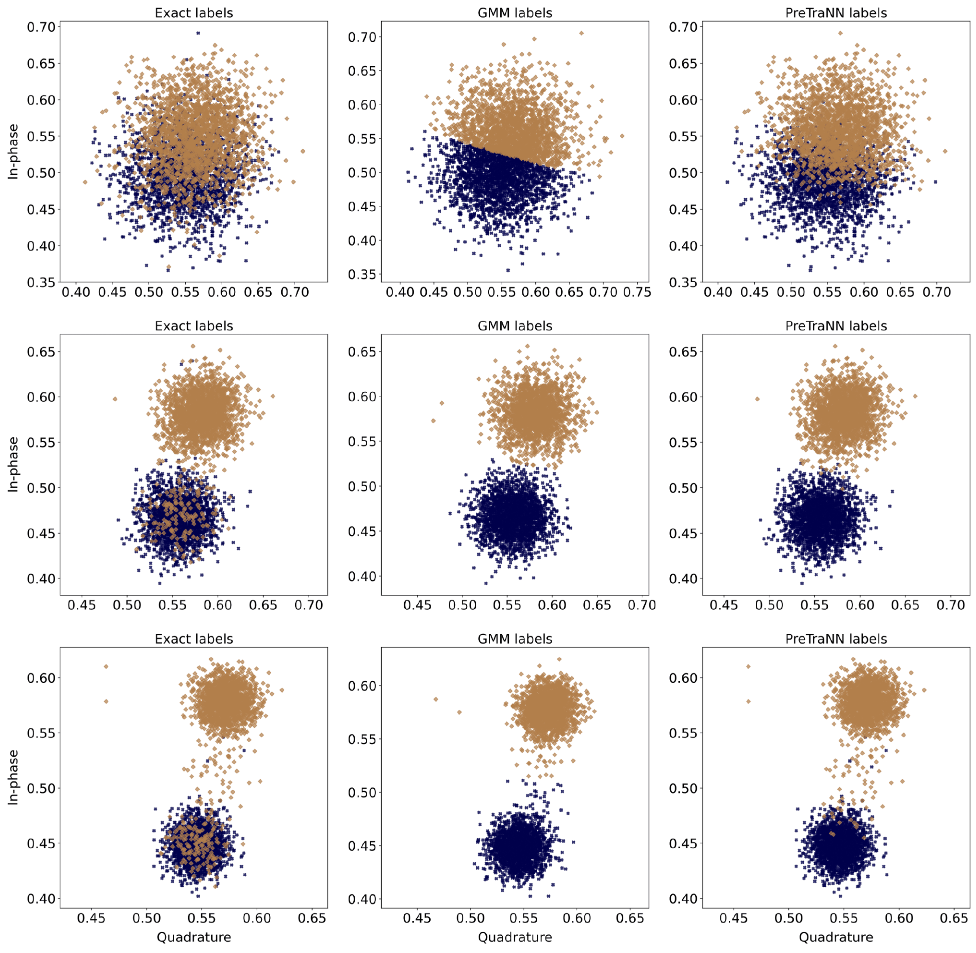

FIG. 7. A pictorial representation of the data set with exact, GMM, and PreTraNN labeling. Each point is the time average of the $I(t)$ and $Q(t)$ signals. The actual label, i.e., the prepared state, is represented in the first column. The GMM- and PreTraNN-method labels are represented in the second and third columns.

We would also like to stress that in other works, such as Ref. [24], the readout accuracy may be greater than the one reported here. As described above, the machine used for this work has a certain level of error in preparing state $|1\rangle$. This, however, is of secondary importance, since the purpose of the present work is not to present new hardware that over-performs compared to the current state-of-the-art hardware but only to propose a method to improve readout in the present machines. Thus, our interest is primarily focused on improving the performance of a given machine from the software point of view.

The classification obtained with PreTraNN not only improves the classification accuracy but also better reproduces the actual distribution of data. In Fig. 7, a comparison of the GMM and PreTraNN labeling results on data with different readout times is reported. The labeling for the FFNN is similar to that for PreTraNN, so it is omitted for clarity. The first column shows data with the actual labels (represented by colors) as they are prepared in the quantum device. The second and third columns, on the other hand, represent the same data but labeled according to the GMM and PreTraNN, respectively. The same analysis is performed for short, medium, and long times (see the rows of the figure). As anticipated, we again conclude that the GMM misses the classification for short times, simply dividing the overlapping distributions in half, while PreTraNN provides a considerably more realistic and accurate classification. The two distributions of overlapping points can now be spotted again.

The exact labels show the asymmetry in the data distribution due to the decay of the excited state: many $|1\rangle$-labeled points lie in the $|0\rangle$ distribution. The comparison between the labels highlights that there are many points belonging to state $|1\rangle$ that even PreTraNN fails to recognize. Probably, many of those points result from the imperfect calibration of the π pulse used to prepare the $|1\rangle$ state on the machine.

Another important measure to take into account is the confusion matrix, which helps to visualize the classification performance of the three methods in comparison with each other. The confusion matrices for the three methods in three different measurement-length setups are reported in Fig. 8. Each row reports the confusion matrices of the three models for a specific measurement length. Clearly, the best

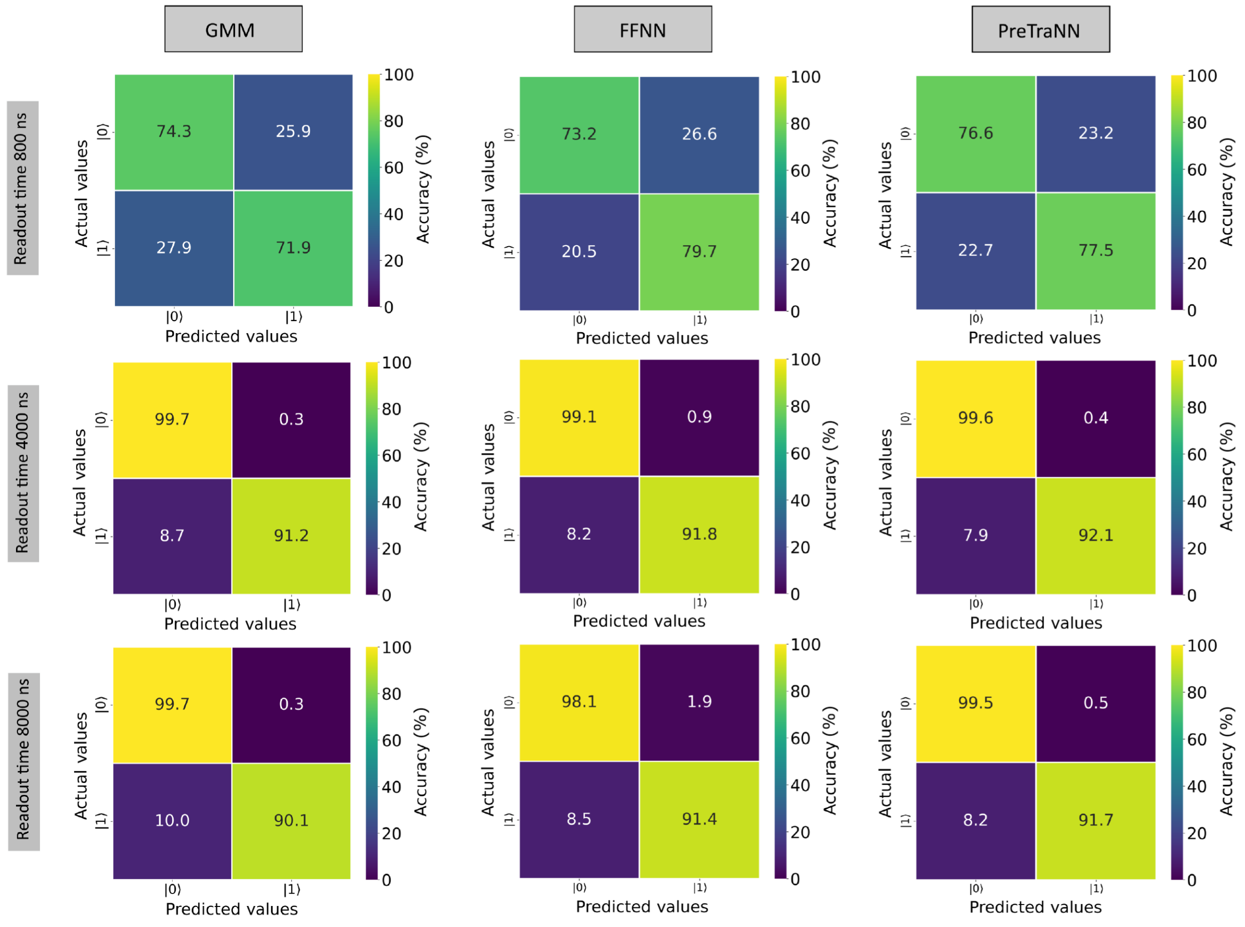

FIG. 8. Confusion matrices for the classification between states $|0\rangle$ and $|1\rangle$ for the three methods for short, medium, and long readout times.

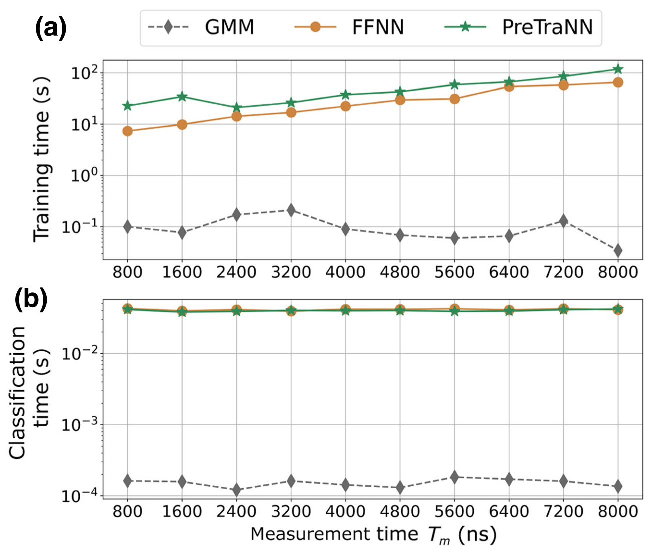

FIG. 9. The training and classification times for the GMM, FFNN, and PreTraNN methods. The times are reported in seconds for a midrange laptop computer. (a) The training time as a function of the measurement time (i.e., the length of the inputs). (b) The classification time. The average times are 0.00013 s for the GMM, 0.039 s for the FFNN, and 0.037 s for PreTraNN.

confusion matrices are those obtained for long times and with the PreTraNN model.

2. Computational cost and scaling

The higher structural complexity of the PreTraNN architecture means longer training and classification times than for the GMM. In the following, we report the results together with some considerations on the scaling of the method.

The training for every neural network is performed with the “early stopping” approach to avoid over- or underfitting. Instead of fixing the number of epochs, the training is stopped when the accuracy of the model does not increase for two epochs in a row. The results are reported in Fig. 9. The upper table shows the training time of each model with respect to the readout length T_m for a 16 000-elements data set, while the lower table, on the other hand, represents the average time for a single-input classification for each method. In both cases, the times are represented in logarithmic scale to spot trends better. Times are reported in seconds and refer to a midrange laptop computer with four cores and 8 GB of RAM.

Considering the training time, it can be noted that the PreTraNN method takes a significantly longer time than the parameter estimation for the GMM (from 2 to 3 orders of magnitude) but not much more than the FFNN, despite the two training sections of PreTraNN. As one might expect, the training time of non-GMM methods increases as the input measurement time increases. In fact, long measurement times correspond to wider neural networks and, therefore, longer optimizations.

TABLE I. Classification times for PreTraNN and the GMM as a function of the input batch size. Each reported time is the result of an average of 100 experiments. The FFNN method is not reported because its behavior follows that of PreTraNN.

Input batch size	Classification time PreTraNN (s)	Classification time GMM (s)
1	0.04200	0.00012
100	0.04300	0.00013
10 000	0.22400	0.00043

From the classification-time point of view, we see that the times for PreTraNN to label a single data point (0.039 and 0.042 s, respectively) are almost equal and much longer than for the GMM (0.00013 s). Moreover, for both methods, the classification time does not depend on the measurement length.

It is important to specify that the classification time for an input batch of size S is not S times the classification time for a single input. We report the actual classification times as a function of the batch size in Table I.

Based on these data, some considerations can be made. First, we can assert that the training for PreTraNN and the FFNN remains easily manageable by any computer, even for the longest measurement times. In fact, the training times, although much larger than for the GMM, remain very small in absolute value. In general, the training process is not a problem since it is done in advance.

On the classification-time side, however, more careful considerations must be made. If only an offline classification is needed, there are no stringent time constraints and the model could be considered fast enough for some applications. If, on the other hand, a real-time or online readout on the machine is needed, the classification times must be below the qubit lifetime. Since state-of-the-art superconducting transmon qubits have a lifetime of 200–500 μs [8,40], in principle we want a classification time that is well below these values, possibly on the order of tens or hundreds of nanoseconds. For this goal, neither the GMM nor PreTraNN has, under the conditions used in this work, the necessary characteristics. Of course, with the use of more powerful computers, the classification time can be reduced by a few orders of magnitude. Moreover, a field-programmable gate array (FPGA) or an application-specific integrated circuit (ASIC) implementation could improve the efficiency of the classification step even more or also improve the training process by implementing it in an online way (see Refs. [41–44]).

To summarize, the ability to perform short-time measurement classification (with higher accuracy) is of great interest in quantum computing. The proposed approach allows for good accuracy for short measurements compared to the GMM. This can be exploited for real-time control systems, e.g., quantum orchestration platforms, leading to measurement speed-up or reducing computational time in error-correction routines. Attention must be paid to the classification speed of the system. At least partially, however, the longer time required to perform classification can be compensated for by shorter measurements (as little as 1000 ns) than those for the GMM (4000 ns) while achieving the same classification accuracy. PreTraNN performs well regardless of the readout time, allowing one to potentially skip the readout time T_m trimming. Moreover, this method can be utilized for the usual two-level qubits or, conversely, extended to arbitrary numbers of levels or qubits with slight modifications in its structure and simply using different data sets. All this considered, the proposed method offers a promising approach to exploit short measurements that disturb the device as little as possible with less computational effort.

B. Three-state qutrit

In this case study, we exploit the possibility of accessing the higher quantum levels of superconducting qubits. We prepare and measure the qubit in the $|0\rangle$, $|1\rangle$, and $|2\rangle$ states and store the obtained data. The whole data set consists of 24 000 elements (8000 for each state), divided into 75% training data and 25% test data. Again, for consideration of how the data set is chosen, see Sec. II C. The architecture of the models is the same as in the previous case (and as defined in Secs. II C and II F). The only difference between the two cases is the number of classes in the data set. This allows us to show the good scaling properties of the model.

1. Classification accuracy

In this subsection, the global-classification accuracy is reported and discussed.

In Fig. 10, we present results for the global accuracy. The PreTraNN method achieves better classification performance for every measurement time. Again, the GMM accuracy presents an increasing and decreasing trend with a maximum located at 4000 ns, while the FFNN, notwithstanding a reduction in the fluctuating trend, obtains a lower classification accuracy than the other two methods, possibly due to training difficulties for high-dimensional data sets. PreTraNN, on the other hand, presents a stable accuracy as a function of the measurement time. For further details on the state-by-state classification accuracy, see Appendix D.

We can also study the performance of PreTraNN as a function of the number of qudit levels. This will give us an idea of how the method scales with the number of point clouds. To achieve this, we compute the difference in percentage points between the global-classification accuracy of PreTraNN and that of the other methods. The

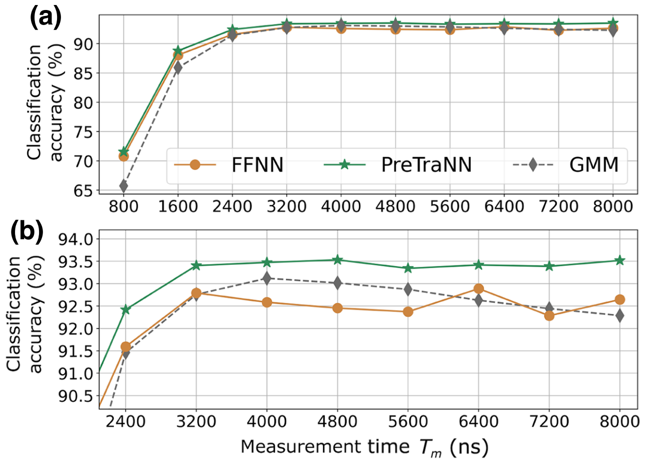

FIG. 10. The global-classification accuracy for the $|0\rangle$-, $|1\rangle$-, and $|2\rangle$-state classification for a qutrit.

difference (in percentage points) between the PreTraNN global accuracy and the GMM global accuracy for the two- and three-level cases for every measurement time T_m is reported in Fig. 11. The lower panel is an enlargement of the middle- and long-time range. In the small panels on the right, the average values for all T_m are highlighted. An increasing value of this difference, as the levels of the system increase, suggests a possible increasing advantage in using the PreTraNN method for increasing system levels. In this case, we observe that this trend can be clearly seen. Figure 12, on the other hand, reports the same calculation referred to the FFNN method. Here too, the trend is clear for both the whole set of measurement times and the medium-long range.

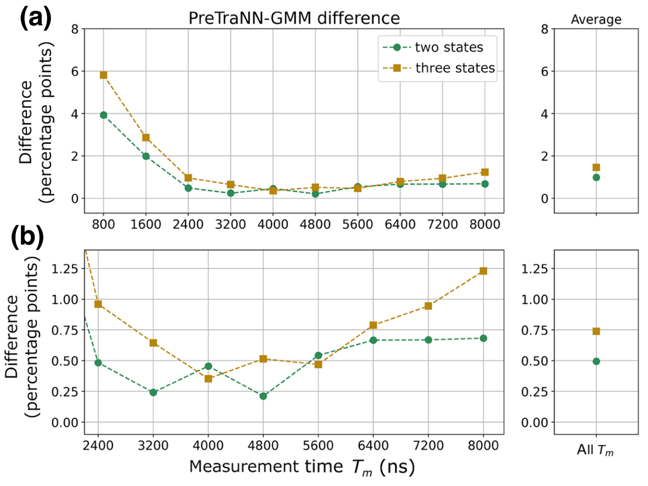

FIG. 11. (a),(b) The difference in percentage points between the accuracy of PreTraNN and GMM for the qubit and qutrit cases for different measurement times T_m: (b) reports the analysis for only medium-long times. The small panels on the right show the average of all values of the respective plot on the left.

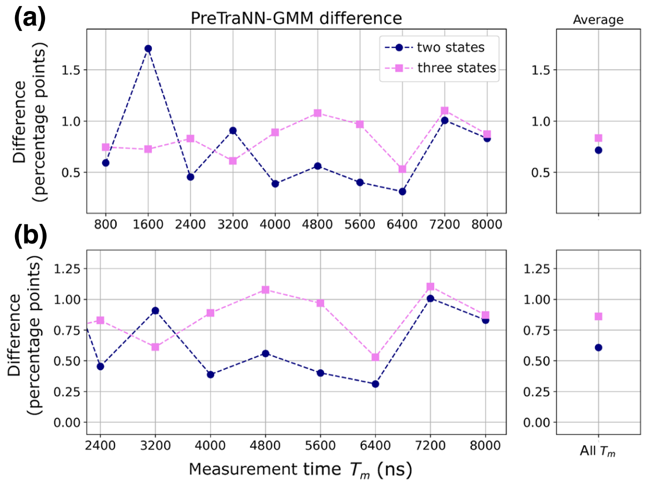

FIG. 12. (a),(b) The difference in percentage points between the accuracy of PreTraNN and the FFNN for the two- or three-qubit-state case: (b) reports the analysis for only medium-long times. The small panels on the right show the average value of the respective plots on the left.

This analysis suggests that there is a marginal increase in the effectiveness of PreTraNN compared to the other two methods as the classes of the data set increase (i.e., as the data-set complexity rises). In other words, the difference in the global-classification accuracy between PreTraNN and the GMM or between PreTraNN and the FFNN is bigger, on average, in the case of the three-class data set, corresponding to qutrit readout data.

This analysis, although limited to two- and three-class problems, suggests that the PreTraNN method should scale well as the qudit dimension increases. We can assume that it also scales well with the number of qubits, since it also reduces to a multiclass data set, but further analysis to better characterize the performance is needed.

Furthermore, PreTraNN requires only minimal structural modifications for different qudit dimensions. One only needs to adjust the number of output nodes in the last section of the network and use an appropriate data set with a different number of classes. While the training times rise due to the increased data-set size (the training time grows linearly with the data-set dimension), the classification time remains the same as for the previous two-state case.

IV. CONCLUSIONS

This work demonstrates that a feed-forward neural network with autoencoder pretraining allows for a robust qubit readout classification scheme with high accuracy and a low dependence on the feature values of the experimental device. It allows for a consistent classification performance even for short readout times, unlike the more traditional schemes affected by overlapping measurement results. It also obtains good results for longer measurement times, whereas the efficiency of the GMM is decreased due to energy-relaxation processes and a simple feed-forward neural network becomes difficult to train properly, resulting in fluctuating results.

In addition, the proposed method allows for good classification on shorter measures, achieving a measurement speed-up.

More importantly, this measurement speed-up is helpful for real-time control systems, e.g., quantum orchestration platforms or quantum error correction, where we need to disturb the system as little as possible.

In general, it is shown that the proposed method performs well for all measurement times, helping to increase the accuracy of the classification results from a software point of view. On the other hand, the classification times for a single measurement are higher than for the standard methods but can be improved with more optimized FPGA and ASIC implementations. Lastly, the proposed approach can be readily extended to an arbitrary number of states (or, possibly, number of qubits) with minimal modification of the model structure and can obtain marginally increasing performance.

ACKNOWLEDGMENTS

This research was partially supported by Quantum Science and Technology in Trento (Q@TN) grants ML-QForge (P.L.). The Lawrence Livermore work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344, with support from Laboratory Directed Research and Development Grant No. 19-DR-005 LLNL-JRNL-842516. P.E.T. acknowledges the Q@TN consortium for support. A.R. is funded by the European Union under the Horizon Europe program through Grant Agreement No. 101080086—NeQST. The views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of CERN, the European Union, or the European Climate, Infrastructure and Environment Executive Agency (CINEA). CERN, the European Union, and the granting authority cannot be held responsible for the views and opinions expressed.

APPENDIX A: NUMERICAL CONSIDERATIONS ON AUTOENCODER

1. Autoencoder latent-space dimension

In the design of the architecture of a neural network, there is no solid theoretical guidance but one has to rely on a heuristic and "trial-and-error" attitude based on experience. However, to make the procedure more quantitative, one can vary the structure in an automated way and study how its metrics vary. In this way, one can identify, within

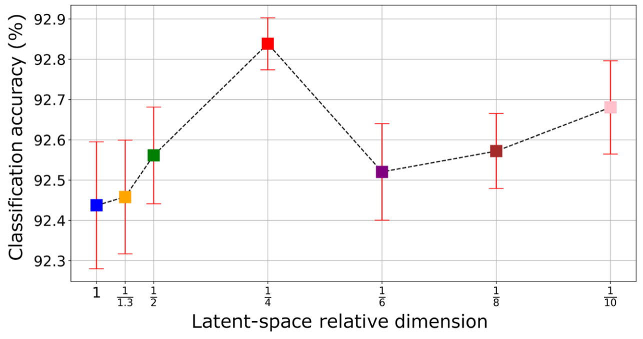

FIG. 13. The PreTraNN global-classification accuracy for the three-state case with 2400-ns readout inputs as a function of the latent-space dimension. The higher accuracy is reached at 1/4 the input dimension.

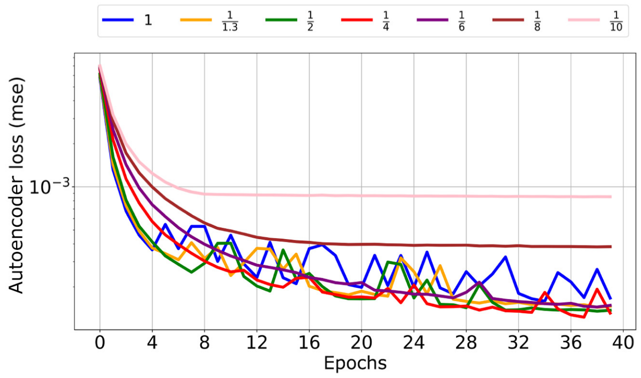

FIG. 14. The autoencoder training loss function as a function of the training epochs for different latent-space relative dimensions. Too large latent dimension (i.e. the ones with relative dimension 1, 1/1.3 or 1/2 the input dimension) present a fluctuating behavior and are useless for feature extraction, while too small latent dimensions do not allow an effective encoding and their loss function remains high (1/8 and 1/10 the input size).

a certain degree of approximation, the architecture that works best for the specific problem.

In the case of the autoencoder, the main parameter is its latent-space size. In principle, a latent space that is too small is not sufficient to perform expressive encoding, while too large a latent space increases the computational cost without extracting, in a compact way, information from the data set. In the limiting case of a latent space equal to the input space, the neural network becomes equivalent to applying an identity to the inputs.

In this appendix, we describe the procedure used in our work to identify the best autoencoder structure. We take the PreTraNN method with trajectories of 2400 ns (150 time steps of 16 ns, i.e., an input dimension of 300 values) and train it for different values of latent space. We start from a latent dimension equal to the input dimension and gradually go down to 1/10 of it. The dimension of the other two inner layers is set to linearly interpolate between the size of the input and the latent space. The decoder has the same structure but reversed. Contextually, three properties of PreTraNN are studied as a function of the latent dimension: the global-classification accuracy, the autoencoder training loss, and the autoencoder training time. To obtain more consistent results, for each latent dimension the training is repeated 10 times with different samplings of the data set and the property values are averaged.

The PreTraNN global-classification accuracy for a decreasing latent-space dimension is reported in Fig. 13. The abscissa shows the size of the latent space in terms of fractions of the input length (so that the information extracted from this case can be scaled directly to the other input lengths). The greatest accuracy—moreover, with the smallest error bars—is achieved with a latent space the size of which is 1/4 that of the input space. In absolute terms, the classification accuracy is quite stable for every latent-space dimension but an increasing trend from 1 to 1/4 can be clearly spotted.

The loss-function values (the mse) during the training of the autoencoder for different latent-space dimensions are represented in Fig. 14. For large latent-space sizes, the training converges faster for the first epochs but then assumes a fluctuating trend. For latent spaces that are small (e.g., 1/10 or 1/8 the size of the input), on the other hand, convergence stalls at much higher values of the loss function. Thus the best values are 1/2, 1/4, and 1/6 the input length.

The training time is reported in Fig. 15, in seconds. Clearly, the training time decreases as the latent space decreases, since the number of network parameters decreases. A short training time is preferable.

Given this PreTraNN behavior, we can choose the latent-space dimension by making a trade-off between the reported metrics. The value that maximizes the classification accuracy and has at the same time good loss-function

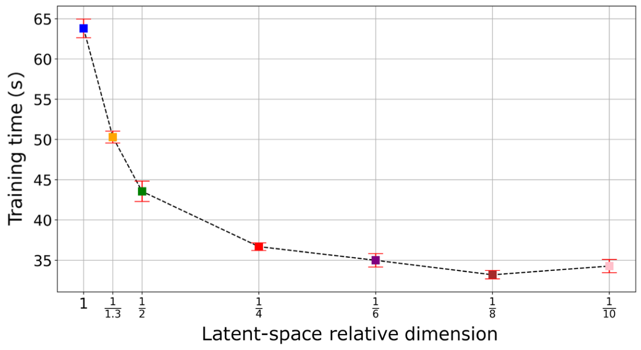

FIG. 15. The autoencoder training time as a function of the latent-space relative dimension. Clearly, larger latent spaces correspond to neural networks with more parameters and thus longer training times.

convergence and a (relatively) short training time is a latent dimension of 1/4 the input size. This is the value chosen to carry out the analysis in this work. The dimension of the two internal layers is set to linearly interpolate between the latent space and the input dimensions.

2. Data-set size and convergence

In order to obtain a good training convergence that maximizes the classification accuracy, an adequate data set is needed. Small data sets are fast to train but usually produce inadequate classification accuracies, while large ones have the opposite behavior. At the same time, the growth of the classification-accuracy capability decreases marginally with an increasing data-set size. Here, we report some analysis on the behavior of PreTraNN as a function of the data-set dimension, studying the same three properties as introduced in Sec. A 1, i.e., the loss function, the classification accuracy, and the training time. Even in this case, we take the PreTraNN method with 2400 ns measurement signals (150 time steps of 16 ns, i.e. an input dimension of 300 values) with a latent space of 75 neurons and train it for different data-set dimensions. We start from a training data set of 3000 elements (1000 elements for each class) and gradually increase its dimension to 60 000 elements (with 75% of them dedicated to training). For each data-set dimension, the training is repeated 10 times with different sampling of the data set and the property values are averaged.

The global-classification accuracy is reported in Fig. 16, as a function of the data-set size. It can be seen that the accuracy increases as the data set grows, even if it grows with decreasing speed.

Figure 17 represents the loss-function values (the mse) during the training of the autoencoder for different configurations. The trend is quite neat. The larger the data set, the better is the convergence, although for large data sets the convergence becomes more unstable.

The training time is reported in Fig. 18, in seconds. As expected, the training time increases linearly with the data-set dimension. A short training time is preferable.

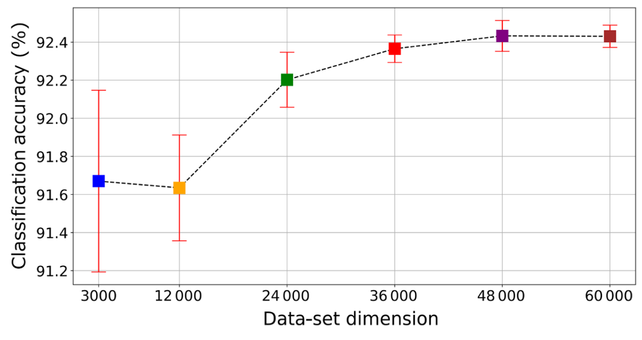

FIG. 16. The global-classification accuracy of PreTraNN as a function of the number of data-set elements.

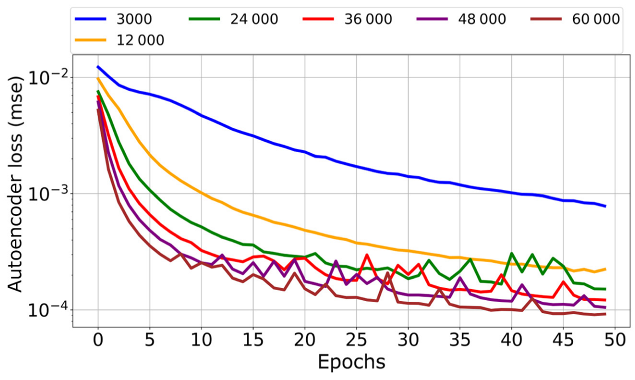

FIG. 17. The autoencoder loss (the mse) as a function of the epochs for increasing data-set size.

Given these results, the trade-off among the accuracy, the loss function, and the training time, in order to maximize effectiveness and minimize cost, is identified in the 24 000-item data set for the three-state case and in the 16 000-item data set for the two-state case.

APPENDIX B: MODEL SPECIFICATIONS

Here, we report the complete characterization of the autoencoder, the PreTraNN, FFNN, and GMM models and their training procedure.

In this work, the building and training of the neural network are performed via the PYTHON package KERAS [45]. On the other hand, for the GMM, the SKLEARN PYTHON package [46] is used.

1. Autoencoder

In every configuration employed in this work, the encoder is composed of an input layer, a first hidden layer, and a second hidden layer connected to the latent layer. The decoder, on the other hand, has the same structure but is mirrored. Therefore, it has a first hidden layer connected to the latent layer, a second hidden layer, and finally an output layer. We employ a full-connectivity network

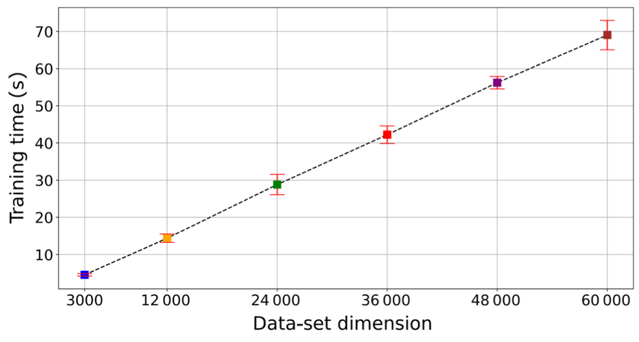

FIG. 18. The training time for an increasing data-set dimension.

TABLE II. The autoencoder specifications. The "Size" column represents the number of neurons for each layer in a fraction of the input dimension L. The "KERAS type" column reports the type of KERAS layer employed.

	Layer	Size	Activation function	KERAS type
Encoder	Input	L	Sigmoid	DENSE
	First hidden	$L \times 3/4$	Tanh	DENSE
	Second hidden	$L \times 2/4$	Tanh	DENSE
	Latent	L/4	Tanh	DENSE
Decoder	First hidden	$L \times 2/4$	Tanh	DENSE
	Second hidden	$L \times 3/4$	Tanh	DENSE
	Output	L	Sigmoid	DENSE

implemented with the DENSE layer specification in KERAS. All the information on the network is reported in Table II.

The training is performed using the Adam stochastic optimization algorithm [47], with the standard configuration implemented in KERAS. The loss function is the *mean-square error* (mse). The training is performed with the EARLY STOPPING procedure, which stops the training if the loss does not decrease for two epochs in a row.

2. FFNN and PreTraNN second section

The second section of PreTraNN is a simple feed-forward neural network. It is composed of an input layer (of the same dimension as the latent layer of the autoencoder), a first hidden layer, and a second hidden layer connected to the output layer. The dimension C of the output layer depends on the number of classes with which we are doing the classification. Hence, $C = 2$ for the qubit classification of Sec. III A, while $C = 3$ for the qutrit classification of Sec. III B. The connectivity between the neurons is full. The optimization algorithm is Adam. The loss function is the *cross-entropy*, which is suitable for classification purposes. The training is performed with the EARLY STOPPING procedure that stops the training if the loss does not decrease for two epochs in a row. Other information is summarized in Table III. The structure of the FFNN model is the same but with a number of input neurons equal to the data-set dimension instead of the latent-layer dimension.

TABLE III. The structure and specifications of the PreTraNN second-section (FFNN) network with KERAS. L is the data-set input length and C is the dimension of the output layer, which changes based on the number of classes.

Layer	Size	Activation function	KERAS type
Input	$L/4$ (L)	Tanh	DENSE
First hidden	$L \times 2/4$ ($2L$)	Tanh	DENSE
Second hidden	$L/4$ (L)	Tanh	DENSE
Output	C	Softmax	DENSE

3. Gaussian mixture model

The GMM is implemented using the SKLEARN package with the standard built-in parameters specifying only the number of classes of the input data set.

APPENDIX C: AUTOENCODER FEATURES

In this appendix, we give examples of the two important autoencoder features: input regeneration and latent-space values. Figure 19 shows an example of 3200-ns (i.e., 400 components) input reconstruction done by the autoencoder. The solid lines represent the original input (divided into the two quadratures), while the lines with markers represent the output of the autoencoder, i.e., the regeneration of the input from its synthetic representation in the latent space of the autoencoder. It can be seen that the reconstruction is quite faithful to the original.

The latent-space representation is presented in Fig. 20. The thin colored lines represent the latent-space values of different inputs, while the thick black line is the average of such lines. It can be seen that the latent-space vectors for the two states are somewhat different on average. Both have zero on average but those for $|0\rangle$ have larger fluctuations and a little structure. In particular, in both plots specific points where all the $\mathbf{h}^i$ vectors follow a definite trend (e.g., the points around 20 and 60 for state $|0\rangle$) can be spotted. These are the differences that allow the increase in classification performance shown in this paper.

One might wonder how the input reconstruction varies as the latent representation varies. To answer this question, we can proceed as follows. We use the encoder to obtain the latent representation of an input, we then vary just one of its values slightly, and finally, we plug the modified latent vector into the decoder to obtain its "reconstruction."

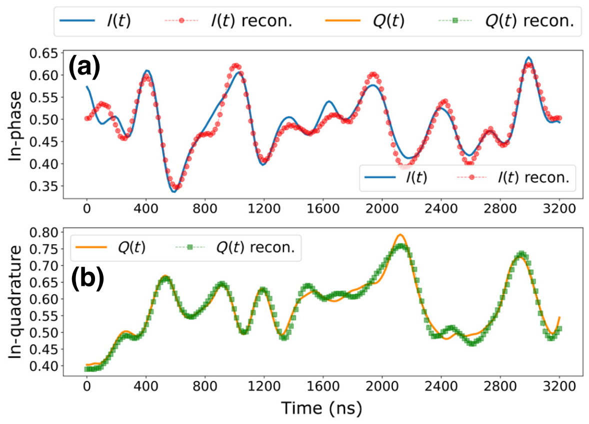

FIG. 19. An example of input regeneration made by the autoencoder. In both panels, the solid lines represent the measurement signal divided into its two quadratures: (a) in-phase (I) and (b) in-quadrature (Q), respectively. The lines with markers represent the input reconstruction made by the autoencoder.

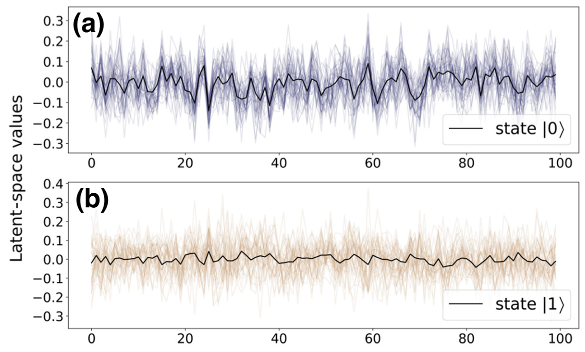

FIG. 20. A representation of the latent space of the autoencoder for states (a) $|0\rangle$ and (b) $|1\rangle$. In both panels, the colored lines are the latent-space representation (i.e., $\mathbf{h}^i$ vector) of inputs for state $|0\rangle$ or $|1\rangle$. The solid black lines represent the average of these values.

We do this several times by varying the input slightly each time. Figure 21 depicts the result of this procedure. The thick lines represent the correct reconstruction of an input (divided into I and Q components), while the thin lines represent the reconstruction for increasing values of the 20th component of the latent representation. We can see that by varying this value slowly, we obtain a slowly varying family of reconstructions.

APPENDIX D: OTHER CLASSIFICATION DATA

In this appendix, we report classification-accuracy data for the single states of the three-level qutrit case introduced in Sec. III B.

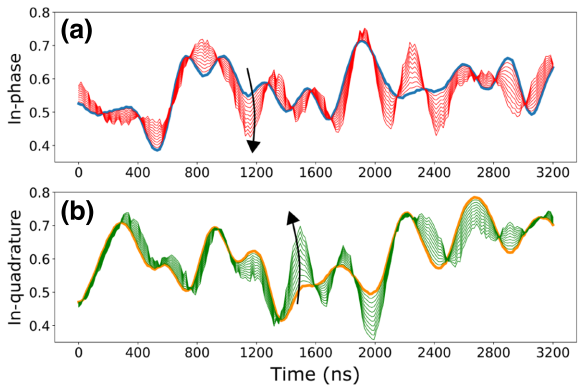

FIG. 21. An example of how the input reconstruction varies if a single value of the latent representation is varied slightly: (a) the in-phase component and (b) the quadrature component. In both panels, the thick lines are the original "correct" input reconstruction and the thin lines represent the reconstructions obtained by slowly varying a single value of the latent representation. In both panels, arrows are used to indicate the direction of changes induced by increasing the latent value.

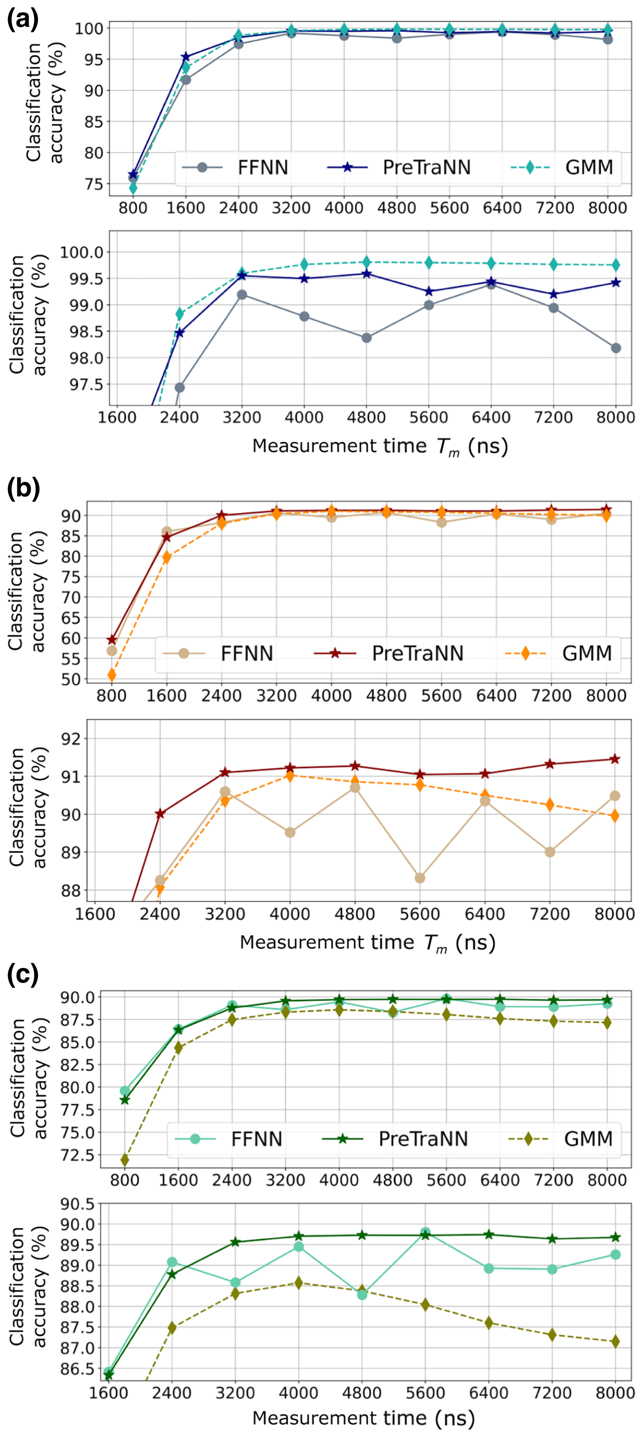

FIG. 22. The state-by-state classification accuracy for the qutrit case. (a) Upper panel: the state-$|0\rangle$ classification accuracy for the three methods as a function of the measurement time in the case of a qutrit. Lower panel: an enlargement of the medium-long times. (b) Upper panel: the state-$|1\rangle$ classification accuracy for the three methods as a function of the measurement time in the case of a qutrit. Lower panel: an enlargement of the medium-long times. (c) Upper panel: the state-$|2\rangle$ classification accuracy for the three methods as a function of the measurement time. Lower panel: an enlargement of the medium-long times.

In Figs. 22(a), 22(b), and 22(c) we show the classification accuracy for, respectively, states $|0\rangle$, $|1\rangle$, and $|2\rangle$. The lower panel of each figure is an enlargement of the (2400–8000)-ns part of the plot to see the details better. Even in this configuration, we can see the same trends as in the two-level case. All methods show bad results for short times, especially the GMM, and the FFNN still exhibits a seesaw pattern that makes it poorly suited to the task. Again, the GMM performs better than PreTraNN in state $|0\rangle$ and worse in the state $|1\rangle$ classification due to the data-distribution asymmetry. For state $|2\rangle$, the difference between the GMM and PrTranNN is even higher, since state $|2\rangle$ can decay not only on the state $|0\rangle$ but also on the state $|1\rangle$.

[1] R. Barends *et al.*, Digital quantum simulation of fermionic models with a superconducting circuit, Nat. Commun. **6**, 7654 (2015).

[2] G. Wendin, Quantum information processing with superconducting circuits: A review, Rep. Prog. Phys. **80**, 106001 (2017).

[3] Z. Yan *et al.*, Strongly correlated quantum walks with a 12-qubit superconducting processor, Science **364**, 753 (2019).

[4] E. T. Holland, K. A. Wendt, K. Kravvaris, X. Wu, W. E. Ormand, J. L. DuBois, S. Quaglioni, and F. Pederiva, Optimal control for the quantum simulation of nuclear dynamics, Phys. Rev. A **101**, 062307 (2020).

[5] F. Arute *et al.*, Quantum supremacy using a programmable superconducting processor, Nature **574**, 505 (2019).

[6] A. Nersisyan, S. Poletto, N. Alidoust, R. Manenti, R. Renzas, C.-V. Bui, K. Vu, T. Whyland, Y. Mohan, and E. A. Sete *et al.*, in *2019 IEEE International Electron Devices Meeting (IEDM)* (IEEE, San Francisco, CA, USA, 2019), p. 31.

[7] L. B. Nguyen, Y.-H. Lin, A. Somoroff, R. Mencia, N. Grabon, and V. E. Manucharyan, High-Coherence Fluxonium Qubit, Phys. Rev. X **9**, 041041 (2019).

[8] H. L. Huang, D. Wu, D. Fan, and X. Zhu, Superconducting quantum computing: A review, Sci. China Inf. Sci. **63**, 1 (2020).

[9] J. Werschnik and E. Gross, Quantum optimal control theory, J. Phys. B **40**, R175 (2007).

[10] J. P. Palao and R. Kosloff, Quantum Computing by an Optimal Control Algorithm for Unitary Transformations, Phys. Rev. Lett. **89**, 188301 (2002).

[11] S. Kirchhoff, T. Keßler, P. J. Liebermann, E. Assémat, S. Machnes, F. Motzoi, and F. K. Wilhelm, Optimized cross-resonance gate for coupled transmon systems, Phys. Rev. A **97**, 042348 (2018).

[12] M. D. Reed, L. DiCarlo, S. E. Nigg, L. Sun, L. Frunzio, S. M. Girvin, and R. J. Schoelkopf, Realization of three-qubit quantum error correction with superconducting circuits, Nature **482**, 382 (2012).

[13] A. D. Córcoles, E. Magesan, S. J. Srinivasan, A. W. Cross, M. Steffen, J. M. Gambetta, and J. M. Chow, Demonstration of a quantum error detection code using a square lattice of four superconducting qubits, Nat. Commun. **6**, 1 (2015).

[14] M. Gong *et al.*, Experimental exploration of five-qubit quantum error-correcting code with superconducting qubits, Nat. Sci. Rev. **9**, 1 (2021).

[15] A. Roggero and A. Baroni, Short-depth circuits for efficient expectation-value estimation, Phys. Rev. A **101**, 022328 (2020).

[16] T. Walter, P. Kurpiers, S. Gasparinetti, P. Magnard, A. Potočnik, Y. Salathé, M. Pechal, M. Mondal, M. Oppliger, C. Eichler, and A. Wallraff, Rapid High-Fidelity Single-Shot Dispersive Readout of Superconducting Qubits, Phys. Rev. Appl. **7**, 054020 (2017).

[17] Y. Sunada, S. Kono, J. Ilves, S. Tamate, T. Sugiyama, Y. Tabuchi, and Y. Nakamura, Fast Readout and Reset of a Superconducting Qubit Coupled to a Resonator with an Intrinsic Purcell Filter, Phys. Rev. Appl. **17**, 044016 (2022).

[18] A. P. Place *et al.*, New material platform for superconducting transmon qubits with coherence times exceeding 0.3 milliseconds, Nat. Commun. **12**, 1 (2021).

[19] A. Blais, R.-S. Huang, A. Wallraff, S. M. Girvin, and R. J. Schoelkopf, Cavity quantum electrodynamics for superconducting electrical circuits: An architecture for quantum computation, Phys. Rev. A **69**, 062320 (2004).

[20] A. Wallraff, D. I. Schuster, A. Blais, L. Frunzio, J. Majer, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf, Approaching Unit Visibility for Control of a Superconducting Qubit with Dispersive Readout, Phys. Rev. Lett. **95**, 060501 (2005).

[21] R. Bianchetti, S. Filipp, M. Baur, J. M. Fink, M. Göppl, P. J. Leek, L. Steffen, A. Blais, and A. Wallraff, Dynamics of dispersive single-qubit readout in circuit quantum electrodynamics, Phys. Rev. A **80**, 043840 (2009).

[22] J. Gambetta, A. Blais, M. Boissonneault, A. A. Houck, D. I. Schuster, and S. M. Girvin, Quantum trajectory approach to circuit QED: Quantum jumps and the zeno effect, Phys. Rev. A **77**, 012112 (2008).

[23] D. A. Reynolds, Gaussian mixture models, Encycl. Biometr. **741**, 827 (2009).

[24] E. Magesan, J. M. Gambetta, A. D. Córcoles, and J. M. Chow, Machine Learning for Discriminating Quantum Measurement Trajectories and Improving Readout, Phys. Rev. Lett. **114**, 200501 (2015).

[25] A. Seif, K. A. Landsman, N. M. Linke, C. Figgatt, C. Monroe, and M. Hafezi, Machine learning assisted readout of trapped-ion qubits, J. Phys. B **51**, 174006 (2018).

[26] B. Lienhard, A. Vepsäläinen, L. C. G. Govia, C. R. Hoffer, J. Y. Qiu, D. Ristè, M. Ware, D. Kim, R. Winik, A. Melville, B. Niedzielski, J. Yoder, G. J. Ribeill, T. A. Ohki, H. K. Krovi, T. P. Orlando, S. Gustavsson, and W. D. Oliver, Deep-neural-network discrimination of multiplexed superconducting-qubit states, Phys. Rev. App. **17**, 014024 (2022).

[27] D. Quiroga, P. Date, and R. Pooser, in *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)* (IEEE, Los Alamitos, CA, USA, 2021), p. 481.

[28] L. A. Martinez, Y. J. Rosen, and J. L. DuBois, Improving qubit readout with hidden Markov models, Phys. Rev. A **102**, 062426 (2020).

[29] X. Wu, S. L. Tomarken, N. A. Petersson, L. A. Martinez, Y. J. Rosen, and J. L. DuBois, High-Fidelity Software-Defined

Quantum Logic on a Superconducting Qudit, Phys. Rev. Lett. **125**, 170502 (2020).

[30] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning* (Springer, New York, NY, USA, 2006), Vol. 4.

[31] Y. Bengio, A. Courville, and P. Vincent, Representation learning: A review and new perspectives, IEEE Trans. Pattern. Anal. Mach. Intell. **35**, 1798 (2013).

[32] L. Pasa and A. Sperduti, Pre-training of recurrent neural networks via linear autoencoders, Adv. Neural Inf. Process Syst. **2**, 3572 (2014).

[33] B. T. Ong, K. Sugiura, and K. Zettsu, in *2014 IEEE International Conference on Big Data (Big Data)* (IEEE, Washington, DC, USA, 2014), p. 760.

[34] L. Chen, H.-X. Li, Y. Lu, C. Warren, C. Križan, S. Kosen, M. Rommel, S. Ahmed, A. Osman, J. Biznárová, A. Fadavi, B. Lienhard, M. Caputo, K. Grigoras, L. Grönberg, J. Govenius, A. Kockum, P. Delsing, J. Bylander, and G. Tancredi, Transmon qubit readout fidelity at the threshold for quantum error correction without a quantum-limited amplifier, npj Quantum Inf. **9**, 26 (2023).

[35] S. Kohler, Dispersive readout: Universal theory beyond the rotating-wave approximation, Phys. Rev. A **98**, 023849 (2018).

[36] Quantum orchestration platform (2021).

[37] X. Yu, M. O. Efe, and O. Kaynak, A general backpropagation algorithm for feedforward neural networks learning, IEEE Trans. Neural Netw. **13**, 251 (2002).

[38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, MA, USA, 2016), http://www.deeplearningbook.org.

[39] A. Shrestha and A. Mahmood, Review of deep learning algorithms and architectures, IEEE Access **7**, 53040 (2019).

[40] M. Kjaergaard, M. E. Schwartz, J. Braumüller, P. Krantz, J. I.-J. Wang, S. Gustavsson, and W. D. Oliver, Superconducting qubits: Current state of play, Annu. Rev. Condens. Matter Phys. **11**, 369 (2020).

[41] I. Westby, X. Yang, T. Liu, and H. Xu, FPGA acceleration on a multi-layer perceptron neural network for digit recognition, J. Supercomput. **77**, 14356 (2021).

[42] R. Sarić, D. Jokić, N. Beganović, L. G. Pokvić, and A. Badnjević, FPGA-based real-time epileptic seizure classification using artificial neural network, Biomed. Sig. Process. Contr. **62**, 102106 (2020).

[43] Y. Jewajinda and P. Chongstitvatana, in *ECTI-CON2010: The 2010 ECTI International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology* (IEEE, Chiang Mai, Thailand, 2010), p. 1050.

[44] S. Gandhare and B. Karthikeyan, in *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)* (IEEE, Vellore, India, 2019), p. 1.

[45] KERAS web site, https://keras.io/.

[46] SKLEARN web site, https://scikit-learn.org/stable/index.html.

[47] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization (2015).