

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, I-38123, Povo, Trento, Italy*

²*INFN-TIFPA Trento Institute of Fundamental Physics and Applications, Trento, Italy*

³*Scientific Computing Department STFC-UKRI, Rutherford Appleton Laboratory, Didcot OX11 0QX, United Kingdom*

⁴*Lawrence Livermore National Laboratory, P.O. Box 808, L-414, Livermore, California 94551, USA*

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 novel readout classification method for superconducting qubits based on a neural network pre-trained with an autoencoder approach. A neural network is pre-trained with qubit readout signals as autoencoders in order to extract relevant features from the data set. Afterwards, the pre-trained 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 lower performance.

I. INTRODUCTION

The construction of a computer exploiting quantum – rather than classical – principles represents a formidable scientific and technological challenge. Nowadays, superconducting quantum processor 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 of all, it is required the ability 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]). In second place, one needs the ability to perform a complete quantum error correction protocol [12–14]. Finally, it is of primary importance to have an high-fidelity qubit readout measurement to extract information from the device especially for observables that are very sensitive to it (see e.g. [15] for an extreme case of this). In addition to a careful design of the system parameters [16, 17] or improvement in fabrication processes extending qubits coherence time [6, 18], readout fidelity can be enhanced through the use of machine-learning techniques.

The currently most common qubit readout technique is the dispersive readout (in the 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 right state of the qubit.

Gaussian mixture model [23] is the most commonly used classification method given its ease of use. It exploits a parametric modelling of the averaged readout signals probability distribution in terms of sum of Gaussians to perform a classification of each measurement. In [24–27] the authors proposed 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 [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 novel semi-supervised machine learning classification method based on autoencoder pre-training applied to the heterodyne readout signal of a superconducting qubit [19, 29]. Autoencoders are a type of artificial neural network designed to encode efficiently a set of data 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 dataset, i.e. those features which allow for the most faithful reconstruction of input data while neglecting noise and non relevant 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 by an autoencoder trained to reconstruct the qubit readout signals dataset. 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

* piero.luchi@unitn.it

measurement time lengths. We remark 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 will be only on the software improvement on present machines.

The paper is organized as follows. In Sec. I, qubit setup and readout, 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 method's applicability 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].

Due to the qubit interaction, the readout resonator undergoes a frequency shift whose value 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 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 qubit's state. The demodulation procedure can extract such information from the signal, discriminating between qubit states.

In our setup (a superconducting qubit controlled via QUA software [36]), a signal of amplitude A and frequency ω_r 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 $\omega_{IF} = \omega_r - \omega_{LO}$ through a signal mixer, with ω_{LO} the frequency of the local oscillator (an electronic component needed by the mixer to change signals frequency). Finally, it has to be demodulated to extract information about the qubit state that the readout signal acquired in the interaction.

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

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

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

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

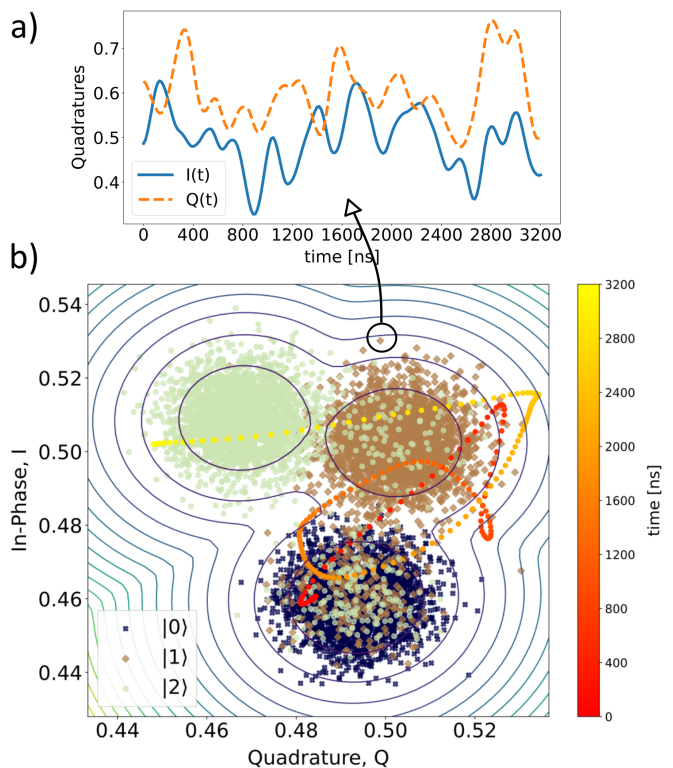


FIG. 1: Pictorial representation of qubit readout data. *Panel a* Example of in-phase, $I(t)$, and quadrature, $Q(t)$, components of heterodyned signal of a single shot obtained via sliced demodulation (as described in Sec. II A). *Panel b* Example of the whole data-set. Each point is the time average of a measurement represented in the I-Q plane for qubit states 0,1, and 2. The lines represent the 2D Gaussians contour plot (see Sec. II B 1) for the 3 Gaussian distribution. The dotted red-yellow line is an example of a measurement signal represented in the I-Q plane. The colours represent the time evolution (in nanoseconds).

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 as displayed in panel (b) of Fig. 1.

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

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

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

with $\Delta t = T_m/C$ the subinterval length and C the number of intervals. In this way, we obtain two time series, $I(t)$ and $Q(t)$ for each measurement. An example of this $I(t)$ and $Q(t)$ sequence is plotted in panel (a) of Fig. 1 or,

represented in the I-Q plane as a trajectory (state-path trajectory), in panel (b) of the same figure. Moreover, this trajectory can be time-averaged to obtain a single point as in the case of full demodulation (See panel b of Fig. 1). In principle, this type of demodulation should retain information that otherwise is lost in the averaging process of the complete demodulation. This information will be exploited in this work to increase 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 (see. Fig. 1) with a distribution that is as Gaussian as possible in order to use the Gaussian Mixture Model 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, low classification accuracy. In contrast the full, sliced, demodulation data retains more information about the qubit state measurements and, in principle, allows for increased accuracy of the state classification. Moreover, as will be 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.

It should be mentioned that data preparation is not error-free. Indeed, it may happen that, due to control errors or environmental coupling, the state ($|0\rangle, |1\rangle$ or $|2\rangle$) that is expected is not actually prepared. There will therefore be data in the dataset that, although labelled with a certain state, actually belong to another one. So the classification will not be 100% accurate even with the model proposed in this paper because the dataset suffers from this inaccuracy.

B. Machine Learning Models

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

1. Gaussian Mixture Model

Gaussian mixture model (GMM) approximate a distribution of data (in this case, the clouds of mean demodulation points in panel (b) of Fig. 1) as a weighted superposition of Gaussian distributions [23]. The GMM models the distribution by training on the dataset 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 (FFNN) are the simplest class of neural networks. Trained over a labelled dataset, 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 $\mathcal{Y} \subseteq \mathbf{R}^n$ which encodes in some way the information on the classes the inputs are divided into. The inputs are the full qubits 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:

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

This optimization is commonly carried out 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 network's weights and biases 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 inputs reconstruction. These functions are defined as:

$$\begin{aligned} f_{\theta^e} &: \mathcal{X} \rightarrow \mathcal{L} \\ g_{\theta^d} &: \mathcal{L} \rightarrow \mathcal{X} . \end{aligned}$$

The function f_{θ^e} takes an input $\mathbf{x}^i \in \mathcal{X} \subseteq \mathbf{R}^m$ from the data-set $\{\mathbf{x}^1, \mathbf{x}^2, \dots\}$ 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:

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

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

C. Model: Neural Network with Autoencoder type Pre-training

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

The first section consists of an encoder f_{θ^e} whose parameters θ^e are pre-trained in advance as an autoencoder over the input dataset. 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 pre-training has the same structure as the encoder but in the reverse order. Given an input of dimension d , we always set $L_1 = \frac{3}{4}d$, $L_2 = \frac{2}{4}d$ and $L_H = \frac{1}{4}d$. The activation functions are the *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's classes. Given d the dimension of the input, we set $L_{N_1} = 2d$ and $L_{N_2} = d$. The activation functions are *tanh* for the internal layers and the *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 vector, i.e $\mathbf{X}^i = [\mathbf{x}_I^i, \mathbf{x}_Q^i]$ so 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. 2.

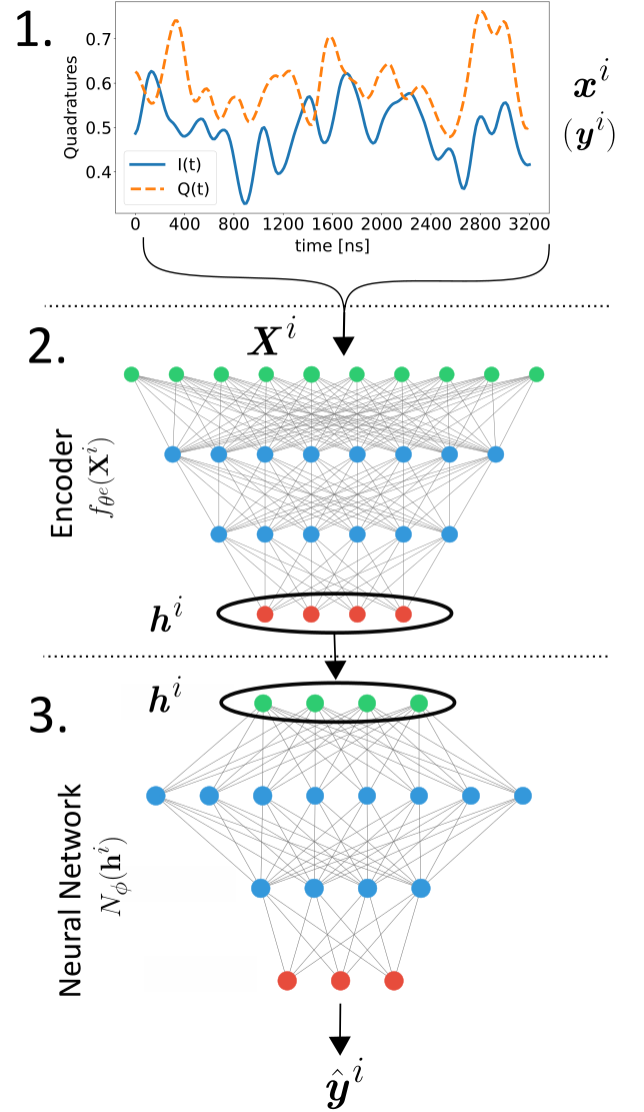


FIG. 2: Pictorial representation of the working principle and the architecture of the PreTraNN method described in Sec. III C.
Section 1: Example of the measurement signal $\mathbf{x}(t)$ we want to classify with PreTraNN.
Section 2: The input $\mathbf{x}(t)^i$ is flattened to obtain \mathbf{X}^i , plugged 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 $\hat{\mathbf{y}}^i$.

1. Training

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

The autoencoder is trained first. The dataset is composed by inputs \mathbf{x}^i with $i = 1, 2, \dots, M$, representing the 2D trajectories in the I-Q plane. The neural network architecture requires a one dimensional vector input so \mathbf{x}^i need to be flattened, stacking the I and the Q components in a single one-dimensional vector. So we compose a new dataset of $\mathbf{X}^i = [\mathbf{x}_I^i, \mathbf{x}_Q^i]$. The parameters

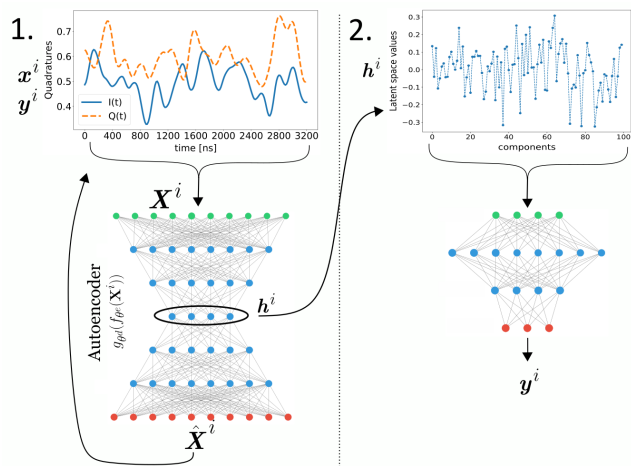


FIG. 3: Pictorial representation of PreTraNN training described in Sec. II C. *Section 1*: The autoencoder is trained to reconstruct the measurement signals. This should train the network to extract the relevant features from each temporal chunk. *Section 2*: After the training, the decoder part of the network is removed, and the encoded representation of data (represented in the plot at the top right) are used as train input dataset for the second section of the PreTraNN model which is trained to classify them into the correct class \mathbf{y}^i

$\theta = [\theta^e, \theta^d]$ 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

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

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

In a second step, the neural network N_ϕ is trained taking as inputs 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's parameters ϕ are obtained by minimizing Eq. (5) where the loss function l is chosen to be the cross entropy loss function, widely used in classification.

A pictorial explanation of the PreTraNN training procedure is depicted in Fig. 3 while a complete specification of the autoencoder structure is reported in Appendix B.

D. Benchmark Methods

We compare the result of the proposed PreTraNN model with two state-of-the-art methods introduced above: the Gaussian mixture model (GMM) and a simple feed-forward neural network (FFNN).

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

The FFNN is instead trained over the readout signals dataset, taking as input the flattened vectors $\mathbf{X}^i =$

$[\mathbf{x}_I^i, \mathbf{x}_Q^i]$ and, as outputs, their labels \mathbf{y}^i . The architecture of the FFNN consists of two inner layers of dimension $L_{FF1} = 2d$ and $L_{FF2} = d$, with d the input dimension, and a output layer. The activation function are the *tanh* for the internal layer and the *softmax* for the output layer. The structure of the FFNN is the same of the second section of the PreTraNN. 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 directly the signals \mathbf{X}^i .

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. Datasets

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

We collect heterodyned readout signals for each qubit state state. Each measurement is obtained by preparing the device in states (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 the 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 $\omega_{IF} = 60 \text{ MHz}$, so its period is $1/\omega_{IF} \approx 16 \text{ ns}$. For this reason, in this work, we took a time window $\Delta t = 16 \text{ ns}$. 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 whose number of elements spans from 50 to 500, to study the efficiency of the classification methods in different configurations.

These measurements (an example of which is shown in panel (a) Fig. 1) are the two-dimensional $\mathbf{x}^i(t)$ trajectories that, flattened to form the \mathbf{X}^i inputs (See Sec. II C), will form the dataset for the PreTraNN and FFNN. The dataset 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 panel (b) Fig. 1). The dataset is then shuffled and split in train and test datasets in a 75% - 25% proportion.

The size of the dataset impacts the accuracy of the method and needs some consideration to avoid under-

fitting or unnecessarily long training times. Such considerations are drawn in Appendix A 2.

III. RESULTS

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

A. Two-state qubit readout

In this case the qubit is prepared and immediately measured in state $|0\rangle$ and $|1\rangle$. The dataset consists of 16000 readout signals (8000 for each state) and it is split into train and test subsets in 75%/25% proportions. Consideration on the choice of the dataset are drawn in Appendix A 1. The PreTraNN, FFNN and GMM setup is the one defined in Sec. II C and Sec. II D.

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. 4 and the global classification between state $|0\rangle$ and $|1\rangle$ in Fig. 5.

We start by considering Fig. 4. First of all, it can be noted that, for short measurements, the GMM displays a very bad classification capability. In fact it misses completely the labelling, assigning almost all values to state $|1\rangle$. This behaviour should be attributed to the fact that, for short measurement times, the points distributions heavily overlap preventing GMM from fitting them appropriately with two Gaussians (see also Fig. 6 for a illustrative example). A second thing that can be noted is that 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. This behaviour has a simple explanation: the qubit ground state (the $|0\rangle$ state) does not have leakage to the excited state (the $|1\rangle$ state) while the contrary is true. As a consequence, there

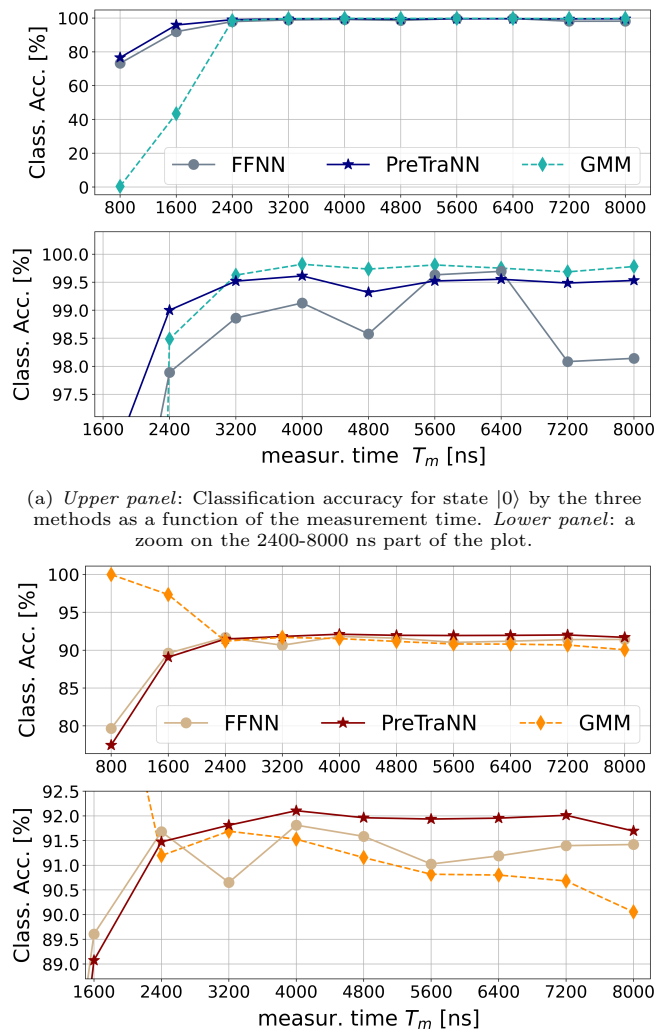


FIG. 4: Classification accuracy comparison, for state $|0\rangle$ and $|1\rangle$ separately, between Gaussian Mixture Model (GMM), the simple feed-forward neural network (FFNN) and the PreTraNN method. The readout time T_m spans from 800 ns to 8000 ns.

is an asymmetry in the data points distributions. This results in states prepared as $|1\rangle$ to be spotted on state $|0\rangle$ distribution due to decay process, while the reverse does not happen. Therefore the GMM, fitting the distribution with two Gaussians, can not handle this asymmetry performing very differently in the two cases. Lastly, PreTraNN method shows a very stable behaviour, while FFNN a fluctuating trend.

In Fig. 5, the global discrimination accuracy between state $|0\rangle$ and $|1\rangle$ is reported. It is obtained averaging the classification accuracies of $|0\rangle$ and $|1\rangle$ states case. The PreTraNN method outperforms the GMM and the FFNN methods for every measurement time. It is interesting to note that, after 2400 ns, as in the previous figure, the FFNN shows an oscillating behaviour and in

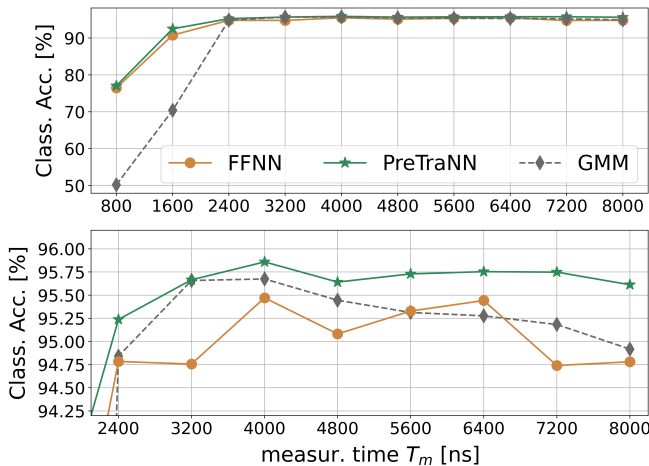


FIG. 5: Global classification accuracy between state $|0\rangle$ and $|1\rangle$ for increasing measurement time T_m . The accuracy obtained with PreTraNN method is higher (or at most equal) to the ones obtained with GMM and FFNN.

general worse results than the other methods, making this method unreliable. We can speculate that this behaviour 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 FFNN is not completely adequate for this purpose. Concerning the GMM, it is reasonable to assume that the improvement we obtain with PreTraNN stems from its greater complexity. Unlike GMM, PreTraNN takes into account the whole measurement "history". Hence, more information to exploit for the classification is available. It is interesting to note that our method also performs better than FFNN although it also considers the entire history of the measurements. This shows that the autoencoder pre-training over input data helps to improve the performance of the classification. In fact we stress again that the FFNN and the PreTraNN's second section are structurally the same. The only difference is that the FFNN classify directly the qubit readout signals, \mathbf{X}^i , while PreTraNN's second section classifies the encoded version of them, $\mathbf{h}^i = f_{\theta^e}$. The feature extraction capability of autoencoders helps to make the classification accuracy higher and more independent of measurement time.

It can also be noted that GMM accuracy has a global maximum at 4000 ns. As mentioned before, for the GMM to work well, the distribution of I-Q points for each qubit state must be as Gaussian as possible. It happens that, for short measurement times, the points distributions overlap since the qubit-resonator response is still in a transient state, while, for long times, decaying processes come into play which make the distribution skewed. Therefore, we can deduce that the length of 4000 ns produces the "most Gaussian" point distribution that allows the GMM to reach the greatest accuracy. This measurement time is therefore the one that should be set for the readout. The PreTraNN method makes less strict the need of this adjustment since it works well for a larger

of such experimental parameters T_m . In general can be seen that, in PreTraNN method, the classification accuracy is only increasing or constant. As a consequence, the trimming is faster and easier since need of finding the maximum accuracy is removed.

We want also 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 was not to present new hardware over-performing the current state-of-the-art one, but only to propose a method to improve readout in the present machines. Thus, interest was primarily focused on improving performance of a given machine from the software point of view.

The classification obtained with PreTraNN, not only improves the classification accuracy, but it also better reproduces the actual distribution of data. In Fig. 6 the comparison of the GMM and PreTraNN labelling result on data with different readout times is reported. The labelling for the FFNN is similar to the PreTraNN one, so it was omitted for clarity purposes. The first column shows data with the actual labels (represented by colours) as they were prepared in the quantum device. The second and third columns, instead, represent the same data but labelled according to GMM and PreTraNN, respectively. The same analysis is performed for short, medium and long times (rows of the figure). As anticipated, we conclude again that the GMM completely misses the classification for short times while the PreTraNN provides a considerably more realistic and accurate classification. The two distributions of points overlapping can now be spotted again.

The exact labels show the asymmetry in the data distribution due to decay of the excited state: many $|1\rangle$ -labelled point lay in $|0\rangle$ 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 state $|1\rangle$ on the machine.

Another important measure to take into account is the confusion matrix, which help to visualize the classification performances of the three method in comparison. In Fig. 7 are reported the confusion matrices for the three methods in three different measurement length setups. Each row reports the confusion matrices of the three models for a specific measurement length. Clearly, the best confusion matrices are those obtained for long times and with PreTraNN model.

2. Autoencoder features

In this section we give examples of the two important autoencoders features: input regeneration and latent space values. Fig. 8 shows an example of 3200 ns

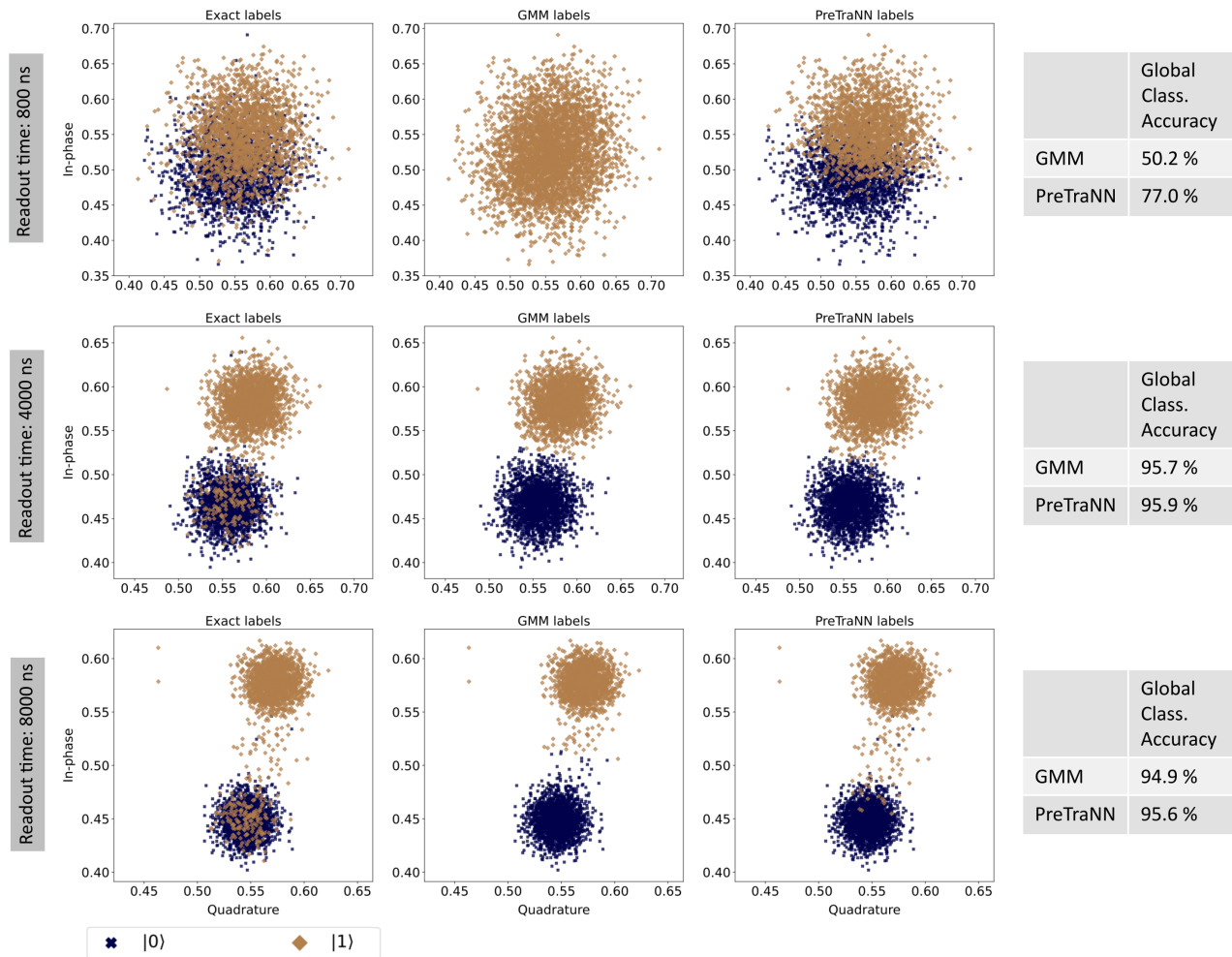


FIG. 6: Pictorial representation of the dataset with exact, GMM's and PreTraNN's labelling. Each point is the time average of the $I(t)$ and $Q(t)$ signals. The actual label, i.e. the prepared state, is represented in the first column. The GMM and PreTraNN methods labels are represented in the second and third columns.

(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. 9. The thin coloured 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 0 on average but those for $|0\rangle$ have larger fluctuations and a bit of structure. In particular, in both plots specific points where all the \mathbf{h}^i vectors follow definite trend (e.g., the points around 20 and 60 for state $|0\rangle$) can be spotted. These differences allows the increase in classification performance shown in this paper.

One might wonder how 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 slightly only one of its values, and finally we plug the modified latent vector into the decoder to obtain its "reconstruction". We do this several times by varying slightly the input each time. Fig. 10 depicts the result of this procedure. The thick lines represented 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 slowly varying this value, we obtain a slowly varying family of reconstructions.

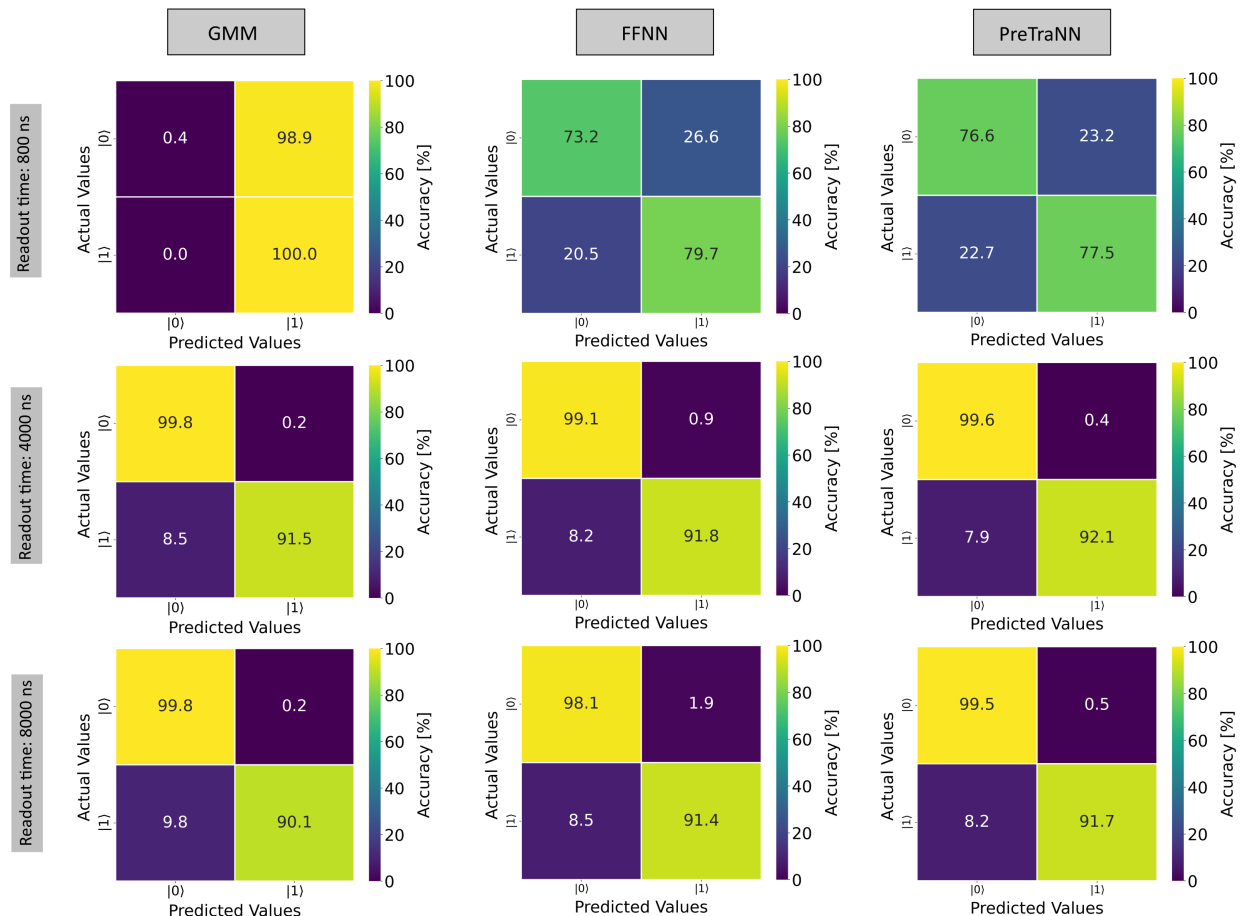


FIG. 7: Confusion matrices for classification between states $|0\rangle$ and $|1\rangle$ for the three methods for short, medium and long readout times.

3. Computational cost and scaling

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

The training for every neural network is performed with the "early stopping" approach to avoid over- or under-fitting. 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. In Fig. 11 are reported the results. The upper table shows the training time of each model with respect of readout length T_m for a 16000 elements dataset, the lower table, instead, represents the average time for a single input classification for each method. In both cases the times are represented in logarithmic scale to better spot trend. Times are reported in seconds and refer to a mid-range laptop computer with 4 cores and 8 GB of RAM.

Considering the training time, it can be noted that PreTraNN method takes a significantly longer time than GMM (from 2 to 3 orders of magnitude) but not much

more than the FFNN, despite the two training stages of the PreTraNN. As one might expect, the training time of non-GMM methods increases as the inputs measurement time increases. In fact, long measurement times correspond to wider neural networks and, therefore, longer optimizations.

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

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

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

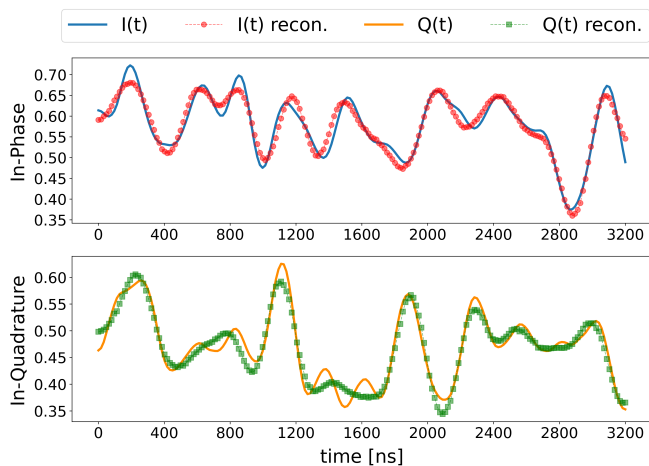


FIG. 8: An example of input regeneration made by the autoencoder. In both panels the solid lines represent the measurement signal divided in its two quadratures, respectively In-phase (I) and In-Quadrature (Q). The lines with markers, instead, represent the input reconstruction made by the autoencoder.

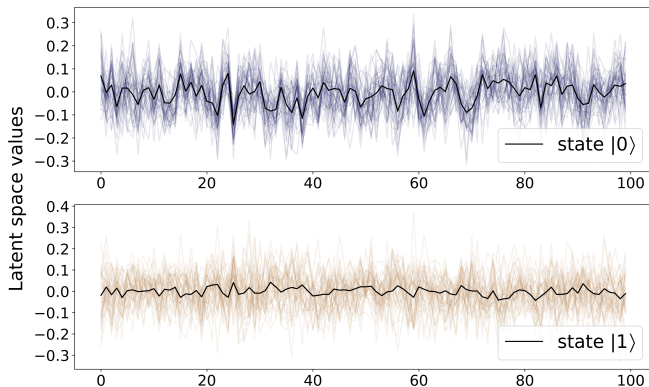


FIG. 9: Representation of latent space of the autoencoder for state $|0\rangle$ (upper) and $|1\rangle$ state (lower). In both panels, the coloured lines are the latent space representation (i.e. \mathbf{h}^i vector) of inputs for state $|0\rangle$ or $|1\rangle$. The solid black lines represent instead the average of these values.

On the classification time side, instead, 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 one instead needs a real-time or online readout on the machine, the classification times must be below the qubit lifetime. Since state-of-the-art superconducting transmon qubits have a lifetime of 200-500 microseconds [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 have, 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 FPGA or an ASIC imple-

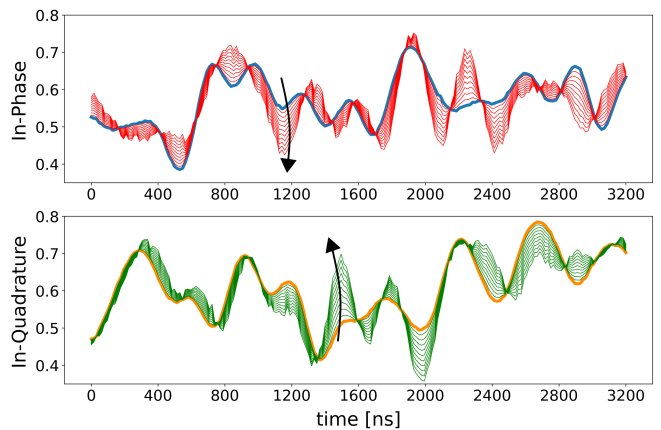


FIG. 10: The figure depicts an example of how the input reconstruction varies if a single value of the latent representation is varied slightly. Upper panel represent the in-phase component, lower panel the in quadrature one. In both panels the thick lines are the original "correct" input reconstruction, the thin lines represent the reconstructions obtained by slowly varying a value of the latent representation. In both panels, arrows are used to indicate the direction of changes induced by increasing the latent value.

input batch size	1	100	10000
Classif. time PreTraNN [s]	0.04200	0.04300	0.22400
Classif. time GMM [s]	0.00012	0.00013	0.00043

TABLE I: Classification times for PreTraNN and GMM as function of inputs batch size. Every reported time is the result of an average of 100 experiment. The FFNN method is not reported because its behaviour follows PreTraNN's.

mentation could improve even more the efficiency of the classification step or also improve the training process implementing it in a online way. See Ref. [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 a good accuracy for short measurements compared to 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 of the GMM (4000 ns) while achieving the same classification accuracy. The PreTraNN performs well regardless of readout time, allowing one to potentially skip the readout time T_m trimming. Moreover, this method can be utilized for usual two-level qubits or, conversely, extended to arbitrary numbers of levels or qubits with slight modifications in its structure and simply using different datasets. All this considered, the proposed method offers a promising approach to exploit short measurements that disturb the

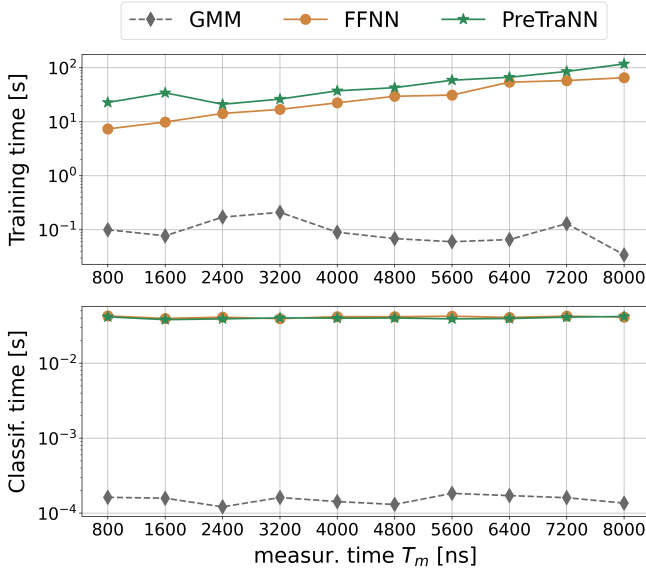


FIG. 11: Training and classification times for GMM, FFNN and PreTraNN methods. The times are reported in seconds for a middle range laptop computer. *Upper panel*: Training time in function of the measurement time (i.e. the length of the inputs). *Lower panel*: Classification time. The average time of GMM is 0.00013 second, for FFNN is 0.039 seconds and PreTraNN 0.037 seconds.

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 $|0\rangle, |1\rangle$ and $|2\rangle$ state and store the obtained data. The whole dataset consists of 24000 element (8000 for each state) divided into 75%

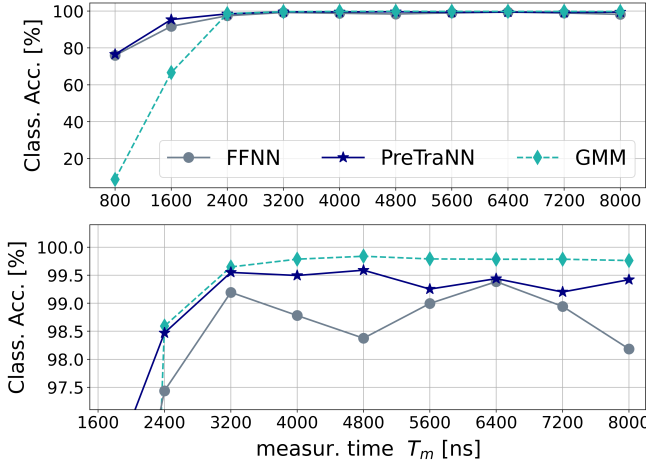


FIG. 12: *Upper panel*: State $|0\rangle$ classification accuracy for the three methods as a function of the measurement time in the case of a qutrit. *Lower panel*: a zoom on the medium-long times.

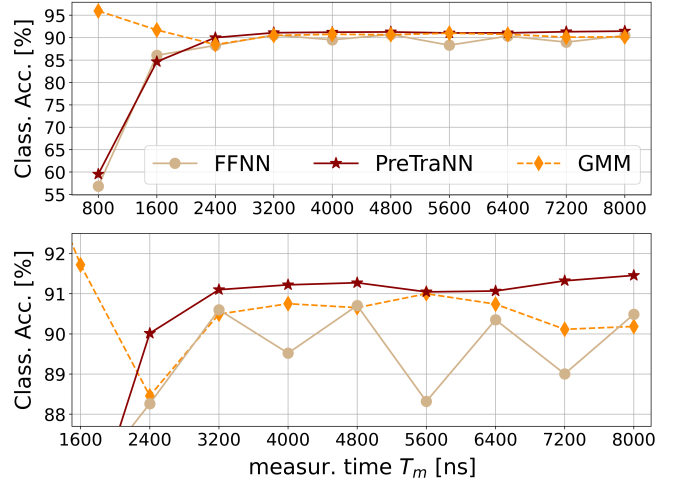


FIG. 13: *Upper panel*: State $|1\rangle$ classification accuracy for the three methods as a function of the measurement time in the case of a qutrit. *Lower panel*: a zoom on the medium-long times.

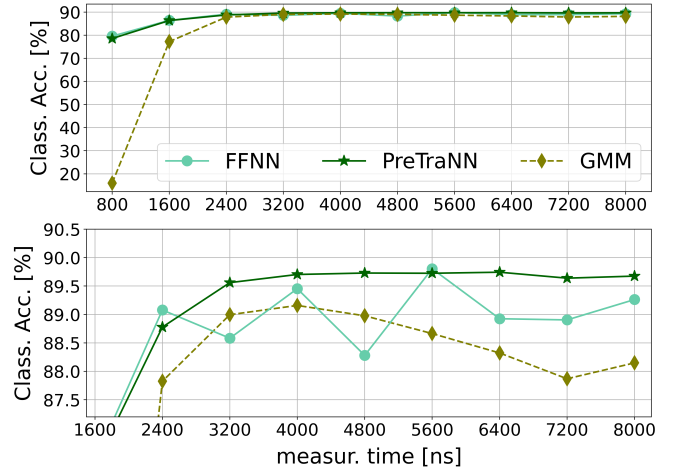


FIG. 14: *Upper panel*: State $|2\rangle$ classification accuracy for the three methods as a function of the measurement time. *Lower panel*: a zoom on the medium-long times.

train data and 25% test data. Again, for consideration on how the dataset is chosen, see. Sec. II C. The architecture of the models is the same as in the previous case (and as defined in Sec. II C and Sec. II F). The only difference between the two cases is the number of classes in the dataset. This allows to show the good scaling properties of the model.

1. Classification accuracy

In this paragraph the global and the state-by-state classification accuracy is reported. In Fig. 12,13,14 we show the classification accuracy for, respectively, state $|0\rangle, |1\rangle$ and $|2\rangle$. The lower panel of each figure is a zoom on the 2400-8000 ns part of the plot to better see the details. Even in this configuration we can see the same

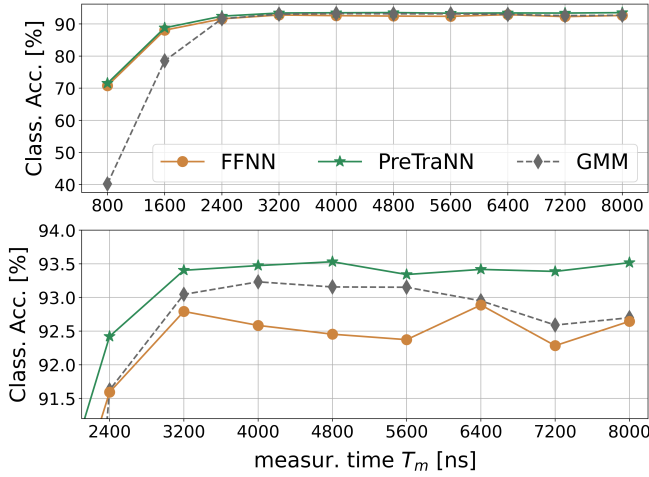


FIG. 15: Global classification accuracy for $|0\rangle, |1\rangle$ and $|2\rangle$ states classification for a qutrit.

trends as in the 2-level case. The GMM misclassifies for low times, and the FFNN still exhibits a seesaw pattern that makes it poorly suited to the task. Again, GMM performs better than PreTraNN in state $|0\rangle$ and worse in state $|1\rangle$ classification due to the data distribution asymmetry.

In Fig. 15 we present, instead, results for the global accuracy. The PreTraNN method achieves better classification performance for every measurement time. Again the GMM accuracy presents a 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 datasets.

We can also study the performances of PreTraNN as a function of the number of qudit levels. This will give us an idea on how the method scales. In Fig. 16 we report the difference in percentage points (p.p.) of the PreTraNN global accuracy with respect to GMM's for the two and three level cases for every measurement time T_m . Each point is the difference in p.p. between the global classification accuracy of the PreTraNN and GMM methods, for a particular T_m . The lower panel zooms on the middle and long times range. On the right panels the average difference for all T_m is highlighted. An increasing value of this difference, as the levels of the system increase, suggest a possible increasing advantage in use the PreTraNN method for increasing system levels. In this case we see that, at least considering medium and long times, this trend can be clearly seen. Considering all measurement times this trend is inverted but we have to consider that the short measurement time values for GMM are not particularly representative. Fig. 17, instead, reports the same calculation referred to FFNN method. Here the trend is clear for both the whole set of measurement times and the medium and long range.

This analysis suggests that there is a marginal increase

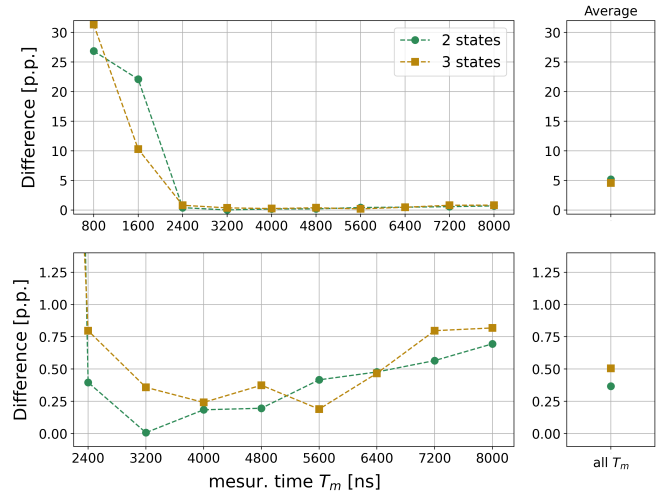


FIG. 16: Difference in percentage points [p.p.] between the accuracy of PreTraNN and GMM for the qubit and qutrit cases for different measurement time T_m . Lower panel report the analysis only for medium-long times. The small panels on the right show the average difference for all values of the respective plot on the left.

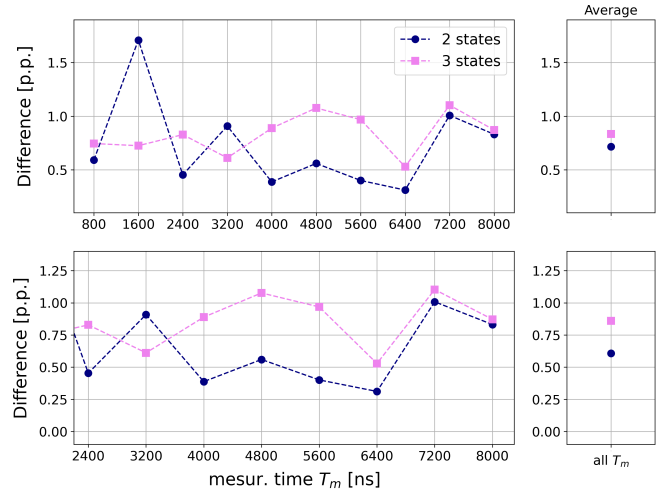


FIG. 17: Difference in percentage points [p.p.] between the accuracy of PreTraNN and FFNN for the 2 or 3 qubit state case. Lower panel report the analysis only for medium-long times. The small panels on the right show the average value of respective plot on the left.

in the effectiveness of PreTraNN compared to the other two methods as the classes of the dataset increase (i.e., as the dataset complexity rises). In other word, the difference in the global classification accuracy between PreTraNN and GMM or between PreTraNN and FFNN is bigger, on average, in the case of the three classes dataset, corresponding to qutrit readout data.

This analysis, although limited to 2 and 3 classes problem, suggests that the PreTraNN method should scale well as the qudit dimension increase. We can assume that it also scales well with the number of qubits but

further analysis is needed to better characterize the performance.

Furthermore, PreTraNN requires only minimal structural modifications for different qudit dimensions. One only need to adjust the number of output nodes in the last stage of the network and to use an appropriate dataset with a different number of classes. While the training times rise due to the increased dataset size (training time grows linearly with the dataset dimension), the classification time remains the same as the previous 2-state case.

IV. CONCLUSION

This work demonstrates that a feed-forward neural network with autoencoder pre-training allows for a robust qubit readout classification scheme with high accuracy and low dependence on the experimental device feature values. It allows for a consistent classification performance even for short readout times, unlike the more traditional schemes affected by the overlapping measurement results. It obtain good results also for longer measurement time where GMM method decrease its efficiency 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

speedup.

More importantly, because of shorter readout times, the measurement speedup increases. This property 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 was shown that the method performs well for all measurement times, helping in increasing classification result from a software point of view. On the other side, the classification times for a single measure are higher than 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 obtaining marginally increasing performances.

V. ACKNOWLEDGMENTS

This research was partially supported by Q@TN grants ML-QForge (PL). The Lawrence Livermore work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 with support from Laboratory Directed Research and Development grant 19-DR-005 LLNL-JRNL-842516. P.E.T. acknowledges the Q@TN consortium for his support

-
- [1] R. Barends, L. Lamata, J. Kelly, L. García-Álvarez, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, *et al.*, *Nat. Commun.* **6**, 7654 (2015).
 - [2] G. Wendin, *Rep. Prog. Phys.* **80**, 106001 (2017).
 - [3] Z. Yan, Y. R. Zhang, M. Gong, Y. Wu, Y. Zheng, S. Li, C. Wang, F. Liang, J. Lin, Y. Xu, *et al.*, *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, *Phys. Rev. A* **101**, 062307 (2020).
 - [5] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, *et al.*, *Nature* **574**, 505 (2019).
 - [6] A. Nersisyan, S. Poletto, N. Alidoust, R. Manenti, R. Renzas, C.-V. Bui, K. Vu, T. Whyland, Y. Mohan, E. A. Sete, *et al.*, “Manufacturing low dissipation superconducting quantum processors,” in *2019 IEEE International Electron Devices Meeting (IEDM)* (2019) pp. 31–1.
 - [7] L. B. Nguyen, Y.-H. Lin, A. Somoroff, R. Mencia, N. Grabon, and V. E. Manucharyan, *Phys. Rev X* **9**, 041041 (2019).
 - [8] H. L. Huang, D. Wu, D. Fan, and X. Zhu, *Sci. Chi. Info. Sci.* **63**, 1 (2020).
 - [9] J. Werschnik and E. Gross, *Jour. Phys. B* **40**, R175 (2007).
 - [10] J. P. Palao and R. Kosloff, *Phys. Rev. Lett.* **89**, 188301 (2002).
 - [11] S. Kirchhoff, T. Keßler, P. J. Liebermann, E. Assémat, S. Machnes, F. Motzoi, and F. K. Wilhelm, *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, *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, *Nature Comm.* **6**, 1 (2015).
 - [14] M. Gong, X. Yuan, S. Wang, Y. Wu, Y. Zhao, C. Zha, S. Li, Z. Zhang, Q. Zhao, Y. Liu, F. Liang, J. Lin, Y. Xu, H. Deng, H. Rong, H. Lu, S. C. Benjamin, C. Z. Peng, X. Ma, Y. A. Chen, X. Zhu, and J.-W. Pan, *Nat. Sci. Rev.* **9** (2021), 10.1093/nsr/nwab011.
 - [15] A. Roggero and A. Baroni, *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, *et al.*, *Phys. Rev. App.* **7**, 054020 (2017).
 - [17] Y. Sunada, S. Kono, J. Ilves, S. Tamate, T. Sugiyama, Y. Tabuchi, and Y. Nakamura, *Phys. Rev. App.* **17**, 044016 (2022).
 - [18] A. P. Place, L. V. Rodgers, P. Mundada, B. M. Smitham, M. Fitzpatrick, Z. Leng, A. Premkumar, J. Bryon, A. Vrajitoarea, S. Sussman, *et al.*, *Nature Comm.* **12**, 1 (2021).

- [19] A. Blais, R.-S. Huang, A. Wallraff, S. M. Girvin, and R. J. Schoelkopf, *Phys. Rev. A* **69**, 062320 (2004).
- [20] A. Wallraff, D. Schuster, A. Blais, L. Frunzio, J. Majer, M. Devoret, S. Girvin, and R. Schoelkopf, *Phys. Rev. Lett.* **95**, 060501 (2005).
- [21] R. Bianchetti, S. Filipp, M. Baur, J. Fink, M. Göppl, P. J. Leek, L. Steffen, A. Blais, and A. Wallraff, *Phys. Rev. A* **80**, 043840 (2009).
- [22] J. Gambetta, A. Blais, M. Boissonneault, A. A. Houck, D. Schuster, and S. M. Girvin, *Phys. Rev. A* **77**, 012112 (2008).
- [23] D. A. Reynolds, *Encycl. Biometr.* **741** (2009).
- [24] E. Magesan, J. M. Gambetta, A. D. Córcoles, and J. M. Chow, *Phys. Rev. Lett.* **114**, 200501 (2015).
- [25] A. Seif, K. A. Landsman, N. M. Linke, C. Figgatt, C. Monroe, and M. Hafezi, *Jour. Phys. B* **51**, 174006 (2018).
- [26] B. Lienhard, A. Vepsäläinen, L. C. 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, *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, 2021) pp. 481–482.
- [28] L. A. Martinez, Y. J. Rosen, and J. L. DuBois, *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, *Phys. Rev. Lett.* **125**, 170502 (2020).
- [30] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, Vol. 4 (Springer, 2006).
- [31] Y. Bengio, A. Courville, and P. Vincent, *IEEE Trans. Patt. Anal. and Mach. Intell.* **35**, 1798 (2013).
- [32] L. Pasa and A. Sperduti, *Adva. Neur. Infor. Process. Syst.* **27** (2014).
- [33] B. T. Ong, K. Sugiura, and K. Zettsu, in *2014 IEEE International Conference on Big Data (Big Data)* (IEEE, 2014) pp. 760–765.
- [34] L. Chen, H.-X. Li, Y. Lu, C. W. Warren, C. J. Krizan, S. Kosen, M. Rommel, S. Ahmed, A. Osman, J. Biznárová, *et al.*, *arXiv preprint arXiv:2208.05879* (2022), 10.48550/arXiv.2208.05879.
- [35] S. Kohler, *Phys. Rev. A* **98**, 023849 (2018).
- [36] “Quantum orchestration platform,” (2021).
- [37] X. Yu, M. O. Efe, and O. Kaynak, *IEEE Transac. Neur. Netw.* **13**, 251 (2002).
- [38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016) <http://www.deeplearningbook.org>.
- [39] A. Shrestha and A. Mahmood, *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, *An. Rev. Cond. Matt. Phys.* **11**, 369 (2020).
- [41] I. Westby, X. Yang, T. Liu, and H. Xu, *Jour. Supercomput.* **77**, 14356 (2021).
- [42] R. Sarić, D. Jokić, N. Beganović, L. G. Pokvić, and A. Badnjević, *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, 2010) pp. 1050–1054.
- [44] S. Gandhare and B. Karthikeyan, in *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)* (IEEE, 2019) pp. 1–4.
- [45] “Keras website,” <https://keras.io/>.
- [46] “Keras website,” <https://scikit-learn.org/stable/index.html>.
- [47] D. P. Kingma and J. Ba, *arXiv preprint arXiv:1412.6980* (2014).

Appendix A: Numerical Consideration on Autoencoder

1. Autoencoder’s latent space dimension

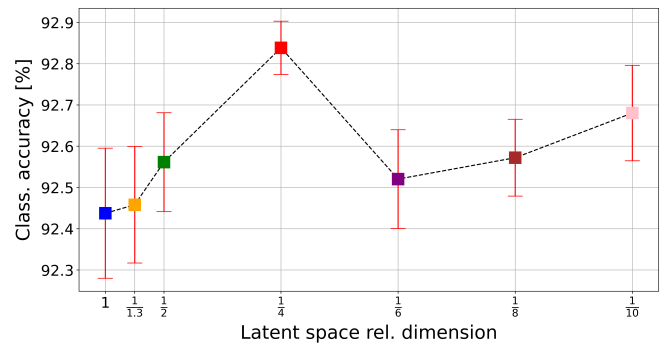


FIG. 18: PreTraNN global classification accuracy for the 3-state case with 2400 ns readout inputs as a function of the latent space dimension. The higher accuracy is reached at 1/4 the input dimension.

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

In the case of the autoencoder, the main parameter is the autoencoder’s latent space size. In principle, a latent space that is too small is not sufficient to perform expressive encoding, while too large of a latent space increases the computational cost without extracting in a compact way information from the dataset. 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 took the PreTraNN with trajectories of 2400 ns (150 time-steps of 16 ns, i.e. inputs dimension of 300 values), and trained it for different values of latent space. We started from a latent dimension equal to the input dimension and gradually went down till one-tenth of it. The dimension

of the other two inner layers was set linearly interpolating between the size of the input and latent space. The decoder had the same structure but reversed. Contextually, three properties of PreTraNN were studied as a function of latent dimension: the global classification accuracy, the autoencoder training loss and autoencoder training time. To obtain more consistent results, for each latent dimension the training was repeated 10 times with different samplings of the dataset and the properties values was averaged.

In Fig. 18 the PreTraNN global classification accuracy for decreasing latent space dimension is reported. 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 whose size is one-fourth 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.

In Fig.19 is represented the loss function values (mean squared error) during the training of the autoencoder for different latent space dimensions. 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, 1/8 the inputs size), on the other hand, convergence stalls at much higher values of loss function. Thus the best values are 1/2, 1/4 and 1/6 of the input length.

In Fig. 20 the training time in seconds is reported. 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 behaviour, we can choose the latent space dimension making a trade-off between the reported metrics. The value which maximize the classification accuracy having at the same time good loss func-

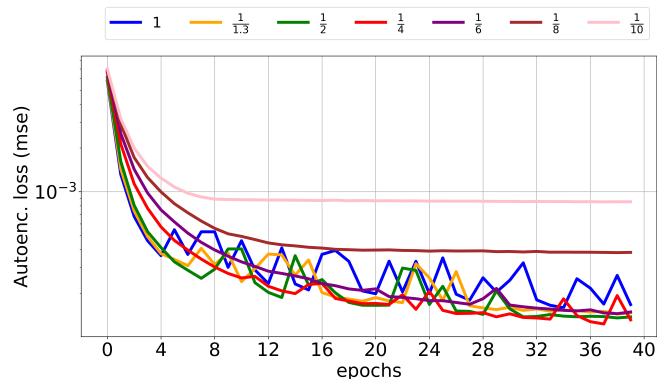


FIG. 19: Autoencoder training loss function as function of training epochs for different latent space relative dimension. To large latent dimension (1, 1/1.3 1/2 times the input size) present a fluctuating behaviour and are useless for feature extraction while to small latent dimension do not allow effective encoding and their loss function remains high (1/8 and 1/10 the input size).

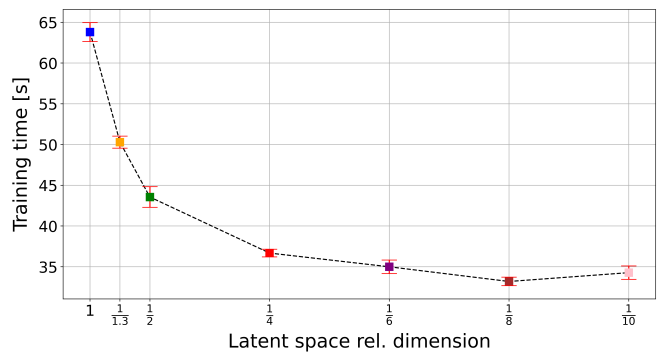


FIG. 20: Autoencoder training time in function of latent space relative dimension. Clearly larger latent spaces correspond to neural networks with more parameters and thus longer training times.

tion convergence and (relatively) short training time is a latent dimension of 1/4 the inputs size. This is the value chosen to carry out the analysis in this work. The dimension of the 2 internal layers is set linearly interpolating between the latent space and the input dimensions.

2. Dataset size and convergence

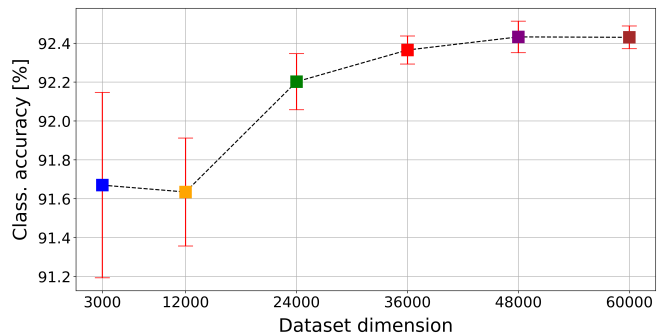


FIG. 21: Global classification accuracy of the PreTraNN as function of the number of dataset elements.

In order to obtain a good training convergence which maximize the classification accuracy an adequate dataset is needed. Small datasets are fast to train but usually produce inadequate classification accuracies, while large ones have the opposite behaviour. At the same time, the growth of the classification accuracy capability is marginally decreasing with increasing dataset size. Here we report some analysis on the behaviour of the PreTraNN in function of the dataset dimension studying the same three properties introduced in the previous section i.e. loss function, classification accuracy and training time. Even in this case we took the PreTraNN with 2400 ns measurement signals (150 time-steps of 16 ns, i.e. inputs dimension of 300 values) with a latent space of 75 neurons, and trained it for different dataset dimension. We started from training dataset of 3000 elements (1000

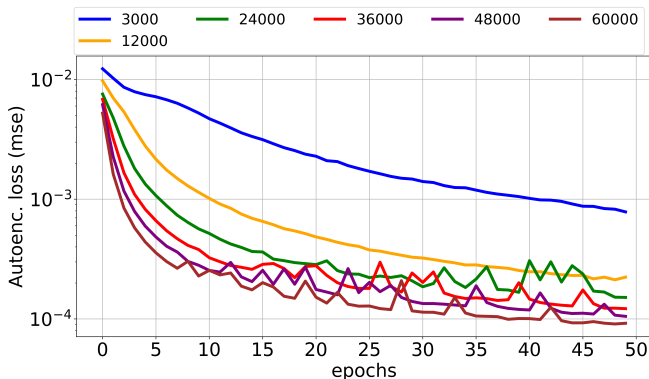


FIG. 22: Autoencoder loss (mean square error) as function of the epochs for increasing dataset size.

element for each class) and gradually increase its dimension till 60000 elements (with a 75% of them dedicated to training). For each dataset dimension the training was repeated 10 times with different sampling of the dataset and the properties values averaged.

In Fig. 21 the global classification accuracy as function of the dataset size is reported. It can be seen that the accuracy increases as the dataset grows even if with decreasing speed.

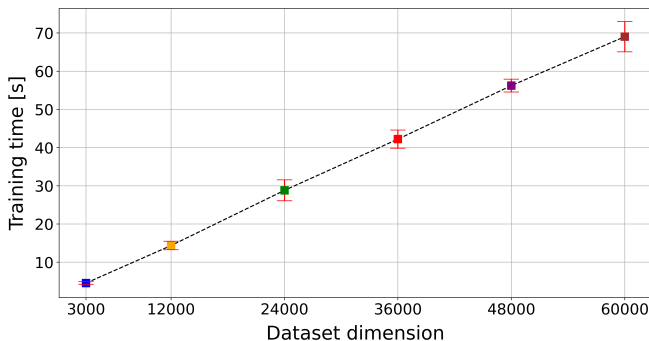


FIG. 23: Training time for increasing dataset dimension.

In Fig. 22 is represented the loss function values (mean squared error) during the training of the autoencoder for different configurations. The trend is quite neat. The larger is the dataset the better is the convergence, although for large data sets the convergence becomes more unstable.

In Fig. 23 the training time in seconds is reported. As expected, the training time increase linearly with the dataset dimension. A short training time is preferable.

Given these results, the trade-off between accuracy, loss function, and training time, in order to maximize effectiveness and minimize cost, was identified in the 24000-item dataset for the 3-states case and the 16000-item dataset for 2-states case.

	Layer	Size	Activ. funct.	Keras type
Encoder	input	L	sigmoid	Dense
	1 th hidden	L3/4	tanh	Dense
	2 nd hidden	L2/4	tanh	Dense
	latent	L/4	tanh	Dense
Decoder	1 th hidden	L2/4	tanh	Dense
	2 nd hidden	L3/4	tanh	Dense
	output	L	sigmoid	Dense

TABLE II: Autoencoder’s specifications. The ”Size” column represent the number of neurons for each layer in fraction of the input dimension L . The ”Keras type” column reports the type of keras layer employed.

Appendix B: Models specifications

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

In this work the neural networks building and training is performed via the python package *Keras* [45]. For the GMM instead the *sklearn* python package [46].

a. Autoencoder In every configuration employed in this work, the encoder is composed by 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 mirrored. So 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 implemented with the *Dense* layer specification in Keras. In Tab. II all the information on the network are reported.

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*. The training is performed with the *Early Stopping* procedure that stops the training if the loss does not decrease for two epoch in a row.

b. FFNN and PreTraNN’s second stage The second stage of the PreTraNN and the is a simple feed-forward neural network. It is composed by a input layer (of the same dimension of the latent layer of 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 we are doing the classification with. Hence, $C = 2$ for qubit classification of Sec. III A, while $C = 3$ for qutrit classification of Sec. III B. The connectivity between the neurons is full. The optimization algorithm is the *Adam*. The loss function is the the *cross-entropy*, suitable for classification purpose. The training is performed with the *Early Stopping* procedure that stops the training if the loss does not decrease for two epoch in a row. Other information are summarized in Tab. III. The structure of FFNN model is the same but with a number of input neurons equal to the dataset dimension instead of the latent layer dimension.

Layer	Size	Activ. funct.	Keras type
Input	$L/4$ (L)	tanh	Dense
1^{th} hidden	$L2/4$ ($2L$)	tanh	Dense
2^{nd} hidden	$L/4$ (L)	tanh	Dense
Output	C	softmax	Dense

TABLE III: Structure and specifications of PreTraNN’s second section (FFNN) network with Keras. L is the dataset inputs length, C is the dimension of the output layer which change based on the number of classes.

c. Gaussian Mixture Model The GMM is implemented with *sklearn* package with the standard build-in parameters specifying only the number of classes of the inputs dataset.