Quantum Information Processing (2023) 22:151
https://doi.org/10.1007/s11128-023-03898-2

Convolutional neural network based decoders for surface codes

Simone Bordoni[1] · **Stefano Giagu**[1]

Received: 25 March 2022 / Accepted: 27 February 2023 / Published online: 19 March 2023

Abstract
The decoding of error syndromes of surface codes with classical algorithms may slow down quantum computation. To overcome this problem it is possible to implement decoding algorithms based on artificial neural networks. This work reports a study of decoders based on convolutional neural networks, tested on different code distances and noise models. The results show that decoders based on convolutional neural networks have good performance and can adapt to different noise models. Moreover, explainable machine learning techniques have been applied to the neural network of the decoder to better understand the behaviour and errors of the algorithm, in order to produce a more robust and performing algorithm.

Keywords Quantum computing · Surface codes · Quantum error correction · Machine learning · Artificial neural networks · Quantum machine learning

Abbreviations

QECC	Quantum error correction code
MWPM	Minimum weight perfect matching
HLD	High level decoder
CNN	Convolutional neural network
FFNN	Feed forward neural network

1 Introduction

In recent years a lot of interest has grown around the possibility of constructing efficient quantum computers. One of the main problems regards the protection of quantum

✉ Simone Bordoni
simone.bordoni@uniroma1.it

✉ Stefano Giagu
stefano.giagu@uniroma1.it

[1] Dipartimento di Fisica, La Sapienza Università di Roma, Piazzale Aldo Moro, 5, Rome, Italy

information from external noise. This process, known as decoherence, is due to the inevitable interaction between the qubits and the environment [1, 2].

A solution may come from quantum error correction codes (QECC), where logical quantum information is stored in the degrees of freedom of a system composed of many physical qubits. In this way, the logical information can be restored even in the presence of single physical qubit errors [3, 4].

Surface codes are a family of QECC [5, 6], their implementation is simpler, with respect to other codes, because only near interactions between the physical qubits is needed. In a surface code a greater number of physical qubits gives better protection from errors, however, this makes the decoding procedure more complicated. The decoding algorithm is implemented by a classical computer and should be carried out fast to not slow down the quantum computation. With classical algorithms it is difficult to obtain a reasonable execution time incrementing the number of physical qubits [7]. A solution to this problem may come from the application of artificial neural networks. Simple models of neural networks already give good decoding accuracy with a constant execution time for small surface codes [8]. In order to decode codes with a larger number of qubits, more complicated neural network models have to be employed. This work focuses on decoders based on convolutional neural networks for different code distances, noise models and error probabilities. To improve the performance of the algorithms different model architectures have been studied to find the best error probabilities for the training set and to reduce the number of trainable parameters of the neural network. In the last section of the article, to get a better understanding of the behaviour of neural network based decoders, explainable machine learning methods have been employed. An original technique to improve the decoder performance, based on data augmentation driven by the results of the model explainability, is also reported.

2 Surface codes

In a surface code physical qubits are arranged on a squared lattice; they are divided in two categories (Fig. 1). The *data qubits* store the quantum information. The *measurement qubits*, also called stabilizers, are used to perform projective measurements, on the nearest neighbour data qubits. Measurement qubits are divided into Z and X types, they are used to perform respectively measurements of the Pauli operators σ_x or σ_z [9]. To simplify the notation in the rest of this work σ_x and σ_z Pauli operators will be called X and Z operators. As all the stabilizers commute, after a measurement cycle the system collapses into an eigenstate of all the stabilizers [10, 11].

A surface code contains the logical information of a qubit. The logical state can be modified with logical operators. An operator of this kind has to preserve all the eigenvalues of the stabilizers [12]. On each border of a surface code there are only X measurement qubits or Z measurement qubits, these borders are called X sides or Z sides of the surface code respectively. Every chain of single qubit X operators, that connect the X sides, works as an X logical operator. This is also true, in the case of Z logical operators, for chains of Z single qubit operators that connect the Z sides. The minimum number of single qubit operators, necessary to change the logical state, is

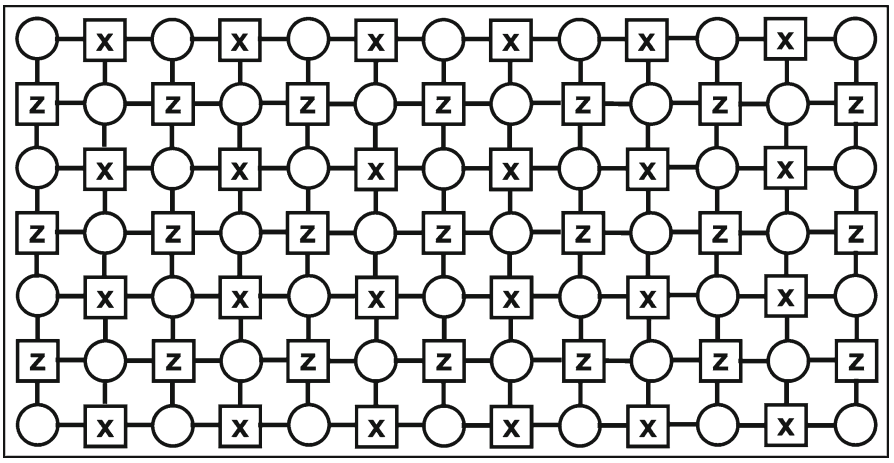

Fig. 1 Schematic representation of a surface code. Data qubits are represented as circles while X and Z measurement qubits are represented as squares. Each qubit is connected with its nearest neighbours

equal to the number of data qubits on a side of the lattice, this important value is called the distance of the code (d) [6]. Figure 2 shows some examples of logical operators in a $d = 5$ surface code.

2.1 Errors

Unlike the classical bit, where only bit flip errors may appear, for a qubit there is a continuous set of possible errors. However, after a measurement cycle, the state of the qubits collapses into a discrete set of possible errors [2–4]. There are three main kind of errors that appear on a surface code.

Data qubit errors are the classical bit flip (X errors), phase flip (Z errors) and a combination of both (Y errors). A single error of this type can be easily identified by observing a change of the eigenvalues in the neighbouring stabilizers (Fig 3).

Measurement errors appear when the projective measurement fails. A single error of this type changes just the value of the considered stabilizer.

Gate errors may appear during the measurement process when imperfect Hadamard and C-NOT gates are used. C-NOT gate errors are the most difficult to be identified because they affect two qubits.

It is possible to test different noise models (or error models) by including only some types of the errors previously described and by changing their probability. In this work many simulations of surface codes have been performed to test different code distances, error models and probabilities. For this purpose a specific python library has been employed[1]. The performance of a decoding algorithm can be very sensitive with respect to the considered error model and characteristics of the surface code [13–15].

In the *depolarising error model* only data qubit errors are considered. Given p the error probability, each data qubit is subject to an X, Z, or Y error with equal probability

[1] E. Villaseñor and B. Criger: https://github.com/evalvarez12/Distributed_Surface_Code.git

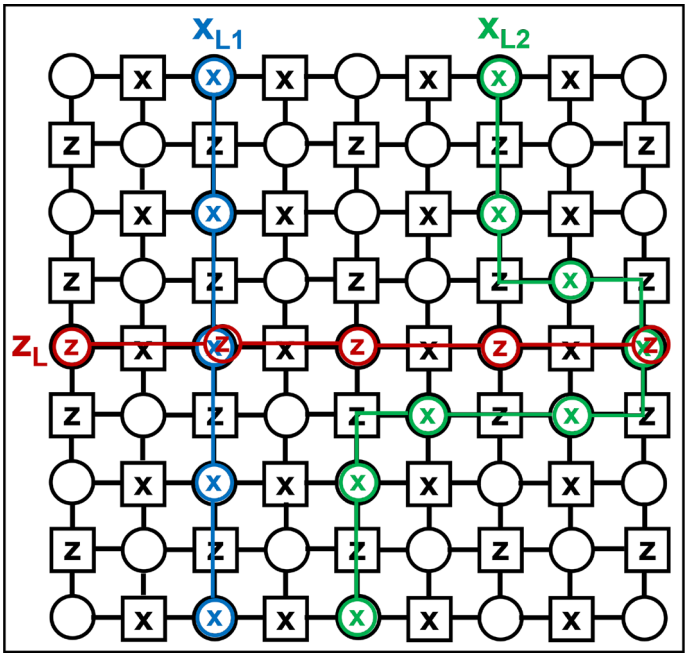

Fig. 2 Examples of logical operators in a $d = 5$ surface code. Z and X logical operators are composed of chains of single qubit Z or X operators connecting the Z or X sides respectively

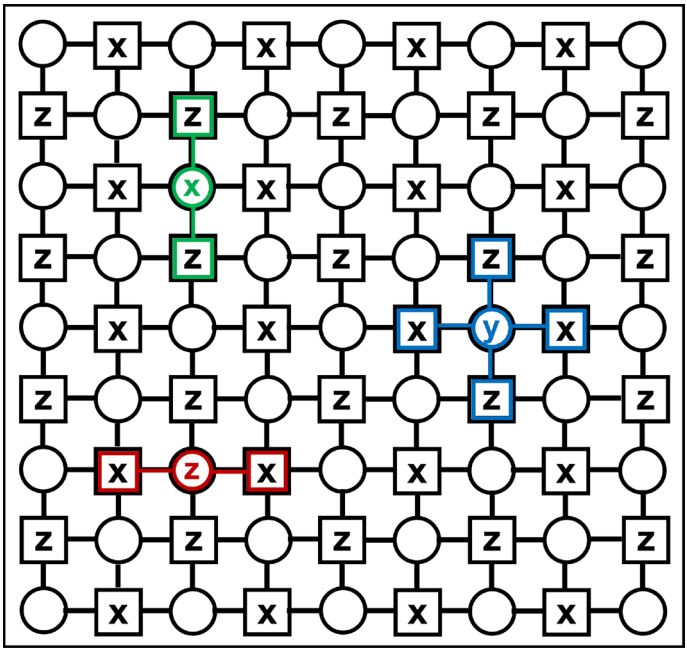

Fig. 3 Signal error produced by single data qubit errors. The highlighted measurement qubits change their measurement value

$p/3$. Perfect measurements and gates are assumed, thus only one measurement cycle is necessary.

In the *circuit noise model* also the measurement qubits are subject to errors and gates are not assumed perfect. In this case it is better to make more measurement cycles to identify measurement errors.

2.2 Decoding algorithms

In a QECC, the Hilbert space that describes the state of the physical qubits can be decomposed into a logical and an ancillary subspace [16]. After error correction the ancillary state, that increases the redundancy of information, is restored. This is not true for the logical state that may be modified, causing a logical error. The performance of a decoding algorithm can be tested by studying the rate at which logical errors appear, as a function of the single qubit error probability.

The minimum weight perfect matching algorithm (MWPM) restores the ancillary state with the minimum number of corrections [17–19]. This decoder is only nearly optimal for bit-flip noise [20] (independent X and Z errors), it represents a standard benchmark for other decoders. When studying the logical error rate with respect to the single qubit error rate a typical behaviour occurs. For low error probabilities, the decoding accuracy increases with the dimension of the code. On the other hand, for high error probabilities, increasing the distance of the code reduces the accuracy of the decoder [6, 14]. The cross-over of these two regimes occurs at the threshold error rate.

In order to use artificial neural networks, it is necessary to transform the original decoding problem in a classification problem. For this purpose, neural network based decoders are composed of two components:

The *simple decoder* analyses the error syndrome and proposes a correction that matches with the syndrome. This algorithm may be implemented by a neural network but it is simpler to use naive decoder. In this work the simple decoder corrects each error with a chain of operators that connects it to the nearest border. In this way the ancillary state of the system is restored.

The *high level decoder* (HLD) takes as input the error syndrome and tries to find out if the correction of the simple decoder has created a logical error. This is a classification problem that may be solved by a neural network.

3 Related works

Many different algorithms, based both on classical methods and machine learning techniques, have been tested for the decoding of surface codes. Apart from the MWPM previously introduced, examples of decoding algorithms not based on neural networks are the renormalization group decoder [21, 22], the cellular automaton [23], the maximum likelihood decoder [24] and the Markov chain Monte Carlo decoder [25]. For these algorithms it is difficult to find a good compromise between decoding accuracy and execution time [7]. As neural network based decoders show a good compromise

between accuracy and execution time, many studies have been carried out to test different neural network architectures. The first studies in this sector [26] show that, for small surface codes, neural network based decoders have a decoding performance, similar to MWPM, for a depolarising noise model. When measurement error and imperfect gates are included it is possible to improve the MWPM algorithm taking into account of more complex noise models [27, 28]. However neural network based decoders remain interesting because they require constant execution time and can easily adapt to different noise models. Some problems arise for high distance surface codes as the number of possible error syndromes increases exponentially. This means that, for the correct training of the HLD, the training set needs to be increased with the dimension of the code to contain the most statistically relevant errors.

To scale the methodology to higher distance surface codes, some interesting approaches have been proposed. A recurrent neural network architecture has been tested by Baireuther et al. [29] for the decoding of correlated errors. Torlai and Melko [30] have tested a decoder based on a stochastic neural network (Boltzmann machine) that is applicable to a wide variety of stabilizer codes. Other interesting studies on deep learning based decoders have been carried out by Krastanov and Jiang [31]. Varsamopoulos et al. [8] have compared decoders based on feed forward neural networks and on recurrent neural networks. Recent works have tested decoders based on distributed neural networks [7], in order to reduce the size of the training set for high distance surface codes. Another interesting novel idea comes from the application of machine learning techniques to an ensemble of classical decoders [32]. Other decoders based on machine learning have been recently tested by Bhoumik et al. [33]. A state of the art decoder, scalable to high distance surface codes, has been created by Meinerz et al. [34] combining convolutional neural networks (to preprocess local information) with a conventional algorithm. Other decoders based on convolutional neural networks have been tested on high distance surface codes for a depolarising or bit flip error models [35, 36].

While this work concentrates on smaller dimension surface codes, with respect to the previously mentioned studies, some new elements have been introduced. A more realistic noise model, that includes measurement errors, has been decoded with a convolutional neural network. The performance of the HLD have been studied with respect to the choice of the training set error probability. The dilated convolution technique has been tested to scale to higher distance codes and to reduce the number of trainable parameters of the neural network. Moreover, in order to understand better the cases where the neural network fails, explainable machine learning techniques have been applied to the HLD.

4 Convolutional neural network based decoders

In this section the characteristics of high level decoders based on feed forward neural networks (FFNN) and convolutional neural networks (CNN) will be tested. Two different noise models will be considered, the simple depolarising noise and a more complicated noise model where measurement errors and more measurement cycles are present. Before the comparison some preliminary studies are necessary for the tuning

Table 1 This table reports the optimal number of trainable weights (P) of the feed forward neural network obtained for different code distances (d) for a depolarising noise

d	P
7	5.67×10^5
9	6.4×10^6
11	3.9×10^6

Table 2 This table reports best hyperparameters of the CNN for different code distances and a depolarising error model

d	CL	DL	N	P
7	3	1	512	2.7×10^6
9	4	1	1024	8.0×10^6
11	6	2	1024	9.2×10^6

d is the code distance, CL is the number of convolutional layers, DL is the number of dense layers, N is the number of neurons in the dense layers and P is the total number of trainable weights

of the hyperparameters. Furthermore, to improve the accuracy of the HLD, convolutional neural networks based on dilated convolution layers, able to better capture local features at different spatial scales, will be tested in this section.

4.1 Depolarising noise

For the depolarising error model datasets of 5×10^6 elements have been generated with a single qubit error probability $p = 0.1$ and for code distances $d = 7, 9, 11$. All the neural networks employed, both FFNN and CNN, share the following characteristics. ReLU activation functions have been used in all hidden layers while Softmax activation functions have been used for the output layer. The loss function employed is categorical cross entropy. The ADAM optimiser has been used, with a mini-batch size of 32 elements. The neural networks have been trained for 20 epochs using 10% of the original dataset as evaluation set.

Feed forward neural networks with different numbers of layers and neurons have been tested on each dataset. The best results, for all the code distances, have been obtained using three hidden layers with a number of trainable weights reported in Table 1.

For the convolutional neural networks, the input format consists of squared matrices with a number of rows and columns equal to $2d - 1$, where d is the distance of the code. Each matrix element represents a qubit of the surface code. The value of the data qubits is set to zero, while the value of the measurement qubits is $\{1, -1\}$. The first parameters of the CNN to be defined are the number of filters and the number of convolutional and dense layers. For the number of filters the best results have been obtained using 64 kernels of dimension 3×3. The number of convolutional and dense layers that showed the best performance are reported in Table 2. Other fine tests have been carried out to improve the performance, for example a stride with value 2 has been tested for the first convolutional layer but this reduced the overall performance. A small increase in the accuracy has been obtained with a padding on the first layer.

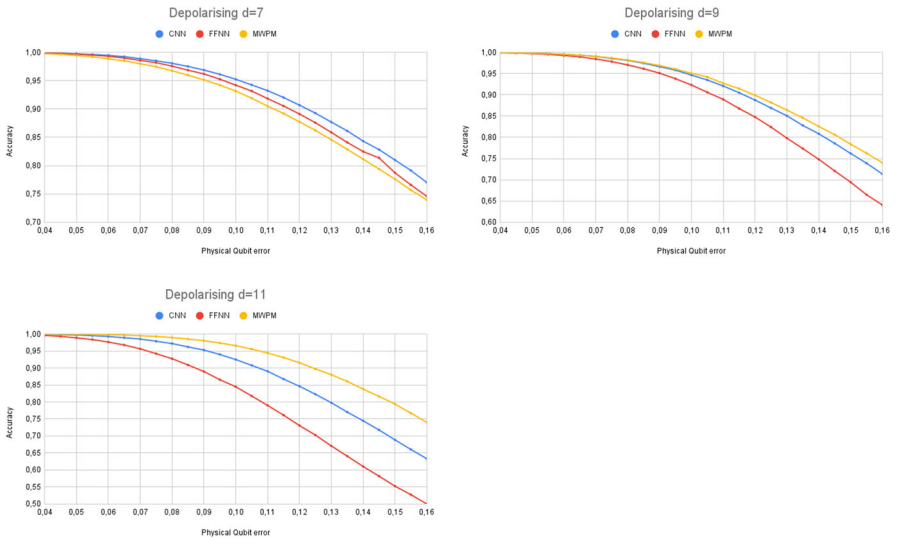

Fig. 4 Comparison between accuracy obtained with HLD based on FFNN and CNN and the accuracy obtained with MWPM for different code distances (d) and single qubit error probabilities (Physical Qubit error)

The accuracy obtained with the previously described models of neural networks is reported in Fig. 4 and has been compared to the accuracy obtained with MWPM for different code distances and single qubit error probabilities. The test on different error probabilities have been made on datasets of 2×10^5 elements. For $d = 7$ codes the performance of the high level decoder, both based on FFNN or CNN, is greater than MWPM. For greater distance codes the HLD has a worse performance than MWPM. This is due to the fact that the dimension of the training set is still too small to include the most statistically relevant error syndromes, necessary for a correct training of the network. However, for these code distances, it is possible to observe the advantage in the use of a CNN with respect to a FFNN.

4.2 Depolarising plus measurement errors

In order to reproduce a more realistic error model it is necessary to include measurement errors and more correction cycles. In this section we will employ the following noise model: before a measurement cycle a depolarising noise is applied with single qubit error probability $p = 0.01$. After applying the depolarising noise, an imperfect measurement cycle is performed. The measurement error probability has been set to $q = 0.01$. Three rounds of depolarising error plus imperfect measurements are carried out on each surface code. This error model is different with respect to the channel noise model where gate errors are included. However, this noise model is easier to simulate, and can be employed for an initial study of the behaviour of high level decoders based on CNN, when measurement errors and many correction cycles are present. The MWPM algorithm still works for a circuit level noise model with many measurement

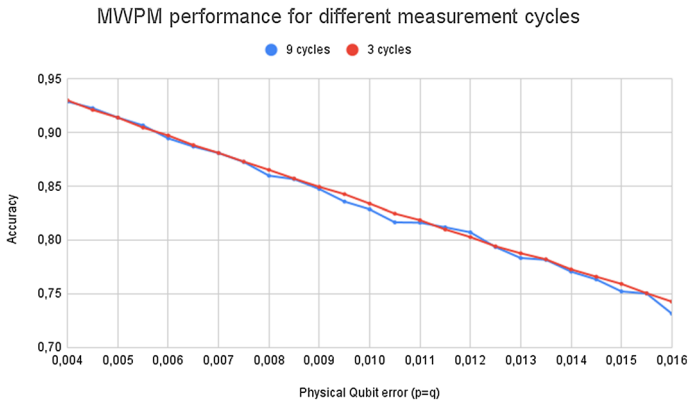

Fig. 5 Performance of MWPM for the depolarising plus measurement errors model using a different number or imperfect measurement cycles (3 and 9). The depolarising error probability (p) has been set equal to the measurement error probability (q). For this noise model a different number of measurement cycles does not modify significantly the decoding problem

cycles, in this case the error matches are carried out on a three dimensional graph [13]. As benchmark, in this section, we employed the simplest version of the MWPM algorithm that considers all measurements as perfect. The performance of this algorithm is sub-optimal but can be improved by adjusting the weights in the matching graphs [37].

It is important to notice that three measurement cycles is not the standard way to benchmark noise models with measurement errors, that is carried out using as many measurement cycles as the distance of the code. However, we have decided to reduce the number of measurement cycles in order to speed up the generation of the datasets thus obtaining more training samples. In fact, the time required to generate large training sets as well as their size was at the limit of the available computational resources, in particular in Sect. 5 where many training sets with different error probabilities have been employed. Reducing the number of imperfect measurement cycles does not significantly change the decoding problem. As a reference, Fig. 5 reports the performance of MWPM obtained on surface codes of dimension $d = 9$ and the previously described noise model using three measurement cycles and nine measurement cycles.

For the depolarising plus measurement errors model it is necessary to find the best hyperparameters for the neural network employed in the HLD. All the characteristics of the neural networks, like kernel dimension and activation functions are the same employed for the depolarising noise. The hyperparameters that were changed to improve the performance regard the number of layers and neurons. The neural networks have been trained on datasets of 2×10^6 elements, using 10% of the original training set as the evaluation set.

For the CNN the best hyperparameters are reported in Table 3. Some tests have been carried out using a FFNN, however no interesting results have been obtained using this architecture.

The accuracy obtained with the HLD, for the depolarising plus measurement errors model, is reported in Fig. 6 and has been compared to the accuracy obtained with

Table 3 Best hyperparameters of the CNN for different code distances for a depolarising plus measurement errors model

d	CL	DL	N	P
7	3	2	1024	6.4×10^6
9	3	2	1024	12.2×10^6
11	4	2	512	7.7×10^6

d is the code distance, CL is the number of convolutional layers, DL is the number of dense layers, N is the number of neurons in the dense layers and P is the total number of trainable weights

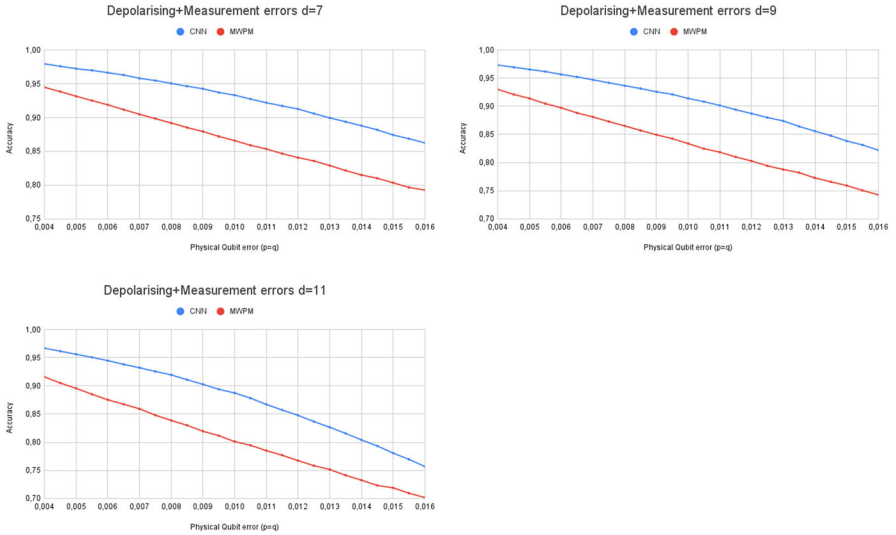

Fig. 6 Comparison between accuracy obtained with HLD based on CNN and the accuracy obtained with the classical decoder MWPM for the depolarising plus measurement errors model, the depolarising error probability (p) has been set equal to the measurement error probability (q)

MWPM. The test on different single qubit error probabilities have been made on datasets of 2×10^5 elements. When measurement errors are included the accuracy of MWPM reduces sensibly while the HLD is able to better adapt to this different noise model.

4.3 Dilated convolution

In order to improve the performance of the HLD for high distance codes, it should be useful to increase the local receptive field without incrementing the number of weights of the kernels. This result can be obtained with a dilated convolution [38]. Three different implementations of a CNN with a dilated convolution have been tested. In the first case only the first layer of the CNN has a dilation factor equal to two. In the second case all the convolutional layers except the first one have dilation factor two and in the third case all the layers have this dilation factor. The training has been performed, for all code dimensions, on the same datasets described in Sect. 4.1 for the

depolarising error model and in Sect. 4.2 for the depolarising plus measurement errors model. The best results have been obtained using a dilated convolution for all the convolutional layers except the first one. The performance of this CNN architecture has been compared with the same model without the dilation factor. The results are reported in Fig. 7 for depolarising error model (top) and depolarising plus measurement errors model (bottom).

The results obtained in this study show that dilated convolution may be a good way to improve the performance of the decoder for high distance codes. In fact, while for codes of distance 7 and 9 the use of dilated convolution doesn't alter the performance sensibly, for codes of distance 11, where there are more convolutional layers, it is possible to obtain a performance improvement.

The performance increase obtained with the dilated convolution is due to a reduction of the number of trainable weights in the neural network. In fact, no padding is used in the layers after the first one, so incrementing the local receptive field reduces the number of neurons of the first dense layer.

5 Choice of the training set

Neural networks are algorithms highly employed for their good generalisation properties. For the HLD it is possible to test these properties by training and testing the neural networks on datasets with different error probabilities. This is also very useful in real applications where the exact physical error rate is unknown.

For the depolarising error model, error probabilities of $p = 0.05; 0.075; 0.1; 0.13$ have been used to train the CNN for each code dimension. Each dataset is composed of 5×10^6 elements. The hyperparameters of the CNN are the ones described in Sect. 4.2 (Table 2). The performance has been evaluated on different error probabilities, the same datasets used in Sect. 4.1. The results of this study are reported in Fig. 8.

Figure 8 shows that a better performance is obtained when the neural network is trained on a higher error probability. This is due to the fact that, in datasets with higher error probability there are statistically relevant samples that are not present in datasets with a lower error probability.

However, when the dimension of the code becomes greater, a high error probability for the training set may cause problems for the training of the neural network. This happened for codes of dimension 11, using an error probability $p = 0.13$, in this case it has been possible to train the network only by starting from the weights of the network previously trained on an error probability $p = 0.1$. This procedure is very interesting as in this way it is possible to add more complicated error syndromes to the training set thus improving the performance of the decoder.

A similar analysis has been carried out for the depolarising plus measurement errors noise model. Four datasets (2×10^6 elements each) with different error probabilities have been generated for different code dimensions. The depolarising error probability is indicated with p while the measurement error probability with q, the four datasets have error probabilities of: $p = q = 0.005$, $p = q = 0.0075$, $p = q = 0.01$ and $p = q = 0.013$.

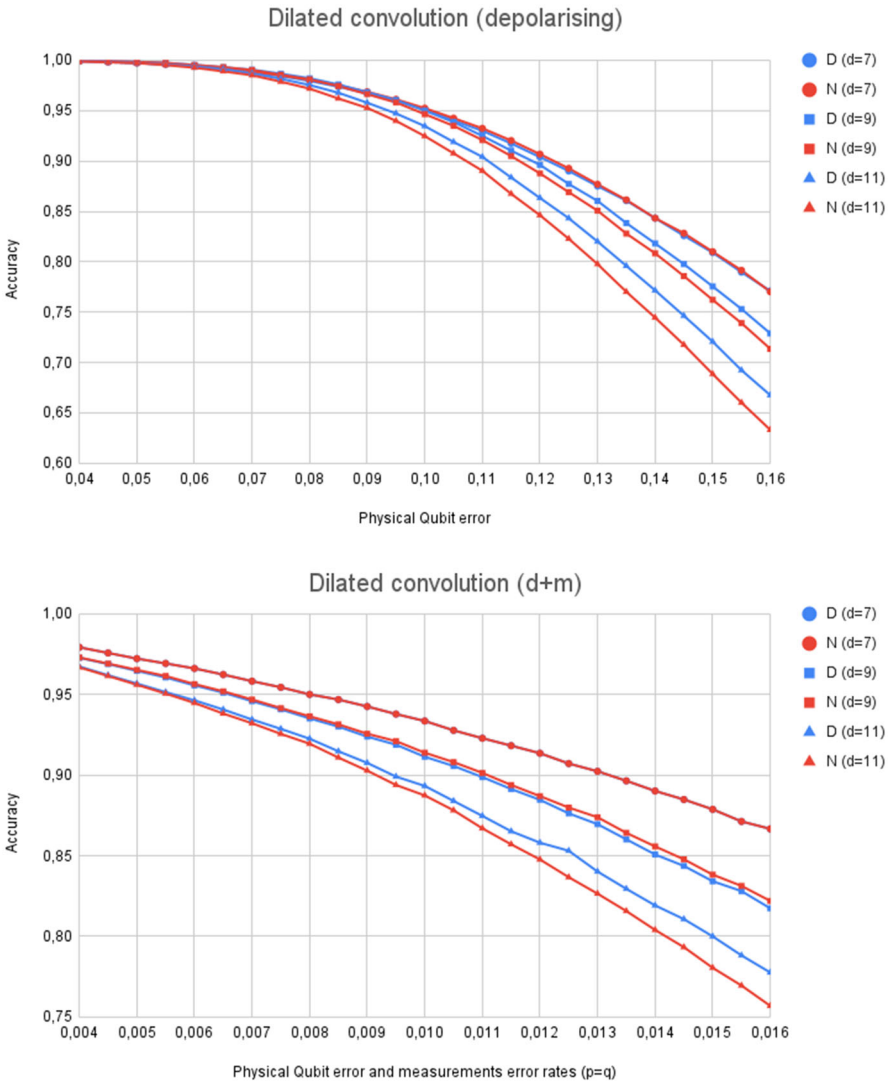

Fig. 7 Comparison of the accuracy, on different code distances (d), obtained with a CNN with no dilation factor (N) and the same model with a dilated convolution with a dilation rate equal to two on all the convolutional layers except the first one (D). For the depolarising error model (Top) the neural networks have been trained on a dataset with $p = 0.1$. While for the depolarising plus measurement errors model (Bottom) $p = q = 0.01$ (three measurement cycles). The graphs report the accuracy of the CNN decoder obtained using the described neural network architectures for different physical qubit error rates

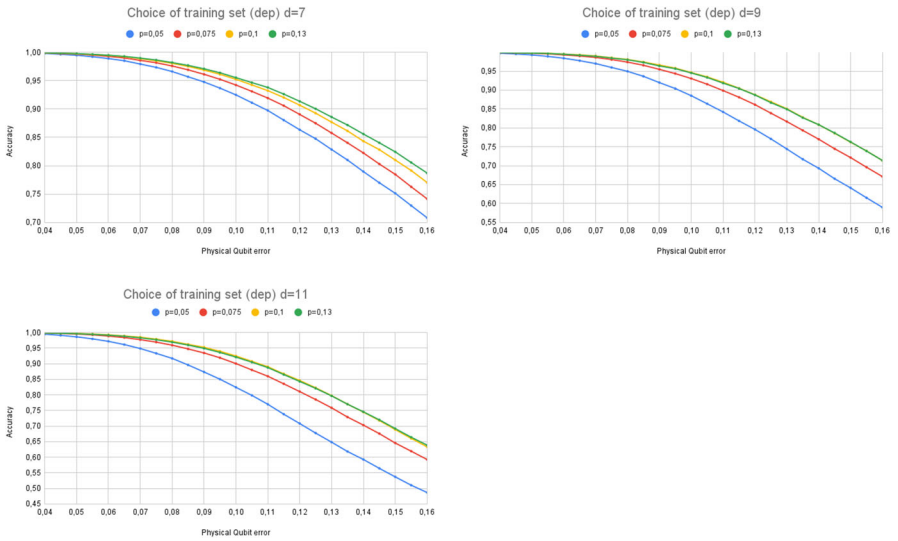

Fig. 8 Accuracy of the HLD trained and tested on different single qubit error probabilities for different code distances d. The training set single qubit error probability (p) is reported in the legend

For each code distance four neural networks have been trained using a single dataset. The CNN architecture and characteristics are the same of Sect. 4.2 (Table 3). The performance test with different error probabilities has been carried out on the same datasets used in Sect. 4.2. The results of this study are reported in Fig. 9, they are similar to the results obtained for the depolarising error model: it is better to use a higher error probability to train the neural network. However, also in this case, a high error probability may cause the neural network not to train correctly. In fact for both codes of distance 9 and 11 it was possible to train the network on an error probability $p = q = 0.013$ only by starting from the weights of the neural network previously trained on a lower error probability.

6 Model explainability

In order to trust complex and less transparent algorithms like artificial neural networks, it is necessary to know why they fail or work correctly. In particular, for convolutional neural networks trained for classification, it is important to understand the inputs that influence more the final decision. Many methods have been developed to construct saliency maps of the input pixels for image classification [39]. For example, with the GradCAM and Occlusion algorithms [40, 41], it is possible to obtain an heatmap of the input pixels based on their relevance for the output probability.

In the occlusion method several small regions of the image are systematically masked (all input pixels of the covered region are set to zero), and the changes in the loss function between the occluded and standard image are recorded. The saliency

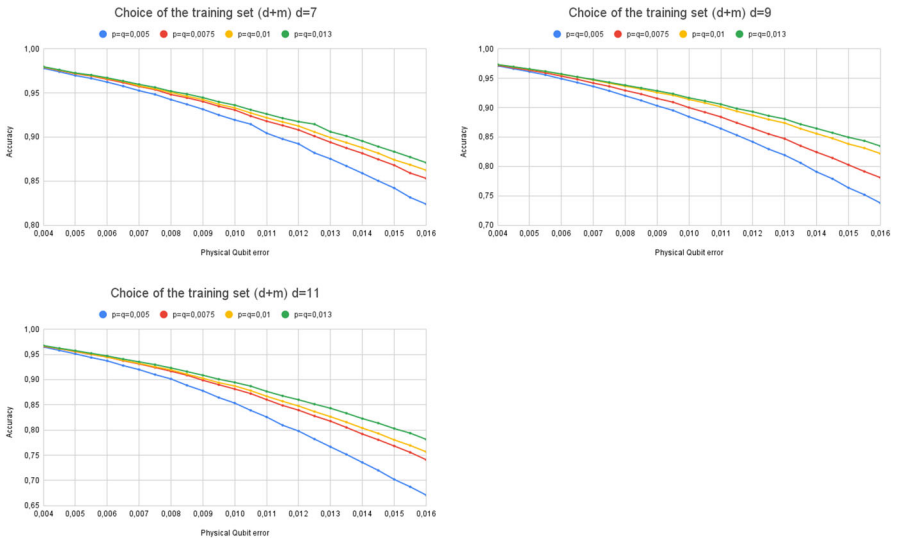

Fig. 9 Accuracy of the HLD trained and tested on different depolarising and measurement error probabilities for different code distances d. The training set single qubit error probability (p) and measurement error probability (q) is reported in the legend

value corresponding to a given masked region is given by:

$$Saliency(\text{region}) = \left[loss(image_{\text{original}}) - loss(image_{\text{masked}})\right]^2$$

We expect that a region that contributes more to the prediction of the model would change significantly the loss if masked. By shifting the masked region horizontally and vertically and repeating the process it is possible to construct the saliency map of the input features. For example, giving an image of 28x28 pixels and a occlusion patch of size 4x4 to mask the image, with a stride of 4 steps, an occlusion saliency map of size 7x7 is obtained. The salience map build in this way is then zoomed back to the original image resolution and overlaid to it.

Saliency maps help understanding if the trained HLD is performing as expected, and so allow to validate the algorithm, but they can also employed to better understand the errors of HLD in order to try to improve their performance. An original example of using salience maps to improve the algorithm itself is presented in Sect. 6.2.

6.1 Saliency map analysis

Occlusion method has been applied to the neural network of the HLD in order to understand how error syndromes are detected. Tests have been carried out for different code distances and neural networks trained on different error probabilities, the results obtained are very similar. In the following saliency maps of the input are reported for a depolarising noise model for codes of distance 11, using the neural network trained on an error probability $p = 0.13$. In fact this is the neural network that showed better

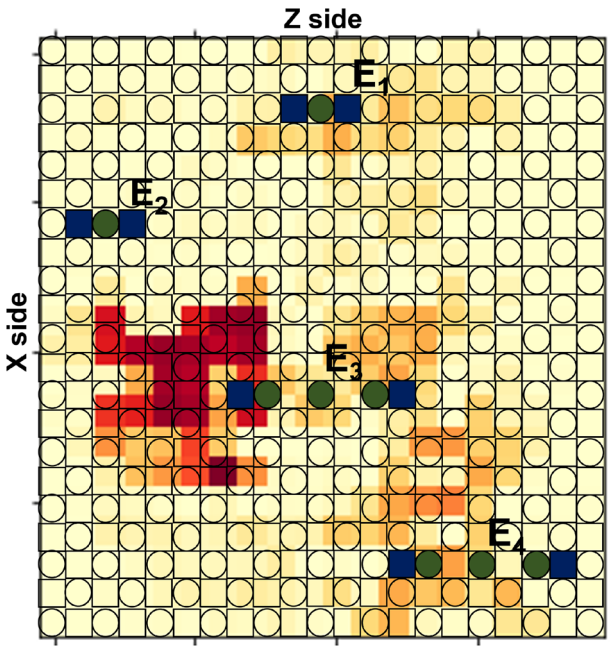

Fig. 10 Saliency map of a manually generated error syndrome analysed with a neural network decoder, the dark red regions are the ones that contribute more to the output. The error syndrome has been overlaid to the map, data qubits with X errors are highlighted in green, while the measurement qubits that report the error signal are highlighted in blue. Two single X errors, E_1 and E_2, and two errors chains composed of three X errors, E_3 and E_4 have been inserted in the surface code. The errors closer to the centre between the X sides (E_1 and E_3) are the ones that contribute more for the classification

results for codes of dimension 11, and with the depolarising noise model it is easier to find the more critical regions of the input. In the implementation of the occlusion method squared occlusion patches of the input of dimension 2×2 have been artificially set to zero. The dimension of the occluded window has been chosen in order to obtain a compromise between the smoothness and the granularity of the saliency map.

The first study regards how single errors or small chains of errors affect the final result, based on their position in the surface code (Fig. 10). Two single X errors have been placed, one in the middle between the X sides (E_1), and one near an X side (E_2). The single decoder is able to correct E_2 but creates an X logical error when trying to decode E_1. The Neural network is able to identify the introduction of a logical error, the region near E_1 contributes to the output probability while the region near E_2 doesn't contribute significantly. This is due to the fact that occluding the region of the error E_1 reverses the output of the neural network (X logical error can't be identified) thus significantly changing the loss.

Two errors chains composed of three X errors have been placed one in the middle between the X sides (E_3) and one near an X side (E_4). The simple decoder is able to correct E_4 but creates a logical error correcting E_3. The regions near the ends of the chain E_3 contribute more to the output than the regions near the ends of the chain E_4. Also in this case this is due to the fact that covering the error syndrome of E_3

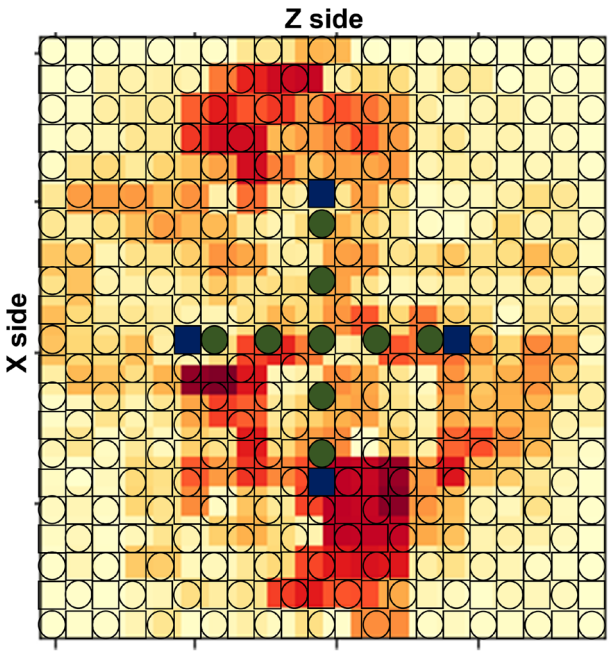

Fig. 11 Saliency map of a manually generated error syndrome analysed with a neural network decoder, the dark red regions are the ones that contribute more to the output. The error syndrome has been overlaid to the map, data qubit with depolarising errors are highlighted in green, while the measurement qubit that report the error signal are highlighted in blue. Two errors chains composed respectively of five X (horizontal) and Z (vertical) errors have been inserted in the middle of the surface code. The HLD is not able to identify the creation of an Y logical error after the correction of the simple decoder

reverse the output of the CNN. Moreover, we have observed also in other examples, that errors placed near the centre of the surface code influence more the output of the decoder.

The left side of the chain E_3 contributes more to the output than the right side. Similar asymmetries can be found in other saliency maps and change with respect to the horizontal or vertical position of the errors chain as well as the presence and position of other errors in the code. For example in the case reported in Fig. 10 the asymmetry is mainly due to the presence of the error E_2. Note that there are active regions of the input between the right side of the chain E_3 and the left side of the chain E_4. This first study considered only X depolarising errors, similar results can be obtained for single Z errors or small chains of Z errors. In this case the errors that contribute more to the output of the decoder are the ones placed near the middle between the Z sides.

For a depolarising error model the minimum weight perfect matching showed a better accuracy than the high level decoder, especially for codes of dimension 11. This means that there are cases where MWPM correctly identifies error chains while the high level decoder fails. An example is reported in Fig. 11 where two error chains, composed of five errors, have been inserted in the code. The first chain is composed

of X errors and is placed horizontally in the centre of the code, the second chain, composed of Z errors, is placed vertically in the centre of the code. Chains of this kind are very dangerous as they may be easily misidentified by a decoder and corrected by matching the error syndrome with the nearest border of the code thus creating a logical error. This is the case of the HLD, that is not able to identify the creation of an Y logical error after the correction of the simple decoder. In particular, the neural network classifies the error syndrome with no logical errors, this means that it is not able to find both the X and Z errors chains. Looking at the saliency map, reported in Fig. 11, it is possible to notice that the most relevant regions of the input are the ones located between the extremity of the chains and the nearest border of the surface code. This is probably due to the fact that the neural network decoder is looking for other errors in these regions in order to find the other extremities of the chains rather than matching X and Z syndromes with a single chain. This kind of error occurs almost every time a chain of length more than five appears in codes of distance 11. For codes of distance 9 and 7 the HLD correctly identifies chains of length 4 and 3 respectively. The MWPM is able to correct errors chains of length less than or equal $(d-1)/2$, where d is the dimension of the code. This means that this algorithm can correct chains of length 5 for $d = 11$ surface codes, while the maximum number of correctable errors in the chains is equal to 4 for $d = 9$ and 3 for $d = 7$. This explains the performance reduction of the HLD with respect to MWPM for codes of distance 11. The introduction of more samples in the training set, containing error chains of length five, can be a solution to improve the performance of HLD. This has been studied in Sect. 6.2.

There are some cases where the HLD is able to correct an error syndrome of a depolarising noise model where MWPM fails. An example is reported in Fig. 12, the error chain that gives problems to MWPM has been isolated by eliminating the other errors present in the code. After the correction of MWPM, an X error chain connecting the X sides of the code is introduced. On the other way, the HLD identifies all the logical errors introduced by the simple decoder. Looking at the saliency map in Fig. 12 it is possible to notice that the central region of the surface code, where two isolated errors are present, is not really relevant for the output. On the other way, the region near the right X side, where there is a small error chain, is the one that mostly affects the output.

6.2 Performance enhancement with data augmentation driven by model explainability

Model Explainability may be an interesting tool to improve the performance of ML algorithms. This can be done, for example, by analysing the critical issues in order to remove or attenuate them. In this section we have applied this idea to improve the performance of HLD.

Figure 11 showed that the CNN decoder fails to recognise some error chains of length 5 in a surface code of dimension 11. As these kinds of error chains can be corrected by MWPM, they account for the better performance of this decoder with

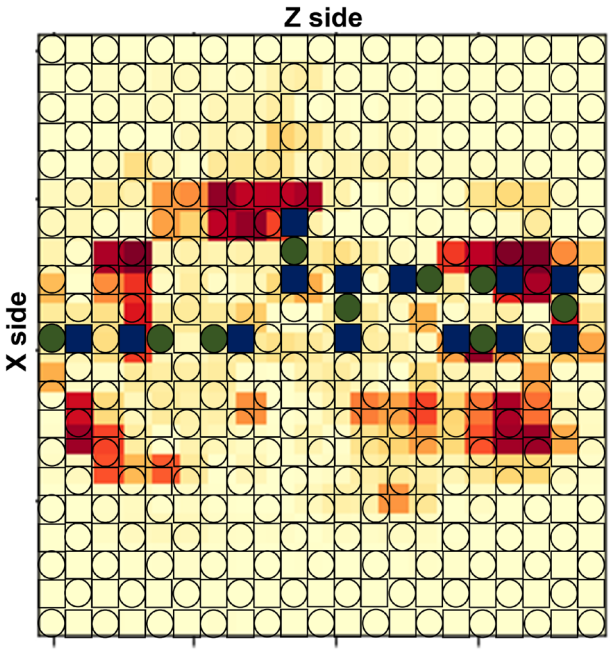

Fig. 12 Saliency map of an error syndrome analysed with a neural network decoder, the dark red regions are the ones that contribute more to the output. The error syndrome has been overlaid to the map, data qubit with X errors are highlighted in green, while the measurement qubit that report the error signal are highlighted in blue. In this case the HLD is able to correct the error syndrome while the MWPM fails. This example has been generated using a depolarising error model (perfect measurements) with single qubit error probability $p = 0.15$, the error chain responsible for the failure of MWPM has been extracted and analysed

respect to a HLD. In order to reduce this problem, we have generated an enhanced augmented data training set that includes some samples of these error chains.

These special samples have been generated with the following procedure. An error chain, composed of five single qubit errors of the same kind and on the same row or column, is added to the surface code. Both the row (or column) position of the chain, as well as the error locations, are randomly drawn. A single X error chain is added with probability $1/3$; with the same probability a single Z error chain is added and, in the other cases, both an X and a Z error chains are added.

The enhanced train set used to improve the performance of the CNN is composed of 7×10^6 samples divided as follows. 10^6 special samples containing error chains of length 5 (as previously described). 5×10^6 samples with a single qubit error probability $p = 0.1$ (standard train set employed in Sect. 4.1). 10^6 samples with a single qubit error probability $p = 0.13$; a higher single qubit error probability helps produce samples with longer error chains.

The employed CNN is the same described in Sect. 4.1 for surface codes of dimension 11. As the enhanced train set contains samples of different kind, the training batch size has been set to 128 to increase training stability. The model has been trained for four epochs, starting from the parameters trained in Sect. 4.1 on the standard train set

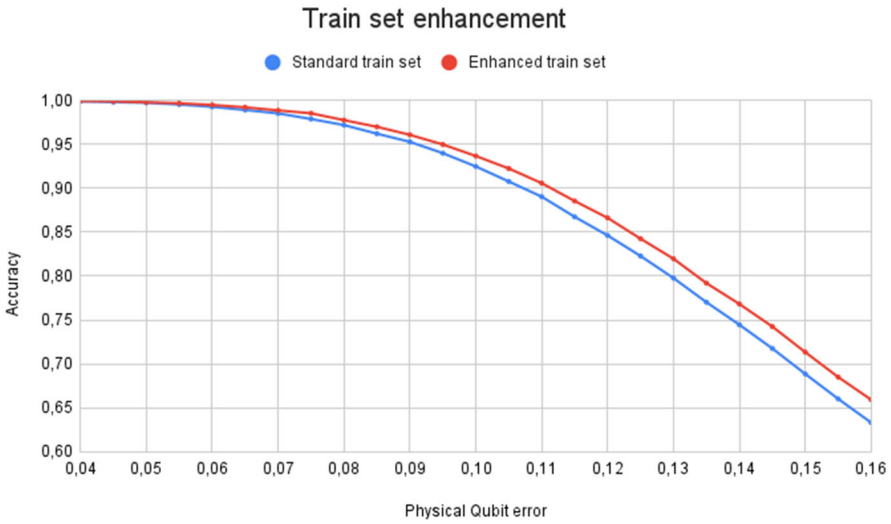

Fig. 13 Performance comparison between HLD trained on a standard train set with single qubit error probability $p = 0.1$ and the model trained on the enhanced train set that includes samples of error chains of length five

with a single qubit error probability $p = 0.1$. The trained model has been tested on different single qubit error probabilities, the same test sets introduced in Sect. 4.1.

Figure 13 reports the performance improvement obtained with the enhanced train set, with respect to the standard train set employed in Sect. 4.1. The performance improvement is significant, about 1% for single qubit error probability $p = 0.1$ and up to 2% for higher single qubit error probabilities. Moreover, the new model is able to correctly decode the example reported in Fig. 11. The new saliency map, obtained with this model, is reported in Fig. 14. A comparison with Fig. 11 shows how the HLD trained on the enhanced dataset gives the same importance to X and Z error signals. This decoder is able to better understand the importance of the error signals at the end of the error chains, as well as the central region of the surface code.

7 Conclusions

Neural network based decoders have proved to be excellent algorithms for the decoding of surface codes due to their constant execution time, good accuracy and adaptability to different noise models. The use of a convolutional architecture, with respect to a dense architecture, helps scaling to higher distance codes. However, as incrementing the distance of the code increments exponentially the possible error syndromes, larger datasets are required for correct training. This makes difficult to apply a simple convolutional architecture to decode really high distance codes. Convolutional neural networks decoders remain interesting as they may be used as the first step of a more sophisticated decoder to process local information [34]. Moreover, it is likely that in the future only small distance surface codes will be available for the first tests.

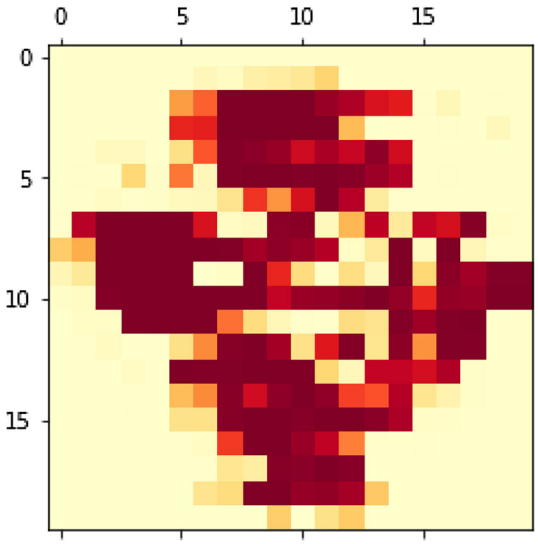

Fig. 14 Saliency map of the same error syndrome reported in Fig. 11, decoded with the HLD trained on the enhanced train set. The dark red regions are the ones that contribute more to the output, they are placed near the error signals and the centre of the surface code. This HLD is able to correctly identify the presence of error chains from a sparse error syndrome

The results obtained in this work are promising and helpful to improve the accuracy obtained with HLD based on convolutional neural networks. Different convolutional neural network architectures and training strategies have been tested for different code distances and noise models. The results suggest that using a training set with a higher error probability helps improving the performance of the decoder, and that successful training benefits can be obtained from a pre-training based on examples on a lower error probability. It was also shown how it is possible to use dilated convolution to reduce the number of parameters that can be trained with decoders based on very deep convolutional neural networks. This suggest a possible way to try to scale the convolutional neural networks based decoder to larger surface codes.

Finally, explainability methods have been proposed and applied to get insights on how error syndromes are classified by the neural network decoder. A better knowledge of the causes of the failure of the decoding is in fact fundamental to improve the performance, robustness and confidence in neural network HLD for real applications. In this respect an original example of use of the salience maps to improve the HLD algorithm has been presented.

Funding Open access funding provided by Universitá degli Studi di Roma La Sapienza within the CRUI-CARE Agreement. this research received no external funding.

Code Availability the datasets and the code used for this study are available on request by contacting the authors.

Declarations

Conflict of interest: the authors have no competing interests to declare that are relevant to the content of this article.

References

1. Schlosshauer, M.: Quantum decoherence. Phys. Rep. **831**, 1–57 (2019). https://doi.org/10.1016/j.physrep.2019.10.001
2. Benenti, G., Casati, G., Strini, G.: Principles Of Quantum Computation And Information - Volume II: Basic Tools And Special Topics. World Scientific Publishing Company, Italy (2007). https://books.google.it/books?id=Its7DQAAQBAJ
3. Roffe, J.: Quantum error correction: an introductory guide. Contemp. Phys. **60**(3), 226–245 (2019). https://doi.org/10.1080/00107514.2019.1667078
4. Knill, E., Laflamme, R., Viola, L.: Theory of quantum error correction for general noise. Phys. Rev. Lett. **84**(11), 2525–2528 (2000). https://doi.org/10.1103/physrevlett.84.2525
5. Kitaev, A.Y.: Fault-tolerant quantum computation by anyons. Ann. Phys. **303**(1), 2–30 (2003). https://doi.org/10.1016/s0003-4916(02)00018-0
6. Fowler, A.G., Mariantoni, M., Martinis, J.M., Cleland, A.N.: Surface codes: Towards practical large-scale quantum computation. Phys. Rev. A. **86**(3) (2012). https://doi.org/10.1103/physreva.86.032324
7. Varsamopoulos, S., Bertels, K., Almudever, C.G.: Decoding surface code with a distributed neural network based decoder (2019) arXiv:1901.10847 [quant-ph]
8. Varsamopoulos, S., Bertels, K., Almudever, C.G.: Comparing neural network based decoders for the surface code. IEEE Trans. Comput. **69**(2), 300–311 (2020). https://doi.org/10.1109/tc.2019.2948612
9. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information: 10th Anniversary Edition, 10th edn. Cambridge University Press, USA (2011)
10. Horsman, C., Fowler, A.G., Devitt, S., Meter, R.V.: Surface code quantum computing by lattice surgery. New J. Phys. **14**(12), 123011 (2012). https://doi.org/10.1088/1367-2630/14/12/123011
11. Litinski, D.: A game of surface codes: large-scale quantum computing with lattice surgery. Quantum **3**, 128 (2019). https://doi.org/10.22331/q-2019-03-05-128
12. Terhal, B.M.: Quantum error correction for quantum memories. Rev. Mod. Phys. **87**(2), 307–346 (2015). https://doi.org/10.1103/revmodphys.87.307
13. Fowler, A.G.: Analytic asymptotic performance of topological codes. Phys. Rev. A. **87**(4) (2013). https://doi.org/10.1103/physreva.87.040301
14. Wang, D.S., Fowler, A.G., Stephens, A.M., Hollenberg, L.C.L.: Threshold error rates for the toric and surface codes (2009) arXiv:0905.0531 [quant-ph]
15. Tomita, Y., Svore, K.M.: Low-distance surface codes under realistic quantum noise. Phys. Rev. A. **90**(6) (2014). https://doi.org/10.1103/physreva.90.062320
16. Gottesman, D.: Stabilizer codes and quantum error correction (1997) arXiv:9705052 [quant-ph]
17. Spitz, S.T., Tarasinski, B., Beenakker, C.W.J., O'Brien, T.E.: Adaptive weight estimator for quantum error correction in a time-dependent environment. Adv. Quantum Technol. **1**(1) (2018). https://doi.org/10.1002/qute.201800012
18. Fowler, A.G.: Minimum weight perfect matching of fault-tolerant topological quantum error correction in average $o(1)$ parallel time (2014) arXiv:1307.1740 [quant-ph]
19. Cook, W., Rohe, A.: Computing minimum-weight perfect matchings. INFORMS J. Comput. **11**(2), 138–148 (1999). https://doi.org/10.1287/ijoc.11.2.138
20. Bravyi, S., Suchara, M., Vargo, A.: Efficient algorithms for maximum likelihood decoding in the surface code. Phys. Rev. A. **90**(3) (2014). https://doi.org/10.1103/physreva.90.032326
21. Duclos-Cianci, G., Poulin, D.: A renormalization group decoding algorithm for topological quantum codes (2010) arXiv:1006.1362 [quant-ph]

22. Duclos-Cianci, G., Poulin, D.: Fast decoders for topological quantum codes. Phys. Rev. Lett. **104**(5) (2010). https://doi.org/10.1103/physrevlett.104.050504
23. Herold, M., Campbell, E.T., Eisert, J., Kastoryano, M.J.: Cellular-automaton decoders for topological quantum memories. npj Quantum Inf. **1**(1) (2015). https://doi.org/10.1038/npjqi.2015.10
24. Bravyi, S., Suchara, M., Vargo, A.: Efficient algorithms for maximum likelihood decoding in the surface code. Phys. Rev. A. **90**(3) (2014). https://doi.org/10.1103/physreva.90.032326
25. Hutter, A., Wootton, J.R., Loss, D.: Efficient markov chain monte carlo algorithm for the surface code. Phys. Rev. A. **89**(2) (2014). https://doi.org/10.1103/physreva.89.022326
26. Varsamopoulos, S., Criger, B., Bertels, K.: Decoding small surface codes with feedforward neural networks. Quantum Sci. Technol. **3**(1) (2017). https://doi.org/10.1088/2058-9565/aa955a
27. Fowler, A.G.: Optimal complexity correction of correlated errors in the surface code (2013) arXiv:1310.0863 [quant-ph]
28. Delfosse, N., Tillich, J.-P.: A decoding algorithm for css codes using the x/z correlations. In: 2014 IEEE International Symposium on Information Theory (2014). https://doi.org/10.1109/isit.2014.6874997
29. Baireuther, P., O'Brien, T.E., Tarasinski, B., Beenakker, C.W.J.: Machine-learning-assisted correction of correlated qubit errors in a topological code. Quantum **2**, 48 (2018). https://doi.org/10.22331/q-2018-01-29-48
30. Torlai, G., Melko, R.G.: Neural decoder for topological codes. Phys. Rev. Lett. **119**(3) (2017). https://doi.org/10.1103/physrevlett.119.030501
31. Krastanov, S., Jiang, L.: Deep neural network probabilistic decoder for stabilizer codes. Sci. Rep. **7** (2017). https://doi.org/10.1038/s41598-017-11266-1
32. Sheth, M., Jafarzadeh, S.Z., Gheorghiu, V.: Neural ensemble decoding for topological quantum error-correcting codes. Phys. Rev. A. **101**(3) (2020). https://doi.org/10.1103/physreva.101.032338
33. Bhoumik, D., Sen, P., Majumdar, R., Sur-Kolay, S., J, L.K.K., Iyengar, S.S.: Efficient decoding of surface code syndromes for error correction in quantum computing (2021) arXiv:2110.10896 [quant-ph]
34. Meinerz, K., Park, C.-Y., Trebst, S.: Scalable neural decoder for topological surface codes (2021) arXiv:2101.07285 [quant-ph]
35. Ni, X.: Neural network decoders for large-distance 2d toric codes. Quantum **4**, 310 (2020). https://doi.org/10.22331/q-2020-08-24-310
36. Davaasuren, A., Suzuki, Y., Fujii, K., Koashi, M.: General framework for constructing fast and near-optimal machine-learning-based decoder of the topological stabilizer codes. Phys. Rev. Res. **2**(3) (2020). https://doi.org/10.1103/physrevresearch.2.033399
37. Fowler, A.G., Whiteside, A.C., McInnes, A.L., Rabbani, A.: Topological code autotune. Phys. Rev. X. **2**(4) (2012). https://doi.org/10.1103/physrevx.2.041003
38. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions (2016) arXiv:1511.07122 [cs.CV]
39. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Object detectors emerge in deep scene cnns (2015) arXiv:1412.6856 [cs.CV]
40. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: visual explanations from deep networks via gradient-based localization. Int. J. Comput. Vision **128**(2), 336–359 (2019). https://doi.org/10.1007/s11263-019-01228-7
41. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks (2013) arXiv:1311.2901 [cs.CV]