ADVANCED QUANTUM TECHNOLOGIES
www.advquantumtech.com

On the Interpretability of Neural Network Decoders

Lukas Bödeker, Luc J. B. Kusters, and Markus Müller*

Neural-network (NN) based decoders are becoming increasingly popular in the field of quantum error correction (QEC), including for decoding of state-of-the-art quantum computation experiments. In this work, established interpretability methods are used from the field of machine learning, to introduce a toolbox to achieve an understanding of the underlying decoding logic of NN decoders, which have been trained but otherwise typically operate as black-box models. To illustrate the capabilities of the employed interpretability method, based on the Shapley value approximation, an examplary case study of a NN decoder is provided that is trained for flag-qubit based fault-tolerant (FT) QEC with the Steane code. The interpretation of particular decoding decisions of the NN is analysed, by doing so it is revealed how the NN learns to capture fundamental structures in the information gained from syndrome and flag qubit measurements, in order to come to a FT correction decision. Further, it is showed that the understanding of how the NN obtains a decoding decision can be used on the one hand to identify flawed processing of error syndrome information by the NN, resulting in decreased decoding performance, as well as for well-informed improvements of the NN architecture. The diagnostic capabilities of the interpretability method presented here can help ensure successful application of machine learning for decoding of QEC protocols.

1. Introduction

Fault-tolerant (FT) quantum error correction (QEC) is a key ingredient to suppress error rates of quantum computers to a degree where scalable quantum algorithms can outperform classical computers.[1] State-of-the-art experiments have demonstrated QEC close to or even below break-even, below which the error rates of corrected logical qubits are reduced as compared to their physical qubit counterparts. Here, impressive breakthroughs in a variety of platforms including trapped ions,[2–13] superconducting systems[14–22] and neutral Rydberg atom systems[23–28] have been achieved. In these experimental realizations, the FT operation of error-corrected logical qubits is shown in different settings, demonstrating key components for large-scale error-corrected quantum computation. These active error correction protocols all have in common that partial information about erroneous processes must be processed in order to determine a recovery operation that – if successful – yields the preservation of the logical information. In error correcting (topological) stabilizer codes,[29] the logical state information of a single qubit is encoded non-locally across many qubits. To detect errors, local parity check operators are measured, which leave logical information untouched. The according measurement outcomes form the error syndrome and can be used to determine a recovery operation to revoke the error with a certain probability. The computational cost of decoding of the syndrome scales, in general, exponentially with growing code-size, when performing optimal decoding that yields the highest success probability of revoking the error (maximum likelihood decoding.[30]) For this reason, depending on the code, there exists a plethora of efficient decoders of suboptimal performance that aim at approximating optimal decoding while lowering the computational effort. To illustrate the vibrant field of finding efficient decoding algorithms, we point out a number of works for two topological code families of surface[31] and color codes.[32] These decoding approaches[33–44] are not meant to be a complete list.

Neural network (NN) based decoders form a versatile decoding paradigm, applicable to a wide range of quantum correcting codes and FT quantum computation protocols.[45–62] These decoders have the advantage of being able to adapt autonomously to the different noise models of any underlying physical implementation through learning, which can be based on numerical simulation or experimental data.[56, 57, 59] Furthermore, trained NN decoders have shown to be capable of generalizing to different decoding situations beyond those they have been trained for, such as, higher code distances or different numbers of syndrome extraction repetitions.[50, 51] On the other side, the performance of these decoders is heavily dependent on choosing an appropriate NN model and the success of its optimization (training). The latter in turn depends on the availability of training data and computational resources for training. The training of a NN

L. Bödeker, L. J. B. Kusters, M. Müller
Institute for Theoretical Nanoelectronics (PGI-2), Forschungszentrum Jülich
52428 Jülich, Germany
E-mail: l.boedeker@fz-juelich.de
L. Bödeker, L. J. B. Kusters, M. Müller
Institute for Quantum Information, RWTH Aachen University
52056 Aachen, Germany

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/qute.202500158

DOI: 10.1002/qute.202500158

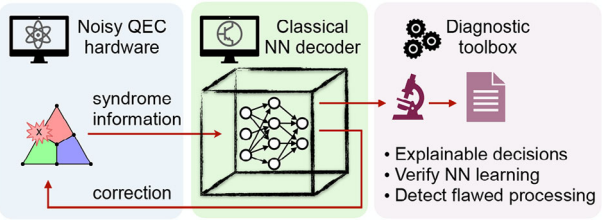

Figure 1. Opening the black box for a neural network (NN) based decoder: The interpretability toolbox offers diagnostic tools to (i) gain an understanding of how the NN decoder has come to a certain correction decision, for given error syndrome information, to (ii) supervise and verify the NN's internal learning process and to (iii) detect flawed information processing by the decoder, which otherwise can impede fault-tolerant operation of noisy QEC codes.

becomes in general a resource-intensive endeavor with growing size of the NN model. For a readily trained NN on the other hand, determining a decoding decision is typically straight-forward and not computationally intensive. Indeed, trained NN decoders have shown good decoding performance whilst showing small overhead in terms of classical computations.[49, 52]

Moreover, in the field of quantum error correction, the solving of hard classical processing tasks is not limited to the problem of conventional decoding but also extends to the protection and correction of bosonic codes[63, 64] or the construction of FT circuits to encode logical states.[65] Naturally, such tasks are potential fields of application of NNs, where promising results have been shown to protect bosonic code states[66, 67] or to find FT encoding circuits.[68] These instances exemplify how relevant the utilization of NNs is in the broader field of quantum error correction and quantum computation.

A caveat regarding the use of NNs and in particular NN decoding is that NN decoders typically act as a black-box, **Figure 1** meaning that the input-output relationship of the neural network is not accessible in the same way as it would be for a human-designed decoding algorithm. In other words, it is in the general case hard to retrace how the NN obtains its decoding result from the input information. This lack of structural understanding of the decision model when employing NNs is a shortcoming, which might potentially limit the applicability of this approach, in particular when operating larger QEC codes under realistic experimental noise and when executing more complex tasks such as error correction of logical quantum algorithms. Furthermore, the black-box type decision character of the decoding process could screen non-optimal performance or the reasons for a failed training of the NN decoder.

In this work, we investigate interpretability methods for NNs to open this black-box, as schematically shown in Figure 1. The techniques we employ are adapted from the field of explainable machine learning, often referred to as Explainable Artificial Intelligence (XAI),[69–71] which is a sub-field of machine learning[72] focusing on increasing the transparency and interpretability of machine learning models. The goal here is to provide insights into how machine learning models make predictions and decisions, in order to check for plausibility and to detect possible malfunctions of the model.[69, 71, 73–79] In XAI a distinction is made between local explanation methods[69, 71, 73, 75, 76, 78, 79] and global explanation methods.[74] Local explanation methods aim to explain individual predictions, while global explanation methods aim to provide insights into the workings of the model as a whole. Global explanations are often obtained by aggregating over many local explanations.[75]

In our work, we will focus on a model-agnostic and local method, the so-called Shapley values[80, 81] for interpreting individual decoding outcomes of a NN decoder. These interpretations are given in terms of an importance score of sub-sets of the input which in our case are parts of the syndrome information. From the individual decoding interpretations, we can derive global conclusions about the decoder working by analyzing the statistics of the interpretations. This Shapley-value based method that we adapt for the field of NN decoding is general in the sense that it can be applied for any kind of NN architecture that supports backpropagation and any underlying QEC decoding task. This includes in particular recurrent NNs as used in this work as well as for instance more modern transformer based NN architectures[57, 58] that are also used in large language models. In principle, it can be applied to interpret the NN decoding for arbitrary QEC codes, at any code distance and any FT protocol or measurement scheme, provided that a NN has been trained successfully beforehand. The latter challenge could pose a bottleneck for the NN based decoding of larger codes, as the question of the scalability of the necessary NN training is still an active research topic.[60] However, if provided with a working NN, the interpretability methods employed in this work cause a computational overhead that scales only linearly with the computational cost of executing a single decoding run.

We show that our interpretability method can be used to certify the learning success of a NN to perform FT decoding. Interestingly, this learning success can be temporally resolved along the training of the NN. This allows one to spectate the learning transition of the NN, first learning simple non-FT decoding strategies, and later in the training process the discovery and adoption of a FT decoding behaviour. Furthermore, we can infer whether the NN has understood how to utilize correlations between X and Z syndromes, relevant for the correction of phase and bit flip errors, respectively, to refine the decoding decisions it is making. Apart from applying the interpretability method to check for desired properties, we show that the employed interpretability analysis also enables the identification of unwanted functional behaviour that the NN can acquire during the training, and which deteriorates the decoding performance and can be attributed to overfitting. Such diagnostic element then allows, in a general setting, for an informed augmentation of the NN decoder to ultimately improve the logical performance of the underlying error correction protocol.

For concreteness, we test our interpretability method based on numerical simulations of a flag-FT[82–84] circuit implementation of a QEC for the distance-3 Steane code[32, 85, 86] under circuit level noise in a quantum memory setting. This choice is motivated by posing the learning challenge for the NN decoder to combine information from syndrome as well as from flag qubits, obtained from various measurement rounds. If this combination of information pieces is successfully learned by the NN, it should then be able to correct for any single fault that occurs during the stabilizer readout, and thereby maintain the FT of the QEC code in combination with the suitably constructed flag-qubit based syndrome readout quantum circuitry. Furthermore, as for other CSS

codes, in this setting it is possible to study the effect of the correlation between X and Z errors for the decoding decision as the respective two decoding problems can be solved independently. Additionally, studying this concrete setting is also motivated by its near-term experimental relevance.[5,6,8,25] In terms of the NN model we choose for decoding, we work with simple recurrent NNs, as previously used in the QEC decoding literature.[51] These NNs are employed and trained to classify the logical-flip parity on the QEC code after T rounds of faulty syndrome measurement.

This manuscript is structured as follows: In Section 2 we explain the NN decoding paradigm and introduce the exemplary QEC setting, which we later test on our interpretation method. In Section 3, we introduce the Shapley value, outline how it can be calculated and explain how it can be used to derive local as well as global interpretations. In Section 4, we then present how the learning progress of the NN decoder is analyzed, focusing on the NN's ability to decode fault-tolerantly. In Section 5 we show, based on a fully trained model, how possible malfunctions of the NN decoder can be uncovered. Finally, Section 6 provides conclusions and an outlook to potential extensions of the work.

2. Neural Network Decoding

Various NN architectures and approaches for decoding have been explored.[45, 49] Given a QEC experiment with T rounds of repeated stabilizer measurements, one can for instance provide the whole space-time syndrome volume[88] as input to the NN to perform the decoding. For growing code distances and therefore large spatial syndrome, these NNs can further be enriched by sparsely connected network parts as elements of their architecture to perform a spatially local preprocessing, as it is done in convolutional NNs.[54, 55, 59] Still, these networks have the caveat that, after training, they are fixed to a single QEC protocol together with a syndrome volume of fixed size as input. If one is, however, faced with a protocol where a variable number of stabilizer measurement rounds is needed, and for which the circuitry that is run is time-translationally invariant, it is more convenient being able to pass the syndrome information to the NN sequentially, in a time-resolved way.[51, 56, 57] Such a procedure is reminiscent of algorithmic decoders that employ a sliding window to process the syndrome history.[89, 90] In order to be able to still use information of stabilizer measurements at different times, the NN is required to be equipped with a memory[91] from which an overall decoding decision can be determined considering the joint syndrome volume information.

The problem of decoding can be formulated in different ways for a NN that is to be trained. A microscopic approach would be to train a NN to determine every fault that has occurred given the syndrome history. A simplified, but for decoding sufficient, task for the NN could be to only predict the logical parity of the accumulated errors. If a NN can predict this logical parity reliably, one could assume that it processes the syndrome information correctly into information about the errors internally. We employ a type of recurrent neural network (RNN) that can process a variable amount of the syndrome readouts, similar as in the works.[46, 49, 51] The RNN is trained to predict the logical parities after a variable number of $\{T_i\}$ rounds of stabilizer measurement such that after the training it will be able to make this prediction after being fed measurement outcomes from $T' \notin \{T_i\}$ rounds of stabilizer measurement Figure 3, see[87] for details about the network specifications.

Quantum memory experiment— This setting is relevant if one wants to protect the information encoded in a logical qubit state $|\psi_L\rangle$ that is left to idle, while a number T of stabilizer measurement cycles is performed, of which one is shown in **Figure 2b** for the Steane code.

For the Steane code, the problem of decoding amounts to a classification problem, where as input the joint vector of syndrome and flag measurements $\vec{s}$ encodes partial information about where in the circuit faults have occurred. The task of the NN is therefore to decide on a logical correction after T rounds of stabilizer measurements, see **Figure 3**. To evaluate the success of the logical correction that is proposed by the NN, the logical state needs to be brought back to the code space. The latter can be done by a simple correction based on the data qubit readout after the T rounds of stabilizer measurement.

In more formal terms, the decoding is broken up according to the decomposed error E as $E = S \cdot P \cdot P_L$, where S is an element of the stabilizer group, P_L is a logical operator (including the identity operator $I^{\otimes n}$), and P is a 'pure error' taking the state out of the logical subspace. The latter is chosen to have the smallest possible support, implicating that no logical operator can be contained in P. As an example based on the Steane code layout in Figure 2a, consider the error $E = X_1X_4X_5X_7$ which can be written as $E = I \cdot X_5 \cdot X_L$, where one representation of the logical error is $X_L = X_1X_4X_7$ and the pure error that causes a nontrivial syndrome is given as $P = X_5$. Given only the syndrome information of this pure error $P = X_5$, a decoder would assert a correction $C = X_5$. Overall, this means that after correction, we are still left with an uncorrected logical error $E \cdot C = X_L$ and therefore a failure of the memory. Let us assume now that the decoder has access to the syndrome history of T rounds of stabilizer measurements and that the error $E = X_1X_4X_5X_7$, is the cumulative error that the data qubits acquired during this time. The correction of the error may now be split up as $C = C_L \cdot C_P$, where the "logical correction" $C_L = X_L^{n=0,1} Z_L^{m=0,1}$ can be determined after correcting the pure error with C_P. The correction of the pure error is conducted based only on the stabilizer violations after the T rounds of syndrome extraction, such that the logical state is brought back to the code space, i.e. $[C_P \cdot P, S] = 0$ for all elements of the stabilizer group S. In the example case, we would still get $C_P = X_5$ as the simple correction, bringing the state back to the code space, $P \cdot C_P = I^{\otimes n}$. In general, the correction of the pure error can however also yield a logical operator. As a consequence, the desired, successful logical correction, can be written as $C_L = S' \cdot P_L \cdot (C_P P)$. For the remaining error $E \cdot C_P = X_1X_4X_7$, the desired logical correction by the decoder based on the syndrome history would be $C_L = X_L$.

Again, by this treatment, determining C_L turns into a classification of whether a logical bit- or phase-flip has occurred or not, based on the full syndrome history. Consequently, the logical parities can be represented as two binary values, which can be predicted using a neural network that is fed with the syndrome flag history. In our simulations, the correction of this pure error C_P is performed by a virtual error correction step based on a perfect final data qubit readout, from which a syndrome can be derived that can be corrected by means of look-up-table.

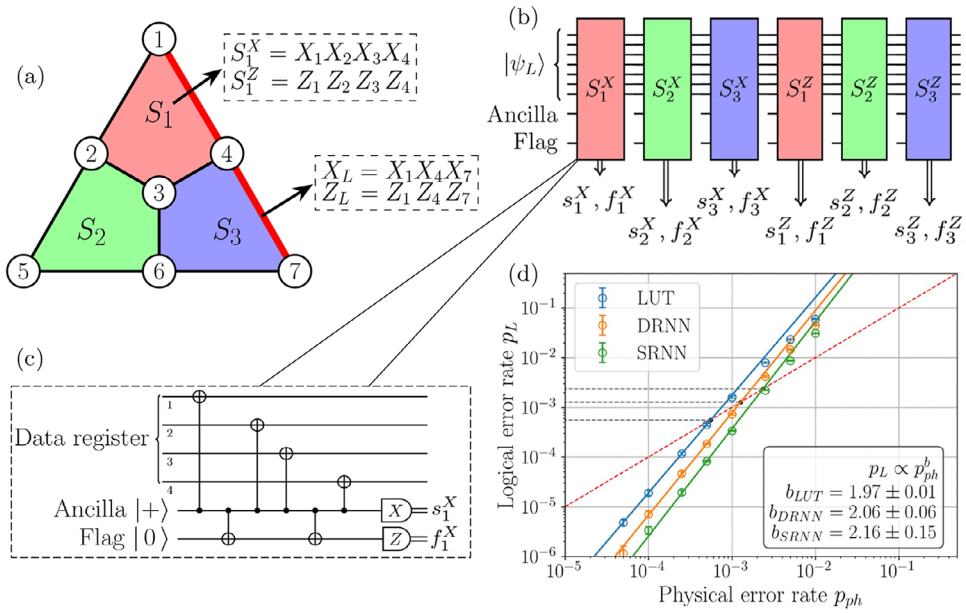

Figure 2. a) The ⟦7, 1, 3⟧ color code. The vertices represent data qubits and the faces represent the $X-$ and $Z-$ type stabilizer generators. The pair of red plaquette stabilizer generators is indicated as an example, as well as a representation of the logical operators. The other four stabilizer generators are defined analogously as the green and blue plaquette. b) Measurement schedule for the flagged stabilizer readout. The measurements return the stabilizer parity and flag information. c) Exemplary flag-based fault-tolerant measurement circuit for the S_1^X stabilizer. The flag qubit that is coupled to the ancilla qubit is capable to identify any potentially dangerous error propagations. d) Performance comparison of three fault-tolerant decoding schemes: look-up table decoder (LUT in blue), dual output recurrent neural network decoder (DRNN in orange) and single output recurrent neural network decoder (SRNN in green). All decoders show quadratic scaling of the logical error rate p_L with the physical error rate p_{ph}, implying the capability of the decoders to operate fault-tolerantly and correct all single faults in the circuit. Error bars are computed as the one-sigma Wilson interval.[87]

Fault-tolerant stabilizer measurement— We consider depolarizing circuit level noise[92] with a physical error parameter p_{ph}. Specifically, we assume noisy qubit initialization and measurement where the outcome is inverted with a probability of $2/3\,p_{ph}$. After every single-qubit gate, one of the three single-qubit Pauli operators $\{X, Y, Z\}$ is applied with a probability $p_{ph}/3$. After every two-qubit gate, one of the 15 two-qubit Pauli operators $\{I, X, Y, Z\}^{\otimes 2} \setminus I \otimes I$ is applied with a probability $p_{ph}/15$. Given this noise model, a single fault on a measurement ancilla qubit can propagate onto multiple data qubits. In particular, in the Steane code no weight-two data qubit error of two Pauli operators of the same type can be corrected. If this spreading of errors was not detectable as in the case of using a single physical ancilla for the readout, the logical information would become irrecoverable. One example for a fault that causes a weight-two data qubit error is shown in Figure 5c. We will call this fault class hook errors in the following. To prevent such hook errors from going unnoticed, there exist several measurement schemes which allow for syndrome extraction while preserving fault tolerance.[93–96] The scheme of our choice[82] exploits an additional measurement ancilla qubit, the so-called flag, that is entangled with a syndrome ancilla during the stabilizer measurement, see Figure 2c. In this way, all dangerous hook errors also propagate onto the flag qubit and cause a nontrivial measurement outcome. When such a flag is raised during one round of stabilizer measurements, all possible weight one faults can be corrected if the stabilizer measurements are repeated. The combination of a raised flag and the syndrome of the subsequent round can unambiguously identify all possible dangerous error propagations or hook-errors. These information-tuples of flag and stabilizer measurement outcomes are therefore also called hook-signatures and will become important to be identified by the decoder. As a result, the employment of the additional flag qubit enables the fault-tolerance of the overall measurement, hence we refer to it as a flag-fault-tolerant readout in the following.

Neural network architecture— The recurrent cell we choose for our RNN is the so-called Long Short Term Memory (LSTM) unit,[91] see[87] for details. It is a reasonably simple and extensively tested NN element that allows for actively memorizing and forgetting information based on the input. The latter is a needed capability to combine measurement outcome information from various consecutive rounds of syndrome extraction for the decoding decision. In the overall NN, LSTM layers are complemented with non-recurrent layers. The embedding of the RNN as the decoder as well as its inner composition is shown schematically in Figure 3. The RNN receives syndrome increments and flag bits as input in a time-resolved and sequential manner. These inputs are then passed to two consecutive layers of LSTM units that allow for combining information of spatially and temporally separated syndrome bits followed by a dense feed-forward neural network (DNN) to perform post-processing. The first LSTM layer uses a many-to-many structure in space and time, i.e., it passes a sequence of outputs $\vec{h}_t^{(1)}\ \forall\, t$ to the next layer given a sequence of inputs $\vec{s}(t)$. On the other hand, the second LSTM layer contracts these sequences to only the final sequence output, $\vec{h}_{(t=T)}^{(2)}$ which is then passed to a DNN layer. A DNN can be understood as a conventional network, consisting of layers of real-number valued

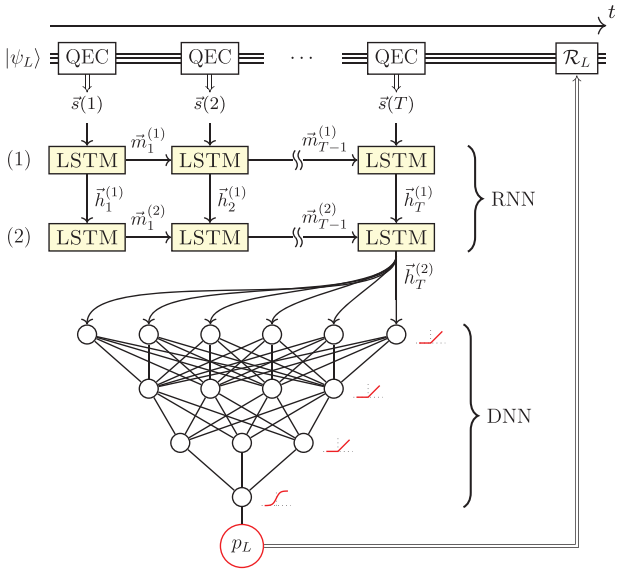

Figure 3. Recurrent neural network (RNN) architecture for logical error correction given a syndrome-flag volume based on Ref. [51]. All stabilizer parities and flags are measured sequentially (QEC block), and the network passes the respective measurement outcomes through two RNN layers using the LSTM architecture. The interlayer messages $\vec{m}_i^{(l)}$ consist of an internal 'cell state' $\vec{c}_i^{(l)}$ combined with the messages $\vec{h}_i^{(l)}$. This is followed by a dense neural network (DNN) for post-processing. The latter network consists of multiple layers of neurons, each equipped with an activation function, indicated schematically by the small red graphs. All but the output neurons have a rectified linear unit (ReLU) as activation function, and the output neuron is equipped with a smooth sigmoid function. The output is the predicted logical flip probability. If a flip is detected, a recovery operation $\mathcal{R}_L$ is performed by applying the corresponding logical operator.

neurons, layer-wise interconnected by trainable-weight matrices. By means of these trainable parameters, the neuron value of the previous layer and a non-linear function, the neuron values of the next layer are calculated. The underlying idea of this structure is that the second-stage DNN can perform a post-processing step to compute the logical flip parity after the RNN has extracted the necessary information from the syndrome-flag time series. The output of the joint network is the probability for the occurrence of a logical flip. A restriction of this architecture is that it can only detect logical errors in either the X or the Z basis at a time, meaning that two networks have to be trained for complete decoding. We will denote this type of decoding as single-headed recurrent neural network decoding (SRNN); it will be contrasted with a network that predicts both the logical phase- and bit- flip parity in a single run. This double-headed recurrent neural network (DRNN)[48, 51] and its performance will be discussed in Section 5.

Simulation of the quantum memory experiment and training — To train the network in a supervised manner, a sufficiently large set of tagged data examples i.e., tuples of (syndrome-flag-volume, logical parity) information is required. We generate such training data by the following procedure:

1. A known logical state $|\psi_L\rangle$ is prepared by means of a noisy projective measurement of stabilizers. For the SRNN training, we choose $|\psi_L\rangle \in \{X_L^{m_x}|0_L\rangle\}_{m_x=0,1}$ for the bit flip decoder and $|\psi_L\rangle \in \{Z_L^{m_z}|+_L\rangle\}_{m_z=0,1}$ for the phase flip decoder. For the DRNN, we choose $|\psi_L\rangle \in \{X_L^{m_x}|0_L\rangle, Z_L^{m_z}|+_L\rangle\}_{m_x=0,1;\, m_z=0,1}$. Each state obtains an equal amount of samples, and the logical parity information of the input state $m_{x/z}^{\text{in}} \in \{0,1\}$ is stored.
2. A number T of stabilizer measurement cycles is performed, producing a sequence of T syndrome plus flag vectors, $\vec{s}(1), \ldots, \vec{s}(T)$, i.e., the schedule in Figure 2b is repeated T times.
3. The data qubits are measured and a classical round of error correction is conducted, upon which the final logical parity m_{out} is well-defined.
4. An output label for the NN training is defined as the logical flip parity $m_{x/z}^L = m_{x/z}^{\text{in}} + m_{x/z}^{\text{out}} \mod 2$.

The vector of all syndrome and flag measurements, denoted as $\vec{s}$, is then used as the training input, whereas the logical flip parity m^L may be used as the output label. After the respective NN model has converged under the training, it is tested with unseen data that can be sampled using the same procedure but was not used for the training before.

Decoding results— The error correction capabilities of the fully trained RNN decoders, including whether they are fault-tolerant, are presented in Figure 2d. Here, the logical error rate per readout round of the described RNN decoders is compared with the one of a flag based sequential look-up table decoder (LUT) over a range of physical error rates. The LUT works by sequentially reading the syndrome and flag values in the readout sequence and performing the most probable correction of fault weight one based on at most two readout round intervals. This implies that all single faults are corrected by the LUT, including hook errors, for more details see.[87] The LUT can be considered as the simplest flag-FT decoder and is used as a benchmark. Both the LUT and the RNN decoders show a scaling of the logical error rate as $p_L \propto p_{ph}^2$ (see Figure 2d). This indicates a fault-tolerant decoding as only two independent faults, each occurring with a probability of $\mathcal{O}(p)$ cause a logical error. The RNN decoders exhibit larger pseudo-thresholds and overall a smaller logical error rate compared to the LUT, indicating that they are capable to correct more faults of weight two or larger by exploiting information and correlations contained in the global set of measurement outcomes from the multi-round syndrome-flag stabilizer readout sequence. By pseudo-threshold, we denominate the error rate at which the respective logical error rate breaks even with the physical error rate, $p_L = p_{ph}$. Hence, one can conclude that the employed NN structure is capable to process the flag information correctly and that it yields, for the present case, clearly better performance than a standard, LUT based decoding approach.

3. NN Decoder Interpretability

The problem of explaining complex models can be viewed as the challenge to create a simpler proxy-model that is accurate to a satisfactory degree. Hence, any explanation for a model can itself be seen as a model. Complex models, such as NNs, are often referred to as 'black-box' models, in which the inner workings are not known or understood. Even though neural networks are built out of modular, simple, and explainable elements, their 'black-

box' nature emerges due to the high degree of connectivity, nonlinearity and from many degrees of freedom.[97, 98] For these reasons, approximations of readily trained NNs may be developed to serve as interpretability models. These explanatory models typically only capture a particular working regime of the model and are therefore only "locally accurate," such as being an approximation around particular input values. Despite this limitation, they can still be useful to understand the underlying complex model such as a NN.

A common method of interpreting NNs is to analyze how the network transforms the input space into the output space by assigning credit to input features. Various heuristic methods have been devised to derive such credit assignment scores.[69, 76, 99] In our work, we will focus on one of these scores, the so-called Shapley value; an alternative choice is outlined in the supplementary material.[87]

The Shapley value— Consider a game where a coalition of N players work together to produce a shared result. The result may be assigned a numerical value, for instance, prize money in a competition. The goal is to fairly assign credit to each individual player for their contribution to achieving the shared result. To assign credit to a single player in the coalition, one analyzes how the coalition performs with that player compared to how the team would, hypothetically, perform without that player. It may be the case that some players only perform well when other players are present as well, or that certain combinations of players actually reduce the effectiveness of a team. For this reason, it is necessary to consider all possible subsets of players to accurately capture interactions between players in defining a good measure of contribution for the individual players. A solution to this problem was found by Shapley.[80] We formally introduce the concept of the Shapley value following Ref. [100].

To begin, we define a few quantities for discussion about n-player games.

Definition 1 (Player set). *Let $\mathcal{N} = \{1, 2, \dots, N\}$ be the set of all players. Each non-empty subset $\mathcal{S} \subseteq \mathcal{N}$ is called a coalition. The set $\mathcal{N}$ is named the 'grand coalition'.*

Definition 2 (Cooperative game). *A cooperative game is defined by the pair $(\mathcal{N}, v)$ where $v : 2^{|\mathcal{N}|} \to \mathbb{R}$ is called the characteristic function, which maps any coalition $\mathcal{S}$ to a real number.*

A game in this sense is defined only by its participants and its characteristic function, without any consideration of the internal workings of the game. The characteristic function may be interpreted as the payoff of the game. The Shapley value is a specific solution concept for the characteristic function and defined as follows:

Definition 3 (Shapley value). *The Shapley value ϕ_i is a single-valued solution concept, given by*

$$\phi_i(\mathcal{N}, v) = \frac{1}{|\mathcal{N}|} \sum_{\mathcal{S} \subseteq \mathcal{N} \setminus \{i\}} \binom{|\mathcal{N}| - 1}{|\mathcal{S}|}^{-1} \underbrace{(v(\mathcal{S} \cup \{i\}) - v(\mathcal{S}))}_{\text{marginal contribution}} \tag{1}$$

It may be interpreted as the average marginal contribution of a player $i \in \mathcal{N}$ over all possible subsets of players $\mathcal{S} \in \mathcal{N} \setminus \{i\}$. The marginal contribution is the value a player i would add to a specific coalition $\mathcal{S}$. The Shapley value has many interesting properties,[87, 100] however its exact calculation is expensive as its computation cost scales as $O(2^{|\mathcal{N}|})$. When evaluating large neural networks that process many input features, it is therefore essential to come up with efficient approximations to the Shapley value. We provide references for a number of state-of-the-art approximations for universal explainability, including linear regression approaches[69, 101–103] and Monte Carlo sampling[104] all with $O(|\mathcal{N}|)$ time complexity. In this work, however, we use a neural network specific method called DeepSHAP introduced recently by Chen *et al.*,[101] which is included as part of the open source SHAP library.[105] It makes use of the linearization of a NN that takes place in the back-propagation procedure. Employing this linear approximation and performing a modified back-propagation, it can be shown that for expanding around an ensemble of average neuron values, the approximate Shapley value of the input features can be obtained. For a rigorous derivation, we refer the reader to Refs. [87, 101]. Note that furthermore, the approximate Shapley value of neurons in intermediate network layers can also be computed. Studying these values, however, is not part of this work, but we note that it can be an additional tool to understand or optimize a NN.

Decoder interpretation— Given a trained NN-based decoder and the DeepSHAP method to approximate the Shapley value efficiently, we now obtain an importance score for each syndrome and flag bit in every individual measurement sequence. A minimal example for such an assignment of importance is illustrated in **Figure 4**, where a RNN is asked to decode two rounds of flagged Steane-code stabilizer measurements. On this individual level, especially for a larger number of measurement rounds, it is possible to understand from the Shapley value distribution how the RNN regards certain syndrome(-flag) combinations to determine the logical flip parity. For instance, low Shapley values for a temporally separated pair of syndrome excitations can be read as the RNN recognizing this as a measurement error. Further, given the prediction of a logical parity flip, one can understand that the syndrome excitations of the largest Shapley values are attributed to the faults that lead to this predicted flip.

Apart from interpreting the decoding decision based on this importance score for individual inputs, one can further interpret the global working of the NN by analyzing the statistics of Shapley values. One interesting aspect of the inner decision logic of a NN decoder is present in the context of flag-fault-tolerant stabilizer measurements during a quantum memory experiment. Here, certain combinations of nontrivial syndrome flag bit pairs are supposed to hold greater importance as they are expected to indicate a logical flip. This suggests analyzing the statistics of Shapley values by inferring the correlation of these pairs of Shapley values corresponding to the mentioned syndrome flag bits. These so-called Shapley value or plainly Shapley correlations can be calculated by determining the Shapley values of all syndrome and flag bits of all input samples and then by empirically calculating their correlation. We calculate the mutual correlation matrix of Shapley values of syndrome and flag bits that are measured in the same round of syndrome extraction, i.e. $\Delta\tau = 0$ in **Figure 5a**.

Indeed, we can identify an excess in the correlation of Shapley values for syndrome-flag bit-pairs that are indicative for hook errors. Generally, there are three syndrome-flag pairings that indicate hook errors. Their Shapley-value correlation can be seen

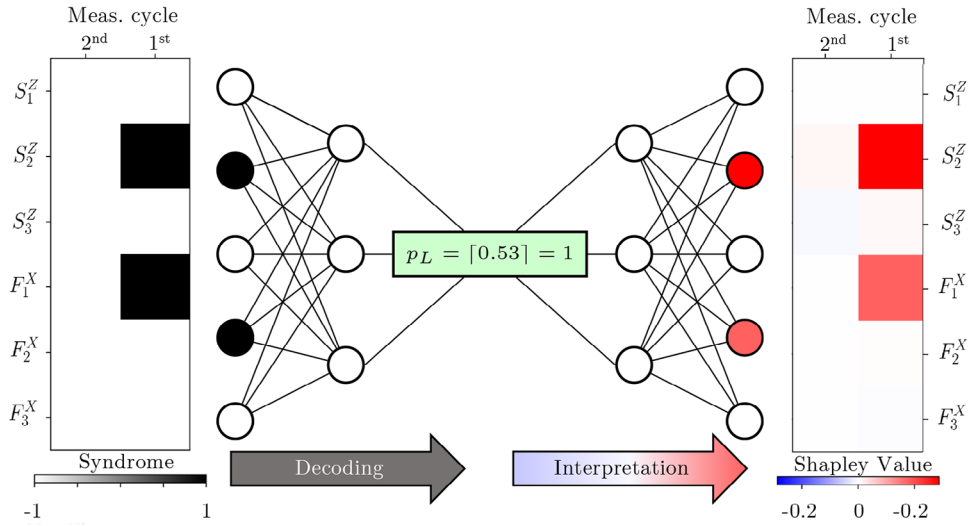

Figure 4. Minimal example of the interpretation procedure of the decoding of a two-round stabilizer measurement sequence. On the left side, the measured syndrome is fed into the NN decoder, which predicts that a hook error and therefore a logical error modulo final correction has occurred. Based on this run of the decoder (input, neuron activations), Shapley values for the input bits are calculated via backpropagation through the NN. These Shapley values are shown in color-coded form on the right side.

in Figure 5a. This is a positive verification of the NN's capability to understand the syndrome flag logic to render the propagation of dangerous errors identifiable. Note that these signatures of hook errors are observed for the bit-flip decoder in the $f_X s_Z$ Shapley correlations of the same measurement round $\Delta\tau = 0$. Similarly, for the phase-flip decoder hook error signatures in the $f_Z s_X$ Shapley correlations are recognized for $\Delta\tau = 1$. Generally, the specific pairings of syndrome flag bits, which are signatures for hook errors, depend on the specifics of the quantum circuitry of the syndrome flag measurement. An example of a fault prop-

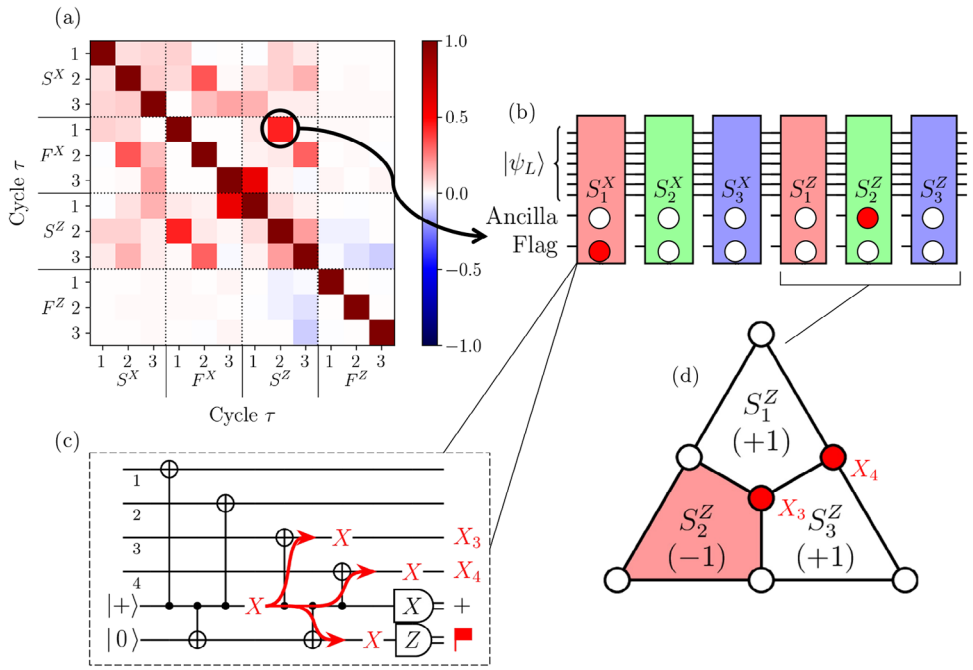

Figure 5. a) The mutual correlation matrix of the Shapley values of syndrome and flag bits of the same QEC round ($\Delta t = 0$). Each three-bit segment bundles one type of bits: syndrome parity or flag parity and X or Z type. The excess of correlation of the Shapley values of the stabilizer bit S_2^Z and flag bit F_1^X is highlighted. b) Illustration of the corresponding hook error signature in a QEC cycle. During the first (X type) stabilizer readout, a fault on the ancilla qubit takes place. The corresponding error propagation on the circuit level is as depicted in c). Besides the error propagation onto the data qubits, it is also shown how the flag qubit picks up the error. Further, the weight 2 data-qubit error will be picked up as a Z-syndrome in the subsequent (Z type) stabilizer readout. In d), this Z-syndrome on the Steane code is shown.

agation, and how it appears as a syndrome and flag combination during the stabilizer readout that we simulate, is illustrated in Figure 5b–d.

Overall, flag bits that are measured whilst the readout of stabilizers of a specific basis show the tendency to mostly correlate with syndrome bits of the respective other basis, as expected. That is to say, f_X flag bits are relevant to determine logical errors mainly in combination with s_Z bits and vice versa. This behavior is consistent with the possible error propagations during the stabilizer measurement, where the propagation of a hook error is shown in Figure 5c. Further, the Shapley values corresponding to syndrome values of the same basis are correlated, as these natively indicate the errors on data qubits of the code. During the repeated stabilizer measurements, single data qubit errors can lead to a logical fault upon being accumulated. Therefore, to perform a good decoding, it is relevant for the NN decoder to consider pairs of syndrome values corresponding to the same stabilizer type, Z or X, respectively.

The correlation of the importance between X and Z syndrome bits is less strongly pronounced. Still, it is present at a statistically significant level. This type of Shapley correlation is an indicator for the network recognizing correlations X and Z errors in the noise processes, which is present in our depolarizing noise model due to the presence of independent Y errors. The correlation of X and Z errors naturally carries over to the syndrome data, where now for instance the X syndrome carries partial information on the presence of X errors. If the NN decoder is given the full syndrome information, one might ask whether the NN has detected such correlation at all during the training or whether it is able to make use of it. By analyzing the Shapley correlation of X and Z syndrome bits, this question can be answered quantitatively, as shown in Figure 5. Hierarchically speaking, the Shapley correlations of the X and Z syndrome are weaker than the mutual correlation among one syndrome species. This is expected as the correlation due to Y errors can only be an additional but not the main factor for determining a decoding decision. The bare presence of Shapley correlations of X and Z syndrome bits alone shows, however, that the NN decoder, after training, is aware of the Y errors. Summarizing the hierarchy of Shapley correlations between syndrome-flag bits, the pairing that indicate hook errors are pronounced most strongly, as here single errors, if decoded incorrectly, can cause a logical flip. Other syndrome-flag pairs of the type (S^Z, F^X) as well as pairing between syndromes still show modest Shapley-value correlations.

4. Monitoring the Learning of Fault Tolerance

It is an interesting question to ask how a NN learns certain competences during the training. Being able to answer this might provide insights that can help in optimizing the learning process or to stop it at a suitable point. In the context of fault tolerance in decoding, we aim to investigate this question in order to reliably determine when a NN decoder has learned to correct all single faults (for a distance-3 code). We analyze thereby the previously introduced RNN decoder and how it is learning to handle FT-breaking hook errors correctly in the Steane code with a flag-based readout scheme.

In order to do so, we approximate Shapley values for network instances during the training. As in the previous discussion, we aim to interpret the networks' capability to correct for hook errors by means of Shapley value correlations that align to signatures of hook errors. Such Shapley correlations are now resolved over the training time. Hence, it is of interest how and when the RNN decoder becomes fault-tolerant and whether we can deduce a good indicator of this learning transition in terms of the Shapley correlation.

To this end, the RNN model is saved after each training epoch; a training epoch being defined as the RNN having seen all training data once in a batchwise fashion. The latter means that the RNN is updated based on a gradient estimation with a subset of the training data, namely a batch of it. For each of these network instances and a verification data set, the Shapley values are calculated, and the decoder performance is tested. We quantify this performance in different ways: First, in **Figure 6b** we show how the fidelity for different physical error rates behaves along the training. Further, we access in Figure 6c whether the RNN decoder training instances are FT. To this end, we show (i) the scaling exponent a of the logical error rate per round against the physical error rate, i.e. $p_L \propto p_{ph}^a$, where a value close to 2 would indicate FT as at least two faults are needed to induce a logical error. Further, (ii) the figure shows the fraction of logical faults occurring for the generation of all possible weight-one fault trajectories for two QEC cycles using a deterministic error placer (DEP). The DEP works by iterating over all possible circuit locations where a fault could occur, and placing the fault to evaluate whether the RNN decoder can recover the initial logical state or not. Furthermore, we aim to quantify the evolution of how well the NN understands to consider the FT-breaking hook error signatures for decoding. We do this by means of the correlations between Shapley values as shown in Figure 6d, where we compare two classes of Shapley value correlations: one corresponds to the hook error signatures, while the other is not related to the question of FT and serves as a baseline.

All of these figures of merit are evaluated for a single-output LSTM bit-flip decoder trained using 100.000 training samples. Without decoding, a logical error occurs roughly for 4% of the weight-one fault circuits in two QEC rounds as sampled with the DEP. Given this, one can understand the early stages of the training of the network. Here, as can be seen in Figure 6c, the failure rate of the RNN decoder is 4% for the first ∼20 training epochs. While the network is on this plateau, it always assigns the trivial correction, which is not to perform any correction, independent of the syndrome. Note that also the infidelity in Figure 6b starts off with a plateau, which as well corresponds to the proportion of Monte Carlo samples which do lead to a logical error. Eventually, for the shown training process, the NN leaves the local loss-function minimum of assigning the trivial correction. At around epoch 16, the decoder starts to be able to decode the first few logical errors effectively, and it becomes fault-tolerant at epoch 27. At around the same time, the infidelity starts decreasing, but it keeps decreasing well beyond the point where the network achieves fault-tolerance.

A corresponding evolution of the averaged hook error Shapley correlations for the bit-flip SRNN decoder is shown in Figure 6d, where as a reference the average Shapley correlations of syndrome flag bit combinations, which do not correspond to a hook error, is shown. Interestingly, both curves start to deviate strongly

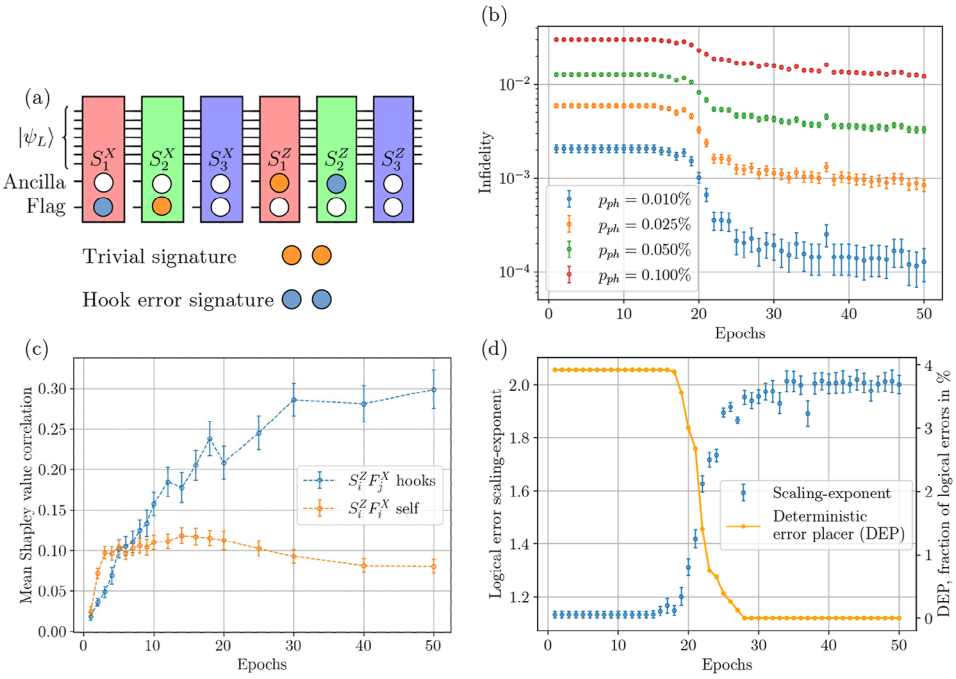

Figure 6. Monitoring the training of the RNN over 50 epochs. The learning of the handling of FT signatures (example in (a), blue) by the NN is shown. For that in a) the two syndrome-flag pairs are illustrated. Here, the trivial (orange) combination corresponds to a weight-one data qubit error, while the hook error signature (blue) indicates a weight-two data qubit error. In b) the logical infidelity is displayed for different physical error rates. Sub-figure c) shows the correlation of Shapley values of pairs of syndrome and flag bits that either indicate a hook error (blue) or are not relevant for FT (orange). According examples are given in (a). In d) two figures of merit to quantify the FT of the decoder are shown. In blue, the scaling exponent of the logical error rate per round with the physical error rate is plotted. Further, a deterministic error-placer (DEP) is employed that iterates over all possible single faults in two rounds of QEC. For these instances, the orange curve shows the ratio of single faults that are uncorrectable for the RNN decoder.

when the FT performance indicators start to improve. It seems that all Shapley correlations increase first, although the ones that do not correspond to hook errors are quickly overtaken by the ones corresponding to bit-flip hook error configurations. After this break-even, they flatten out and start decreasing again. One can conjecture that the network initially decreases the loss function by attributing importance to incorrect flag-syndrome combinations, perhaps simply using the unspecific activation of both, syndrome and flag as a marker of logical errors. But then the NN quickly learns that only specific combinations are important for a good decoding. Altogether, the deviation of the two correlations can be used to track the starting point of the processes to learn the significance of the flag bit for the NN, yielding a first indicator for the start of the FT learning process. Around epoch 30, the NN has managed to become FT as the logical error scaling exponent reaches a stable value of 2. This FT decoding behavior is corroborated by the DEP test, which does not show any residual uncorrected weight-one faults. Accordingly, the Shapley value correlation of the bit-flip hook configurations flattens out above a value of 0.25. This logistic behavior can be taken as a signal for the termination of the learning process of FT. The performance in terms of the logical error rate, however, continues to decrease further and one could even see minimal improvement beyond epoch 50. The reason for this is the learning of how to correct more weight-2 faults that occur. Despite this being an, in principle, desirable effect, one needs to carefully choose the point of termination of training as this learning of specific higher-weight faults, present in the training data, might be seen as over-fitting. For this reason, it is important to evaluate the network performance and properties for an independent data set as done in this analysis.

5. Diagnosing NN Malfunction for the Two-headed NN

The second Steane QEC decoder we will analyze in more detail in the following is based on a recurrent neural network that has two output neurons, each for predicting one of either logical bit or phase flip error parity. This network structure can be seen as an augmentation of the previous one, and it is also similar to the ones studied in Refs. [48, 51]. This means that both error types can be decoded within one forward-pass of the network, given the full syndrome flag volume as input. Using such a network, keeping the overall network size fixed, practically halves the computational cost of the decoder. However, the more important motivation for this proposed architecture is to harness the correlation between X and Z syndrome more directly to potentially improve the decoding performance. The hope is that by predicting the logical X and Z parity in one pass, the network can more efficiently integrate these correlations in its decision-making process as both syndrome species are treated on the same footing in the training procedure. To underline this reasoning, one can think of the single output architecture; for it, half of the input information (e.g., the Z syndrome to correct for X errors) has far greater importance to compute the logical parity compared to the

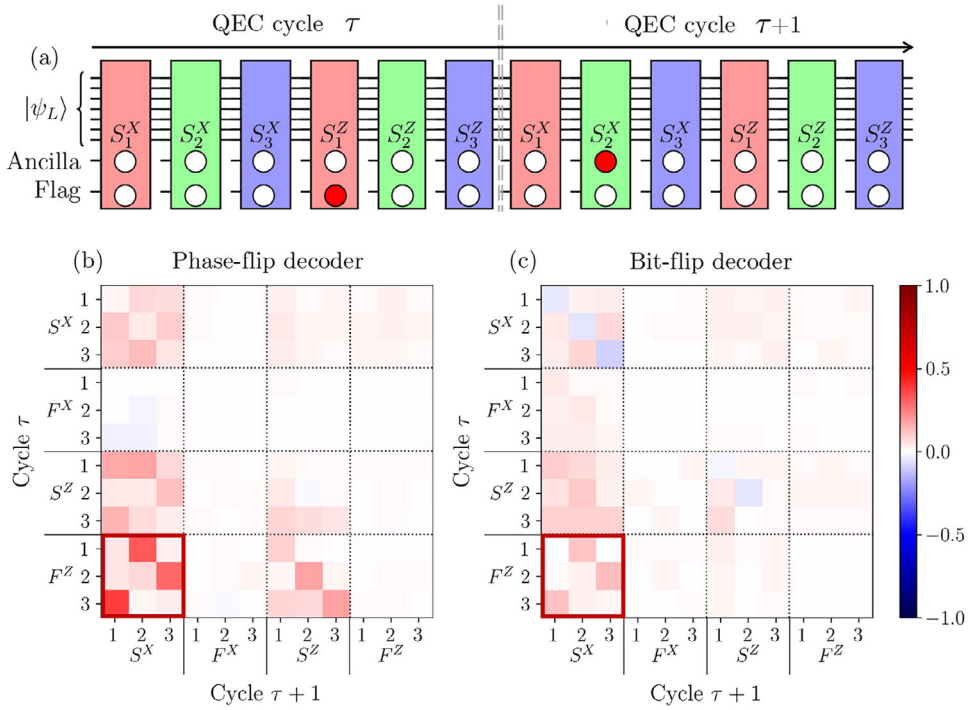

Figure 7. The Shapley-correlation analysis for the dual output RNN decoder, that is predicting simultaneously logical bit and phase flip parity. In a) the typical syndrome flag signature for a *Z*-type hook error is shown, which stretches over two consecutive QEC cycles. In b) the Shapley correlation for the logical phase flip decoding is shown, the quadrant of the expected syndrome flag correlation excess according to the hook signatures is highlighted. In c) the Shapley value correlations between syndrome and flag bits of two consecutive rounds for the logical bit flip decoder are shown. An excess of Shapley value correlations in the same quadrant as for the phase flip decoder can be observed. These syndrome flag pair, however, carry no special information on the logical bit flip parity. In consequence, we can take these correlations as a sign for an incorrect use of phase-flip decoder logic to detect logical bit flips, pointing to a malfunction of the network.

other part such that the less important part of the input might be pathologically underused.

For training the network, one must consider that labels of the desired decoder output are incomplete for each sample since for each execution of the memory experiment, the logical state can only be read out in the *X or* the *Z* basis as it would be the case in a real-world experiment. This problem can be mitigated by using, alternatingly, only one of the output neurons for the backpropagation during training. In this setting, the cost function that is minimized adapts dynamically depending on the measurement basis to evaluate the correctness of the prediction of the appropriate output. Upon successfully training this new NN decoder, we will now investigate its performance and Shapley value correlations. In Figure 2, the logical error rate of the dual-output decoder can be compared to two single output LSTM networks. The single-output network variant outperforms the new architecture by quite a margin, reaching a pseudo-threshold at physical error rates roughly twice as high as for the dual-output decoder. We conjecture that there may be some unwanted internal network interference between the two computations of the respective logical parities, which causes the dual output LSTM network to underperform. This observation opposes the architectural ansatz to improve the decoding precision by having a dual output.

In the following, we analyze the decision-making of the dual-output NN decoder based on the Shapley interpretability method further. Let us first consider the syndrome-flag signature of hook errors that would cause a logical phase flip. These generally span over two consecutive QEC cycles. An example of such a syndrome flag pair is illustrated in **Figure** 7a and rigorously all such signatures are given by the combinations $(F^t_{Z_i} S^{t+\Delta t}_{X_j})$ at $\Delta t = 1$ for $j = i + 1 \mod 3$. Naturally, it is important for the phase-flip decoder to recognize these signatures. In the correlation plots of Figure 7 (*b*) this can be seen as a correction excess when considering the dual RNN decoding of initial $|\pm_L\rangle$ states against logical phase flips. In sub-figure Figure 7 (*c*) the Shapley correlations for the bit-flip decoding with the dual RNN decoder is shown for syndrome-flag combinations of subsequent readout rounds, i.e. $\Delta t = 1$. These correlations are not expected to show any excess, as no bit-flip hook error signatures exist here. On the contrary, the correlation excess of the phase-flip decoding in sub-figure, Figure 7 (*c*) can be observed as an artifact in the correlation structure of the bit-flip decoder in sub-figure Figure 7 (*d*).

This signature is an indicator of a badly tuned network where both species of syndrome and flag information influence the decoding decisions mutually. This mutual influence should in principle enable a better correction of correlated *X* and *Z* errors that originate from *Y* errors. We observe the opposite, that the performance of respective bit- and phase-flip decoders is decreased, as shown in Figure 2d. The understanding of this observation can now be aided by the previous discussion of Shapley correlations in the bit- and the phase-flip decoder. It seems that the RNN is not able to discriminate the respective information on

bit- and phase-flip errors to a sufficient degree. Correspondingly, in this architecture, the phase-flip error-information can be seen as noise for the bit-flip decoding, rather than as additional information and vice versa. It should however be noted that the correct hook error configurations are still most pronounced and that the dual-output network can still decode both syndrome bases fault-tolerantly. Altogether, this is a minimal example of the interpretation of a NN decoder where light could be shed on an internal malfunction that was noticeable as a sub-optimally performing decoder. As a consequence, for the double-output architecture, we observe that this malfunction overshadows any positive effect of exploiting correlations of X and Z errors, which should potentially be more feasible for this NN architecture.

6. Conclusion and Outlook

In this work, we have employed an efficient local interpretation method for NNs, the Shapley value approximator DeepSHAP.[101] We have used it to explain the decoding decision of a RNN-based decoder for fault-tolerant operation of the Steane QEC code. The specific LSTM-based RNN architectures we analyze as examples are inspired by earlier works.[48, 51] Our simulations confirm that these networks can be trained to become fault-tolerant decoders, and that they outperform sequential look-up-table decoders by a significant margin. We derive an indicator, based on Shapley value correlations, of the NN for having learned a fault-tolerant decoding behavior. This understanding is based on the flag-fault-tolerant readout scheme that is used to measure the stabilizer generators of the Steane code. Evaluating appropriate Shapley value correlations is thereby independent of the performance analysis of the decoder. For simulated quantum memory experiments, the analysis of the scaling of the logical error rate with the physical error rate or the extensive placing of errors is a viable way to confirm FT. In an experimental setting, however, where this is not possible, the analysis of Shapley value correlations yields an alternative criterion.

In our work, we further analyze the learning of correlations by the recurrent neural networks to determine a decoding decision. Most notably, we can identify all hook error signatures and correlations between X and Z errors. For future works, it can be interesting to consider other importance scores than the Shapley value which are suited more naturally to calculate feature-correlations, such as, e.g., the so-called 'Shapley interaction values'.[77, 100] Lastly, we present a dual output LSTM network, decoding X and Z in parallel, which shows suboptimal performance after training. Employing our interpretability framework, this suboptimal performance can be understood. We suggest this method as a tool to analyze shortcomings of NN decoders also in other settings to aid in the network-engineering.

In future work, it may be interesting to observe if and how neural-network decoders are able to handle noise models which include error types such as qubit loss, spatially correlated errors, biased noise, and non-Markovian noise. Moreover, it would be interesting to see how XAI explanation techniques can be used to optimize performance of NN-based decoders, both for the discussed Steane code but also for different and larger codes as well as for general more complex FT protocols. Here, interpretable neural-network decoders could aid in the theoretical analysis of such larger codes by identifying and highlighting key features which neural networks were able to learn. Going beyond the task of decoding, the interpretability methods presented in this work are also applicable for other NN-based solutions in the wider field of quantum error correction and quantum computing. Here, depending on the problem setting one has to clarify the relevant interpretability questions to be asked as we have done in this work in the context of error propagation and FT. It would furthermore be interesting to analyze the internal distribution of Shapley values in the NN to open the black box even wider.

Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements

The authors acknowledge support by the Deutsche Forschungsgemeinschaft through Grant No. 449905436 and the ERC Starting Grant QNets through Grant Number 804247. M.M. furthermore acknowledges funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – Cluster of Excellence Matter and Light for Quantum Computing (ML4Q) EXC 2004/1 – 390534769, funding by the German federal ministry of research, technology and space (BMFTR) via the VDI within the project NeuQuant (project number 13N17065), funding by the U.S. ARO Grant No. W911NF-21-1-0007, and from the European Union's Horizon Europe research and innovation programme under grant agreement No 101114305 ("MILLENION-SGA1" EU Project). This research is also part of the Munich Quantum Valley (K-8), which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus. The authors gratefully acknowledge the computing time provided to them at the NHR Center NHR4CES at RWTH Aachen University (project number p0020074). This was funded by the Federal Ministry of Education and Research, and the state governments participating on the basis of the resolutions of the GWK for national high performance computing at universities.

Open access funding enabled and organized by Projekt DEAL.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions Statement

L.B. and L.K. contributed equally to this project. L.B. conceived the project idea, L.K. set up the numerical tools and code. Both L.B. and L.K. performed the simulations, analyzed the data and prepared the figures. All authors contributed to the writing of the manuscript. L.B. and M.M. supervised the project.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Keywords

explainable artificial intelligence, fault tolerance, interpretability, neural network decoding, quantum error correction

Received: February 27, 2025
Revised: May 23, 2025
Published online: July 9, 2025

[1] A. M. Dalzell, S. McArdle, M. Berta, P. Bienias, C.-F. Chen, A. Gilyén, C. T. Hann, M. J. Kastoryano, E. T. Khabiboulline, A. Kubica, G. Salton, S. Wang, F. G. S. L. Brandão, *arXiv:2310.03011* **2023**.
[2] R. Stricker, D. Vodola, A. Erhard, L. Postler, M. Meth, M. Ringbauer, P. Schindler, T. Monz, M. Müller, R. Blatt, *Nature* **2020**, *585*, 207.
[3] J. Hilder, D. Pijn, O. Onishchenko, A. Stahl, M. Orth, B. Lekitsch, A. Rodriguez-Blanco, M. Müller, F. Schmidt-Kaler, U. Poschinger, *Phys. Rev. X* **2022**, *12*, 011032.
[4] L. Egan, D. M. Debroy, C. Noel, A. Risinger, D. Zhu, D. Biswas, M. Newman, M. Li, K. R. Brown, M. Cetina, C. Monroe, *Nature* **2021**, *598*, 281.
[5] L. Postler, S. Heußen, I. Pogorelov, M. Rispler, T. Feldker, M. Meth, C. D. Marciniak, R. Stricker, M. Ringbauer, R. Blatt, P. Schindler, M. Müller, T. Monz, *Nature* **2022**, *605*, 675.
[6] L. Postler, F. Butt, I. Pogorelov, C. D. Marciniak, S. Heußen, R. Blatt, P. Schindler, M. Rispler, M. Müller, T. Monz, *PRX Quantum* **2024**, *5*, 030326.
[7] C. Ryan-Anderson, J. G. Bohnet, K. Lee, D. Gresh, A. Hankin, J. P. Gaebler, D. Francois, A. Chernoguzov, D. Lucchetti, N. C. Brown, T. M. Gatterman, S. K. Halit, K. Gilmore, J. A. Gerber, B. Neyenhuis, D. Hayes, R. P. Stutz, *Phys. Rev. X* **2021**, *11*, 041058.
[8] C. Ryan-Anderson, N. Brown, M. Allman, B. Arkin, G. Asa-Attuah, C. Baldwin, J. Berg, J. Bohnet, S. Braxton, N. Burdick, J. P. Campora, A. Chernoguzov, J. Esposito, B. Evans, D. Francois, J. P. Gaebler, T. M. Gatterman, J. Gerber, K. Gilmore, D. Gresh, A. Hall, A. Hankin, J. Hostetter, D. Lucchetti, K. Mayer, J. Myers, B. Neyenhuis, J. Santiago, J. Sedlacek, T. Skripka, et al., *arXiv:2208.01863* **2022**.
[9] S. Huang, K. R. Brown, M. Cetina, *Sci. Adv.* **2024**, *10*, eadp2008.
[10] A. Paetznick, M. Da Silva, C. Ryan-Anderson, J. Bello-Rivas, J.P. Campora III, A. Chernoguzov, J. Dreiling, C. Foltz, F. Frachon, J. Gaebler, T. Gatterman, L. Grans-Samuelsson, D. Gresh, D. Hayes, N. Hewitt, C. Holliman, C. V. Horst, J. Johansen, P. Siegfried, A. Sundaram, D. Tom, S. J. Wernli, M. Zanner, R. P. Stutz, K. M. Svore, *arXiv:2404.02280* **2024**.
[11] C. Ryan-Anderson, N. Brown, C. Baldwin, J. Dreiling, C. Foltz, J. Gaebler, T. Gatterman, N. Hewitt, C. Holliman, C. Horst, J. Johansen, D. Lucchetti, T. Mengle, M. Matheny, Y. Matsuoka, K. Mayer, M. Mills, S. A. Moses, B. Neyenhuis, J. Pino, P. Siegfried, R. P. Stutz, J. Walker, D. Hayes, *arXiv:2404.16728* **2024**.
[12] B. W. Reichardt, D. Aasen, R. Chao, A. Chernoguzov, W. van Dam, J. P. Gaebler, D. Gresh, D. Lucchetti, M. Mills, S. A. Moses, B. Neyenhuis, A. Paetznick, A. Paz, P. E. Siegfried, M. P. da Silva, K. M. Svore, Z. Wang, M. Zanner, *arXiv:2409.04628* **2024**.
[13] I. Pogorelov, F. Butt, L. Postler, C. D. Marciniak, P. Schindler, M. Müller, T. Monz, *Nat. Phys.* **2025**, *21*, 298.
[14] M. Takita, A. W. Cross, A. D. Córcoles, J. M. Chow, J. M. Gambetta, *Phys. Rev. Lett.* **2017**, *119*, 180501.
[15] Google Quantum AI, *Nature* **2021**, *595*, 383.
[16] S. Krinner, N. Lacroix, A. Remm, A. Di Paolo, E. Genois, C. Leroux, C. Hellings, S. Lazar, F. Swiadek, J. Herrmann, G. J. Norris, C. K. Andersen, M. Müller, A. Blais, C. Eichler, A. Wallraff, *Nature* **2022**, *605*, 669.
[17] Y. Zhao, Y. Ye, H.-L. Huang, Y. Zhang, D. Wu, H. Guan, Q. Zhu, Z. Wei, T. He, S. Cao, F. Chen, T.-H. Chung, H. Deng, D. Fan, M. Gong, C. Guo, S. Guo, L. Han, N. Li, S. Li, Y. Li, F. Liang, J. Lin, H. Qian, H. Rong, H. Su, L. Sun, S. Wang, Y. Wu, et al., *Phys. Rev. Lett.* **2022**, *129*, 030501.
[18] Google Quantum AI, *Nature* **2023**, *614*, 676.
[19] R. S. Gupta, N. Sundaresan, T. Alexander, C. J. Wood, S. T. Merkel, M. B. Healy, M. Hillenbrand, T. Jochym-O'Connor, J. R. Wootton, T. J. Yoder, A. W. Cross, M. Takita, B. J. Brown, *Nature* **2024**, *625*, 259.
[20] Google Quantum AI and Collaborators, *Nature* **2025**, *638*, 920.
[21] N. Lacroix, A. Bourassa, Google Quantum AI, *Nature* **2025**.
[22] I. Besedin, M. Kerschbaum, J. Knoll, I. Hesner, L. Bödeker, L. Colmenarez, L. Hofele, N. Lacroix, C. Hellings, F. Swiadek, A. Flasby, M. B. Panah, D. C. Zanuz, M. Müller, A. Wallraff, *arXiv:2501.04612* **2025**.
[23] T. M. Graham, Y. Song, J. Scott, C. Poole, L. Phuttitarn, K. Jooya, P. Eichler, X. Jiang, A. Marra, B. Grinkemeyer, M. Kwon, M. Ebert, J. Cherek, M. T. Lichtman, M. Gillette, J. Gilbert, D. Bowman, T. Ballance, C. Campbell, E. D. Dahl, O. Crawford, N. S. Blunt, B. Rogers, T. Noel, M. Saffman, *Nature* **2022**, *604*, 457.
[24] I. Cong, H. Levine, A. Keesling, D. Bluvstein, S.-T. Wang, M. D. Lukin, *Phys. Rev. X* **2022**, *12*, 021049.
[25] D. Bluvstein, H. Levine, G. Semeghini, T. T. Wang, S. Ebadi, M. Kalinowski, A. Keesling, N. Maskara, H. Pichler, M. Greiner, V. Vuletić, M. D. Lukin, *Nature* **2022**, *604*, 451.
[26] D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter, J. P. B. Ataides, N. Maskara, I. Cong, X. Gao, P. S. Rodriguez, T. Karolyshyn, G. Semeghini, M. J. Gullans, M. Greiner, V. Vuletic, M. D. Lukin, *Nature* **2024**, *626*, 58.
[27] P. S. Rodriguez, J. M. Robinson, P. N. Jepsen, Z. He, C. Duckering, C. Zhao, K.-H. Wu, J. Campo, K. Bagnall, M. Kwon, T. Karolyshyn, P. Weinberg, M. Cain, S. J. Evered, A. A. Geim, M. Kalinowski, S. H. Li, T. Manovitz, J. Amato-Grill, J. I. Basham, L. Bernstein, B. Braverman, A. Bylinskii, A. Choukri, R. DeAngelo, F. Fang, C. Fieweger, P. Frederick, D. Haines, M. Hamdan, *arXiv:2412.15165* **2024**.
[28] M. Bedalov, M. Blakely, P. Buttler, C. Carnahan, F. T. Chong, W. C. Chung, D. C. Cole, P. Goiporia, P. Gokhale, B. Heim, Garrett T. Hickman, E. B. Jones, R. A. Jones, P. Khalate, J.-S. Kim, K. W. Kuper, M. T. Lichtman, S. Lee, D. Mason, N. A. Neff-Mallon, T. W. Noel, V. Omole, A. G. Radnaev, R. Rines, M. Saffman, E. Shabtai, M. H. Teo, B. Thotakura, T. Tomesh, A. K. Tucker, *arXiv:2412.07670* **2024**.
[29] D. Gottesman, *arXiv:quant-ph/9705052* **1997**.
[30] N. Sundaresan, T. Yoder, Y. Kim, M. Li, E. Chen, G. Harper, T. Thorbeck, A. Cross, A. Córcoles, M. Takita, *arXiv:2203.07205* **2022**.
[31] E. Dennis, A. Kitaev, A. Landahl, J. Preskill, *J. Math. Phys.* **2002**, *43*, 4452.
[32] H. Bombin, M. A. Martin-Delgado, *Phys. Rev. Lett.* **2006**, *97*, 180501.
[33] A. G. Fowler, *arXiv preprint arXiv:1307.1740* **2013**.
[34] S. Bravyi, M. Suchara, A. Vargo, *Phys. Rev. A* **2014**, *90*, 032326.
[35] G. Duclos-Cianci, D. Poulin, *Phys. Rev. Lett.* **2010**, *104*, 050504.
[36] O. Higgott, T. C. Bohdanowicz, A. Kubica, S. T. Flammia, E. T. Campbell, *Phys. Rev. X* **2023**, *13*, 031007.
[37] N. Delfosse, V. Londe, M. E. Beverland, *IEEE Trans. Inf. Theory* **2022**, *68*, 3187.
[38] N. Delfosse, A. Paetznick, J. Haah, M. B. Hastings, *arXiv:2309.15354* **2023**.
[39] N. Delfosse, *Phys. Rev. A* **2014**, *89*, 012317.
[40] C. Chamberland, A. Kubica, T. J. Yoder, G. Zhu, *New J. Phys.* **2020**, *22*, 023019.
[41] A. Kubica, N. Delfosse, *Quantum* **2023**, *7*, 929.
[42] J. F. S. Miguel, D. J. Williamson, B. J. Brown, *Quantum* **2023**, *7*, 940.
[43] P. Parrado-Rodríguez, M. Rispler, M. Müller, *Phys. Rev. A* **2022**, *106*, 032431.
[44] L. Berent, L. Burgholzer, P.-J. H. Derks, J. Eisert, R. Wille, *Quantum* **2024**, *8*, 1506.
[45] G. Torlai, R. G. Melko, *Phys. Rev. Lett.* **2017**, *119*, 030501.
[46] S. Varsamopoulos, B. Criger, K. Bertels, *Quantum Sci. Technol.* **2017**, *3*, 015004.

[47] S. Krastanov, L. Jiang, *Sci. Rep.* **2017**, *7*, 1.
[48] P. Baireuther, T. E. OBrien, B. Tarasinski, C. W. J. Beenakker, *Quantum* **2018**, *2*, 48.
[49] C. Chamberland, P. Ronagh, *Quantum Sci. Technol.* **2018**, *3*, 044002.
[50] S. Varsamopoulos, K. Bertels, C. G. Almudever, *IEEE Trans. Comput.* **2020**, *69*, 300.
[51] P. Baireuther, M. D. Caio, B. Criger, C. W. J. Beenakker, T. E. O'Brien, *New J. Phys.* **2019**, *21*, 013003.
[52] S. Varsamopoulos, K. Bertels, C. G. Almudever, *Quantum Mach. Intell.* **2020**, *2*, 1.
[53] P. Andreasson, J. Johansson, S. Liljestrand, M. Granath, *Quantum* **2019**, *3*, 183.
[54] A. Davaasuren, Y. Suzuki, K. Fujii, M. Koashi, *Phys. Rev. Res.* **2020**, *2*, 033399.
[55] K. Meinerz, C.-Y. Park, S. Trebst, *Phys. Rev. Lett.* **2022**, *128*, 080505.
[56] B. M. Varbanov, M. Serra-Peralta, D. Byfield, B. M. Terhal, *arXiv:2307.03280* **2023**.
[57] J. Bausch, A. W. Senior, F. J. H. Heras, T. Edlich, A. Davies, M. Newman, C. Jones, K. Satzinger, M. Y. Niu, S. Blackwell, G. Holland, D. Kafri, J. Atalaya, C. Gidney, D. Hassabis, S. Boixo, H. Neven, P. Kohli, *Nature* **2024**, *635*, 834.
[58] H. Cao, F. Pan, Y. Wang, P. Zhang, *arXiv:2307.09025* **2023**.
[59] S. Bordoni, S. Giagu, *Quantum Inf. Process.* **2023**, *22*, 151.
[60] S. Gicev, L. C. Hollenberg, M. Usman, *Quantum* **2023**, *7*, 1058.
[61] V. Ninkovic, O. Kundacina, D. Vukobratovic, C. Häger, A. G. i Amat, *arXiv:2408.05170* **2024**.
[62] A. S. Maan, A. Paler, *arXiv:2408.07038* **2024**.
[63] C. Vuillot, H. Asasi, Y. Wang, L. P. Pryadko, B. M. Terhal, *Phys. Rev. A* **2019**, *99*, 032344.
[64] K. H. Wan, A. Neville, S. Kolthammer, *Phys. Rev. Res.* **2020**, *2*, 043280.
[65] T. Peham, L. Schmid, L. Berent, M. Müller, R. Wille, *PRX Quantum* **2025**, *6*, 020330.
[66] Y. Zeng, W. Qin, Y.-H. Chen, C. Gneiting, F. Nori, *Phys. Rev. Lett.* **2025**, *134*, 060601.
[67] M. Puviani, S. Borah, R. Zen, J. Olle, F. Marquardt, *Phys. Rev. Lett.* **2025**, *134*, 020601.
[68] R. Zen, J. Olle, L. Colmenarez, M. Puviani, M. Müller, F. Marquardt, *arXiv:2402.17761* **2024**.
[69] S. Lundberg, S.-I. Lee, *arXiv:1705.07874* **2017**.
[70] C. Molnar, *Github* **2024**.
[71] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, W. Samek, *PloS one* **2015**, *10*, e0130140.
[72] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press, **2016**, http://www.deeplearningbook.org.
[73] M. T. Ribeiro, S. Singh, C. Guestrin, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* **2016**, pp. 1135–1144.
[74] F. Doshi-Velez, B. Kim, *arXiv:1702.08608* **2017**.
[75] G. Montavon, W. Samek, K.-R. Müller, *Digital Signal Process.* **2018**, *73*, 1.
[76] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, K.-R. Müller, *Explainable AI: interpreting, explaining and visualizing deep learning* **2019**, pp. 193–209.
[77] M. Sundararajan, K. Dhamdhere, A. Agarwal, *Proceedings of the 37th International Conference on Machine Learning* **2020**, *119*, 9259.
[78] M. Sundararajan, A. Taly, Q. Yan, *Proceedings of the 34th International Conference on Machine Learning* **2017**, *70*, 3319.
[79] D. Smilkov, N. Thorat, B. Kim, F. Viégas, M. Wattenberg, *arXiv:1706.03825* **2017**.
[80] L. S. Shapley, *Contributions to the Theory of Games* **1953**, *2*, 307.
[81] A. Shrikumar, P. Greenside, A. Kundaje, *Proceedings of Machine Learning Research* **2017**, *70*, 3145.
[82] C. Chamberland, M. E. Beverland, *Quantum* **2018**, *2*, 53.
[83] C. Chamberland, A. Kubica, T. J. Yoder, G. Zhu, *New J. Phys.* **2020**, *22*, 023019.
[84] R. Chao, B. W. Reichardt, *Phys. Rev. Lett.* **2018**, *121*, 5.
[85] A. M. Steane, *Phys. Rev. Lett.* **1996**, *77*, 793.
[86] H. Bombin, *arXiv:1311.0277* **2013**.
[87] *See supplemental material*.
[88] A. G. Fowler, M. Mariantoni, J. M. Martinis, A. N. Cleland, *Phys. Rev. A* **2012**, *86*, 032324.
[89] S. Huang, S. Puri, *arXiv:2311.03307* **2023**.
[90] L. Skoric, D. E. Browne, K. M. Barnes, N. I. Gillespie, E. T. Campbell, *Nat. Commun.* **2023**, *14*, 7040.
[91] Y. Bengio, P. Simard, P. Frasconi, *IEEE Trans. Neural Networks* **1994**, *5*, 157.
[92] Y. Tomita, K. M. Svore, *Phys. Rev. A* **2014**, *90*, 062320.
[93] A. M. Steane, *Phys. Rev. Lett.* **1997**, *78*, 2252.
[94] P. W. Shor, *Proceedings of the 37th Annual Symposium on Foundations of Computer Science* **1996**, 56.
[95] E. Knill, *Nature* **2005**, *434*, 39.
[96] B. W. Reichardt, *Quantum Sci. Technol.* **2020**, *6*, 015007.
[97] K. Hornik, M. Stinchcombe, H. White, *Neural Networks* **1989**, *2*, 359.
[98] S. H. Singh, F. van Breugel, R. P. N. Rao, B. W. Brunton, *Nat. Mach. Intell.* **2023**, *5*, 58.
[99] Y. Zhang, P. Tiňo, A. Leonardis, K. Tang, *IEEE Trans. Emerging Top. Comput. Intell.* **2021**, *5*, 726.
[100] B. Rozemberczki, L. Watson, P. Bayer, H.-T. Yang, O. Kiss, S. Nilsson, R. Sarkar, *arXi:2202.05594* **2022**.
[101] H. Chen, S. M. Lundberg, S.-I. Lee, *Nat. Commun.* **2022**, *13*, 4512.
[102] C. Frye, D. de Mijolla, T. Begley, L. Cowton, M. Stanley, I. Feige, *arXiv:2006.01272* **2021**.
[103] I. Covert, S.-I. Lee, *arXiv:2012.01536* **2021**.
[104] H. Yuan, H. Yu, J. Wang, K. Li, S. Ji, *Proceedings of Machine Learning Research* **2021**, *139*, 12241.
[105] S. Lundberg, *Github* **2018**.