Recurrent Quantum Neural Networks

Johannes Bausch

Abstract

Recurrent neural networks are the foundation of many sequence-to-sequence models in machine learning, such as machine translation and speech synthesis. With applied quantum computing in its infancy, there already exist quantum machine learning models such as variational quantum eigensolvers which have been used e.g. in the context of energy minimization tasks. Yet, to date, no viable recurrent quantum network has been proposed.

In this work we construct the first quantum recurrent neural network (QRNN) with demonstrable performance on non-trivial tasks such as sequence learning and integer digit classification. The QRNN cell is built from parametrized quantum neurons, which, in conjunction with amplitude amplification, creates a nonlinear activation of polynomials of its inputs and cell state, and allows the extraction of a probability distribution over predicted classes at each step.

To study the model's performance, we provide an implementation in pytorch, which allows the relatively efficient optimization of parametrized quantum circuits with tens of thousands of parameters, and which demonstrates that the model does not appear to suffer from the vanishing gradient problem that plagues many existing quantum classifiers and classical RNNs. We establish a QRNN training setup by benchmarking optimization hyperparameters, and analyse suitable network topologies for simple memorisation and sequence prediction tasks from Elman's seminal paper (1990). We then proceed to evaluate the QRNN on MNIST classification, by feeding the QRNN each image pixel-by-pixel; with a network utilizing only 12 qubits we reach a test set accuracy over 95% when discriminating between the digits '0' and '1'.

1 Introduction

Optimizing recurrent neural networks for long sequences is a challenging task: applying the same RNN cell operator iteratively often suffers from the well-studied vanishing or exploding gradients problem, which results in poor training performance [PMB12]. While long short-term memories or gated recurrent units (LSTMs and GRUs) with their linear operations acting on the cell state have been proposed as a way of circumventing this problem, they too are typically limited to capturing about 200 tokens of context, and e.g. start to ignore word order with increasing sequence lengths (more than 50 tokens away) [ZQH15; Dab08; Kha+18].

This failure of existing recurrent models to capture very long sequences surely played a role in the advent of alternative, non-recurrent models applicable to sequence-to-sequence tasks, such as transformer-type architectures with self-attention. Yet while these alternatives often outperformg LSTMs, they feature a fixed-width context window that does not easily scale with the sequence length; extensions thereof are an active field of research [Al-+19; Dai+19; KKL20].

Beyond training modifications such as truncated backpropagation [AFF19] which attempt to mitigate the vanishing gradient problem for recurrent neural networks directly, there have been numerous proposals to parametrize recurrent models in a way which limits or eliminates gradient decay, by ensuring the transfer operation at each step preserves the gradient norm; examples include orthogonal

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

[Vor+17; CM19] or unitary recurrent neural networks [ASB15; Wis+16; Jin+17]. Yet how to pick an parametrization for the RNN cell that is easy to compute, allows efficient training, and performs well on real-world tasks? This is a challenging question [HR16].

In this work, we propose a recurrent neural network model which is motivated by the emergent field of quantum computation. The interactions of any quantum system can be described by a Hermitian operator $\mathbf{H}$, which, as a solution to the Schrödinger equation, creates the system's time evolution under the unitary map $\mathbf{U} = \exp(-it\mathbf{H})$ [Sch26]. The axioms of quantum mechanics thus dictate that any quantum algorithm made up of a sequence of individual unitary quantum gates of the form of $\mathbf{U}$, is intrinsically unitary. This means that a parametrized quantum circuit serves a prime candidate for a unitary recurrent network.

Such parametrized quantum circuits have already found their way into other realms of quantum machine learning, one prominent example being variational quantum eigensolvers (VQE), which can serve as a variational ansatz for a quantum state, much akin to how a feed-forward network with a final softmax layer serves as parametrization for a probability distribution [WHB19; McC+16; Per+14; Jia+18]. VQEs have been deployed successfully e.g. in the context of minimizing energy eigenvalue problems within condensed matter physics [Cad+19], or as generative adversarial networks to load classical probability distributions into a quantum computer [ZLW19].

To date, classical recurrent models that utilize quantum circuits as sub-routines (i.e. where no quantum information is propagated) [GI19], early work on analysing Hopfield networks on quantum states [All+], or running classical Hopfield networks by a quantum-accelerated matrix inversion method [Reb+18] have been proposed; yet neither of them have the features we seek: a concrete quantum recurrent neural network with a unitary cell that allows to side-step the problem of gradient decay, and can ideally be implemented and trained on current classical hardware—and potentially on emerging quantum devices in the short-to-mid term.

In this work we construct such a quantum recurrent neural network (QRNN), which features demonstrable performance on real-world tasks such as sequence learning and handwriting recognition. Its RNN cell utilizes a highly-structured parametrized quantum circuit that deviates significantly from circuits used in the VQE setting. Its fundamental building block is a novel type of quantum neuron to introduce a nonlinearity, and in conjunction with a type of fixed-point amplitude amplification, allows the introduction of measurements (which are projectors, and not unitary operations) such that the overall evolution nonetheless remains arbitrarily close to unitary.

With an implementation in pytorch, we repeat several of the learning tasks first proposed in Elman's seminal paper "Finding Structure in Time" [Elm90], which we utilize to assess suitable model topologies such as the size of the cell state and structure of the parametrized RNN cell, and for benchmarking training hyperparameters for well-established optimizers such as Adam, RMSProp and SGD. As a next step, we evaluate the QRNN on MNIST classification, and find that feeding images pixel-by-pixel allows a discrimination of pairs of digits with up to 98.6% accuracy. Using modern data augmentation techniques the QRNN further achieves a test set performance on all digits of $\approx 95\%$. In order to demonstrate that the model indeed captures the temporal structure present within the MNIST images, we use the QRNN as a generative model, successfully recreating handwritten digits from an input of "0" or "1". As a final experiment, we assess whether the gradient quality decays for long input sequences. In a task of recognizing base pairs in a DNA string, we found that even for sequences of 500 bases training performance remains unaffected.

Although its performance is yet to compete with state-of-the-art scores e.g. on the MNIST dataset, our proposal is the first quantum machine learning model capable of working with training data as high-dimensional as images of integer digits, and it is the first variational quantum algorithm capable of being trained with thousands of parameters.

2 Recurrent Quantum Neural Networks

2.1 A Primer in Quantum Computation

A quantum system on n qubits lives on the n-fold tensor product Hilbert space $\mathcal{H} = (\mathbb{C}^2)^{\otimes d}$ with resulting dimension 2^d. A quantum state is a unit vector $\psi \in \mathcal{H}$, which in the context of quantum computation is commonly denoted in bra-ket notation $|\psi\rangle \in \mathcal{H}$; its conjugate transpose with $\langle\psi| =$

$|\psi\rangle^\dagger$; then the inner product $\langle\psi|\psi\rangle = \|\psi\|_2^2$ denotes the square of the 2-norm of ψ. $|\psi\rangle\langle\psi|$ then denotes the outer product, resulting in a rank 2 tensor. Computational basis states are given by $|0\rangle = (1,0)$, $|1\rangle = (0,1)$, and composite basis states are defined by e.g. $|01\rangle = |0\rangle \otimes |1\rangle = (0,1,0,0)$.

A quantum gate is then a unitary operation $\mathbf{U}$ on $\mathcal{H}$; if the operation acts non-trivially only on a subset $S \subseteq [n]$ of qubits, then $\mathbf{U} \in \mathrm{SU}(2^{|S|})$; to act on $\mathcal{H}$ we extend $\mathbf{U}$ to act as identity on the rest of the space, i.e. $\mathbf{U}_S \otimes \mathbb{1}_{[n]\setminus S}$. This trivial extension is generally omitted, and one denotes where the gate acts in a quantum circuit as e.g. fig. 1: the first gate $\mathbf{R}(\theta)$ denotes a single-qubit unitary acting on the second qubit from the bottom, and depending on the parameter θ. The line with the dot extending from the gate denotes a "controlled" operation, which if the control e.g. only acts on a single qubit itself denotes the block-diagonal unitary map $|0\rangle\langle 0| \otimes \mathbb{1} + |1\rangle\langle 1| \otimes \mathbf{R}(\theta) = \mathbb{1} \oplus \mathbf{R}(\theta)$; it means "if the control qubit is in state $|1\rangle$ apply $\mathbf{R}(\theta)$". This statement on basis states extends linearly to $\mathcal{H}$. Sequences of gates are calculated as matrix products, and circuits such as fig. 1 are read left-to-right. As shown in fig. 2, depending on the degree of the employed neurons the number of parameters can be tuned to grow linearly up to exponentially in the number of qubits used in the hidden state; and as seen in fig. 3 also linearly in the number of stages employed.

Single-qubit projective measurements (the only kind we need) are given by a hermitian 2×2 matrix $\mathbf{P}$, such as $\mathbf{M}\,|1\rangle\langle 1| = \mathrm{diag}(0,1)$; the complementary outcome is then $\mathbf{M}^\perp = \mathbb{1} - \mathbf{M}$. In the circuit they are denoted by meters. Given a quantum state $|\psi\rangle$, the post-measurement state is $\mathbf{M}\,|\psi\rangle\,/p$ with probability $p = \|\mathbf{M}\,|\psi\rangle\,\|_2$. This is also the post-selection probability to guarantee a measurement outcome $\mathbf{M}$; this probability can be amplified close to 1 using $\sim \sqrt{1/p}$ rounds of amplitude amplification ([Gro05], and see suppl. mat. for a more extensive discussion).

Statements such as "entangling gate" are non-essential to understand the workings of the recurrent model: all quantum recurrent neural networks within this proposal are executable on classical hardware where the "hidden state" on n qubits is represented by an array of size 2^n, and the set of parameters is given by the collection of all parameterized quantum gates used throughout, resulting in matrices with parametrized entries. To execute a QRNN classically, we use a series of matrix-vector multiplications for gates, and matrix-vector multiplications with successive renormalization of the state to have norm 1 for measurement and postselection operations. When executed on quantum hardware the matrix multiplications come "for free", and the hidden state on n qubits which takes exponential memory classically can be stored on $\sim n$ qubits alone; we discuss the incurred caveats more in due course (see also suppl. mat., section 3).

2.2 Parametrized Quantum Gates

Typical VQE quantum circuits are very dense, in the sense of alternating parametrized single-qubit gates with entangling gates such as controlled-not operations. This has the advantage of compressing a lot of parameters into a relatively compact circuit. On the other hand, while it is known that such circuits form a universal family, their high density of entangling gates and lack of correlation between parameters results in highly over-parametrized models that are hard to train on classification tasks for inputs larger than a few bits [Ben+19].

In this work, we construct a highly-structured parametrized quantum circuit where few parameters are re-utilized over and over. It is built mainly from a novel type of quantum neuron which rotates its target lane according to a non-linear activation function applied to polynomials of its binary inputs (eqs. (2) and (3); figs. 1, 2 and 5 in section 2.3). These neurons are combined in section 2.4 to form a structured RNN cell, as shown in fig. 3. The cell is a combination of an input stage that, at each step, writes the current input into the cell state. This is followed by multiple work stages that compute with input and cell state, and a final output stage that creates a probability density over possible predictions. Applying these QRNN cells iteratively on the input sequence as shown in fig. 3 results in a recurrent model much like traditional RNNs.

During training we perform quantum amplitude amplification (see [Gue19]) on the output lanes, to ensure that the we measure the correct token from the training data at each step. While measurements are generally non-unitary operations, the amplitude amplification step ensures that the measurements during training are as close to unitary as we wish.

While the resulting circuits are comparatively deep as compared to a traditional VQE circuit, they require only as many qubits as the input and cell states are wide.

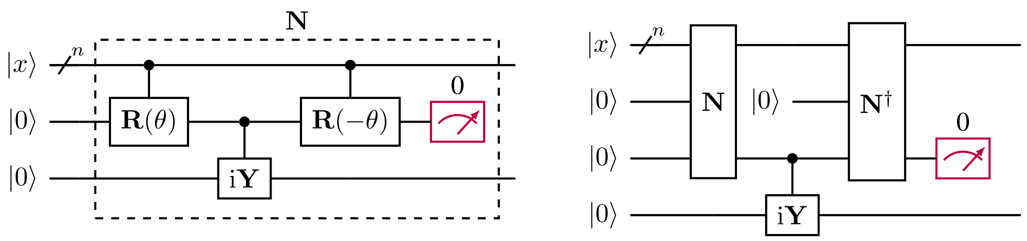


Figure 1: Quantum neuron by [CGA17]; left a first order neuron, right a second order application. Recursive iteration yields higher-order activation functions, respectively. The purple meter indicates a postselection (using fixed-point amplitude amplification) as described in [Tac+19].

2.3 A Higher-Degree Quantum Neuron

The strength of classical neural networks emerges through the application of nonlinear activation functions to the affine transformations on the layers of the network. In contrast, due to the nature of quantum mechanics, any quantum circuit will necessarily be a linear operation.

However, this does not mean that no nonlinear behaviour occurs anywhere within quantum mechanics: a simple example is a single-qubit gate $\mathbf{R}(\theta) := \exp(\mathrm{i}\mathbf{Y}\theta)$ for the Pauli matrix $\mathbf{Y}$ [NC10, eq. 4.1.1], which acts like

$$\mathbf{R}(\theta) = \exp\left(\mathrm{i}\theta\begin{pmatrix}0 & -\mathrm{i}\\ \mathrm{i} & 0\end{pmatrix}\right) = \begin{pmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{pmatrix},$$

i.e. as a rotation within the two-dimensional space spanned by the computational basis vectors of a single qubit, $\{|0\rangle, |1\rangle\}$.[1] While the rotation matrix itself is clearly linear, we note that the amplitudes of the state—$\cos\theta$ and $\sin\theta$—depend non-linearly on the angle θ. If we raise the rotation to a controlled operation $\mathbf{cR}(i, \theta_i)$ conditioned on the i^{th} qubit of a state $|x\rangle$ for $x \in \{0,1\}^n$, one can derive the map

$$\mathbf{R}(\theta_0)\mathbf{cR}(1,\theta_1)\cdots\mathbf{cR}(n,\theta_n)\,|x\rangle\,|0\rangle = |x\rangle\,\big(\cos(\eta)\,|0\rangle + \sin(\eta)\,|1\rangle\,\big) \quad \text{for } \eta = \theta_0 + \sum_{i=1}^{n}\theta_i x_i. \tag{1}$$

This corresponds to a rotation by an affine transformation of the basis vector $|x\rangle$ with $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, by a parameter vector $\theta = (\theta_0, \theta_1, \ldots, \theta_n)$. This operation extends linearly to superpositions of basis and target states; and due to the form of $\mathbf{R}(\theta)$ all newly-introduced amplitude changes are real-valued.[2]

This cosine transformation of the amplitudes by a controlled operation is already non-linear; yet a sin function is not particularly steep, and also lacks a sufficiently "flat" region within which the activation remains constant, as present e.g. in a rectified linear unit. Cao et al. [CGA17] proposed a method for implementing a linear map on a set of qubits which yields amplitudes that feature such steeper slopes and plateaus, much like a sigmoidal activation function. The activation features an order parameter $\mathrm{ord} \geq 1$ that controlls the steepness, as shown in fig. 5; the circuit which gives rise to such an activation amplitude is shown in fig. 1. On pure states this quantum neuron gives rise to a rotation by an angle $f(\theta) = \arctan(\tan(\theta)^{2^{\mathrm{ord}}})$, where $\mathrm{ord} \geq 1$ is the order of the neuron. Starting from an affine transformation η for the input bitstring x_i as given in eq. (1), this rotation translates to the amplitudes

$$\cos(f(\eta)) = \frac{1}{\sqrt{1+\tan(\eta)^{2\times 2^{\mathrm{ord}}}}} \quad\text{and}\quad \sin(f(\eta)) = \frac{\tan(\eta)^{2^{\mathrm{ord}}}}{\sqrt{1+\tan(\eta)^{2\times 2^{\mathrm{ord}}}}}, \tag{2}$$

emerging from normalising the transformation $|0\rangle \longmapsto \cos(\theta)^{2^{\mathrm{ord}}}\,|0\rangle + \sin(\theta)^{2^{\mathrm{ord}}}\,|1\rangle$ as can be easily verified. For $\mathrm{ord} = 1$, the circuit is shown on the left in fig. 1; for $\mathrm{ord} = 2$ on the right. Higher orders can be constructed recursively.

[1]Signs and factors in the exponent are convention; in [NC10, eq. 4.5] the form given is $\exp(-\mathrm{i}\mathbf{Y}\theta/2)$.
[2]Complex amplitudes are not necessary for the power of quantum computing.

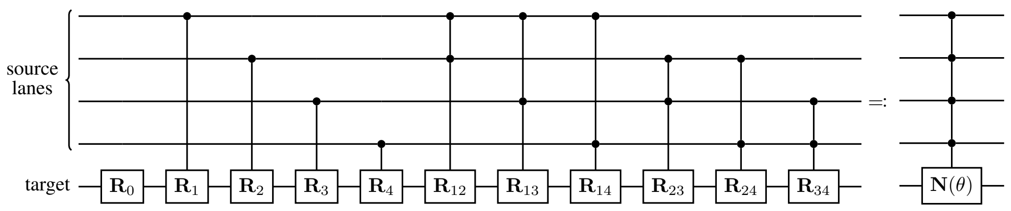


Figure 2: Degree $d = 2$ controlled rotation for quantum neuron shown in fig. 1, on $n = 4$ input neurons; the controlled rotations are $\mathbf{R}_I := \mathbf{R}(\theta_I)$ for $I \subset [n]$ with $|I| \le d$. The quantum neuron thus carries a parameter vector $\theta \in \mathbb{R}^D$ for $D = \sum_{i=0}^{d} \binom{n}{i}$.

This quantum neuron is a so-called repeat-until-success (RUS) circuit, meaning that the ancilla that is measured (purple meter in fig. 1) indicates whether the circuit has been applied successfully. When the outcome is zero, the neuron has been applied. When the outcome is one, a (short) correction circuit reverts the state to its initial configuration. Started from a pure state (e.g. $|x\rangle$ for $x \in \{0,1\}^n$, as above) and repeating whenever a 1 is measured, one obtains an arbitrarily-high success probability.

Unfortunately this does not work for a control in superposition, such as a state $(|x\rangle + |y\rangle)/\sqrt{2}$, for $x \ne y$ two bit-strings of length n. In this case, the amplitudes within the superposition will depend on the history of success. Using a technique called fixed-point oblivious amplitude amplification [Gue19; Tac+19], one can alleviate this issue, and essentially post-select on measuring outcome 0 while preserving unitarity of the operation to arbitaily high accuracy. This comes at the cost of performing multiple rounds of these quantum circuits (and their inverses), the number of which will depend on the likelihood of measuring a zero (i.e. success) in first place. This naturally depends on the parameters of the neuron, θ, and the input state given. We emphasise that by choosing the individual postselection probabilities large enough, the overall likelihood of succeeding does *not* shrink exponentially in the number of quantum neurons used. In the following we will thus simply assume that the approximate postselection is possible, and carefully *monitor* the overhead due to the necessary amplitude amplification in our empirical studies in section 4; we found that in general the postselection overhead remained mild, and tended to converge to zero as learning progressed. An analytical estimate on how this postselection probability shrinks is given in the suppl. mat..

The specific activation function this quantum neuron gives rise to is depicted in fig. 5. We point out that other shapes of activation functions can readily be implemented in a similar fashion [de +19].

In this work, we generalize this quantum neuron by increasing the number of control terms. More concretely, η as given in eq. (1) is an affine transformation of the boolean vector $x = (x_1, \ldots, x_n)$ for $x_i \in \{0,1\}$. By including multi-control gates—with their own parametrized rotation, labelled by a multiindex θ_I depending on the qubits $i \in I$ that the gate is conditioned on—we obtain the possibility to include higher degree polynomials, namely

$$\eta' = \theta_0 + \sum_{i=1}^{n} \theta_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} \theta_{ij} x_i x_j + \ldots = \sum_{\substack{I \subseteq [n] \\ |I| \le d}} \theta_I \prod_{i \in I} x_i, \tag{3}$$

where d labels the degree of the neuron; for $d = 2$ and $n = 4$ an example of a controlled rotation that gives rise to this higher order transformation η' on the bit string x_i is shown in fig. 2. In this fashion, higher degree boolean logic operations can be directly encoded within a single conditional rotation: an AND operation between two bits x_1 and x_2 is simply $x_1 x_2$.

2.4 QRNN Cell

The quantum neuron defined in section 2.3 will be the crucial ingredient in the construction of our quantum recurrent neural network cell. Much like for classical RNNs and LSTMs, we define such a cell which will be applied successively to the input presented to the network. More specifically, the cell is comprised of in- and output lanes that are reset after each step, as well as an internal cell state which is passed on to the next iteration of the network.

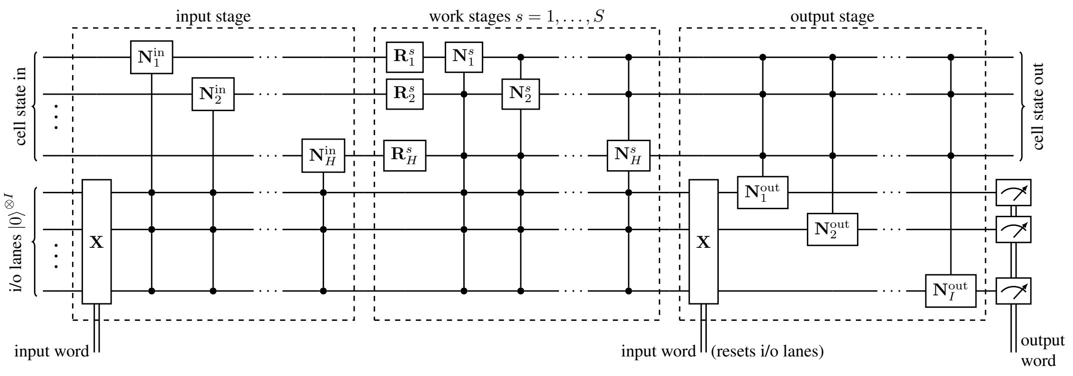

input stage
work stages $s = 1, \dots, S$
output stage
cell state in
i/o lanes $|0\rangle^{\otimes I}$
cell state out
$\mathbf{N}_1^{\text{in}}$
$\mathbf{N}_2^{\text{in}}$
$\mathbf{N}_H^{\text{in}}$
$\mathbf{R}_1^s$
$\mathbf{R}_2^s$
$\mathbf{R}_H^s$
$\mathbf{N}_1^s$
$\mathbf{N}_2^s$
$\mathbf{N}_H^s$
$\mathbf{N}_1^{\text{out}}$
$\mathbf{N}_2^{\text{out}}$
$\mathbf{N}_I^{\text{out}}$
X
X
input word
input word
(resets i/o lanes)
output word

Figure 3: Quantum recurrent neural network cell. Each controlled quantum neuron $\mathbf{cN}_j^i$ is implemented as explained in section 2.3 and fig. 1 and comes with its own parameter vector θ_j^i, where we draw the control lanes from the rotation inputs as depicted in fig. 2, with ancillas omitted for clarity. The $\mathbf{R}_j^i$ are extra rotations with a separate parameter set ϕ_j^i.

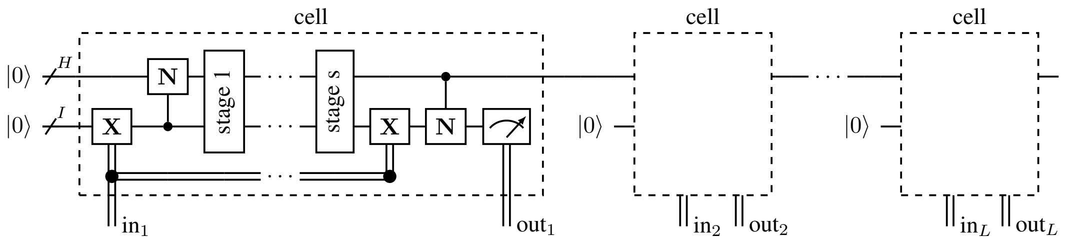


Figure 4: Quantum recurrent neural network, by applying the same QRNN cell constructed in section 2.4 iteratively to a sequence of input words $\text{in}_1, \dots, \text{in}_L$. All input and ancilla qubits used throughout can be reused; we thus need $H + I + \text{ord}$ qubits, where H is the cell state workspace size, I the input token width (in bits), and ord the order of the quantum neuron activation, as explained in section 2.3.

2.5 Sequence to Sequence Model

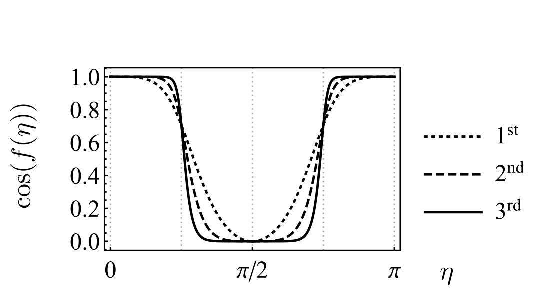
In order to be able to apply the QRNN cell constructed in section 2.4, we need to iteratively apply it to a sequence of input words $\text{in}_1, \text{in}_2, \dots, \text{in}_L$. This is achieved as depicted in fig. 4.

The output lanes out_i label a measured discrete distribution p_i over the class labels (which we can do by reading out the statevector weights if running a simulation on a classical computer; or by repeated measurements on quantum hardware). This distribution can then be fed into an associated loss function such as cross entropy or CTC loss.

3 Implementation and Training

We refer the reader to the suppl. mat. for a brief summary on how the overall procedure is assembled from the depicted quantum circuits in figs. 1 to 4. We implemented the QRNN in pytorch, using custom quantum gate layers and operations that allow us to extract the predicted distributions at each step. As this is in essence a simulation of a quantum computation, we take the following shortcuts: instead of truly performing fixed-point amplitude amplification for the quantum neurons and output lanes during training, we postselect; as aforementioned, we kept track of the postselection probabilities which allows us to compute the overhead that would be necessary for amplitude amplification. We further extract the output probability distribution instead of estimating it at every step using measurements.

In our experiments we focus on character-level RNNs. Just like in the classical case, the sequence of predicted distributions $\{p_i\}$ is fed into a standard `nn.CrossEntropyLoss` to minimize the distance



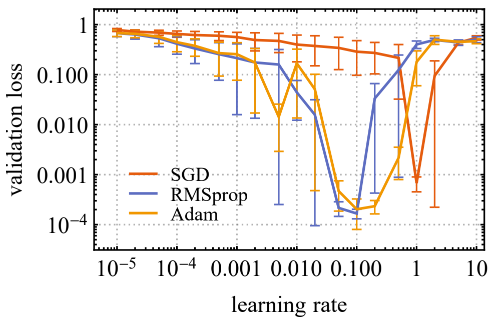


Figure 5: Left: Quantum neuron amplitude $\cos(f(\eta))$, as given in eq. (2). Shown are the first to third order activations $o = 1, 2, 3$. Right: training performance of SGD, RMSprop and Adam optimizers over a range of learning rates, as described in section 4.1.

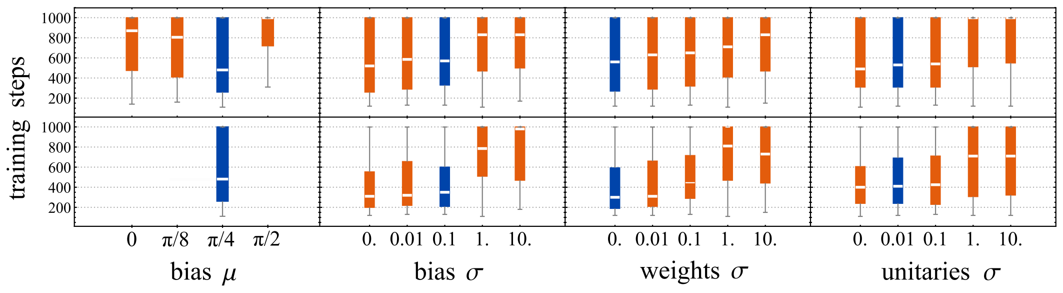


Figure 6: Box whisker plot for parameter initialization of QRNN cell as described in section 4.2. In the second row we chose only the runs with bias $\mu = \pi/4$. Marked in blue is the chosen defaults.

to a target sequence. With pytorch's autograd framework we are able to perform gradient-based learning directly; on quantum hardware either gradient-free optimizers such as L-BFGS, NatGrad would have to be utilized, or numerical gradients extracted [WGK20]. All experiments were executed on 2-8 CPUs, and required between 500MB and 35GB of memory per core.

4 Empirical Results

4.1 Sequence Memorization

The first task we implement is whether the network can learn to reproduce the two sequences `44444...4` and `12312...3`. For a QRNN with 5 stages and a workspace size of 5 (982 parameters) this poses no problem, and—as aforementioned—the postselection overhead during training is mild, and we present a plot in the supplementary material. With this setup, we benchmark optimizer and learning rate hyperparameters; our findings are summarized in fig. 5.

We found the L-BFGS optimizer commonly used with VQE circuits highly numerically unstable, resulting in many runs with `NaN`s; we thus excluded it from this experiment. SGD has a very narrow window of good learning rates; RMSprop and Adam are less sensitive to this choice, with Adam generally outperforming the former, which makes it our default choice for all following experiments.

4.2 Finding Structure in Time

In his 1990 paper, Elman describes two basic sequence learning tasks to evaluate structure in time [Elm90]. The first task is that of learning XOR sequences, which are binary strings $s = s_1 s_2 s_3 \ldots s_L$ such that each third digit is the XOR value of the preceding two, i.e. $s_{3i} = s_{3i-1} \oplus s_{3i-2}$; one example being $s =$ `000 011 110 011 101`.

Due to the simplicity of the test, we use a QRNN with workspace size 4 and a single work stage to explore which parameter initialization converges to a validation loss threshold of 10^{-3} first. As shown in fig. 3, there are two groups of parameters: those for the neurons, and those for the single-

Digit Set	Method	Data Augmentation	Accuracy $[\%]$
$\{0, 1\}$	QRNN (12 qubits)	none	98.6 ± 0.4
$\{3, 6\}$	VQE (17 qubits) [FN18]	ambiguous samples removed	98
	QRNN (12 qubits)	none	90.8 ± 0.7
$\{$even, odd$\}$	VQE (10 qubits) [Gra+19]†	none	82
full MNIST	uRNN [ASB15]	none	95.1
	LSTM [ASB15]	none	98.2
	EXPRNN [CM19]	none	98.7
	QFD ($>$ 200 qubits) [KL20]‡	PCA, slow feature analysis	98.5
	QRNN (10 qubits)	PCA, t-SNE	94.6 ± 0.4

Table 1: Classification of MNIST using QRNNs on 12 qubits (workspace size 8, 2 stages, neuron degree 2; 1212 parameters) with input presented pixel-by-pixel; resp. t-SNE augmented classification using 10 qubits (workspace size 6, 2 stages, neuron degree 3; 1292 parameters). †) Image data prepared in superposition, i.e. as a state $\propto \sum_i v_i |i\rangle$, where $v_i \in [0, 1]$ is the pixel value at position i in the image. This model is thus at most as powerful as a linear discriminator and always performs worse than logistic regression. ‡) The paper exploits a well-known quantum speedup in performing sparse linear algebra [HHL08], and extensively depends on quantum random access memory (QRAM), recently shown to allow "de-quantization" of claimed exponential quantum speedups [Tan19]. The authors do not explicitly state a qubit count, so it is lower-bounded from the number of qubits required for the QRAM alone.

qubit unitaries within the work stages. Each quantum neuron as depicted in fig. 2 and eq. (3) itself comprises two parameter sets: a bias gate $\mathbf{R}_0$ with angle θ_0, and the weights (all other parameters). Choosing to initialize each of them with a normal distribution with mean μ and width σ, we have four parameter group hyperparameters: bias μ, bias σ, weights σ (the mean is already captured in the bias gate), and unitaries σ (for which we chose the mean to be zero by default).

Our findings and choices for the default initialisation are collected in fig. 6. The most influential meta parameter is the bias $\mu = \pi/2$—which, as shown in fig. 5, places the initial polynomial η at the steepest slope of the activation function, which results in a large initial gradient.

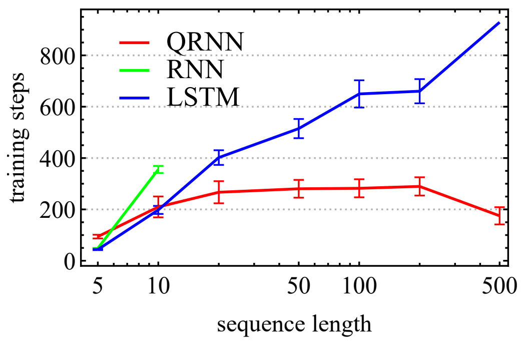
The second task described in [Elm90] is that of learning the structure of a sentence made up of the three words "ba", "dii" and "guuu"; an example being "ba dii ba guuu dii". Having seen the letter 'd', the network thus has to remember that the next two letters to predict are "ii", and so on. We chose this slightly more difficult task to assess how the QRNN topology influence convergence speed. We found that the neuron order $\mathrm{ord} = 2$ performs best, which results in an activation function with relatively steep flanks, but a not-too-small derivative around its extrema. The other parameters are discussed in the supplementary material.

4.3 MNIST Classification

To test whether we can utilize our QRNN setup to classify more complex examples, we assess its performance on integer digit classification, using the MNIST dataset ($55010 : 5000 : 10000$ train:validate:test split; images cropped to 20×20 pixels first, then downscaled to size 10×10). While this is a rather untypical task for a recurrent network, there exist baselines both for a comparison with a classical RNN, as well as with quantum classifiers.

We choose two "scanlines" across each image: one left-to-right, top-to-bottom; the other one top-to-bottom, left-to-right. This means that two bits of data are presented at each step. The output labels are then simply binary values of the numbers 0 to 9, which have to be written to the output at the last steps of the sequence (in little Endian order). We found that pairs of digits such as '0' and '1' (arguably the easiest ones) could be discriminated with $> 98\%$ success probability; but even more complicated pairs like '3' and '6' could be distinguished with $> 90\%$ likelihood.

In addition to classifying digits by presenting images pixel-by-pixel, we also used 2D and 3D t-SNE clustering as data augmentation in a first step. The RNN was then presented with each coordinate,



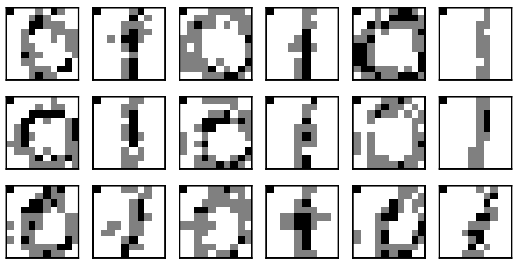


Figure 7: Left: Expected number of training steps to reach a validation loss 0.5×10^{-4}. Shown are QRNN (red; 2 work stages, cell state size 6, 1965 parameters), RNN (blue; 2 layers, hidden layer size 80, and a final linear layer; 20808 parameters) and LSTM (green; 2 layers, hidden layer size 40, and a final linear layer; 21448 parameters) for DNA sequence recognition task described in section 4.5. Right: QRNN-generated handwritten digits '0' and '1', as explained in section 4.4. The QRNN topology is the same as for MNIST classification, with 1212 free parameters (section 4.3).

discretized to 2, 5 and 8 bits of precision. While the preprocessing step drastically improved classification performance, it still falls far short from state-of-the-art scores on MNIST. We summarize our findings and a comparison with existing literature in table 1.

4.4 QRNNs as Generative Models

Instead of classifying digits, QRNNs can also be used as generative models. With a 0 or 1 presented to the QRNN at the first step, we train it to generate handwritten digits. As shown in fig. 7, the network indeed learns the global structure of the two digits.

4.5 Long Sequence Tests

To assess our claims of high-quality gradients even in a situation of long sequences, we set up a test set consisting of gene sequences made up of the bases `GATC`; a single `U` is then inserted at a random position within the first half of the string, and the task of the network is to identify the base following after the `U`, similar to the arguably more challenging "copying memory" task described in [CM19]. For instance, the label to identify for the sequence `'AGAUATTCAGAAT'` is `'A'`. We repeat this classification task for multiple sequence lengths and several initial seeds, and stop after the validation loss is below a threshold of 5×10^{-4}.

Using the number of steps required to reach the threshold as a metric of success, we found that a QRNN with a workspace size of six, two work stages, and activation degree 3 (resulting in 1956 parameters) retains trainability even in a situation of string lengths of 500 bases. In contrast, as shown in fig. 7, an RNN (2 layers, hidden size 80; 20808 parameters) and LSTM (2 layers, hidden size 40; 21448 parameters) show a clear increase in convergence time, or fail to converge within the set limit of 16000 steps.

5 Conclusion

Many more benchmarks and baselines could be assessed and established; for instance the aforementioned copy task described in [CM19], addition, or the Penn Treebank. With our selection we strived to strike a balance between low-level and high-level tasks that fit within the memory constraints imposed by an intrinsically quantum model. While many existing proposals for recurrent models with unitary or orthogonal constraints do not parametrize the entire Stiefel manifold, this is not the case for our model. To see this we observe that in fig. 3 the parametrized single-qubit rotations are interlaced with entangling gates (the neurons). It is straightforward to see that the work stages alone already form a strict superset of existing VQE models. The latter are proven to be universal for quantum computation [McC+16], and hence densely fill the entire Hilbert space $\mathrm{SU}(2^n)$ in the limit of taking deeper and deeper circuits.

6 Broader Impact

Without doubt, existing recurrent models—even simple RNNs—outclass the proposed QRNN architecture in this paper in real-world learning tasks. In part, this is because we cannot easily simulate a large number of qubits on classical hardware: the memory requirements necessarily grow exponentially in the size of the workspace, for instance, which limits the number of parameters we can introduce in our model—on a quantum computer this overhead would vanish, resulting in a linear execution time in the circuit depth.

What should nevertheless come as a surprise is that the model *does* perform relatively well on non-trivial tasks such as the ones presented here, in particular given the small number of qubits (usually between 8 and 12) that we utilised. As qubit counts in real-world devices are severely limited—and likely will be for the foreseeable future—learning algorithms with tame system requirements will certainly hold an advantage.

Moreover, while we motivate the topology of the presented QRNN cell given in fig. 3 by the action of its different stages (writing the input; work; writing the output), and while the resulting circuits are far more structured than existing VQE setups, our architecture is still simplistic as compared to the various components of an RNN, let alone an LSTM. In all likelihood, a more specialized circuit structure (such as going from an RNN to an LSTM) will outperform the "simple" quantum recurrent network presented herein.

Beyond the exploratory aspect of our work, our main insights are twofold. On the classical side—as discussed in the introduction—we present an architecture which *can* run on current hardware and ML implementations such as pytorch; and which is a candidate parametrization for unitary recurrent models that hold promise in circumventing gradient degradation for very long sequence lengths. On the quantum side, we significantly advance the field of variational circuits for quantum machine learning tasks; allowing ingestion of data of more than a few bits of size; demonstrate that models with large parameter counts can indeed be evaluated and trained; and that classical baselines such as MNIST classification are, indeed, within reach when using a more sophisticated model.

Finally, our work is the first recurrent and entirely quantum neural network presented to date. Variants of it might find application in conjunction with other quantum machine learning algorithms, such as quantum beam search [BSP19] in the context of language modelling. With a more near-term focus in mind, modelling the evolution of quantum systems with noisy dynamics is a task currently addressed using classical recurrent models [Flu+20]. Due to the intrinsic capability of a QRNN to keep track of a quantum state it holds promise to better capture the exponentially-growing phase space dimension of the system to be modelled.

Acknowledgements

J. B. acknowledges support of the Draper's Research Fellowship at Pembroke College, and wishes to thank Stephanie Hyland and Jean Maillard for helpful discussions and suggestions.

References

[PMB12] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "Understanding the exploding gradient problem". In: *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*. ICML'13. JMLR.org, Nov. 2012, pp. III–1310–III–1318. arXiv: 1211.5063.

[ZQH15] Chenxi Zhu, Xipeng Qiu, and Xuanjing Huang. "Transition-based dependency parsing with long distance collocations". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2015.

[Dab08] Ewa Dabrowska. "Questions with long-distance dependencies: A usage-based perspective". In: *Cognitive Linguistics* 19.3 (Jan. 2008).

[Kha+18] Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. "Sharp Nearby, Fuzzy Far Away: How Neural Language Models Use Context". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long*

Papers). Stroudsburg, PA, USA: Association for Computational Linguistics, 2018, pp. 284–294.

[Al-+19] Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. "Character-Level Language Modeling with Deeper Self-Attention". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (July 2019), pp. 3159–3166.

[Dai+19] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context". In: (Jan. 2019). arXiv: 1901.02860.

[KKL20] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. "Reformer: The Efficient Transformer". In: *International Conference on Learning Representations*. 2020.

[AFF19] Christopher Aicher, Nicholas J Foti, and Emily B Fox. "Adaptively Truncating Backpropagation Through Time to Control Gradient Bias". In: *UAI*. May 2019. arXiv: 1905.07473.

[Vor+17] Eugene Vorontsov, Chiheb Trabelsi, Samuel Kadoury, and Chris Pal. "On orthogonality and learning recurrent networks with long term dependencies". In: *ICML*. Jan. 2017. arXiv: 1702.00071.

[CM19] Mario Lezcano Casado and David Martínez-Rubio. "Cheap Orthogonal Constraints in Neural Networks: A Simple Parametrization of the Orthogonal and Unitary Group". In: *ICML*. 2019.

[ASB15] Martin Arjovsky, Amar Shah, and Yoshua Bengio. "Unitary Evolution Recurrent Neural Networks". In: (Nov. 2015). arXiv: 1511.06464.

[Wis+16] Scott Wisdom, Thomas Powers, John R Hershey, Jonathan Le Roux, and Les Atlas. "Full-Capacity Unitary Recurrent Neural Networks". In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., Oct. 2016, pp. 4887–4895. arXiv: 1611.00035.

[Jin+17] Li Jing, Caglar Gulcehre, John Peurifoy, Yichen Shen, Max Tegmark, Marin Soljačić, and Yoshua Bengio. "Gated Orthogonal Recurrent Units: On Learning to Forget". In: *Neural Computation* 31.4 (June 2017), pp. 765–783. arXiv: 1706.02761.

[HR16] Stephanie L Hyland and Gunnar Rätsch. "Learning Unitary Operators with Help From u(n)". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI'17. AAAI Press, July 2016, pp. 2050–2058. arXiv: 1607.04903.

[Sch26] E. Schrödinger. "An Undulatory Theory of the Mechanics of Atoms and Molecules". In: *Physical Review* 28.6 (Dec. 1926), pp. 1049–1070.

[WHB19] Daochen Wang, Oscar Higgott, and Stephen Brierley. "Accelerated Variational Quantum Eigensolver". In: *Physical Review Letters* 122.14 (Apr. 2019), p. 140504.

[McC+16] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. "The theory of variational hybrid quantum-classical algorithms". In: *New Journal of Physics* 18.2 (Feb. 2016), p. 23023.

[Per+14] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. "A variational eigenvalue solver on a photonic quantum processor". In: *Nature Communications* 5.1 (Sept. 2014), p. 4213. arXiv: 1304.3061.

[Jia+18] Zhang Jiang, Jarrod McClean, Ryan Babbush, and Hartmut Neven. "Majorana loop stabilizer codes for error correction of fermionic quantum simulations". In: (Dec. 2018). arXiv: 1812.08190.

[Cad+19] Chris Cade, Lana Mineh, Ashley Montanaro, and Stasja Stanisic. "Strategies for solving the Fermi-Hubbard model on near-term quantum computers". In: (Dec. 2019). arXiv: 1912.06007.

[ZLW19] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. "Quantum Generative Adversarial Networks for learning and loading random distributions". In: *npj Quantum Information* (2019). arXiv: 1904.00043.

[GI19] Laszlo Gyongyosi and Sandor Imre. "Training Optimization for Gate-Model Quantum Neural Networks". In: *Scientific Reports* 9.1 (Dec. 2019), p. 12679.

[All+] R. Allauddin, K. Gaddam, E.C. Behrman, J.E. Steck, and S.R. Skinner. "Advantages of quantum recurrent networks: an examination of stable states". In: *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*. IEEE, pp. 2732–2737.

[Reb+18] Patrick Rebentrost, Thomas R. Bromley, Christian Weedbrook, and Seth Lloyd. "Quantum Hopfield neural network". In: *Physical Review A* 98.4 (Oct. 2018), p. 042308.

[Elm90] J Elman. "Finding structure in time". In: *Cognitive Science* 14.2 (June 1990), pp. 179–211.

[Gro05] Lov K. Grover. "Fixed-Point Quantum Search". In: *Physical Review Letters* 95.15 (Oct. 2005), p. 150501.

[Ben+19] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. "Parameterized quantum circuits as machine learning models". In: (June 2019). arXiv: 1906.07682.

[Gue19] Gian Giacomo Guerreschi. "Repeat-until-success circuits with fixed-point oblivious amplitude amplification". In: *Physical Review A* 99.2 (Feb. 2019), p. 022306.

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge: Cambridge University Press, 2010, p. 676.

[CGA17] Yudong Cao, Gian Giacomo Guerreschi, and Alán Aspuru-Guzik. "Quantum Neuron: an elementary building block for machine learning on quantum computers". In: (Nov. 2017). arXiv: 1711.11240.

[Tac+19] Francesco Tacchino, Chiara Macchiavello, Dario Gerace, and Daniele Bajoni. "An artificial neuron implemented on an actual quantum processor". In: *npj Quantum Information* 5.1 (Dec. 2019), p. 26.

[de +19] Fernando M. de Paula Neto, Teresa B. Ludermir, Wilson R. de Oliveira, and Adenilton J. da Silva. "Implementing Any Nonlinear Quantum Neuron". In: *IEEE Transactions on Neural Networks and Learning Systems* (2019), pp. 1–6.

[WGK20] David Wierichs, Christian Gogolin, and Michael Kastoryano. "Avoiding local minima in variational quantum eigensolvers with the natural gradient optimizer". In: (Apr. 2020). arXiv: 2004.14666.

[FN18] Edward Farhi and Hartmut Neven. "Classification with Quantum Neural Networks on Near Term Processors". In: (Feb. 2018). arXiv: 1802.06002.

[Gra+19] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti. "An initialization strategy for addressing barren plateaus in parametrized quantum circuits". In: (Mar. 2019). arXiv: 1903.05076.

[KL20] Iordanis Kerenidis and Alessandro Luongo. "Quantum classification of the MNIST dataset via Slow Feature Analysis". In: *Phys. Rev. A* (May 2020). arXiv: 1805.08837.

[HHL08] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum algorithm for solving linear systems of equations". In: *Physical Review Letters* 103.15 (Nov. 2008), p. 150502. arXiv: 0811.3171.

[Tan19] Ewin Tang. "A quantum-inspired classical algorithm for recommendation systems". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing - STOC 2019*. New York, New York, USA: ACM Press, 2019, pp. 217–228.

[BSP19] Johannes Bausch, Sathyawageeswar Subramanian, and Stephen Piddock. "A Quantum Search Decoder for Natural Language Processing". In: (Sept. 2019). arXiv: 1909.05023.

[Flu+20] E. Flurin, L. S. Martin, S. Hacohen-Gourgy, and I. Siddiqi. "Using a Recurrent Neural Network to Reconstruct Quantum Dynamics of a Superconducting Qubit from Physical Observations". In: *Physical Review X* 10.1 (Jan. 2020), p. 011006.