A hybrid machine-learning algorithm for designing quantum experiments

L. O'Driscoll,[1] R. Nichols,[1] and P. A. Knott[1,*]

[1] *Centre for the Mathematics and Theoretical Physics of Quantum Non-Equilibrium Systems (CQNE), School of Mathematical Sciences, University of Nottingham, University Park, Nottingham, NG7 2RD, UK.*

(Dated: December 11, 2018)

We introduce a hybrid machine-learning algorithm for designing quantum optics experiments that produce specific quantum states. Our algorithm successfully found experimental schemes to produce all 5 states we asked it to, including Schrödinger cat states and cubic phase states, all to a fidelity of over 96%. Here we specifically focus on designing realistic experiments, and hence all of the algorithm's designs only contain experimental elements that are available with current technology. The core of our algorithm is a genetic algorithm that searches for optimal arrangements of the experimental elements, but to speed up the initial search we incorporate a neural network that classifies quantum states. The latter is of independent interest, as it quickly learned to accurately classify quantum states given their photon-number distributions.

As artificial intelligence (AI) and machine learning develop, their range of applicability continues to grow. They are now being utilised in the fast-growing field of quantum machine learning [1–3], with one particular application demonstrating that AI is an effective tool for designing quantum physics experiments [4–8]. In this vein, here we introduce a hybrid algorithm that designs and optimises quantum optics experiments for producing a range of useful quantum states, including Schrödinger cat states [9–11] and cubic phase states [12].

The core of our algorithm, named Ada (or AdaQuantum [13]) and introduced in [14] and [4], uses a genetic algorithm to search for optimal arrangements of quantum optics experimental equipment. Any given arrangement will output a quantum state of light, and the algorithm's task is to optimise the arrangement to find states with specific properties. To assess the suitability of a given state, we require a *fitness function* that takes as input a quantum state and outputs a number – the *fitness value* – that quantifies whether the state has the properties we desire or not. Our previous works largely focused on quantum metrology, where our algorithm found quantum states with substantial improvements over the alternatives in the literature [4, 14]. While in [4, 14] the fitness function assessed the phase-measuring capabilities of the states, in this paper instead we look at producing a range of useful and interesting states (introduced below) to a high fidelity, and hence we use as our fitness function the fidelity to our target states.

The search space for our genetic algorithm is huge, which means that typically the algorithm has to simulate and evaluate a vast number of quantum optics experiments in order to finally find strong solutions. This introduces a major challenge in our work: the speed, efficiency, and effectiveness of our algorithm depend in a large part on simulating and evaluating the experiments as quickly and accurately as possible. If short-cuts could be found that allow a given experimental set-up to be evaluated approximately without the full simulation being performed, then this would greatly improve our optimisation: the approximate evaluations would provide a quick guide for the search to progress in the right direction, and subsequently the exact evaluation, using the full quantum simulation, would provide the precise fitness value, thus validating and fine-tuning the search. Efficient computational models of this sort for approximating the fitness function are often known as surrogates or meta-models [15].

In this paper we use one such approximation during the evaluation stage: instead of explicitly calculating the fidelity of our output state to the desired states, we use a deep neural network (DNN) that learns to classify what type of state has been outputted, and approximately how close this state is to the target state. Explicitly calculating the fidelity the large number of times required to run our algorithm takes a surprisingly long time, whereas using our DNN we get a useful, albeit modest, speed-up. While in the work presented here the speed-up is only small, our results can be seen as a proof of principle, paving the way for much more demanding fitness functions, such as the Bayesian mean squared error [14, 16], to be approximately evaluated in this way. Once the DNN has guided the search in the right direction, the fidelity is then used to provide the exact fitness function (see Section III A for the full description of our algorithm).

Our DNN for classifying quantum states is likely to be of independent interest, as it quickly learnt to recognise a quantum state just given its photon-number distribution. This has potential uses in a wide range of applications, such as quantum computing or quantum cryptography, where it can be valuable to quickly recognise what type of quantum state has been produced.

Our hybrid algorithm, utilising a genetic algorithm to perform the automated search and a neural network to classify the outputted quantum states, quickly and efficiently found new quantum optics experiments to produce a range of useful quantum states to a high fidelity (see Section II). This complements the recent work of Arrazola *et al.* [7] who used machine learning to also find quantum optics experiments to produce a range of

* Contact email address: Paul.Knott@nottingham.ac.uk

specific states, with one key difference being that our experiments are specifically designed to be practical, which comes at the cost of our states being of a lower fidelity than those in [7].

This paper is structured as follows. First we introduce a general experimental scheme for engineering optical quantum states, and introduce the various quantum-optics experimental elements that are typically used. We then introduce the main goal of our work – finding experimental set-ups to create specific quantum states – and present our results. We then describe the methods used to generate these results: firstly the genetic algorithm, which searches through different arrangements of the experimental equipment to find those that produce our desired states; and secondly our neural network, which classifies quantum states and enables a speed-up in our search algorithm.

I. QUANTUM STATE ENGINEERING WITH LIGHT

The state-engineering scheme we use is shown in Fig. 1. Firstly, two input states, $|\psi_{0a}\rangle$ and $|\psi_{0b}\rangle$, are input into the two modes. The two modes then pass through a sequence of operators $\hat{O}_i$ where $i = 1, .., m$, and the final step is to perform a heralding measurement on one mode, producing the final state $|\psi_f\rangle$. With appropriate choices of input states, operators, and measurements this scheme is able to replicate a wide range of the quantum state engineering protocols in the literature, such as [9–11, 17, 18]. The main difference between the present paper and the usual methods is that we don't *choose* the input states, operators, and measurements, but instead we use a genetic algorithm to *search* for states with the desired properties (see Section III A for details of the algorithm).

Our objective is to find *practical* experiments, and so we construct our state engineering protocols from elements of an experimentally-ready toolbox of quantum optics states, operators, and measurements, which is summarised in the table below.

Input States	**Apparatus**
Fock, $\|n\rangle$	Beam splitter, $\hat{U}_T$
Coherent, $\|\alpha\rangle$	Phase shift, $e^{i\hat{n}\theta}$
Squeezed, $\|z\rangle$	Displacement, $\hat{D}(\alpha)$
TMSV, $\|z\rangle_{12}$	Number measurement, $\|n\rangle\langle n\|$

Here we only introduce the most important details of the toolbox; more details can be found in Appendix A. Firstly, the input states we include are the single-mode squeezed vacuum $|z\rangle$, the two-mode squeezed vacuum (TMSV) $|z\rangle_{12}$, the coherent state $|\alpha\rangle$, and Fock states $|n\rangle$; the parameters z, α and n are constrained by what is possible experimentally [19–24].

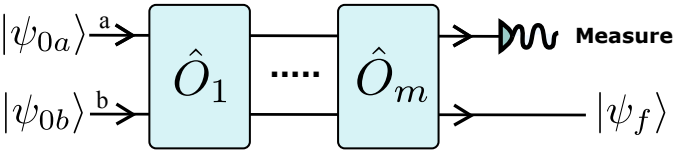

FIG. 1. The state engineering scheme we consider begins with two input states, $|\psi_{0a}\rangle$ and $|\psi_{0b}\rangle$, which are input into the two modes. The states then subsequently pass through a number of operators $\hat{O}_i$. To produce the final quantum state $|\psi_f\rangle$ a heralding measurement is performed on one mode.

Next are the operators, of which the most important is the beam splitter $\hat{U}_T$, where T is the probability of transmission, which serves to mix and entangle the two modes, enabling more exotic and useful states to be produced when part of the entangled state is measured. The other operators we use are the displacement operator $\hat{D}(\alpha)$ and the phase shift $e^{i\hat{n}\theta}$. In addition we include the identity operator $\hat{\mathbb{I}}$, because we are promoting the easiest-to-implement schemes, which would contain as many identities as possible.

The final step of the state engineering scheme is to perform a heralding measurement on one mode of the final state. If, for example, we wish to herald on the one photon state, we can perform a photon-number resolving detection (PNRD), and only keep the output state if one photon is detected. A measurement outcome of one photon therefore *heralds* the desired final state. The heralding measurement corresponds to acting on the two-mode, pre-measurement state with $\langle 1| \otimes \hat{\mathbb{I}}$, followed by normalisation. We are then left with the single mode final state $|\psi_f\rangle$. Recent progress in PNRD (for example using transition edge sensors [18, 25]) has enabled detections of larger numbers of photons possible, so here we allow for heralding number measurements of up to 8 photons.

Many more states, operators and measurements can be included in this toolbox. As discussed in our paper that fully introduces our algorithm [14], the algorithm is design with flexibility in mind, so it is straightforward for more elements to be added (or removed) from the toolbox, depending on the available equipment or desired goal. But here we consider a simplified toolbox to discover whether such a limited toolbox of experimentally viable elements can still produce a range of quantum states to a high fidelity. In [14] we include experimental noise, but in this work we stick to pure states and perfect operators/measurements.

II. FINDING EXPERIMENTS TO ENGINEER SPECIFIC QUANTUM STATES

The task we set our algorithm, Ada, is to find experimental designs to produce a range of specific "target" states to a high fidelity (where the fidelity between two pure states $|\psi\rangle$ and $|\phi\rangle$ is defined as $F \equiv |\langle\psi|\phi\rangle|^2$).

FIG. 2. An artist's impression of our algorithm, Ada, which designs experiments for engineering quantum states. The algorithm is designed for flexibility, and can produce a wide range of quantum states: the illustration depicts, among others, the production of a Schrödinger-cat state, a three-headed cat state [26], and a GKP state [12]. Artwork by Joseph Namara Hollis.

The target states are shown in the table below (normalisation is omitted): these states have a range of properties and are studied and used by both theorists and experimentalists. Fig 2 shows an artist's impression of our algorithm, Ada, producing a range of quantum states.

Name	**State**
Cat	$\|\text{cat}\rangle \backsim \|\alpha\rangle + e^{i\theta}\|-\alpha\rangle$
Squeezed Cat	$\hat{S}(z)\|\text{cat}\rangle$
Zombie	$\|\alpha\rangle + \left\|e^{2\pi i/3}\alpha\right\rangle + \left\|e^{4\pi i/3}\alpha\right\rangle$
ON	$\|0\rangle + \delta\|n\rangle$
Cubic Phase	$\exp\left(i\gamma\hat{q}^3\right)\hat{S}(z)\|0\rangle$

Here $\delta, \gamma \in \mathbb{R}$; $\hat{S}(z)$ is the squeezing operator, given by $\hat{S}(z) = \exp\left[\frac{1}{2}(z^*\hat{a}^2 - z\hat{a}^{\dagger 2})\right]$, where $z \in \mathbb{C}$ and $\hat{a}^\dagger$ and $\hat{a}$ are the creation and annihilation operators, respectively [27]; and $\hat{q}$ is the position quadrature operator, given by $\hat{q} = (\hat{a} + \hat{a}^\dagger)/\sqrt{2}$.

The first state we search for is the optical Schrödinger cat state. This state is inspired from Schrödinger's famous thought experiment in which he proposed to put a macroscopic system – a cat – in a superposition of two distinct states [28]. The implications of this thought experiment still spark heated debate and disagreement, but what escapes controversy is that it would be both important and interesting to create a macroscopic superposition of two distinct states. The optical Schrödinger cat state is moving towards this goal, as it is a superposition of two distinct coherent states, $|\alpha\rangle$ and $|-\alpha\rangle$. While the magnitude of α so far produced in experiments is far from making the state macroscopic, the optical Schrödinger cat state might eventually be produced as a macroscopic superposition, given that $|\alpha\rangle$ is the state produced from a 'perfect' laser. Optical Schrödinger cat states (often referredas just 'cat states') have been produced in a number of experiments, such as [9–11].

The next two states are derived from the Schrödinger cat state. First we can consider *squeezing* a cat state by applying the squeezing operator, as shown in the table. The resulting *squeezed cat state*, whilst in one sense being more exotic than the cat state, isn't necessarily more difficult to produce experimentally [10, 11]. The squeezed cat state can, for example, provide substantial enhancements in quantum metrology [29]. Next, instead of making a superposition of *two* coherent states with different phases as in the cat state, we can make a superposition of *three*, with the relative phases now differing by $2\pi/3$. Such a state can be called a three-headed cat state [30, 31], but here we prefer the name *zombie cat state*, as it represents the superposition of three distinct macroscopic states: dead, alive, and undead.

We used the algorithm to search for cat states, squeezed cat states, and zombie cat states, with the following parameters: $\alpha \in [0, 2]$, $\theta \in [0, 2\pi]$, and $z \in \mathbb{C}$ with $|z| \in [0, 1.4]$.

Next we consider the *ON state*, which is a superposition of the vacuum $|0\rangle$ with an n-photon state $|n\rangle$. By controlling the relative weighting δ it has been shown that the quantum Fisher information of this state, which is often used to quantify the state's phase-measuring potential, can be made arbitrarily large whilst keeping the photon number arbitrarily small [32] (but this state cannot actually achieve infinitely-precise measurements [33, 34]). These states are also important for continuous variable quantum computation because they can be used as a resource to implement cubic phase gates [35]. The latter enable universal quantum computation [36] (a non-Gaussian gate such as a cubic phase gate is essential, as Gaussian gates alone cannot be universal [36]). The ON states we searched for have $\delta \in [0, 1]$ and $n \in [1, 10]$.

Another state useful for continuous variable quantum computation is the *cubic phase state*, which too can be used as a resource to enable cubic phase gates [12, 35, 37–39]. The cubic phase states we searched for have $\gamma \in [0, 0.25]$ and $z \in \mathbb{R}$ with $z \in [0, 1.4]$. Note that "true" cubic phase states are only obtained for infinite squeezing.

Below, in Section III, we fully introduce our algorithm, Ada. But for now Ada can be seen as a black box: we input a request for which quantum states we would like, and Ada outputs experimental designs to produce these states. We ran Ada to find the 5 states introduced above, and obtained the results in table I. All of the states were found to a fidelity above 96%, and two of them were over 99.7%. As an example, to produce an ON state to a

State	Experiment	Parameters	Fidelity
Cat	$\langle 6\|\hat{U}_{T_1}\|z_1, z_2\rangle$	$z_1 = 0.701e^{4.10i}, z_2 = 0.156e^{0.847i}, T_1 = 0.407$	99.85%
Squeezed Cat	$\langle 4\|\hat{U}_{T_2}\|z_3\rangle_{12}$	$z_3 = 1.28e^{0.422i}, T_2 = 0.499$	99.78%
Zombie	$\langle 4\|\hat{D}_1(\alpha_1)\hat{U}_{T_3}\|z_4, 0\rangle$	$z_4 = 1.26e^{2.64i}, T_3 = 0.724, \alpha_1 = 2.16e^{0.265i}$	96.84%
ON	$\langle 8\|\hat{U}_{T_4}\|z_5\rangle_{12}$	$z_5 = 0.985e^{6.28i}, T_4 = 0.606$	97.77%
Cubic Phase	$\langle 5\|\hat{U}_{T_5}\|z_6, 1\rangle$	$z_6 = 0.586e^{3.14i}, T_5 = 0.612$	96.11%

TABLE I. The results found by running Ada to find experimental schemes to produce 5 different quantum states. Normalisation constants are also omitted. The final column shows the fidelity of the output state with the target state. All of these states can be made using present day experimental equipment, and all experimental designs produced states with a fidelity of at least 96% with their target state. See Fig. 3 for a schematic of the experiment to produce an ON state.

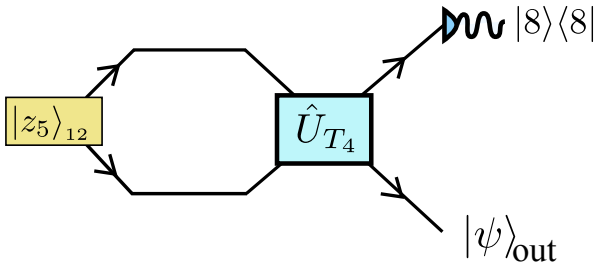

FIG. 3. A sample experiment designed by Ada: this arrangement of quantum optics elements creates a NO state to a fidelity of 97.77%. See Table I for details and parameter values.

fidelity of 97.77% we can send a two-mode squeezed vacuum state through a beam splitter, then do an 8-photon heralding measurement on one mode of the output. The state in the remaining mode will then be close to the target ON state – a schematic of this experiment is shown in Fig. 3.

Similar schemes – with similar simplicity – were found for all 5 states. As discussed above, all of the experimental elements used here are accessible with current technology, with perhaps the biggest challenge in producing the states in table I being the larger-number heralding measurements [18, 25]. The purpose of this paper is to introduce the method of using such an algorithm, and demonstrate its effectiveness; future work will undertake a deeper analysis of the capabilities of Ada and the experiments and states that it has produced.

III. METHODS

A. Using a genetic algorithm to design experiments

We will first introduce the general method of using a genetic algorithm (GA) to design experiments, before describing in more detail how our specific algorithm works. In order to use a GA, we must encode each possible arrangement of states, operators and measurements into a vector, which is known as a *genome*. The genome contains all the information necessary to re-construct a given experimental setup, including all the parameters of the experimental elements. The GA then starts by creating a collection of genomes, which together are known as the *population*. Next, the experimental setup corresponding to each genome is simulated, and the fitness function for each output state is evaluated. The fitness function must take a quantum state as an input, and output a number, the fitness value. The latter quantifies whether the states has the properties we desire or not. In this paper our fitness function will be a measure of how close the output state is to one of those states we are searching for, which are introduced in Section II. But as discussed in [14], the flexibility of our algorithm allows for a wide range of fitness functions to be used to find quantum states for any number of applications.

After the population of genomes is assessed for their fitness, the "fittest" genomes – i.e. the genomes with the largest fitness values – are then selected, and a new population of genomes (the *children*) is generated by mixing some of the genomes together (*crossover*), by randomly modifying (*mutating*) others, and keeping some genomes unchanged (the *elite children*). This next population should, in principle, be comprised of genomes that are "fitter" than before. This process repeats through a number of *generations*, until it is unlikely that any more generations will result in improvements. At this stage, if the algorithm has been designed appropriately, then the fittest genomes will encode optimised solutions. Through this process, our GA evolves quantum experiments that are highly suited to the task at hand. A flow chart of our algorithm is given in Fig. 4.

In order to assess each genome, we must simulate the quantum optics experiment that this genome encodes, then evaluate the fitness function on the output state. As introduced in [14], our simulation of the quantum optics experiments utilises a number of techniques to increase its efficiency. The result is a powerful simulation that allows us to simulate experiments with a truncation of up to 170 photons (in two modes) in the order of seconds, thus allowing a broad range of exotic states – with important contributions at high photon numbers – to be assessed.

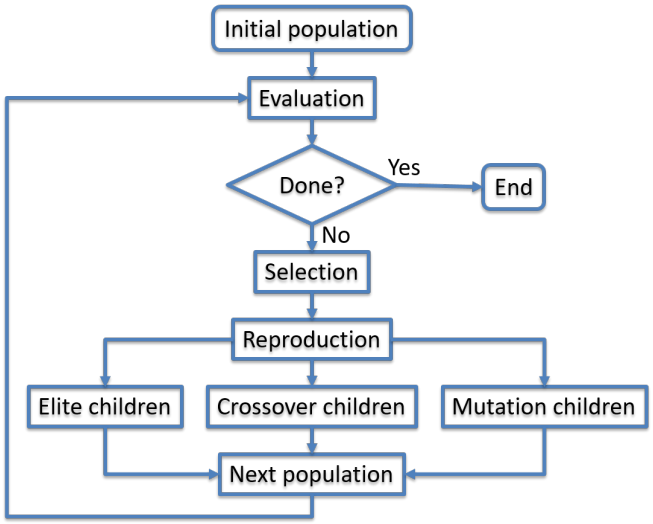

FIG. 4. Flowchart of a genetic algorithm – see Section III A for details of the various steps.

1. *Our three-stage algorithm*

One significant challenge in our approach is that to simulate each experiment accurately we need to truncate the Hilbert space at a high number, but the higher the truncation is, the slower the algorithm will run. Our search space of quantum experiments is huge, and to do an effective search we need to evaluate a large number of different experiments, but this becomes increasingly more computationally expensive for larger truncations. To overcome this, our GA, introduced in [14], has three stages:

1. A large number of random genomes are created and evaluated. In this stage the truncation of the Hilbert space is small (we vary this, but it's often around 30), and therefore the quantum simulation is only approximate. But it is fast. A collection of genomes with the best fitness values are selected for the next stage. While the simulation here is only approximate, it still provides a valuable guide as to where the GA is likely to find experiments with higher fitness values – this stage seeds the GA in the subsequent stage with a substantially better starting population than picking purely at random.

2. We run Matlab's inbuilt GA [40] with a medium-sized population. The simulation is less approximate than stage 1 (because the truncation is larger, around 80), and hence slower. This stage only runs for a set number of generations, usually 10. This stage performs a medium-speed global search, and provides the final stage with a strong population.

3. In the final stage, the simulation is accurate but slow. In this stage, the fitness function will first simulate the circuit specified by the input genome at a very low truncation, then repeat this, increasing the truncation on each iteration, until either the average number of photons in the final state, converges, or until the maximum truncation is reached (where the maximum truncation is specified by the user). This ensures the results are reliable and accurate, while still running in a reasonable time. Here we again use Matlab's inbuilt GA [40], but the population is smaller, and the search is more local.

2. *Overcoming challenges in evaluating our fitness function*

As our task is to find specific states, the obvious fitness function here is to evaluate the fidelity between the target state and the state outputted in each simulation. For example, if we wish to find a cat state, then we can calculate the fidelity to a cat state. But *which* cat state should we be evaluating against? An (unnormalised) cat state given by $|\alpha\rangle + e^{i\theta}|-\alpha\rangle$ has three real-valued parameters (the value of θ and the magnitude and phase of α), so we should compare against every combination of these parameters. Even restricting to small cat states, e.g. $|\alpha| \leq 2$, and discretising the parameters, we still have a large number of cat states to compare against (in the order of 10^5 for this paper). This is not a problem in stages 2 and 3 of our algorithm, because the run-time is dominated by simulating each experiment. But in stage 1, where the truncation is small, the overhead from evaluating the fidelity becomes significant.

This problem is exacerbated when we consider how our algorithm will commonly be used to design new quantum experiments. Generally, we expect a user to specify which states, operators, and measurements they have available, and then to run the algorithm to find which states, and to what fidelity, can be produced with their given equipment. In this case, in stage 1 of the algorithm ideally we would search for *all* of the quantum states of interest (e.g. the 5 classes used in this paper) simultaneously, but this would require a vast number of fidelities to be computed for each simulation (around 10^6 for this paper).

We overcome this problem by using a deep neural network (DNN) to classify each quantum state, thus bypassing the need to calculate each fidelity. The DNN is introduced in detail in the next section, but it suffices for now to consider the DNN as a black box for which we input a quantum state, and the DNN outputs a classification. For this paper, the DNN therefore tells us which of the states introduced in Section II our input state is closest to. For example, if we input the state $|\alpha = 1\rangle + |\alpha = -1\rangle$ into the DNN, it should give the output "cat state". We add a class of state that we call "other", so that we know whenever a simulation produces a state that is not close to any of our desired states.

When used in stage 1 of our algorithm, this method of using a DNN has two significant advantages over the

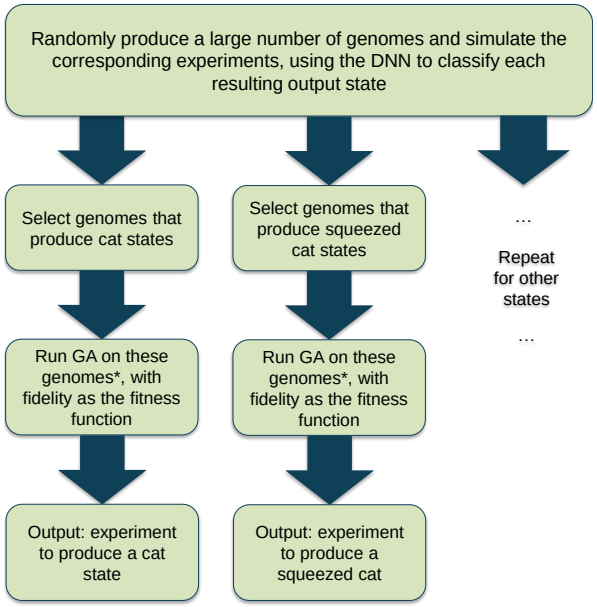

FIG. 5. A flowchart of the overall structure of our hybrid algorithm that incorporates both a deep neural network (DNN) and a genetic algorithm (GA) to design experiments to produce a range of quantum states.
*The GA runs in stages 2 and 3 of the 3-stage algorithm discussed in Section III A 1 of the main text.

alternative method of calculating the fidelities. Firstly, once the DNN is trained to classify states it runs substantially faster than calculating the fidelity. Secondly, the DNN classifies for all 5 states (or more) simultaneously. We therefore use the DNN as follows: In stage 1, we evaluate a large number ($8\text{x}10^6$) of genomes, using the DNN to classify each. We then take the $5\text{x}10^6$ genomes that come closest to producing cat states, according to the DNN, and then send these genomes through stages 2 and 3 (stage 3 uses 10^4 genomes), using the fidelity as the fitness function. We then repeat stages 2 and 3 for the other 4 states. Using this approach, the algorithm found all the experiments introduced in Section II to produce the desired states, all to a fidelity above 96%. When used in this way in stage 1, the DNN is around two orders of magnitude faster than calculating all the fidelities. The overall structure of our hybrid algorithm that incorporates both the DNN and the GA is shown in Fig. 5.

B. A neural network for classifying quantum states

Having explained *how* and *why* we choose to utilise a DNN for classifying quantum states, we will now introduce neural networks, and explain how we construct and train ours.

A classifier deep neural network (DNN) is a machine learning technique used to classify data into a set of classes. In our case, we input a quantum state, and the DNN will output a probability distribution as to whether the state is a cat state, squeezed cat state, zombie cat state, ON state, cubic phase state, or none of the above. Data is input to the DNN as a vector $\vec{x} \in \mathbb{R}^n$. The network is built up of layers of "neurons". In a given layer, each neuron is assigned a value that is calculated by first taking a linear combination of the values in the previous layer, then doing a non-linear activation function. Mathematically, this is expressed as:

$$\vec{x}_{i+1} = \sigma\left(M_i\vec{x_i} + \vec{b_i}\right)$$

Where $\vec{x}_{i+1}$ are the neuron values for the next layer, M_i is matrix of *weights*, $\vec{b_i} \in \mathbb{R}^n$ is a *bias* vector, and σ is the activation function, which is applied element-wise.

Our final layer has 6 neurons, each corresponding to a class of state; this layer uses a different activation function than the rest of the network and produces a probability distribution, corresponding to the probability that the network thinks the input state was each of the classes (e.g. if the first neuron in the output layer is 0.9, the network has determined that the input state has a 90% probability of being a cat state). The neuron with the maximum probability therefore indicates which class the input state is predicted to belong to. The values of M_i and $\vec{b_i}$ must be *learnt* by the network so that it produces the desired output – this is achieved by training, which is described below. A visualisation of the neural network is shown in Fig. 6.

Data must be inputted to the neural network as a real, finite-dimensional vector, but our quantum states belong to a complex, infinite-dimensional Hilbert space. We choose to convert the complex coefficients of each state (in the truncated Fock basis) to real numbers by taking the modulus of the Fock coefficients (throughout the paper we refer to the set of moduli of the Fock coefficients as the "number distribution"). We tried inputting the phase information as extra inputs to the DNN, but this didn't improve the accuracy.

To train the neural network, we first need some labelled training data. This was generated by sampling parameter values for the states from a random distribution. The coefficients of the quantum states in the Fock basis were then calculated using either the analytic expressions (where possible), or by matrix representations of operators acting on the vacuum state (in both cases using a truncated Hilbert space) [27]. Some states with random coefficients were also generated (labelled *other*). The generated training states are inputted to the neural network and a loss function (the softmax cross entropy) is computed using the values of the output layer, and the actual label of the state. The aim of training the network is to modify the values of the biases and weights so that this loss function is minimised; to achieve this, we used an optimisation algorithm called Adam [41], a variation on the stochastic gradient descent method commonly used in machine learning.

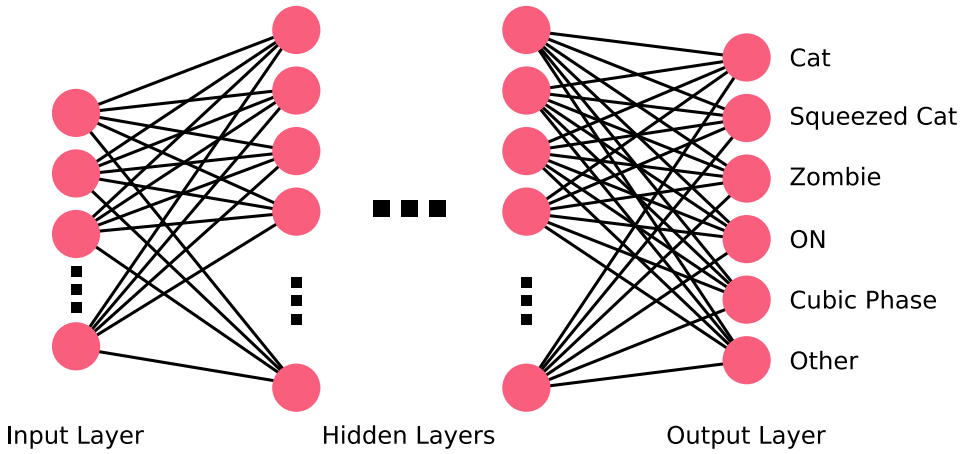

FIG. 6. Our deep neural network (DNN) for classifying quantum states. We input the number distribution of a quantum state, and the DNN outputs a probability distribution corresponding to the probabilities that the inputted state was one of a fixed set of classes (introduced in Section II): cat state, squeezed cat state, zombie cat state, ON state, cubic phase state, or none of the above.

A common problem encountered in machine learning is overfitting, which occurs when the model learnt by the neural network fits to outliers in your training data set, but doesn't generalise well (as an extreme example, if the DNN has enough free parameters it can "memorise" the whole training set given enough training iterations). One method to detect this is to split your data set in two: training and testing. The network is trained using only the data in the training set. Its accuracy can then be evaluated on the testing data set (which the network has never seen before). If the accuracy of the network on the testing set is significantly less than its accuracy on the training set, then it's likely that overfitting has occurred. Using larger training/testing data sets can help to avoid overfitting, as well as techniques such as dropout and regularisation, all of which we experimented with here.

We implemented our DNN in TensorFlow [42]. We used a training data-set containing $10,000$ states, and a testing data-set of $3,000$ states, where approximately $1/6^{\text{th}}$ of the states in each data-set belong to each class. After some experimentation, we settled on a DNN consisting of 3 fully-connected hidden layers, comprising 25, 25 and 10 neurons, respectively. After $5,000$ steps (*epochs*) of training, the network classified the test data with an accuracy of 99.3%. Our DNN is available on GitHub [43] (which includes the code for generating states for the training data).

A standard metric to monitor how well a neural network is classifying data, aside from the accuracy, is the confusion matrix. This is a 6x6 matrix (in our case), where the rows represent the actual classes of the states, and the columns represent the classes predicted by the network. The entry in the i^{th} row and j^{th} column shows how many states of class i were classified as class j. The confusion matrix calculated using our test data was:

	Cat	Sq.-Cat	Zombie	ON	CP	Other
Cat	497	3	0	0	0	0
Sq.-Cat	0	500	0	0	0	0
Zombie	1	0	499	0	0	0
ON	0	0	1	499	0	0
CP	5	7	3	0	485	0
Other	0	0	0	1	0	499

From this we can see that the network is best at classifying squeezed cat states, however it has also classified 10 states incorrectly as squeezed cat states. It is also clear that the network is worst at classifying cubic phase states, which is to be expected as they have the most complex structure of all the states we're interested in. Also note that no states were incorrectly classified as "other", meaning we were unlikely to falsely classify states that we're looking for as being useless.

IV. CONCLUSION

We have introduced a hybrid machine learning algorithm for designing quantum optics experiments. A genetic algorithm was used to search for optimal arrangements of experimental elements that produce a range of useful and interesting optical quantum states; and a deep neural network was used to speed up the evaluations of each experimental arrangement by quickly and accurately classifying quantum states. Combining these techniques, our algorithm found experimental arrangements to produce all 5 states we asked it to, all to a high fidelity. This demonstrates the power and flexibility of the technique of using methods from artificial intelligence and machine learn-

ing to design and optimise quantum physics experiments.

Acknowledgements: We thank Joseph Namara Hollis for the artwork in Fig 2. We acknowledge discussions with Gerardo Adesso, Ryuji Takagi, Tom Bromley, and Ender Özcan. P.K. acknowledges support from the Royal Commission for the Exhibition of 1851. L.O'D. was supported by a vacation bursary from EPSRC.

A. References

[1] V. Dunjko and H. J. Briegel, arXiv preprint arXiv:1709.02779 (2017).
[2] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Nature **549**, 195 (2017).
[3] M. Schuld, I. Sinayskiy, and F. Petruccione, Contemporary Physics **56**, 172 (2015).
[4] P. A. Knott, New Journal of Physics **18**, 073033 (2016).
[5] M. Krenn, M. Malik, R. Fickler, R. Lapkiewicz, and A. Zeilinger, Physical review letters **116**, 090405 (2016).
[6] A. A. Melnikov, H. P. Nautrup, M. Krenn, V. Dunjko, M. Tiersch, A. Zeilinger, and H. J. Briegel, Proceedings of the National Academy of Sciences p. 201714936 (2018).
[7] J. M. Arrazola, T. R. Bromley, J. Izaac, C. R. Myers, K. Brádler, and N. Killoran, arXiv preprint arXiv:1807.10781 (2018).
[8] K. K. Sabapathy, H. Qi, J. Izaac, and C. Weedbrook, arXiv preprint arXiv:1809.04680 (2018).
[9] A. Ourjoumtsev, H. Jeong, R. Tualle-Brouri, and P. Grangier, Nature **448**, 784 (2007).
[10] K. Huang, H. Le Jeannic, J. Ruaudel, V. B. Verma, M. D. Shaw, F. Marsili, S. W. Nam, E. Wu, H. Zeng, Y.-C. Jeong, et al., Physical review letters **115**, 023602 (2015).
[11] J. Etesse, M. Bouillard, B. Kanseri, and R. Tualle-Brouri, Physical review letters **114**, 193602 (2015).
[12] D. Gottesman, A. Kitaev, and J. Preskill, Physical Review A **64**, 012310 (2001).
[13] Note1, the algorithm, Ada, is named after Ada Lovelace, the worlds first computer programmer, and resident of Nottingham, where our own algorithm was born.
[14] R. Nichols, L. Mineh, J. Rubio, J. C. F. Matthews, and P. A. Knott, arXiv preprint arXiv:1812.01032 (2018).
[15] Y. Jin, Swarm and Evolutionary Computation **1**, 61 (2011).
[16] J. Rubio, P. Knott, and J. Dunningham, Journal of Physics Communications **2**, 015027 (2018).
[17] T. J. Bartley, G. Donati, J. B. Spring, X.-M. Jin, M. Barbieri, A. Datta, B. J. Smith, and I. A. Walmsley, Physical Review A **86**, 043820 (2012).
[18] T. Gerrits, S. Glancy, T. S. Clement, B. Calkins, A. E. Lita, A. J. Miller, A. L. Migdall, S. W. Nam, R. P. Mirin, and E. Knill, Physical Review A **82**, 031802 (2010).
[19] M. Mehmet, S. Ast, T. Eberle, S. Steinlechner, H. Vahlbruch, and R. Schnabel, Opt. Express **19**, 25763 (2011).
[20] K. Müller, A. Rundquist, K. A. Fischer, T. Sarmiento, K. G. Lagoudakis, Y. A. Kelaita, C. Sanchez Muñoz, E. del Valle, F. P. Laussy, and J. Vučković, Phys. Rev. Lett. **114**, 233601 (2015).
[21] J. Claudon, J. Bleuse, N. S. Malik, M. Bazin, P. Jaffrennou, N. Gregersen, C. Sauvan, P. Lalanne, and J. Gérard, Nature Photon. **4**, 174 (2010).
[22] O. Morin, V. DâĂŹAuria, C. Fabre, and J. Laurat, Opt. Lett. **37**, 3738 (2012).
[23] A. Ourjoumtsev, R. Tualle-Brouri, and P. Grangier, Phys. Rev. Lett. **96**, 213601 (2006).
[24] K. Huang, Ph.D. thesis, l'École Normale Supérieure de Paris and East China Normal University. (2015).
[25] P. C. Humphreys, B. J. Metcalf, T. Gerrits, T. Hiemstra, A. E. Lita, J. Nunn, S. W. Nam, A. Datta, W. S. Kolthammer, and I. A. Walmsley, arXiv preprint arXiv:1502.07649 (2015).
[26] S.-Y. Lee, C.-W. Lee, J. Lee, and H. Nha, arXiv preprint arXiv:1505.06000 (2015).
[27] S. M. Barnett and P. M. Radmore, *Methods in theoretical quantum optics*, vol. 15 (Oxford University Press, 2002).
[28] E. Schrödinger, Naturwissenschaften **23**, 823 (1935).
[29] P. Knott, T. Proctor, A. Hayes, J. Cooling, and J. Dunningham, Physical Review A **93**, 033859 (2016).
[30] S.-Y. Lee, C.-W. Lee, H. Nha, and D. Kaszlikowski, JOSA B **32**, 1186 (2015).
[31] L.-y. Jiang, Q. Guo, X.-x. Xu, M. Cai, W. Yuan, and Z.-l. Duan, Optics Communications **369**, 179 (2016).
[32] (????).
[33] M. J. W. Hall and H. M. Wiseman, New J. Phys. **14**, 033040 (2012).
[34] M. J. W. Hall, D. W. Berry, M. Zwierz, and H. M. Wiseman, Phys. Rev. A **85**, 041802 (2012).
[35] K. K. Sabapathy and C. Weedbrook, Phys. Rev. A **97**, 062315 (2018).
[36] P. Kok and B. W. Lovett, *Introduction to optical quantum information processing* (Cambridge University Press, 2010).
[37] S. Ghose and B. C. Sanders, Journal of Modern Optics **54**, 855 (2007).
[38] R. Takagi and Q. Zhuang, arXiv preprint arXiv:1804.04669 (2018).
[39] M. Gu, C. Weedbrook, N. C. Menicucci, T. C. Ralph, and P. van Loock, Physical Review A **79**, 062318 (2009).
[40] T. M. Inc., *Global optimization toolbox: User's guide (r2018a)*, https://uk.mathworks.com/help/gads/index.html. Last accessed 2018-07-17.
[41] D. P. Kingma and J. Ba (2014), arXiv:1412.6980 [cs.LG], 1412.6980, URL https://arxiv.org/abs/1412.6980.
[42] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., in *OSDI* (2016), vol. 16, pp. 265–283.
[43] *Our DNN and quantum-state generator on GitHub*, https://github.com/lewis-od/Quantum-Optics (2018).
[44] M. G. A. Paris, Phys. Lett. A **217**, 78 (1996).
[45] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, 2010).

Appendix A: Quantum optics toolbox details

Input states - The squeezed vacuum is given by $|z\rangle = \hat{S}(z)|0\rangle$ where the squeezing operator is $\hat{S}(z) = \exp\left[\frac{1}{2}(z^*\hat{a}^2 - z\hat{a}^{\dagger 2})\right]$ and $z = re^{i\theta_s}$ where r is the (positive and real) amplitude, $\theta_s \in [0, 2\pi]$ is the squeezing angle and $\hat{a}$ ($\hat{a}^\dagger$) is the annihilation (creation) operator. Squeezed states can be made up to $r \approx 1.4$, but this is extremely challenging experimentally so we set the limit to $r = 1.3$ [19]. Similarly, the two mode squeezed vacuum is given by $|z\rangle_{12} = \hat{S}_{12}(z)|0,0\rangle$, where the two mode squeezing operator is $\hat{S}_{12}(z) = \exp(z^*\hat{a}\hat{b} - z\hat{a}^\dagger\hat{b}^\dagger)$, where $\hat{a}$ and $\hat{b}$ act on modes 1 and 2, respectively, and again z is complex. The coherent state is given by $|\alpha\rangle = \hat{D}(\alpha)|0\rangle$ where the displacement operator is $\hat{D}(\alpha) = \exp(\alpha\hat{a}^\dagger - \alpha^*\hat{a})$, $\alpha = |\alpha|e^{i\theta_c}$ where $|\alpha|$ is the amplitude, and $\theta_c \in [0, 2\pi]$ is the coherent state phase. The amplitude of the coherent state can be large in experiments, so instead it is limited by the numerical methods we use: we set the limit to $\alpha = 4$. The final input state is the Fock state of which the simplest is the vacuum $|0\rangle$. Single photons, $|1\rangle$, can be emitted from a quantum dot [20, 21] or heralded [22]. We also consider the two photon state, $|2\rangle$, which has been made in [23, 24]. Higher number Fock states can be made, e.g. by heralding, but are challenging to produce to a high fidelity and are not included here.

Operators - The beam splitter is described by the unitary operator $\hat{U}_T = e^{-i\theta_b(e^{i\phi_b}\hat{a}^\dagger\hat{b} + e^{-i\phi_b}\hat{a}\hat{b}^\dagger)}$, where $\hat{a}$ and $\hat{b}$ are annihilation operators for the two modes, and we choose the arbitrary phase to be $\phi_b = -\pi/2$. Here $T = \cos^2\theta_b$ is the transmissivity of the beam splitter and therefore for a 50:50 beam splitter $\theta_b = \pi/4$ giving $\hat{U}_{T=50}$. Next, the displacement operator, $\hat{D}(\beta)$ (defined above), is implemented by mixing the state with a large local oscillator at a highly transmissive beam splitter [44] (β has the same restrictions as α). The phase operator is given by $e^{i\hat{n}\theta}$ where $\hat{n} = \hat{a}^\dagger\hat{a}$ and $\theta \in [0, 2\pi]$.

Measurements - After we have applied a number of operators we perform a heralding measurement on one mode of the final state. For example, if we wish to herald on the one photon state we can perform a number resolving detection [18, 25], and only keep runs that measure one photon. The measurement is given by a projection [45]: to follow the single photon example we project with $|1\rangle\langle 1| \otimes \hat{\mathbb{I}}$. We are then left with a separable state $|1\rangle \otimes |\psi_f\rangle$, but we can ignore the measurement mode, and after normalisation we are left with the final one mode state: $|\psi_f\rangle$. This whole process can be more easily modeled by acting on the two-mode, pre-measurement state with $\langle 1| \otimes \hat{\mathbb{I}}$. In the main text we drop the identity and just write $\langle 1|$, and this measurement is always performed on the first mode of Fig. 1.