SUPPLEMENTARY INFORMATION

A. SUPPLEMENTARY NOTE: Quantum dot environment

In DRL, among the key components is the formal model of the environment with which the agent interacts. Therefore, we build an environment from the quantum dot device for training our algorithm and name it the *quantum dot environment* (QDE). The QDE was developed to be compatible with the OpenAI Gym interface [1]. This environment is ready to be used for benchmarking and training existing DRL algorithms. In addition, this environment is useful for interested DRL researchers to develop their methods for improving quantum technologies.

The voltage space of the QDE is delimited by a 640 mV × 640 mV window defined by the barrier gates. The window is divided in 32 mV × 32 mV blocks and the agent is initialised in a randomly selected block.

State

A state s is the statistics set, of a block, consisting of the means μ and standard deviations σ of the current in each of the nine sub-blocks.

Instead of making densely overlapping blocks by a moving kernel horizontally and vertically, we propose to represent each state by 3 blocks per dimension for simplicity. In other words, the image is represented by 3^d blocks where d is the dimension. The dimension d corresponds to the number of gates used. In our setting, each state in 2 gates includes 9 blocks as the tensor of $9 \times 32 \times 32$ dimensions and 3 gates will have the tensor of $27 \times 32 \times 32$. Examples of the state and blocks using two gates can be found in Fig. 1 of the main text.

The design of this state representation will bring two major advantages. This approach ensures that the agent is less prone to over-fitting during training and more robust to experimental noise. The second benefit is for scalability as we can efficiently extend to a higher number of dimensions. While taking a grid scan measurements, which scale exponentially with the number of dimensions, we can use random sampling techniques to obtain convergence in the values of the state representation. This random sampling technique will scale more favourably with higher dimensions. Under this representation, the state includes a statistics vector of 9×2 dimensions.

Action

Our action space includes increasing (+) or decreasing (−) each gate voltage. We have specially designed two actions to modify both gates simultaneously. In higher dimensional setting, such as controlling $d > 2$ gates, this action space can be generalised wherein the number of actions is $2 \times d + 2$.

Reward

The reward function is carefully constructed to force the agent to learn to navigate through the voltage landscape and identify bias triangles using the fewest N measurements. We follow the popular Taxi-v2 environment* to design the reward scores. We summarise the components of the reward function in Table 1. We note that the utility of the agent is robust with respect to different magnitudes of these scores, provided that the detection of a pair of bias triangles receives a much higher score than block measurement steps.

Supplementary Table 1. Summary of the reward function.

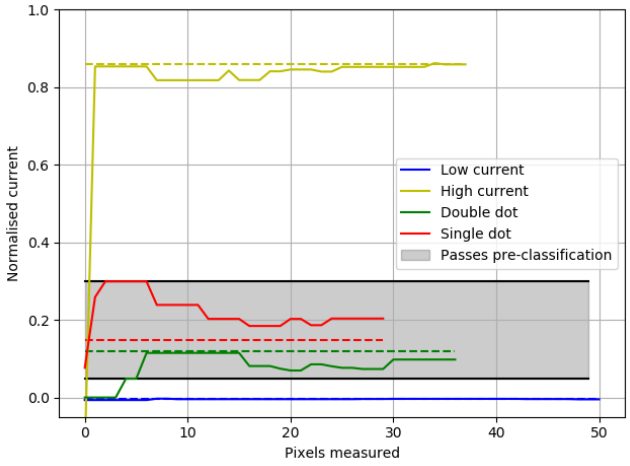
Instance	**Reward**	**Termination**
Each block measured	-1	False
Bias triangle detected	+10	True
N equal to $N_{\max}$	-10	True

We assign high reward to our target state of bias-triangles. We encourage the algorithm to find the bias-triangles using the fewest number of measurement by designing the reward score as follows. We assign the highest reward $r = +10$ to the bias-triangles location. Then, other states will take $r = -1$. The maximum number of steps per episode during training is set as 300. Beyond this threshold, if the algorithm cannot find the bias-triangles, it will terminate and assign $r = -10$. The maximum number of steps controls how far away from a starting point the device-measurement can go.

B. SUPPLEMENTARY NOTE: Random Pixel Measurement and Pre-classifier

To assess the state of a block, the algorithm first conducts a random pixel measurement. Pixels are repeatedly sampled at random from the block and the statistics are calculated for each sub-block until convergence. The convergence of both the mean and standard deviation of each sub-block must be satisfied before the measurement is terminated. The convergence is accepted if the mean change in the values of the state representation is less than a threshold percentage (one percent threshold for this paper) of the state representation prior to the update. The state vector is then assessed by the pre-classification stage. If the mean current values, of any of the sub-blocks,

* https://gym.openai.com/envs/Taxi-v2/



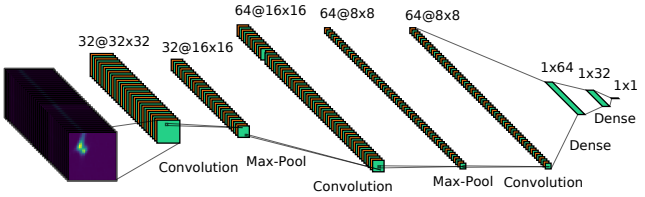
Supplementary Figure 1. **Pixel sampling convergence** for the random pixel measurement in sub-blocks taken from pinch-off (low current), open (high current), single dot, and double dot regimes. The threshold values set for the pre-classification tools are indicated. Therefore, any sub-block within the grey shaded region will pass the pre-classification. The dashed lines represent the true means after measuring the sub-blocks using a 1 mV × 1 mV resolution scan.

falls between the threshold values, calculated using the initialisation one-dimensional scans, then the block is pre-classified as a boundary region. The convergence of the normalised mean current of a sub-block, in the random pixel sampling measurement method is shown in Fig. 1, for low and high current, as well as single and double dot sub-blocks. The sub-blocks with single and double dot transport features fall within the pre-classifier threshold values and therefore, in the full algorithm, would be measured using a grid scan and evaluated by the CNN binary classifier.

Satisfactory convergence for a block is achieved in fewer than 50 pixel measurements in all regimes, compared to the 1,024 pixels measured in a grid scan of the block. This represents a huge improvement in measurement efficiency and the evaluation of a state of the DRL agent.

C. SUPPLEMENTARY NOTE: CNN Binary Classifier

A Convolutional Neural Network (CNN) [2, 3] is a multilayered neural network with a special architecture to detect complex patterns. To decide whether the agent has found bias triangles in a given block, the algorithm uses a CNN as a binary classification tool. If the CNN outputs a value greater than a 0.5 threshold, corresponding to the classification of a pair of bias triangles, then the algorithm is terminated. An optimisation of the threshold value could enable us to reduce the number of false positive classifications.



Supplementary Figure 2. **CNN binary classifier architecture.**

Supplementary Table 2. CNN binary classifier confusion matrix

Confusion Parameters	
True positive	18%
False positive	4%
True negative	76%
False negative	2%
F-measure	85%
Accuracy	94%

Network Architecture

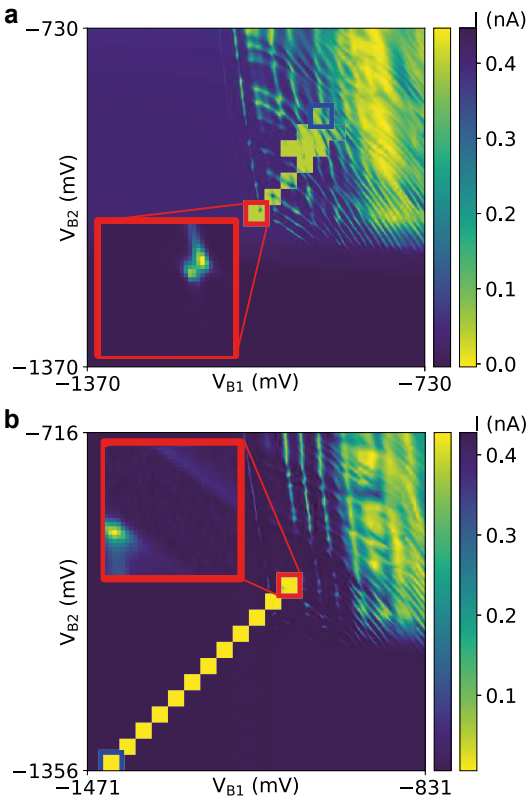
We summarise in Fig. 2 the network architecture and hyperparameters used. There are a total of 320065 trainable parameters. The convolutional layers have a Rectified Linear Unit (RELU) activation function, while the dense layers have and Exponential Linear Unit (ELU) activation function. The final, output, layer has a Sigmoid activation function.

Confusion Matrix

The CNN was trained over 10 epochs using 11425 data points in the training set, 4896 in the validation set and 6994 in the test set. Data was augmented by applying rotations. We trained the network using an Adam optimiser [4] and a binary cross-entropy loss function. Regularisation was achieved using a L2 regulariser, set to 0.0001, as well as drop-out for the dense layers, set at 0.1. In Table 2 we present a summary of the prediction results on the test set of the binary classification problem. The confusion parameter representation is useful for analysing the types of error that a classifier typically makes. The F-measure and accuracy are other commonly used metrics to analyse the efficacy of a binary classification tool.

Positive Examples

As the DRL agent navigates through the environment, the algorithm evaluates each block using the pre-classification protocol. If the block passes the pre-classifier stage, a grid scan of the block is measured and the CNN



Supplementary Figure 3. **Example trajectories**. (**a,b**) Example trajectories from Fig. 4 **a, b** of the main text, respectively, with the insets showing the windows which triggered the stopping condition for the algorithm.

binary classification tool is used to evaluate the block. If the CNN positively classifies the block as containing bias triangles the algorithm is terminated and the run treated as successful. In Fig. 3 we show the blocks that were positively classified by the CNN, causing the algorithm to terminate during the real-time measurements.

D. SUPPLEMENTARY NOTE: DRL Decision Agent

Network Architecture

We first summarise the network architecture and hyperparameters used in Table 3. We further illustrate the model architecture in Fig. 4.

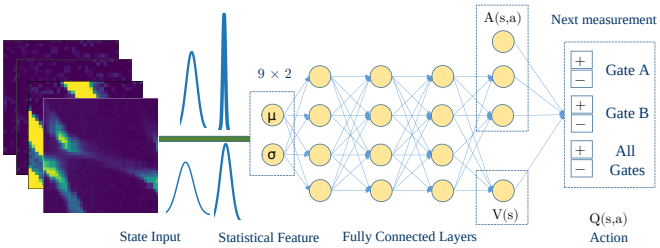
Training

We present the pseudo code for the training of the DRL decision agent with prioritised experience replay in Fig. 5. The training process starts as follows. An agent initially will make random action choices to gain experiences which will be stored in a replay buffer B. From this buffer, the data sample will be randomly selected at a rate proportional to the temporal difference (TD) error. Particularly, it prefers to pick samples with unexpected transitions since these contain more information to learn than from others samples. Then, the neural network will be updated given such 'unexpected' samples to improve the networks policy for the next iterations.

We then illustrate the learning process of our DRL agent by showing the N measurements required to locate bias triangles, in the training and testing devices as a function of the number of training episodes in Fig. 6. This training was performed to assess the learning rate of the DRL network and is performed in an environment where the bias triangles are labelled in advance by human experts and the CNN and pre-classifier modules are not used.

Supplementary Table 3. Deep Reinforcement Learning Architecture

Hyper-parameters Used	
Discount factor	0.5
Optimizer	Adam
Number of episodes	10000
Mini batch-size	32
Decay rate in ϵ greedy	$1e^{-4}$
Replay buffer	20000
PER-β (start, final, no steps)	$(1.0, 0.6, 1000)$
Learning rate	$2.5e^{-6}$
FC Layers	$128, 64, 32$
FC Layers (Dueling)	$64, 1$



Supplementary Figure 4. **Summary of DRL framework.** Our deep reinforcement learning framework using a statistical state representation.

Performance

We summarise the online performance, in Table 4 and the offline performance in Table 6, of the DRL decision agent with respect to the random decision agent. We use the two-tailed Wilcoxon signed rank test [5] to assess the null hypothesis that the DRL and Random agent's performances are drawn from the same distribution. The p-value, given in Table 5 and Table 7 for online and offline tests respectively, represents the confidence in the null

Algorithm 1 Training dueling deep Q-network with prioritised experience replay.

Input: Replay buffer B, neural network model weight θ, minibatch size k, $\Delta = 0$, $p_1 = 1$

Observe s_0 and make the action $a_0 = \pi_\theta(s_0)$
for step $t = 1$ to T **do**
  Observe s_t, r_t based on the action a_{t-1}
  Store transition $(s_{t-1}, a_{t-1}, s_t, r_t)$ in buffer B
  for $j \leq k$ # prioritised experience replay **do**
    Sample transition $j \sim P_j = p_j / \sum_i p_i$
    Compute IS weight $w_j = (N \times P_j)^{-\beta} / \max_i w_i$
    Compute TD-error $\delta_j = r_j + \gamma_j \max_{a'} Q(s_{j+1}, a') - Q(s_j, a_j)$ and update $p_j \leftarrow |\delta_j|$
    Accumulate weight-change $\Delta \leftarrow \Delta + w_j \times \delta_j \times \nabla_\theta Q(S_{j-1}, A_{j-1})$
  end for
  Update the network parameter using gradient descent: $\theta \leftarrow \theta + \alpha\Delta$ and reset $\Delta = 0$
end for

Supplementary Figure 5. **Pseudocode.** Training the dueling deep Q-network with prioritised experience replay.

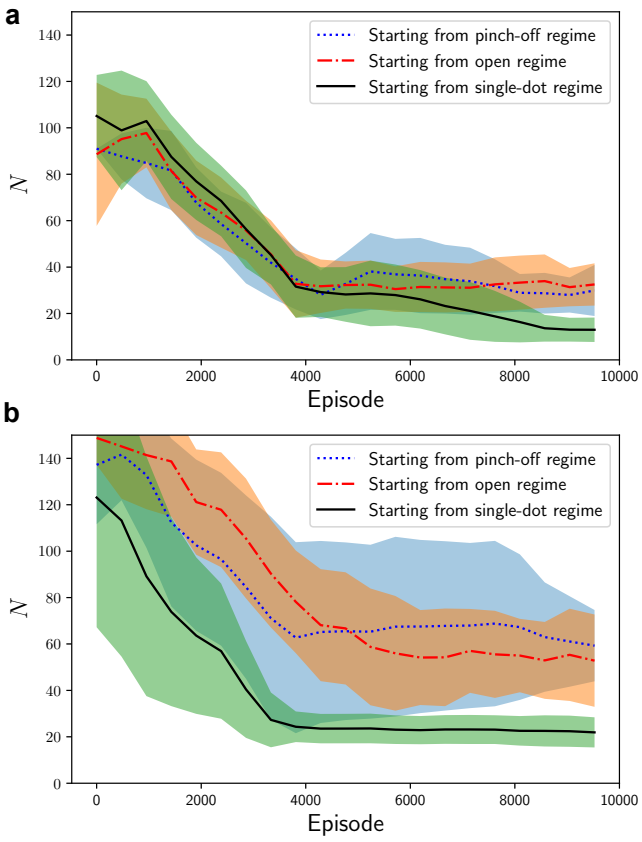
Supplementary Table 4. Summary of the **online** performance of the DRL and Random decision agents online in the two parameter regimes. The performance metrics used are the number of blocks measured, N, before identifying a pair of bias triangles and the corresponding lab time.

Agent	**DRL**	**Random**
Region I median lab time (s)	932	683
10% percentile time (s)	228	222
90% percentile time (s)	2430	4844
Region II median lab time (s)	822	989
10% percentile time (s)	181	349
90% percentile time (s)	1766	1500
Region I $\bar{N}$	41	54
10% percentile N	9	15
90% percentile N	104	135
Region II $\bar{N}$	32	85
10% percentile N	10	50
90% percentile N	94	143

hypothesis. The null hypothesis can only be rejected with confidence, at a level of 2%, in the online results in the case of $\bar{N}$ in region II. For the offline results, the null hypothesis can be rejected for $\bar{N}$ in both regions. A one-tailed Wilcoxon signed rank test demonstrates that the median of the differences $(\bar{N}_{\text{DRL}} - \bar{N}_{\text{Random}})$ is less than zero. We can therefore conclude that the DRL agent offers a statistically significant advantage over the random agent.

1. *Policy*

In the reinforcement learning context, a policy defines what an agent does to accomplish a task. We present

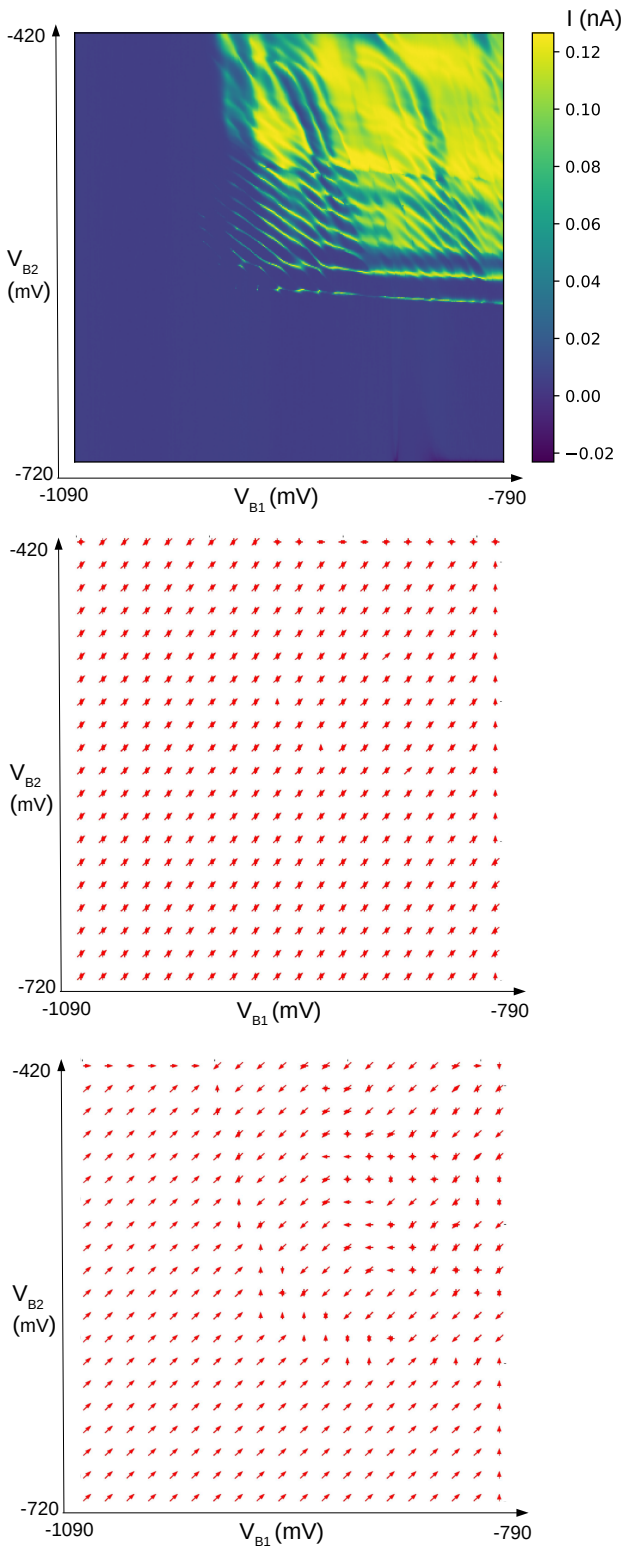


Supplementary Figure 6. **Training convergence.** We evaluate the performance of our DRL agent during training in an environment with different starting locations. The bias triangles are labelled in advance by experts rather than using the CNN and pre-classifier modules. **a** The performance is recorded on the same quantum device in which training is performed and **b** a different device. This demonstrates that our algorithm is flexible and robust against device variability.

Supplementary Table 5. Summary of the Wilcoxon signed rank test analysis on the **online** performance of the DRL and Random decision agents in the two parameter regions. The performance metrics used are the number of blocks measured, N, before identifying a pair of bias triangle and the corresponding lab time.

	Wilcoxon signed rank p-value
Region I lab time	0.72
Region II lab time	0.58
Region I N	0.72
Region II N	0.02

the optimal policies at different training stages in Fig. 7 wherein we use arrows to indicate the action, i.e., the direction to move in the gate voltage space to perform the next measurement. The algorithm learns that it should move towards more positive gate voltages if the state is pinch-off (low-current) or go towards more negative gate voltages if the state is the open regime (high-current).



Supplementary Figure 7. **Optimal policies.** For a current map **Upper**, we plot the optimal policies learned at **Center** early stage and **Lower** later stage of the training process. The arrows indicates, at the given location, the optimal direction to move. Two arrows represent two probable actions which both with high chances of being optimal. At the early stage of the training, the agent's policy has more uncertainty regarding which action to select. While at the later stage, after being trained, it consistently makes the optimal decision.

Supplementary Table 6. Summary of the **offline** performance of the DRL and Random decision agents online in the two parameter regions. The performance metrics used are the number of blocks explored, N, before identifying a pair of bias triangles (the lower the N, the better the performance). For offline experiments, lab times are not a performance metric to be considered.

	DRL agent	**Random agent**
Region I $\bar{N}$	6	22
10% percentile N	1	2
90% percentile N	64	46
Region II $\bar{N}$	17	30
10% percentile N	2	3
90% percentile N	31	101

Supplementary Table 7. Summary of the Wilcoxon signed rank test analysis on the **offline** performance of the DRL and Random decision agents in the two parameter regions. The performance metrics used are the number of blocks measured, N. For offline experiments, lab times are not a performance metric to be considered.

	Wilcoxon signed rank p-value
Region I N	<0.001
Region II N	<0.001

E. SUPPLEMENTARY NOTE: The Nelder-Mead numerical optimisation method

We construct a fitness function by taking the L^2-norm of a probability vector defining a difference metric between the current state, i.e. a given transport regime, and the target state or transport regime. In slight variance from the implementation in [6], we have defined the probability vector of the current state as $(p(s), 1-p(s))^T$ and the target vector defined as $(1,0)^T$. s is a coordinate in gate voltage space and this coordinate's fitness value is calculated, as above, by evaluating the CNN prediction $p(s)$ of the probability that a window ($32\,\mathrm{mV} \times 32\,\mathrm{mV}$) defined around s contains bias triangles. Thus, in single and double dot transport regimes, the value of $p(s)$ should be higher than in the pinch-off and open regimes. The value of the L^2-norm should have minima at the locations of bias triangles. The Nelder-Mead numerical optimisation method, with two gate voltages as free parameters, then automated the location of these minima. This method converges on local minima, in n dimensions, by evaluating a set of $n+1$ test coordinates within the optimisation landscape, called a simplex. We defined the initial simplex similarly to [6], as the fitness value of the starting (s) and two additional coordinates obtained by reducing the voltage on each of the barrier gate voltages one at a time by $75\,\mathrm{mV}$.

SUPPLEMENTARY REFERENCES

[1] Brockman, G. *et al.* OpenAI Gym. Preprint at `http://arxiv.org/abs/1606.01540` (2016).

[2] Lecun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).

[3] Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolutional neural networks. *NeurIPS* **25** ,1097-1105, (2012)

[4] Kingma, D. P. & Ba, J. L. Adam: A method for stochastic optimization. Preprint at `https://arxiv.org/abs/1412.6980` (2015).

[5] Wilcoxon, F. Individual Comparisons by Ranking Methods. *Biometrics Bulletin* **1**, 80 (1945).

[6] Zwolak, J. P., Kalantre, S. S., Wu, X., Ragole, S. & Taylor, J. M. QFlow lite Dataset: A Machine-Learning approach to the charge states in quantum dot experiments. *PLoS ONE* **13**, 10 (2018).