# Predicting disruptive instabilities in controlled fusion plasmas through deep learning

Julian Kates-Harbeck<sup>1,2,3</sup>\*, Alexey Svyatkovskiy<sup>4,5</sup> & William Tang<sup>3,4</sup>

Nuclear fusion power delivered by magnetic-confinement tokamak reactors holds the promise of sustainable and clean energy<sup>1</sup>. The avoidance of large-scale plasma instabilities called disruptions within these reactors<sup>2,3</sup> is one of the most pressing challenges<sup>4,5</sup>, because disruptions can halt power production and damage key components. Disruptions are particularly harmful for large burning-plasma systems such as the multibillion-dollar International Thermonuclear Experimental Reactor (ITER) project<sup>6</sup> currently under construction, which aims to be the first reactor that produces more power from fusion than is injected to heat the plasma. Here we present a method based on deep learning for forecasting disruptions. Our method extends considerably the capabilities of previous strategies such as first-principles-based<sup>5</sup> and classical machine-learning<sup>7-11</sup> approaches. In particular, it delivers reliable predictions for machines other than the one on which it was trained—a crucial requirement for future large reactors that cannot afford training disruptions. Our approach takes advantage of high-dimensional training data to boost predictive performance while also engaging supercomputing resources at the largest scale to improve accuracy and speed. Trained on experimental data from the largest tokamaks in the United States (DIII-D12) and the world (Joint European Torus, JET<sup>13</sup>), our method can also be applied to specific tasks such as prediction with long warning times: this opens up the possibility of moving from passive disruption prediction to active reactor control and optimization. These initial results illustrate the potential for deep learning to accelerate progress in fusion-energy science and, more generally, in the understanding and prediction of complex physical systems.

Tokamaks use strong magnetic fields to confine high-temperature plasmas, with the goal of creating the conditions for extracting power from the resulting fusion reaction in the plasma<sup>14</sup>. However, the thermal and magnetic energy in the tokamak can drive plasma instabilities that lead to disruptions<sup>2</sup>—a central science and engineering challenge facing practical power production from nuclear fusion. Disruptions abruptly destroy the plasma's magnetic confinement, thus terminating the fusion reaction and rapidly depositing the plasma energy into the confining vessel<sup>3,4</sup> (see the section on 'Disruptions' in the Supplementary Information for details). The resulting thermal and electromagnetic force loads can irreparably damage key device components. However, if an impending disruption is predicted with sufficient warning time<sup>3</sup>, a disruption mitigation system (DMS), using techniques such as massive gas or shattered pellet injections<sup>15</sup>, can be triggered. The DMS terminates the discharge but substantially reduces the deleterious effects of the disruption. Present guidance for the minimum required warning time for successful disruption mitigation on ITER is about 30 milliseconds, although it is in general set by the exact response time of the DMS and may be reduced in the future through progress in DMS technologies<sup>3</sup>. Throughout this paper, we describe the predictive performance of all methods at this 'deadline' of 30 milliseconds before disruption. However, even longer warning times could allow for a 'soft' rampdown of the plasma current or alternative active plasma control, avoiding disruption without terminating the discharge<sup>3</sup>.

Although plasma instabilities and disruptions are in theory predictable from first principles<sup>16</sup>, this has proven to be extremely challenging, because an accurate physical model<sup>5</sup> would need to take into account, first, a vast range of spatiotemporal scales; second, multiphysics considerations; and third, the complexity of disruption causes and precursor events<sup>17</sup>. Just as for many other fundamental questions across the physical sciences<sup>18,19</sup>, the inherent complexity of the problem can make first-principles-based approaches impractical on their own.

On the other hand, recent statistical and classical machine-learning approaches (we will refer here to machine-learning models that do not apply deep-learning paradigms as 'classical' algorithms) based on realtime measured data have shown promising results<sup>7–10</sup>; although they still have shortcomings, they represent the state of the art<sup>3</sup> for disruption prediction. Here we introduce the fusion recurrent neural network (FRNN)—a new disruption-prediction method based on deep learning that builds on these pioneering efforts and extends the capabilities of data-driven approaches in several crucial ways.

Specifically, our method delivers predictions for devices unseen during training; uses the information contained in high-dimensional diagnostic data, such as profiles, in addition to scalar signals; avoids the need for extensive feature engineering and selection<sup>20,21</sup>; and enables rapid training times through high-performance computing. The cross-device prediction in particular will be key for powerful near-future burning plasma machines such as ITER, as they cannot withstand more than a few<sup>3</sup> disruptions. Accordingly, training data from such devices can be expected to be scarce.

Deep neural networks<sup>22</sup> in general consist of many layers of parameterized nonlinear mappings, whose parameters are trained ('learned') using backpropagation. They have been successful at learning to extract meaningful features from high-dimensional data such as speech, text and video. In particular, recurrent neural networks (RNNs) powerfully handle sequential data by maintaining information in an internal state that is passed between successive time steps, in addition to taking into account new input data at every time step. Meanwhile, convolutional neural networks (CNNs) can learn salient, low-dimensional representations from high-dimensional data by successively applying convolutional and downsampling operations. As the first application of deep learning to disruption prediction, the specific architecture of FRNN combines both recurrent and convolutional components to extract spatiotemporal patterns from multimodal and high-dimensional sensory inputs. The overall workflow and detailed architecture of our approach are presented in Fig. 1.

Missing a real disruption or calling it too late (false negative) is costly because its damaging effects go unmitigated, while triggering a false alarm (false positive) wastes experimental time and resources. Changing the alarm threshold value for the scalar 'disruptivity' output of a prediction model (Fig. 1d) allows a trade-off between these two economic operation factors. A low threshold means that the alarm is triggered more easily, which will result in fewer missed disruptions but

<sup>&</sup>lt;sup>1</sup>Department of Physics, Harvard University, Cambridge, MA, USA. <sup>2</sup>Program for Evolutionary Dynamics, Harvard University, Cambridge, MA, USA. <sup>3</sup>Princeton Plasma Physics Laboratory, Princeton, NJ, USA. <sup>4</sup>Princeton Institute for Computational Science and Engineering, Princeton University, Princeton, NJ, USA. <sup>5</sup>Present address: Microsoft, One Microsoft Way, Redmond, WA, USA. \*e-mail: jkatesharbeck@g.harvard.edu



**Fig. 1** | **System overview and disruption-prediction workflow. a**–**e**, The top image shows an interior view of the JET tokamak, with a nondisruptive plasma on the left and a disruptive plasma on the right. Diagnostics (**a**) provide streams of sensory data (**b**) which are fed to the RNN-based deep learning algorithm (**c**) every 1 ms, producing a corresponding 'disruptivity' output at every time step (**d**). If the output crosses a preset threshold value (dashed horizontal line), a disruption alarm is called (red star). This alarm triggers mitigation action, such as gas injection (**e**) into the tokamak, to reduce the deleterious effects of the impending disruption. **f**, A detailed schematic of our deep-learning model. The input data consist of scalar zero-dimensional (0D) signals and 1D profiles. Nlayers of convolutional (containing  $N_{\rm F}$  filters each) and downsampling (max-pooling) operations reduce the dimensionality of the profile data and extract salient low-dimensional representations (features; here, 1D

more false alarms, and vice versa for a high threshold. This trade-off is captured as a receiver–operator characteristic (ROC) curve<sup>23</sup> (see the Methods subsection 'Target functions' and Extended Data Fig. 1 for details). The area under this ROC curve (AUC)—our metric for evaluating algorithms—lies between 0 and 1, and measures the ability of a predictive method to catch real disruptions early enough, while at the same time causing few false positives.

In order to assess our algorithm, we trained it to predict disruptive and nondisruptive outcomes using past experimental data from the JET and DIII-D tokamaks, currently comprising over 2 terabytes. Training our model effectively required solutions to several unique challenges, such as training with diverse and long sequences and finding signal normalizations that scale appropriately between machines (see the Methods subsections 'Data considerations' and 'Algorithm and training details' for further information, and Extended Data Tables 1, 2 for a detailed summary of the signals and datasets used). We compare FRNN to the previous state of the art represented by support vector machines (SVMs)<sup>10</sup> and small multilayer perceptrons (MLPs)<sup>8</sup>, as well as to other promising models from the machine-learning literature such as random forests<sup>24</sup> and gradient-boosted trees<sup>25</sup>. Table 1 reports AUC features). These features are concatenated with the 0D signals and fed into a multilayer long/short-term memory network (LSTM) with *M* layers, which also receives its internal state from the last time step (T = t - 1) as input. The resulting final feature vector ideally contains salient information from the past temporal evolution ( $T \le t - 1$ ) and the present state of all signals (T = t). This vector is fed through a fully connected layer to produce the output. Panel **a** has been modified from an image of the interior of JET obtained from the EUROfusion media library at www. euro-fusion.org/media-library.  $I_{p,target}$ , plasma current target;  $I_{l_b}$  internal inductance; LM, locked-mode amplitude;  $I_p$ , plasma current;  $P_{in}$ , input power;  $P_{rad,core}$  core radiated power;  $\beta$ , normalized plasma pressure;  $n_e$ , electron density;  $W_{MHD}$ , plasma energy;  $P_{rad}$  total radiated power;  $T_e(\rho)$ , electron-temperature profile;  $n_e(\rho)$ , electron-density profile.

values for the best version of our model and the best classical model using various datasets. In all of our tests, gradient-boosted trees performed the best among classical models. Ultimately, only a closed-loop implementation during live experimental operation that is subject to the associated unforeseeable circumstances can provide definitive evidence of the merits of a predictive method—and may also lead to additional insights through the process of implementation and debugging in the live plasma control system. However, the large and representative archival datasets used here cover a wide range of operational scenarios, and thus provide substantial evidence as to the relative strengths of the methods considered.

For the DIII-D dataset, we sample both the training and the testing examples uniformly across all experimental runs ('shots'). Thus, this dataset requires the least 'generalization' (the ability of the algorithm to learn patterns during training that transfer to new and possibly unseen situations, in this case the testing set). In this setting, classical methods and our proposed method are competitive, with the classical method performing slightly better. However, FRNN improves further in performance after the 30-millisecond deadline (providing improved predictive performance if mitigation technology becomes faster in the future),

#### Table 1 | Prediction results

	Single machine		Cross-machine		Cross-machine with 'glimpse'
Training set	DIII-D	JET (CW)	JET (CW)	DIII-D	$DIII\text{-}D+\delta$
Testing set	DIII-D	JET (ILW)	DIII-D	JET (ILW)	JET (ILW) – $\delta$
Best classical model	0.937	0.893	0.636	0.616	0.851
FRNN OD	0.890	0.952	0.761	0.817	0.879
FRNN 1D	0.922	-	-	0.836	0.911

Performance of the best models on test datasets, measured as AUCs at 30 ms before a disruption. We compare FRNN with (1D) and without (0D) profile information and the best classical approach. The best model for each data set is shown in bold. The last column shows results for cross-machine testing with a small amount (a 'glimpse') of data,  $\delta$ , from the testing machine added to the training set (see text). A score of 1.0 represents perfect performance and 0.5 is equivalent to random guessing. Because the relevant diagnostic for 1D profiles was not available on most JET shots from the carbon wall dataset, 1D profiles are not included when training on JET data. CW, carbon wall.

performs as well as the classical method in the 'interesting'<sup>3</sup> region of the ROC curve with high true positives and low false positives, and provides better generalization for threshold choices (Extended Data Fig. 1).

For the JET dataset, training and testing data are drawn from slightly different distributions. The testing set is drawn after an upgrade to the device, in which the internal wall was changed from a carbon wall to an ITER-like wall (ILW) made of beryllium<sup>13</sup>, resulting in different physical boundary conditions as well as different shot and operations characteristics<sup>10</sup>. Here the superior generalization abilities of FRNN become clear.

Being able to learn generalizable disruption-relevant features from one tokamak and apply them to another will be key to a disruption predictor for ITER, where no extensive disruption campaigns can be executed to generate training data. The second and third columns of Table 1 show the results for cross-machine performance, where both training and validation data come from one machine, and testing is performed on the other. This is a difficult task, complicated by various subtle factors (see the Supplementary Information section 'Challenges in cross-machine training'), which has presented challenges to earlier work<sup>11</sup>. The results show that, in this setting, only our deep-learning approach is able to transfer substantial generalizable knowledge from one machine to the other. The results are particularly strong for the ITER-relevant case of training on a machine with smaller physical size and less stored energy (DIII-D) and generalizing to a 'big' unseen machine (JET). As far as we are aware, this is the first demonstration of substantial cross-machine generalization for machine-learning-based disruption prediction.

Although it is not possible to obtain thousands of training shots (including a sufficient number of disruptions) from a new machine such as ITER, a small amount of simulated or real (perhaps low-power or low-current) disruptive shots<sup>3</sup> may be feasible. To simulate this scenario, we sample a small set,  $\delta$ , of shots from the testing set on the big machine (JET), and give the algorithms access to these during training (see the Methods subsection 'Experimenting with a small number of shots from the test machine'). Encouragingly, all models greatly benefit from this 'glimpse' at the testing set (see the last column of Table 1). Generalization is particularly strong for the deep-learning model. Using only a very few JET shots, FRNN can reach a performance that is competitive with that of models trained on the full JET dataset using the same restricted set of signals available on both machines. These results are highly relevant to disruption prediction on the ITER, as they demonstrate the feasibility of training well-performing models without the need for many disruptive training shots from the target machine.

Given that manual dimensionality reduction and feature engineering (that is, the extraction of useful low-dimensional summaries or representations from high-dimensional data<sup>26</sup>) would first be necessary, classical methods have been unable to take advantage of higher-dimensional signals such as profiles. Profiles are one-dimensional data that capture the dependence of a relevant plasma parameter, such as the electron temperature or density, on the radius as measured from the plasma core to the edge. This radial dependence is generally the most important degree of freedom, as variations along the poloidal or toroidal degrees of freedom are subject to much greater particle mobility and



**Fig. 2** | **Example predictions on real shots from DIII-D and JET. a**, **c**, Shots from DIII-D; **b**, shot from JET. For each shot, the top two panels show scalar signals; the next two show profile signals; and the bottom panel shows the model output as a function of time. T = 0 is defined as the first time point for which all signals are present in the database, which can differ from the standard DIII-D and JET time base. Only a representative subset of the signals used by the algorithm is plotted, and each signal is shown in its normalized form (see the Methods subsection 'Normalization' for details and Extended Data Table 1 for descriptions of each signal). The red stars and dashed vertical lines indicate alarms. Disruptive shots (**b**, **c**) have a vertical red line at the 30 ms deadline before the disruption.

**a**, DIII-D shot 148,778: a false alarm is triggered about 5,200 ms into the shot by a minor disruption. Careful inspection reveals two separate minor disruptions in close succession, corresponding to the spikes in the output and the resulting alarms. **b**, JET shot 83,413: the slow rise in radiated power allows our deep-learning approach (FRNN1D; black) to correctly predict the disruption hundreds of milliseconds in advance; this is missed by the best classical model (yellow; see text). **c**, DIII-D shot 159,593, only the deep-learning model with access to profile information (black) can correctly predict the oncoming disruption; it is missed by the model that is trained solely on scalar signals (yellow).



**Fig. 3** | **High-performance computing results. a**, Accumulated fraction of detected disruptions (main image) and AUC values achieved (inset) on DIII-D as a function of time to disruption for two models, optimized for performance at deadlines of 30 ms and 1,000 ms before the disruption, respectively. **b**, Time required to complete one pass over the dataset (one 'epoch') during training versus the number of GPUs engaged. Experimental data are compared with a semi-empirical theoretical scaling model (see Supplementary Information section, section 'Derivation of scaling model') and ideal scaling. The relative errors (measured as empirical standard deviations) of the experimental data are  $\pm 2.5\%$  and are much smaller than the size of the circular symbols. The inset shows actual training progress, measured via mean training loss (that is, the difference between the target and the realized output of the model; decreasing curves) and validation AUCs (increasing curves) for various numbers of GPUs ( $N_{GPU}$ ) as a function of scaled execution time

resulting faster averaging times owing to the structure of the confining magnetic fields14. Profiles could provide rich new physics information and insight, and many reaction metrics and control mechanisms already relate to their temporal evolution<sup>27</sup> (see the Supplementary Information section 'Extensions and future work'). Although profile data show large differences between machines, and are at present of limited quality and temporal availability (see the Methods subsection 'Data challenges'), our algorithm is nonetheless able to benefit from these data and to generalize between machines. Performance of the best deep-learning models (including performance on cross-machine prediction) increases universally when including profiles (see Table 1). This demonstrates that there is a wealth of predictive, disruption-relevant information contained in multimodal, high-dimensional data-a critical fusion physics insight. These findings are further corroborated by explicit analyses of signal importance (Extended Data Fig. 2). The ability of our deep-learning model to take advantage of these new physics data without resorting to the use of hand-tuned features or invoking

(execution time ×  $N_{GPU}$ ). The best validation AUC value is denoted by a star. The 256-GPU run shows some initial indications that the pattern of convergence is changing, while still giving final testing AUC as good as the other runs. **c**, Results for hyperparameter tuning with 10<sup>4</sup> GPUs with parallel random search across 100 models, trained on 100 GPUs each. In the main image, the time to solution for finding a model of a given validation AUC in this scenario of engaging 10<sup>4</sup> GPUs is compared to the case of using only a single GPU for the same search. The solid line shows the actual ratio between the times to solution, while the dashed line indicates the ideal ratio (speed-up) of 10<sup>4</sup>. The inset shows the time required for finding the best model when using a total of 1, 10<sup>2</sup> or 10<sup>4</sup> GPUs. For 10<sup>2</sup> GPUs, we distinguish between training the 100 models serially but using 10<sup>2</sup> GPUs for each model (parallel training), and running 100 models in parallel, trained on 1 GPU each (parallel tuning), which both achieve an acceleration (speed-up) of nearly 100 times.

human expertise is key. Higher-quality, more densely available, and potentially even higher-dimensional signals—such as two-dimensional electron cyclotron emission imaging (ECEi) data<sup>28</sup> (see the Methods subsection 'Data challenges' for more examples)—will add even more predictive power to deep-learning models and might lead to new physics insights in the future (see the Supplementary Information section 'Extensions and future work').

Figure 2 shows time series of various example shots and the resulting algorithmic predictions. In Fig. 2a, an example false alarm is triggered on a DIII-D shot at about 5,200 milliseconds into the shot. However, this false positive remains a plausible prediction, as the observed symptoms are consistent with a 'minor disruption' (see the Supplementary Information section 'Disruptions')—an event characterized by a thermal quench (that is, a rapid loss of thermal energy to the plasma-facing components) without a current quench (a loss in plasma current)<sup>3</sup>. Accompanied by only minor disturbances in the plasma current, this is evident in, first, the drop in  $\beta$  (the ratio of thermal to magnetic pressure

in the plasma); second, the peaking and rapid change in the temperature (and density) profiles; and third, the spiking locked-mode.

The fact that false alarms are often understandable like this, and intuitively 'make sense', gives confidence and physical interpretability to the model. By serving as a reliable measure of 'disruptivity' as exemplified in this shot, FRNN could serve as an analysis tool to filter databases and help identify causes, precursors and other events relevant to disruption physics<sup>2,29</sup>, thus supporting discovery science in this area. As is visible from the random spikes in Fig. 2b, we find qualitatively that false alarms from the classical methods are often erratic and not as easily attributed to physically meaningful events.

Figure 2b shows an example of a disruptive shot that is missed by the best performing classical algorithm (gradient-boosted trees) but is correctly caught by our method. Although no sudden events occur near the disruption at the end of this shot, FRNN does pick up on the slow (roughly 1,000 millisecond) rise of the core-radiated power, while gradient-boosted trees do not. This is probably because of its lack of access to temporal information (see the Methods subsection 'Training for classical models').

Figure 2c compares FRNN that is trained (and tested) only on scalar signals (yellow) with a model that is trained on all signals, including profiles (black), from a disruptive DIII-D shot. As can be seen from the drop in  $\beta$ , the morphological change in the profiles and the locked-mode spikes (as well as the later spikes in radiated power), starting at around 3,350 milliseconds, some events are clearly taking place in the plasma that resulted in a disruption. However, only the model that is trained using profiles is able to correctly interpret the early warning signs. Access to one-dimensional profile information qualitatively changes the prediction and allows early detection of the disruption, which is missed by the model without access to profiles.

Optimizing a modern machine-learning model is an iterative process. Selection of well-performing hyperparameters—that is, parameters of the model that are not optimized during training and need to be set manually, such as the learning rate—requires searching a highdimensional space (see Extended Data Table 3 for a comprehensive list of hyperparameters and values that performed well). Evaluating any point in this space entails running full model training and inference. To make this approach practical, it is essential to reduce the time required to train a single model, and to increase the number of models that can be trained in parallel in a given amount of time. Growing model sizes, datasets, and amounts of one- or even higher-dimensional data will only make these demands more challenging.

We address these issues with three levels of parallelism, which together enable the engagement of high-performance computing (HPC) at the largest scale in order to reduce the time to solution. First, graphical processing unit (GPU) computing accelerates training over single-machine, multicore central processing unit (CPU) execution by roughly 10 to 20 times. Using the message passing interface (MPI) standard, we next implement a distributed, synchronous, data-parallel training approach<sup>30</sup> to engage large numbers of GPUs at once. Finally, we parallelize the random hyperparameter search by training many such distributed multi-GPU models in parallel.

An important application of hyperparameter tuning is the ability to tune models for a specific task, such as providing much earlier disruption warnings, thus possibly enabling active plasma control without the need for shutdown<sup>3</sup>. In Fig. 3a we show the results of using hyperparameter tuning to select models for optimal prediction performance at 30 and 1,000 milliseconds, respectively, before the disruption. The tuned models display qualitatively distinct behaviour, which generalizes to the testing set: the model tuned for 30 milliseconds shows better performance closer to the disruptions, while the model tuned at 1,000 milliseconds shows superior performance at times further away.

Figure 3b shows the excellent strong scaling of FRNN's data-parallel training up to at least 6,000 GPUs using the Oak Ridge Leadership Computing Facility (OLCF) supercomputer Titan. We have replicated this scaling on the Pascal-P100-powered TSUBAME 3.0 and Volta-powered OLCF Summit supercomputers, as well as with mixed

floating-point compute precisions<sup>31</sup>. In the inset of Fig. 3b, we study training progress as a function of execution time multiplied by the number of GPUs. The fact that the curves approximately collapse indicates that actually training a model to convergence also scales nearly ideally with the number of GPUs used.

Figure 3c shows the results of hyperparameter tuning runs on 100 parallel random models, each trained with 100 GPUs, engaging a total of 10<sup>4</sup> GPUs. We compare this performance to scenarios in which only 1 or 100 GPUs are engaged to perform the same search. The black curve compares the time required for the search when using  $10^4$  GPUs to that when using a single GPU, as a function of the AUCs of the models found. Parallel search becomes increasingly effective for higher AUC values, because those values occur more rarely. The inset shows near-perfect acceleration (speed-up) in finding the best model when using  $10^2$  or  $10^4$  GPUs, demonstrating effective engagement of supercomputing systems comprising  $O(10^4)$  GPUs—the scale of the largest supercomputers available today<sup>32</sup>—and a resulting overall time-to-solution of only half an hour.

Ultimately, the goal will be not just to mitigate disruptions but to avoid them entirely if possible. Models that learn a salient representation of the state of the reactor—such as the method presented here could lie at the core of a deep reinforcement learning<sup>33</sup> approach. Using training reactors or simulated data with synthetic diagnostics<sup>34</sup>, these models could be trained to directly control the reactor while minimizing disruptivity and also optimizing arbitrary objectives such as fusion power output. This also highlights the potential for synergy between machine learning and more traditional modelling and simulation efforts.

Using the example of predicting disruptions in fusion reactors, our paper highlights the potential of deep learning to complement theory, simulations and experiments in the analysis, prediction and control of highly complex physical systems. With the rapidly growing availability of multimodal and high-dimensional data across several disciplines, our findings—as well as some of the associated challenges and insights—have clear implications for the applicability of deep learning to fusion science.

#### **Online content**

Any methods, additional references, Nature Research reporting summaries, source data, statements of data availability and associated accession codes are available at https://doi.org/10.1038/s41586-019-1116-4.

Received: 16 May 2018; Accepted: 30 January 2019; Published online 17 April 2019.

- Mote, C. Jr, Dowling, A. & Zhou, J. The power of an idea: the international impacts of the grand challenges for engineering. *Engineering* 2, 4–7 (2016).
- Schuller, F. Disruptions in tokamaks. Plasma Phys. Contr. Fusion 37, A135 (1995).
- De Vries, P. et al. Requirements for triggering the ITER disruption mitigation system. Fus. Sci. Technol. 69, 471–484 (2016).
- Lehnen, M. et al. Disruptions in ITER and strategies for their control and mitigation. J. Nucl. Mater. 463, 39–48 (2015).
- Tang, W. et al. Scientific grand challenges: fusion energy science and the role of computing at the extreme scale (US Department of Energy's Office of Fusion Energy Sciences, Workshop March 18–20, Washington DC, 2009).
- Aymar, R., Barabaschi, P. & Shimomura, Y. The ITER design. *Plasma Phys. Contr. Fusion* 44, 519 (2002).
- Wroblewski, D., Jahns, G. & Leuer, J. Tokamak disruption alarm based on a neural network model of the high-beta limit. *Nucl. Fusion* 37, 725 (1997).
- Cannas, B., Fanni, A., Marongiu, E. & Sonato, P. Disruption forecasting at JET using neural networks. *Nucl. Fusion* 44, 68 (2004).
- Murari, A. et al. Prototype of an adaptive disruption predictor for JET based on fuzzy logic and regression trees. *Nucl. Fusion* 48, 035010 (2008).
- Vega, J. et al. Results of the JET real-time disruption predictor in the ITER-like wall campaigns. *Fusion Eng. Des.* 88, 1228–1231 (2013).
- 11. Windsor, C. et al. A cross-tokamak neural network disruption predictor for the JET and ASDEX upgrade tokamaks. *Nucl. Fusion* **45**, 337 (2005).
- Luxon, J. L. A design retrospective of the DIII-D tokamak. Nucl. Fusion 42, 614 (2002).
- Matthews, G. et al. JET ITER-like wall—overview and experimental programme. Phys. Scr. 2011, 014001 (2011).
- Freidberg, J. P. Plasma Physics and Fusion Energy (Cambridge Univ. Press, 2008).



- Taylor, P. et al. Disruption mitigation studies in DIII-D. *Phys. Plasmas* 6, 1872–1879 (1999).
- Tang, W. M. & Chan, V. Advances and challenges in computational plasma science. *Plasma Phys. Contr. Fusion* 47, R1 (2005).
- 17. De Vries, P. et al. Survey of disruption causes at JET. *Nucl. Fusion* **51**, 053018 (2011).
- Carleo, G. & Troyer, M. Solving the quantum many-body problem with artificial neural networks. Science 355, 602–606 (2017).
- Carrasquilla, J. & Melko, R. G. Machine learning phases of matter. Nat. Phys. 13, 431–434 (2017).
- Rattá, G. et al. Feature extraction for improved disruption prediction analysis at JET. Rev. Sci. Instr. 79, 10F328 (2008).
- Rattá, G. et al. Improved feature selection based on genetic algorithms for real time disruption prediction on JET. Fusion Eng. Design 87, 1670–1678 (2012).
- LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* 521, 436–444 (2015).
   Bradley, A. P. The use of the area under the ROC curve in the evaluation of
- machine learning algorithms. *Pattern Recognit.* **30**, 1145–1159 (1997). 24. Liaw, A. et al. Classification and regression by randomForest. *R News* **2**, 18–22
- Llaw, A. et al. Classification and regression by random rorest. *R News* 2, 18–22 (2002).
   Chen, T. & Guestrin, C. XGoost: a scalable tree boosting system. In *Proc. 22nd*
- Chen, I. & Guestrin, C. XGoost: a scalable tree boosting system. In Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 785–794 (ACM, 2016).
- 26. Chollet, F. Deep Learning With Python (Manning Publications, 2018).
- Barton, J. E., Wehner, W. P., Schuster, E., Felici, F. & Sauter, O. Simultaneous closed-loop control of the current profile and the electron temperature profile in the TCV tokamak. In *American Control Conference (ACC)* 3316–3321 (IEEE, 2015).
- Tobias, B. et al. Commissioning of electron cyclotron emission imaging instrument on the DIII-D tokamak and first data. *Rev. Sci. Instr.* 81, 10D928 (2010).
- De Vries, P., Johnson, M., Segui, I. & Contributors, J. E. Statistical analysis of disruptions in JET. Nucl. Fusion 49, 055011 (2009).
- 30. Goyal, P. et al. Accurate, large minibatch SGD: training ImageNet in 1 hour. Preprint at https://arxiv.org/abs/1706.02677 (2017).
- Svyatkovskiy, A., Kates-Harbeck, J. & Tang, W. Training distributed deep recurrent neural networks with mixed precision on GPU clusters. In Proc. Machine Learning on HPC Environments 10 (ACM, 2017).
- Top500 supercomputers. Available at https://www.top500.org/lists/2017/11/ (2018/01/11).
- Mnih, V. et al. Human-level control through deep reinforcement learning. Nature 518, 529–523 (2015).
- 34. Coelho, R. et al. Synthetic diagnostics in the European Union integrated
- tokamak modelling simulation platform. *Fus. Sci. Technol.* 63, 1–8 (2013).
  35. Litaudon, X. et al. Overview of the JET results in support to ITER. *Nucl. Fusion* 57, 102001 (2017).

Acknowledgements We are grateful to E. Feibush from the US Department of Energy (DOE) Princeton Plasma Physics Laboratory (PPPL) and the Princeton Institute for Computational Science and Engineering (PICSciE) for assisting with visualization and data collection; to W. Wichser, C. Hillegas, J. Wells, S. Matsuoka, R. Yokota and T. Gibbs for supporting our supercomputing efficiency to T. Donne for facilitating collaborations with JET; to E. Joffrin, R. Buttery and T. Strait for leading the internal reviews of this work at JET and DIII-D; to A. Murari, J. Vega and the associated JET data analysis team for discussions of their classical machine-learning methods; and to M. Maslov for support with the JET data. We also thank R. Nazikian, N. Logan, M. Parsons and M. Churchill of PPPL; K. Felker of Princeton University; R. Granetz and C. Rea of the Massachusetts Institute of Technology (MIT); and P. DeVries of ITER for support and for discussions. We thank the JET contributors<sup>35</sup> and management as well as General Atomics (GA) and its DIII-D tokamak project for access to their fusion databases. J.K.-H. was supported by the DOE Computational Science Graduate Fellowship Program of the Office of Science and National Nuclear Security Administration in the DOE under contract DE-FG02-97ER25308. A.S. is supported by PICSciE, and W.T. by PPPL and PICSciE. This work was carried out within the framework of the EUROfusion Consortium, with funding from the Euratom research and training programme 2014-2018 under grant 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission. This material is based upon work supported by the US DOE, Office of Science, Office of Fusion Energy Sciences, using the DIII-D National Fusion Facility, a DOE Office of Science user facility, under award DE-FC02-04ER54698 Disclaimer: this report was prepared as an account of work sponsored by an agency of the US Government. Neither the US Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the US Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the US Government or any agency thereof.

**Reviewer information** *Nature* thanks Ned R. Sauthoff and the other anonymous reviewer(s) for their contribution to the peer review of this work.

Author contributions J.K.-H. conceived the idea, wrote the code including the HPC and MPI features, curated the datasets, ran and analysed computational experiments, generated the theoretical scaling analysis, produced the figures, and wrote the manuscript. A.S. extended and co-authored the code base and ran computational experiments, including initial deployment of the code on supercomputers. W.T. supervised and supported the implementation of the project at all stages, and initiated collaborations with JET and leading supercomputing facilities. All authors contributed to editing the manuscript.

Competing interests The authors declare no competing interests.

#### Additional information

**Extended data** is available for this paper at https://doi.org/10.1038/s41586-019-1116-4.

Supplementary information is available for this paper at https://doi.org/ 10.1038/s41586-019-1116-4.

Reprints and permissions information is available at http://www.nature.com/ reprints.

**Correspondence and requests for materials** should be addressed to J.K.-H. **Publisher's note:** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© The Author(s), under exclusive licence to Springer Nature Limited 2019

#### **METHODS**

**Data considerations.** *Data and preprocessing.* The data for individual experimental runs (or 'shots') are stored as separate time traces for every signal, with sampling periods of between approximately  $1 \times 10^{-5}$  and  $1 \times 10^{-1}$  seconds. For each shot, we read in all relevant signals, and cut the signals to the range of times during which all signals contain data. We then resample the signals to a common sampling rate of 1 ms using causal information (that is, for any given time we always use the last known value before the time in question).

Each time step contains a vector of *n* signals (see Extended Data Table 1). For multidimensional signals, their values are simply concatenated onto the global input vector. A single shot then contains  $n \times T$  scalar values, where *T* is the length of the shot. The full dataset includes several thousand shots from both the JET and the DIII-D tokamaks. Only shots that have data for all signals are included. See Extended Data Table 2 for a summary of the full dataset. Overall, the size of our dataset from DIII-D and JET amounts to about 2 TB—comparable with some of the largest published machine-learning datasets<sup>36</sup>.

*Data challenges.* A fusion plasma is a complex dynamical system with an unknown internal state which evolves according to physical principles and emits a time series of observable data<sup>14</sup>. Capturing the history and present physical state of the plasma should allow predictions about its future behaviour, including the possibility of disruption. Noisy and incomplete data make this a challenging statistical task.

Observable data are captured as scalars and 1D profiles by various passive diagnostics, such as magnetic measurements, electrical probes, visible and ultraviolet spectroscopy, bolometry, electron cyclotron emission (ECE) and X-ray measurements, as well as active diagnostics such as Thomson scattering, light detection and ranging (LIDAR), interferometry, or diagnostic neutral beams<sup>37</sup>. Future work may also consider higher-dimensional sources of data such as such as 2D ECEi imaging<sup>28</sup>, 2D magnetic equilibria<sup>38</sup>, or fast camera data<sup>39</sup>.

Raw experimental data are difficult to work with directly using machinelearning methods. For instance, the relevant physical timescales and experimental sampling frequencies of the different signals span several orders of magnitude. While many dynamic variables in the plasma change within milliseconds or faster, each shot can last anywhere from roughly 1 to 40 seconds. We choose a time step of 1 ms to resolve the fastest relevant dynamics without including excessive data. This sampling results in training examples with sequence lengths of order  $O(10^4)$ .

For each shot, if there is a disruption, this only occurs at the end. Moreover, depending on the machine, disruptions can be quite rare (less than 10% of shots). This means that the actual learning signal for disruption events is quite sparse. We used up-weighting<sup>40</sup> of positive examples (see the hyperparameter  $\lambda$  in Extended Data Table 3) to stabilize training and found that it was often able to increase performance.

Signals are often noisy or exist only partially. We use only shots that have at least some data for every desired signal. However, in contrast with past work, we do not exclude any shots that are based on 'bad' or statistically unusual data. Some signals in the experimental databases are computed using noncausal information (for example, temporal averaging with a time-centred window, usually with a width of about 20 ms). We shift such signals in time to ensure that the algorithm does not have access to any future information at any given time. This approach means that for some signals the algorithm is seeing slightly 'old' data, giving a conservative estimate for prediction performance.

Some signals are not stored consistently in the database. For instance, the input power signal on DIII-D changed its units from MW to kW around shot 156,000. This was not corrected for during our analysis. Because the algorithm divides all shots by the same numerical scale, shots before this change incorrectly appear to the machine-learning algorithms to have a very low value of input power. Thus, the signal importance of the input power on DIII-D is probably underestimated in Extended Data Fig. 2.

Profile data available at present are of limited quality and temporal resolution. Profiles are available at best every 20 ms for DIII-D and every 50 ms for JET, and are often poorly reconstructed or missing entirely. They are also shifted in time to accommodate for noncausal filtering in the EFIT equilibrium reconstruction. Additionally, the data are qualitatively different between machines, consisting of noisy raw data on JET and smooth fitted functions on DIII-D.

The shots in more recent JET ILW campaigns (after shot 84,000 or thereabouts) are run at higher power and plasma current, have higher disruption rates, and are often affected by active DMSs<sup>10,41</sup>. Many shots are terminated by the DMS long before the onset of a disruption. In such cases, it is impossible to know whether any disruption would have actually occurred. Training on affected shots is challenging, as the ground truth disruption signal is hidden by the 'competing risk' of the mitigation action, which also obscures physics signals very close to the disruption. Moreover, there may be a systematic bias in terms of which shots are affected by the DMS. This makes a fair assessment against data without such terminations impossible. Although such data are thus not directly comparable with the other datasets considered here, we have nonetheless tested our method

on the later JET campaigns, in order to determine its ability to handle these more 'high-performance' plasmas. We restricted the disruptive shots to unmitigated and unintentional disruptions. The resulting 'late' JET ILW dataset (as opposed to the 'early' ILW campaigns considered in the main text) and the associated performance values are described in Extended Data Tables 4, 5. We find that this large dataset seems to be more difficult to classify overall, leading to slightly lower AUC values throughout compared with when testing is performed on the earlier ILW data. However, consistent with the results presented in the main text, the deep-learning approach again shows strong predictive capabilities and generalizes better from the JET CW and DIII-D training data to the ILW testing data than do classical approaches. The large size of the dataset also allowed us to both train and test models on random subsets of the late ILW data (with a split of 50% training, 25% validation and 25% testing data; the same split was used for the DIII-D data in the main text). The results demonstrate again that in this setting, where training and testing sets come from the same distribution-consistent with the DIII-D results in the main text-all methods show strong predictive capabilities and the classical methods perform essentially as well as the deep-learning approach.

The computer-science community has established a strong example of providing unified, open datasets (for example, ImageNet, IMDB, Penn Treebank, and so on)<sup>35,42,43</sup> against which new machine-learning methods can be tested. This allows a direct and fair comparison between various methods and leads to measurable incremental progress. In practice, the separation and complexity of the various international experimental facilities make the construction of such unified databases more challenging for the fusion community. Thus, most data currently exist in separately managed databases. We have taken the approach of implementing not only our own method but also a generalized interface that allows a user to plug and play any machine-learning algorithm that adheres to a 'train' and 'predict' application programming interface (API). This allows direct comparison and benchmarking between variants of our RNN approach and other machine-learning methods, including the state of the art as used in past publications, such as SVMs and MLPs, as well as recently popular classical methods such as random forests or gradient-boosted trees<sup>24,25</sup>. We believe that the continual development of a wide variety of methods and a direct comparison on exactly the same data is key for accurately measuring progress and for allowing detailed and transparent comparisons of the relative strengths and weaknesses of all methods. To simplify database access once permission has been obtained, we have included in the code base<sup>44</sup> object-oriented code based on human readable signal names, which fetches raw data from the appropriate original databases and performs error checking; this is key for generating training datasets reliably and at scale.

We have found empirically that absolute predictive performance can be quite sensitive to ad hoc choices about the dataset, such as the precise group of shots that are used (and which ones are excluded because of bad or abnormal data, intentional disruptions, or other criteria) and which signals are used. In our approach, we use all shots from a given time period. We exclude a shot only if, for any of our desired signals, it does not contain data at all. This means that our dataset includes shots with known bad data, intentional disruptions, testing shots, and so on. Although this can hurt performance, it is the approach that is most conservative, the least ad hoc, and the most representative of live, closed-loop operation. Overall, improved handling of these data issues may raise absolute performance beyond the levels reported here. Thus, although absolute performance numbers are important and will be key for the application of disruption prediction to ITER, we also invite the reader to pay particular attention to the relative performance of different methods, as these highlight their relative strengths and weaknesses.

Algorithm and training details. Training the neural network effectively requires overcoming several unique challenges, such as the need for generalizable signal normalization, poorly defined target functions not directly related to the ultimate learning objective (high area under the ROC curve), and a need for stateful training<sup>26</sup> on very long ( $O(10^4)$ ) sequences of varying length. In this section we describe our approach to overcoming these challenges in our training procedure. We also provide a comprehensive list of tunable hyperparameters for our model in Extended Data Table 3. All deep-learning models were implemented using Keras<sup>45</sup> and Tensorflow<sup>46</sup>.

*Normalization.* Neural networks typically expect their inputs to lie in similar numerical ranges across all dimensions. Moreover, they expect a signal of equal amplitude to have equal meaning across examples. This poses a substantial challenge in the use of raw physical signals as inputs to any neural network architecture. Because the raw signals have values in the range  $10^{-6}$  to  $10^{19}$ , the signals must be normalized such that they all lie around  $10^0$ . Moreover, many signals (such as the plasma current, the stored energy, or even the timescale itself) will have differing characteristic scales on different tokamak machines. The normalization should ideally have the property that signals that have the same 'physical meaning' from different machines are mapped to the same numerical value after normalization. As suggested previously<sup>11</sup>, physically motivated dimensionless combinations of the raw measurements are a sensible option for generating such input data.

However, we find empirically (the particular normalization scheme used is in essence a tunable hyperparameter of the model, just like any other) that the best-performing method is to simply normalize each signal by its 'global numerical scale' across the entire dataset. This automatically brings signals to a reasonable numerical range and scales appropriately to different tokamak devices. Thus, the 'normalized form' of each signal (which is how signals are plotted in Fig. 2 and how the actual algorithm receives them) is simply the original signal value divided by this global numerical scale, which is computed as follows. For each shot, we compute the standard deviation of a single signal across that shot (multidimensional signals are counted as one signal, because gradient information is important in such signals and would be distorted if each channel were normalized individually). Then we define the 'global numerical scale' of that signal as the median across all shots of those per-shot standard deviations. Given that a small fraction of shots contains strong outlier data points that lie orders of magnitude outside of their typical range (which could distort the computation of the standard deviation), the median provides a resilient way of obtaining aggregate scale information from all shots. No shots are removed or filtered out from the datasets for having outlying or unusual data. To further ensure that outliers do not deteriorate performance, we also clip each signal to lie within  $(-100\sigma, +100\sigma)$ , where  $\sigma$  is its corresponding numerical scale, although we find that this does not measurably affect performance.

To make profiles scalable between machines, they are at every time step stored not as a function of real spatial position, but rather as a function of normalized toroidal magnetic flux ( $\rho$ ). In Extended Data Table 1 we give a comprehensive list of signals, including their respective units and global numerical scales.

*Target functions.* The ultimate goal of this learning project is to predict the onset of disruptions. The exact definition of what target function the neural network should learn to approximate is important for the architecture of the model and ultimately for its performance. While ultimately a shot is either disruptive or not (that is, the decision is binary: 0 or 1), the RNN needs to return an output value at every time step. For a nondisruptive shot, the output should clearly always be 0 or 'nondisruptive'. However, in a disruptive shot, the best choice for the 'target output' is less obvious. Shortly before the disruption the output should be 1 or 'disruptive', but this is not necessarily true several seconds before the disruption. It is also unclear which choice for such a target function would ultimately result in the highest possible AUC—the ultimate performance metric that we are trying to optimize.

Our solution defines a parameter  $T_{\text{warning}}$  such that the target function is 1 if the time to disruption is  $T_{\text{D}} - t < T_{\text{warning}}$  and 0 otherwise  $(T_{\text{D}} - t)$  is the time to disruption, where  $T_{\text{D}}$  is the time at which the disruption occurs and t is the current time). The intuition is that the neural network should not be able to know about a disruption more than  $T_{\text{warning}}$  away. Setting  $T_{\text{warning}}$  too high might lead to many false positives, while setting it too low might cause the algorithm to fail to learn 'early warning of disruptions. On JET, for instance, we find empirically that values of  $T_{\text{warning}}$  of around 10 s work best. We also tried predicting  $T_{\text{D}} - t$  or  $log_{10}(T_{\text{D}} - t)$  directly using a regression loss function. The log version performs well for the DIII-D tokamak, but not on JET.

We also implemented a 'max hinge' loss in the hope of more closely approximating the ultimate learning objective: a high ROC area. This loss merely considers the maximum output value across all time steps and penalizes it if it does not cross the threshold in a disruptive sample, or if it does cross the threshold in a nondisruptive sample. The penalty is an *L*1 hinge loss with threshold minus 1 for nondisruptive time steps and threshold plus 1 for time steps within  $T_{\rm warning}$  of a disruption. The intuition is that, in the final evaluation of a shot, only the maximum value of the network matters: either it triggers an alarm or not. Thus, this loss should give a more direct incentive for the network to optimize the area under the ROC curve. In practice, we find that 'max hinge' performs about as well as a standard hinge loss with the same parameters (for the standard hinge loss, the same loss is applied individually for every time step, not just at the time step of maximum output).

A user of a deployed version of this predictive system must define an alarm threshold, such that when the RNN output signal reaches a certain value, an alarm is triggered and thus disruption mitigation actions are engaged. This alarm threshold allows the user to trade off between maximizing true positives and minimizing false positives. A true positive is a true disruption that is correctly caught by the algorithm (that is, an alarm is triggered). A false positive is an alarm that is triggered even though there was not going to be a disruption. We define the true-positive rate as the fraction of real disruptive shots for which the algorithm triggers an alarm before the 30 ms deadline. The false-positive rate is the fraction of nondisruptive shots for which the algorithm triggers an alarm at any point in time. As the alarm threshold is raised (harder to cause alarms), there will be fewer false positives, but also fewer true positives. As the threshold is lowered (easier to cause alarms), there will be more false positives, but also more true positives. By varying the threshold, an ROC curve that plots the true-positve rate versus the false-positive rate (see Extended Eata Fig. 1) is traced out, describing the predictive performance of the algorithm holistically. To capture this overall trade-off, we use the AUC to measure the performance of a given method.

Training on long sequences. The typical duration of shots and the sampling rate imply a length of about  $1 \times 10^4$  samples per shot. We approximate the computation of the gradient of the loss with respect to the model parameters by truncated backpropagation through time<sup>47</sup>. We feed 'chunks' of  $T_{\rm RNN} = 128$  time steps at a time to the RNN. The gradients are then computed over this subsection, the internal states are saved, and then the next chunk is fed to the RNN while using the last internal states from the previous chunk as the initial internal states of the current chunk. This allows the RNN to learn long-term dependencies while truncating the gradient backpropagation through time to  $T_{\rm RNN}$  time steps.

*Mini-batching*. Mini-batching<sup>48</sup> is an important technique for improving GPU performance<sup>49</sup> and accelerating training convergence of deep-learning models. The gradients of the loss with respect to the parameters are computed for several examples in parallel and then averaged. For this to work efficiently, the architecture for the forward and backward pass of each gradient computation needs to be equal for all the examples computed in parallel. This is not possible if different training examples have different lengths. Thus, training on sequences of diverse lengths is a large and open problem for many sequence-based learning tasks<sup>47</sup>, particularly for sequences of vastly differing lengths. The traditional approach of bucketing<sup>47,50</sup> would not work in our case because the sequence length is strongly correlated with whether shots are disruptive or nondisruptive, and thus individual batches would be biased.

We implement a custom solution based on resetting the internal state of individual examples within a mini-batch (Extended Data Fig. 3). Because there is a persistent internal state between successive chunks in time, it is not possible to use more than one chunk from a given shot in a given mini-batch (chunks that are successive in the shot must also be presented to the RNN in successive mini-batches during training such that the internal state can persist correctly).

To train batchwise with a batch size of *M*, we need *M* independent (that is, stemming from different shots) time slices of equal length to feed to the GPU. We do this by maintaining a buffer of M separate shots. At every training step, the first  $T_{\text{RNN}}$  time slices of the buffer are fed as the next batch. The buffer is then shifted by  $T_{\rm RNN}$  time steps. Before adding shots to the buffer, they are cut at the beginning to be a multiple of  $T_{\text{RNN}}$  steps. Every time a shot is finished in the buffer (for example, the light green shot in Extended Data Fig. 3), a new shot is loaded (dark green) and the RNN internal states of the corresponding batch index are reset for training. It is this ability to reset the internal state of select batch indices that allows batchwise training on shots of varying lengths. The internal states of the other batch indices are maintained and only reset when a new shot is begun in their respective index of the buffer. Thus, the internal state persists during learning for the entire length of any given shot. This allows the RNN to learn temporal patterns much longer than the unrolling length  $T_{\rm RNN}$  and potentially as long as the entire shot. The random offsets of the shots against each other and random shuffling of the training set provide a mixture of disruptive and nondisruptive samples for the network at every batch to stabilize training. The fetching of shots and filling of the buffer are performed in a separate computational thread to pipeline neural network training work with data-loading work.

*Hyperparameters.* Overall, the data normalization, training procedure and model architecture produce a large number of hyperparameters that must be tuned in order to maximize predictive performance. These hyperparameters include numerical values such as the learning rate and the number of LSTM layers, but also more abstract categorical variables such as the precise model architecture or the normalization algorithm used for different signals. We summarize these parameters in Extended Data Table 3.

Throughout this work and for each dataset, the 'best' model is found by hyperparameter tuning. This is done by random search in the respective hyperparameter space of each method—that is, by training a number of models with random hyperparameters on the training set and choosing the one with the highest performance on the validation set. Note that the validation set is from the same distribution as the training set, since we assume that a real application would not have access to any data from the testing set at training time. Thus, hyperparameter tuning might not find the truly best model, because the optimization metric is performance on the validation set and not the test set itself. In all of our tests, gradient-boosted trees<sup>25</sup> performed the best among classical models, leading to the results in Fig. 2, Extended Data Fig. 1 and Table 1. All deep-learning models are trained with early stopping using the validation AUC as the metric, with a patience of three epochs<sup>51</sup>. The best-performing models for Table 1 are obtained in this way by using 20 random trials for each method.

*Experimenting with a small number of shots from the test machine.* To simulate the scenario of being able to run a few disruptive shots on the test machine for cross-machine prediction, we remove a set  $\delta$  of shots from the testing set on the 'big' machine (JET) by sampling random shots until a fixed number of five disruptive shots has been sampled. In our experiment,  $\delta$  contains 5 disruptive shots and 16 nondisruptive shots. The training and validation data from the 'small' machine (DIII-D) are augmented with this set  $\delta$  to have both more accurate

training and a better measure of validation performance, and the best crossmachine model is retrained without extra tuning. Moreover, we apply no particular importance-weighting or loss adjustment for these extra shots. It is possible that the positive effect of the additional shots could be even further enhanced by such methods. The numbers reported in Table 1 were generated using this procedure.

We also tested the same scenario by sampling shots chronologically instead of randomly from the testing set for the same hyperparameters. The idea behind this approach is that this may more closely resemble the true distribution of shots that one would have access to during a new campaign on a new machine. We found that this approach did not change the results greatly beyond the generally expected stochastic fluctuations in AUC values of order  $\pm$  0.01 (which are due to random training and parameter initializations). The overall ordering of methods and qualitative range of performance remained the same.

Finally, we also performed tests with numbers of disruptive shots that were different than five. While some stochastic fluctuations are as always expected, we find that performance generally increases monotonically for zero to seven shots, and saturates after about seven disruptive shots. Increasing the number of disruptive shots also improves the fraction of models (given randomly chosen hyperparameters) that converge to strong cross-machine performance during training. Given that the shots used are removed from the testing set on which the method is ultimately evaluated, it is not possible with this approach to make a fair comparison of performance for large numbers of removed shots, as the testing set would become very different.

Training for classical models. Training on large datasets is problematic for classical methods, because training algorithms often do not scale well to HPC environments. SVMs, for example, have a training cost quadratic in the number of examples<sup>52</sup>, which makes very large datasets unfeasible. Additionally, parallel algorithms for training single models across many worker nodes are lacking. We use a similar approach to that of ref.<sup>10</sup> for producing features to train the classical machine-learning models here. At every time step, features are extracted for each signal from a time window comprising the last 32 ms. Given that classical methods cannot learn to automatically extract patterns of various temporal scales from arbitrary sequence lengths, this window size represents a manually tuned tradeoff between detecting long and short temporal patterns that might be relevant for disruption prediction. For each time window of each signal, we compute the mean, maximum and standard deviations, as well as the four parameters of a third-order polynomial fit. Thus, for n signals, we have a 7n-dimensional feature vector at every time step. We then train the models by considering each time step a separate 'training example'. We train on a random subset of 106 such examples to avoid prohibitively long training times. The target value is the same as in the 'hinge' target for the deep-learning model (that is, -1 or 1). We implemented random forests, SVMs with linear and nonlinear kernels, multilayer perceptrons with a single hidden layer, and gradient-boosted trees. All classical machine-learning models are implemented in Scikit-Learn<sup>53</sup>, and we use XGBoost<sup>25</sup> to provide functionality for the gradient-boosted trees.

*Distributed training.* In our code, we use python multiprocessing to parallelize preprocessing, shot loading, downloading and basically all components of the preparation and training pipeline. The vast majority of the computational load, however, occurs during the model training phase. While effective massive-scale parallelization of neural network training is an important open research question<sup>30,54</sup>, the idea of data-parallel training is already being used for the largest and most advanced deep-learning models to date<sup>55</sup>.

Most state-of-the-art industrial algorithms<sup>46,56</sup> use a parameter server approach with centralized communication paradigms. By contrast, our MPI implementation allows us to take advantage of highly optimized divide-and-conquer communication routines with logarithmic scaling in the number of processes. As communication is often the bottleneck in distributed training systems, efficient implementation of this component of the training algorithm is key. We empirically observe a very high ratio of computation to communication time (greater than 90% to 10%) during distributed training, even on hundreds of GPUs.

The distributed training sequence can be described as follows: (1) N models are run with their own copy of the current parameters (W); (2) each computes a gradient step on a different subset (mini-batch) of the data using backpropagation; (3) the gradients are reduced (averaged) using a global reduction, such that every model has a copy of the averaged gradient; (4) each model updates the parameters W using the averaged gradient information; and (5) efficient communication is achieved using a custom MPI implementation.

This effectively amounts to training with a large batch size that is the original batch size,  $N_{\text{examples/batch}}$ , multiplied by the number of workers:  $N_{\text{examples/batch}} \rightarrow N_{\text{worker}} \times N_{\text{examples/batch}}$ . To actually achieve a speed-up for training, we then multiply the learning rate by  $N_{\text{worker}}$ . This means that the algorithm is taking fewer learning steps, but each step is larger in magnitude and has smaller variance (because it is based on more data, owing to the larger batch size).

Our parallelized MPI implementation is also used for massively parallel batchwise inference, which speeds up the computation of validation metrics between training epochs. To run batchwise inference, all shots are padded in the end with zeros to be of the same length. Because information enters the RNN only causally, these paddings do not influence the computation in the earlier sections of the shot and can then simply be cut off to obtain the final shot output.

Scaling studies. The experiments illustrated in Fig. 3b were performed on the Titan supercomputer<sup>57</sup>, and we have replicated these scaling results on both the TSUBAME 3.0 and OLCF Summit supercomputers<sup>58,59</sup>. The hyperparameter tuning experiment described in Fig. 3c, engaging 10<sup>4</sup> GPUs by training 100 models in parallel, each using 100 GPUs, was also conducted on the Titan supercomputer and on the JET dataset. The 1 and 100 GPU scenarios are fictitious, because the time required to actually run these scenarios would have been prohibitively large. The ratio of time required to train a single model using 1 instead of 100 GPUs was estimated using scaling data as in Fig. 3a. Specifically, the estimate is obtained by comparing timings between 4 and 100 GPUs and extrapolating from there down to 1, because 4 is the smallest machine architecture that is equal in configuration to 100 GPUs (since each node has 4 GPUs). The scenario of training the 100 models serially (one at a time) was modelled by considering a large number  $(5 \times 10^3)$ of randomized serial arrangements of the 100 already recorded runs, extracting results (such as the time required to find a model of a certain validation AUC) from each of those fictitious reorderings, and averaging the results over all arrangements.

Figure 3c shows some initial indications that convergence patterns are changing when using 256 GPUs or more. Although it is known that deep neural networks become harder to train to full accuracy with many worker GPUs<sup>30</sup>—which corresponds to very large batch sizes—we expect that with larger models (in terms of trainable parameters), larger datasets and higher-dimensional signals, even greater parallelism than that reported in Fig. 3c and the main text will become practical for single-model training. Moreover, promising recent techniques such as learning-rate warm-up, scaling or cycling<sup>30,60</sup> will probably also extend the practical range of GPUs.

Signal-importance studies. In order to prioritize investments in higher-quality data acquisition, and to gain new scientific/physics insights, it is important to quantify the importance of the various signals to the predictability of disruptions. To this end, we train a model with just a single signal at a time and measure the final prediction performance (Extended Data Fig. 2a). This is then a proxy for the disruption-relevant information contained in the respective signal. We also train a model with all signals but a single signal left out (see Extended Data Fig. 2b for results). By comparing the performance to a model trained on all signals (green), the relative drop in performance is a measure of how important that signal was for the full model. Naturally, a model trained on many signals might incorporate high-order interactions between signals, whose effects are not well measured by either of these two approaches. Moreover, the results are stochastic and vary according to model instantiations (due to random training initialization) and hyperparameters. Thus, these estimates should be seen only as a first-order measure of signal importance. Given that these studies require training and testing several models in parallel, as for hyperparameter tuning, they again can be sped up greatly using HPC.

#### Data availability

The data for this study have restricted access, with permission required from the management of EUROfusion and General Atomics. DIII-D data shown in figures in this paper can be obtained in digital format by following the links at https://fusion.gat.com/global/D3D\_DMP.

#### Code availability

The code used in this work is open source and available from ref. 44.

- Deng, J. et al. Imagenet: a large-scale hierarchical image database. In IEEE Conf. on Computer Vision and Pattern Recognition 248–255 (IEEE, 2009).
- Zavaryaev, V. et al. in *Plasma Diagnostics* (eds Kikuchi, M., Lackner, K. & Tran, M. Q.) Ch. 4, 360–534 (International Atomic Energy Agency, 2012).
- Ferron, J. et al. Real time equilibrium reconstruction for tokamak discharge control. Nucl. Fusion 38, 1055 (1998).
- Alonso, J. et al. Fast visible camera installation and operation in JET. In AIP Conference Proceedings Vol. 988, 185–188 (AIP, 2008).
- Zadrozny, B., Langford, J. & Abe, N. Cost-sensitive learning by costproportionate example weighting. In *Third IEEE International Conference on Data Mining* 435–442 (IEEE, 2003).
- Moreno, R. et al. Disruption prediction on JET during the ILW experimental campaigns. Fus. Sci. Technol. 69, 485–494 (2016).
- Maas, A. L. et al. Learning word vectors for sentiment analysis. In Proc. 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies Vol. 1 142–150 (Association for Computational Linguistics, 2011).



- Marcus, M. P., Marcinkiewicz, M. A. & Santorini, B. Building a large annotated corpus of English: the Penn Treebank. Comput. Linguist. 19, 313–330 (1993).
- Kates-Harbeck, J. & Svyatkovskiy, A. FRNN Codebase. https://github.com/ PPPLDeepLearning/plasma-python (2017).
- 45. Chollet, F. et al. Keras. https://github.com/fchollet/keras (2015).
- Abadi, M. et al. Tensorflow: large-scale machine learning on heterogeneous distributed systems. Preprint at https://arxiv.org/abs/1603.04467 (2016).
- Graves, A. Generating sequences with recurrent neural networks. Preprint at https://arxiv.org/abs/1308.0850 (2013).
- Dean, J. et al. Large scale distributed deep networks. In Proc. 25th Internation Conference on Neural Information Processing Systems, vol. 1 1223–1231 (2012).
- 49. Chetlur, S. et al. cuDNN: efficient primitives for deep learning. Preprint at https://arxiv.org/abs/1410.0759 (2014).
- Khomenko, V., Shyshkov, O., Radyvonenko, O. & Bokhan, K. Accelerating recurrent neural network training using sequence bucketing and multi-GPU data parallelization. In *IEEE First International Conference on Data Stream Mining & Processing (DSMP)* 100–103 (IEEE, 2016).
- 51. Ruder, S. An overview of gradient descent optimization algorithms. Preprint at https://arxiv.org/abs/1609.04747 (2016).
- Chang, C.-C. & Lin, C.-J. LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. 2, 27 (2011).
- Pedregosa, F. et al. Scikit-learn: machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830 (2011).

- Das, D. et al. Distributed deep learning using synchronous stochastic gradient descent. Preprint at https://arxiv.org/abs/1602.06709 (2016).
- 55. Wu, R., Yan, S., Shan, Y., Dang, Q. & Sun, G. Deep image scaling up image recognition. Preprint at https://arxiv.org/abs/1501.02876 (2015).
- Chen, T. et al. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. Preprint at https://arxiv.org/ abs/1512.01274 (2015).
- Titan: advancing the era of accelerated computing. Oak Ridge National Laboratory https://www.olcf.ornl.gov/olcf-resources/compute-systems/titan/ (accessed 2 April 2018).
- Morgan, T. P. Japan keeps accelerating with Tsubame 3.0 Al supercomputer. *The Next Platform* https://www.nextplatform.com/2017/02/17/japan-keepsaccelerating-tsubame-3-0-ai-supercomputer/ (accessed 2 April 2018).
- Summit: Oak Ridge National Laboratory's 200 PetaFlop Supercomputer. Oak Ridge National Laboratory (https://www.olcf.ornl.gov/olcf-resources/ compute-systems/summit/ (accessed 2 April 2018).
- Smith, L. N. Cyclical learning rates for training neural networks. In 2017 IEEE Winter Conf. on Applications of Computer Vision 464–472 (IEEE, 2017).
- Kingma, D. & Ba, J. Adam: a method for stochastic optimization. Preprint at https://arxiv.org/abs/1412.6980 (2014).
- Gentile, C. & Warmuth, M. K. Linear hinge loss and average margin. In Proc. 1998 Conf. on Advances in Neural Information Processing Systems 225–231 (MIT Press, 1999).



Extended Data Fig. 1 | ROC curves from the test set for our model and the best classical model, for DIII-D and JET. a, DIII-D; b, JET. The true positive rate is the fraction of disruptive shots that are labelled disruptive in advance of the 30 ms deadline. The false positive rate is the fraction of nondisruptive shots that are labelled disruptive at any time. The areas under the curves correspond to the values in Table 1. The insets show the fraction of detected disruptions as a function of the time to disruption for an 'optimal' threshold value. On the corresponding ROC curve of the same colour, this optimal threshold defines a point that is indicated by a circle (see main text for details). The inset also shows the 30 ms detection deadline as vertical red line. In **a**, the AUC is slightly higher for the classical method (see Table 1), but FRNN performs equally well in the interesting upper left region of high true positives and low false positives. Moreover, only our approach provides additional detections between 30 ms and 10 ms to the disruption, reacting to the spikes in radiated power that often occur on this timescale before the disruption (see P<sub>rad.core</sub> in Fig. 2c). Thus FRNN could provide improved predictive performance if mitigation technology becomes faster in the future. In addition, a threshold value in practice needs to be selected for calling



alarms. The best threshold value is estimated by optimizing it on the training set, in the hope that it will still perform well on the unseen testing set. We define the 'best' threshold as the value that maximizes the quantity TP – FP, where TP is the true positive rate and FP is the false positive rate. This is equivalent to finding the point on the ROC curve furthest in the 'northwest' direction. For FRNN, the threshold generalizes excellently (black and purple circles). For the classical approach, although the overall ROC curve is encouraging, the threshold estimate is poor (orange square) and far from its ideal position (orange circle). For each method, the fraction of detected disruptions is shown in the inset as a function of time until disruption by using the threshold values corresponding to the circle positions, which for the classical method we determine manually with knowledge of the testing set (to give a conservative and maximally favourable estimate of its performance). Median alarm times are about 500-700 ms on DIII-D and around 1,000 ms on JET. Encouragingly, a majority of disruptions is detected with large warning times of hundreds of milliseconds-sufficient for disruption mitigation (requiring around 30 ms) and key to possible future preventative plasma control without the need for shutdown.



**Extended Data Fig. 2 | Signal-importance studies.** Signals are ordered from top to bottom in decreasing order of importance. Signals are defined in Extended Data Table 1. Models were trained on the DIII-D dataset. **a**, Test set AUC values achieved by models trained on a single signal at a time. The AUC value is representative of how much information is contained in that single signal. For comparison, we also show the performance for a model trained on all signals (green bar). **b**, Test AUC values for a model trained on all signals except the labelled one. In this case, the drop in performance compared with the performance of the model trained on all signals (green bar) is a measure of how important the given signal is for the final model. The exact results for both figures are in general stochastic and vary over hyperparameters and for each new training session, so only general trends should be inferred. It appears consistently that the locked-mode, plasma current, radiated power and *q*<sub>95</sub> signals contain a large amount of disruption-relevant information, similar



to the results of past studies of signal importance on JET<sup>21</sup>. Both panels in particular panel **a**, which measures the information content of a single signal at a time—also confirm that there is a large amount of information in the profile signals. With higher-quality reconstructions, more frequent sampling and better (causal) temporal filtering (to obviate the need to shift the signal in time and thus lose time-sensitive information), they are likely to become even more relevant. This indicates that higher-dimensional data probably contain much useful information that should be considered in the future. Panel **b** also highlights another benefit of deep learning, which is that almost all additional signals increase performance, or at least do not have a substantial negative impact. Signals can thus generally be used without having to worry about confusing the algorithm and reducing performance, and therefore without having to spend much time on signal selection. For other methods, signal selection (for example, removing correlated, noisy or noninformative signals) is key<sup>21</sup>.



# number

**Extended Data Fig. 3** | **Snapshot of the training buffer.** The figure illustrates how data are fed to the RNN for batchwise training with a batch size of *M*. Each horizontal bar represents data from a shot, and different colours indicate different shots. A colour change in a given row means that a new shot starts. At every time step, the leftmost chunk is cut from the

buffer and supplied to the training algorithm, and all shots are shifted to the left. When a shot is finished (as the lighter green bar is about to be), a new shot is loaded into the buffer, and the internal state of the RNN at that batch index is reset. See the Methods subsection 'Mini-batching' for details.

# Extended Data Table 1 $\mid$ Signals considered and availability on the machines

Signal description	Avail. on machines	Symbol	Numerical scale DIII-D	Numerical scale JET
Plasma current	DIII-D, JET	I <sub>p</sub>	$3.81 imes10^{-1}$ MA	$5.03 imes10^{-1}$ MA
Plasma current target	DIII-D	$I_{p, target}$	$3.93 imes10^{-1}$ MA	_
Plasma current error	DIII-D	$I_{p,\mathrm{error}}$	$3.10 imes10^{-2}$ MA	_
Plasma current direction	DIII-D	I <sub>ndirect</sub>	1.0	_
Internal inductance	DIII-D, JET	$I_i$	$2.02 imes10^{-1}$	$1.51 imes10^{-1}$
Electron temperature profile	DIII-D,JET*	$T_e( ho)$	$9.53 imes10^{-1}~ m keV$	1.53 keV
Electron density profile	DIII-D,JET*	$n_e( ho)$	$1.47 imes10^{19}~\mathrm{m}^{-3}$	$2.98 imes10^{19}~\mathrm{m}^{-3}$
Electron density	DIII-D, JET	n <sub>e</sub>	$1.19 imes10^{19}~\mathrm{m}^{-3}$	$4.69 imes10^{19}~\mathrm{m}^{-3}$
Input power	DIII-D , JET	P <sub>in</sub>	$1.85 imes10^{6}~{ m W}$	$4.47 imes10^{6}~{ m W}$
Input beam torque	DIII-D	Pin	1.47 Nm	-
Radiated power (core)	DIII-D, JET	P <sub>rad,COTE</sub>	$4.58 imes10^2 extrm{ W}$	$4.05 imes10^4~{ m W}$
Radiated power (edge)	DIII-D, JET	$P_{rad, edge}$	$4.94 imes10^2~{ m W}$	$2.72 imes10^4~{ m W}$
Radiated power (total)	JET	P <sub>rad</sub>	_	$2.22 imes10^{6}~{ m W}$
Plasma energy	DIII-D, JET	W <sub>MHD</sub>	$2.79 imes10^5~ m J$	$1.20 imes10^{6}~{ m J}$
Safety factor (magnetic field pitch)	DIII-D, JET	<b>q</b> <sub>95</sub>	1.0	1.0
Normalized plasma pressure (ratio	DIII-D	$\beta$	$6.91 imes10^{-3}$	_
of thermal to magnetic pressure)				
Locked mode (nonrotating instabil-	DIII-D, JET	LM	$1.14 imes10^{-6}$ T	$5.72 imes10^{-5}$ T
ity) amplitude				
Normalized toroidal magnetic flux	DIII-D, JET	ρ	1.0	1.0

Electron-density and electron-temperature profiles (denoted by an asterisk) are available only on the more recent JET campaigns (ILW). Each signal is normalized by its numerical scale before feeding to the machine-learning algorithms. The numerical scale is computed as the median of the standard deviations of the signal for all shots (see the Method subsection 'Normalization').



## Extended Data Table 2 | Datasets used here

Machine	Shot range	Total shots	Num. disruptive	Note
JET train	66027 - 79853	2894	215	JET carbon wall campaigns C23-C27b
JET validate	66027 — 79853	1425	87	JET carbon wall campaigns C23-C27b
JET test	81852 — 83793	1191	174	JET ITER-like wall campaigns C28-30
DIII-D train	125500 - 168555	1734	407	DIII-D shots since 2006
DIII-D validate	125500 — 168555	853	197	DIII-D shots since 2006
DIII-D test	125500 — 168555	862	206	DIII-D shots since 2006

Shots were obtained from the respective machines. All shots that contain data for all signals were used. No shots were discarded for bad or abnormal data, or for being known testing shots or intentional disruptions.

Hyperparameter	Explanation	Best value
$\eta$	Learning rate	$1.7 \times 10^{-5}$
$\gamma$	Learning rate decay per epoch	0.985
$\lambda$	Weighting factor for positive examples	16
Ν	Number of recurrent neural network	2
	(RNN) layers	
n <sub>cells</sub>	Number of RNN cells per layer	200
T <sub>RNN</sub>	Number of timesteps through which	128
	backpropagation is run	
RNN type	Type of RNN	Long short term memory
		network (LSTM)
n <sub>batch</sub>	Batch size	128
T <sub>warning</sub>	Time at which the target function for a	10000
	disruptive shot becomes positive in ms	
Target	Type of target function	TTD (Hinge)
Normalizer	Type of normalization scheme	Divide by $\sigma$
dt	timestep	0.001 s
T <sub>min,warn</sub>	minimal acceptable warning time	0.03s
Hide warning	Whether to hide the last $T_{min,warn}$	False
	timesteps during training (these wouldn't	
	be available in a real shot)	
N <sub>F</sub>	Number of convolutional filters	128
Μ	Number of convolutional layers	128
L <sub>filters</sub>	Size of 1D convolutional filters	3
L <sub>pool</sub>	Size of 1D max pooling	2
Optimizer	Stochastic optimization scheme	Adam
Dropout	Dropout probability	0.1
L2 Regularization	Weight regularization of all layers	10 <sup>-2</sup>
Clip norm	Maximum norm of gradients	10

Extended Data Table 3 | Hyperparameters to be optimized, explanations and well-performing values

The Adam optimizer is described in ref.  $^{61}$ . The 'hinge' target is -1 before  $T_{warning}$ , then +1 for disruptive shots. It requires a hinge loss<sup>62</sup>. The normalization scheme referenced in the 'Normalizer' row divides each signal by its global numerical scale across the dataset (see the Methods subsection 'Normalization'). Although all quoted parameters perform well on both tokamaks, the specific values shown here are found by optimizing for validation performance on JET.



## Extended Data Table 4 | Data from the later JET ILW campaigns

Machine	Shot range	Total shots	Num. disruptive	Note
JET	83794 - 92504	5002	369	JET ILW C31-C37

Only nondisruptive shots and unintentional disruptive shots without active mitigation were used. Thus, the shots used contain only a small subset of the total number of disruptive shots (1,952) from these campaigns. Of those considered, all shots that contain data for all signals were used. No shots were discarded for bad or abnormal data.

# Extended Data Table 5 | Prediction results on the late ILW data

	Single machine		Cross-machine	Cross-Machine with "glimpse"
Training Set	JET (CW)	JET (ILW late)	DIII-D	DIII-D + $\delta$
Testing Set	JET (ILW late)	JET (ILW late)	JET (ILW late)	JET (ILW late) $-\delta$
Best classical Model	0.880	0.951	0.483	0.899
FRNN 0D	0.923	0.952	0.793	0.907
FRNN 1D	_	0.956	0.824	0.922

Test set performance of the best models, measured as AUCs at 30 ms before disruption. We compare FRNN with (1D) and without (0D) profile information and the best classical approach. The best model for each dataset is shown in bold. As in Table 1, the last column shows results for cross-machine testing with a small amount ('glimpse') of data,  $\delta$ , from the testing machine added to the training set (see text for details). A score of 1.0 indicates perfect performance and 0.5 is equivalent to random guessing. Because the relevant diagnostic for 1D profiles was not available on most JET shots from the CW dataset, 1D profiles are not included when training on JET CW data.