We have developed SQL databases on 4 tokamaks for training disruption warning algorithms.

<table>
<thead>
<tr>
<th>Machine</th>
<th>Shots</th>
<th>Time slices (records)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-MOD</td>
<td>5507</td>
<td>498925</td>
</tr>
<tr>
<td>EAST</td>
<td>14713</td>
<td>1209217</td>
</tr>
<tr>
<td>DIII-D</td>
<td>10258</td>
<td>2356519</td>
</tr>
<tr>
<td>KSTAR</td>
<td>4219</td>
<td>773083</td>
</tr>
</tbody>
</table>

~50 plasma parameters are recorded at each time slice:

- shot (primary key)
- time (primary key)
- time_until_disrupt
- ip
- lp_error
- dip_dt
- dlpprog_dt
- v_loop
- p_rad
- p_oh
- p_icrf
- p_lh
- p_nbi
- rad_input_frac
- rad_loss_frac
- n_equal_1_mode
- pressure_peaking
- zcur
- z_error
- v_z
- z_times_v_z
- V_0
- v_mid
- v_edge
- beta_n
- beta_p
- dbetap_dt
- kappa
- li
- dli_dt
- dWmhd_dt
- H98
- n_e
- dn_dt
- r_dd
- q95
- q0
- qstar
- lower_gap
- upper_gap
- power_supply_railed
- Greenwald_fraction
- Te_width
- Intentional_disruption
We train our prediction algorithms on a subset of the signals in the databases.

By examining the many signals in our databases we have identified a subset of signals that show a clear change in behavior on *some* disruptions on *some* machines:

<table>
<thead>
<tr>
<th>Signal description</th>
<th>Variable name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percent error between measured and programmed plasma current, ((I_p - I_{prog})/I_p)</td>
<td>ip_error_frac</td>
</tr>
<tr>
<td>Poloidal beta, (\beta_p)</td>
<td>betap</td>
</tr>
<tr>
<td>Greenwald density fraction, (n/n_G)</td>
<td>n/nG</td>
</tr>
<tr>
<td>Safety factor at 95% of minor radius, (q_{95})</td>
<td>q95</td>
</tr>
<tr>
<td>Normalized internal inductance, (\ell_i)</td>
<td>li</td>
</tr>
<tr>
<td>Radiated power fraction, (P_{rad}/P_{input})</td>
<td>prad_frac</td>
</tr>
<tr>
<td>Loop voltage, (V_{loop} ) [V]</td>
<td>Vloop</td>
</tr>
<tr>
<td>Stored plasma energy, (W_{th} ) [J]</td>
<td>Wmhd</td>
</tr>
<tr>
<td>(n = 1) mode amplitude, normalized to (B_{tor})</td>
<td>n_equal_1_normalized</td>
</tr>
<tr>
<td>Electron temperature profile width, normalized to plasma minor radius</td>
<td>Te_width_normalized</td>
</tr>
</tbody>
</table>

We use this subset to train and test our machine learning algorithms.
Disruption precursor behavior is very different on C-Mod, DIII-D, and EAST:

**DIII-D:** $l_i$ starts to increase 400-800 ms before a disruption occurs on a significant fraction of disruptions.

**EAST:** $l_i$ shows almost no change in behavior before a disruption occurs.

**C-Mod:** $l_i$ starts to decrease, but only ~5 ms before a disruption occurs.

Note different time scales
Disruption precursor behavior is very different on C-Mod, DIII-D, and EAST

**DIII-D:** $\ell_i$ starts to increase 400-800 ms before a disruption occurs on a significant fraction of disruptions.

**EAST:** $\ell_i$ shows almost no change in behavior before a disruption occurs.

**C-Mod:** $\ell_i$ starts to decrease, but only $\sim$5 ms before a disruption occurs.

*Note different time scales*
Disruption precursor behavior is very different on C-Mod, DIII-D, and EAST:

**C-Mod**: $V_{\text{loop}}$ does not change until just a couple of milliseconds before disruptions occur.

**DIII-D**: $V_{\text{loop}}$ starts to increase, but not until ~20 ms before disruptions occur.

**EAST**: $V_{\text{loop}}$ starts to increase ~100 ms before a disruption occurs on a significant fraction of disruptions.

*Note different time scales*
Some basic concepts of our application of AI machine learning to disruption prediction

- We formulate our application as a **supervised classification** problem, specifically a **binary classification** problem
  - Every time slice in the database is known a priori to belong to one of only two possible ‘classes’
  - We choose our two classes to be:
    (0) far from disrupt or belongs to a non-disruptive discharge
    (1) close to disrupt

- How to specify close to disrupt and far from disrupt classes?
  - Initially we picked the dividing time, $\tau_{\text{class}}$, by looking at signal behaviors for each machine (as seen on preceding slides):
    - DIII-D: $\tau_{\text{class}} = 350 \text{ ms}$
    - EAST: $\tau_{\text{class}} = 100 \text{ ms}$
    - C-Mod: $\tau_{\text{class}} = 40 \text{ ms}$
  - More recently, we use an optimization procedure (described later)
Most of our effort has focused on an AI Machine Learning method known as Random Forests

Random Forests consist of many independent, uncorrelated decision trees

• Each decision tree tries to divide up the space of plasma physics time slice data into the specified classes, based on objective splitting rules.

There are a number of reasons why Random Forests is an attractive Machine Learning method:

— The architecture of a Random Forest involves only one design parameter (# of trees), which is easily optimized
— Different features (plasma parameters), with vastly different numerical ranges, present no issues
— The degree to which each feature contributes to the classification decision can be easily characterised (“white box”)
Comparison of Random Forest performance on DIII-D, EAST, and C-Mod

Results are for flattop period only, for all shots in each machine’s database

<table>
<thead>
<tr>
<th>Machine</th>
<th>True Negatives</th>
<th>False Positives</th>
<th>True Positives</th>
<th>False Negatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIII-D</td>
<td>202022</td>
<td>544</td>
<td>5310</td>
<td>3601</td>
</tr>
<tr>
<td>EAST</td>
<td>94415</td>
<td>271</td>
<td>528</td>
<td>1843</td>
</tr>
<tr>
<td>C-Mod</td>
<td>43313</td>
<td>98</td>
<td>496</td>
<td>1025</td>
</tr>
</tbody>
</table>

"recall" = TP/(TP+FN) = fraction of "close to disrupt" that are correctly predicted

<table>
<thead>
<tr>
<th>Machine</th>
<th>&quot;recall&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIII-D</td>
<td>59.6 %</td>
</tr>
<tr>
<td>EAST</td>
<td>22.3 %</td>
</tr>
<tr>
<td>C-Mod</td>
<td>32.6 %</td>
</tr>
</tbody>
</table>

Miss rate = 1 - recall = “close to disrupt” that are not caught
False alarm fraction = FP/(TP+FP)

"F1, F2, Fβ scores" = weighted combinations of miss rate and false alarm rate
The performance metrics shown above are based on the individual timeslices.

- There are large differences in prediction success rates for the different tokamaks (~60% for DIII-D, 33% for C-Mod, 22% for EAST).
- The best ‘true positive’ rate (DIII-D) is still only ~60%, which is not good enough for ITER.

Can we do better than this?

- Yes, since we really want to know:
  
  "Will this shot disrupt soon?"

  *not*

  "Is this individual timeslice close to disrupting?"

- *Basically, we want to see a bunch of high-valued timeslices before we declare that a shot will disrupt soon. This reduces problems posed by measurement errors, and algorithm prediction errors.*
Timeslice – by – Timeslice

vs

Shot – by – Shot

Red curve is timeslice-by-timeslice disruptivity prediction

Black curve is an ‘alarm’ signal, which transitions no more than once per shot

The alarm is triggered *only* if the disruptivity surpasses a high threshold and then remains above a low threshold for a specified length of time.
We choose the optimal values for these 4 parameters:

- \( \tau_{\text{class}} \)
  - high threshold
  - alarm window
  - low threshold

by maximizing the metrics, \( F_1 \) or \( F_2 \), which are weighted averages of the true positive rate and the false positive rate.
Optimisation using F2 metric can significantly increase disruption prediction rate for all 3 tokamaks.
But, the improved disruption prediction rate also results in a higher false alarm rate

<table>
<thead>
<tr>
<th>$thr$</th>
<th>$F_γ$ - score</th>
<th>True Positive Rate</th>
<th>False Positive Rate</th>
<th>Average Warning Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>85%</td>
<td>0.57 (F1)</td>
<td>58%</td>
<td>3%</td>
<td>306 ms</td>
</tr>
<tr>
<td>65%</td>
<td>0.66 (F2)</td>
<td>96%</td>
<td>20%</td>
<td>246 ms</td>
</tr>
</tbody>
</table>

Using the $F_2$ metric for optimization of the shot-by-shot analysis (i.e. alarm signal) can improve the true positive rate significantly, but at the expense of a higher false positive rate (more false alarms).

$$F_γ = \frac{TP}{(1 + \gamma^2)TP + \gamma^2 FN + FP}$$

$TP$: True Positives  
$FN$: False Negatives  
$FP$: False Positives
A Random Forests predictor has been installed and run in real time in the DIII-D plasma control system

- Random Forest predictor was trained on all the discharges in the DIII-D database, i.e. **without regard to different types of disruptions**
- Trained on flattop only (i.e. timeslices for which $|\text{d}I_p/\text{d}t| < 1 \text{ kA/s}$)
- Trained on 9 signals that come to PCS in real time
- Algorithm was converted into C language to run in real time in the PCS
- On each PCS cycle, the algorithm receives 9 parameter values from PCS, executes through 500 decision trees, and outputs a disruptivity value. Total computation time is ~200-300 $\mu$s per timeslice
- *The RF algorithm can also determine the relative importances of the 9 parameters in real time (“white box algorithm”)*

- $n_{\text{equal}_1}$
- $q_{95}$
- $n/n_G$
- $\text{ip}_{\text{error}}$ Fraction
- $L_i$
- $\beta_p$
- $V_{\text{loop}} [V]$
- $W_{\text{mhd}} [J]$
- $T_{\text{e width}}$ Normalised
Our algorithm has run in real time in the DIII-D PCS on 900+ discharges (66% non-disruptive, 6% flattop disruptions)

Example of non-disruptive discharge

Disruptivity prediction is generally lower than 0.15 during non-disruptive discharges

Typical computing time 200-300 μs (PCS cycle time is 250 μs)
Our algorithm has run in real time in the DIII-D PCS on 900+ discharges (66% non-disruptive, 6% flattop disruptions)

Example of flattop disruption

Disruptivity rises to high value starting ~0.25 s before disruption

The most important signals can be determined in real time ($B_{n=1}/B_0$, $n/n_G$, $q_{95}$ in this example)
Our Random Forests disruption predictor has run between shots on EAST, during a run dedicated to studying vertical stability

- DPRF ran successfully between shots – gathered data on ~35 discharges

- 1st panel: disruptivity vs time → relative probability of disrupting within the next 100 ms

- 2nd panel: relative contribution of the 13 signals that we used for training and predicting. The 3 most important signals are highlighted
Summary

- We are developing Random Forests (RF) disruption prediction algorithms, trained on our disruption databases on multiple tokamaks
  - Substantial differences between machines, both in disruption physics, and in prediction success rates
- Shot-by-shot analyses can improve on the prediction rate compared to timeslice-by-timeslice analyses, but at the expense of a higher rate of false positives.
- A real time RF predictor was installed and ran in the DIII-D PCS, with encouraging results, and an RF predictor ran between shots on EAST
Near term work planned

- Install and run real time Random Forests predictor in the EAST PCS
- Develop real time shot-by-shot analysis (i.e. alarm trigger)
- Continue developing KSTAR database, and add KSTAR into our multi-machine work
- Test the concept of a universal disruption predictor
- Try other AI algorithms, such as recurrent neural networks
Funding acknowledgements

This work supported by the U.S. Department of Energy, Office of Science, and Office of Fusion Energy Sciences under awards DE-FC02-99ER54512, DE-SC0014264, DE-SC0010720, DE-SC0010492, DE-FC02-04ER54698.

National Natural Science Foundation of China Grants 11475002, 11775262, 11475224, 11575247, 11475225, 11775266 and 11505235

Extra slides
This poster presents 5 principal sub-topics:

- Development of similar disruption databases on 4 tokamaks
- Random Forests and timeslice-by-timeslice prediction performance
- Optimisation of shot-by-shot prediction performance
- Real time prediction on DIII-D, and between-shot prediction on EAST
- Comparison between machines of disruption-relevant signals, and prediction performance
For every plasma discharge, disruptive and non-disruptive, we take time slice data at regular intervals for the entire discharge.
For each disruptive shot, we take *additional* time slices at a higher sampling rate for period before the disruption.
Shot-by-Shot analysis
Optimise F2 metric

- $\tau_{\text{class}} = 325 \text{ ms}$
- High disruptivity = 0.35
- Low disruptivity = 0.05
- Alarm window = 5 ms
- True positives: 77%
- False positives: 19%
- Median Warning [s]: $0.15 \pm 0.13$
Shot-by-Shot analysis
Optimise F2 metric

\( \tau_{\text{class}} = 200 \text{ ms} \)
high disruptivity = 0.45
low disruptivity = 0.05
alarm window = 5 ms
True positives: 81%
False positives: 7%
Median Warning [s]: 0.25 \pm 0.15
Shot-by-Shot analysis

Optimise F2 metric

\[ \tau_{\text{class}} = 375 \text{ ms} \]
\[ \text{high disruptivity} = 0.35 \]
\[ \text{low disruptivity} = 0.05 \]
\[ \text{alarm window} = 5 \text{ ms} \]
\[ \text{True positives:} 83\% \]
\[ \text{False positives:} 5\% \]
\[ \text{Median Warning [s]:} 0.30 \pm 0.20 \]