Supplementary Material

1 SUPPLEMENTARY DATA

1.1 New BS and MP crossing list

The new list of bow shock (BS) and magnetopause (MP) crossings by Cassini can be found at: https://github.com/ikitcheng/BS_MP_Crossings_Cassini/blob/master/dataset/Master_BS_MP_Crossing_List_04_16_incCMJcorr_incIKCcorr.txt

1.2 Changes to the original BS and MP crossing list

The changes made to the previous crossing list can be found at: https://github.com/ikitcheng/BS_MP_Crossings_Cassini/blob/master/dataset/changes_to_BS_MP_crossings.xlsx

1.3 CNN Classifier Architecture

The CNN architecture comprised of a ResNet called 'pretrained_model' with 18 convolutional layers for feature extraction and a fully connected neural network called 'head' for classification. The model is summarised below:

```
TransferredNet(
  (pretrained_model): ResNet(
    (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding
      =(3, 3), bias=False)
    (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
      track_running_stats=True)
    (relu): ReLU(inplace=True)
    (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation
      =1, ceil_mode=False)
    (layer1): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
          padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
          track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
          padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
          track_running_stats=True)
      )
      (1): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
          padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
          track_running_stats=True)
        (relu): ReLU(inplace=True)
```

```
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
        padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    )
  )
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2),
        padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
         track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
        padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
         track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2),
          bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
           track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
        padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
         track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
        padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
         track_running_stats=True)
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2),
        padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
         track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
        padding=(1, 1), bias=False)
```

```
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
          track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2),
           bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
            track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
         padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
          track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
         padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
          track_running_stats=True)
    )
  )
  (layer4): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2),
         padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
          track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
         padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
          track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2),
           bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
            track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
         padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
          track_running_stats=True)
      (relu): ReLU(inplace=True)
```

```
        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
           padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
            track_running_stats=True)
      )
    )
    (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
    (fc): Identity()
  )
  (head): Sequential(
    (0): Linear(in_features=512, out_features=16, bias=True)
    (1): Tanh()
    (2): Linear(in_features=16, out_features=3, bias=True)
  )
)
```

2 SUPPLEMENTARY TABLES AND FIGURES

This section contains supplementary figures accompanying the main manuscript.

2.1 Magnetopause crossings observed in the interval 2005-08-15 12:00:00 to 2005-08-15 20:00:00

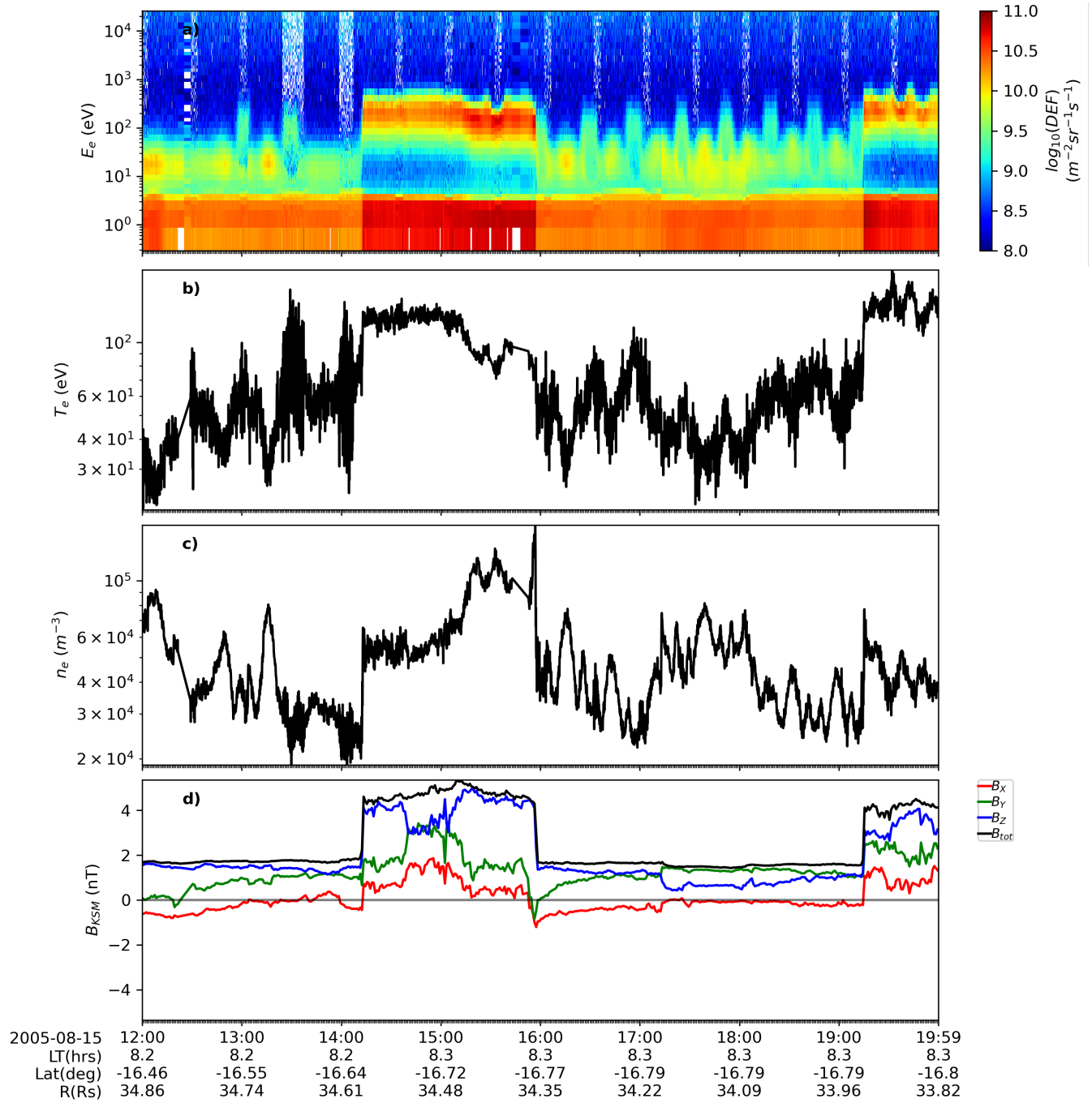

Figure S1. An example of magnetopause crossing in which the magnetosheath magnetic fields were not as turbulent as typical magnetopause crossings, with a small standard deviation over the magnetosheath interval. The panels are as follows: (a) Electron energy-time spectrogram of differential energy flux (DEF) from CAPS/ELS anode 5. (b, and c) Electron temperature and number density derived from ELS anode 5. (d) The magnitude and KSM components of the magnetic field. Spacecraft trajectory information is shown below the x-axis.

2.2 Magnetopause crossing observed in the interval 2009-02-10 17:13:00 to 2009-02-10 19:13:00

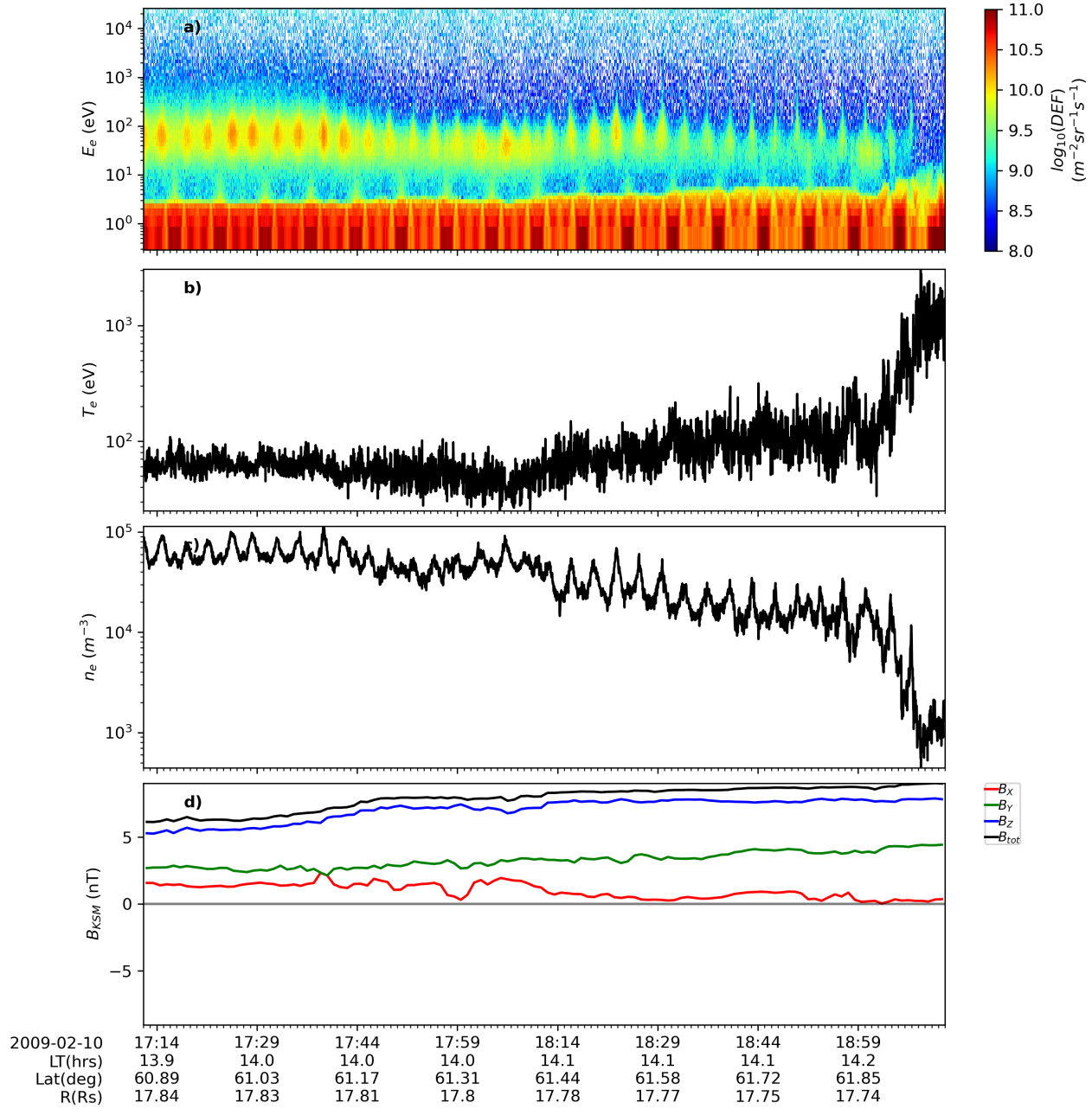

Figure S2. An example of magnetopause crossing in which independently derived, expert-identified catalogues had inconsistent labels for the time of crossing. The panels are as follows: (a) Electron energy-time spectrogram of differential energy flux (DEF) from CAPS/ELS anode 5. (b, and c) Electron temperature and number density derived from ELS anode 5. (d) The magnitude and KSM components of the magnetic field. Spacecraft trajectory information is shown below the x-axis.

2.3 Histogram of the time intervals between consecutive crossings

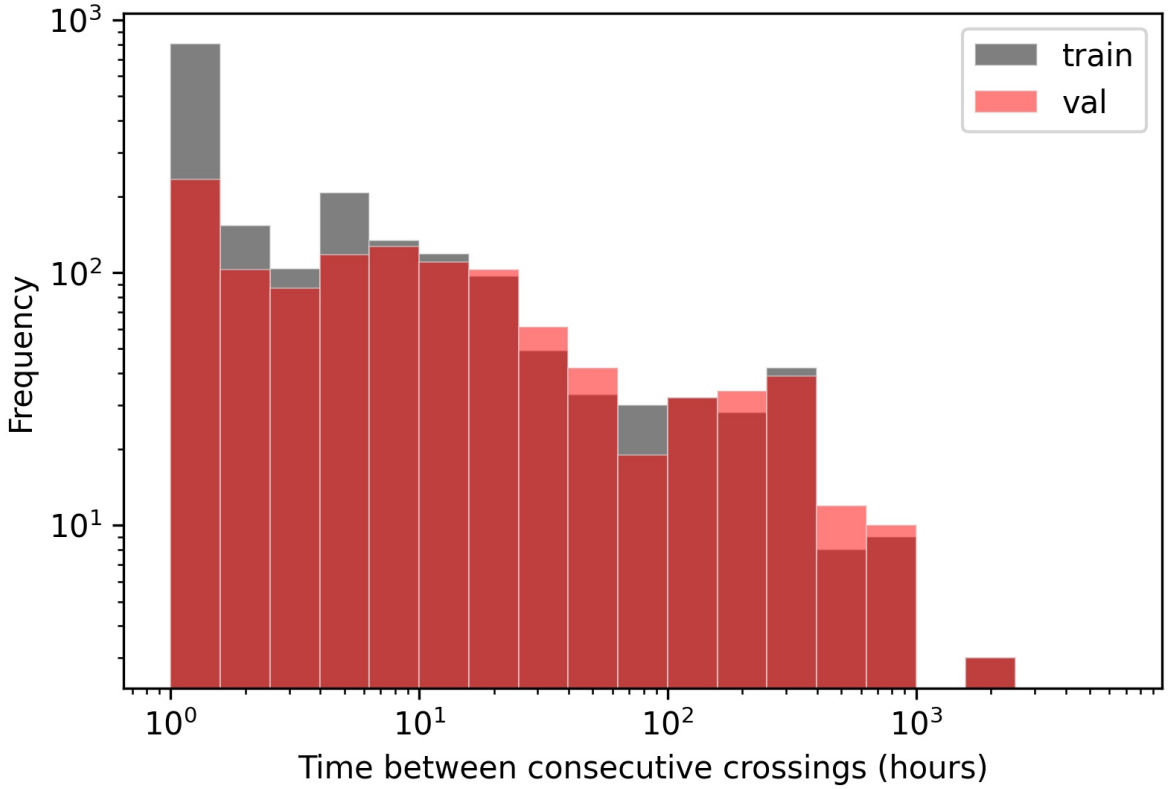

Figure S3. A histogram of time intervals (in hours) between consecutive crossings in the training and validation dataset used to train the convolutional neural network (CNN) classifier.

2.4 Histogram of crossing epochs (on Cassini mission timeline)

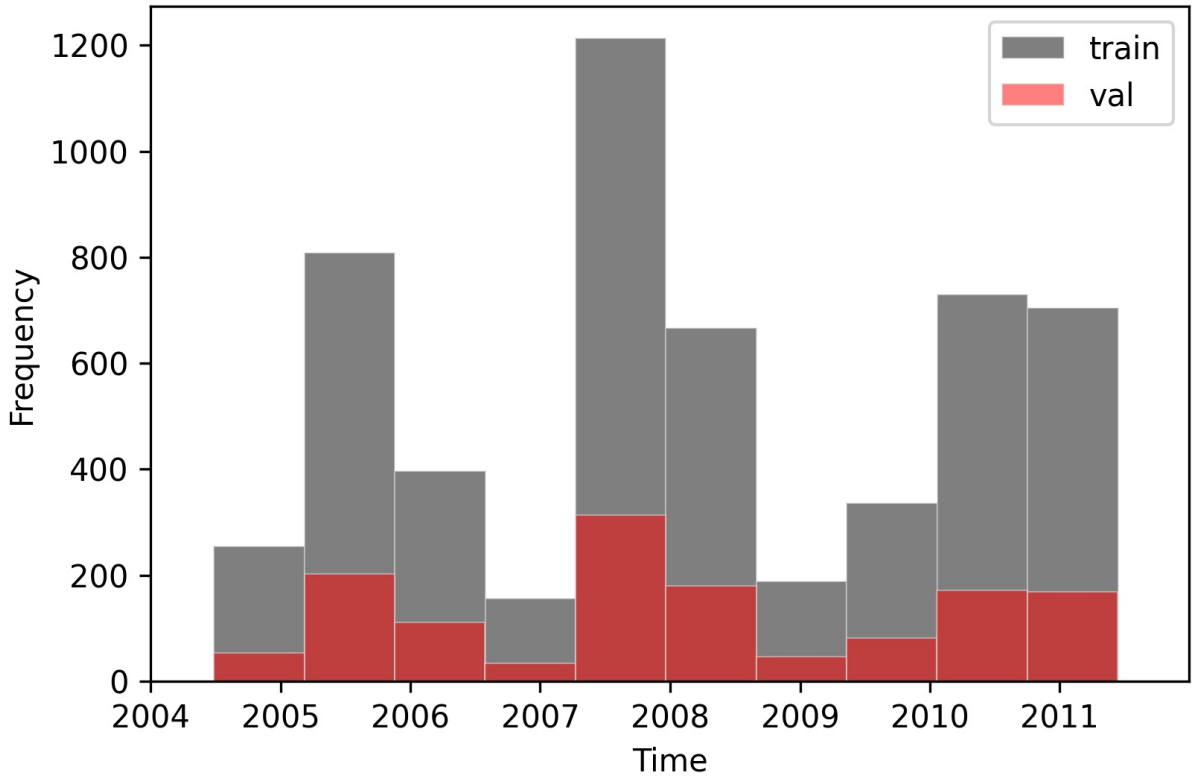

Figure S4. A histogram of crossing epochs (on Cassini mission timeline) in the training and validation dataset used to train the convolutional neural network (CNN) classifier.

2.5 Cross validation performance metrics

CV (10-fold)	Acc_train	Acc_val	AUC_train	AUC_val	Err_train	Err_val	Variance	
1	0.89	0.9	0.91	0.92	0.45	0.55	0.1	
2	0.96	0.92	0.99	0.96	0.12	0.29	0.17	
3	0.96	0.95	0.99	0.97	0.11	0.23	0.12	
4	0.96	0.94	0.99	0.96	0.12	0.21	0.09	
5	0.95	0.96	0.98	0.97	0.15	0.18	0.03	
6	0.95	0.91	0.98	0.95	0.14	0.27	0.13	
7	0.95	0.9	0.97	0.96	0.15	0.29	0.14	
8	0.96	0.96	0.99	0.98	0.11	0.2	0.09	
9	0.93	0.95	0.97	0.98	0.21	0.26	0.05	
10	0.93	0.92	0.98	0.97	0.21	0.29	0.08	
	0.94	**0.93**	**0.98**	**0.96**	**0.18**	**0.28**	**0.10**	Mean
	0.02	0.02	0.02	0.02	0.10	0.10	0.04	Stddev

Table S1. 10-fold cross-validation (CV) results for ResNet18 classifier on Cassini CAPS electron energy spectrogram data between 2004 and 2011. The metrics included for both the train and validation datasets are: Accuracy (Acc), Area under the ROC curve (AUC), and the average cross entropy loss (Err). The last column in the table is the variance of the individual models. The last two rows of the table are the mean and the population standard deviation values for each metric.

2.6 Histograms of test performance metrics for the cnn-caps model from the bootstrap method

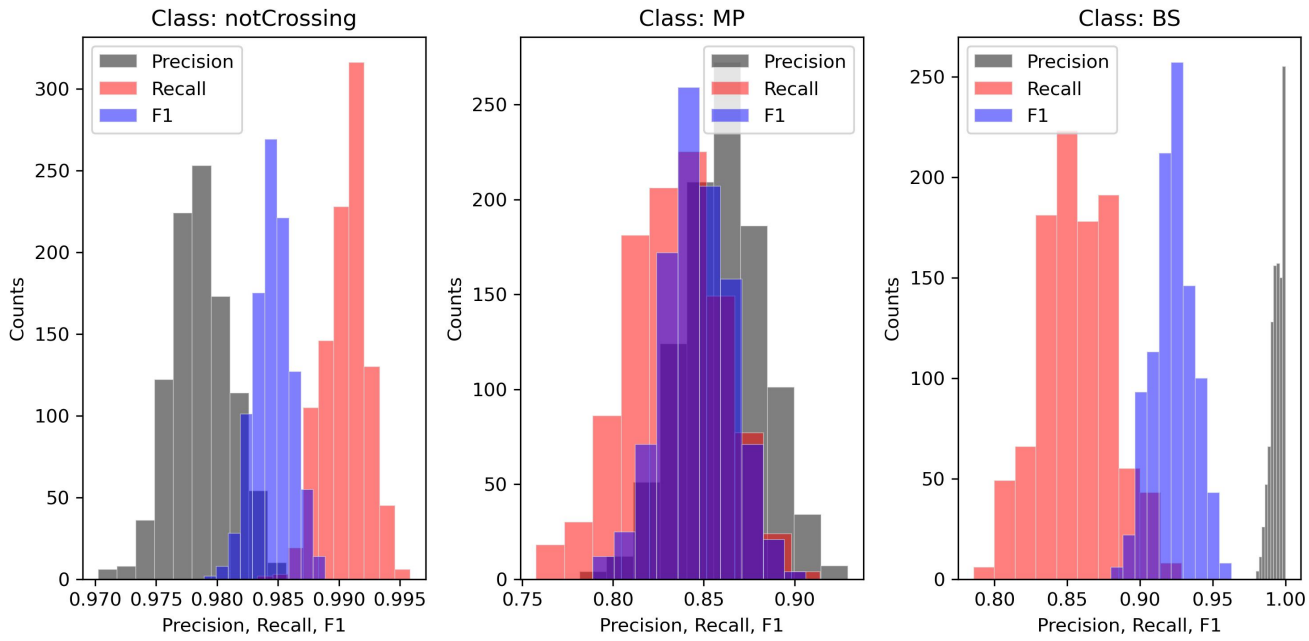

Figure S5. Histograms of test precision, recall and F1 metrics for 'notCrossing', 'MP' and 'BS' identification by the cnn-caps model were determined from 1000 bootstrap samples of the test dataset with replacement. Each sample contains the same number of instances as the original test dataset size (3017). The uncertainties in each metric were determined from the standard deviation of the 1000 bootstrap samples.