2020 Ground Systems Architectures Workshop

Telemanom

Anomaly Detection for Spacecraft/Rover Telemetry Using Long Short-Term Memory Networks

Valentino Constantinou
Kyle Hundman*
Connor Francis
Chris Laporte
Ian Colwell
Chris Mattmann
Tom Soderstrom

Jet Propulsion Laboratory
California Institute of Technology

* now affiliated with American Family Insurance.

1. Introduction: Increasing Operational Demands

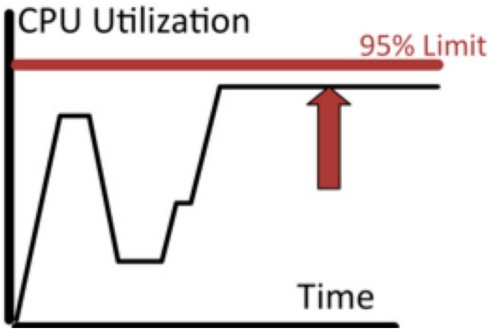
2. Anomaly Detection for SMAP and MSL

3. Wrap-Up

Increasing operational demands to analyze spacecraft telemetry can no longer be met with traditional approaches.

Motivation

- Thresholding, expert systems
  - Reliance on expert knowledge
  - Custom
  - Not complete
  - Accuracy
  - Appropriate limits change



Simple example of anomaly that would be undetected by a threshold
~40% of anomalies in experiments are of this nature

- Increasing data rates
  - **SWOT, NISAR = 3-5 TB daily**
- Smaller missions (e.g. cubesats)
  - **Less people for ops**
- Condensed mission operations

- High volumes of testbed data
- Investigative aspect
  - Focused, prioritized telemetry review
  - Help with causal fault analysis: what anomalies were detected leading up to a failure?

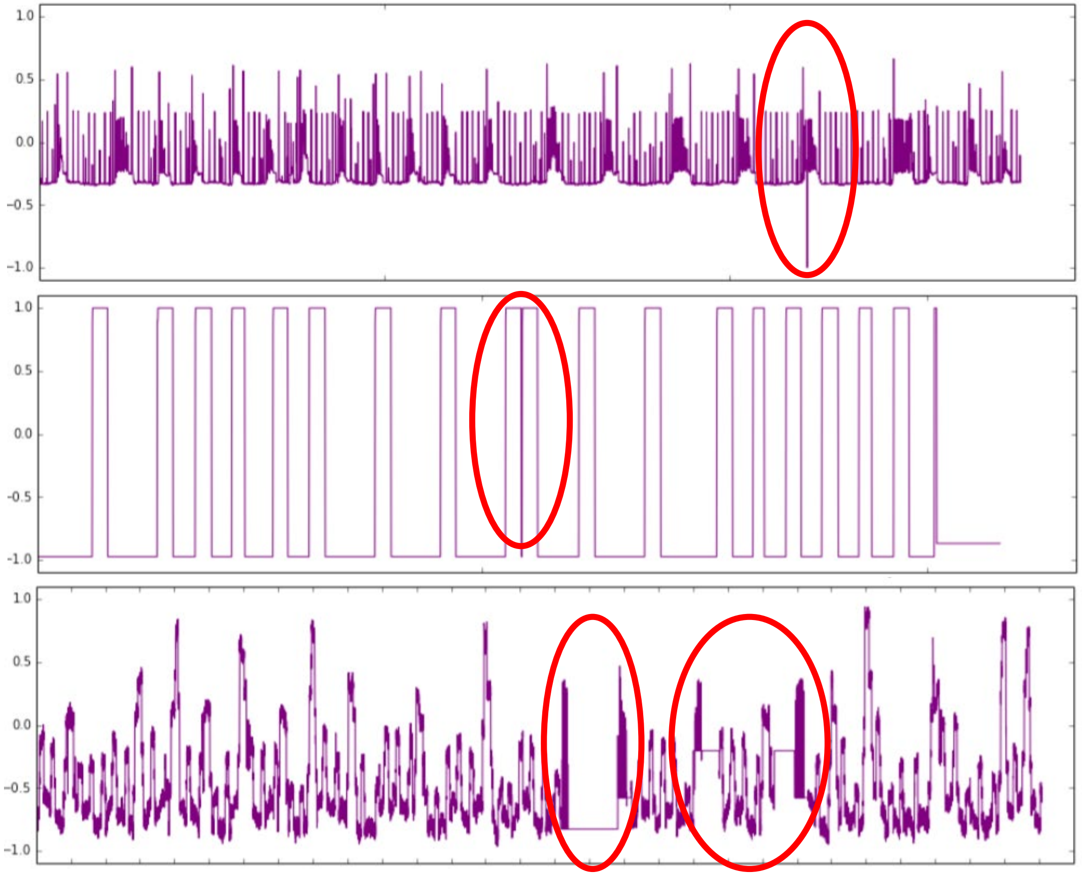
Anomaly Categories

Chandola et al. 2007

1. Point

2. Contextual

3. Collective (sequential)



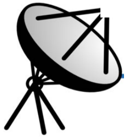
Data volume and velocity – along with anomaly complexity – requires an alternative approach to detecting anomalous behaviors on spacecraft.

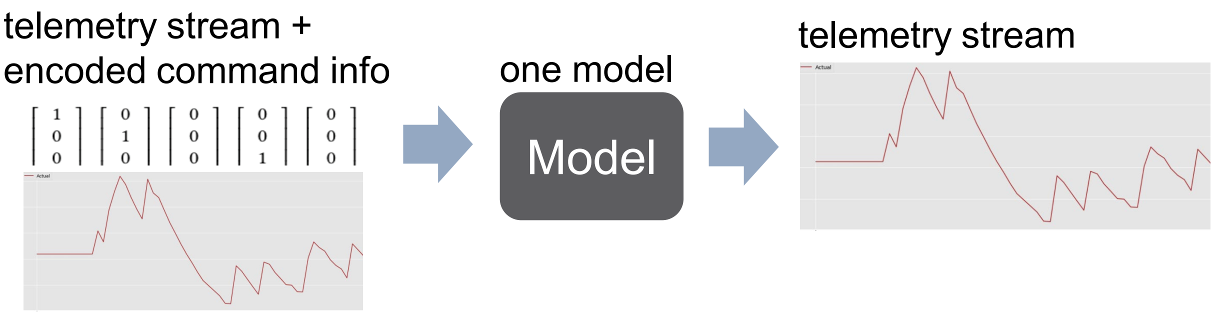
1. Introduction: Increasing Operational Demands

2. **Anomaly Detection for SMAP and MSL**

3. Wrap-Up

Overview





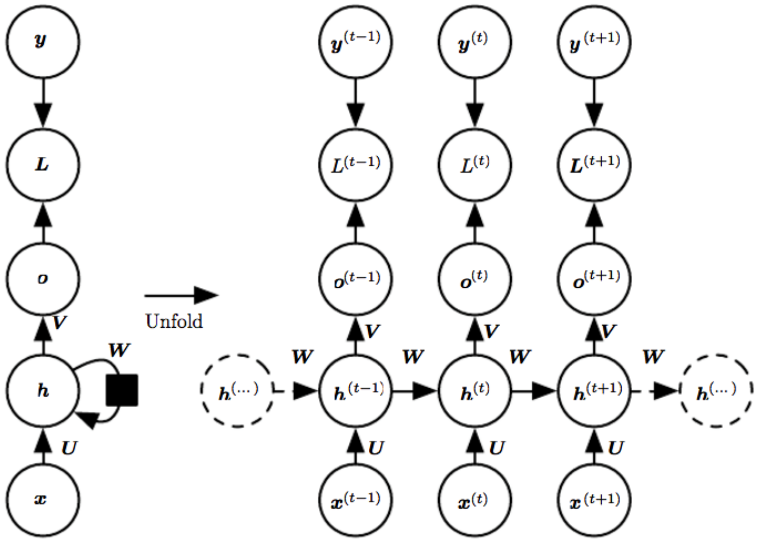
- Use Recurrent Neural Networks (LSTMs) to predict incoming telemetry values using recent telemetry, commands, and event records (EVRs) as inputs
- Where predictions are substantially different from actual telemetry values, these are identified as potentially anomalous events
  - Novel method for defining “substantially different”

[Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding]

Recurrent Neural Nets

$$\begin{aligned}\boldsymbol{h}^{(t)} &= g^{(t)}(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t-1)}, \boldsymbol{x}^{(t-2)}, \ldots, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)}) \\ &= f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}).\end{aligned}$$

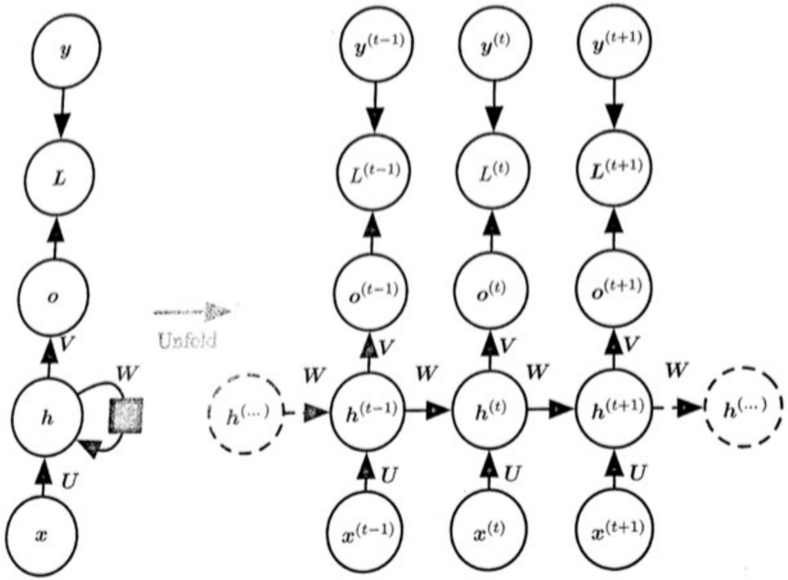
- Memory (lossy summary)
- Parameter sharing
  - Extend model to apply to different lengths and generalize across time steps
    - Don't have to have separate parameters for each time value
- Recurrence
  - Always has same input size regardless of sequence length



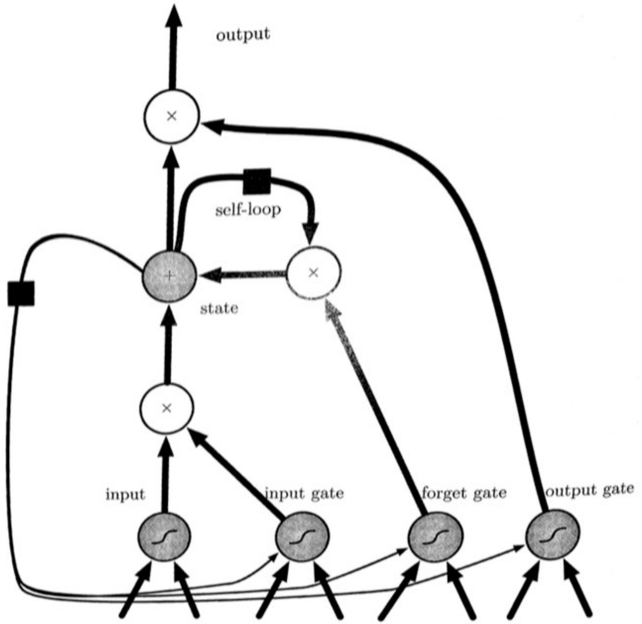
Ian Goodfellow, Yoshua Bengio, Aaron Courville, 2016. *Deep Learning.* MIT Press. http://deeplearningbook.org

From RNNs to LSTMs (Goodfellow et. al, 2016)

RNN



LSTM



Core contribution (1997): Self-loops
Crucial addition (2000): Condition loop on context (with another hidden unit)

Ian Goodfellow, Yoshua Bengio, Aaron Courville, 2016. *Deep Learning.* MIT Press. http://deeplearningbook.org

Formulation

$$X = \Bigg\{ \overbrace{\begin{bmatrix} x_1^{(t-l_s)} \\ x_2^{(t-l_s)} \\ \vdots \\ \boxed{x_m^{(t-l_s)}} \end{bmatrix}, \ldots, \begin{bmatrix} x_1^{(t-1)} \\ x_2^{(t-1)} \\ \vdots \\ \boxed{x_m^{(t-1)}} \end{bmatrix}, \begin{bmatrix} x_1^{(t)} \\ x_2^{(t)} \\ \vdots \\ \boxed{x_m^{(t)}} \end{bmatrix}}^{\text{Model inputs at step } t}, \underbrace{\begin{bmatrix} x_1^{(t+1)} \\ x_2^{(t+1)} \\ \vdots \\ \boxed{x_m^{(t+1)} = y^{(t)}} \end{bmatrix}}_{l_p = 1} \Bigg\}$$

Telemetry Values

h = historical window of errors
l_s = sequence length

$$e^{(t)} = \left[\hat{y}^{(t)} - y^{(t)}\right]$$

$$\mathbf{e} = \left[e^{(t-h)}, \ldots, e^{(t-l_s)}, \ldots, e^{(t)}\right]$$

Single-Channel Prediction

$$\mathbf{t} = \{[106], [107], [108], [109], [110], [111]\}$$

Cmd sent to Module A (T/F)

Cmd received by Module A (T/F)

Cmd sent to Module B (T/F)

Telemetry Value

$$X = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1.40 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 1.40 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 1.40 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1.45 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 1.45 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1.40 \end{bmatrix} \right\}$$

Same command info for every channel

$$\hat{\mathbf{y}} = \{[1.39], [1.39], [1.36], [1.48], [1.46], [1.41]\}$$

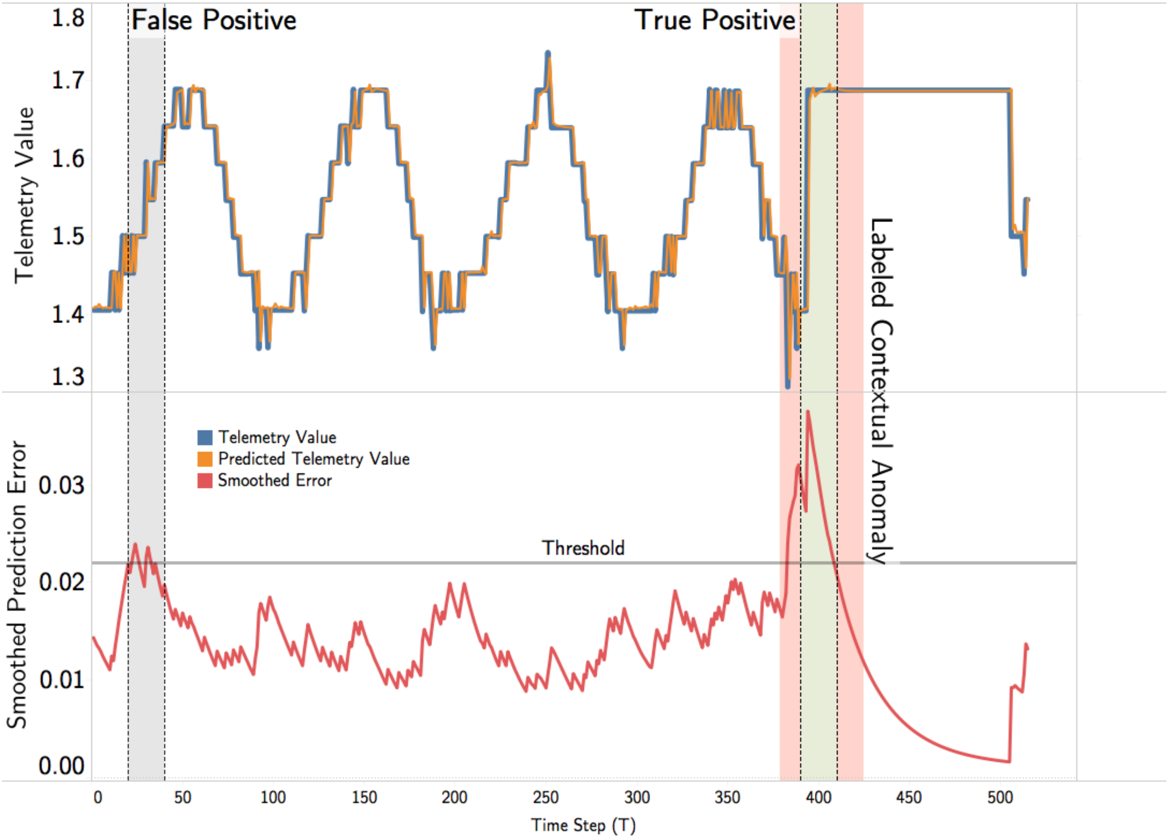
$$\mathbf{e} = \{[0.01], [0.01], [0.04], [0.03], [0.01], [0.01]\}$$

$$\mathbf{e}_s = \{[.016], [.014], [.015], [.017], [.015], [.012]\}$$

Reconstruction Errors

Actuals and Prediction

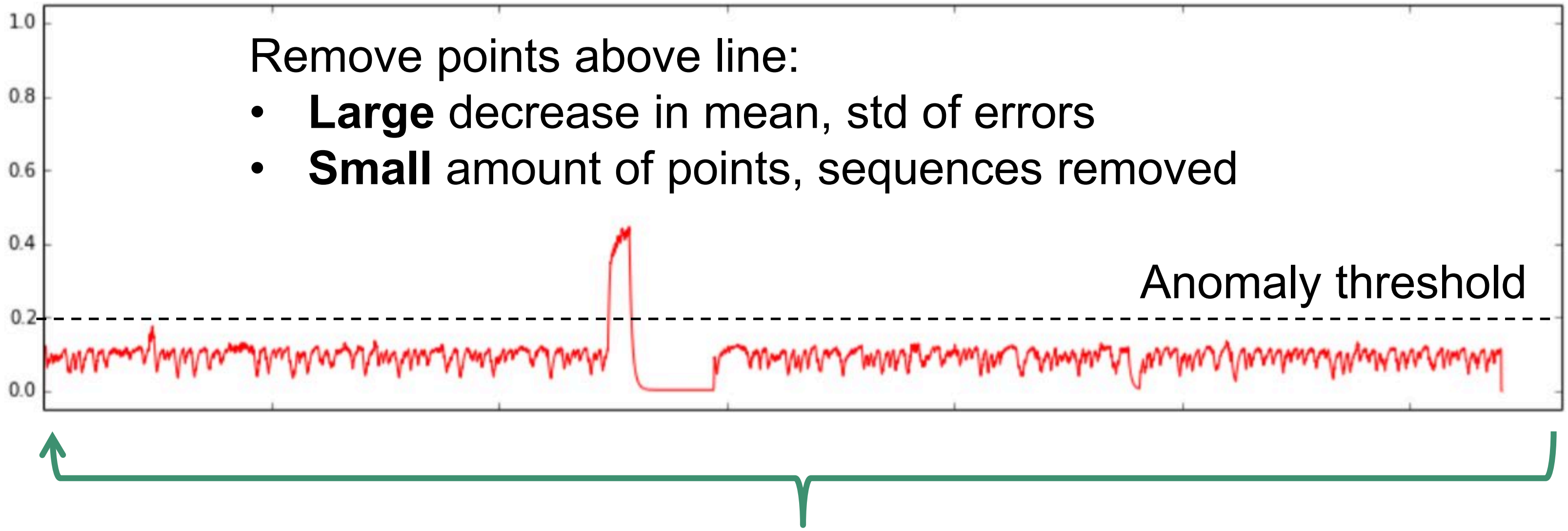
Raw Reconstruction Error



Dynamic Anomaly Threshold

Anomalous

Reconstruction Error

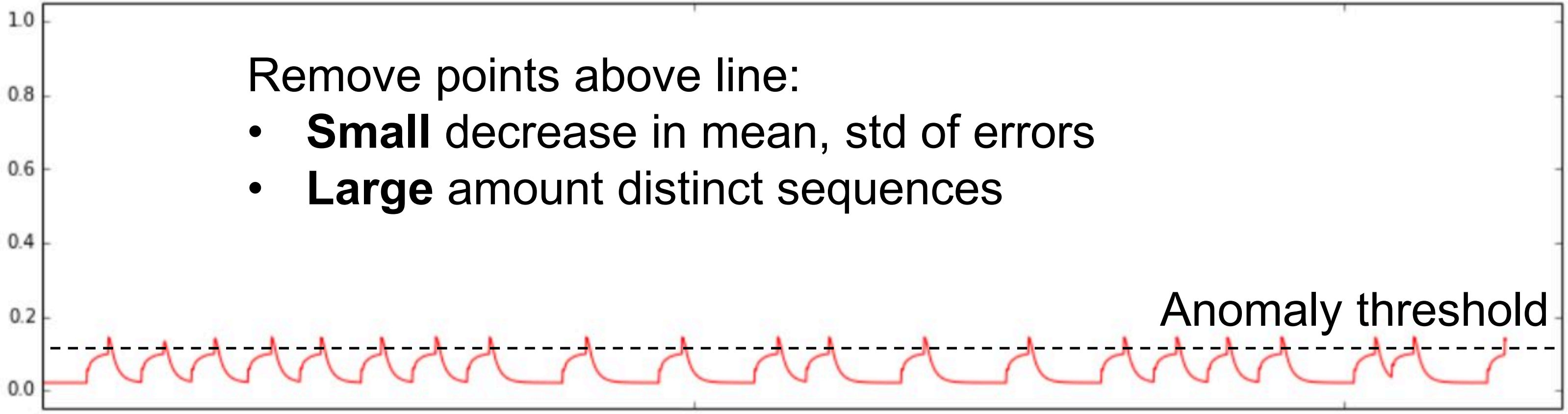


Window size (h)

Dynamic Anomaly Threshold

Nominal

Reconstruction Error



Dynamic Anomaly Threshold

Smoothed errors

$$\mathbf{e}_s = [e_s^{(t-h)}, \ldots, e_s^{(t-l_s)}, \ldots, e_s^{(t-1)}, e_s^{(t)}]$$

Candidate thresholds

$$\epsilon = \mu(\mathbf{e}_s) + \mathbf{z}\sigma(\mathbf{e}_s)$$

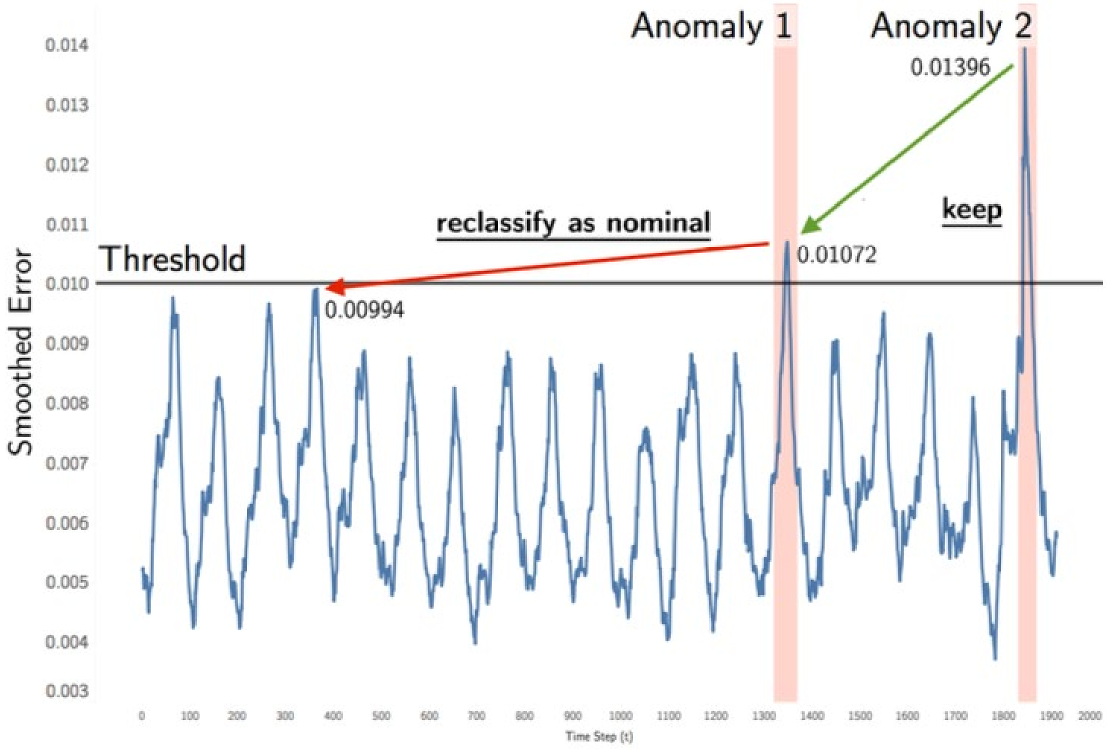
Threshold

$$\epsilon = argmax(\epsilon) = \frac{\Delta\mu(\mathbf{e}_s)/\mu(\mathbf{e}_s)) + (\Delta\sigma(\mathbf{e}_s)/\sigma(\mathbf{e}_s)}{n(\mathbf{e}_a) + n(\mathbf{E}_{seq})^2}$$

Definitions

$$\Delta\mu(\mathbf{e}_s) = \mu(\mathbf{e}_s) - \mu(\{e_s \in \mathbf{e}_s | e_s < \epsilon\})$$
$$\Delta\sigma(\mathbf{e}_s) = \sigma(\mathbf{e}_s) - \sigma(\{e_s \in \mathbf{e}_s | e_s < \epsilon\})$$
$$\mathbf{e}_a = \{e_s \in \mathbf{e}_s | e_s > \epsilon\}$$
$$\mathbf{E}_{seq} = \text{continuous sequences of } e_a \in \mathbf{e}_a$$

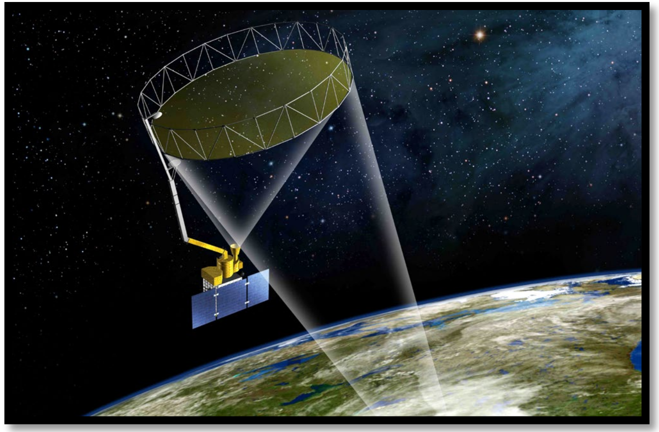
Pruning



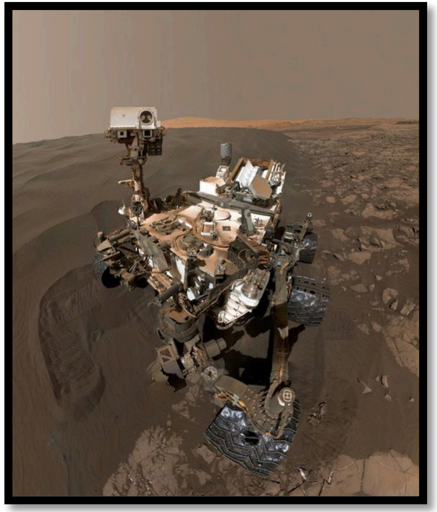
$$\mathbf{e}_{max} = [0.01396, 0.01072, 0.00994]$$

$$p = 0.1$$

Experiments – Two Representative Spacecraft



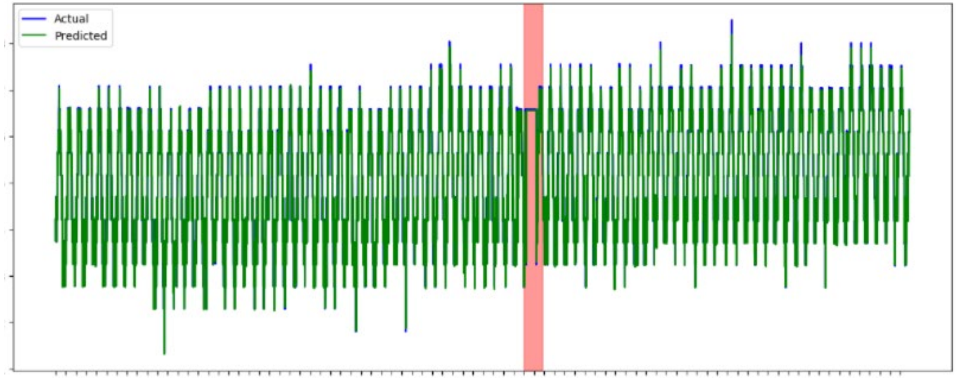
Soil Moisture Active Passive (SMAP)
- Higher, more consistent data rates
- Fewer, more routine behaviors



Mars Science Laboratory (Curiosity or MSL)
- More channels (12k)
- Less data, less consistent delivery
- Extremely varied behaviors
  - Training on recent data isn't enough

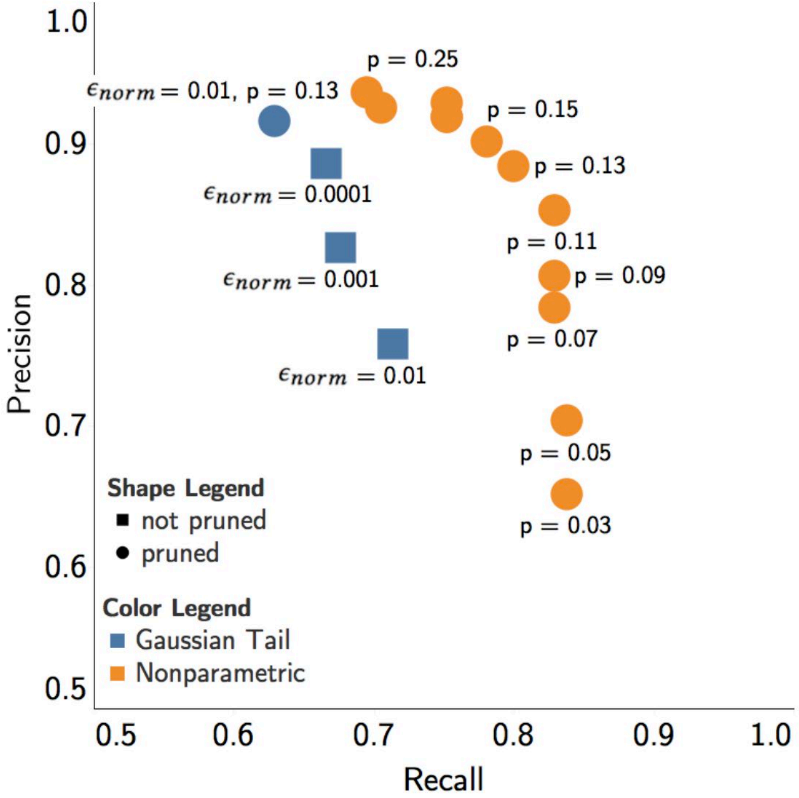
Experiments – Incident Surprise, Anomaly Reports (ISAs)

- Scraped ISAs to find mentions of telemetry channels
  - Ex. "On DOY 192, in the time range from 09:21z through 10:47z, the following channels were found to have odd constant values: A-3, A-4, A-5, A-6, G-3"

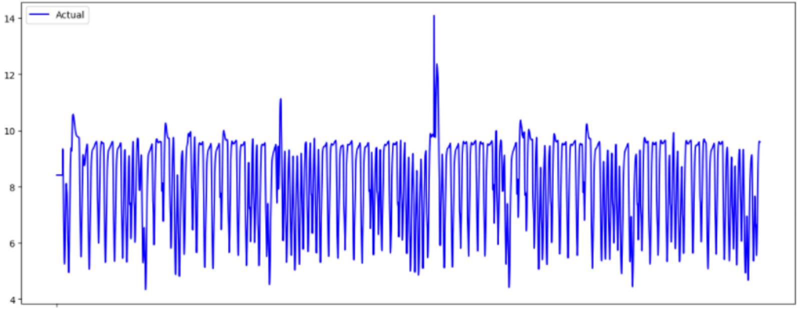


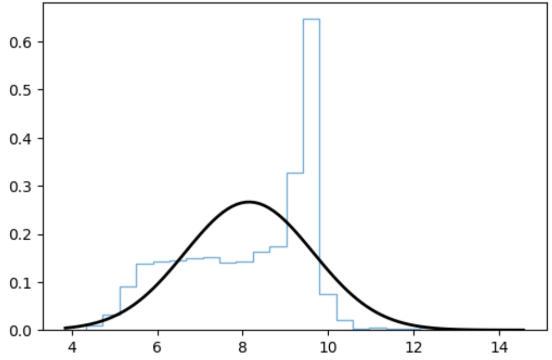
  - Labeled anomalous ranges for 112 unique ISA anomalies
  - Significant portion of contextual anomalies (39%)

Results



Not Gaussian

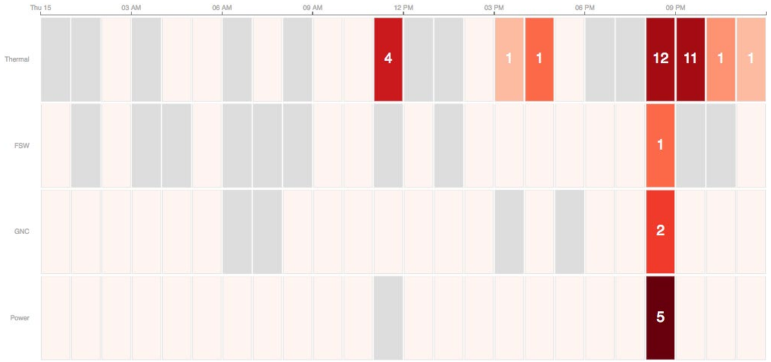




p-value in normal test: 1.17603615763e-237

Initial Pilot: SMAP

- Deployed end-to-end autonomous system
- Monitored ~750 core telemetry channels from Aug 2017 – May 2018
    - Detected multiple verified anomalous events
        - Partial eclipse (Feb 15, 2018)
- Radar (HPA) failure investigation
    - Ran system ~2 months prior to failure, detected many of same telemetry oddities that were identified during peer review process following failure



1. Introduction: Increasing Operational Demands

2. Anomaly Detection for SMAP and MSL

3. Wrap-Up

Future Work: Mars Science Laboratory (MSL)

- SMAP is an Earth-orbiting mission with routine behavior
  - Data downlinked to the Deep Space Network every 72 minutes
  - Science objective the same from day-to-day → some commands for routine behavior, others not

- Extension to non-routine missions such as rovers is an added challenge
  - Irregular behaviors make defining nominal (normal) behavior for training more difficult → requires operator knowledge of mission plan
  - Additional context is needed by the model to make accurate predictions of telemetry
    - For MSL, we include more command information (versus SMAP) and event records, called EVRs

- Early progress on MSL is promising
  - Successfully detected Martian sandstorm with only a small number of Thermal telemetry channels
    - This is the same sandstorm that doomed Opportunity (Mars Exploration Rover, MER)
  - Early implementation achieving a high prediction accuracy for Thermal channels (~98%)
    - Next step is to expand to additional Thermal channels and other telemetry channel types

Summary

- We demonstrate the viability of using LSTMs for predicting anomalies in spacecraft telemetry channels at scale.

- We outline a nonparametric dynamic thresholding approach that does not rely on labels (difficult to obtain in many anomaly detection tasks).

- We identified key challenges and early results for implementation of such approaches on non-routine missions such as MSL.

- We presented current and future work in generalization of our Telemanom system with the goal of providing anomaly detection as a service for missions.

Give it a try!

Paper:

Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding

Github (paper-specific code):

https://github.com/khundman/telemanom

Getting in touch:

vconstan@jpl.nasa.gov

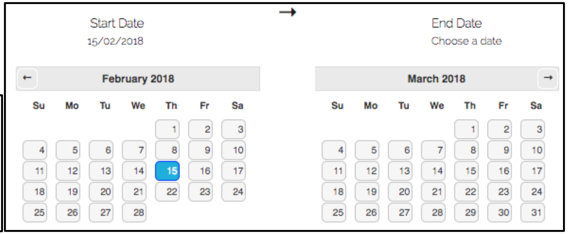
Jet Propulsion Laboratory
California Institute of Technology

jpl.nasa.gov

Appendix

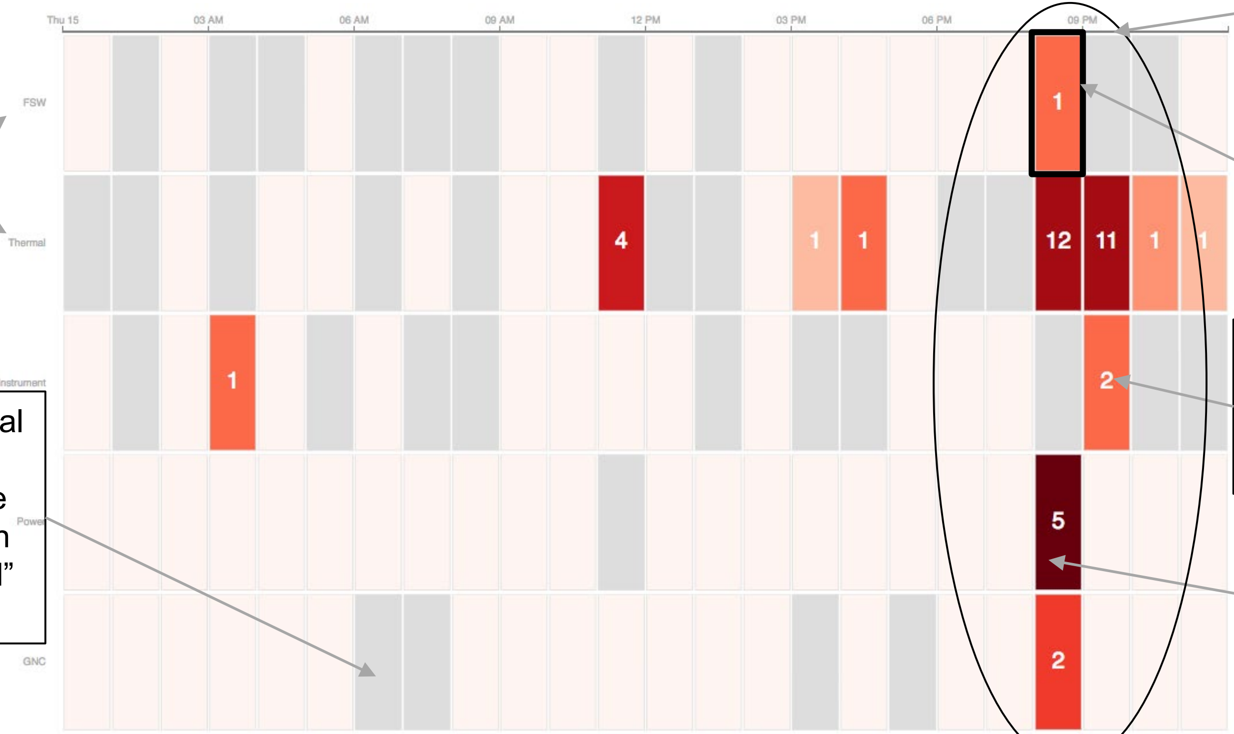
Interface: Top-Level Summary

Start by selecting a start and end date to look at



February 15th, 2018 Partial Solar Eclipse anomaly

Subsystem



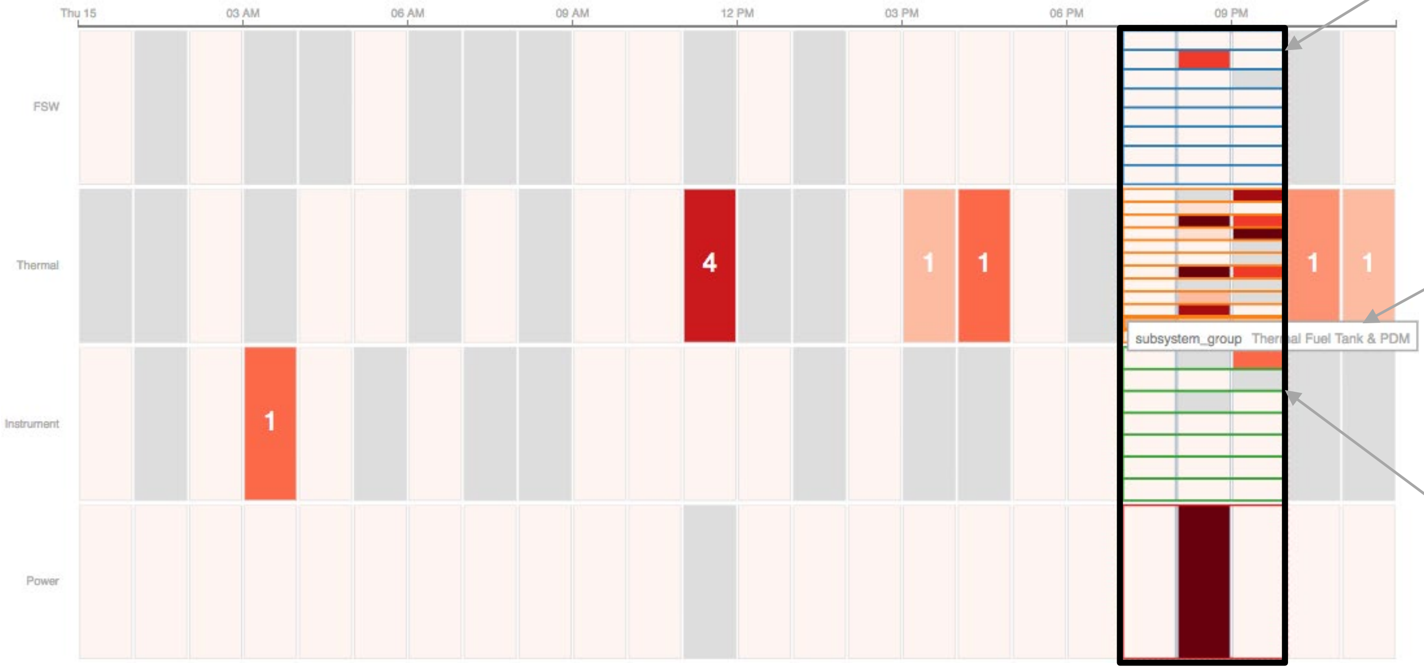
Each box indicates one hour aggregate time window (adjustable)

Count of channel anomalies in subsystem during hour time window

Gray boxes are potential anomalies that the system has learned are false positives with high likelihood ("suppressed" anomalies)

Darker color indicates

Interface: Drilldown

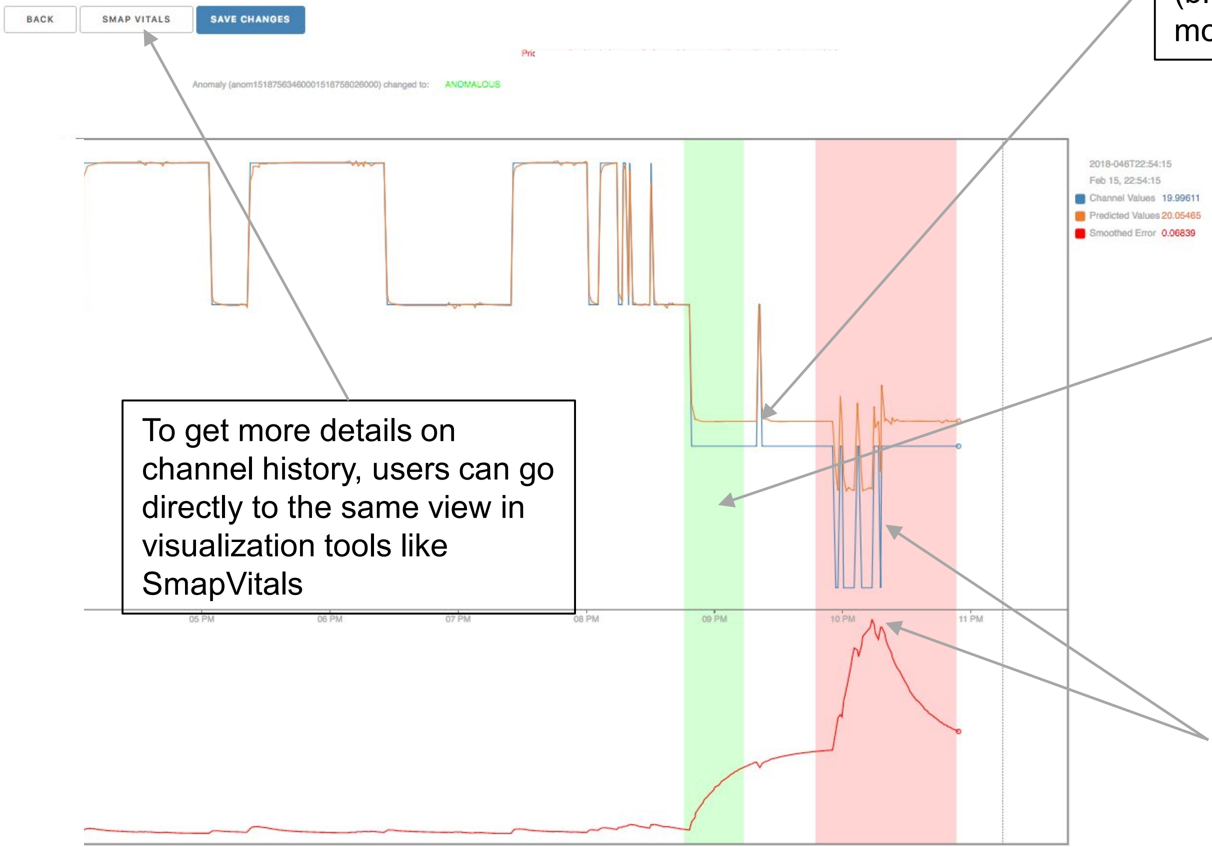


Clicking and dragging across an area allows for looking down a level to channel groups with subsystems

Each row represents a group of channels and hovering shows the group name

Clicking takes the user into a similar view but in the next level down for the selected window

Interface: Drilldown (cont.)



Users can drill down into the raw telemetry for each channel (blue) and compare to the model predictions (orange)

Users can click to tag anomalies as true or false positives, which are used by the system to refine results

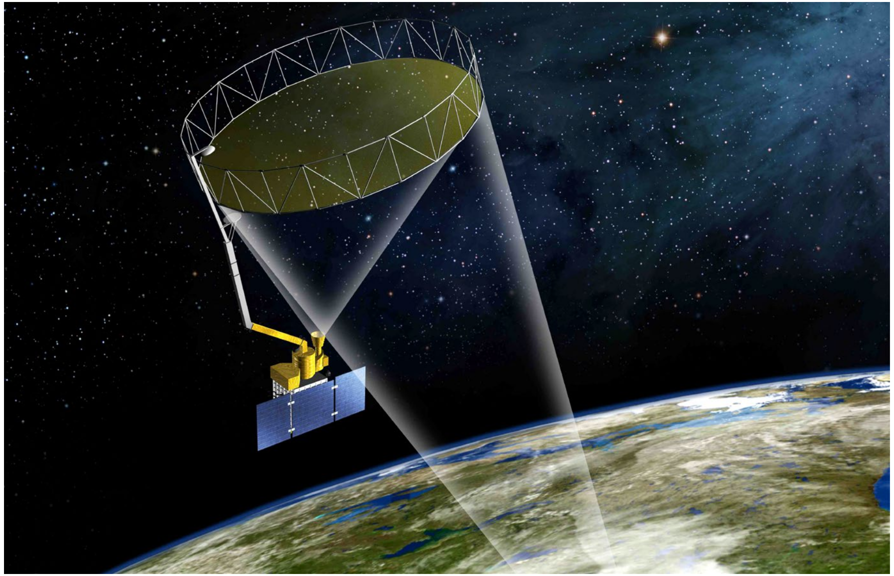
True = green
False = gray
Unlabeled = red

To get more details on channel history, users can go directly to the same view in visualization tools like SmapVitals

Where prediction errors are large, anomalies are flagged

Soil Moisture Active Passive (SMAP)

- Routine operations
- Major radar failure
- ~4,000 telemetry channels
  - Power, CPU, RAM, Thermal, Radiation, counters
  - 14 command modules
  - 4B values
- Challenges
  - Semi-supervised
  - Complexity, diversity
  - Scale



Abstract and Bio

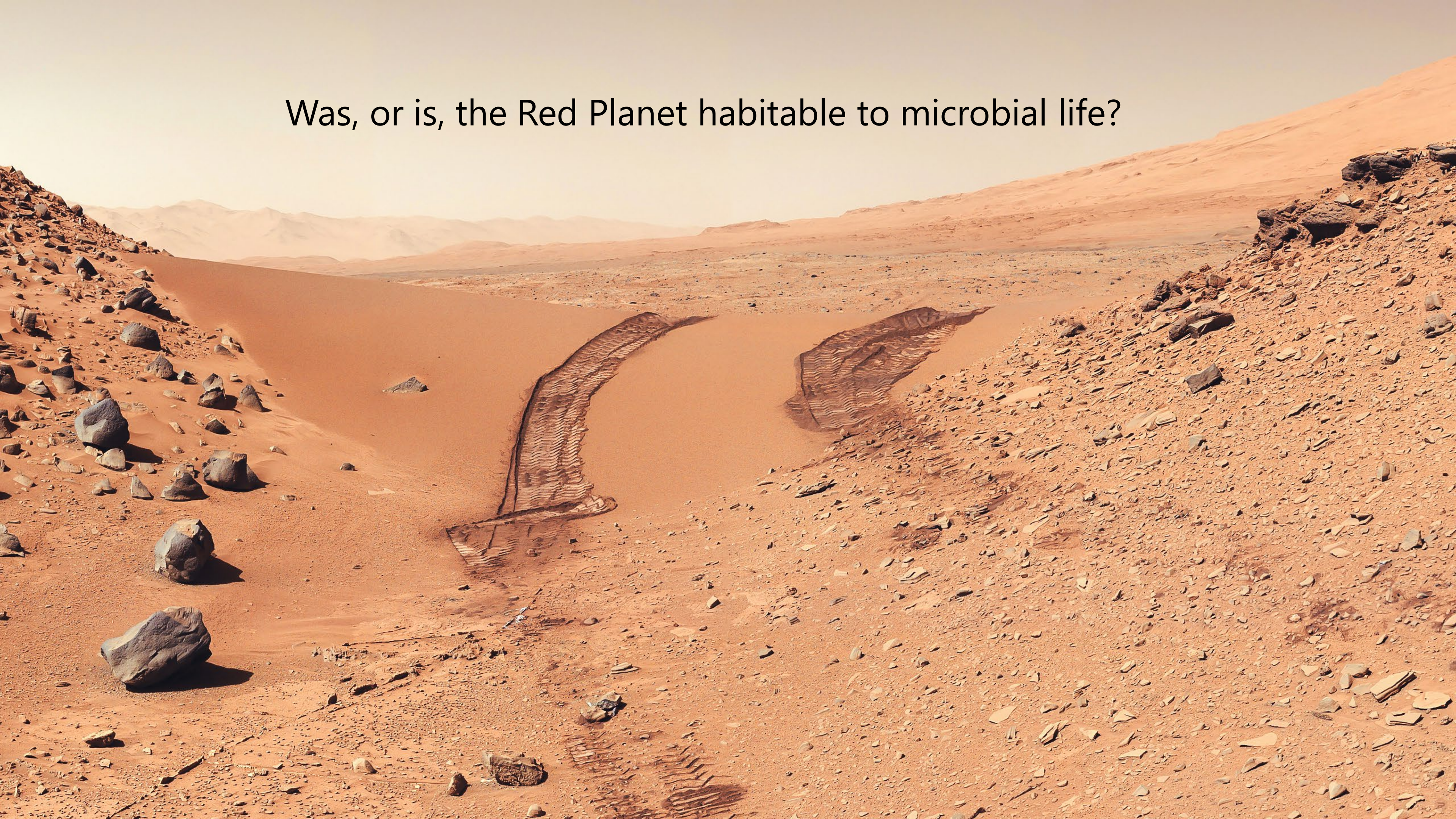
Abstract: The NASA Jet Propulsion Laboratory (JPL) is a leader in the robotic study and exploration of the solar system. As spacecraft send back increasing amounts of telemetry data, improved anomaly detection systems are needed to lessen the monitoring burden placed on operations engineers and reduce operational risk. The majority of spacecraft monitoring systems only target a subset of anomaly types and often require costly expert knowledge to develop and maintain due to challenges involving scale and complexity. In this talk, we demonstrate the effectiveness of Long Short-Term Memory (LSTMs) networks, a type of deep learning model called a Recurrent Neural Network (RNN), in overcoming these issues using expert-labeled telemetry anomaly data from two JPL spacecraft - the Soil Moisture Active Passive (SMAP) satellite and the Mars Science Laboratory (MSL) rover, Curiosity. A complementary unsupervised and nonparametric anomaly thresholding approach developed during a pilot implementation of an anomaly detection system for these missions is detailed, along with a false positive mitigation strategy. We also highlight current work in the generalization of a time-series telemetry anomaly detection framework and challenges to overcome in applying telemetry anomaly detection for missions such the Mars Science Laboratory (MSL).

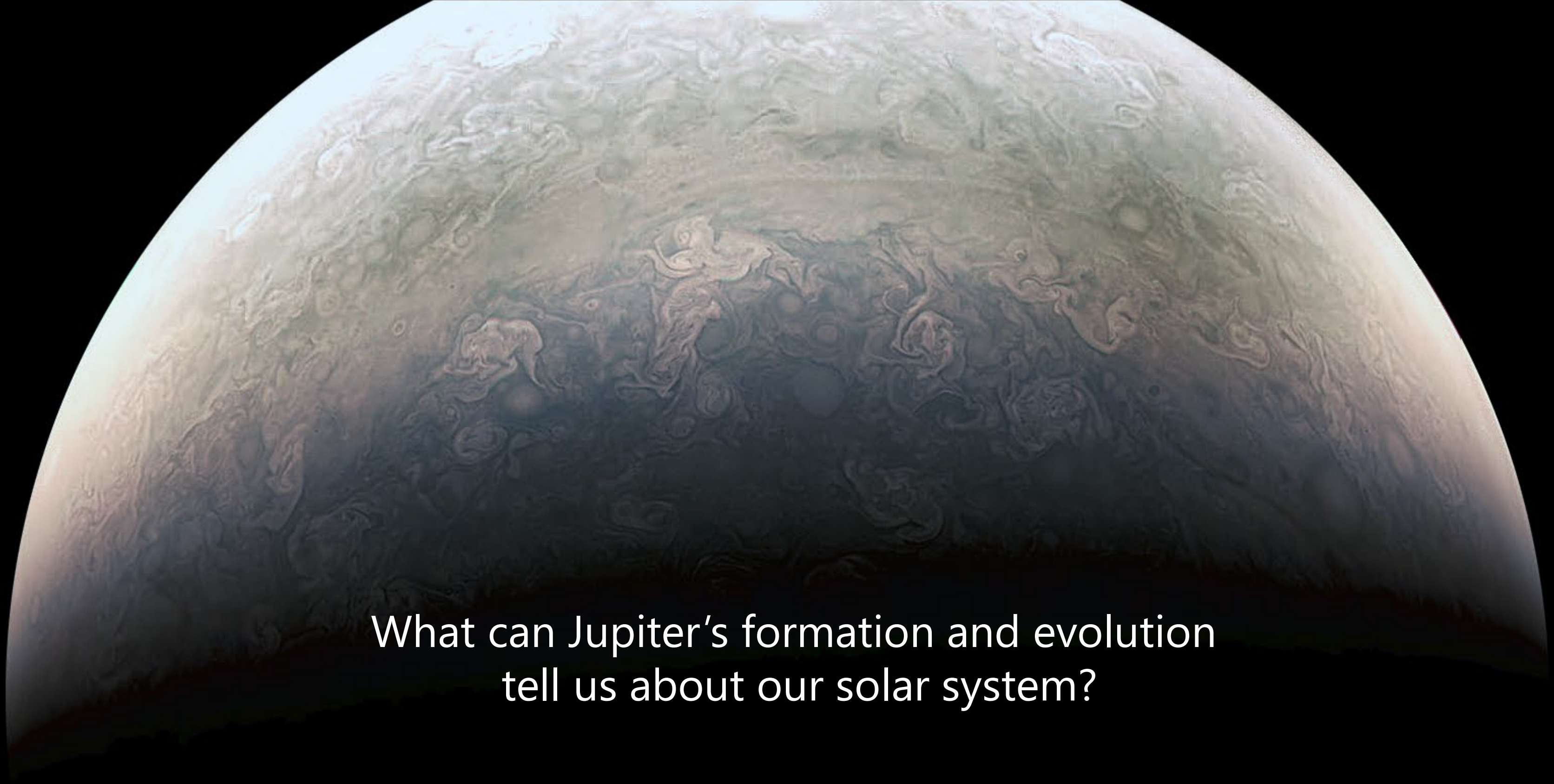
Bio: Valentino Constantinou is a data scientist in the NASA Jet Propulsion Laboratory's (JPL) Chief Technology and Innovation Office, pursuing and developing solutions for effective anomaly detection and management. He serves as the principal investigator for a multi-year effort in developing anomaly (alarm) management products to direct operations engineers' attention during periods of anomaly (alarm) flood, and is part of the government team for the DARPA ASED program (social engineering attacks). Although originally from Cyprus, Valentino grew up in East Tennessee prior to leaving the area for his current home in Los Angeles, California. He holds a master's degree in Analytics (MSiA) from Northwestern University and a bachelor's degree in Economics from the University of Tennessee, where he spent his time prior to joining JPL in February 2017.

1. **Introduction: the Jet Propulsion Laboratory**
2. Anomaly Detection for SMAP and MSL
3. Generalization of Telemanom Framework
4. Wrap-Up

Leaders in the robotic exploration of the solar system.

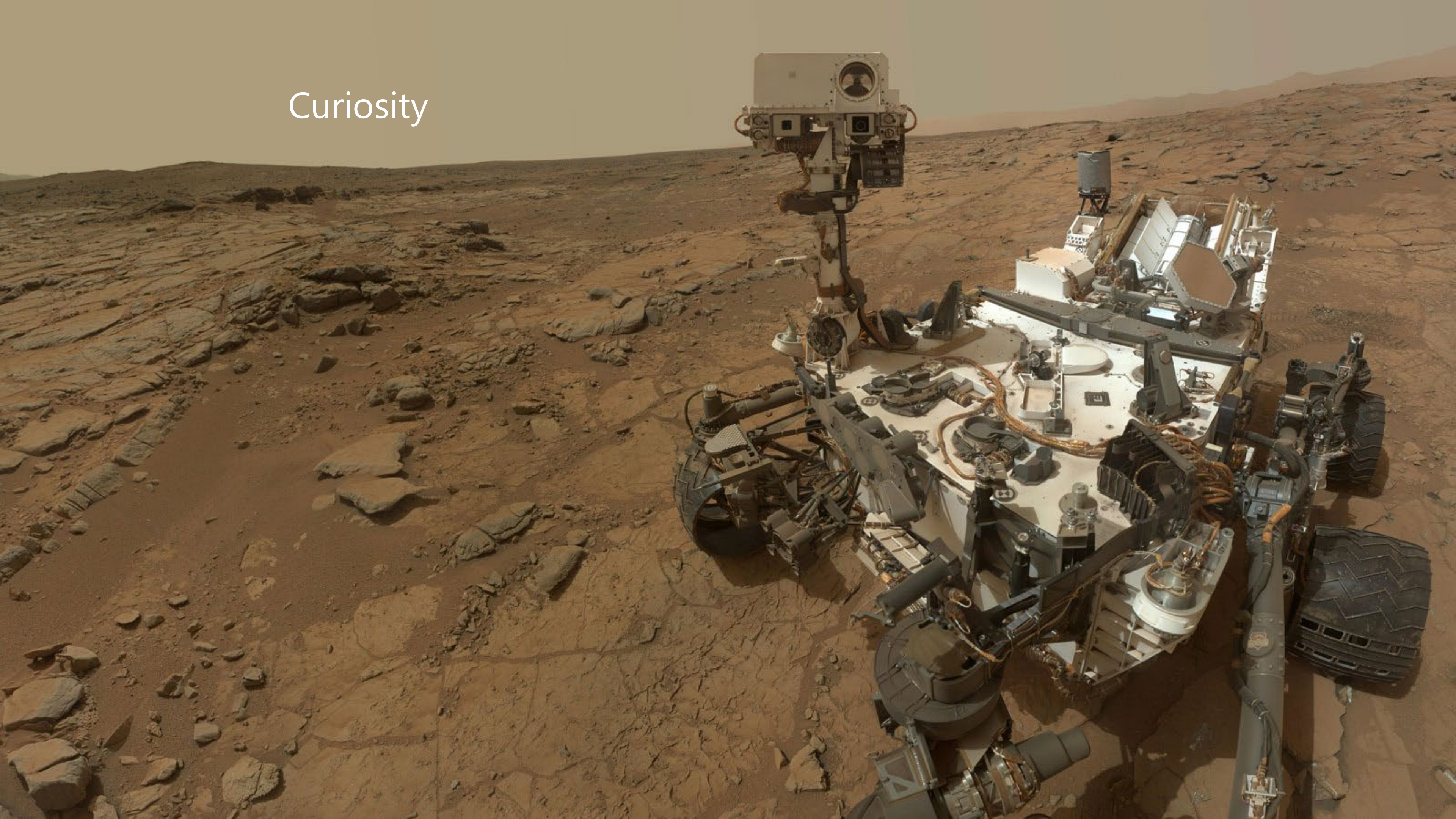
We look to answer questions like…

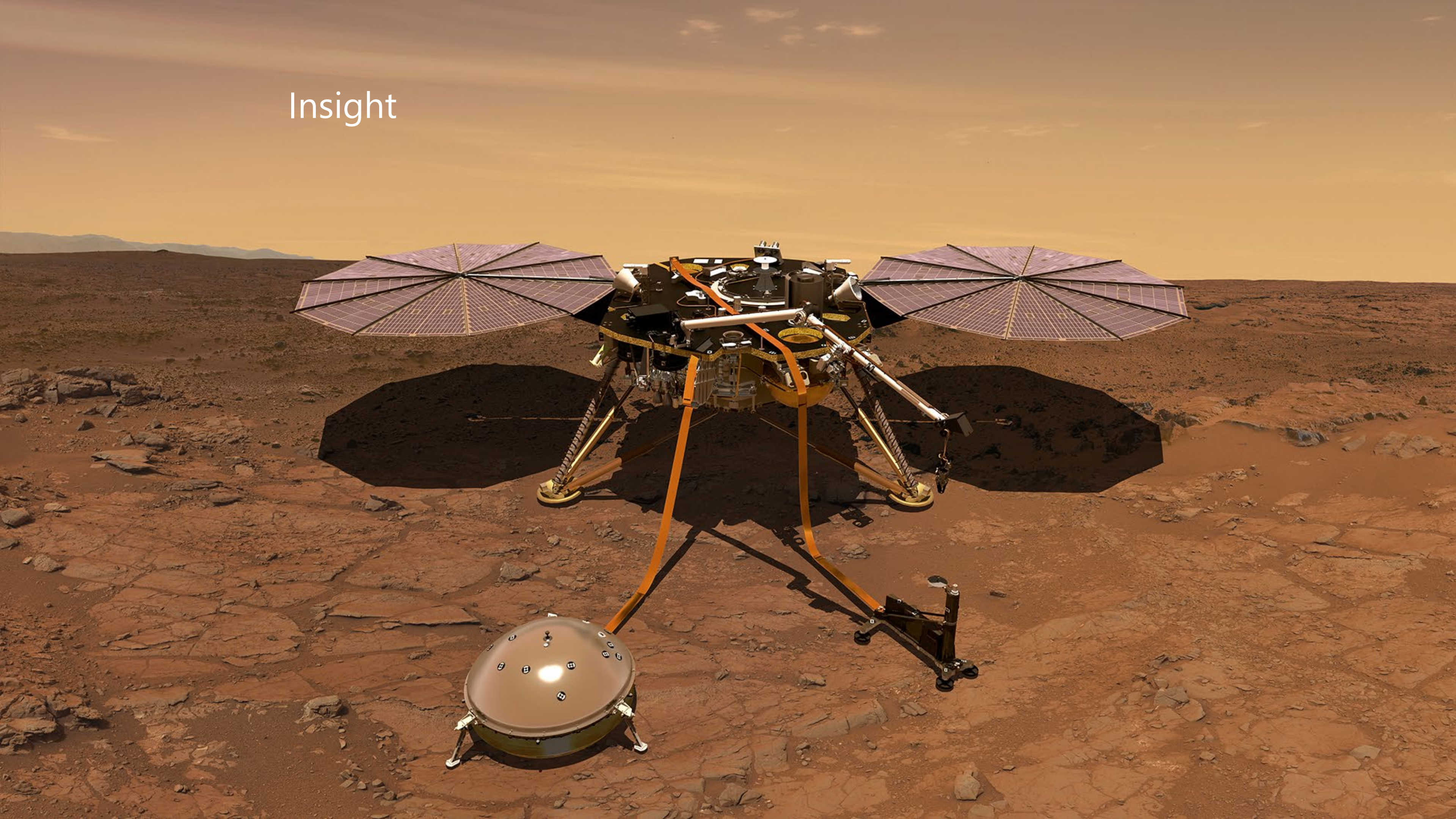

Was, or is, the Red Planet habitable to microbial life?



What can Jupiter’s formation and evolution tell us about our solar system?

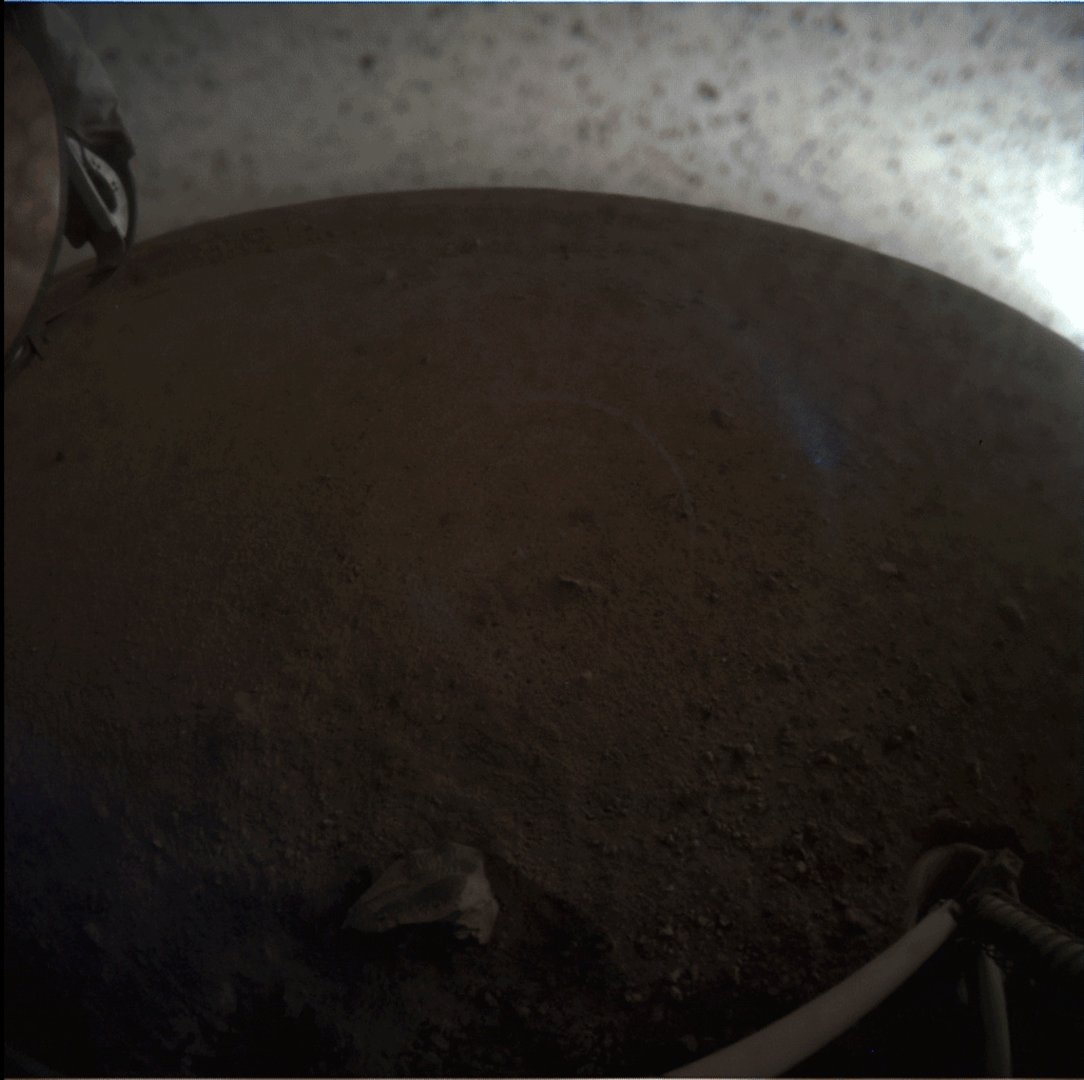
To answer these questions, we require robots like…


Curiosity

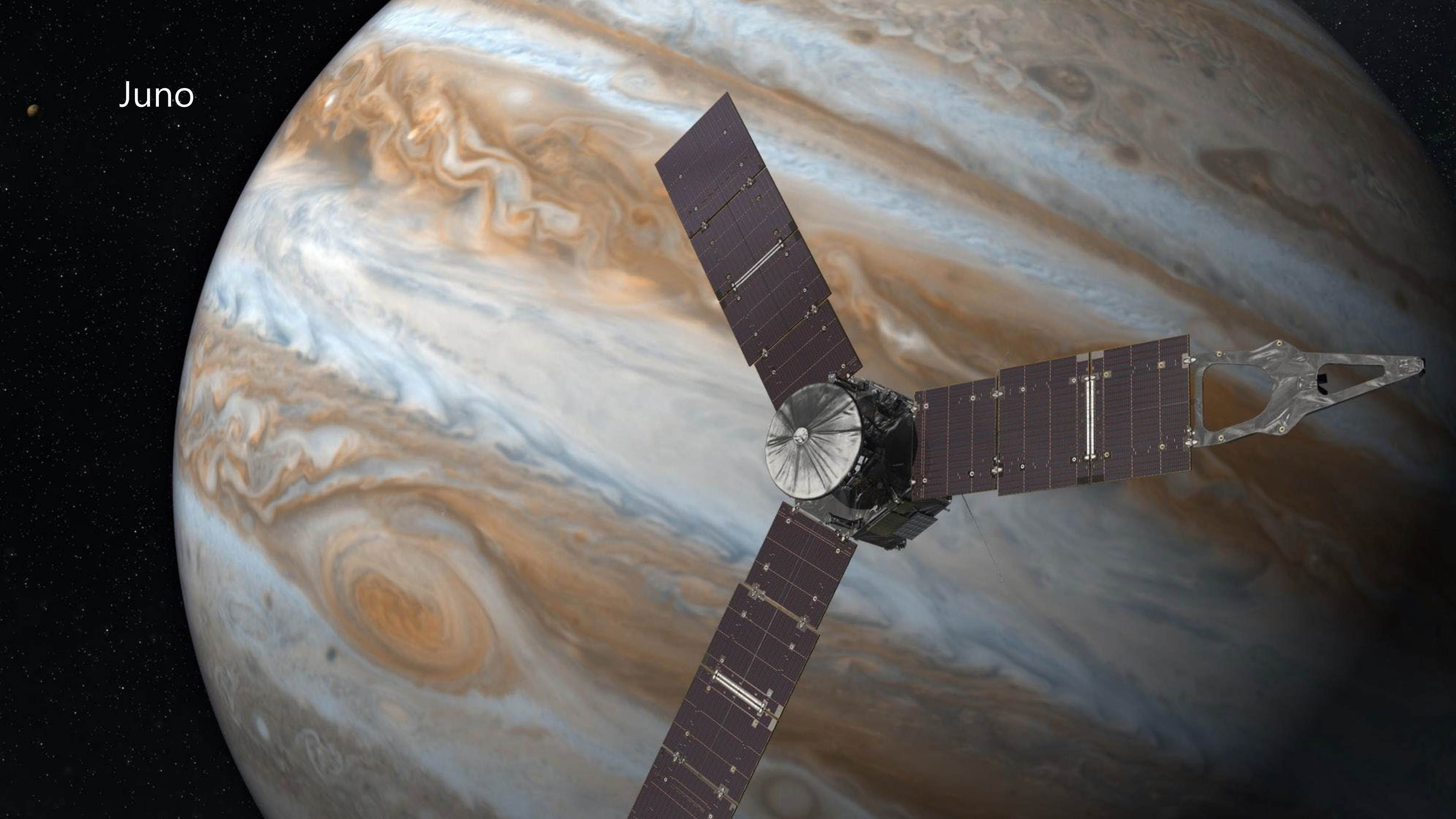

Insight

Insight

Seismometer deployment in early February



Juno



To design, build, and operate these spacecraft requires an enormous human undertaking.


Curiosity

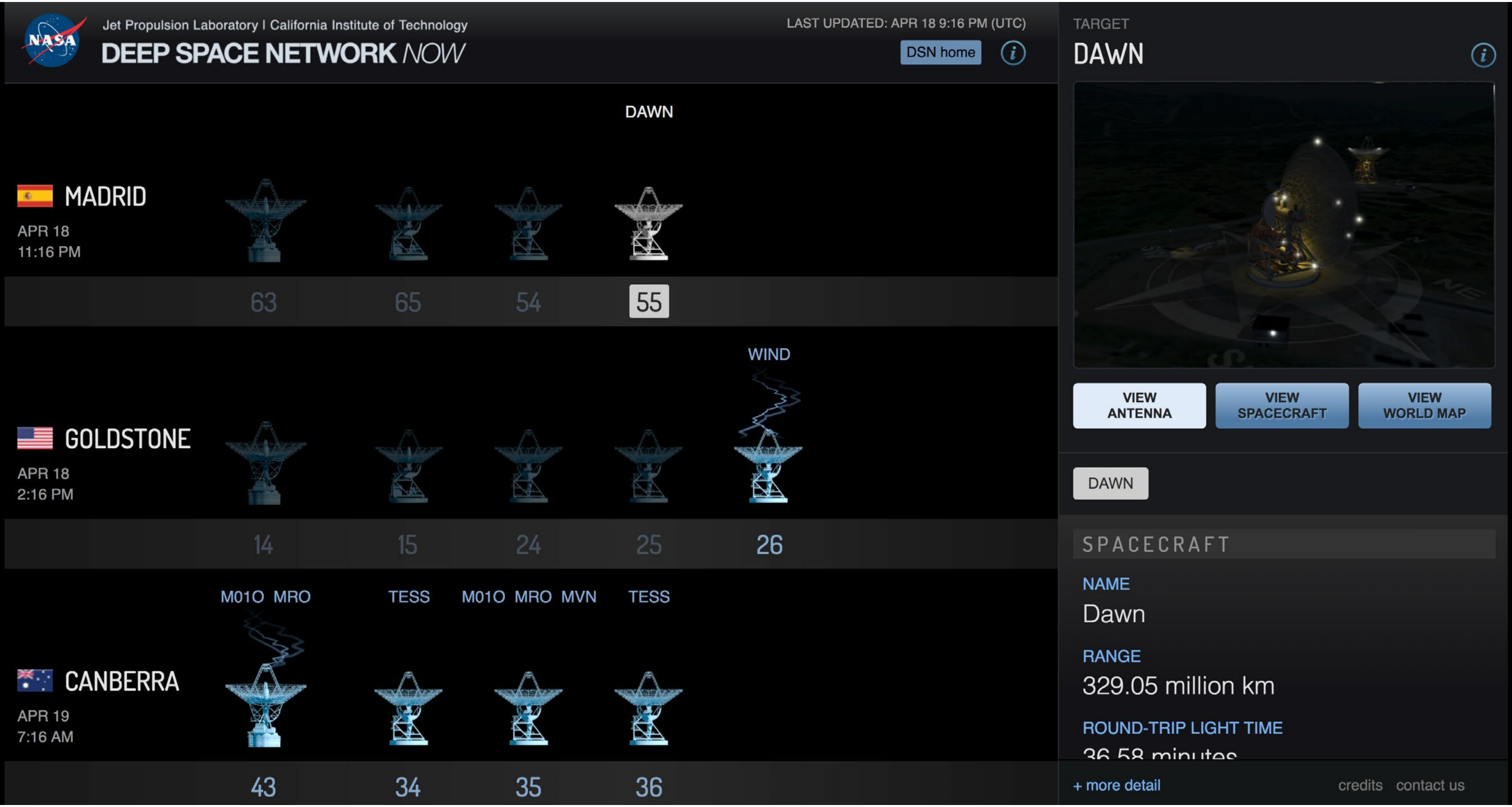
Juno

And with that, the complete tracking of systems, failures, and anomalous behavior of spacecraft.

This requires some data… quite a bit of it!

Charles Elachi Mission Control Center

DSN Now



DSN Now: https://eyes.nasa.gov/dsn/dsn.html

How can we as data scientists aid JPL in its mission of exploring the solar system?

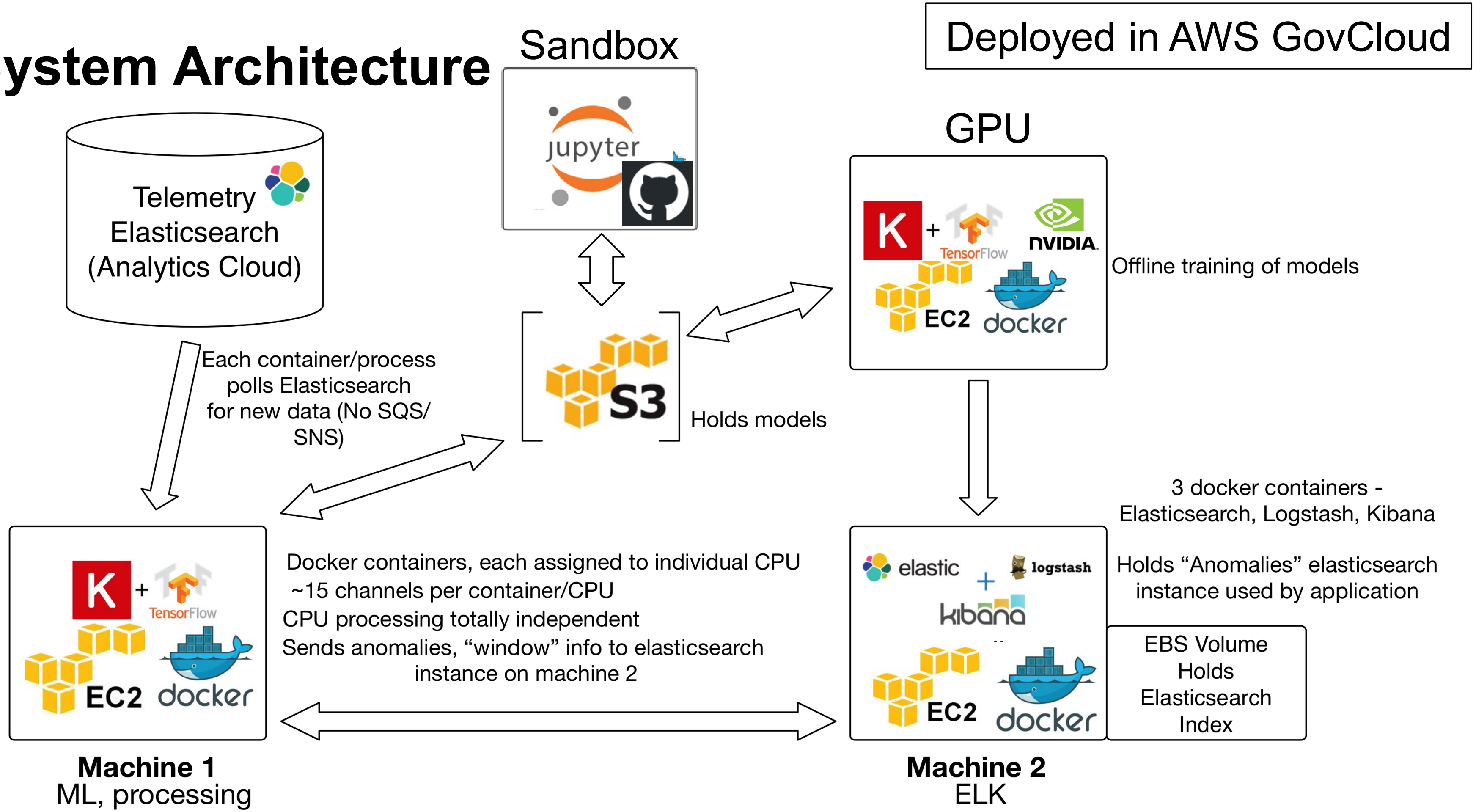
1. Introduction: the Jet Propulsion Laboratory

2. Anomaly Detection for SMAP and MSL

3. **Generalization of Telemanom Framework**

4. Wrap-Up

System Architecture



From Research to Anomaly Detection as a Service

- Research work yielded positive results and identified areas of improvement. Extending our SMAP implementation to MSL required a small amount of code generalization → identified this as an opportunity.

- Team currently engaged in development of Telemanom as an extensible framework for anomaly detection in time-series applications.

- Goals are the following:
  - Improve the reliability and maintainability of the Telemanom system
  - Allow users to play with the Telemanom system locally, then deploy to AWS S3 for production
  - Ensure the framework is extensible → one example is to allow researchers and data scientists to use their own error thresholding approaches within the rest of the Telemanom system
  - Provide anomaly detection as a service to missions and other applications on the laboratory, such as in detecting disturbances in network traffic (cyber-security)

Making Telemanom Extensible

- ModelTrainer:
  - Model implemented for SMAP is a shallow, two-layer LSTM network.
  - We've abstracted the training process to allow users to use their own model architectures and types within the broader Telemanom system.
    - Currently works with Keras + TensorFlow → goal to add Torch as well

- Errors:
  - Implements our non-parametric approach for detecting anomalies from sequences of model errors.
  - Code currently being abstracted so that others can write their own error thresholding approaches as a function or class that is inherited within Errors.

```
def get_anomalies(chan, e_s, y_test, z, window, eval_start_idx):
    """Find anomalous sequences of smoothed error values that are above error
    threshold (epsilon)

    Args:
        chan (str): channel id
        e_s (list): smoothed errors between y_test and y_hat values
        y_test (array): 2D test targets corresponding to true telemetry values
            at each timestep (n timesteps x n predictions at each timestep)
        z (float): number of standard deviations above mean corresponding to
            epsilon
        window (dict): holds arrays of window values and metadata
        eval_start_idx (int): indicates where new yet-to-be-evaluated test data
            begins in the window arrays


    Returns:
        i_anom (list): indices of errors that are part of an anomalous sequences
    """

                    .

                    .

                    .

    return i_anom
```

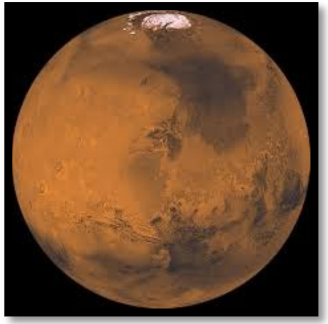
Making Telemanom Extensible

- Groups:
  - A new class that allows users to define how model inputs are defined and organized by spacecraft system, subsystem, subsystem group, etc.
  - Previously functionality hard-coded as a JSON → works but limiting.
  - New functionality will allow mission operators to dynamically adjust the groupings and model inputs in a sandbox → this kicks off a model training process, results later shown to operators.
- Server:
  - Previous implementation could only be run in production → models were trained in a sandbox, but the server could only be instantiated in the production environment.
  - New functionality allows users new of Telemanom to start a server locally and to test the system before deploying to production.

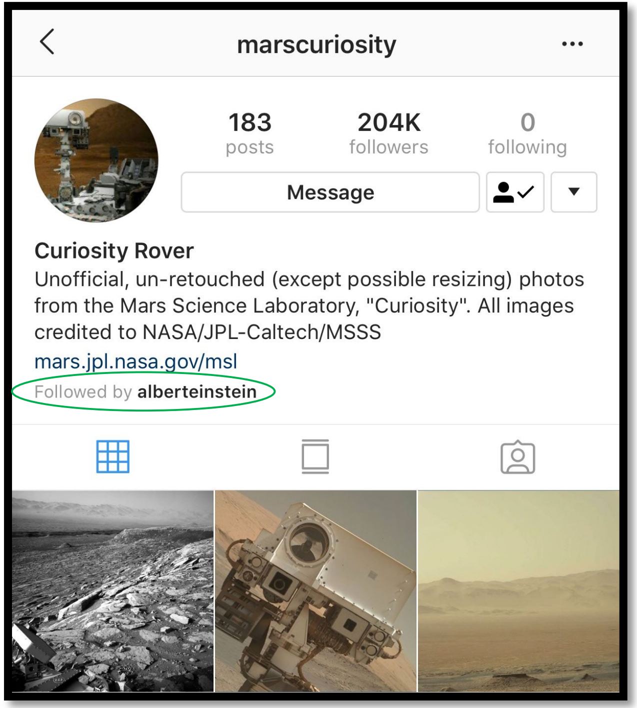
```
from telemanom.groups import Admin, Node

t = Node("Thermal")


t.add_child(
    Node("ripa_pressure",
         name="RIPA Pressure",
         children=[
             Node(
                 "THRM-0768",
                 name="THRM-P-RIPAGAS_A",
                 input="123B",
                 attributes={"yLabel": "psi", "model": "LSTM_A"}
             ),
             Node(
                 "ripa pressure subsystem",
                 name="RIPA pressure subsystem B",
                 children=[
                     Node(
                         "THRM-0832",
                         name="THRM-P-RIPAGAS_B",
                         input="123B",
                         attributes={"yLabel": "psi", "model": "LSTM_A"}
                     )
                 ],
             ),
         ]
    )
)
```

Selfies

Celebrities



Mars



Social Media

Telemanom as a Service

- The overall goal of the current generalization effort is to allow for Telemanom to exist as a service → goal is to become a core offering in our ground systems.

- Many applications where telemetry anomaly detection could be implemented:
  - Future missions with increased data rates and thousands of telemetry channels
  - Cyber-security applications to detect disturbances in network traffic

- Further development of the user interface to allow for a sandbox for operators to play with:
  - model inputs (e.g. which commands, EVRs should be considered – domain knowledge)
  - system groupings (e.g. which telemetry channels should be grouped, such as Thermal or Voltage channels)