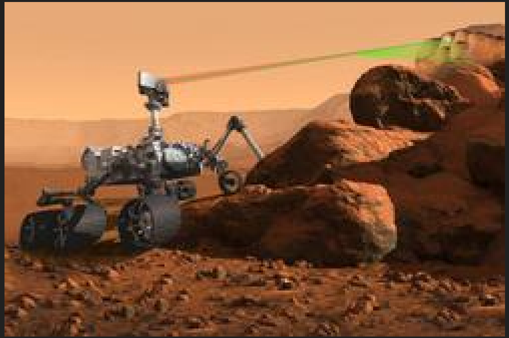
Automated Terrain Mapping of Mars

Team Strata:
Jorge Felix
Tsosie Schneider
Sean Baquiro
Matthew Enright

Mentor: Dr. Maggie Vanderberg

Surface of Mars
Credit: NASA

Our Sponsor



SuperCam Project [Credit NASA]

Dr. Ryan Anderson

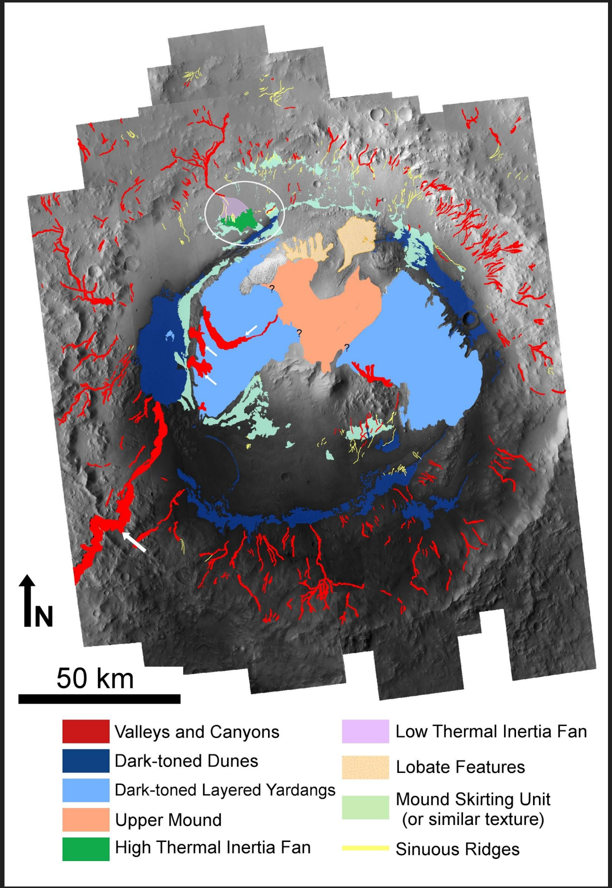
- Physical Scientist
- Research on Gale Crater
- Geologic Mapping and Characterization of Mars

USGS Astrogeology Science Center

- Innovative research on planetary cartography
- Develop software of planetary remote Sensing data

Problem Statement

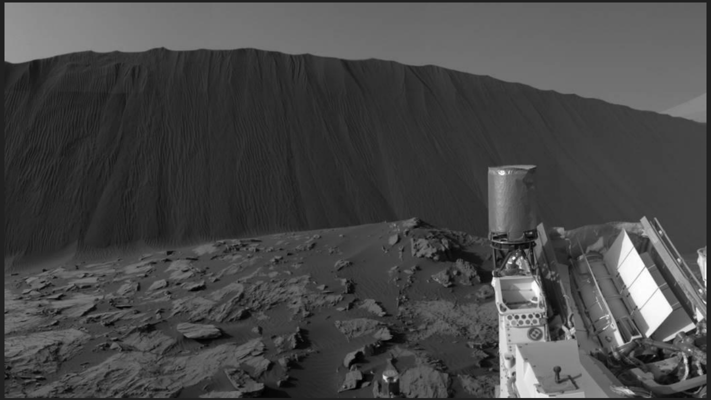
- An efficient approach to mapping terrains
- Manual Method occurs by hand
  - Time consuming
  - Inefficient
  - Inconsistent



Manually Mapped Image
Credit: Mars Journal

Importance

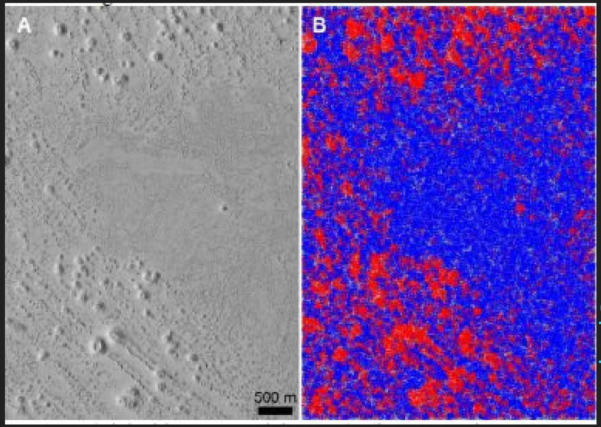
- History of Mars through geological processes
- Learn about planet’s formation
- Produce regional maps for potential landing sites
- NASA proposal



Dark Toned Dunes Credit NASA

Existing Solutions?

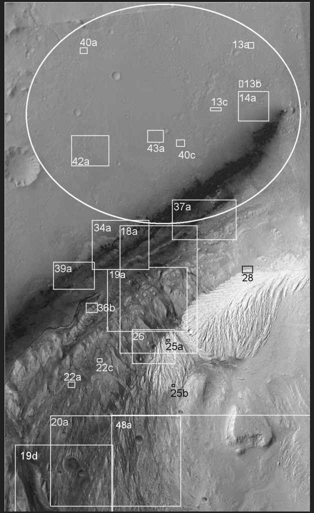
- No reliable automated terrain mapping algorithms
- Tool developed in U of A
  - Used a Convolutional Neural Network
  - Automated detection of impact craters on Mars

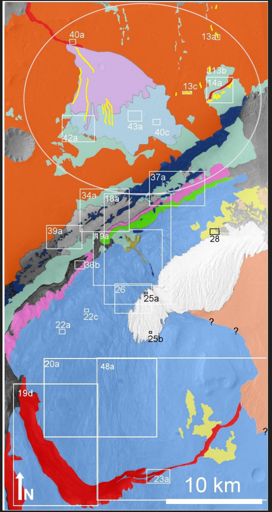


Credit: L. F. Palafox1 , A. M. Alvarez2 , C.W. Hamilton1 , Lunar and Planetary Laboratory, University of Arizona

Solution Overview

- Load JP2 images for analysis
- Train the Neural Network
- Produce annotated JP2 with marked terrains
- Simple command line interface



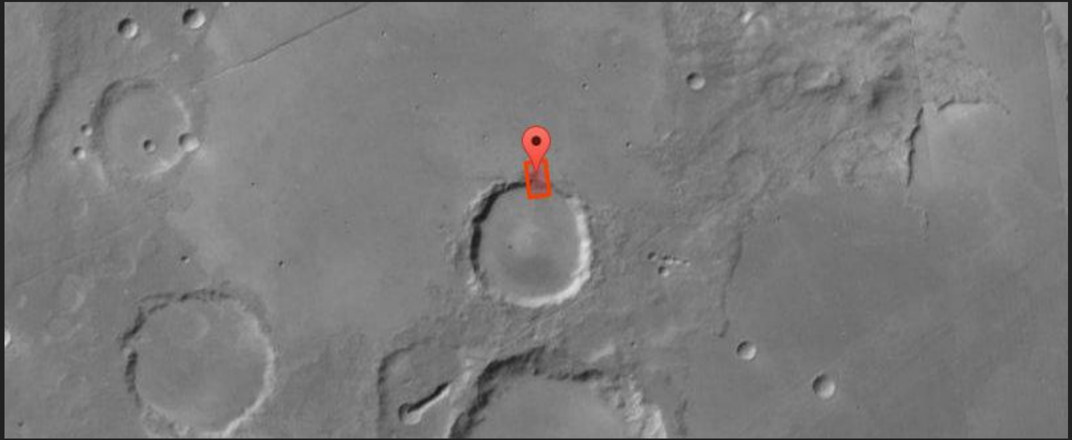


Credit: Mars Journal

Functional Requirements

- HiRISE will provide high resolution images and CTX will provide context images of Mars' surface

HiRISE

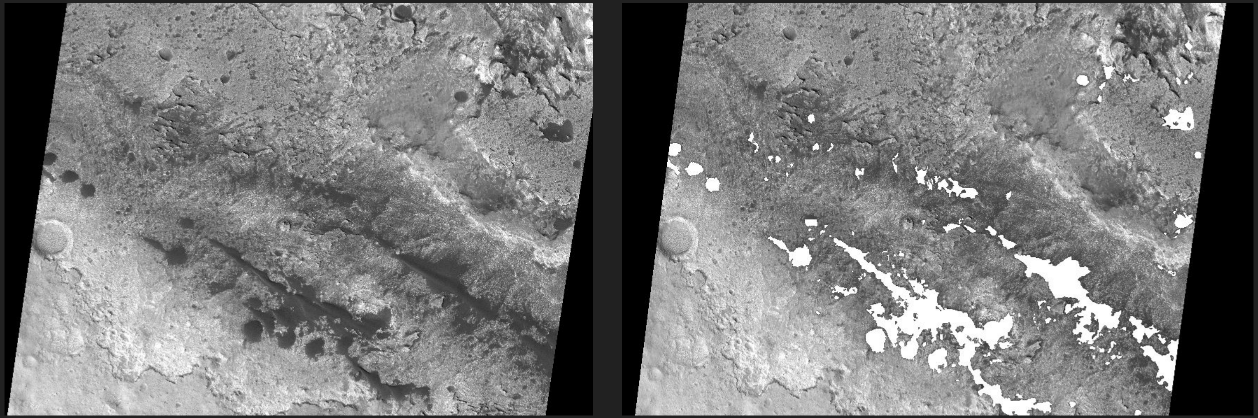


CTX

Credit: NASA/JPL/University of Arizona

Functional Requirements

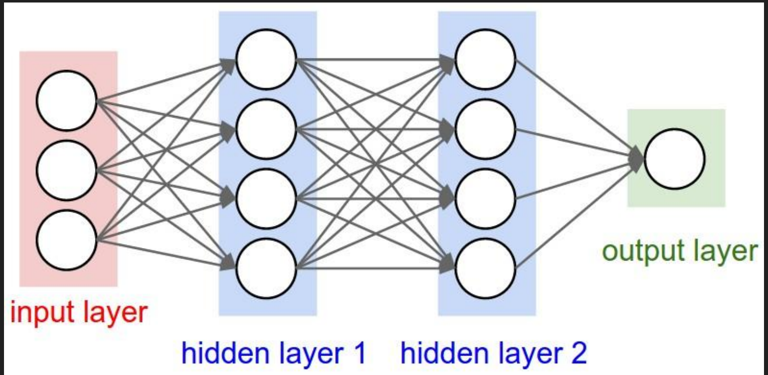
- Load and georeference multiple data sets
- Identify terrain types and features
- Map features across multiple input images



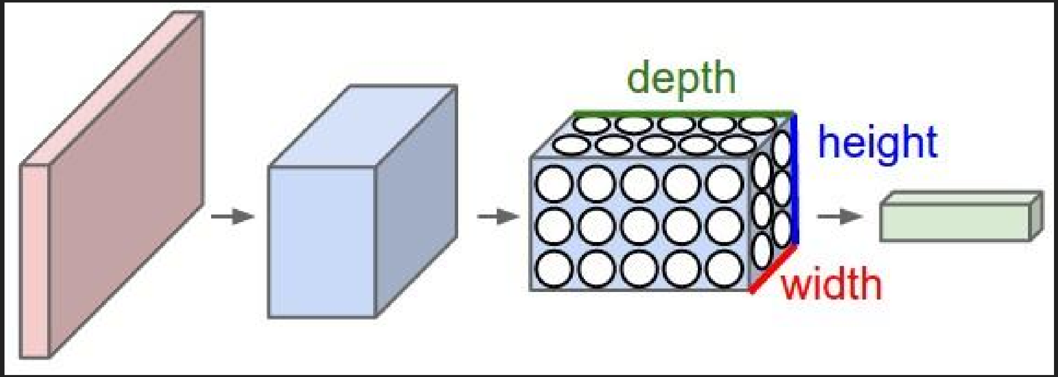
Credit: Ryan Anderson

Functional Requirements

- CNN's take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way.



Classic Neural Network

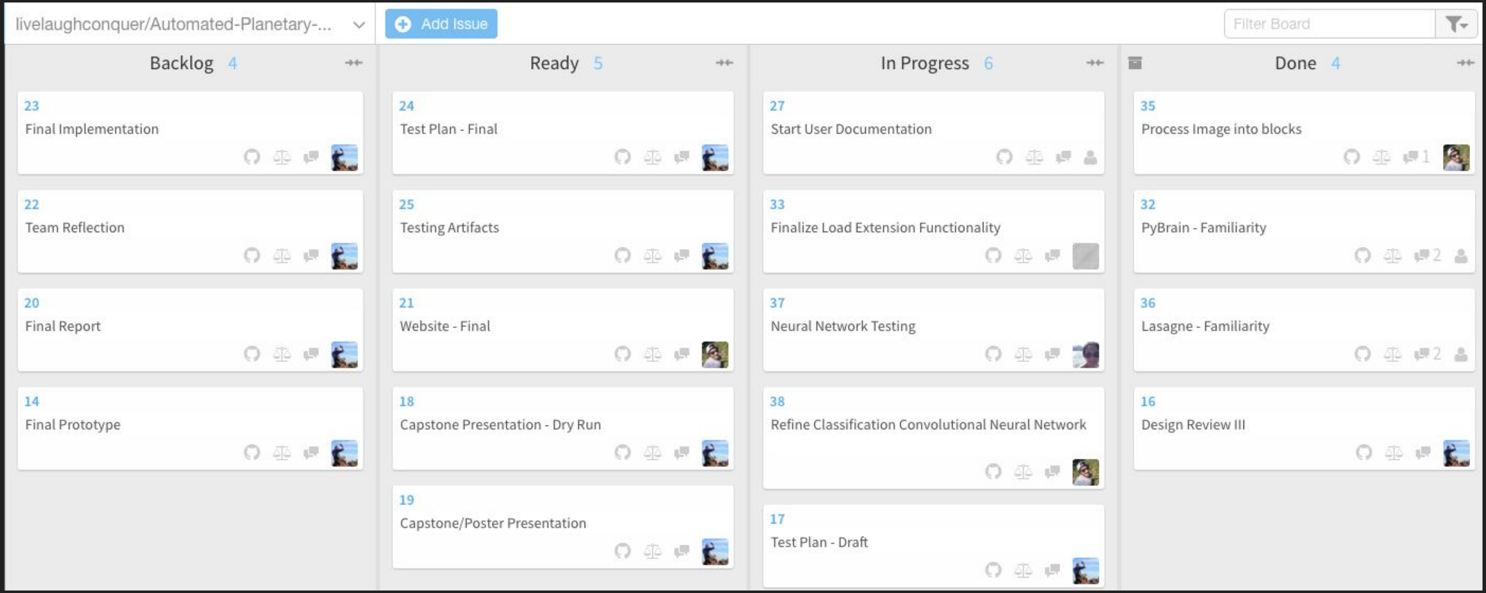


Convolutional Neural Network

[Credit: Stanford University]

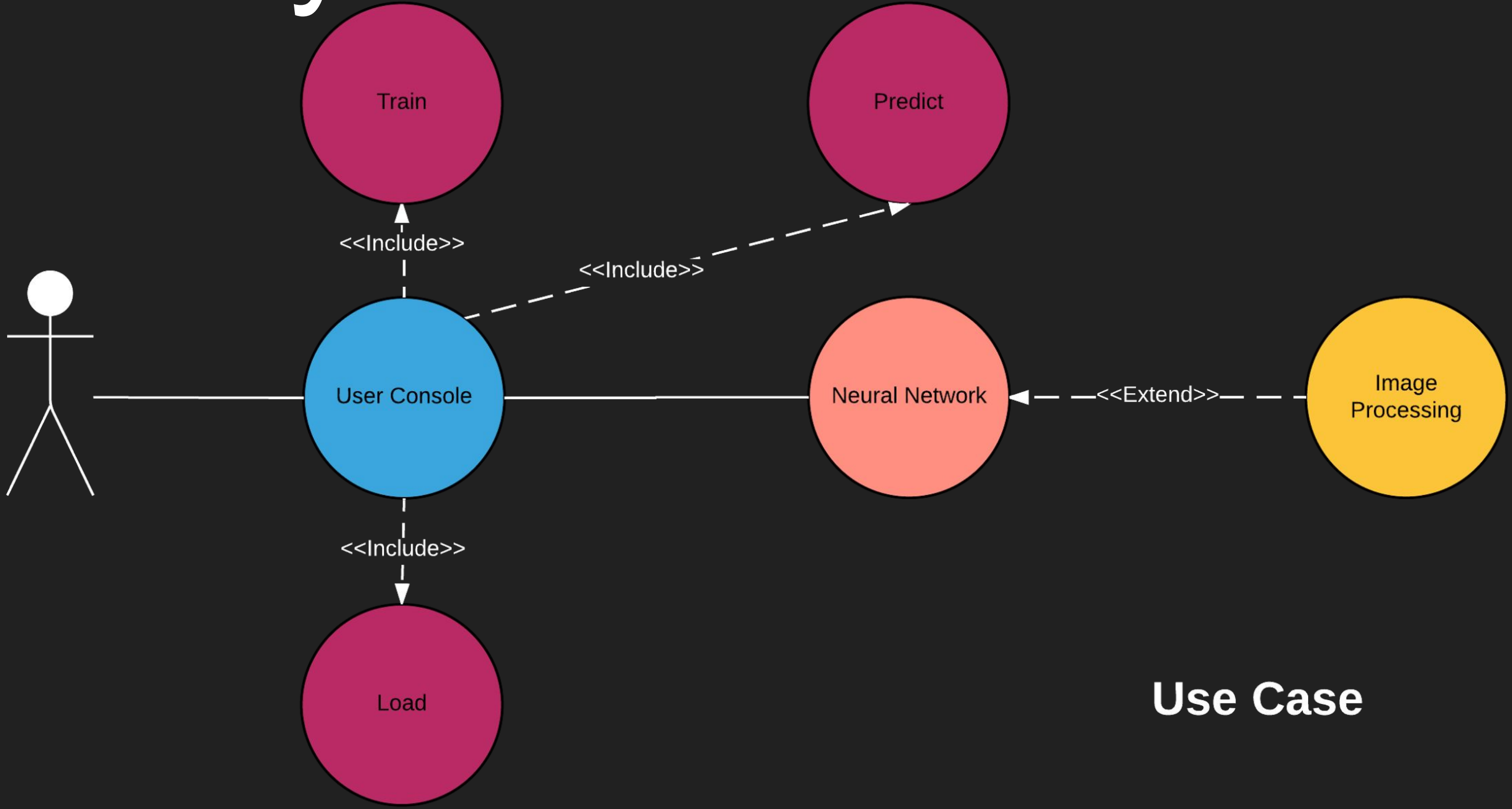
Development Methodology

- Agile Development Process (Scrum)
- Weekly meetings
- Waffle.io



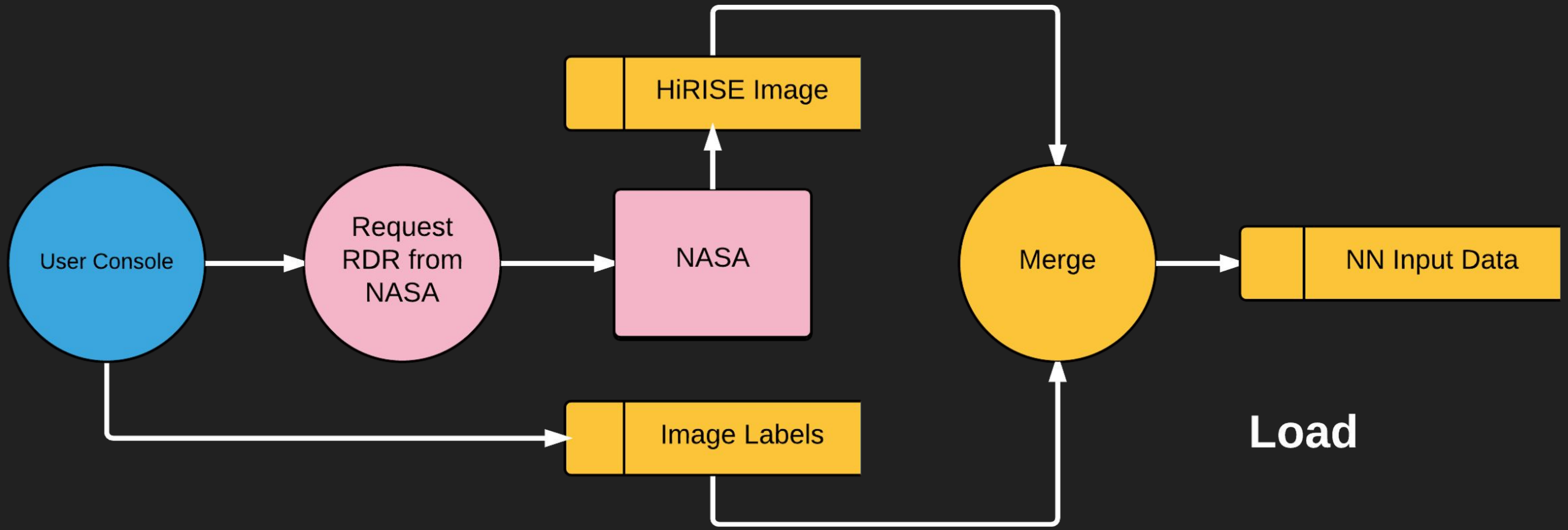
Waffle.io

Hybrid Architecture

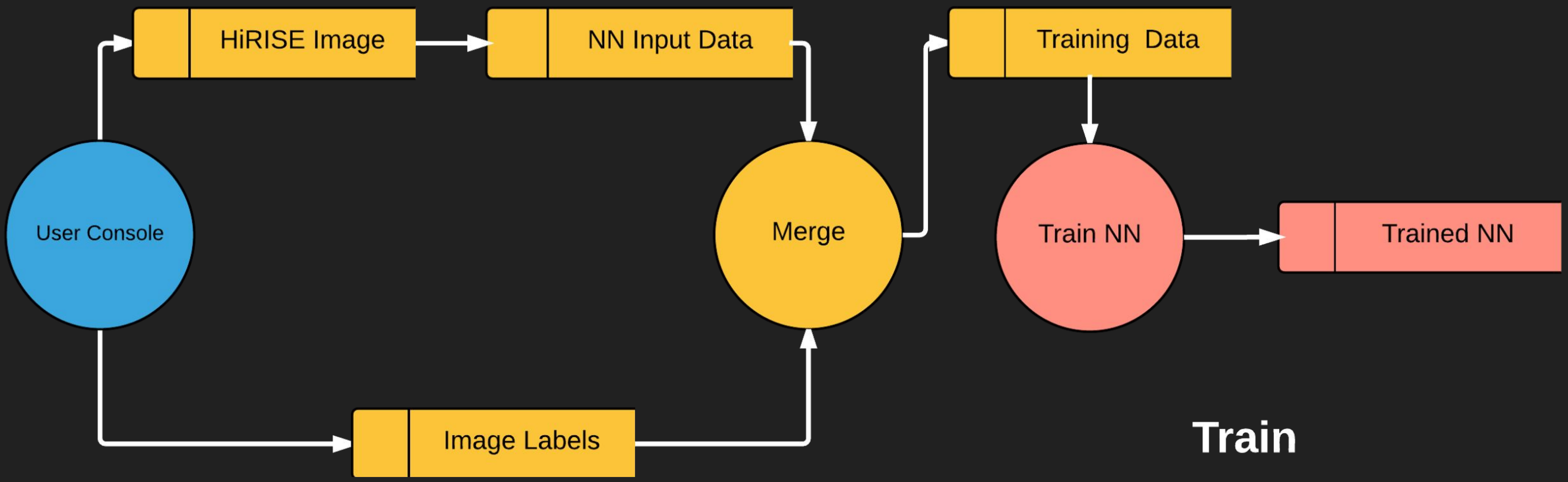


Use Case

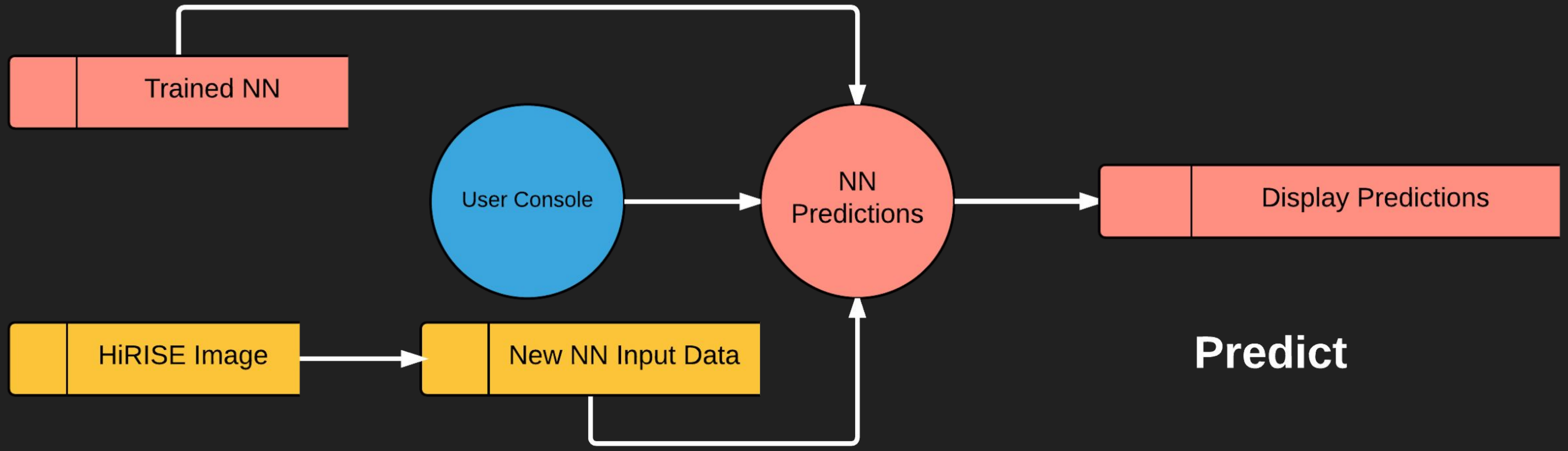
Architecture Dataflow



Architecture Dataflow



Architecture Dataflow

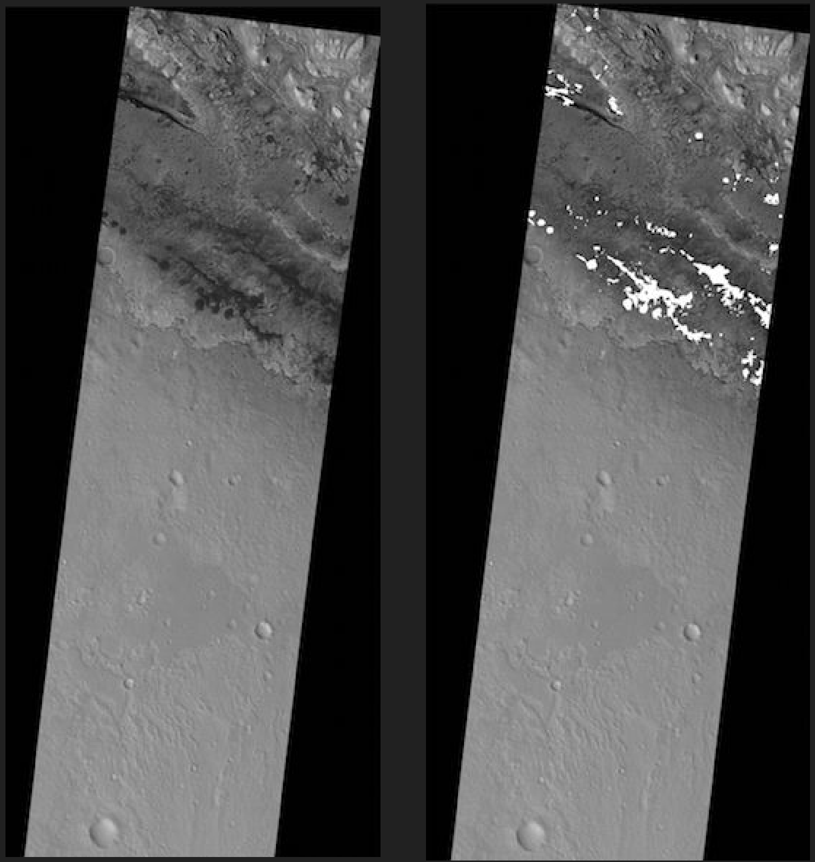


Implementation

Automated Terrain Mapping of Mars Project Schedule				
#	Task	Start Date	Duration(Days)	End Date
1	Implementation	1/19/16	81	4/9/16
1.1	Process JP2 Image	1/19/16	9	1/28/16
1.2	Extract Image Data	2/1/16	11	2/12/16
1.3	Integrate C++ and Python	2/1/16	11	2/12/16
1.4	Process Training Data	2/12/16	17	2/29/16
1.5	Train Neural Network	3/1/16	39	4/9/16
1.6	Process Image Data into JP2	3/20/16	10	3/30/16
2	Testing	4/10/16	25	5/5/16
3	Documentation/User Guide	4/10/16	25	5/5/16
4	UGRAD Presentation	4/29/16	1	4/29/16
Automated Terrain Mapping of Mars Project Schedule				

Implementation

1.1 JP2 image processing
1.2 Image data extraction
1.3 C++/Python Integration
1.4 Training image data processing

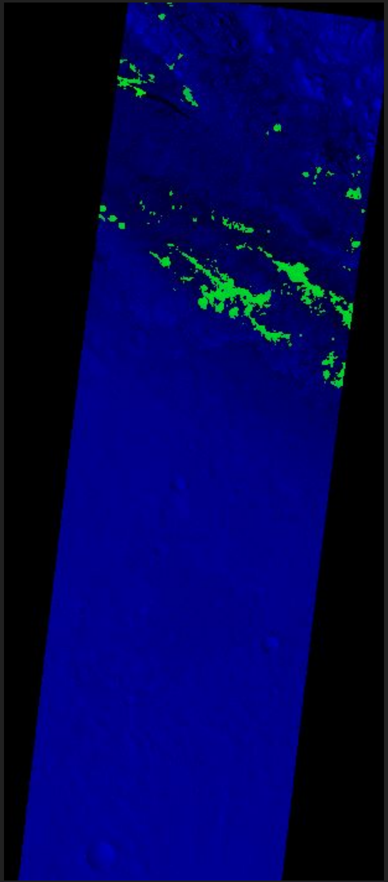


Test image (left)
Training image (right)

Implementation

Pre-processing image data extraction output

- blue band = original test data
- green band = training data



Implementation

1.5 Neural Network Training

- **Create**
- Train
- Predict

```
def convolutionalNeuralNetwork(epochs):
    net = NeuralNet(
        layers=[ #three layers: Input, hidden, and output
            ('input', layers.InputLayer),
            ('conv1', layers.Conv2DLayer),
            ('pool1', layers.MaxPool2DLayer),
            ('conv2', layers.Conv2DLayer),
            ('pool2', layers.MaxPool2DLayer),
            ('conv3', layers.Conv2DLayer),
            ('pool3', layers.MaxPool2DLayer),
            ('hidden4', layers.DenseLayer),
            ('hidden5', layers.DenseLayer),
            ('output', layers.DenseLayer),
        ],
```

Convolutional Neural Network

Implementation

1.5 Neural Network Training

- Create
- **Train**
- Predict

```
D:\Documents\Automated-Planetary-Mapping-of-Mars\Neural Network>python convoluti
onal_NN.py train test2.tif train2.tif --epochs=5
Training network....
test2.tif
train2.tif
Loading images....

Image dimensions:
3144 11543
 8 bit

Shape of test image data followed by train image data:
(35280L, 1L, 32L, 32L)
(35280L,)

Number of success sand dune blocks followed by negative image blocks:
6041
29239
```

Network Training Output

Implementation

1.5 Neural Network Training

- Create
- Train
- **Predict**

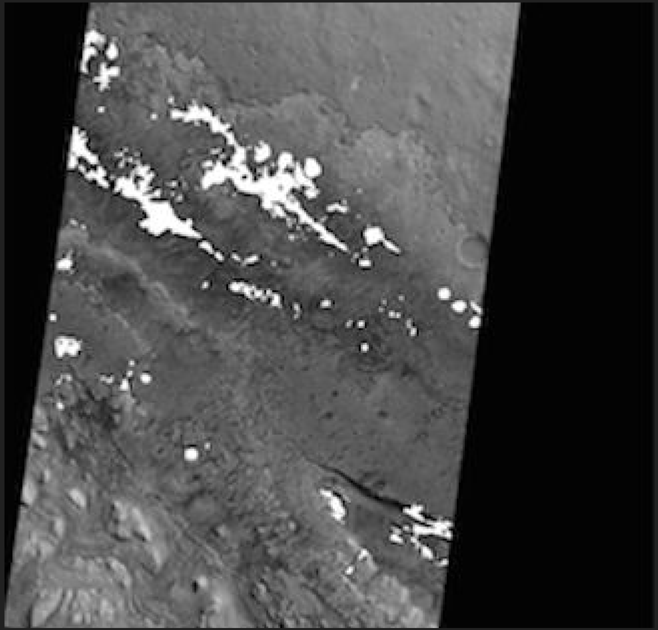
```
D:\Documents\Automated-Planetary-Mapping-of-Mars\Neural Network>python convoluti
onal_NN.py predict train.tif
3144 11543
 8 bit
Loading trained network data....

Making predictions....

Dune blocks detected followed by negative blocks.
78 35202

Adding predictions to input image....

Writing image to directory....
Predictions done.
```

Prediction output generated

Implementation

1.6 Output data processing

1.1 JP2 image processing
1.2 Image data extraction
1.3 C++/Python Integration
1.4 Training image data processing



Mapped JP2 Image
(features in white)

Testing

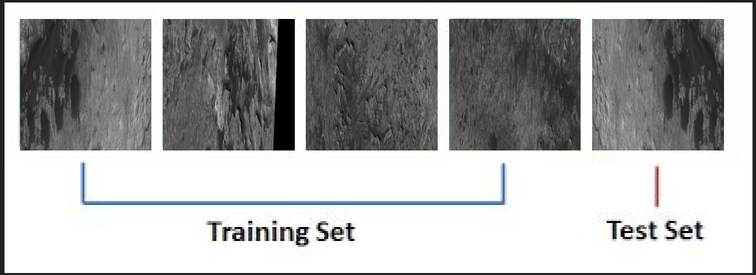
- Unit Testing
- Cross validation
- Usability testing

Unit Testing

- PyUnit framework
  - Image processing functions
  - Neural network creation functions
  - Python helper functions

10-fold Cross Validation

- 10-Fold Cross Validation
  - Divided data into 10 sets
  - Train on 9 sets
  - Validate on 1
  - Detect and prevent overfitting



Example 5-fold cross validation

Usability Testing

- User study on console interface

Dr. Ryan Anderson [Credit: USGS]

Command Prompt

```
D:\Documents\Automated-Planetary-Mapping-of-Mars\Neural Network>python convoluti
onal_NN.py --help
Usage: convolutional_NN.py [OPTIONS] COMMAND [ARGS]...

  This program is designed to allow the user to load image data, train a
  neural network on the image data,  or make predictions based on the
  stored neural network data.

  Commands:

  load: Loads image data. Input test and train file as an argument.

  Example: python convolutional_NN.py load testFile.tif trainFile.tif

  train: Input testFile trainFile, and number of epochs to train data.
  Loads image data and trains the convolutional neural network to  detect
  sand dunes. Saves trained network on a pickle file.

  Example: python convolutional_NN.py train testFile.tif trainFile.tif
  --epochs=10

  predict:Using existing trained network pickled data, make predictions  on
  pickle data. Input image as an argument.

  Example: python convolutional_NN.py predict inputFile.tif

Options:
  --help  Show this message and exit.

Commands:
  load     Loads and prints image specs.
  predict  Make predictions on input image.
  train    Train convolutional neural network.

D:\Documents\Automated-Planetary-Mapping-of-Mars\Neural Network>_
```

Challenges and Risks

Challenges

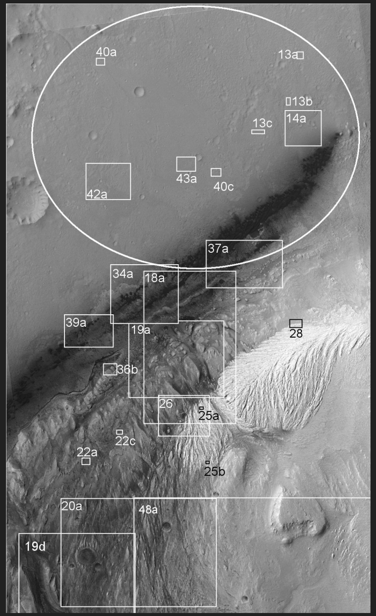
- Installation problems (Boost, Theano, Lasagne)
- Lack of physical memory

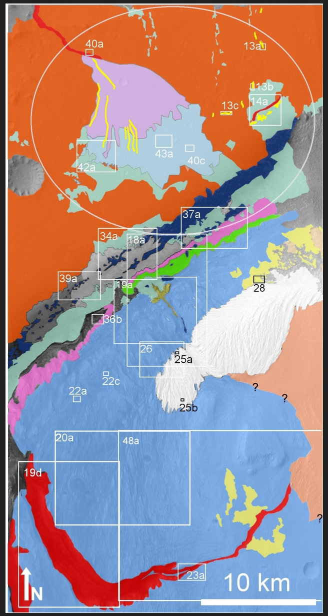
Risks

- Higher end machine requirement poses a risk for users with older machines.

Conclusion

- Automating the annotation process
- Taking in an orbital data set with a terrain type of interest
- Applying a Neural Network
- Produce results as a color-coded image





[Credit: Mars Journal]

Questions?