Classifying images of stars & galaxy and asteroids using Convolutional Neural Networks

Authors

Dongwei Sun
UC Riverside
900 University Ave,
Riverside, CA 92521

Ponmanikandan Velmurugan
UC Riverside
900 University Ave,
Riverside, CA 92521

Ivan Neto
UC Riverside
900 University Ave,
Riverside, CA 92521

Julian Beaulieu
Texas A&M University
400 Bizzell St,
College Station, TX 77843

Kai-Chun Chen
UC Riverside
900 University Ave,
Riverside, CA 92521

LLNL-PRES-XXXXXX
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

Team 4 Introduction

Dongwei Sun
UC Riverside
900 University Ave, Riverside, CA 92521

Ponmanikandan Velmurugan
UC Riverside
900 University Ave, Riverside, CA 92521

Ivan Neto
UC Riverside
900 University Ave, Riverside, CA 92521

Kai-Chun Chen
UC Riverside
900 University Ave, Riverside, CA 92521

Julian Beaulieu
Texas A&M University
400 Bizzell St, College Station, TX 77843

Outline

Team 4 Introduction

Background

Mini Challenge

Major Challenge

Summary

Acknowledgement

Outline

Team 4 Introduction

Background

Mini Challenge

Major Challenge

Summary

Acknowledgement

Introduction of Machine Learning

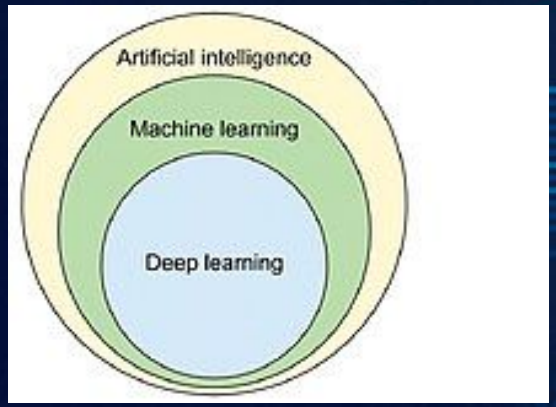


Fig.1

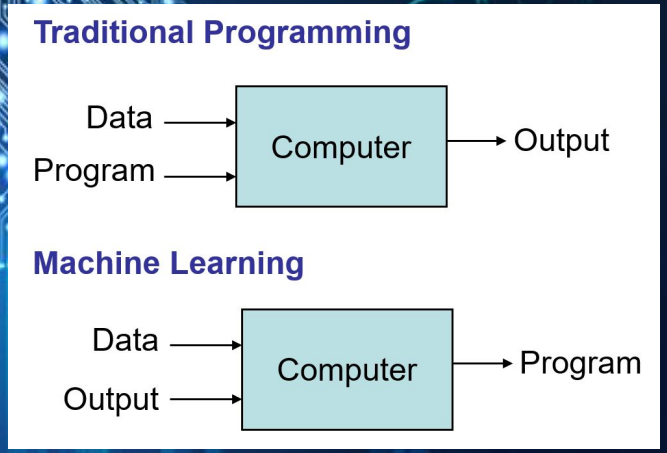


Fig.2

	Supervised Learning	Unsupervised Learning
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

Fig.3

Supervised Learning

- Apply a prediction function to a feature representation of the image to get the desired output:

f( ) = “fruit”

f( ) = “animal”

$y = f(\mathbf{x})$

output — prediction function — Image feature

Fig.4

Unsupervised Learning

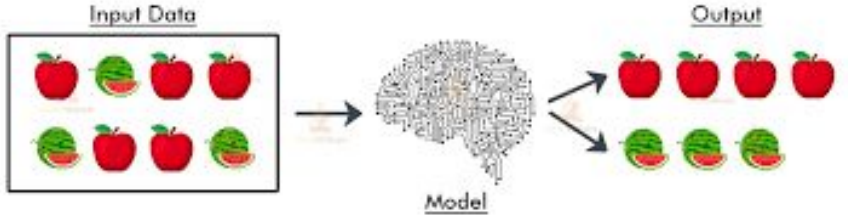
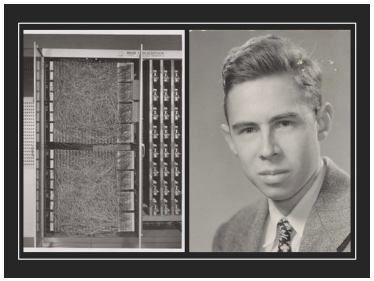


Fig.5

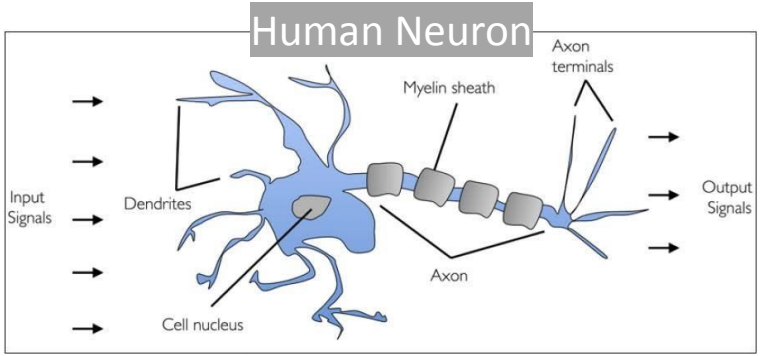
Neural Networks

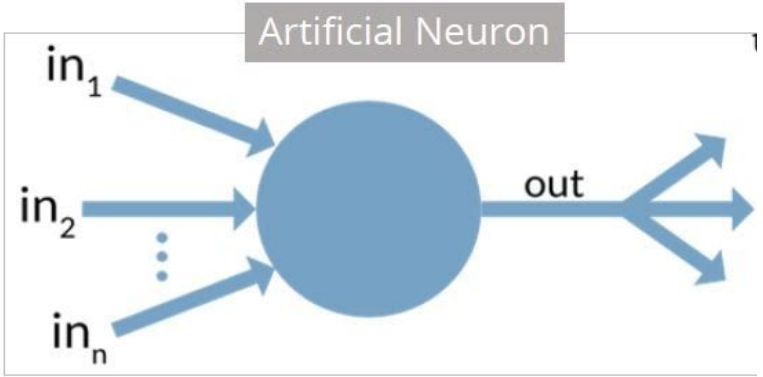
- ➔ Mimic the operations of a human brain to recognize relationships between vast amounts of data
- ➔ Neuron - Basic computation unit [input - process - output]
- ➔ Activation Function - Defines the output for an input
  - ◆ Linear
  - ◆ ReLU
  - ◆ Heaviside
- ➔ Perceptron - Simplest type of Neural Network - Classifier



Left: **Mark I Perceptron machine -** Image Recognition
Right: **Dr. Frank Rosenblatt**
PhD Experimental Psychology, Cornell, 1956

Source:
https://machinelearningknowledge.ai/timeline/perceptron-neuron-that-can-learn/

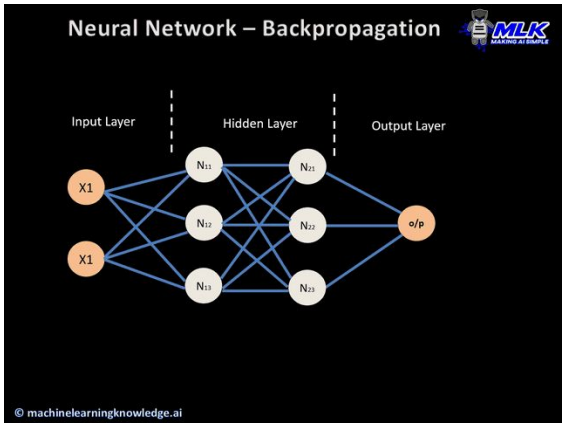




Source: https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron

Neural Networks

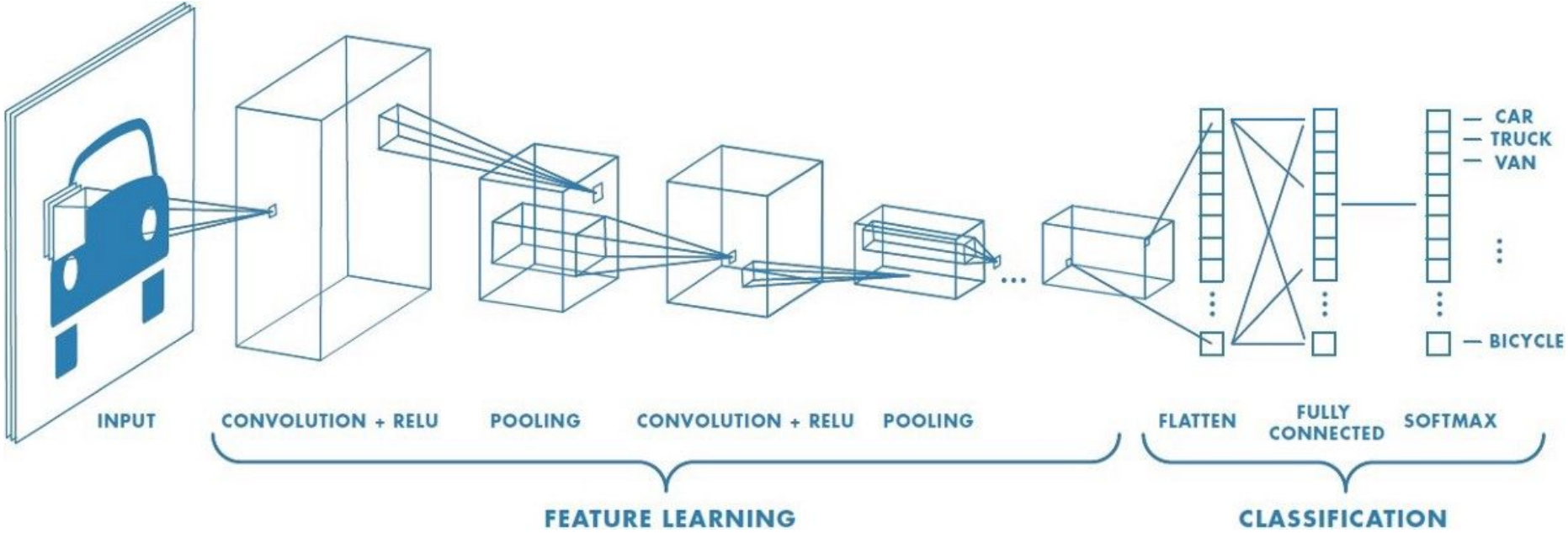
- ➔ Multiple perceptron layers (MLPs) - Welcomes non-linearity
- ➔ Learning with Forward and Backward propagations
- ➔ Forward Pass - Make predictions
- ➔ Backward Pass - Adjust weights with observed loss



Source: https://machinelearningknowledge.ai/animated-explanation-of-feed-forward-neural-network-architecture/

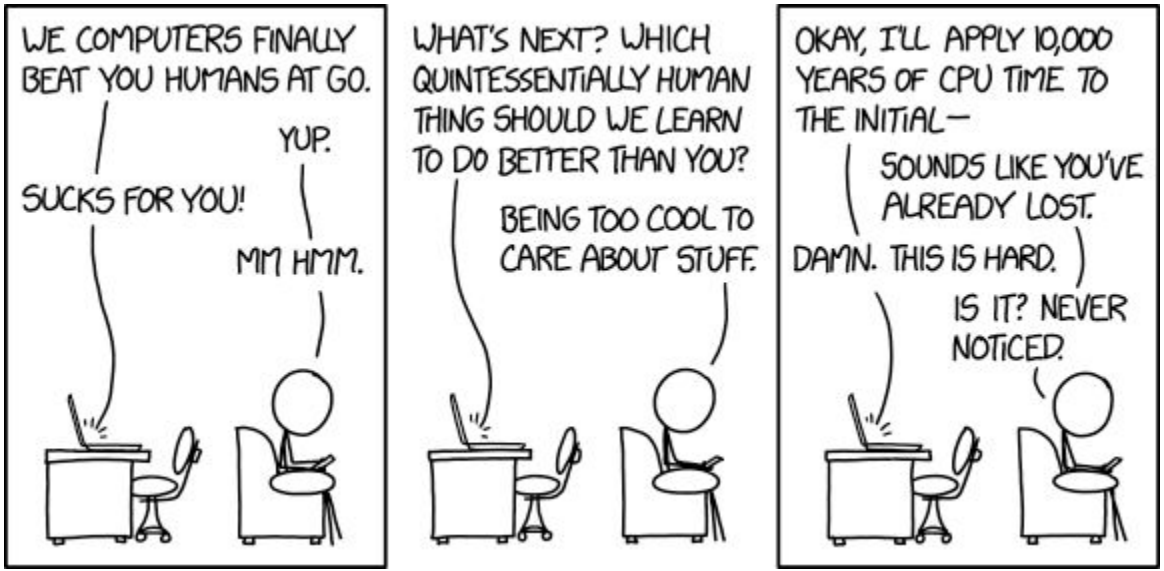
Source: https://xkcd.com/1838/

Convolutional Neural Network



Source: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Cannot Model Everything



Source: https://xkcd.com/1875/

Outline

Team 4 Introduction

Background

Mini Challenge

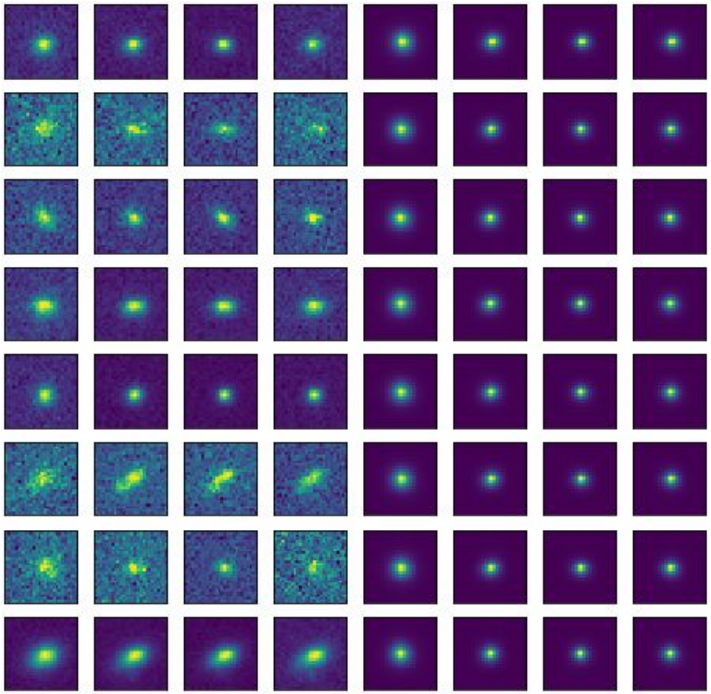
Major Challenge

Summary

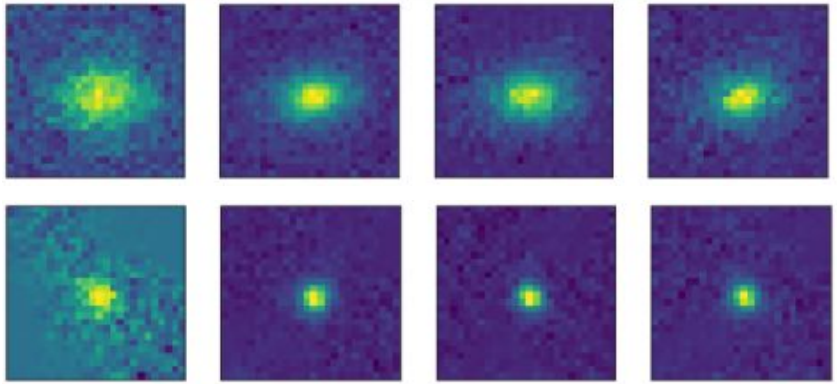
Acknowledgement

Mini Challenge - Intro

Star vs. Galaxy Classification



- ~ 34,000 images
- 8 channels per image
- First 4 channels seem to be more indicative and rich.
- Last 4 channels might have some hidden features that a model could detect



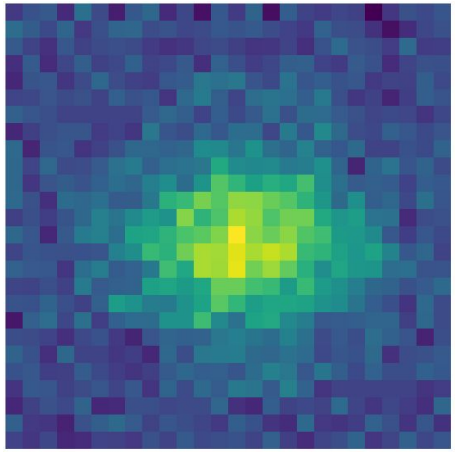
Class = 0
galaxy

Class = 1
star

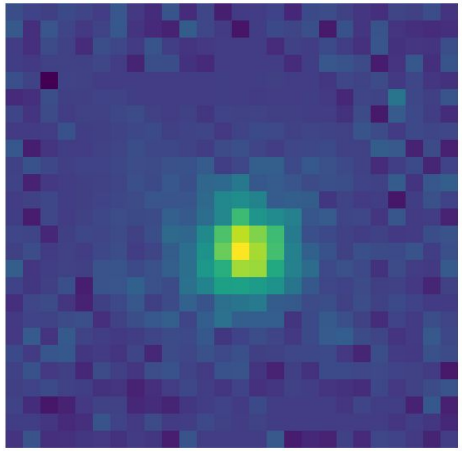
Mini Challenge - Intro

Aim

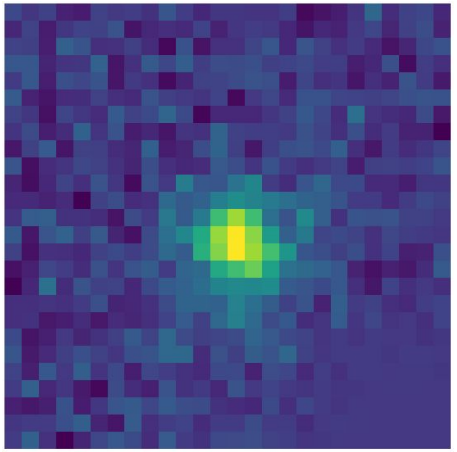
Differentiate between star and galaxy 26x26 images using Deep Learning Models.



Class = ???

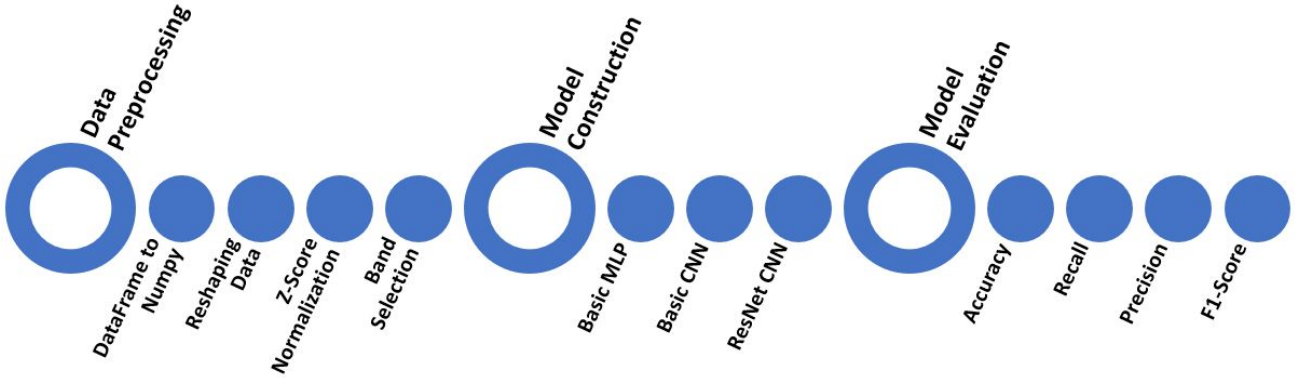


Class = ???



Class = ???

Mini Challenge - Methodology



Model Construction

- We decided on three different models, to see which one would perform the best on this data set.

Data Preprocessing

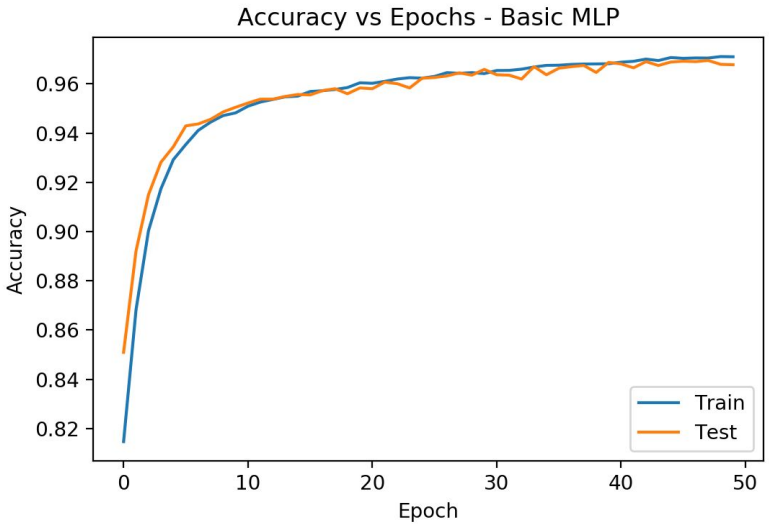
- We had to convert Pandas DataFrame to several Numpy arrays
  - Reshaped data to (26, 26)
- Normalized using z-score
- Selected all 8 bands

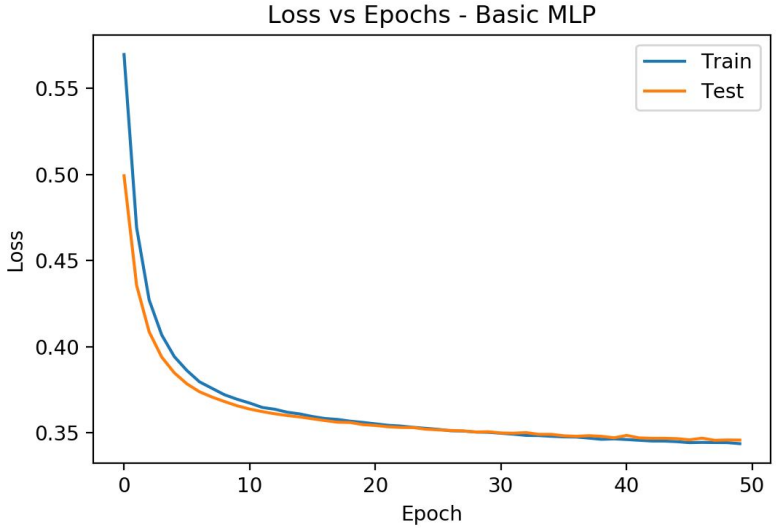
Model Evaluation

- Accuracy, Precision, and Recall scores (both micro and macro)
- F1 Score and classification reports for every model

Mini Challenge - Results

Accuracy & Loss vs Epoches - Multi-Layer Perceptron Model

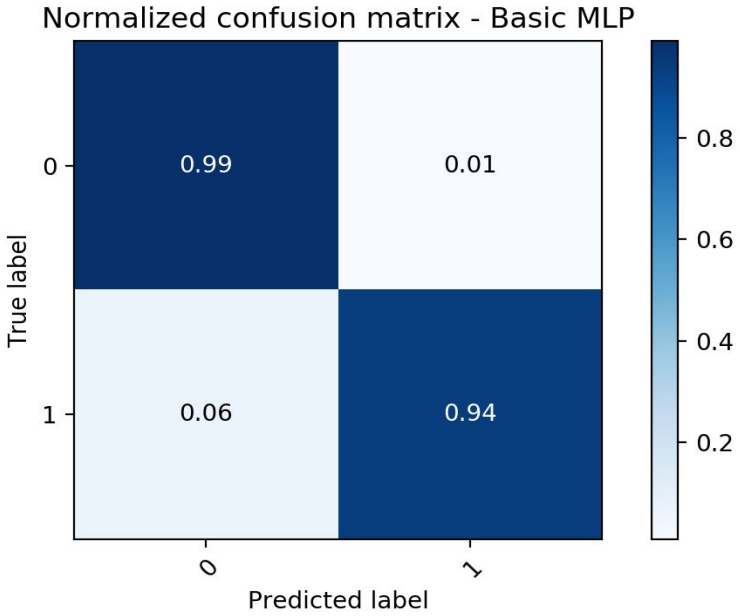


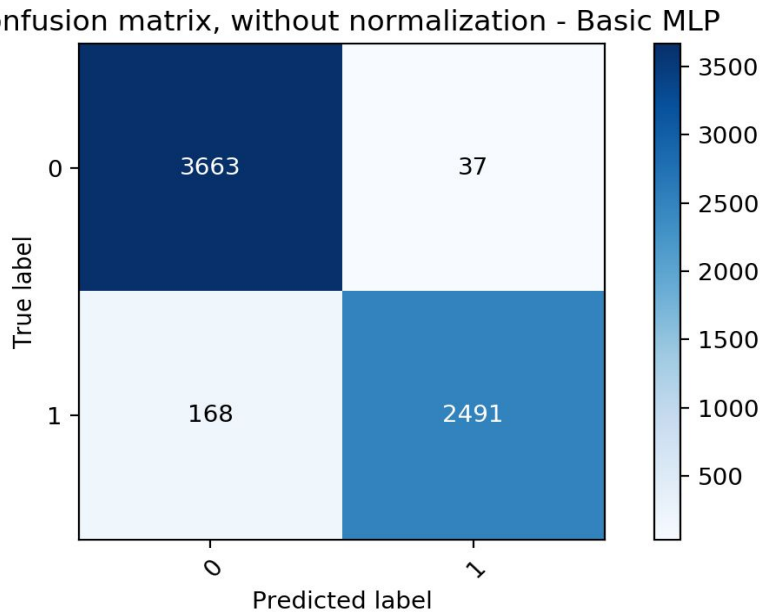


Total training time: 340.37s (5.67m)
Learning rate: 0.01
Batch size: 64

Mini Challenge - Results

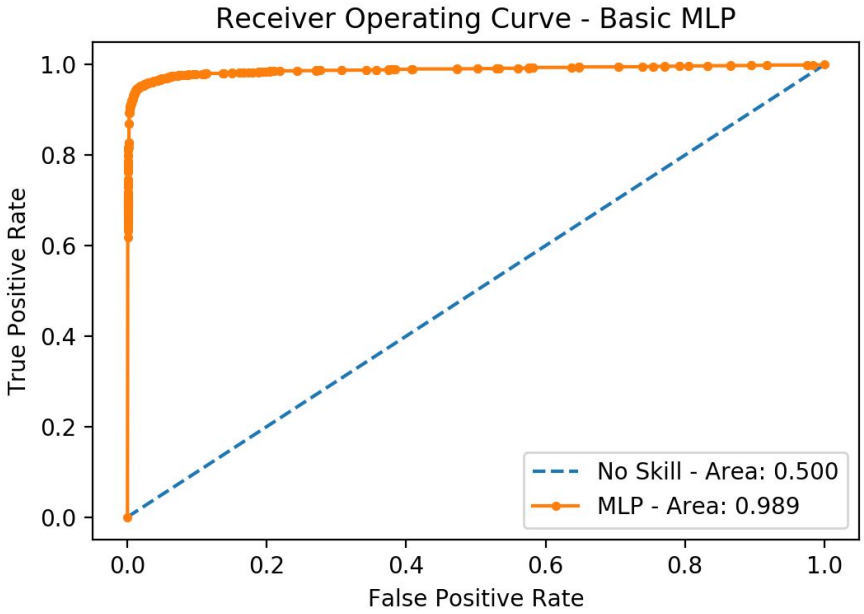
Confusion Matrix - Multi-Layer Perceptron Model

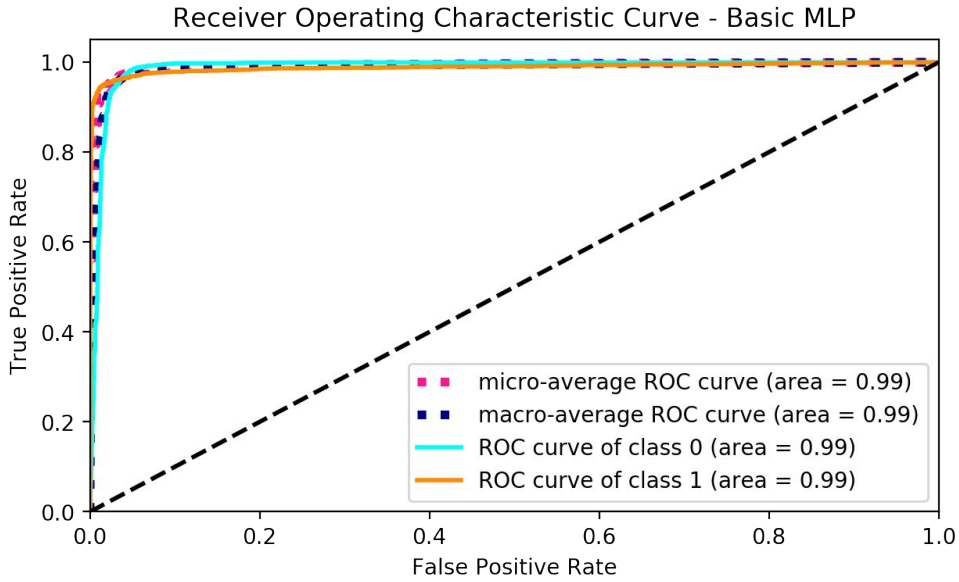




Mini Challenge - Results

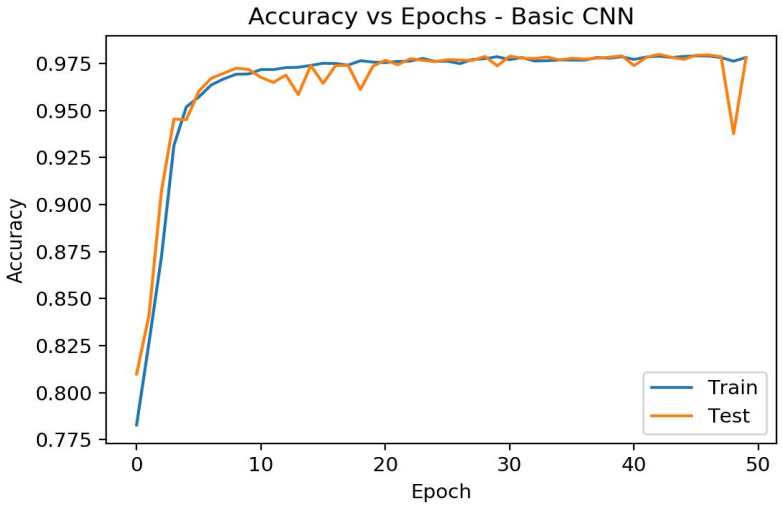
ROC Curve - Multi-Layer Perceptron Model

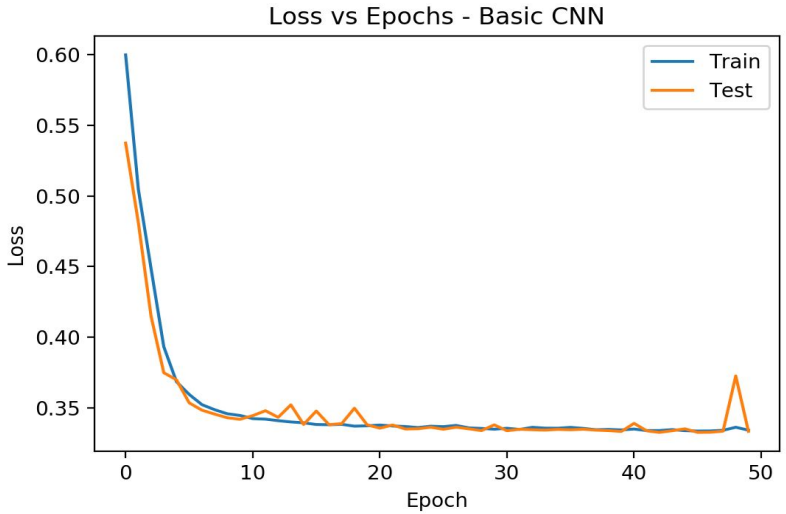




Mini Challenge - Results

Accuracy & Loss vs Epoches - Convolutional Neural Network

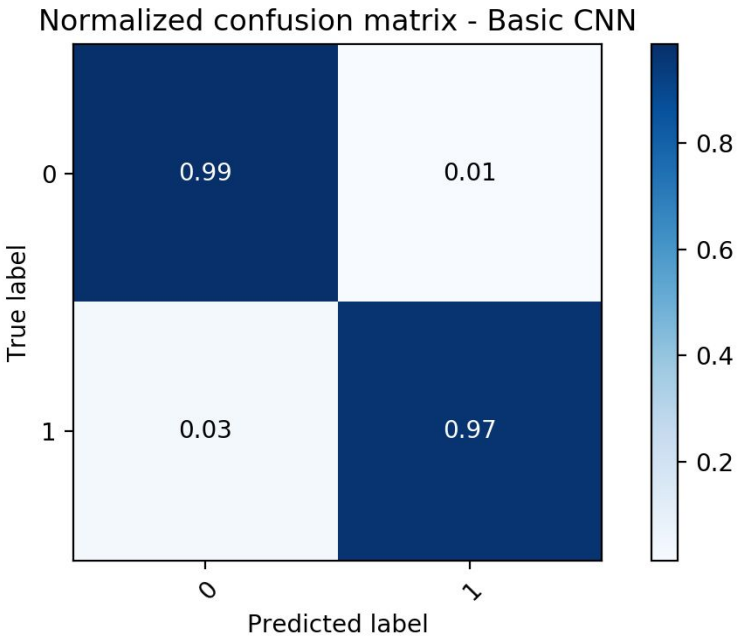


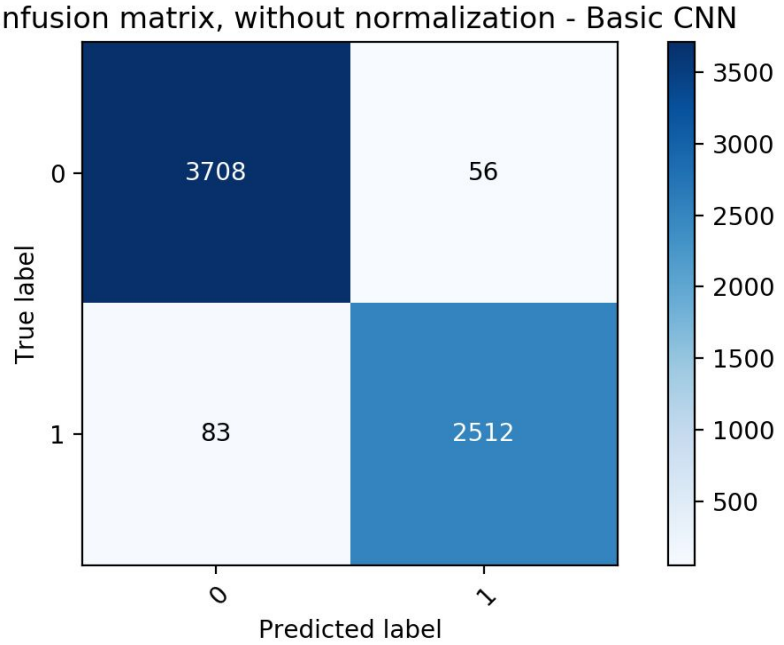


Total training time: 820.72s (13.68m)
Learning rate: 0.01
Batch size: 64

Mini Challenge - Results

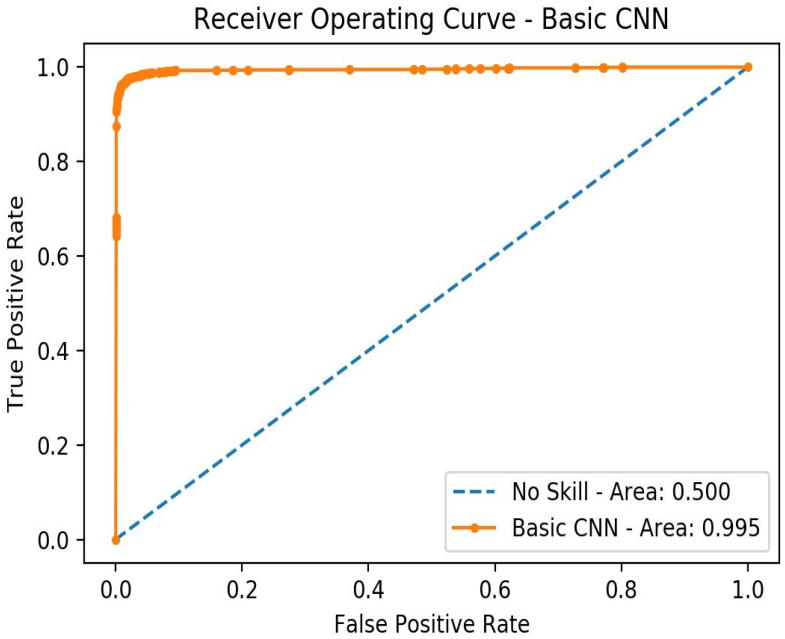
Confusion Matrix - Convolutional Neural Networks

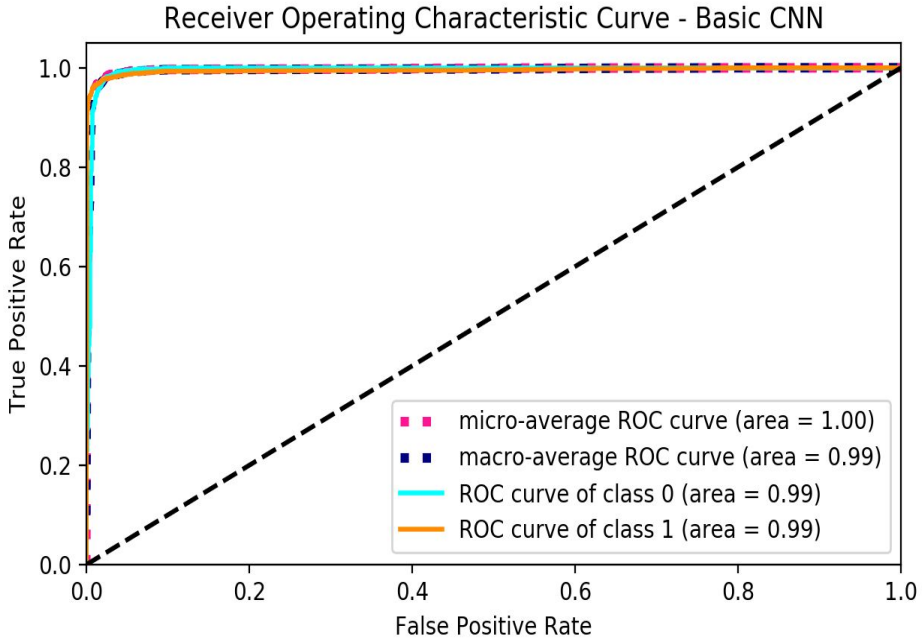




Mini Challenge - Results

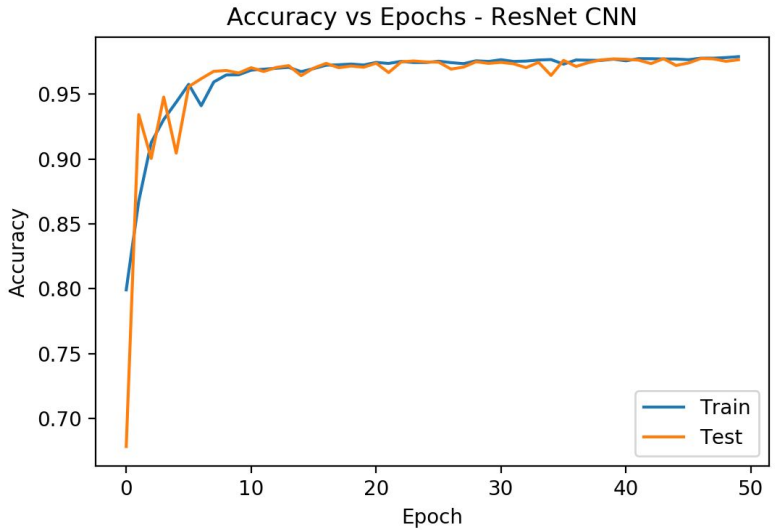
Confusion Matrix - Convolutional Neural Networks

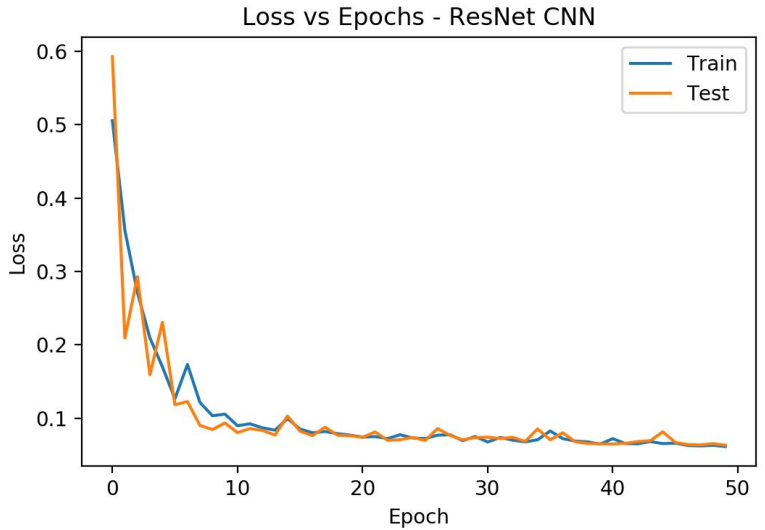




Mini Challenge - Results

Accuracy & Loss vs Epoches - ResNet CNN

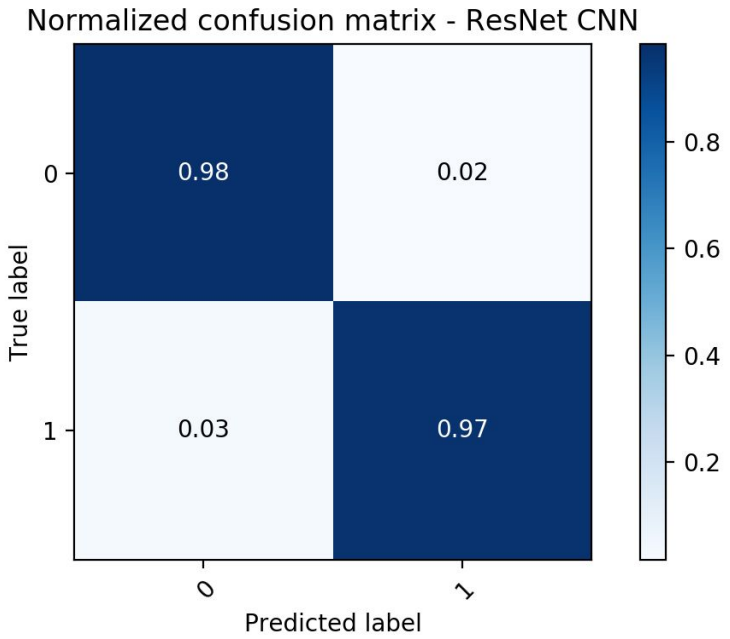


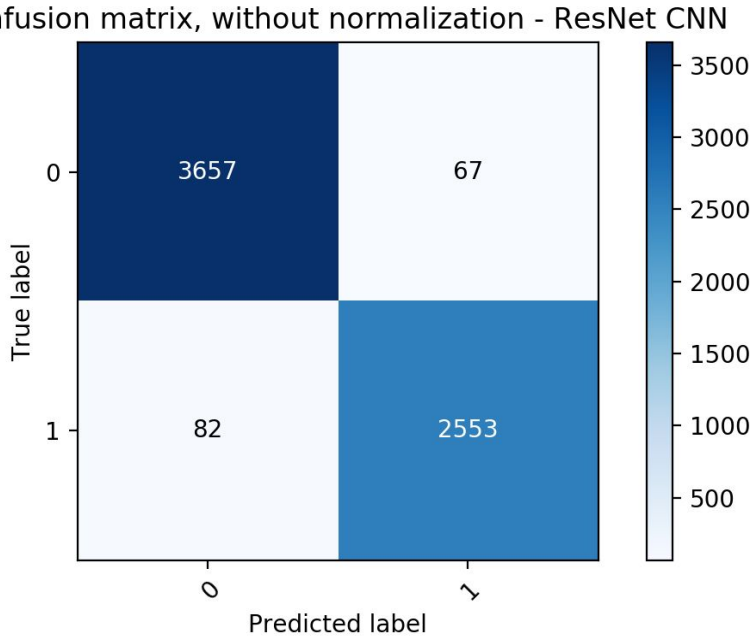


Total training time: 1193.01s (19.88m)
Learning rate: 0.01
Batch size: 64

Mini Challenge - Results

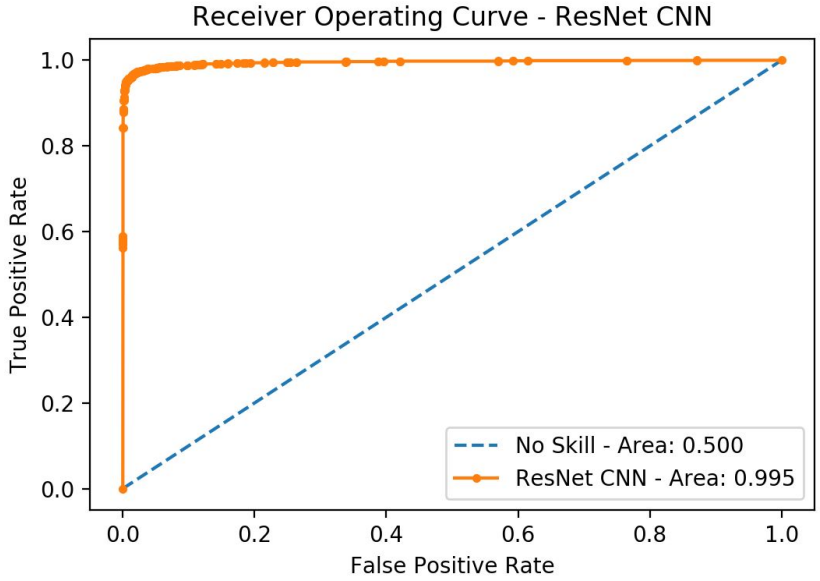
Confusion Matrix - ResNet CNN

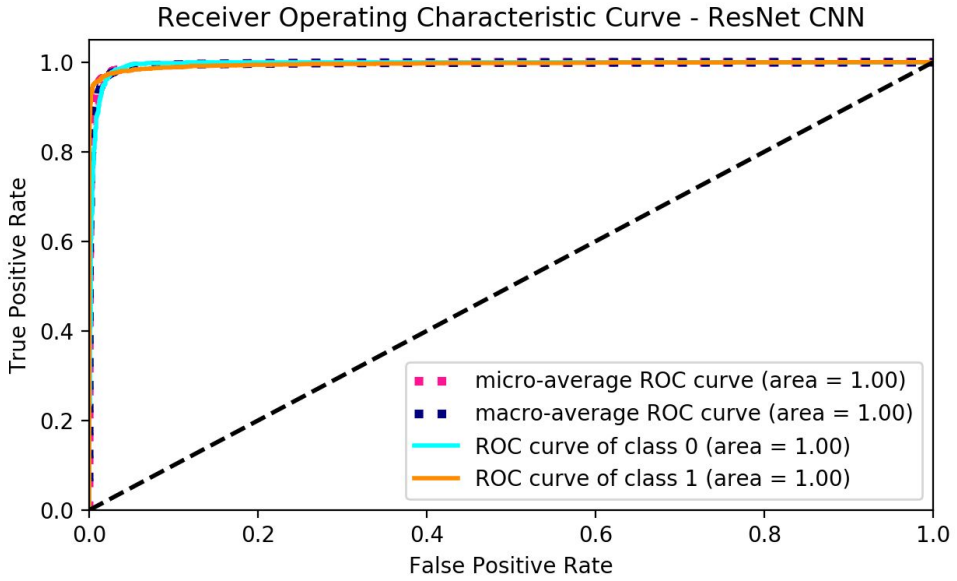




Mini Challenge - Results

Confusion Matrix - ResNet CNN





Mini Challenge - Results

Overall Model Metrics

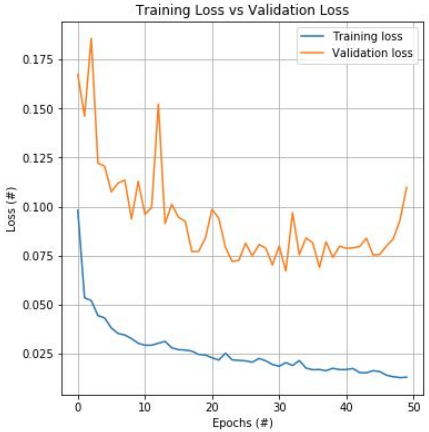
Multi-Layer Perceptron				Basic Convolutional Neural Network				ResNet Convolutional Neural Network			
Class Label	Precision (%)	Recall (%)	F1-Score (%)	Class Label	Precision (%)	Recall (%)	F1-Score (%)	Class Label	Precision (%)	Recall (%)	F1-Score (%)
galaxy	95.61	99.00	97.28	galaxy	97.81	98.51	98.16	galaxy	97.82	98.24	98.03
star	98.54	93.68	96.05	star	97.82	96.80	97.31	star	97.45	96.85	97.15
Micro avg (%)	96.78	96.78	96.78	Micro avg (%)	97.81	97.81	97.81	Micro avg (%)	97.67	97.67	97.67
Macro avg (%)	97.08	96.34	96.66	Macro avg (%)	97.81	97.66	97.73	Macro avg (%)	97.64	97.55	97.59
Weighted avg (%)	96.84	96.78	96.76	Weighted avg (%)	97.81	97.81	97.81	Weighted avg (%)	97.67	97.67	97.67

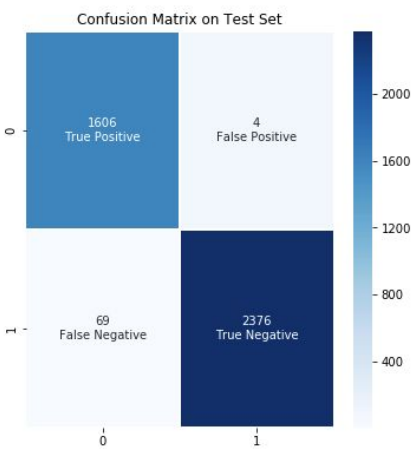
- Overall best performance was achieved by Basic Convolutional Neural Network model.
  - 1) Basic Convolutional Neural Network
  - 2) ResNet Convolutional Neural Network
  - 3) Multi-Layer Perceptron

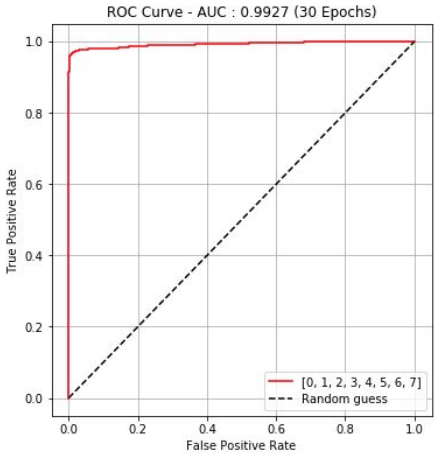
- MLP model achieved impressive recall score for galaxy data

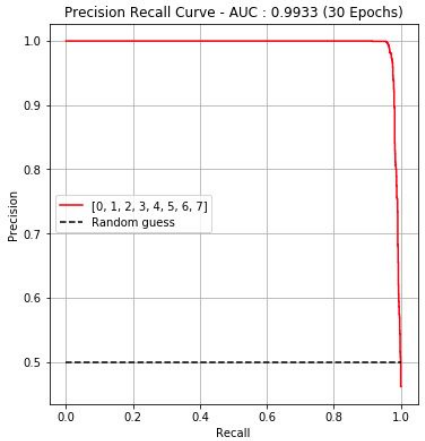
Mini Challenge - Results - Band Configurations

CNN Model - Performance with all bands









Optimal Model Complexity @ 30 epochs

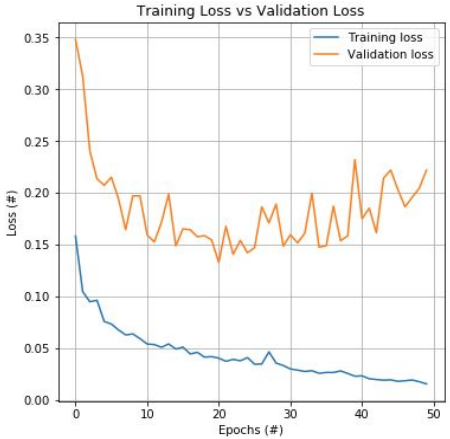
50 epoch train time : 15 mins 29 secs
30 epoch train time : 08 mins 05 secs

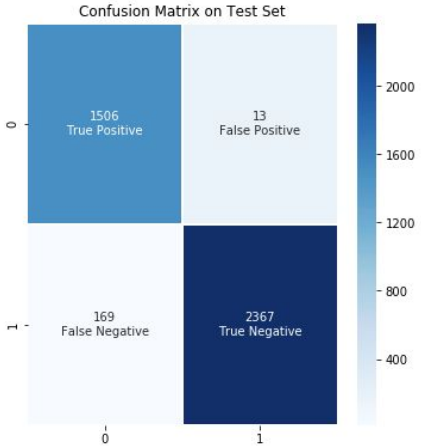
Other Performance Metrics:

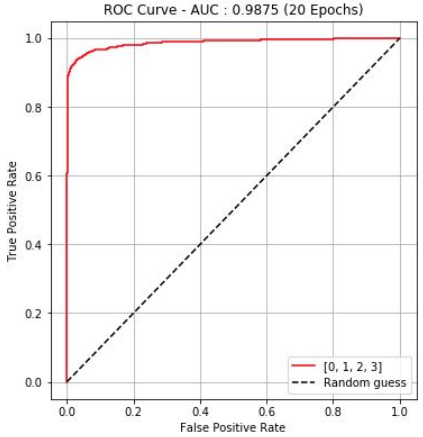
```
('Overall Accuracy : ', 98.19975339087547)
('Star Accuracy : ', 99.83193277310924)
('Galaxy Accuracy : ', 95.88059701492537)
('Precision : ', 95.88059701492537)
('Recall : ', 99.75155279503105)
('F-Measure : ', 97.77777777777777)
```

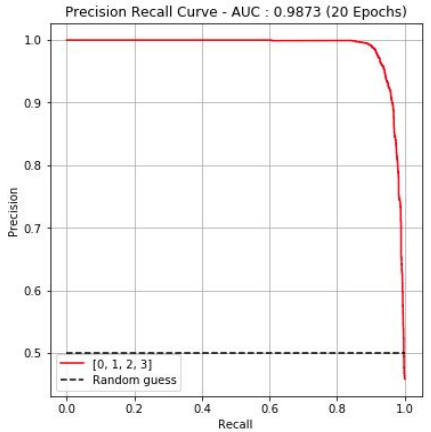
Mini Challenge - Results - Band Configurations

CNN Model - Performance with bands 0 - 3









Optimal Model Complexity @ 20 epochs

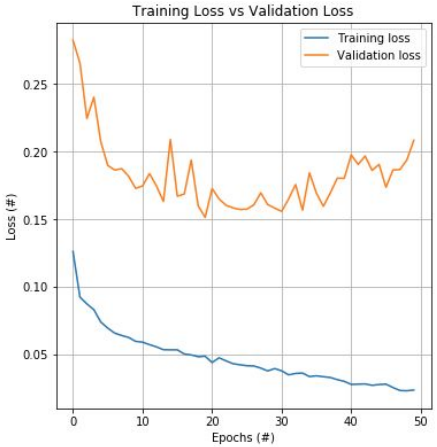
50 epoch train time : 07 mins 55 secs
20 epoch train time : 02 mins 57 secs

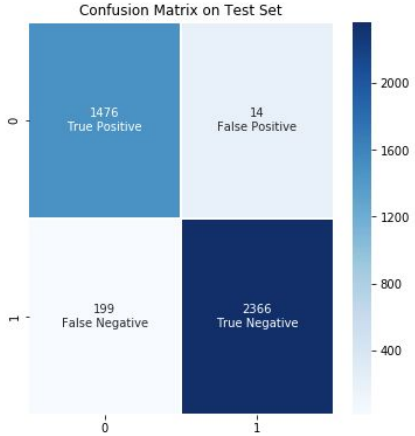
Other Performance Metrics:

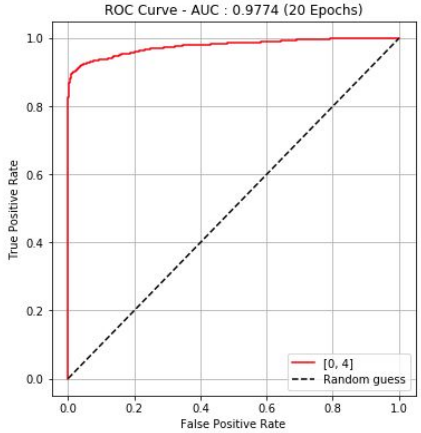
```
('Overall Accuracy : ', 95.51171393341554)
('Star Accuracy : ', 99.45378151260505)
('Galaxy Accuracy : ', 89.91044776119404)
('Precision : ', 89.91044776119404)
('Recall : ', 99.14417379855168)
('F-Measure : ', 94.30181590482154)
```

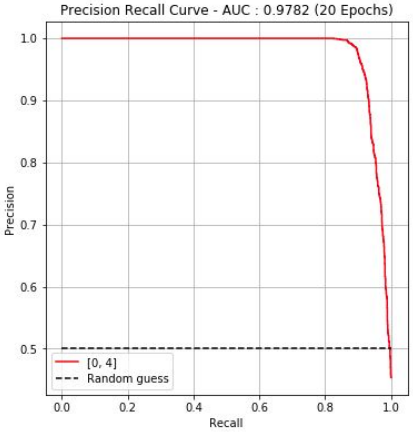
Mini Challenge - Results - Band Configurations

CNN Model - Performance with bands 0, 4









Optimal Model Complexity @ 20 epochs

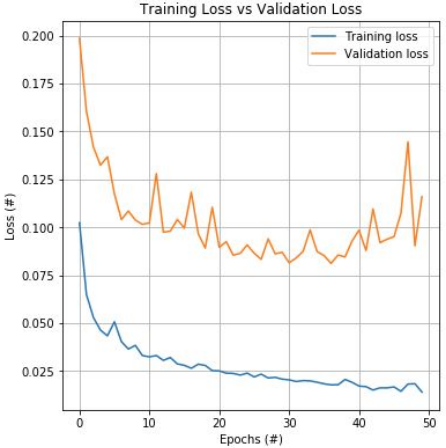
50 epoch train time : 08 mins 41 secs
20 epoch train time : 04 mins 57 secs

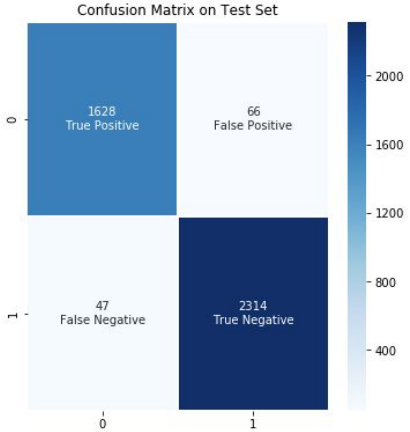
Other Performance Metrics:

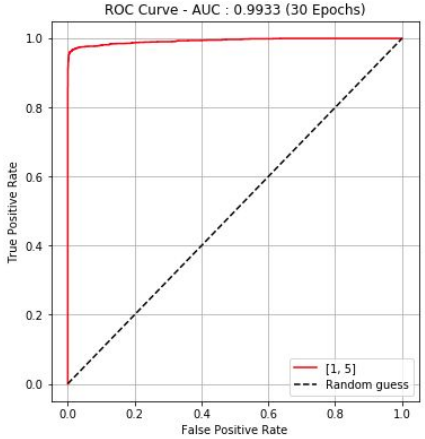
```
('Overall Accuracy : ', 94.74722564734896)
('Star Accuracy : ', 99.41176470588235)
('Galaxy Accuracy : ', 88.11940298507463)
('Precision : ', 88.11940298507463)
('Recall : ', 99.06040268456375)
('F-Measure : ', 93.27014218009478)
```

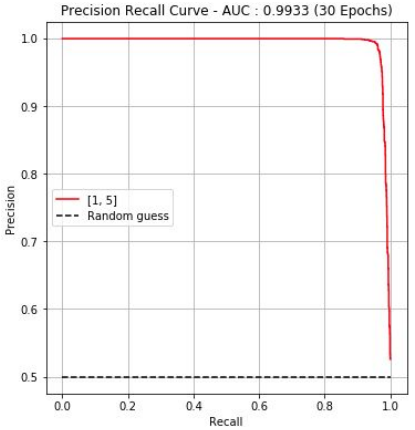
Mini Challenge - Results - Band Configurations

CNN Model - Performance with bands 1, 5









Optimal Model Complexity @ 30 epochs

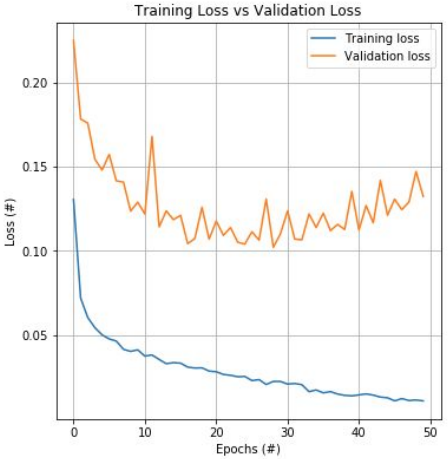
50 epoch train time : 07 mins 18 secs
30 epoch train time : 06 mins 23 secs

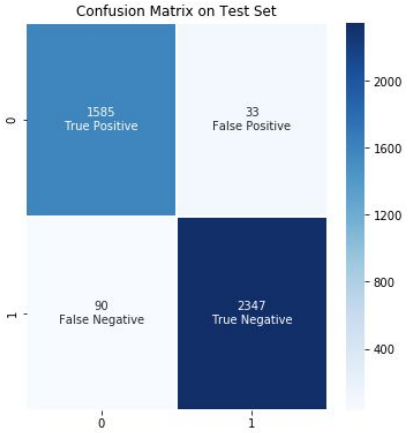
Other Performance Metrics:

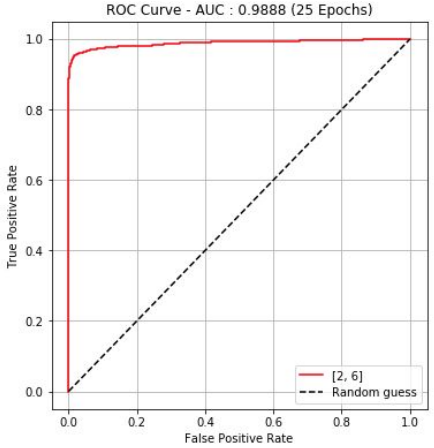
```
('Overall Accuracy : ', 97.21331689272503)
('Star Accuracy : ', 97.22689075630252)
('Galaxy Accuracy : ', 97.19402985074626)
('Precision : ', 97.19402985074626)
('Recall : ', 96.1038961038961)
('F-Measure : ', 96.64588898783022)
```

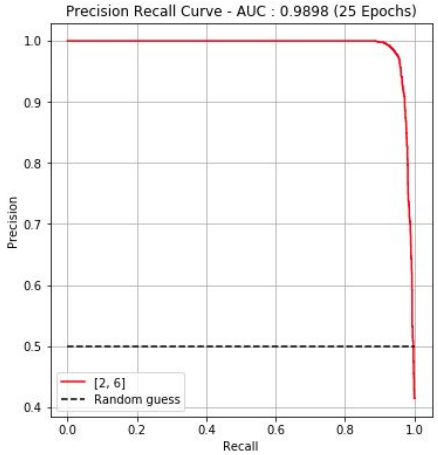
Mini Challenge - Results - Band Configurations

CNN Model - Performance with bands 2, 6









Optimal Model Complexity @ 25 epochs

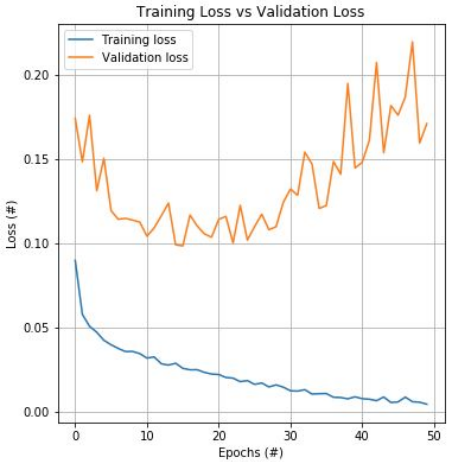
50 epoch train time : 07 mins 36 secs
25 epoch train time : 07 mins 31 secs

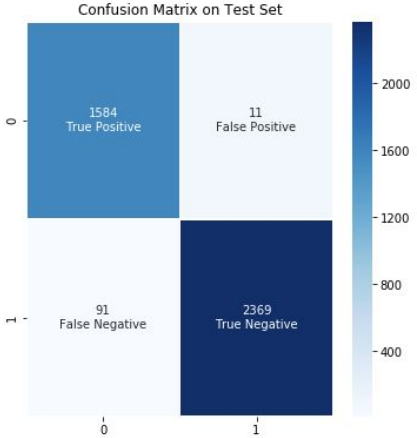
Other Performance Metrics:

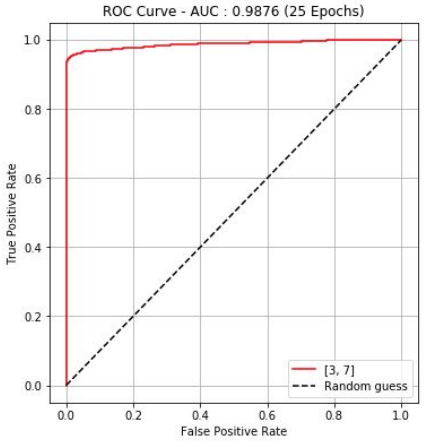
```
('Overall Accuracy : ', 96.96670776818742)
('Star Accuracy : ', 98.61344537815127)
('Galaxy Accuracy : ', 94.6268656716418)
('Precision : ', 94.6268656716418)
('Recall : ', 97.96044499381952)
('F-Measure : ', 96.26480412997266)
```

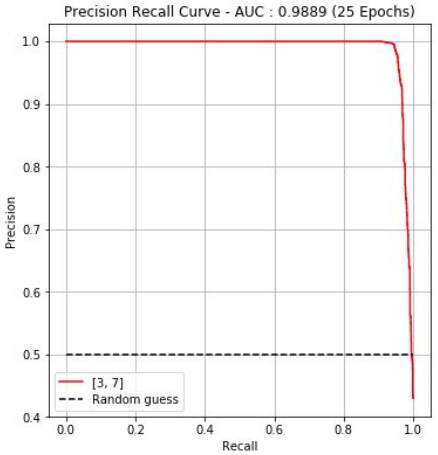
Mini Challenge - Results - Band Configurations

CNN Model - Performance with bands 3, 7









Optimal Model Complexity @ 25 epochs

50 epoch train time : 12 mins 19 secs
25 epoch train time : 07 mins 27 secs

Other Performance Metrics:

```
('Overall Accuracy : ', 97.4845869297164)
('Star Accuracy : ', 99.53781512605042)
('Galaxy Accuracy : ', 94.56716417910448)
('Precision : ', 94.56716417910448)
('Recall : ', 99.3103448275862)
('F-Measure : ', 96.88073394495412)
```

Mini Challenge - Results - Band Configurations

CNN Model - Overall Metrics for Each Configuration

Bands 0-8			
Class Label	Precision (%)	Recall (%)	F1-Score (%)
galaxy	96	100	98
star	100	97	98
Micro avg (%)	98	98	98
Macro avg (%)	98	98	98
Weighted avg (%)	98	98	98

Bands 0-3			
Class Label	Precision (%)	Recall (%)	F1-Score (%)
galaxy	90	99	94
star	99	93	96
Micro avg (%)	96	96	96
Macro avg (%)	95	96	95
Weighted avg (%)	96	96	96

Bands 0, 4			
Class Label	Precision (%)	Recall (%)	F1-Score (%)
galaxy	99	92	96
star	88	99	93
Micro avg (%)	95	95	95
Macro avg (%)	94	96	94
Weighted avg (%)	95	95	95

Bands 1, 5			
Class Label	Precision (%)	Recall (%)	F1-Score (%)
galaxy	97	96	97
star	97	98	98
Micro avg (%)	97	97	97
Macro avg (%)	97	97	97
Weighted avg (%)	97	97	97

Bands 2, 6			
Class Label	Precision (%)	Recall (%)	F1-Score (%)
galaxy	95	98	96
star	99	96	97
Micro avg (%)	97	97	97
Macro avg (%)	97	97	97
Weighted avg (%)	97	97	97

Bands 3, 7			
Class Label	Precision (%)	Recall (%)	F1-Score (%)
galaxy	95	99	97
star	100	96	98
Micro avg (%)	97	97	97
Macro avg (%)	97	98	97
Weighted avg (%)	98	97	97

Outline

Team 4 Introduction

Background

Mini Challenge

Major Challenge

Summary

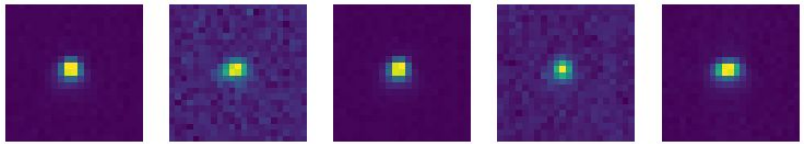
Acknowledgement

Main Challenge - Intro

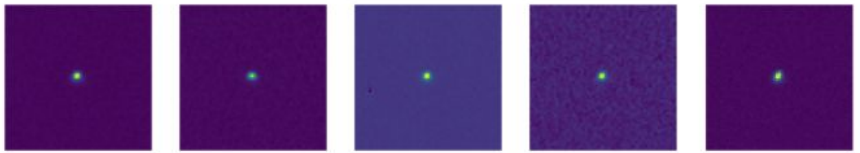
Asteroid Detection

Difference Images (3080x3072 px)

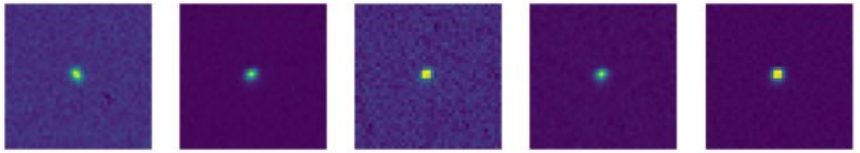
Individual Asteroid Images (26x26)



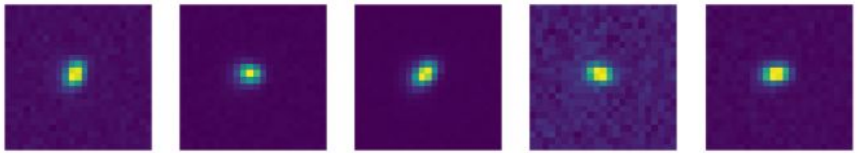
Offset of 20px



Offset of 30px



Offset of 40px



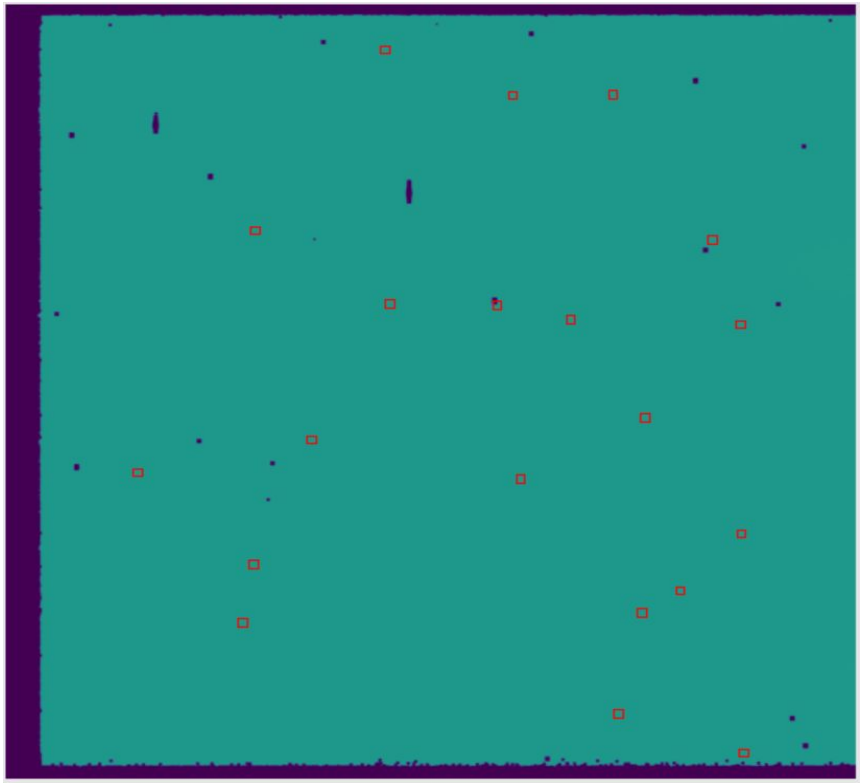
Main Challenge - Intro

Asteroid Detection

	magnitude	length	angle	box
0	16.104747	1	228.358088	galsim.BoundsI(2139,2240,1925,2026)
1	18.222819	3	165.819034	galsim.BoundsI(2131,2234,476,579)
2	15.979216	1	281.243538	galsim.BoundsI(812,913,1147,1248)
3	18.184195	2	95.195193	galsim.BoundsI(2352,2454,1006,1108)
4	16.298675	3	356.967492	galsim.BoundsI(2080,2183,466,569)
5	16.305862	2	56.170222	galsim.BoundsI(1654,1756,2119,2221)
6	16.967739	1	211.522894	galsim.BoundsI(1543,1644,1406,1507)
7	18.127119	1	270.145747	galsim.BoundsI(2707,2808,1119,1220)
8	15.172116	2	153.090408	galsim.BoundsI(2140,2242,2428,2530)
9	17.504909	3	295.527269	galsim.BoundsI(1139,1242,2915,3018)
10	15.029287	3	332.281818	galsim.BoundsI(871,974,110,213)
11	17.396485	3	96.447372	galsim.BoundsI(224,327,1743,1846)
12	18.680659	1	79.306272	galsim.BoundsI(1078,1179,2589,2690)
13	17.984784	3	272.557400	galsim.BoundsI(1659,1762,103,206)
14	15.566666	1	305.983155	galsim.BoundsI(1300,1401,2083,2184)
15	18.794665	2	180.476585	galsim.BoundsI(523,625,1734,1836)
16	15.269838	1	67.397933	galsim.BoundsI(1847,1948,1343,1444)
17	16.462608	3	123.557740	galsim.BoundsI(2617,2720,2903,3006)
18	17.662699	1	128.526359	galsim.BoundsI(2613,2714,500,601)
19	15.841308	3	137.533233	galsim.BoundsI(449,552,2121,2224)]

- Every big (approx. 3000x3000) difference images has a Pandas DataFrame
- Streaks are objects moving in a straight line (asteroids in this case).
- Magnitude = Magnitude of the streak
- Length = Length of the streak
- Angle = Angle of the streak
- Box coordinates of the injected asteroids
  - width = ‘length’ + 100
  - height = ‘length’ + 100

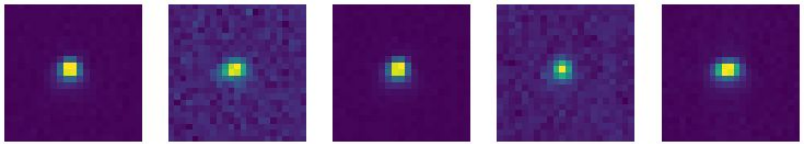
Main Challenge - Intro



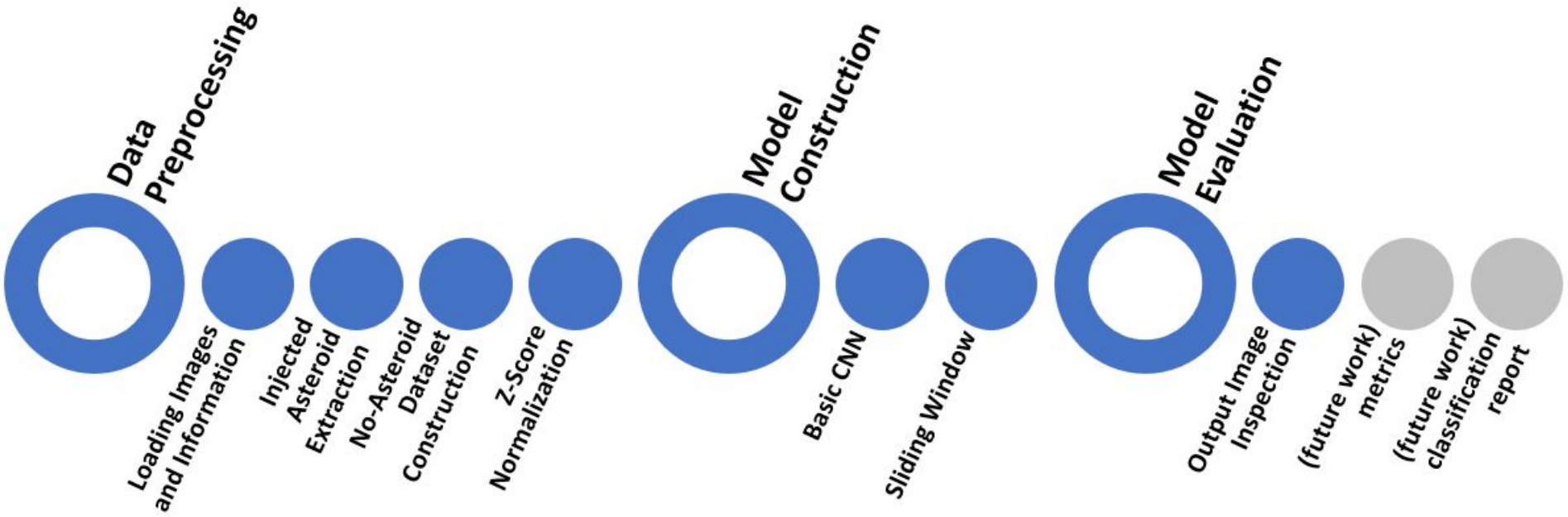
Aim

- Detect the presence of asteroids in the big (3080x3072 px) difference images.
- 26x26 boxes around injected asteroids (given targets)
- Asteroid images (around red boxes) were extracted to get an asteroid class data set.

Individual Asteroid Images (26x26)

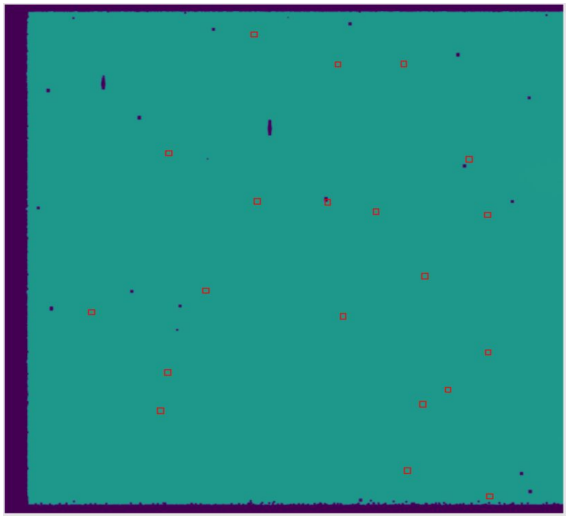


Main Challenge - Methodology



Main Challenge - Methodology

Data Preprocessing



- Red boxes are the injected asteroids

No-Asteroid Data Set

- Difference images was randomly sampled with a 50/50 split between asteroid and no-asteroid sample numbers.
- Samples were checked to ensure they were not within injected asteroid boxes.
- (26x26)

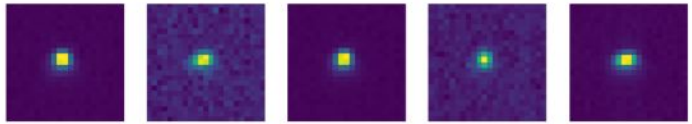
Z-Score Normalization

- Normalized to improve model metrics.

Asteroid Data Set

- Collected using `asteroid_injected_information_#.npz` files and difference images.
- (26 x 26)

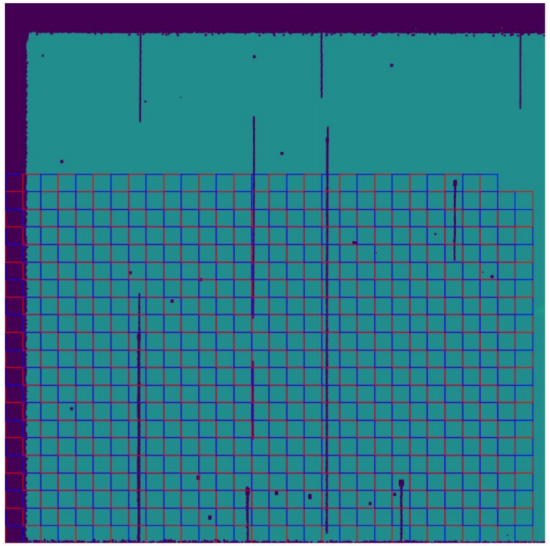
Individual Asteroid Images (26x26)



Main Challenge - Methodology

Model Construction

Sliding Window Model



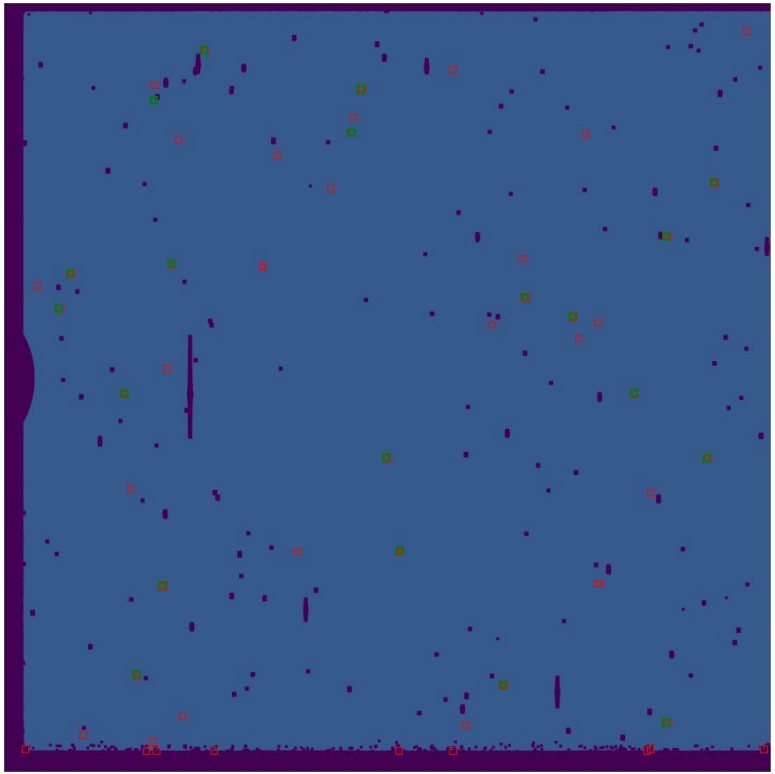
- This was exaggerated (200x200 images)
  - In reality it is (26x26 images)

Basic Convolutional Neural Network

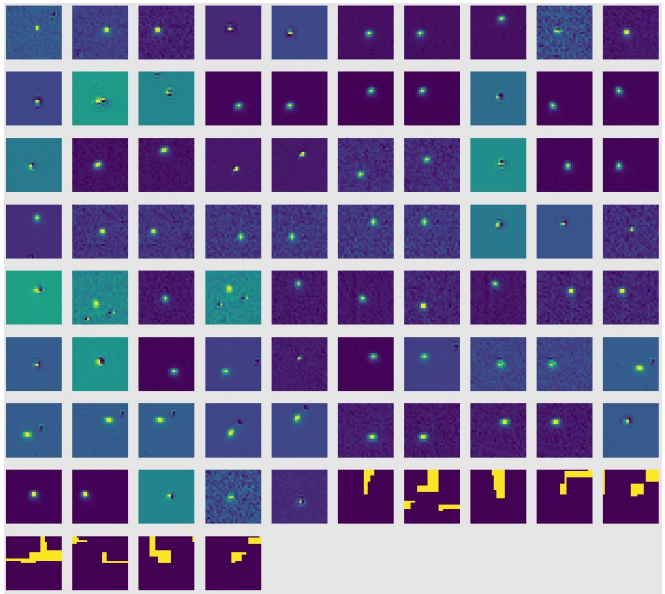
```
import torch.nn.functional as F

class basic_CNN(torch.nn.Module):
    def __init__(self):
        super(basic_CNN, self).__init__()
        self.conv1 = torch.nn.Conv2d(in_channels=n_bands, out_channels=6, kernel_size=
        self.pool = torch.nn.MaxPool2d(kernel_size=4, stride=2)
        self.conv2 = torch.nn.Conv2d(in_channels=6, out_channels=3, kernel_size=4)
        self.fc1 = torch.nn.Linear(in_features=12, out_features=120)
        self.fc2 = torch.nn.Linear(120, 500)
        self.fc3 = torch.nn.Linear(1000, 500)
        self.fc4 = torch.nn.Linear(500, 2)
        self.dropout = torch.nn.Dropout(0.2)

    def forward(self, x):
        x = self.conv1(x)
        x = F.relu(x)
        x = self.pool(x)
        x = self.conv2(x)
        x = F.relu(x)
        x = self.pool(x)
        x = torch.flatten(x, 1)
        x = self.fc1(x)
        x = F.relu(x)
        x = self.dropout(x)
        x = self.fc2(x)
        x = F.relu(x)
        x = self.dropout(x)
        x = self.fc3(x)
        x = F.relu(x)
        x = self.dropout(x)
        x = self.fc4(x)
        return x
```

```
def forward(self, x):
    x = self.conv1(x)
    x = F.relu(x)
    x = self.pool(x)
    x = self.conv2(x)
    x = F.relu(x)
    x = self.pool(x)
    x = torch.flatten(x, 1)
    x = self.fc1(x)
    x = F.relu(x)
    x = self.dropout(x)
    x = self.fc2(x)
    x = F.relu(x)
    x = self.dropout(x)
    x = self.fc3(x)
    x = F.relu(x)
    x = self.dropout(x)
    x = self.fc4(x)
    return x
```

Main Challenge - Results



Matrix of images believed to be asteroids

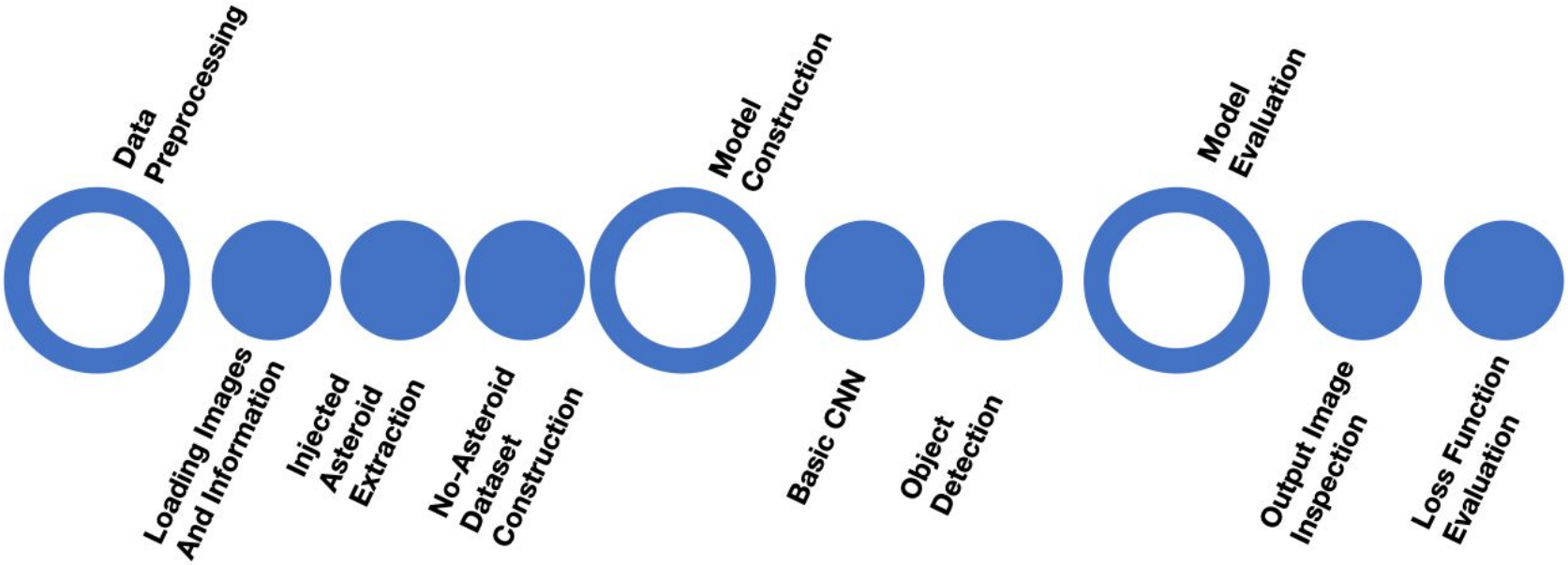


Confidence = 1.0
Threshold > 0.99
Sliding_shift = 7 px

- Sliding window yielded results
- Notice the last predictions were inaccurate
  - Future work needed
- We do not have model metrics yet

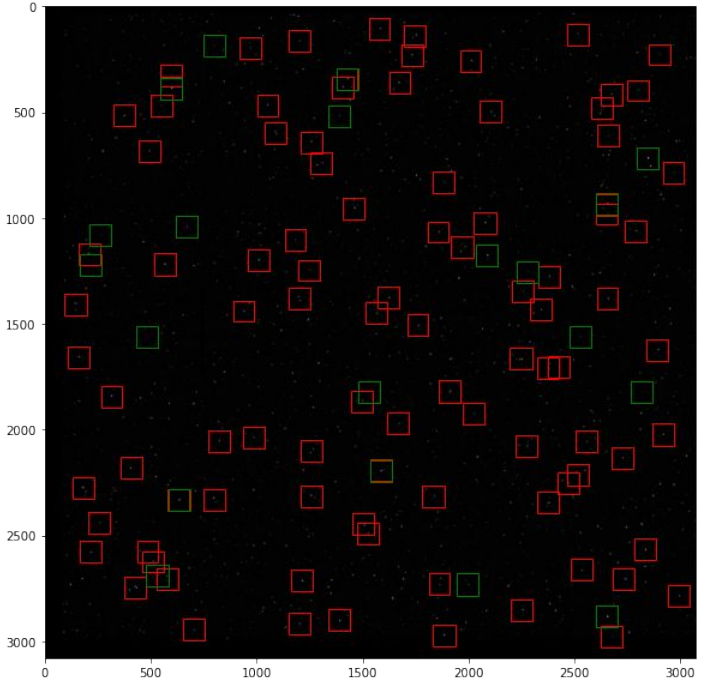
Main Challenge - Methodology

Alternative using Object Detection



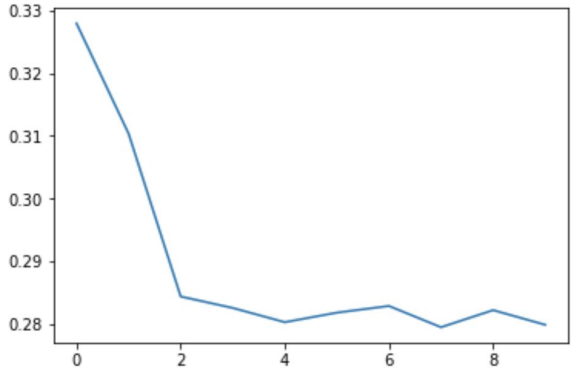
Main Challenge - Methodology

Alternative using Object Detection

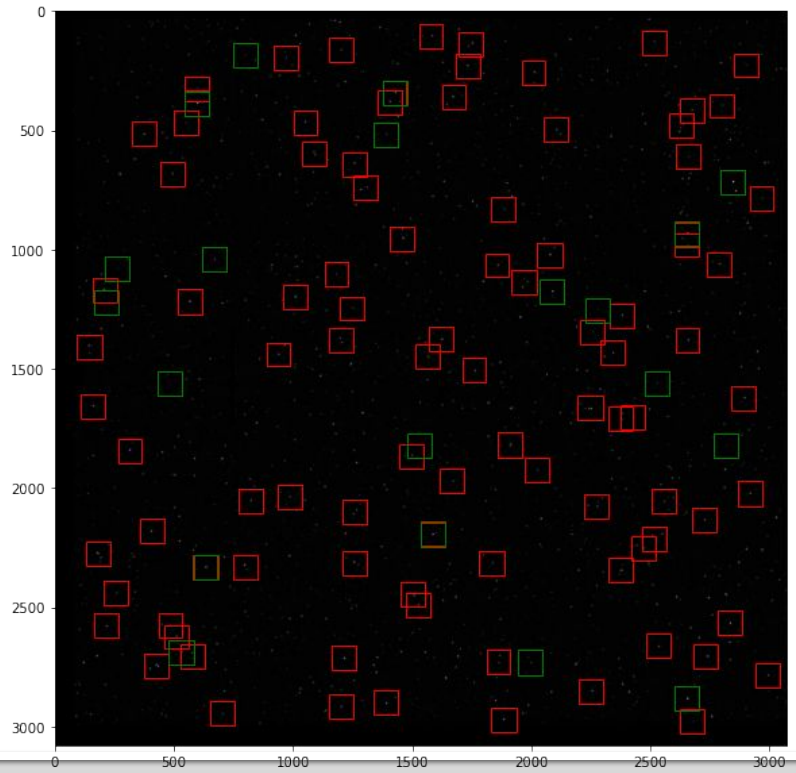


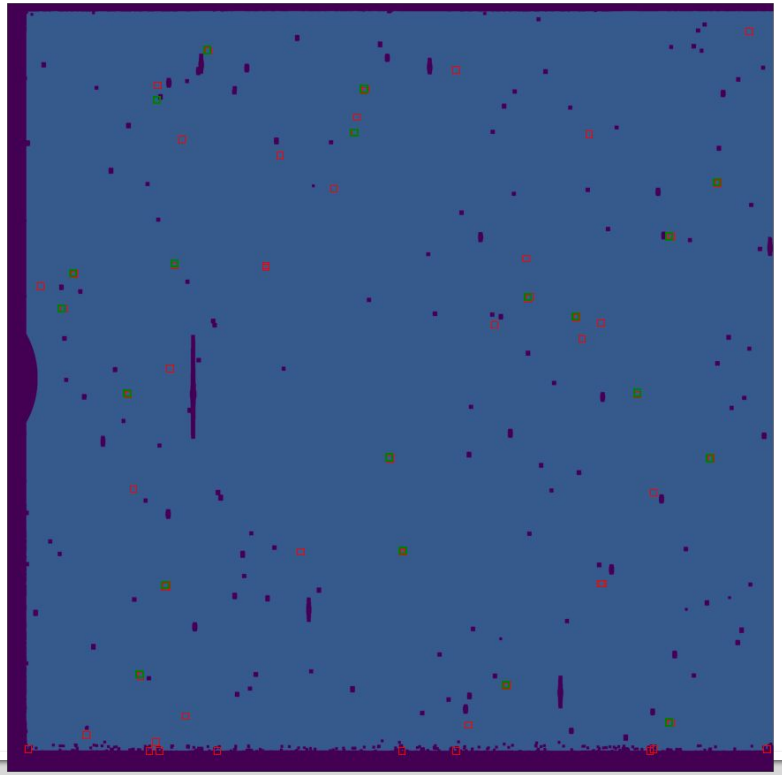
Set up

- 70/30 Train/Val Data Split
- Extract Asteroid locations on the given images
- Train Object detection Algorithm on data



Main Challenge - Methodology





Summary

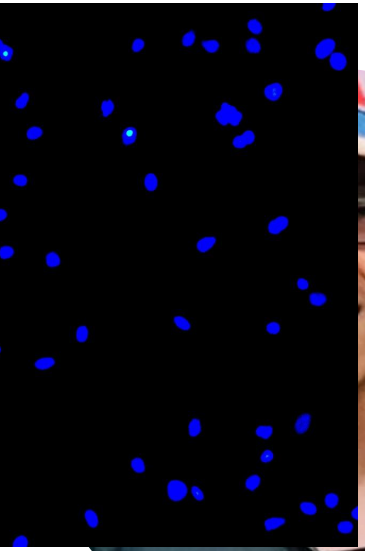
Mini Challenge:

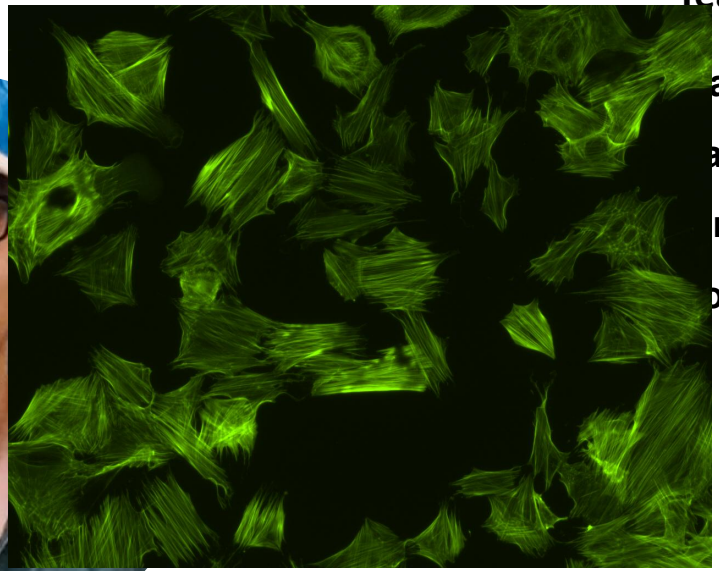
- We implemented 3 models (MLP, CNN, ResNET) to perform classification.
- All the three models yielded similar results.
- Trained CNN model on subset of bands and found the metrics to be fairly similar.

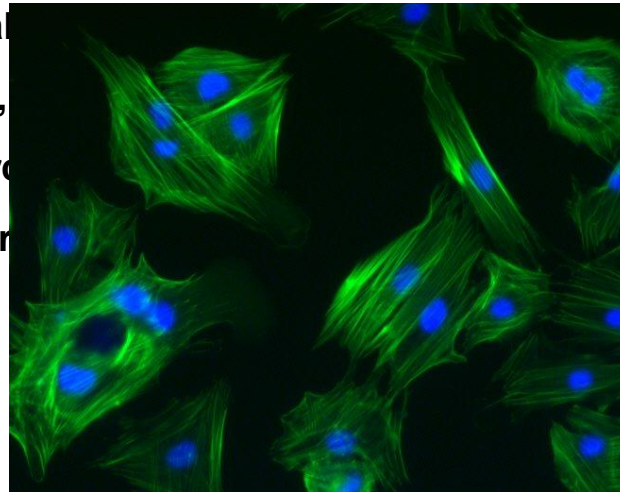
Major Challenge:

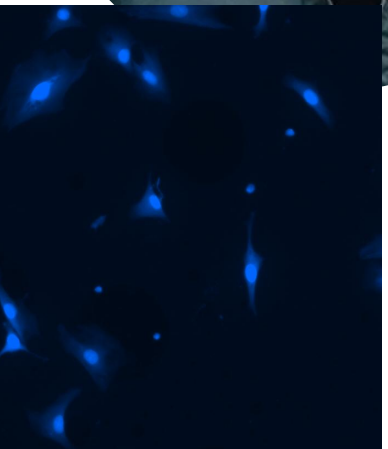
- Devised two ways of locating asteroids with two CNN models
  - Sliding Window Method
  - Region Proposal Network
- Compared visual results side by side
- With more time, will be able to predict true asteroids among the matrix of observed asteroids

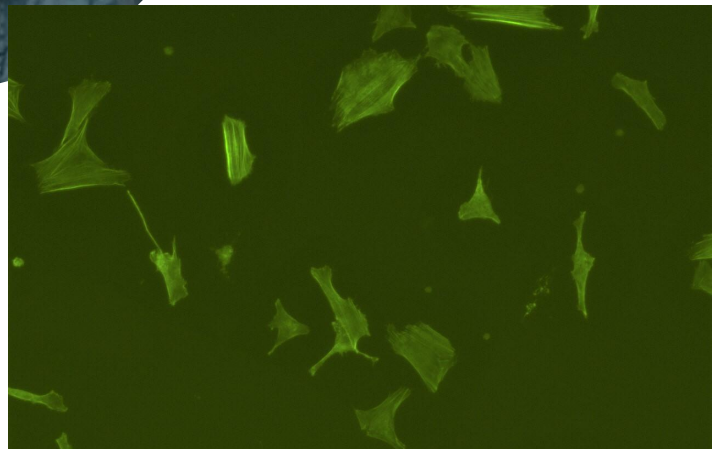
Team 4: Dongwei Sun

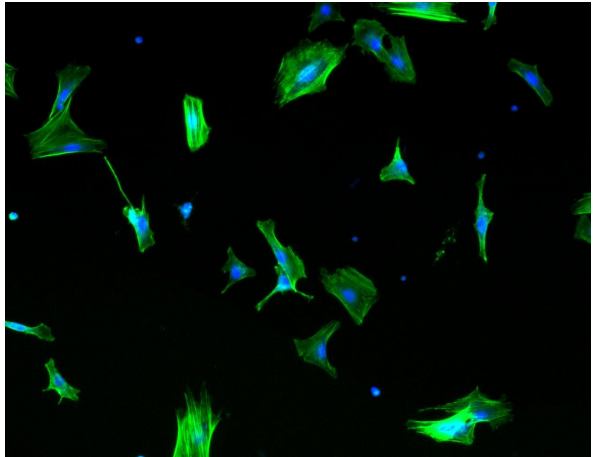












Team 4: Ivan Neto

Degree: Computer Science Undergraduate

Skills: Python, Keras, Scikit-learn

Things learned: PyTorch, Anaconda, Basic Astronomy, Working with Images

With more time: PyTorch in depth, optimizing my models better, regularization in depth

Team 4: Ponmanikandan Velmurugan (Mani)

Degree: Masters' in Computer Science

Skills: Python, Numpy, Pandas, Sklearn

Things learned: Pytorch, Neural Networks, GitHub, Big datasets, Image processing

With more time: K-fold cross validation, Different models, Modelling with GPU

Team 4: Kaichun Chen(Kevin)

Degree:Physics Undergraduate

Skills:Python

Things learned:PyTorch, Working with Images

With more time:Get familiar with Pytorch, using and optimizing models.

Team 4: Julin

Degree: Masters of Science in Computer Science

Skills: Python, Detecto, Pytorch

Things learned: Detecto, cleaning data

With more time: Get the model more accurate

Acknowledgements





Professor Vagelis Papalexakis
Jen Bellig
Brian Gallagher
Ryan Dana
Braden Soper
LC Support Team
All the lecturer speakers
Other teams