In-Orbit Object Detection with Computer Vision [1] [2] [3]

Ryan T. White

Florida Institute of Technology

Department of Mathematical Sciences Seminar

[1] Based on joint work with Dr. Markus Wilde, Trupti Mahendrakar, Nathan Fischer, and Andrew Ekblad

[2] Acknowledgement for contributions from Rebecca Smith and Nouraldean El-Chariti

[3] Special thanks to the ORION Lab for permitting use of its facilities

Problem: Dysfunctional or Obsolete Satellites

When the multi-million-dollar satellites... malfunction or run out of fuel... they are either flown towards the atmosphere to burn up, or propelled to a region... known as the graveyard orbit [so it] won't pose a threat to other spacecraft.

The problem with this system is that the technology onboard the satellite may be working, but it all gets tossed away with the spacecraft simply because repairs or refueling are not possible.

"There is no other area of human activity where we build something that's worth a half-billion dollars or a billion dollars, and never look at it again, never fix it, and never upgrade it," –Gordon Roesler (DARPA)[4]

[4] Janice Cantieri. Fixing satellites in space. Astronomy.com. https://astronomy.com/news/2017/12/fixing-satellites-in-space

Problem: Space Junk

Hundreds of millions of pieces of space junk orbit the Earth daily, from chips of old rocket paint, to shards of solar panels, and entire dead satellites. This cloud of high-tech detritus whirls around the planet at about 17,500 miles per hour. At these speeds, even trash as small as a pebble can torpedo a passing spacecraft.[5]

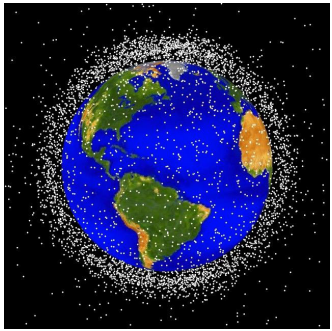


Figure: Space junk tracked by NASA

Figure: Artist's rendering of space junk

[5] J. Chu. Space Junk: The Cluttered Frontier. *MIT News*. https://news.mit.edu/2017/space-junk-shards-teflon-0619

Long-Term Goals

Identify in-orbit objects in real-time with local hardware, devise a guidance and navigation system allowing autonomous "chaser satellites" to perform in-orbit satellite repairs, capture space debris, etc.

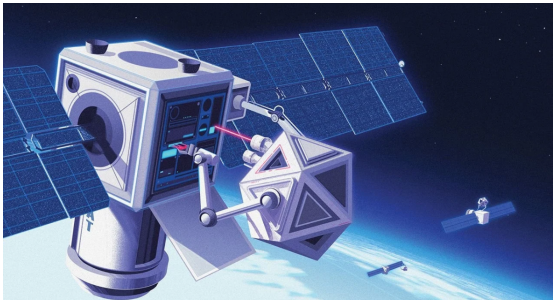


Figure: Artist's rendering[6] of a chaser satellite performing an in-orbit repair

[6] Image by S. Chivers (2018, Nov 24), appears in "Coming, ready or not... It will soon be possible to send a satellite to repair another... Or to destroy it." *The Economist*.

Short-Term Goals

- **Gather and annotate** satellite images for training data
- Train, tune, and test **image classifiers**
- Train, tune, and test **object detectors** on images
- **Test on video feeds** captured under realistic conditions
- **Improve the object detector** with an algorithm selectively applying an internal regression model
- Implement and test methods with a **small computational budget**

Phase 1: Satellite Component **Classification**

- Gather and label open-source image data (100 from each class)[7]

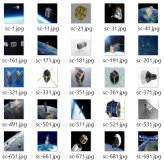
(a) Spacecraft

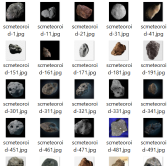
(b) Meteoroid

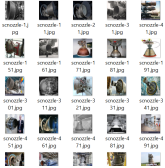
(c) Nozzle

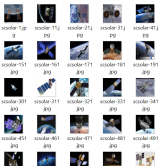
(d) Fuel Tank

(e) Solar Panel









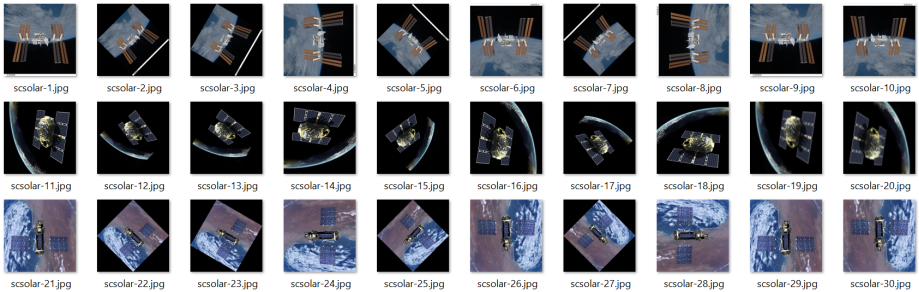
- Train, tune, and test convolutional neural networks to classify satellite components

[7] 99% of this was done by Trupti Mahendrakar

Data Preparation

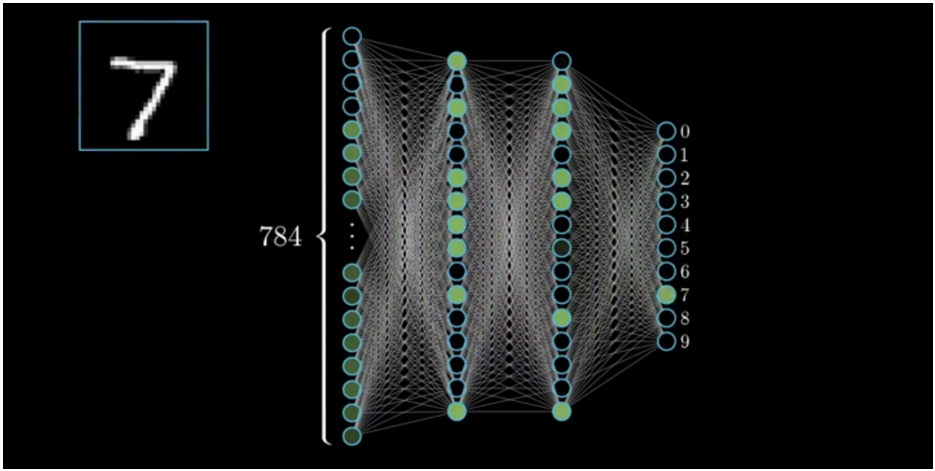
Data gathering and **preprocessing** followed these steps:

1. Shape images to a common $512 \times 512 \times 3$ ($786,432$ dimensions) without distorting aspect ratios
2. **Augment** the data with rotations[8]



3. Randomize the data and split into random batches of size 32

[8] done by Trupti Mahendrakar



9

[9] G. Sanderson. But what is a neural network? 3Brown1Blue. https://www.youtube.com/watch?v=aircAruvnKk

For the input image $x \in [0,1]^d$, the **activations** the layers are computed as

$$a^0 = x \qquad \text{(the input)}$$
$$a^l = f(W^l a^{l-1}) \text{ for } l = 1, ..., L-1$$
$$\mathcal{N}(x) = a^L = S(W^L a^{L-1}) \qquad \text{(the output)}$$

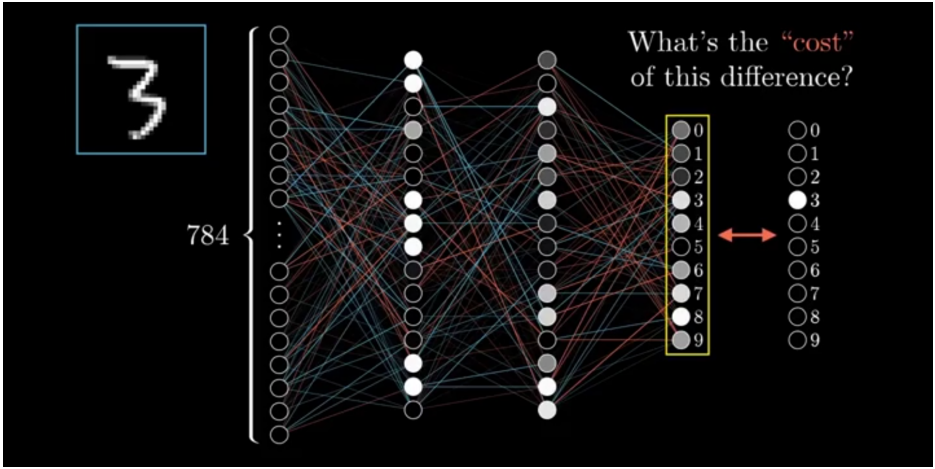
where

$$S(y)_j = \frac{e^{y_j}}{\sum_{i=1}^{k} e^{y_i}}$$

is the **softmax** function ensuring the final activations $a^L \in [0,1]^k$ correspond to a probability distribution, and

$$f(x) = \max(0, x)$$

is the **ReLU activation function** (applied elementwise).



10

[10] G. Sanderson. But what is a neural network? 3Brown1Blue. https://www.youtube.com/watch?v=aircAruvnKk

An Optimization Problem

- Given a training dataset of image-label pairs $(x_1, y_1), ..., (x_n, y_n)$ with $x_i \in [0,1]^d$ and $y_i \in \{\mathbf{e}_1, ..., \mathbf{e}_k\}$, the neural network attempts to solve the optimization problem

$$\min_{W^1,...,W^L} L(X, Y, W^1, ..., W^L)$$

$$\min_{W^1,...,W^L} \sum_{i=1}^{n} \sum_{j=1}^{k} [y_{ij} \ln \mathcal{N}(x_i)_j + (1 - y_{ij}) \ln(1 - \mathcal{N}(x_i)_j)] + \lambda \sum_{k=1}^{L} \|W^k\|_F^2,$$

  i.e. we hope to choose the weights that minimize the **cross-entropy loss function**.

- This is a difficult **non-convex problem** in extremely high dimensional space (perhaps in $\mathbb{R}^N$ for $N > 100$m).

- In practice, we must settle for **local minima** where the net performs well in experiments.

- **Gradient descent** simply updates weights as

$$\theta_{t+1} = \theta_t - \alpha \nabla L_t,$$

- **Backpropagation**[11] computes gradients efficiently in neural networks
- A faster variant, **mini-batch gradient descent**[12], makes steps based on random samples of inputs
- **Adam**[13] (adaptive moment estimation) uses decaying averages of past gradients and squared gradients

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla L_t \qquad v_t = \beta_2 v_{t-1} + (1 - \beta_2) \nabla L_t^2$$
$$\hat{m}_t = \frac{m_t}{1 - \beta_1} \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2}$$

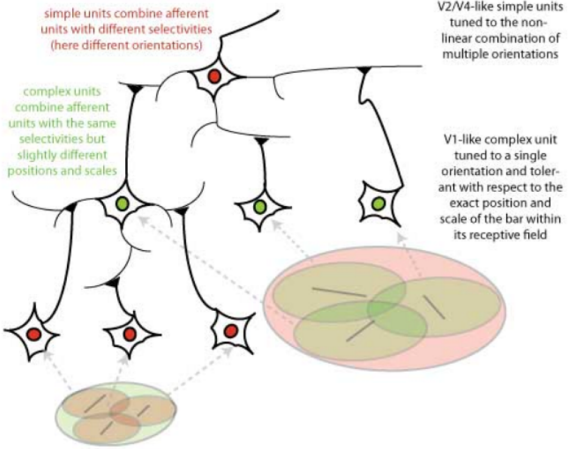
and makes steps as

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}$$

[11] D. Rumelhart, G. Hinton, R. Williams (1986). Learning representations by back-propagating errors. *Nature*. **323** (6088): 533–536.
[12] H. Robbins, S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, **22** (3):400–407, 1951
[13] D. Kingma, J. Ba (2015). Adam: a Method for Stochastic Optimization. *International Conference on Learning Representations*, 1–13.

Convolutional Neural Networks (CNNs)

Inspired by **studies of the visual cortex** by Nobel laureate neurophysiologists Hubel and Wiesel in the 60s[14]



[14] T. Serre and T. Poggio (2010). A neuromorphic approach to computer vision. *Communications of the ACM* 53, 10

The Neocognitron

Fukushima[15] (1980) created the **neocognitron** with a layered structure of neurons

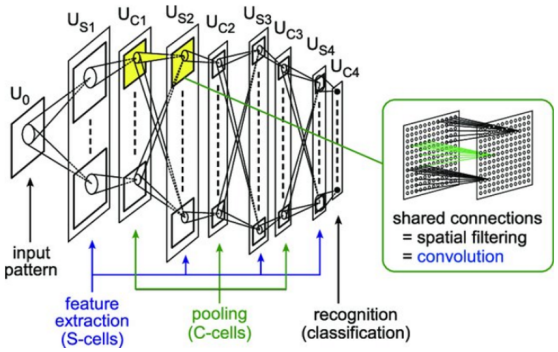


Figure: Fukushima's neocognitron

- S-cells extract local features – **convolutional** layers of units whose receptive fields cover a patch of the prior layer
- C-cells tolerate deformations of features – **pooling** layers reduce dimensionality by merging subsets of the prior layer
- Local features in the input are integrated gradually and classified in the higher layers

[15] K. Fukushima (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position", Biological Cybernetics, **36**[4], pp. 193-202

Image Kernels (a Prelude to Convolutional Layers)

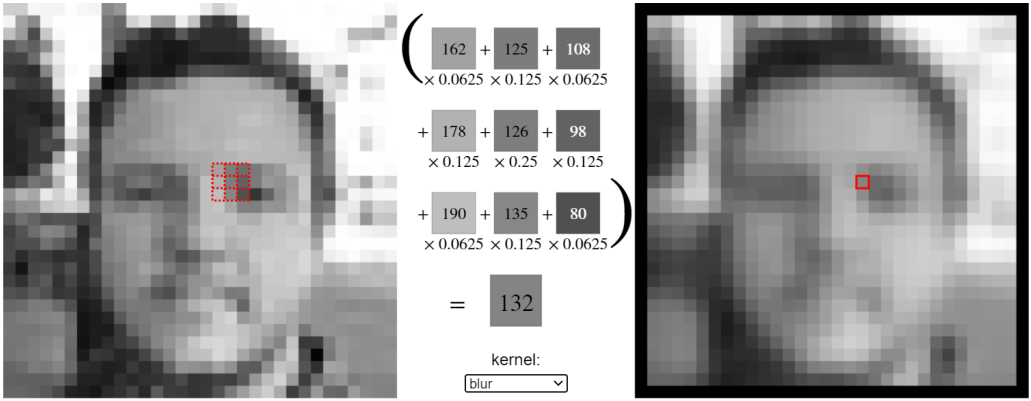
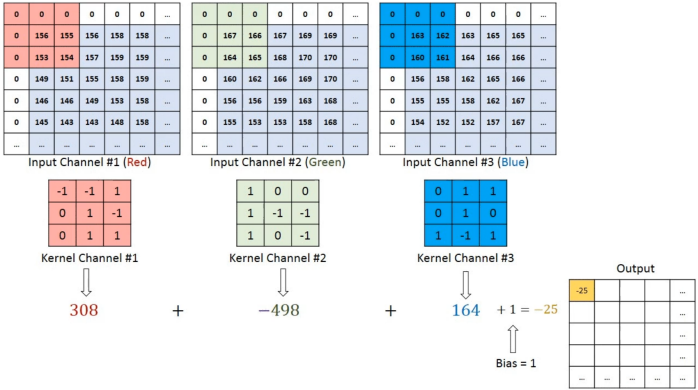


Figure: Image kernels[16] are commonly used in image processing to blur, sharpen, detect edges, etc.

[16] V. Powell (2018). Image Kernels Explained Visually. https://setosa.io/ev/image-kernels/

Convolutional Layers

- Convolutional layers "scan" overlapping patches of the input volume, element-wise multiply by a learnable tensor, and add the elements

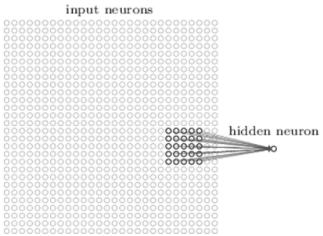


17

[17] B. Boehmke (2020). Computer vision and CNNs: MNIST revisited. https://rstudio-conf-2020.github.io/dl-keras-tf/04-computer-vision-cnns.html

Convolutional Layers - Computation

- Let a_{jk} be an activation of the neuron in the jth row and kth column of a convolutional layer
- Let W be an 5×5 image kernel (A.K.A. filter or feaure map)



[18]

- The activation of a connected neuron in the **next layer** will be

$$f\left(b + \sum_{l=0}^{4}\sum_{m=0}^{4} w_{lm} a_{j+l,k+m}\right)$$

[18] M. Nielsen (2015). Neural Networks and Deep Learning. Determination Press. Ch 6.

Convolutional Layers - Sparsity

- Convolutional layers introduce **sparsity**: each output neuron is only connected to a small set of neurons in the previous layer.

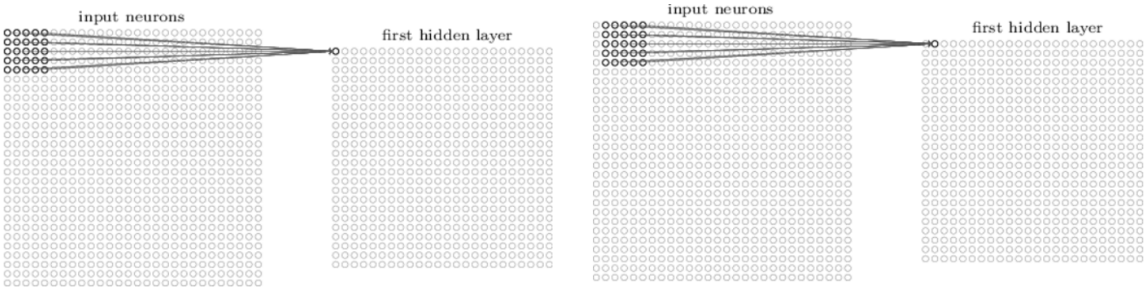
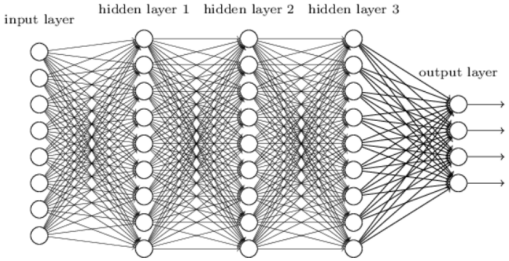


Figure: Connections between layers[19]

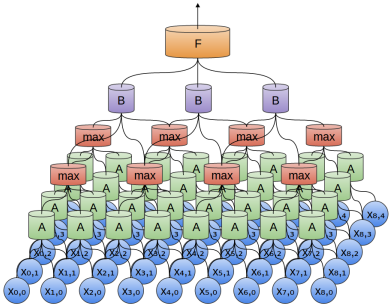
[19] M. Nielsen (2015). Neural Networks and Deep Learning. Determination Press. Ch 6.

Convolutional Layers - Sparsity

- Convolutional layers introduce **sparsity**: each output neuron is only connected to a small set of neurons in the previous layer.



(a) Fully connected NN have dense connections[20]



(b) CNNs have sparse connections[21]

[20] M. Nielsen (2015). Neural Networks and Deep Learning. Determination Press. Ch 1.

[21] C. Olah (2014). Conv Nets: A Modular Perspective. https://colah.github.io/posts/2014-07-Conv-Nets-Modular/

Pooling (Downsampling)

- Pooling layers synthesize information from small patches of neurons and reduce dimensionality

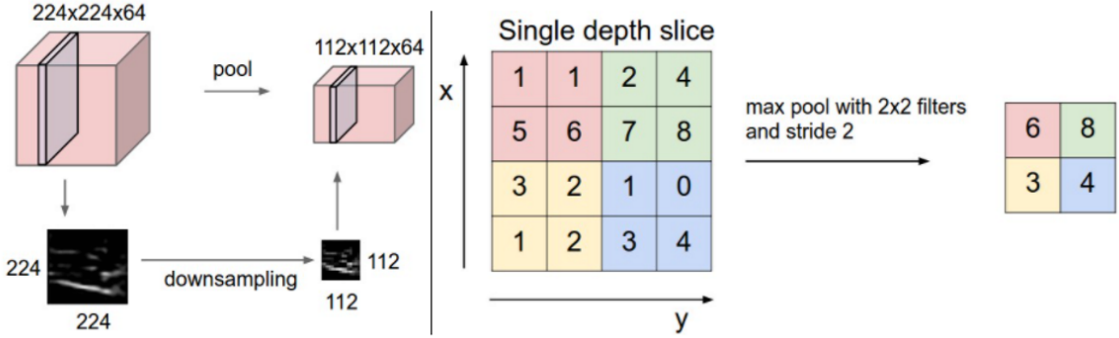
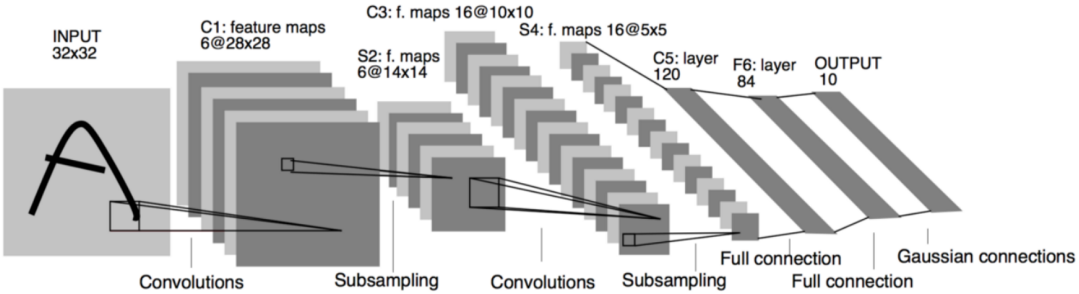


Figure: A max-pooling layer[22]

[22] A. Karpathy et. al. (2012). Stanford CS231n notes on CNNs. https://cs231n.github.io/convolutional-networks/

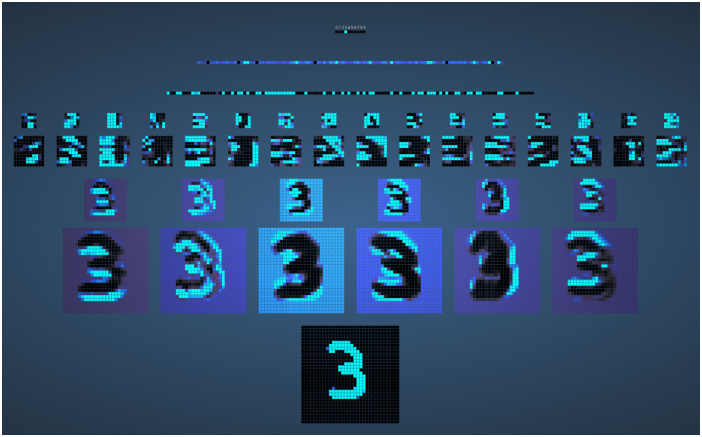
- LeCun et. al. (1998) created LeNet[23], the first truly effective CNN for computer vision



- Used by the USPS to read Zip codes handwritten on mail

[23] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324

LeNet Visualization



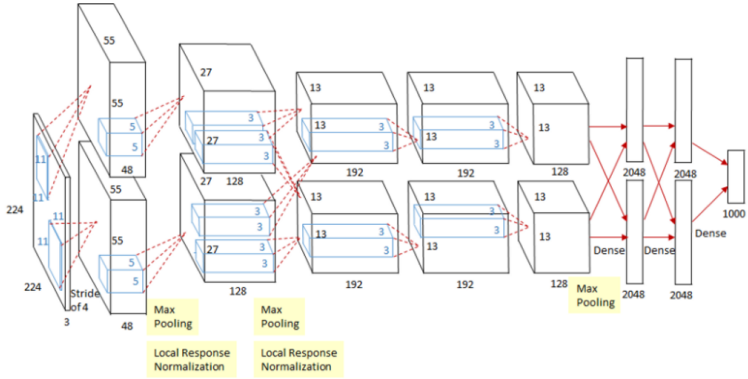
24

[24] A. Harley (2015). An Interactive Node-Link Visualization of Convolutional Neural Networks. In: Bebis G. et al. (eds) *Advances in Visual Computing. ISVC 2015. Lecture Notes in Computer Science*, vol 9474. `http://www.cs.cmu.edu/~aharley/vis/conv/flat.html`

Computer Vision after LeNet: ImageNet

Modern CNNs – AlexNet

- Krizhevsky, Sutskever, Hinton (2012) develop AlexNet[25], the first effective **GPU-based** CNN



[25] A. Krizhevsky, I. Sutskever, G. Hinton (2012). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*. **60** (6): 84–90.

Modern CNNs – AlexNet's Impact

- The paper is **a bit** influential...

TITLE	CITED BY	YEAR
Imagenet classification with deep convolutional neural networks A Krizhevsky, I Sutskever, GE Hinton Advances in neural information processing systems 25, 1097-1105	84727	2012

- AlexNet precipitated a **renaissance of neural nets for computer vision**

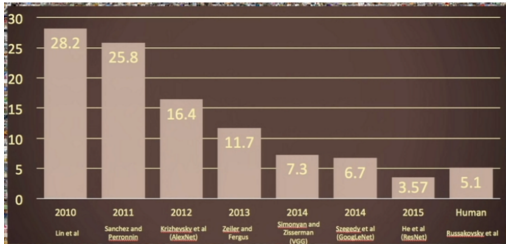


Figure: ImageNet before and after GPU-acceleration

Modern CNNs – VGGNet

- Simonyan and Zisserman (2014) developed VGG-16[26], which is similar but larger than AlexNet

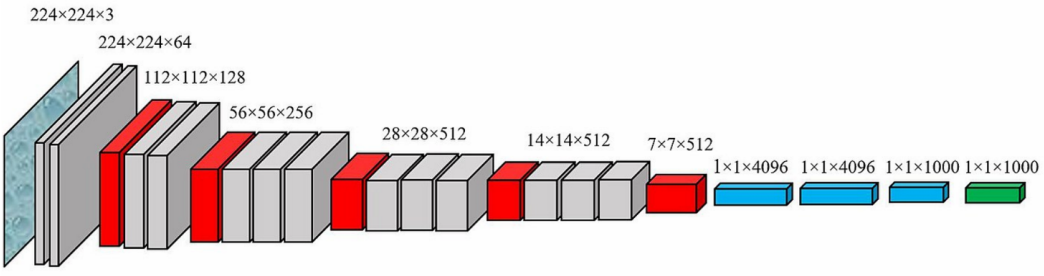


Figure: The VGG-16 Architecture with convolutions, max-pooling, fully-connected, and softmax layers

- After much testing, VGGNet worked best for our image classifier

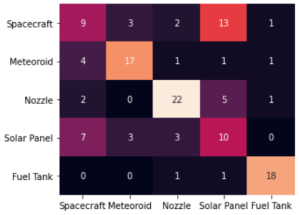
[26] K. Simonyan, A. Zisserman (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations*

Preprocessing, **training**, and **tuning** followed these steps:

1. Load a neural network pretrained on ImageNet **without** its top fully-connected layers
2. Convert images to $224 \times 224 \times 3$ and standardize data in each channel
3. Feed images through the net to **extract features** to a lower dimensional data representation (25,088-dim)
4. Randomly **split** 75% of images (and their rotations) for training and 25% of original images for testing.
5. Use 10-fold cross-validation to **tune** L^2 **regularization parameter** for logistic regression classifier[27] on the feature-exracted training set.
6. Test the model on the **unseen** test set.

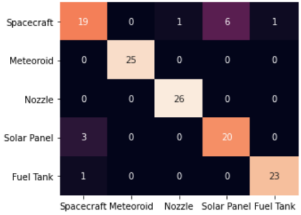
[27] Optimization by limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algoritm

Satellite Component **Image Classification** - Results



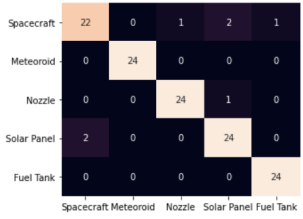
(a) NASNet Large

	precision	recall	f1-score	support
Spacecraft	0.41	0.32	0.36	28
Meteoroid	0.74	0.71	0.72	24
Nozzle	0.76	0.73	0.75	30
Solar Panel	0.33	0.43	0.38	23
Fuel Tank	0.86	0.90	0.88	20
accuracy			0.61	125
macro avg	0.62	0.62	0.62	125
weighted avg	0.61	0.61	0.61	125



(b) VGG-16

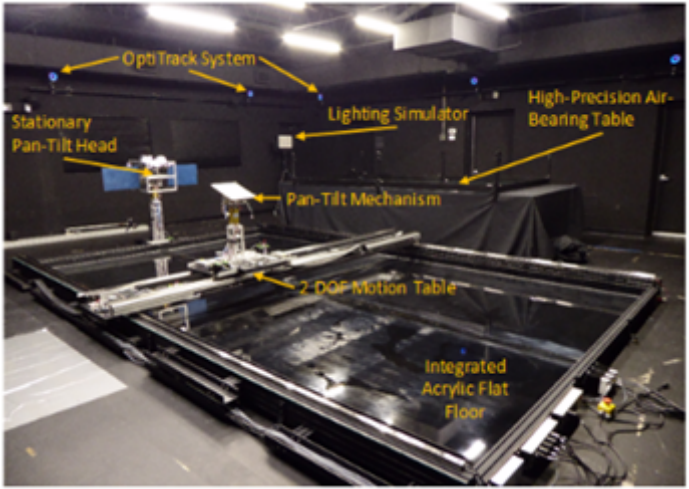
	precision	recall	f1-score	support
Spacecraft	0.83	0.70	0.76	27
Meteoroid	1.00	1.00	1.00	25
Nozzle	0.96	1.00	0.98	26
Solar Panel	0.77	0.87	0.82	23
Fuel Tank	0.96	0.96	0.96	24
accuracy			0.90	125
macro avg	0.90	0.91	0.90	125
weighted avg	0.90	0.90	0.90	125



(c) VGG-19

	precision	recall	f1-score	support
Spacecraft	0.92	0.85	0.88	26
Meteoroid	1.00	1.00	1.00	24
Nozzle	0.96	0.96	0.96	25
Solar Panel	0.89	0.92	0.91	26
Fuel Tank	0.96	1.00	0.98	24
accuracy			0.94	125
macro avg	0.95	0.95	0.95	125
weighted avg	0.94	0.94	0.94	125

ORION Lab Test Bed Set-Up



ORION Lab Video Capture

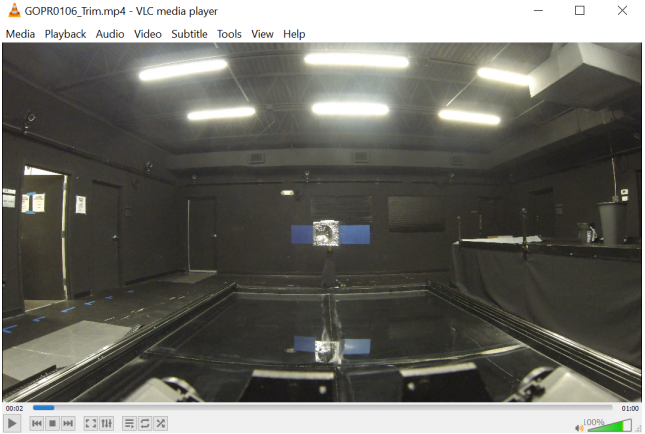
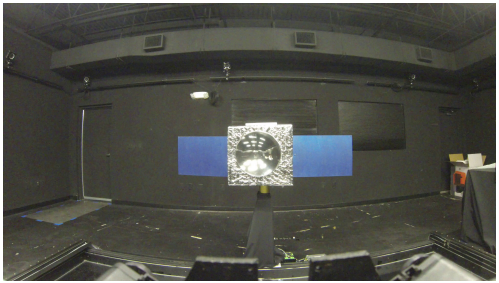


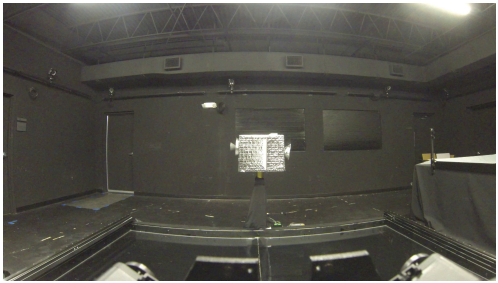
Figure: Video Captured in the Lab

Video Preparation for Testing

1. **Trim** the videos and **extract** still frames
2. Shape frames to $224 \times 224 \times 3$ and standardize in each color channel
3. Select some frames with only **1 class displayed** (experiment: solar or nozzle)



(a) **only** solar panel visible



(b) **only** nozzle visible

A Small Lab Experiment

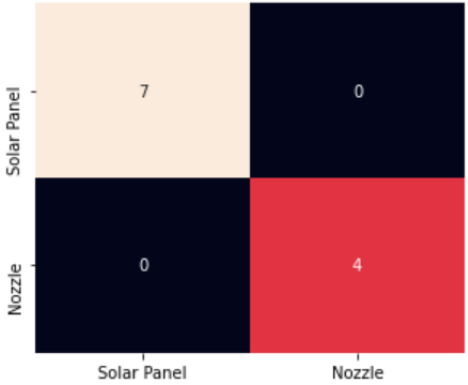
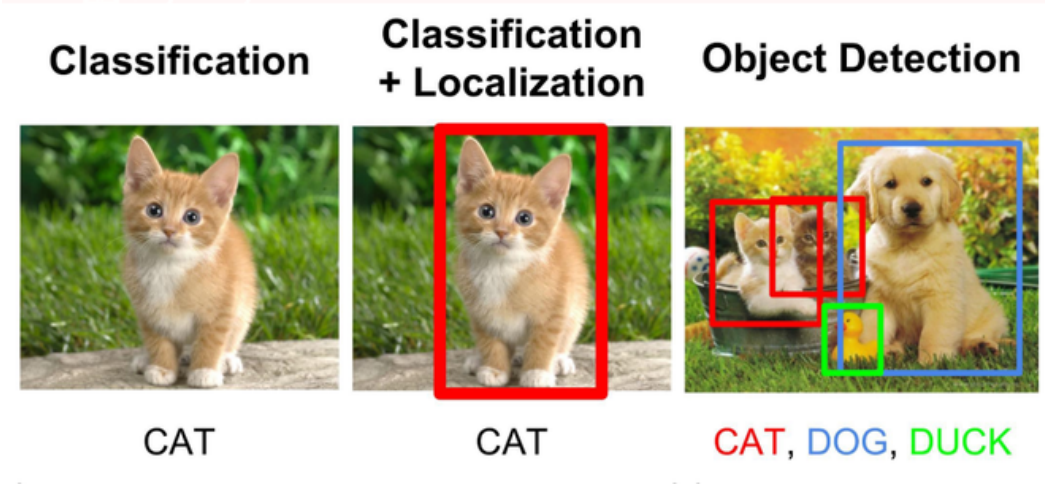


Figure: A VGG-19 experiment on a small set of data is perfect

Image Classification vs. Object Detection

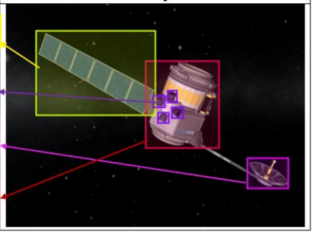


28

[28] F. Li, A. Karpathy, J. Johnson (2016). http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

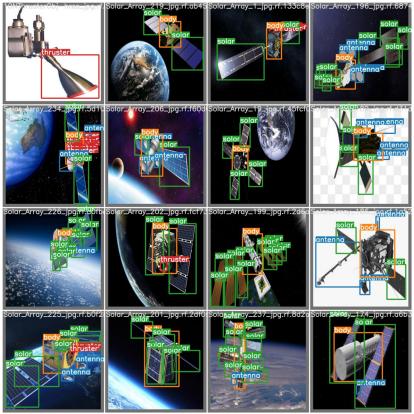
Phase 2: Satellite Component **Object Detection**

- **Gather** open-source satellite image data and **annotate** components

Class	#Annotations	Example
Solar Array	1204	
Thrusters	737	
Antennas	692	
Satellite Bodies	416	

- **Train and tune** the YOLOv5 object detector on the dataset
- **Test** the object detector on still images
- **Test** the object detector on video

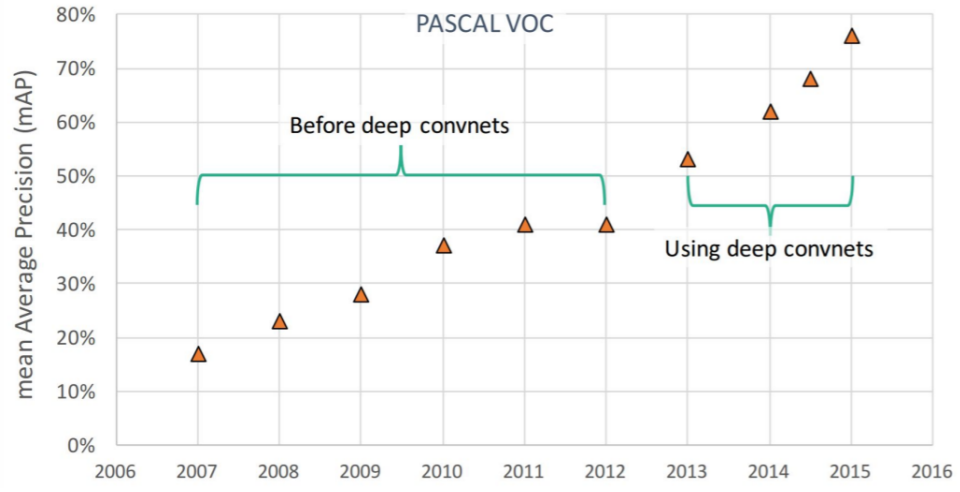
Dataset



29

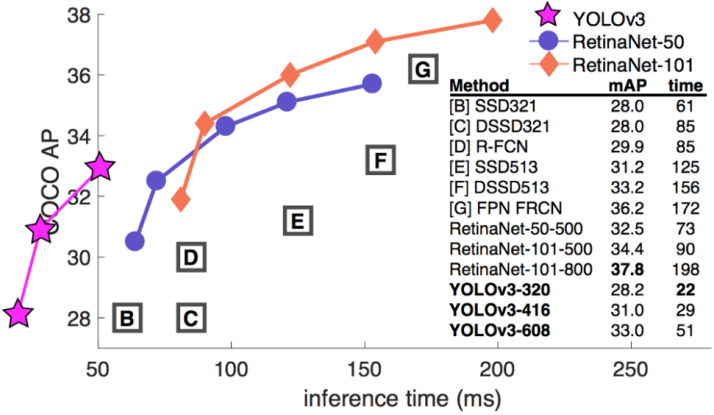
[29] Data gathered and annotated by Trupti Mahendrakar, Nathan Fischer, Andrew Ekblad, and Rebecca Smith

Object Detection Performance



30

[30] R. Girshick (2015). https://slidetodoc.com/recent-developments-in-object-detection-pascal-voc-before/



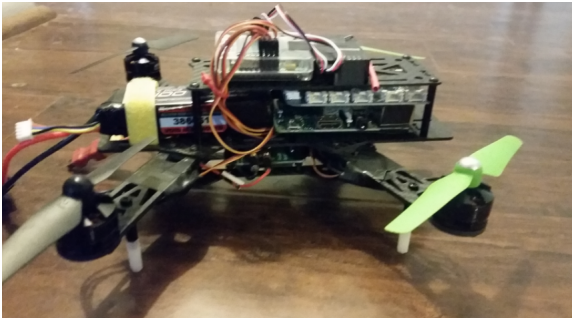
Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

31

[31] J. Redmond and A. Farhadi (2018). YOLOv3: An Incremental Improvement. https://arxiv.org/abs/1804.02767

YOLO - You Only Look Once[32]

- YOLO is the **fastest** object detection algorithm
- With **low on-board computational power**, YOLO is ideal
- Future plans to test YOLO on a Raspberry Pi on-board a drone with a camera in the lab



[32] J. Redmon, S. Divvala, R. Girshick and A. Farhadi (2016). You Only Look Once: Unified, Real-Time Object Detection, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Bounding Boxes

- Object detection requires us to answer many more questions than classification:
  1. Is there an object?

     Let $p \in \{0, 1\}$ indicate presence
  2. If it exists, where is it?

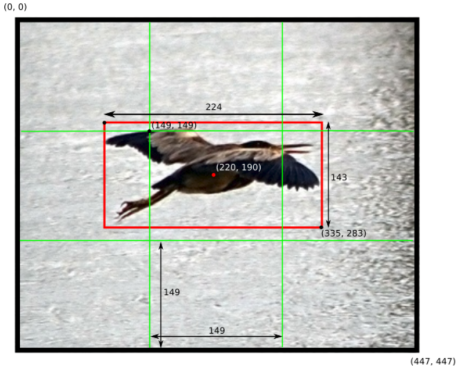
     Let (x_c, y_c) be the center of the object
  3. How large is it?

     Let w and h be the width and height of a bounding box
  4. What is it?

     Let $c_i \in \{0, 1\}$ indicate if the object is from the ith class

- Define the label of an object as

$$y = (p, x_c, y_c, w, h, c_1, \ldots, c_k)$$

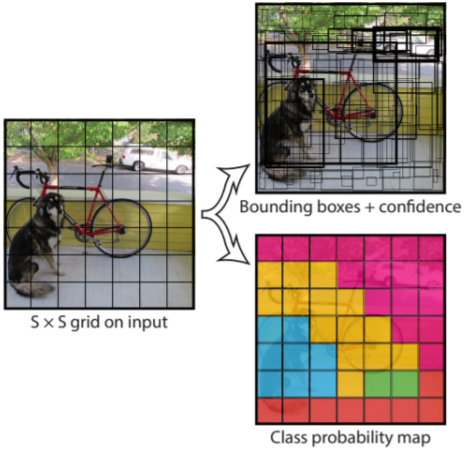


If class 2 is "bird" and $k = 4$, its label is

$$y = (1, 220, 190, 224, 143, 0, 1, 0, 0)$$

YOLO: Grid

- In **each** grid square, YOLO predicts B labels for bounding boxes, including both:
  1. Bounding boxes
  2. Predicted class probabilities
- Predict grid-square probabilities by average of probabilities weighted by objectness p of boxes
- How do we choose between boxes detecting the same object?



33

[33] J. Redmon, S. Divvala, R. Girshick and A. Farhadi (2016). You Only Look Once: Unified, Real-Time Object Detection, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*

YOLO Loss Function

- The loss function is a sum of three parts:

1. Classification Loss:

$$\sum_{i=0}^{S^2} \mathbb{1}_{\{p_i=1\}} \sum_{j=1}^{k} (c_{ij} - \hat{c}_{ij})^2$$

2. Localization Loss:

$$\lambda_{\text{coordinates}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{\{\text{best box for cell } i=j\}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right]$$

3. Confidence Loss

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} \left[\mathbb{1}_{\{\text{best box for cell } i=j\}} + \left(1 - \mathbb{1}_{\{\text{best box for cell } i=j\}}\right) \lambda_{\text{no object}} \right] (p_i - \hat{p}_i)^2$$

YOLO: Non-Max Suppression

- We will suppress less-good boxes through the following procedure.
- Discard boxes with $p < 0.6$
- While more than 1 box remains,
  - Choose the box with the highest p as a prediction
  - Discard remaining boxes with IoU ≥ 0.5 with the box chosen above

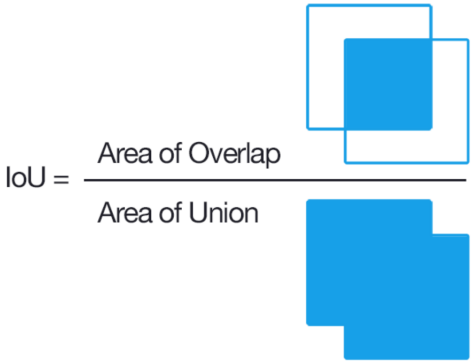
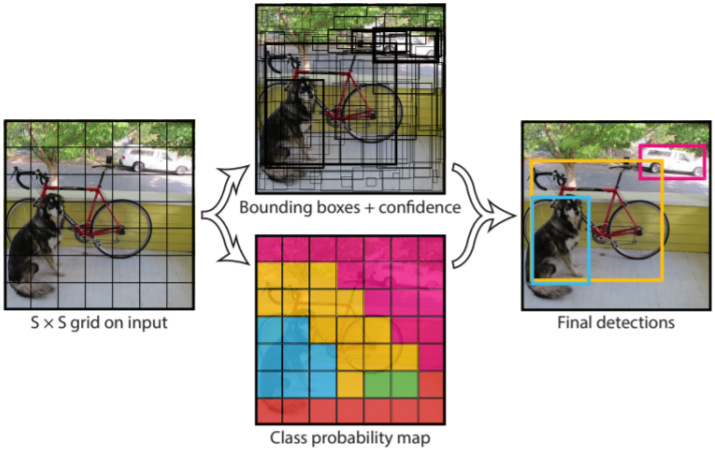


Figure: Intersection over Union

YOLO Detection

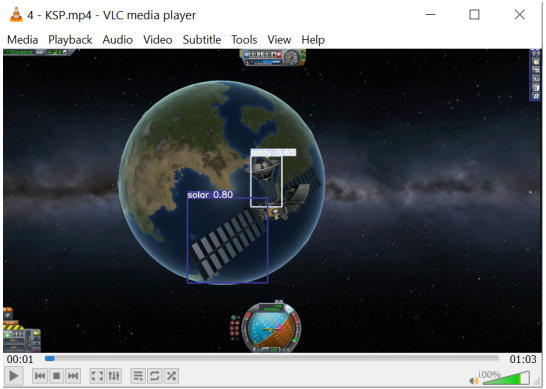


34

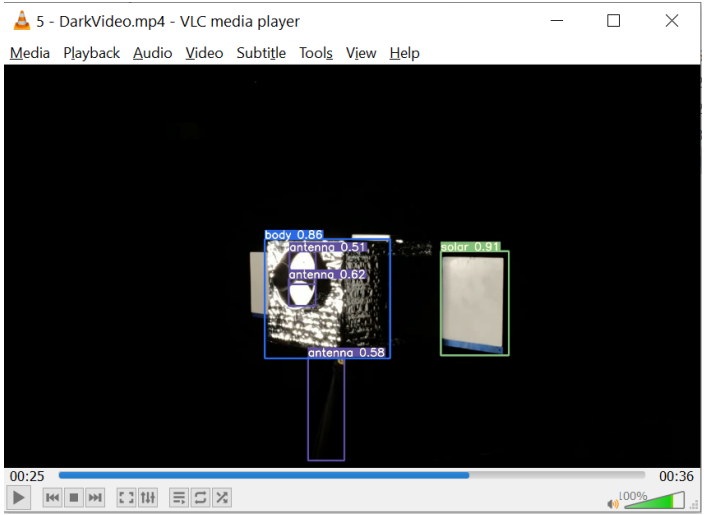
[34] J. Redmon, S. Divvala, R. Girshick and A. Farhadi (2016). You Only Look Once: Unified, Real-Time Object Detection, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*

Performance in a Video Game Capture

- Still images are too easy: we need it to work on **video feeds**
- We find an interesting testing ground with the video game Kerbal Space Program



Performance on Video Captured in the Lab



Phase 3

- Improve accuracy by exploiting **temporal patterns**
  - In videos, YOLO simply detects objects in each frame in isolation
  - Build a time-series regression model to make **short-term predictions** when YOLO "loses" an object
  - Requires logic to differentiate between correct and incorrect "losses"
- Test on **restrictive hardware**
  - Tiny YOLOv3 should run **at least 4.28 frames per second** on Raspberry Pi + Neural Compute Stick[35]
  - Raspberry Pi + NCS + PiCamera should mount on a quad-copter to **simulate in-orbit computational resources and maneuverability**
- **Ultimate Goal: Convince Trupti and Markus to let me drive the drone**

[35] A. Rosebrock (2020). YOLO and Tiny-YOLO object detection on the Raspberry Pi and Movidius NCS. https://www.pyimagesearch.com/2020/01/27/yolo-and-tiny-yolo-object-detection-on-the-raspberry-pi-and-movidius-ncs/

References

Much was cited above, but there are a few unnamed sources implicitly used herein.

1. F. Li, A. Karpathy, et. al. (2020). CS231n: Convolutional Neural Networks for Visual Recognition course lectures and notes. `http://cs231n.stanford.edu/2020/`
2. A. Ng. Lectures on Object Detection. Coursera: Convolutional Neural Networks, Week 3. `https://www.coursera.org/learn/convolutional-neural-networks/home/week/3`
3. G. Jocher, Ultralytics (2021). YOLOv5. `https://github.com/ultralytics/yolov5`
4. T. Mahendrakar, R. T. White, M. Wilde (2021). Real-time Satellite Component Recognition YOLO V5. (accepted at 35th Annual Small Satellite Conference)
5. T. Mahendrakar, R. T. White, M. Wilde, A. Rivikin, J. Cutler, K. Watkins, A. Ekblad, N. Fischer (2021) Use of artificial intelligence for feature recognition and flightpath planning around non-cooperative resident space objects. (under consideration at AIAA ASCEND 2021)