Deep Learning Final Project

Group No. 3

Index

Original Plan

Duckietown

- Started in MIT, 2016

Fig. 1. In Duckietown, inhabitants (duckies) are transported via an autonomous mobility service (Duckiebots). Duckietown is designed to be inexpensive and modular, yet still enable many of the research and educational opportunities of a full-scale self-driving car platform.

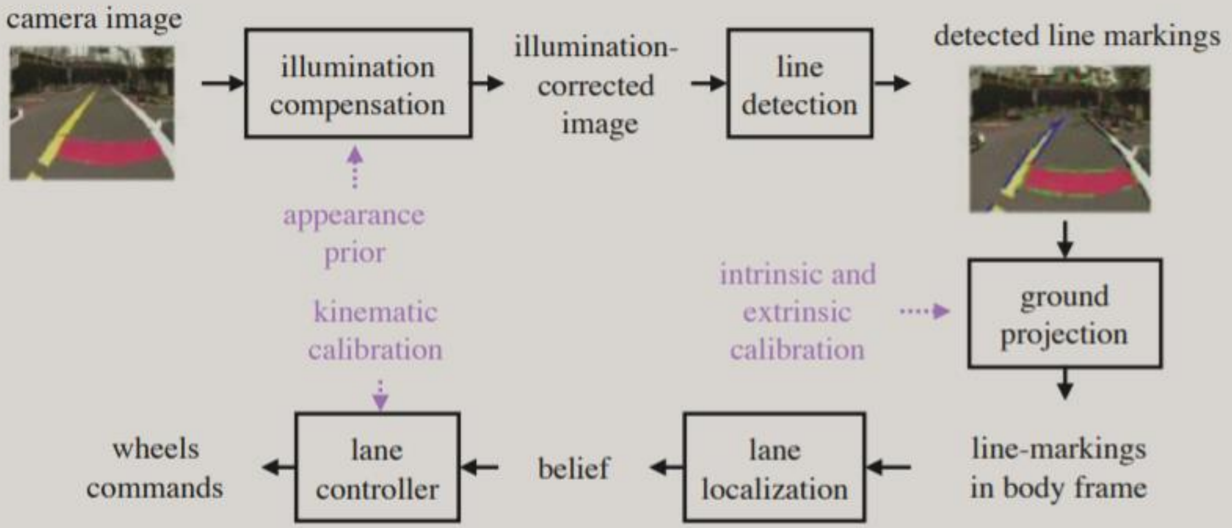
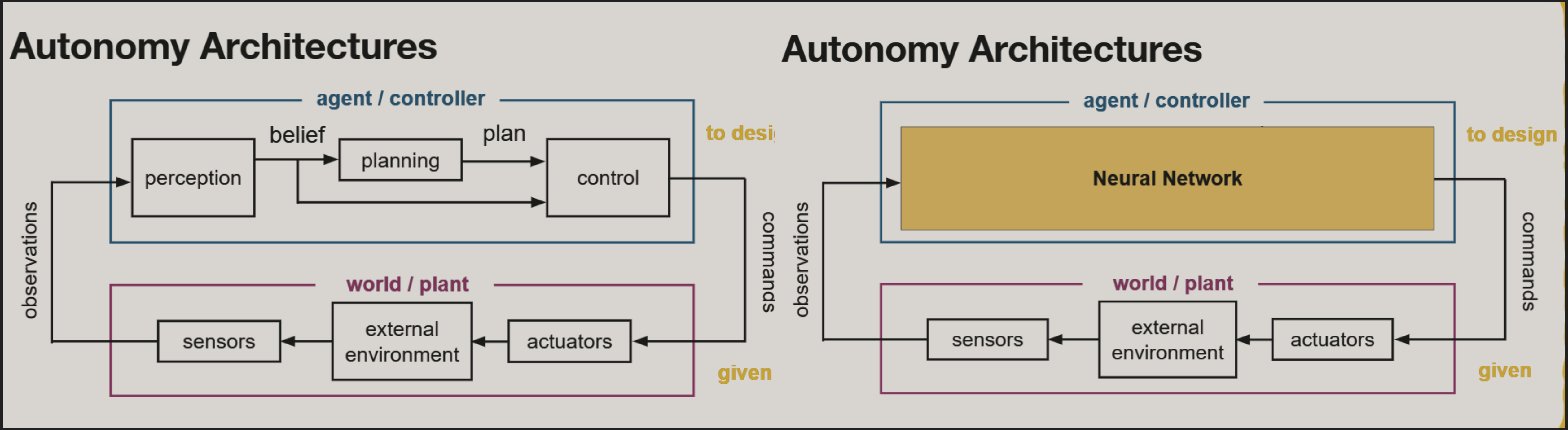


Fig. 2. The lane following pipeline runs on-board at 10Hz with a resolution of 320x240 and a latency of 110ms. The purple text indicates prior information.

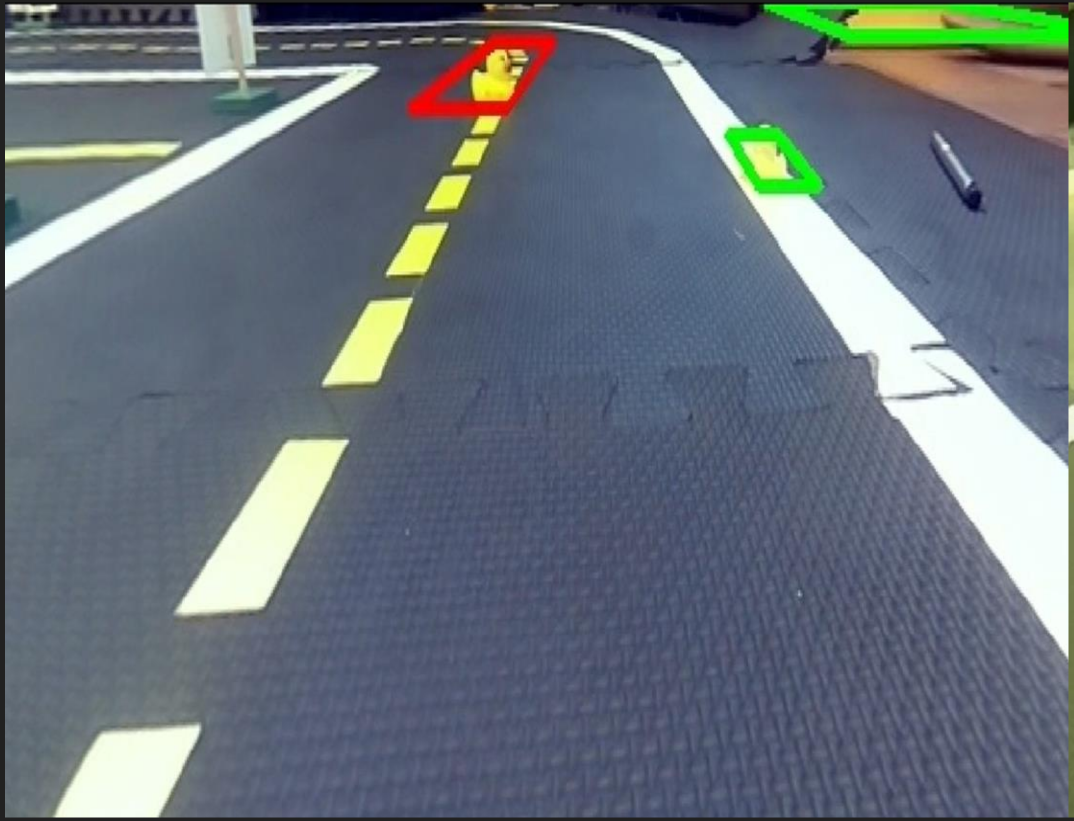
Autonomous Driving

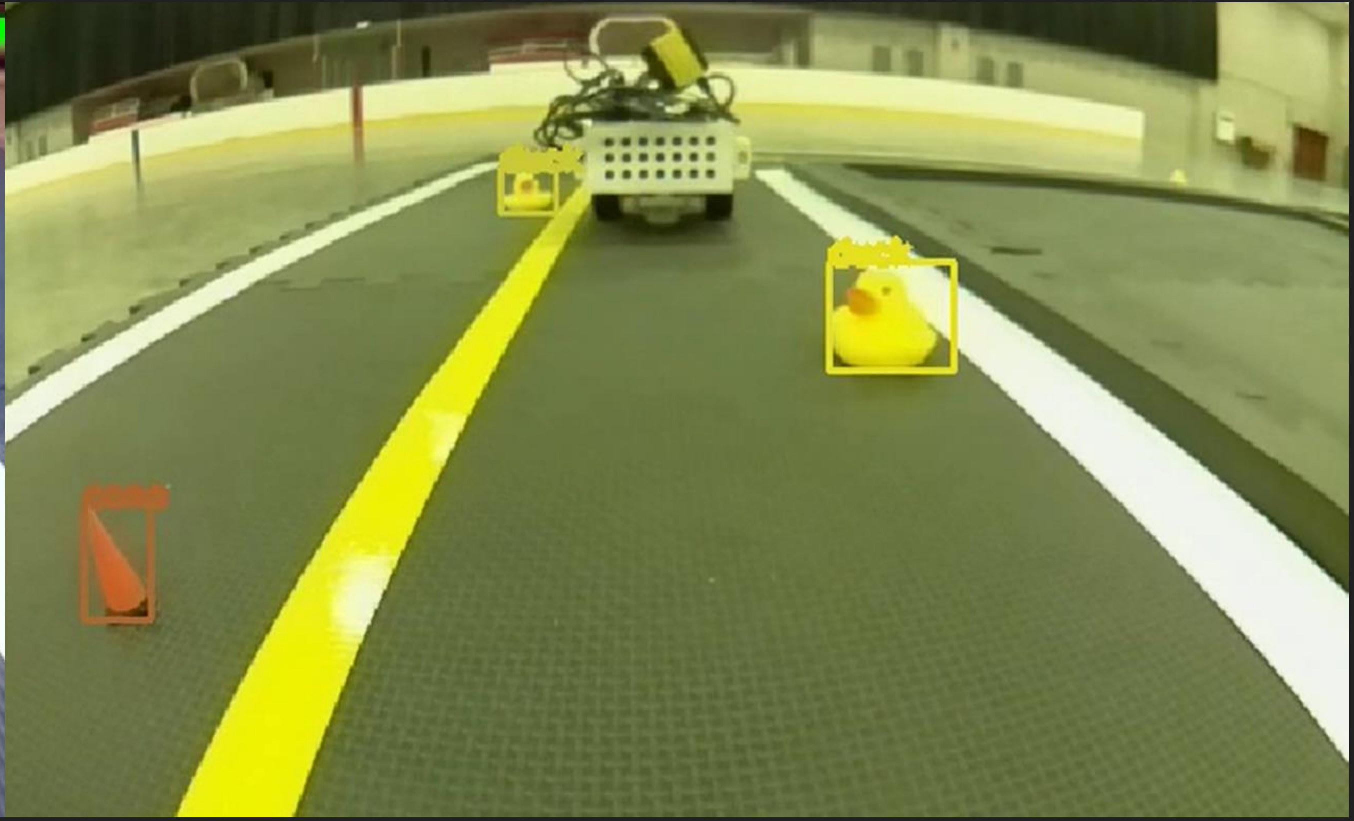
- Supervised learning: object detection (bounding boxes and recognition) + steering angle and speed



Object Detection

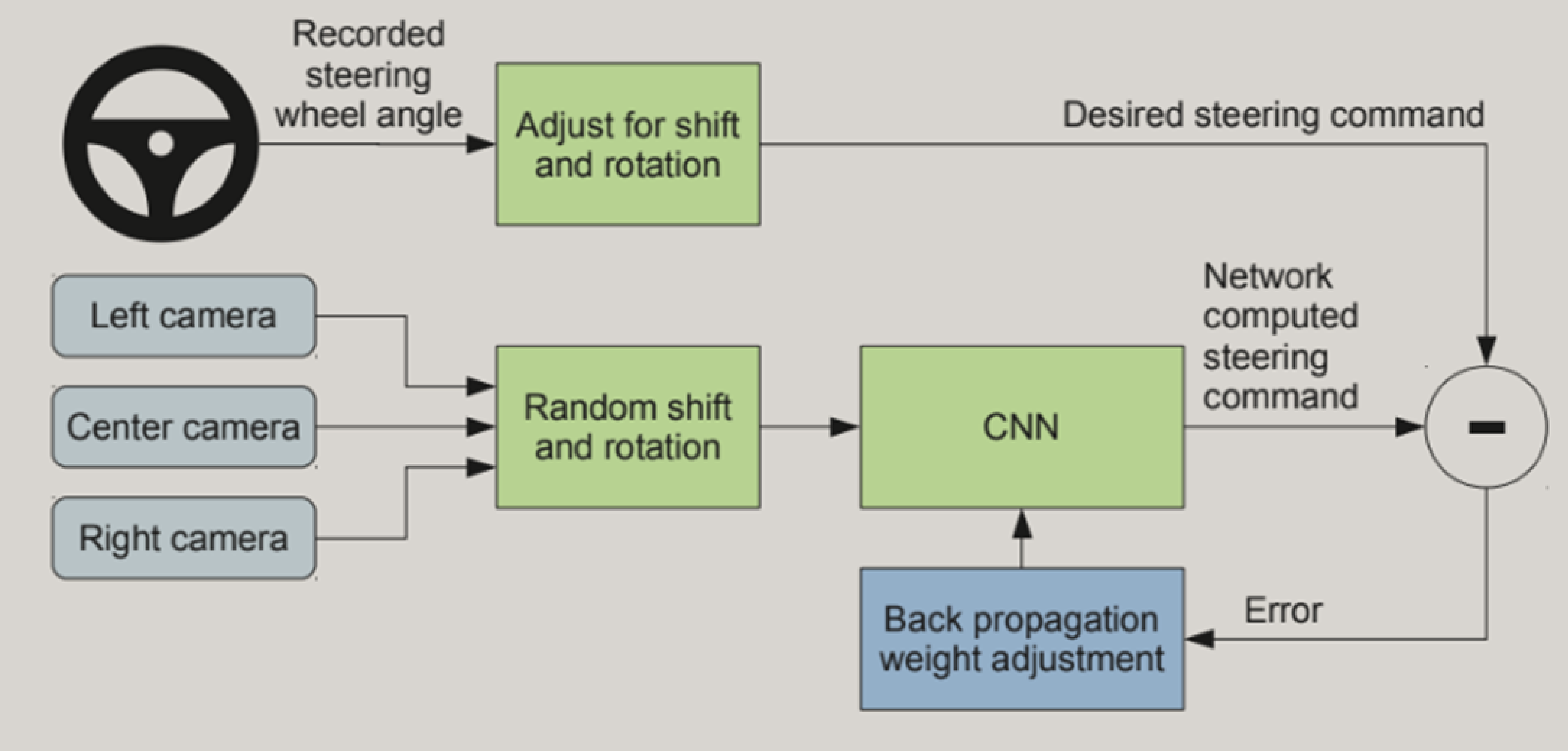
- Object Classification + Bounding Boxes





Self-driving: motor control

- Given image: specific steering angle and speed / motor voltage



Preprocessing

- Given image (X) >> 15 angles
  (divide range of angles into 15) (Y)
- Resize image to 101x101
- Speed is constant

```
joystick/24285.jpg 1.36500000954 0.386400014162
joystick/24286.jpg 2.11500000954 0.386400014162
joystick/24287.jpg 1.60500001907 0.386400014162
joystick/24288.jpg 1.60500001907 0.386400014162
joystick/24290.jpg 1.60500001907 0.386400014162
joystick/24291.jpg 1.60500001907 0.386400014162
joystick/24292.jpg 1.60500001907 0.386400014162
joystick/24293.jpg 6.10500001907 0.386400014162
joystick/24294.jpg 5.59499979019 0.386400014162
joystick/24295.jpg 5.83500003815 0.386400014162
joystick/24296.jpg 5.3250002861 0.386400014162
joystick/24297.jpg 5.3250002861 0.386400014162
joystick/24298.jpg 3.55500006676 0.386400014162
joystick/24300.jpg 2.80500006676 0.386400014162
joystick/24301.jpg 5.3250002861 0.386400014162
joystick/24302.jpg 5.3250002861 0.386400014162
joystick/24303.jpg 4.30499982834 0.386400014162
joystick/24304.jpg 4.81500005722 0.386400014162
joystick/24305.jpg 4.30499982834 0.386400014162
joystick/24306.jpg 4.06500005722 0.386400014162
joystick/24307.jpg 3.55500006676 0.386400014162
joystick/24308.jpg 2.80500006676 0.386400014162
```

Models

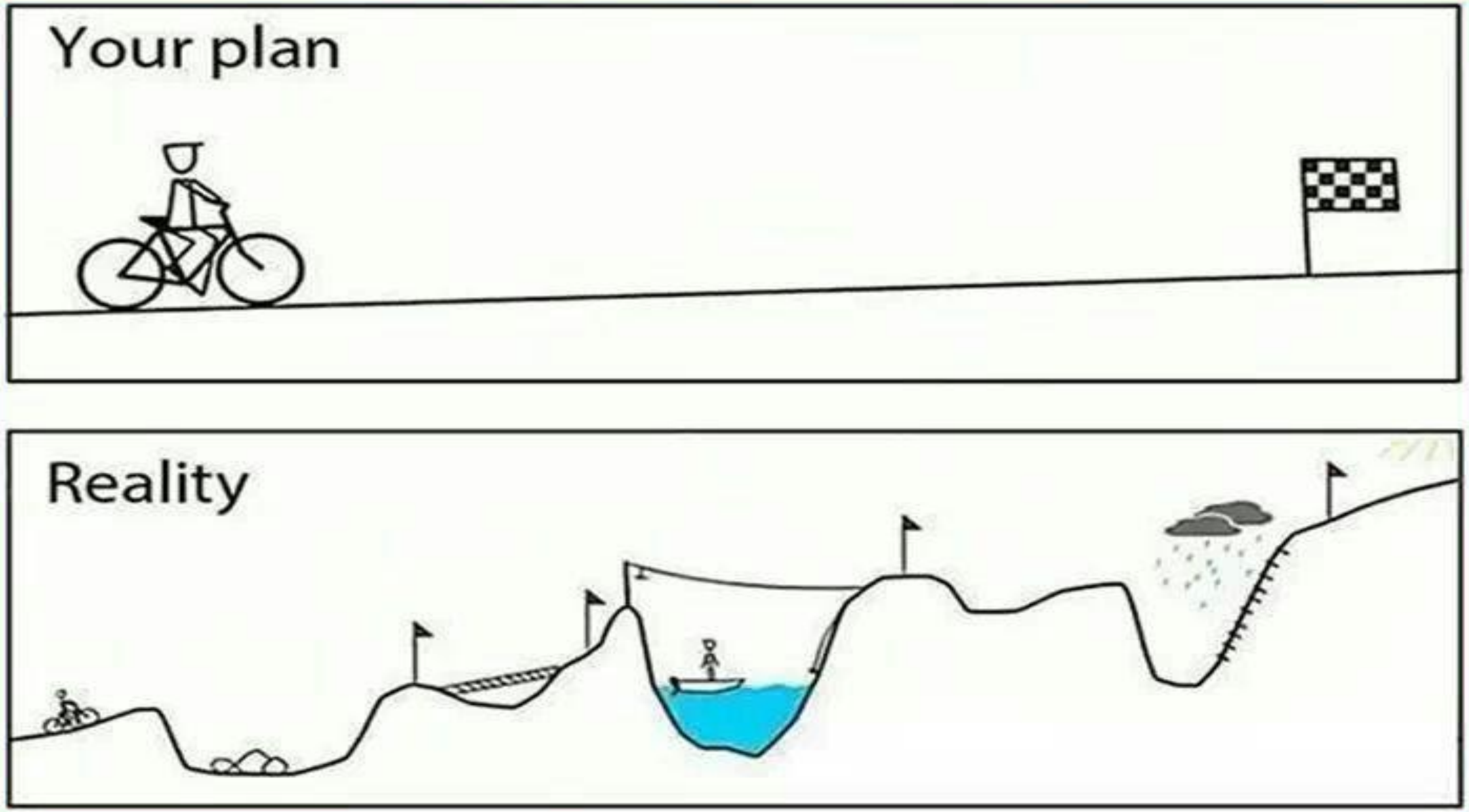
- V1: Conv (I: 101x101, F: 10x10, S:1, P:0, ReLu) + Max Pooling (2x2) + Conv(I: 46x46, F: 9x9, S:1, P:0, ReLu) + FC (1444) + Softmax (O: 15) using only data provided by TA (DS_0)
- V2: Conv (I: 101x101, F: 10x10, S:1, P:0, ReLu) + Max Pooling (2x2) + Conv(I: 46x46, F: 9x9, S:1, P:0, ReLu) + FC (1444) + Softmax (O: 15) using DS_0 + Data recorded by us (DS_1)

Models

- V3.1: VGG-16 using DS_0 + DS_1
- V3.2: InceptionResNetV2 using DS_0 + DS_1
- V3.3: MobileNet using DS_0 + DS_1
- V4: V3.1/3.2/3.3 (choose best) using DS_0 + DS_1 + Data from the internet (DS_2) and/or [DS_0 + DS_1] with transformations

Models

- V5: V4 but change classification output to more than 15 (30/60?)
- V6 (Final): V5 + Higher resolution images
- Above two require changing ncs_joy_mapper_node.py



What We Actually Did

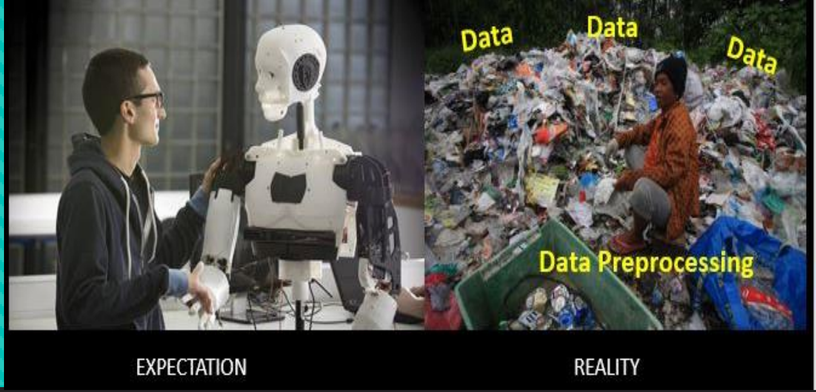
Pre-processing

- Bag2txt.py: takes .bag file and outputs images with angle (15 classes)
- Challenges:
  - Weird bugs: invalid classes (-1) and more pictures than labels
  - Imbalanced classes
  - No scalability

```python
image_name = str(self.n)+ ".jpg"
cv2.imwrite("ele/"+image_name, cv_image)

if self.i == 9:
    self.test_arr[self.omega] += 1
    if self.test_arr[self.omega] > 300:
        continue
    f2.write("ele/"+image_name+" "+str(self.omega)+"\n")
    self.i = 0
else:
    self.train_arr[self.omega] += 1
    if self.train_arr[self.omega] > 300:
        continue
    f1.write("ele/"+image_name+" "+str(self.omega)+"\n")
    self.i += 1
if (self.n%1000 == 0):
    print("image crop:",self.n)
self.n += 1
self.t = 0
```

Pre-processing



- Modifications to bag2txt.py to allow for (easy) inclusion of new datasets, only one label.txt file, and output more or less imbalanced sets
- Datasets:
  - 4 K: bag2txt.py V1, only TA data
  - 6 K: bag2txt.py with our initial set of modifications, only TA data
  - 24 K: bag2txt.py without care for class balance, only TA data
  - 3 K: only ours
  - 9 K: TA data (balanced) + ours
  - 27 K: TA data (imbalanced) + ours

Data Augmentation

- Histogram of current classes, calculate mean and standard deviation
- Based on this generate new images (and labels) by adding (random) Gaussian and Poisson noise
- Reduced standard deviation of sets

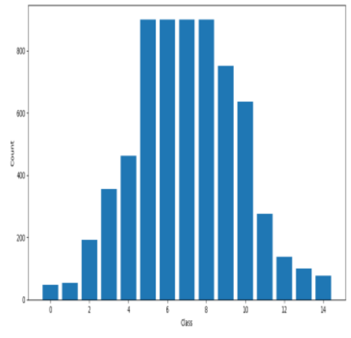
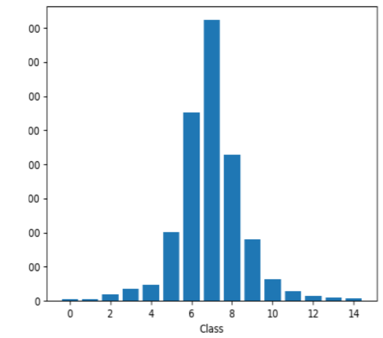
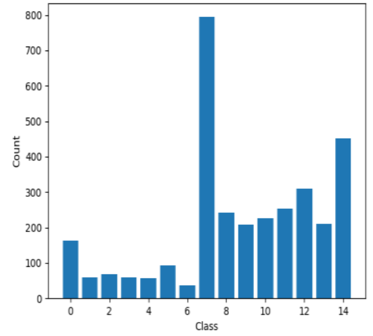
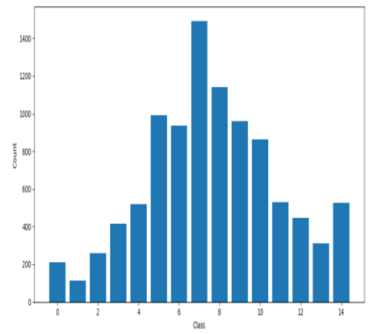
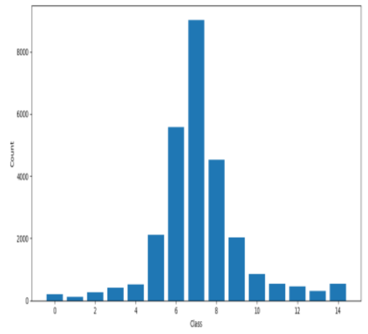
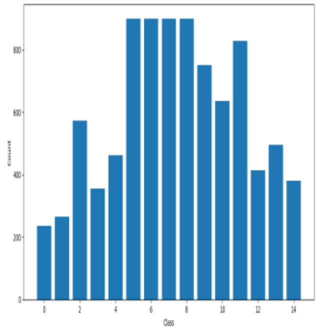
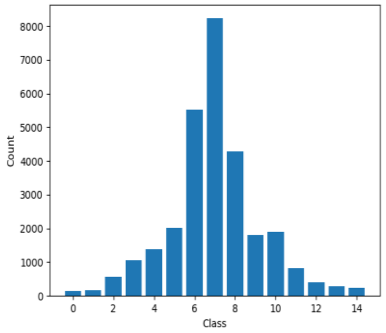
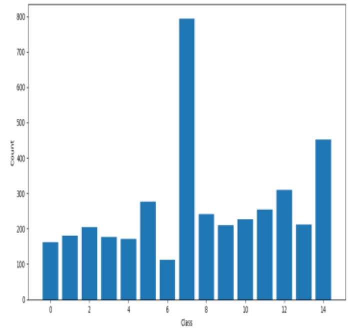
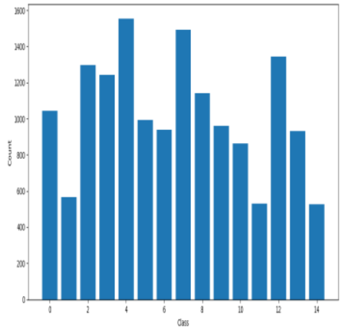
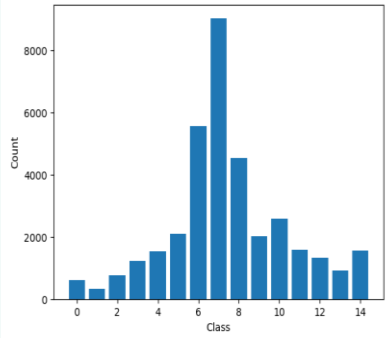
```
183M    ./data_unbalanced_noduck_taonly_noaug
51M     ./data_balanced_noduck_taonly_noaug
218M    ./data_unbalanced_noduck_taonly_aug
4.0K    ./saved_models
32M     ./data_unbalanced_duck_oursonly_aug
26M     ./data_unbalanced_duck_oursonly_noaug
11G     ./trash
74M     ./data_balanced_combined_noaug
68M     ./data_balanced_noduck_taonly_aug
132K    ./plots
118M    ./data_balanced_combined_aug
```

```
1.4G    ./data_unbalanced_noduck_taonly_noaug
386M    ./data_balanced_noduck_taonly_noaug
218M    ./data_unbalanced_noduck_taonly_aug
4.0K    ./saved_models
32M     ./data_unbalanced_duck_oursonly_aug
209M    ./data_unbalanced_duck_oursonly_noaug
11G     ./trash
582M    ./data_balanced_combined_noaug
68M     ./data_balanced_noduck_taonly_aug
132K    ./plots
118M    ./data_balanced_combined_aug
```

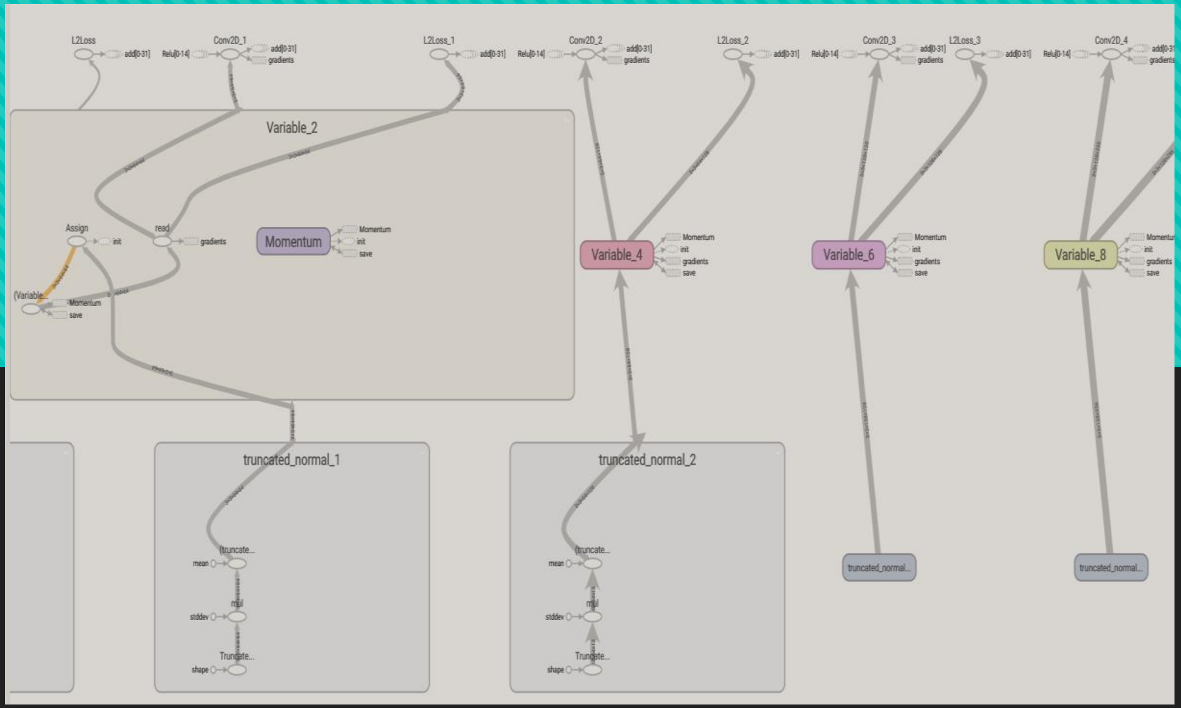
Data Augmentation

```
Pictures before augmentation:  27464
data_unbalanced_combined_aug/labels.txt
Numbers per class before aug:  {0: 209, 1: 113, 2: 259, 3: 414, 4: 518, 5: 2117
, 6: 5573, 7: 9028, 8: 4536, 9: 2020, 10: 862, 11: 530, 12: 448, 13: 310, 14: 5
27}
Avg number per class if balanced:  1830.9333333333334
Standard deviation pre-data-aug:  2497.6985798041273
Current amount of pictures:  28000
Current amount of pictures:  29000
Current amount of pictures:  30000
Current amount of pictures:  31000
Current amount of pictures:  32000
Current amount of pictures:  33000
Current amount of pictures:  34000
Current amount of pictures:  35000
Total pictures after augmentation:  35844
Numbers per class after augmentation:  {0: 627, 1: 339, 2: 777, 3: 1242, 4: 155
4, 5: 2117, 6: 5573, 7: 9028, 8: 4536, 9: 2020, 10: 2586, 11: 1590, 12: 1344, 1
3: 930, 14: 1581}
Standard deviation post-data-aug:  2243.5571101861133
```

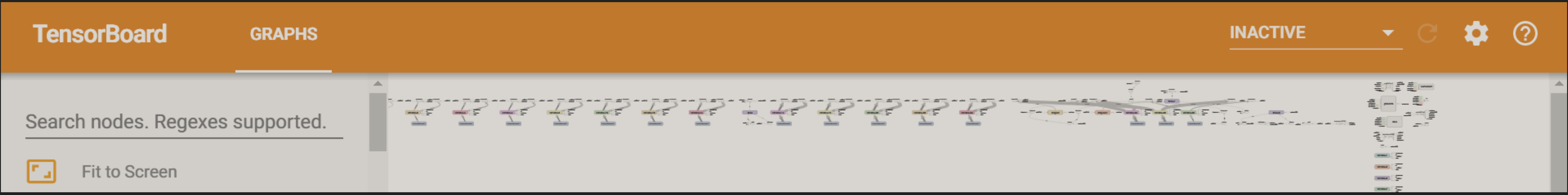
Data Augmentation

AugOrNot/Dataset		TA Only (Imbalanced)	Ours Only (Imbalanced)	Combined (Balanced)	Combined (Imbalanced)
NoAug					
DataAug					

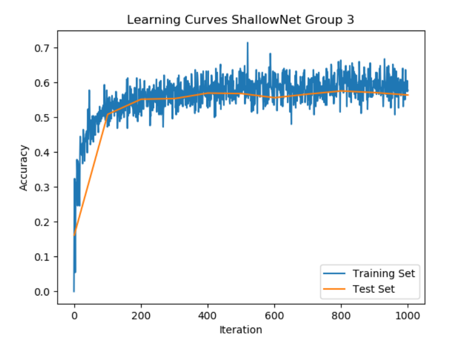
Architecture

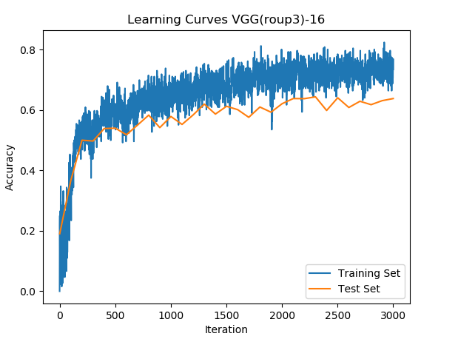


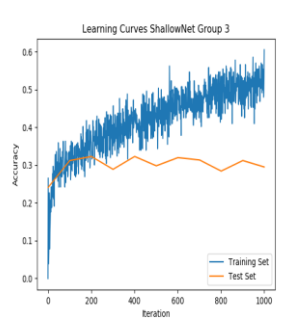
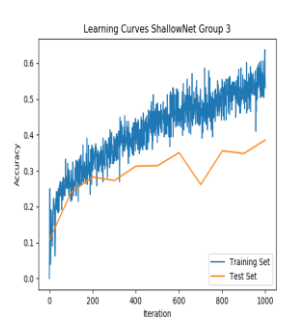
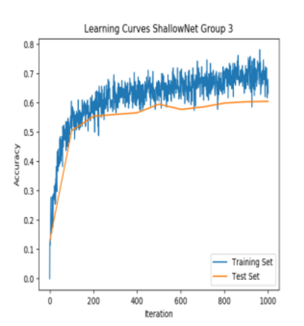
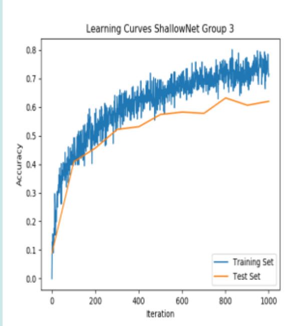
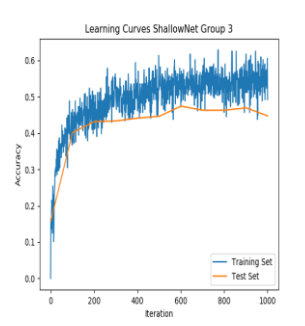
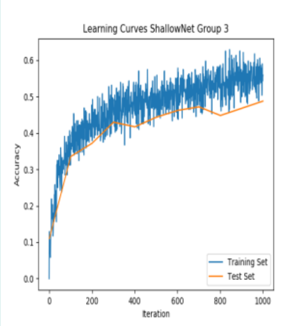
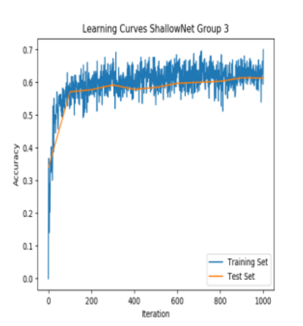
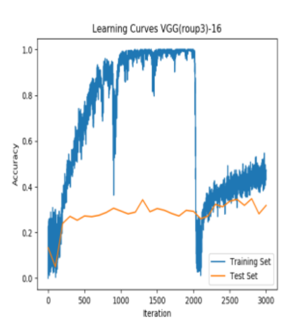
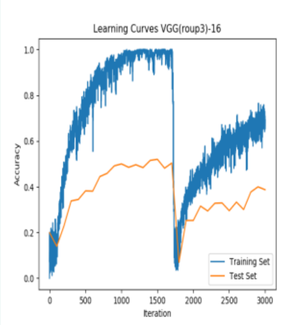
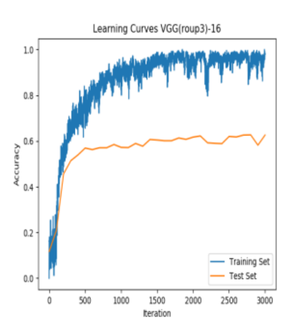
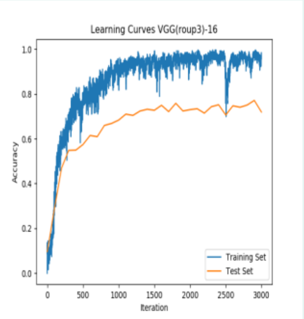
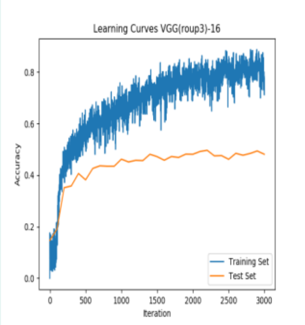
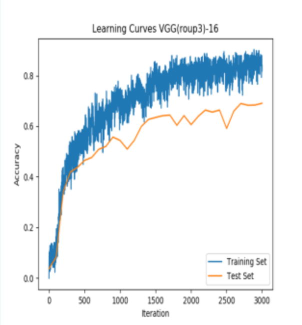
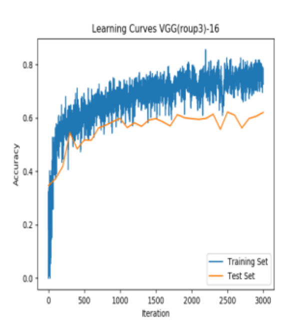
- 30+ models
- 2 baseline architectures:
  - VGG(roup3)-16: VGG-16 with no dropout and L2 regularization on all layers
  - ShallowNet: 2 Conv Layers + 2 FCs



Learning Curves





Network Architecture /Examples	3232	3978	6682	8992	9712	15416	24232
ShallowNet (3232 – 28896)							
VGG(roup3)-16 (3232 – 28896)							

Training

- Most models were trained using RTX 2080 Ti, 16 GB RAM
- VGG(roup3)-16 for either 1000/ 3000 epochs
- ShallowNet for 1000 epochs
- Different L2 regularization coefficient from 0.5 to 0.0005

Deployment

- The most successful model was trained for 100 epochs, unknown regularization (probably 0.05) and smallest net
- Hypothesis:
  - Training so short it doesn't get to overfit to training/testing set
  - Model so small inference time is closer to real-time performance

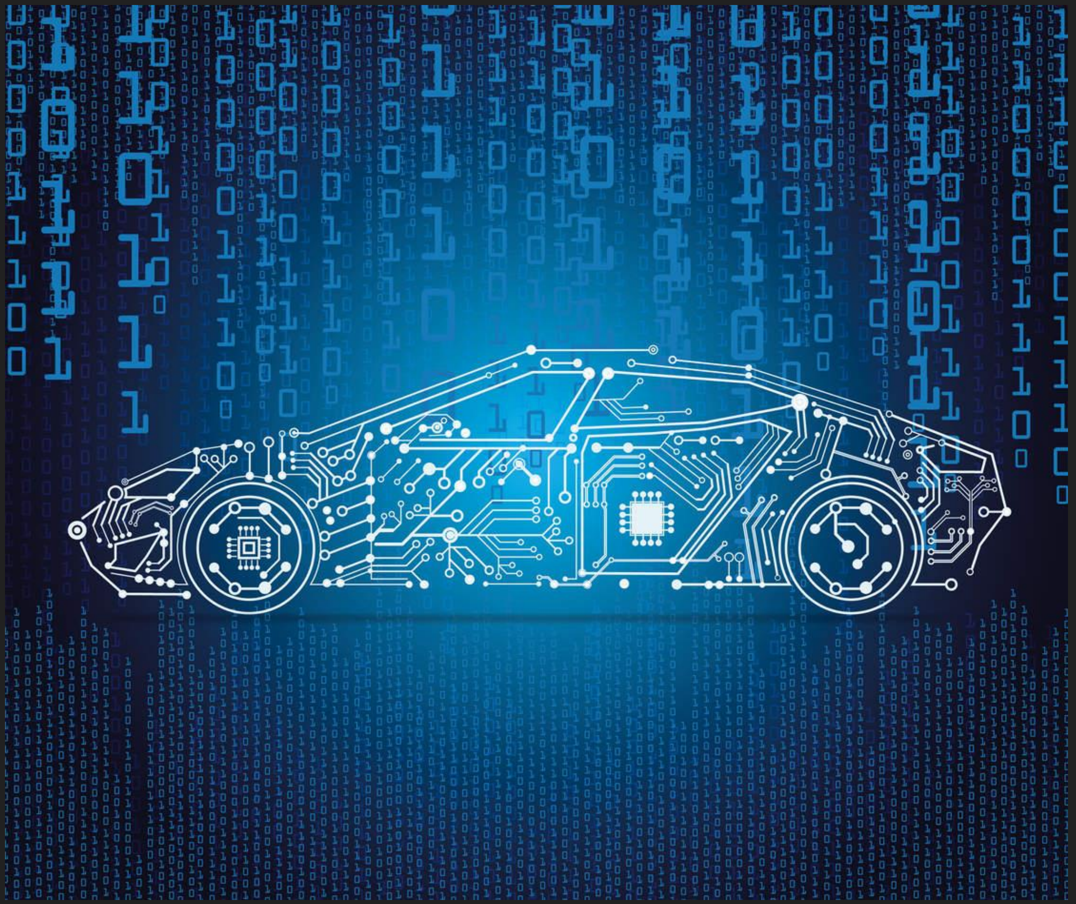
```diff
 filter_size1 = 5
-num_filters1 = 16
 filter_size2 = 5
-num_filters2 = 36
-fc_size = 128
```

```diff
 filter_size1 = 5
+num_filters1 = 32
 filter_size2 = 5
+num_filters2 = 64
+fc_size = 512
```

Discussion

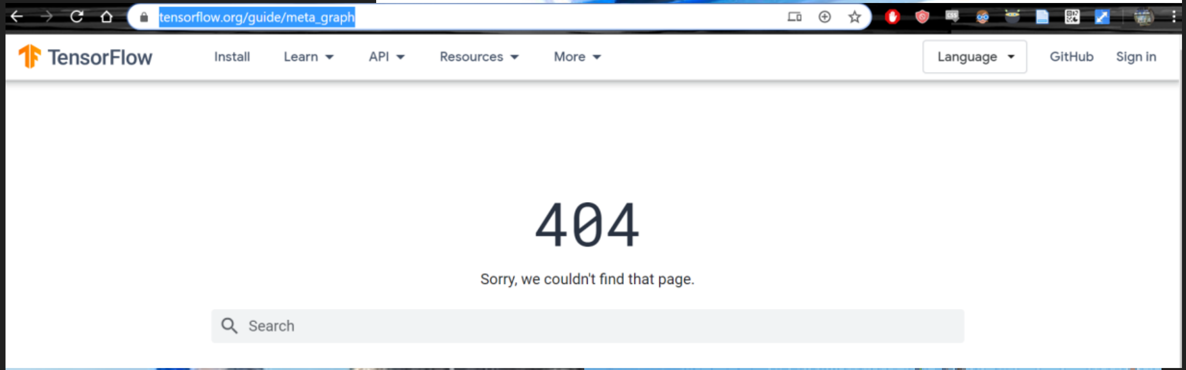
General Problems

- Lack of time: require good organization and work distribution
- Many parameters to work with: integration of software and hardware
  - Hard to troubleshoot sources of error



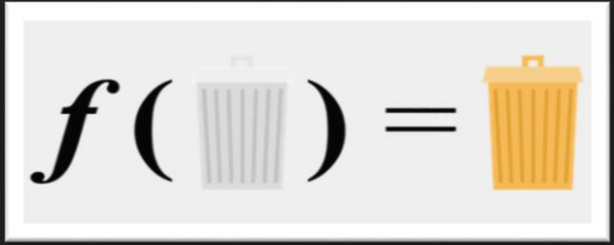
Deep Learning Problems

- Compilation of graph (no ArgMax/BatchNorm?)
- TensorFlow 1.xx compatibility/lack of documentation
- Inference time? Camera FPS?



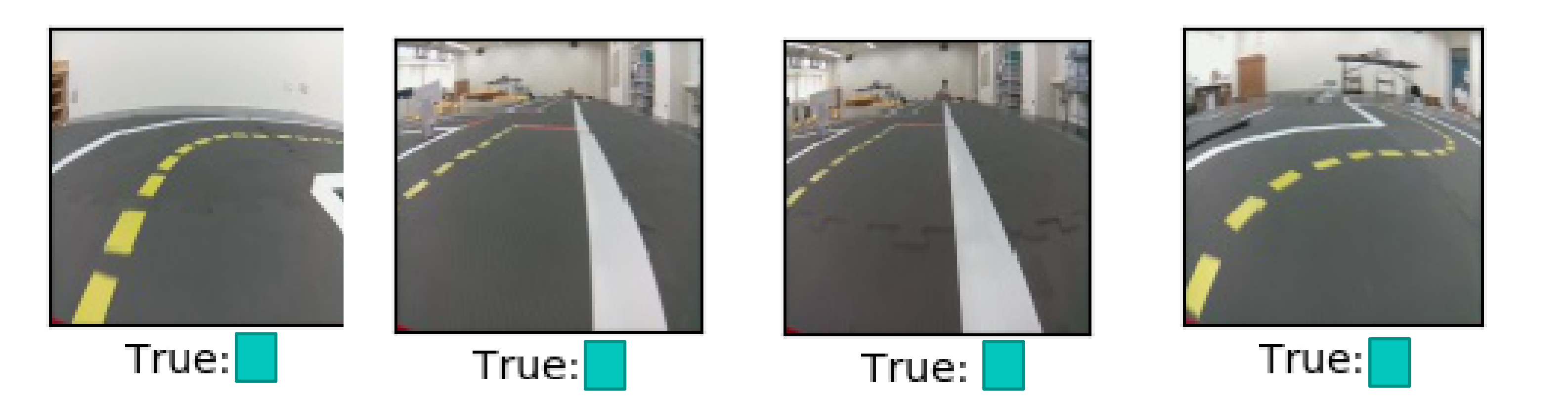
Deep Learning Problems

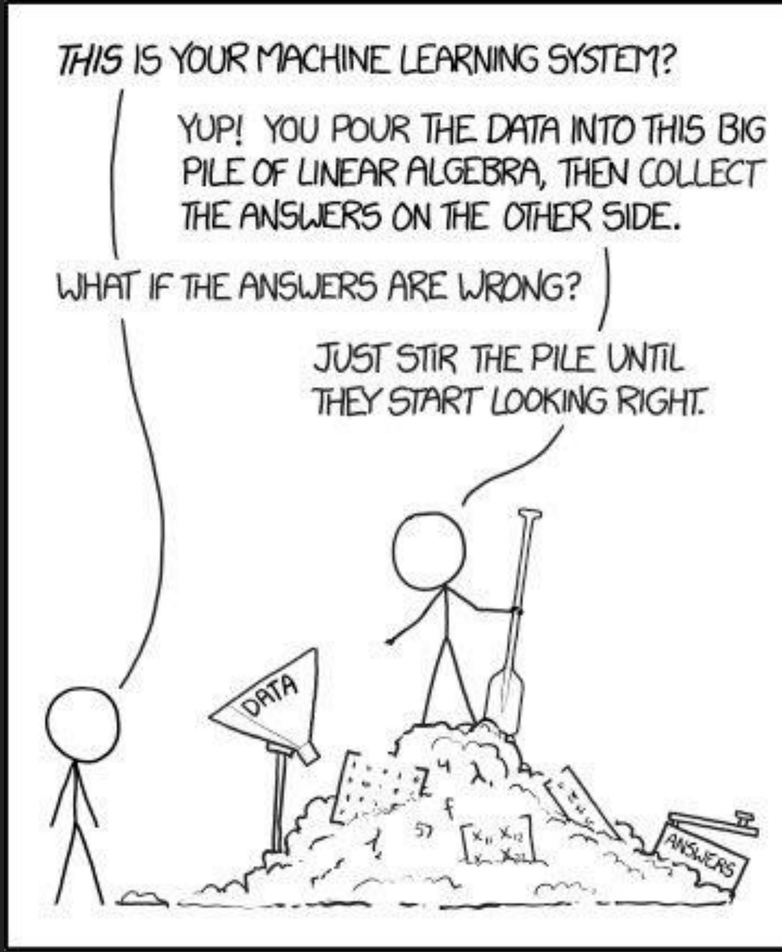
- Quality of Data: GIGO



- Explainability:

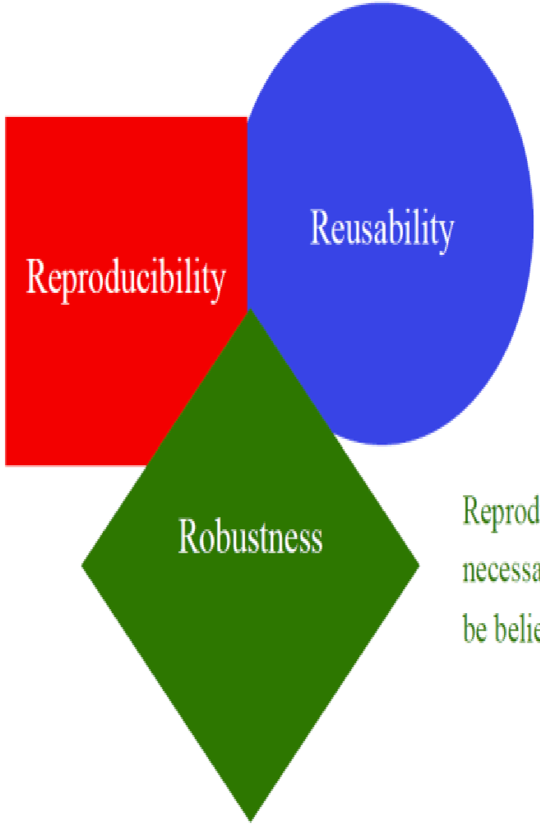
  What is the model actually learning?





DL Problems: Reproducibility

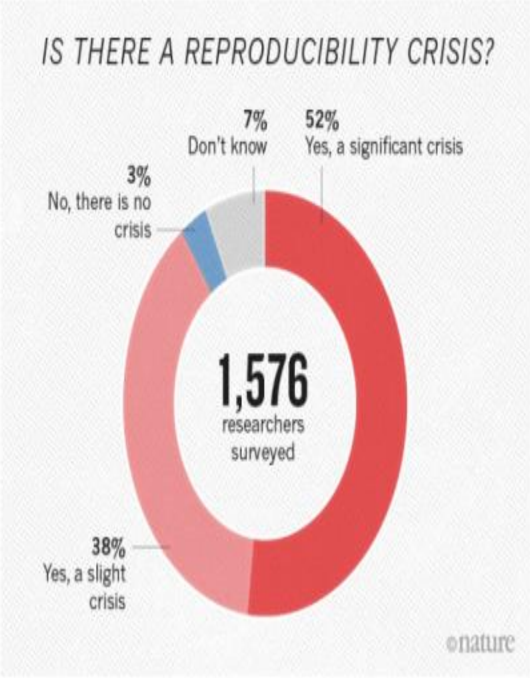
"**Reproducibility** refers to the ability of a researcher to duplicate the results of a prior study….

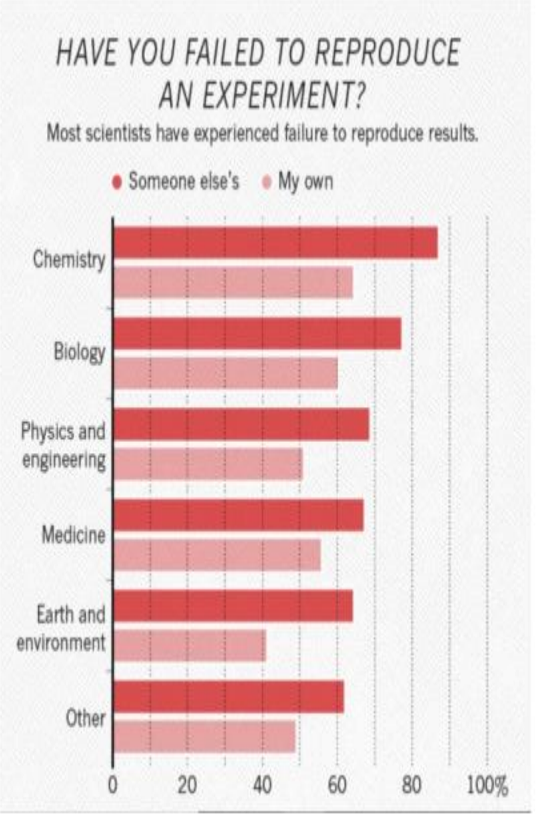


Using the same materials as were used by the original investigator.

Reproducibility is a minimum necessary condition for a finding to be believable and informative."

Bollen et al.
National Science Foundation, 2015.





DL Problems: Reproducibility

Evaluation Metrics:

A. Technical replicability
   1. Code available
   2. Public dataset
B. Statistical replicability
   1. Variance reported
C. Conceptual replicability
   1. Multiple datasets

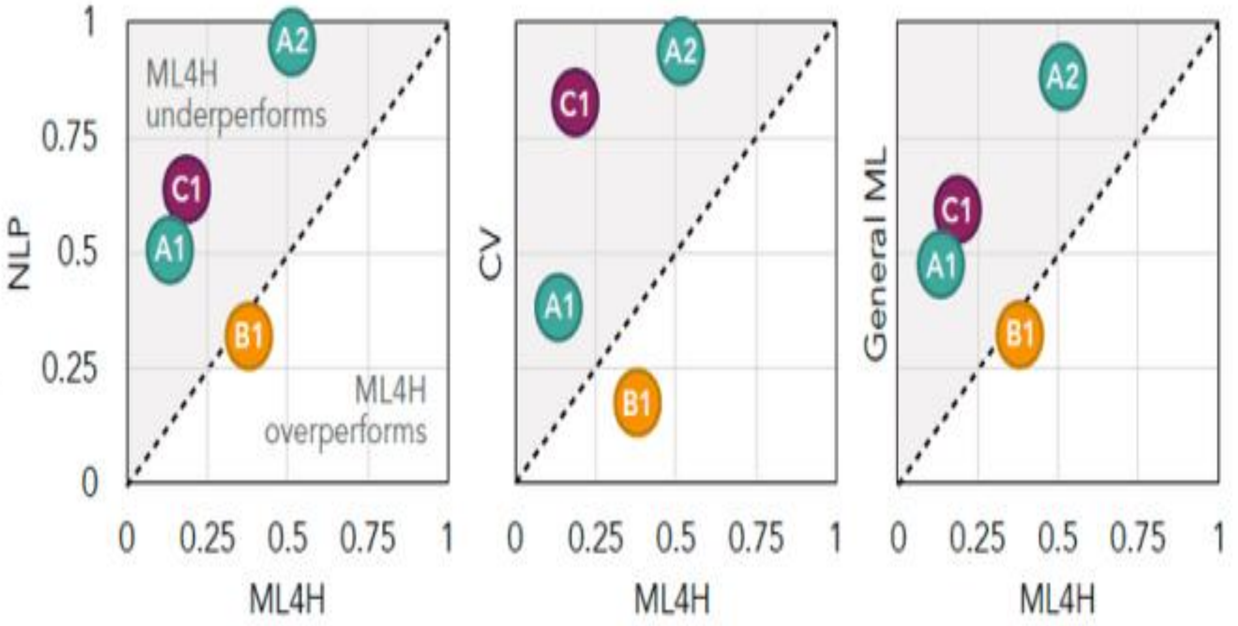


Figure 1: Fraction of papers satisfying certain conditions by ML field. See the Appendix (Section 5) for detailed descriptions of the underlying data collection procedure. Note that ML4H consistently lags other subfields of machine learning on all measures of reproducibility save inclusion of proper statistical variance.

The Machine Learning Reproducibility Checklist (Version 1.2, Mar.27 2019)

For all **models** and **algorithms** presented, check if you include:

- ❑ A clear description of the mathematical setting, algorithm, and/or model.
- ❑ An analysis of the complexity (time, space, sample size) of any algorithm.
- ❑ A link to a downloadable source code, with specification of all dependencies, including external libraries.

For any **theoretical claim**, check if you include:

- ❑ A statement of the result.
- ❑ A clear explanation of any assumptions.
- ❑ A complete proof of the claim.

For all **figures** and **tables** that present empirical results, check if you include:

- ❑ A complete description of the data collection process, including sample size.
- ❑ A link to a downloadable version of the dataset or simulation environment.
- ❑ An explanation of any data that were excluded, description of any pre-processing step.
- ❑ An explanation of how samples were allocated for training / validation / testing.
- ❑ The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results.
- ❑ The exact number of evaluation runs.
- ❑ A description of how experiments were run.
- ❑ A clear definition of the specific measure or statistics used to report results.
- ❑ Clearly defined error bars.
- ❑ A description of results with central tendency (e.g. mean) & variation (e.g. stddev).

Conclusion

- This project allowed us to get in touch with close to real life deep learning systems and all the challenges associated with them.
- Deep learning is a powerful tool but as researchers and engineers it’s important to evaluate the tradeoffs.

References

- 1.Baker, M. 1,500 scientists lift the lid on reproducibility. Nature News 533, 452 (2016).
- 2.Paull, L. et al. Duckietown: An open, inexpensive and flexible platform for autonomy education and research. in 2017 IEEE International Conference on Robotics and Automation (ICRA) 1497–1504 (IEEE, 2017). doi:10.1109/ICRA.2017.7989179.
- 3.Paull et al. - 2017 - Duckietown An open, inexpensive and flexible plat.pdf.
- 4.McDermott, M. B. A. et al. Reproducibility in Machine Learning for Health. arXiv:1907.01463 [cs, stat] (2019).
- 5.Pineau, J. ReproducibilityChecklist-v1.2. 1.
- 6.Pineau - ReproducibilityChecklist-v1.2.pdf.
- 7.AIdea. https://aidea-web.tw/topic/27dc9bbd-3ac5-458a-af41-abf23bd37e84?focus=intro.
- 8.Duckietown: a playful road to learning robotics and AI by Duckietown Foundation — Kickstarter. https://www.kickstarter.com/projects/163162211/duckietown-a-playful-road-to-learning-robotics-and.
- 9.End-to-End Deep Learning for Self-Driving Cars. https://devblogs.nvidia.com/deep-learning-self-driving-cars/.1
- 0.Intel® Movidius™ Neural Compute Stick | Intel® Software. https://software.intel.com/en-us/movidius-ncs.
- 11.NeurIPS 2019 Conference. https://neurips.cc/Conferences/2019/.
- 12.TensorFlow. TensorFlow https://www.tensorflow.org/.


ARTIFICIAL INTELLIGENCE
COLLABORATION PLATFORM
AIdea
Topics
Projects
Competition
Career
Login
Portal / icme / Embedded Deep Learning Object Detection Model Compression Competition for Traffic in Asian Countries
Car
Car
Car
Car
Car
Car
Car
Car
Embedded Deep Learning Object Detection Model Compression Competition for Traffic in Asian Countries
Begin
12/01
Closed
01/30

Thank you!