New Light Technologies, Matt, Kai

When it pours, it

By
Scott Armstrong
Faisal Kalema
Richard Ling
5/15/2020

Agenda

- Overview / Problem Statement
- Modeling Process
- Data Collection
- Data Cleaning
- Flood Detection
- Flood Classification
- Conclusion
- Next step

Overview

Floods:

- Pictures spread quickly
- Damage home, infrastructures and people's lives
- Corr(Depth, Severity) = 1 !

Problem Statement:

Use images to detect flooding and its severity

Modeling Process

Two Models:

- Binary Classification with MLP
  - *Is the image flooded or not?*
- Object detection with YOLO v3 and Pytorch
  - *Level of water depth*

Data Collection

Web-scraping to get flood images:

NY Times API: 300 images

All articles with "Flooding" tag, 2004-2020

Gettyimages: 400 images

Search : ‘ Hurricane Katrina ‘

Data Cleaning

Classification Model:

- Keep all images, even non-floods. Use to classify flood vs non-flood.

Object detection:

- Side view
- Show water level

Flood Detection: Criteria

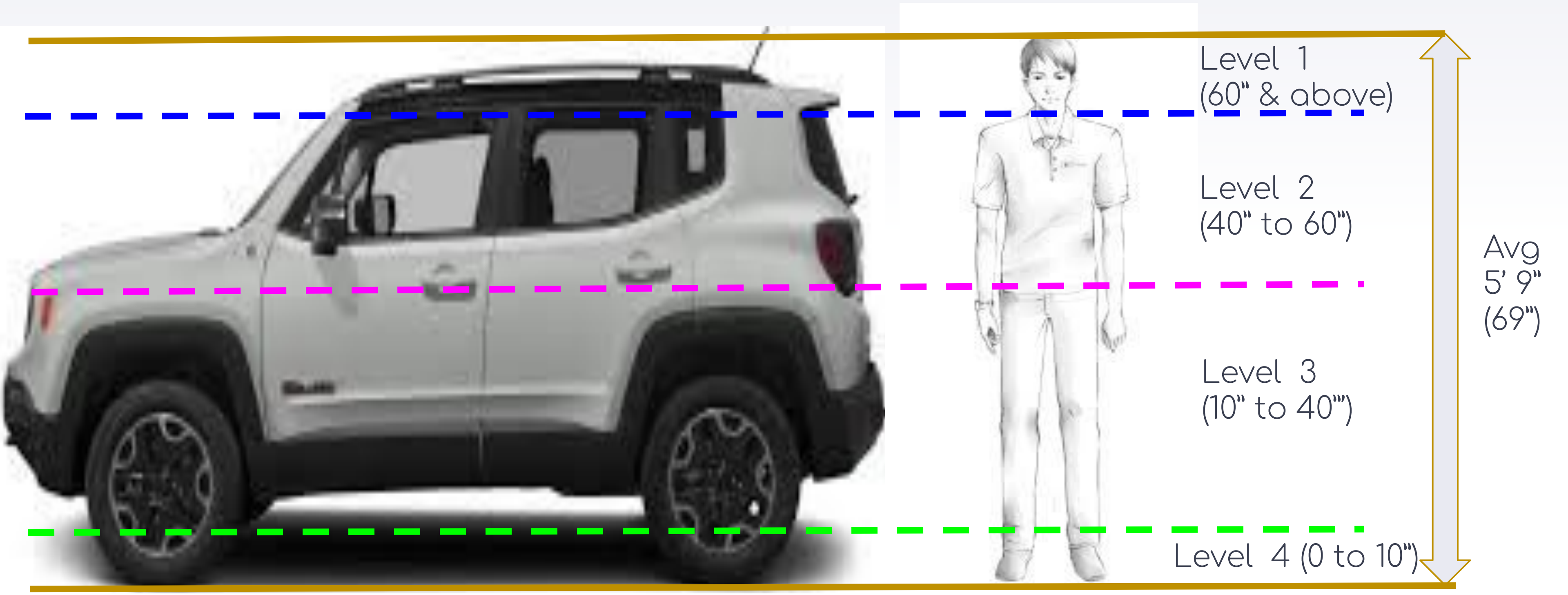
Four Levels using labelImg:

Level 1 to Level 4, from deep to shallow.

Assumptions:

- People are around average height.
- Ignore babies and children.
- Car heights are similar to average height of a person.
- For buses, level 1 means more than ⅔ of the bus is underwater.

Flood Detection: Criteria



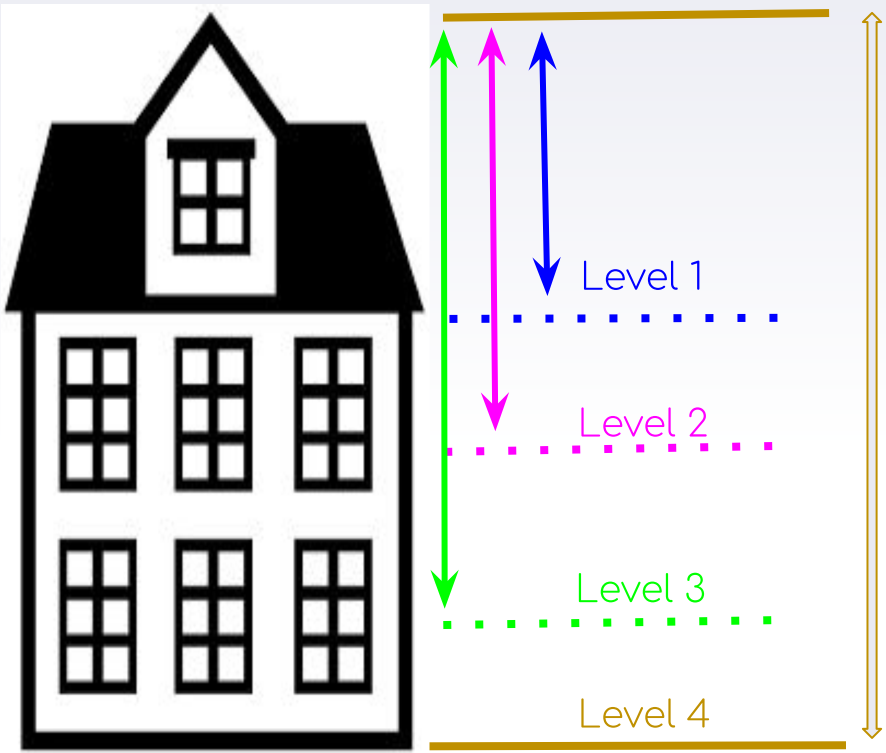
Flood Detection: Label Ex.

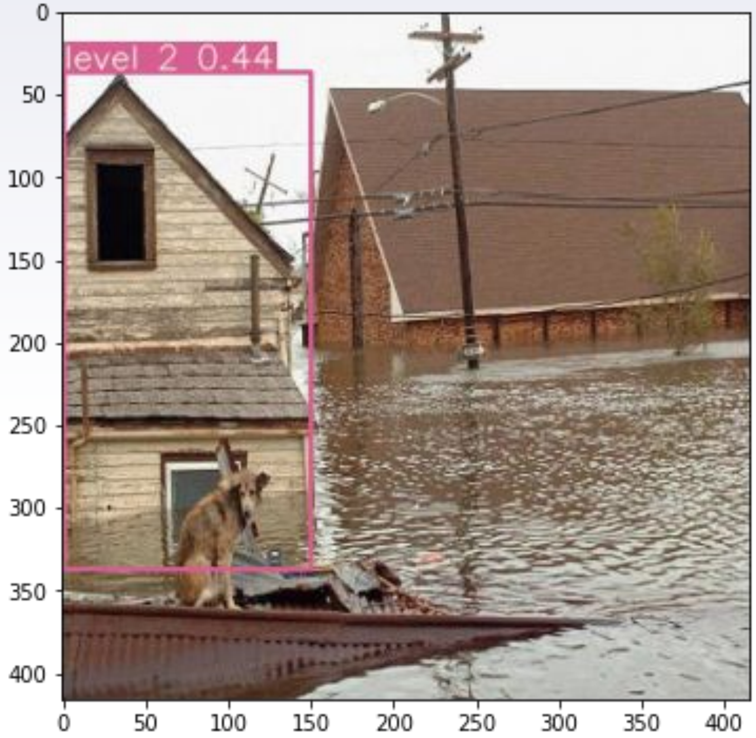
Level 1, bus example. This bus height is ~ 100 inches

Flood Detection: Label Ex.

Human Label:

Level 2 ~ at waist level


Level 1
Level 2
Level 3
Level 4


level 2 0.44

Flood Detection: RoboFlow

#roboFlow created by Joseph Nelson
Model from RoboFlow

Total label images fed into RoboFlow: 302 images

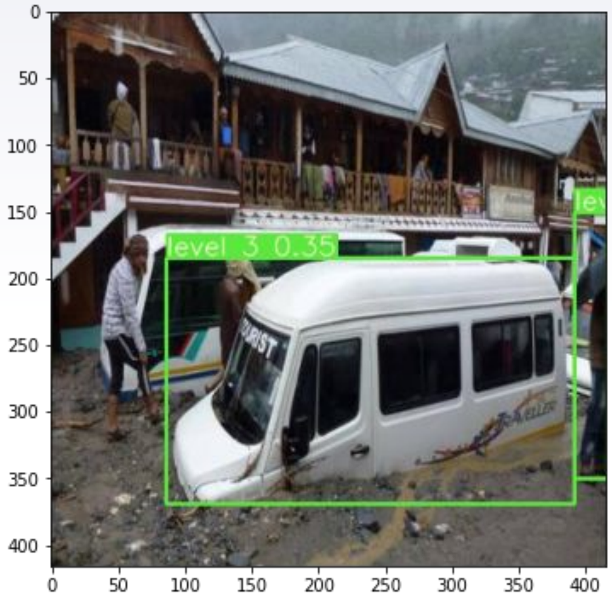
Augmentation:

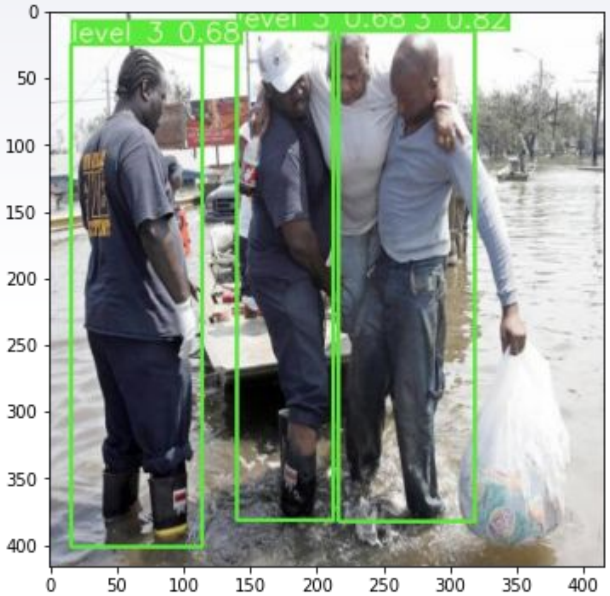
Darken and Brighten 40%.

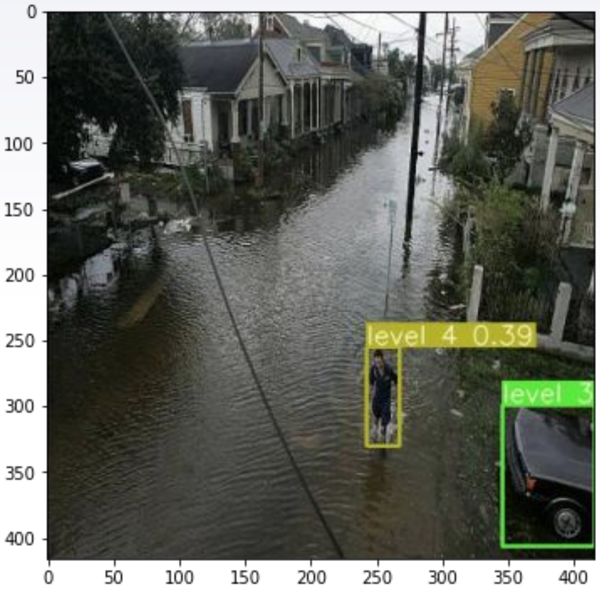
For model to train on different exposures and more training data.
i.e. not just dark or bright.

Total Images for modeling: 906 images (Each image x 3)

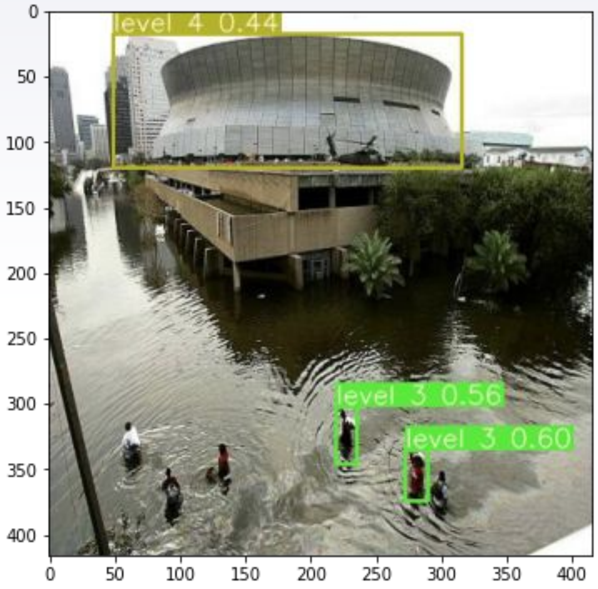
Flood Detection: Results 1

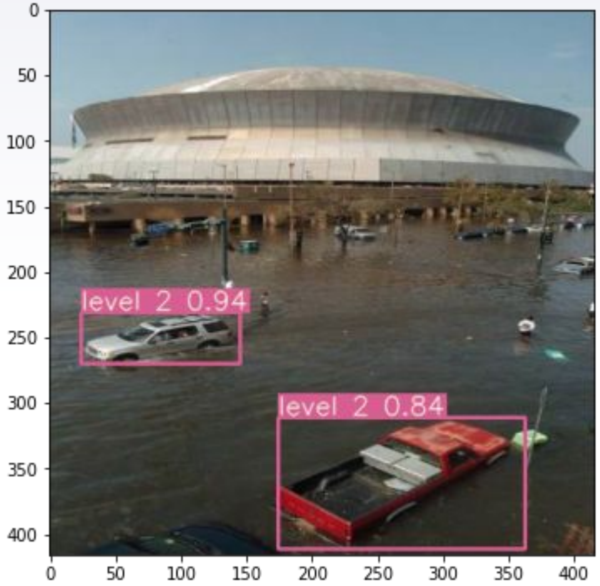


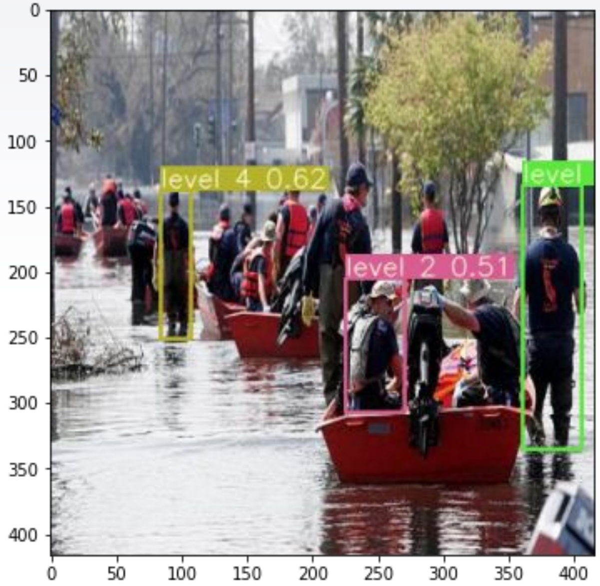




Flood Detection: Results 2







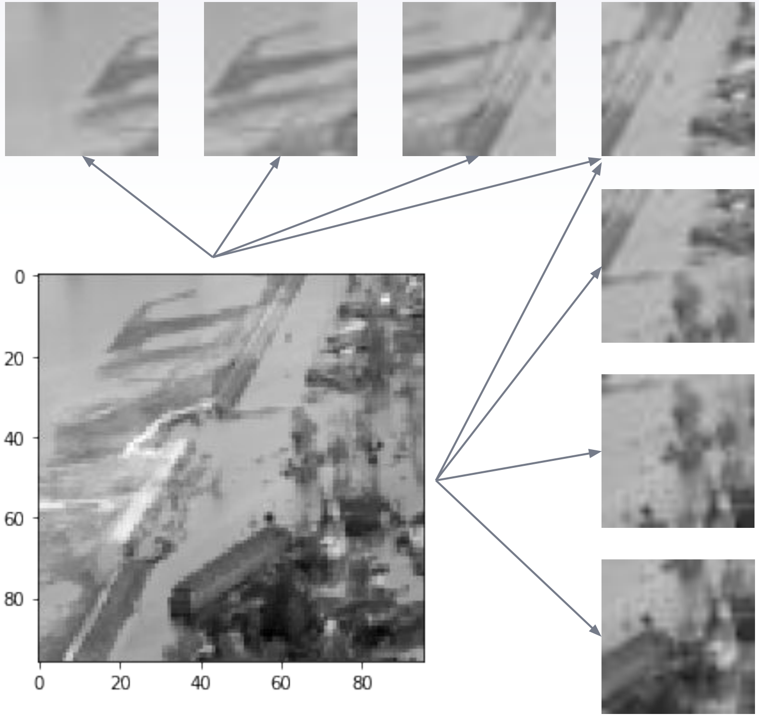
Overall: 60 / 88 Total labels

About 68%

Flood Classification: Augmentation

32x32 grayscale result images

original image

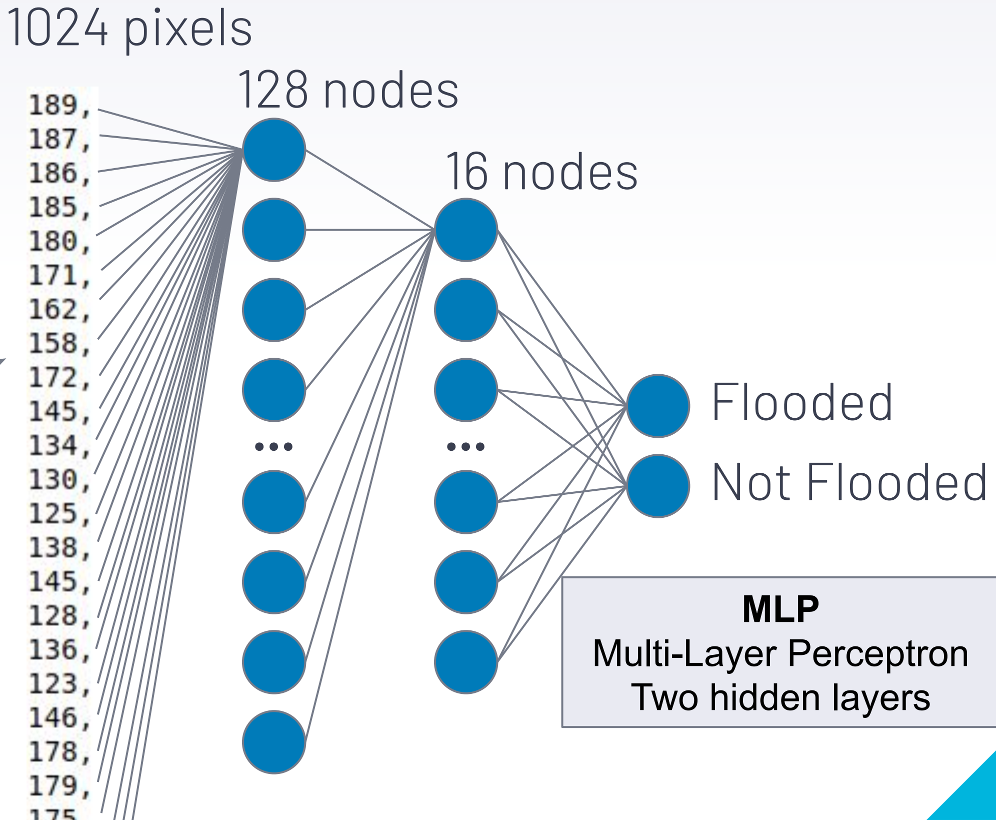


Flood Classification: Modeling

32x32

pixel array

```
array([[189, 187, 186, ..., 109, 114, 168],
       [184, 185, 186, ..., 117, 101, 129],
       [181, 183, 186, ..., 126, 107, 122],
       ...,
       [123, 112, 125, ..., 116, 125, 127],
       [116, 115, 147, ..., 180, 162, 140],
       [113, 122, 167, ..., 169, 162, 158]])
```



MLP
Multi-Layer Perceptron
Two hidden layers

Flood Classification: Testing

Base accuracy ≈ 55-60%

Validation (sub-images)

accuracy	sensitivity	specificity	precision	ROC AUC
0.824	0.805	0.852	0.89	0.829

Test (whole images)

accuracy	sensitivity	specificity	precision	ROC AUC
0.64	0.612	0.675	0.698	0.644

To classify a whole image: split image into all sub-images and average the result

Abstracted away using a custom estimator class FloodImageClassifier

Flood Classification: False Positives

Flood Classification: False Negatives

https://flood-image-classifier.herokuapp.com/

Flood Classification: Correctly Classified

Conclusion

Object Detection:

- With proper labeling, it can detect objects in the images and level of the flood.

Binary Classification:

- A simple neural network can be reliable at distinguishing between flooded and non-flooded images

Going Forward....

Object Detection:

- Build our own object detection model with more tunings parameters.
- Add more images of different objects. People, cars, houses, trees, street signs, etc.
- Add in perspective measurements to measure the objects.

Flood Classification:

- More images
- More processor time for wider and deeper models

Questions/Comments