



Torc Robotics: Rain Detection Solution “RainSense AI”

Alyssa Lowe, Darius Holland, Ethan Weaver, Jason Andrade,
Jonathan Samuel, Minho Cho

Table of Contents

- Project Description
 - Context and Background
 - Target Users, Project Goals, Main Functionality, Unique Features
- System Build Process
 - IDEs and Tools
 - Technical Details and Design Choices
- Demonstration
- Lessons Learned & Reflection
 - Reflection and Retrospective
 - Difficulties
 - Room for Improvement
 - Future Possibilities

Project Description

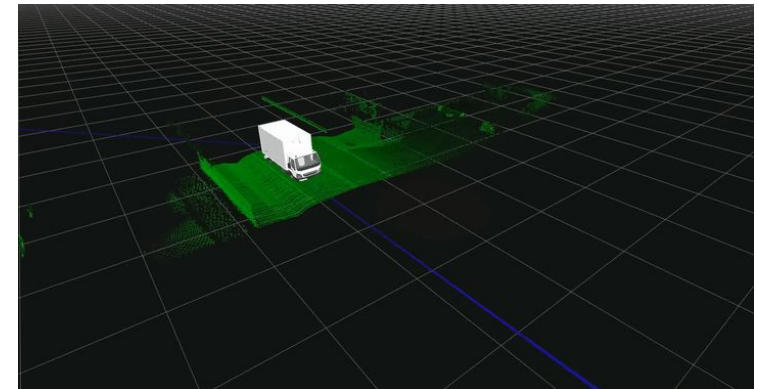
TORC

- Project Client: Torc Robotics
- Specializes in self-driving trucks
- Goal: Freight trucks that travel safely in changing weather conditions
- Uses LiDAR, camera, and radar sensors
- Vehicles use ROS (Robot Operating System)



LiDAR (Light Detection and Ranging)

- Laser-based, remote sensing method that utilizes light to measure ranges between objects. Ex. Vehicle, street sign, rain drops
- Generates 3D image of a vehicle's surroundings



Target Users

Personas

- Data Scientists at Torc
- Machine Learning Engineers at Torc

The logo for TORC, consisting of the letters 'TORC' in a bold, red, sans-serif font.

Stakeholders

- Weekly meetings to review progress and blockers
- Torc contextualized the problem with use cases & provided real world examples
- Used these meetings to prioritize tasks and ensure product success

Project Goals

Initial User Story

*As a **Data Scientist** at Torc,*

*I want to **classify** the current state of the vehicle as **raining** or **not raining***

*So that I can give the autonomous vehicle **useful information**.*

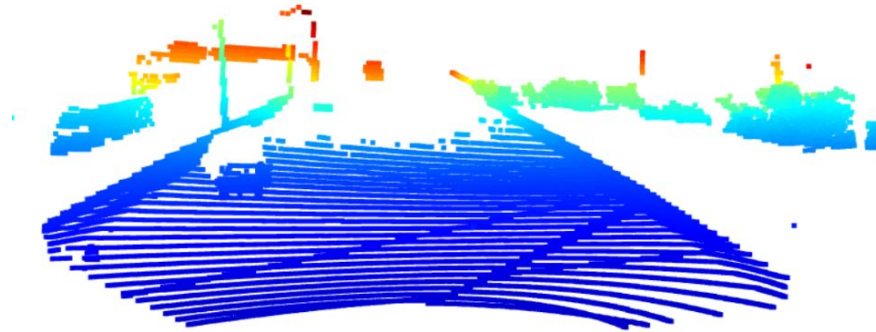
Final User Story

*As a **Data Scientist** at Torc,*

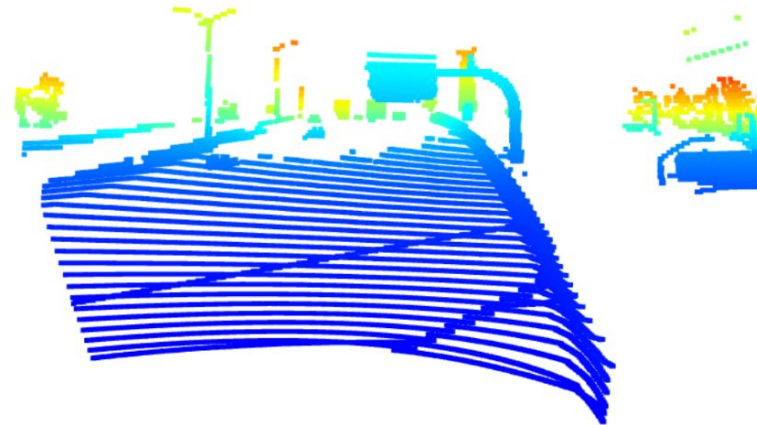
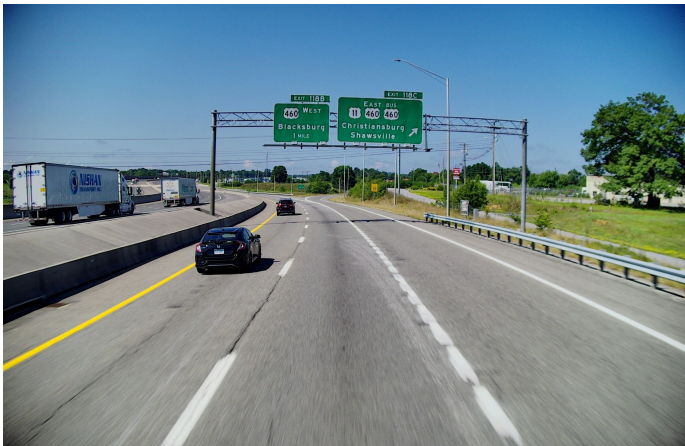
*I want to deploy an application that uses **an image** data classifier and an algorithm for **LiDAR** spatial data to determine the current state of the vehicle as **raining** or **not raining***

*So that I can give the autonomous vehicle **useful information**.*

Data Visualization



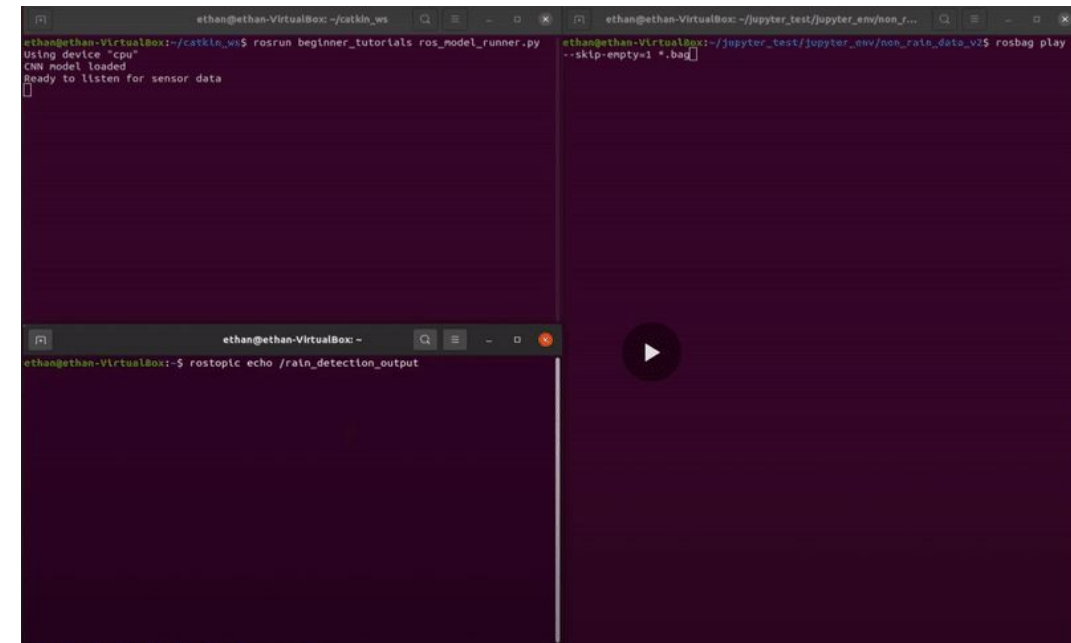
With Rain



No Rain

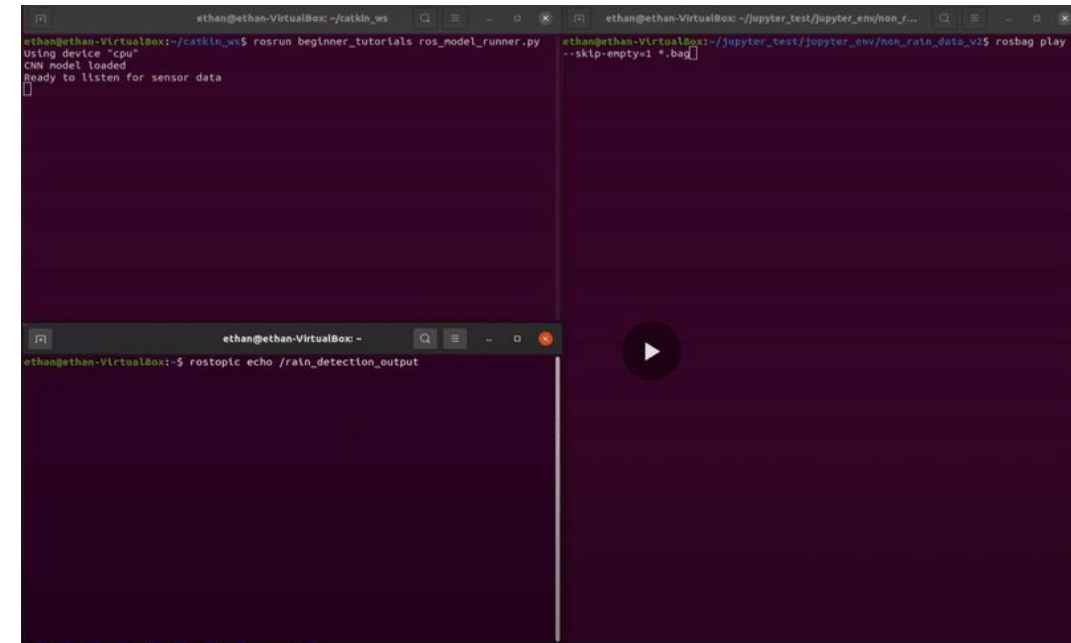
Main Functionality for “RainSense AI”

- Rain predictor utilizing images & LiDAR modalities from real world driving sessions for training, validation, & testing
- Deployed solution ingests ROS data captured directly by the vehicle’s sensors & determines *rain* or *no rain* in real-time
 - Two Processes: CNN image classifier, LiDAR anomaly detection
 - CNN image classifier for images produced by onboard camera
 - Anomaly detection for LiDAR point cloud data produced from 2 onboard LiDAR sensors
 - Custom weighting algorithm



Main Functionality for “RainSense AI”

- Rain predictor utilizing **Images & LiDAR**
- Real world driving sessions for training, validation, & testing
- Solution ingests ROS data captured by the vehicle’s sensors & determines *rain* or *no rain* in real-time.
 - Two Processes: CNN image classifier, LiDAR anomaly detection
 - Custom weighting algorithm



The screenshot displays two terminal windows from a virtual machine. The left window shows the execution of a ROS model runner, with output indicating the model is loaded and ready for sensor data. The right window shows a ROS bag file being played back. A third terminal window at the bottom shows a rostopic echo command monitoring the /rain_detection_output topic.

```

ethan@ethan-VirtualBox:~/catkin_ws$ rosrn beginner_tutorials ros_model_runner.py
Using device "cpu"
CNN model loaded
ready to listen for sensor data

ethan@ethan-VirtualBox:~/jupyter_test/jupyter_env/roa_rain_data_v2$ rosbag play
--skip-empty=1 *.bag

ethan@ethan-VirtualBox:~$ rostopic echo /rain_detection_output

```

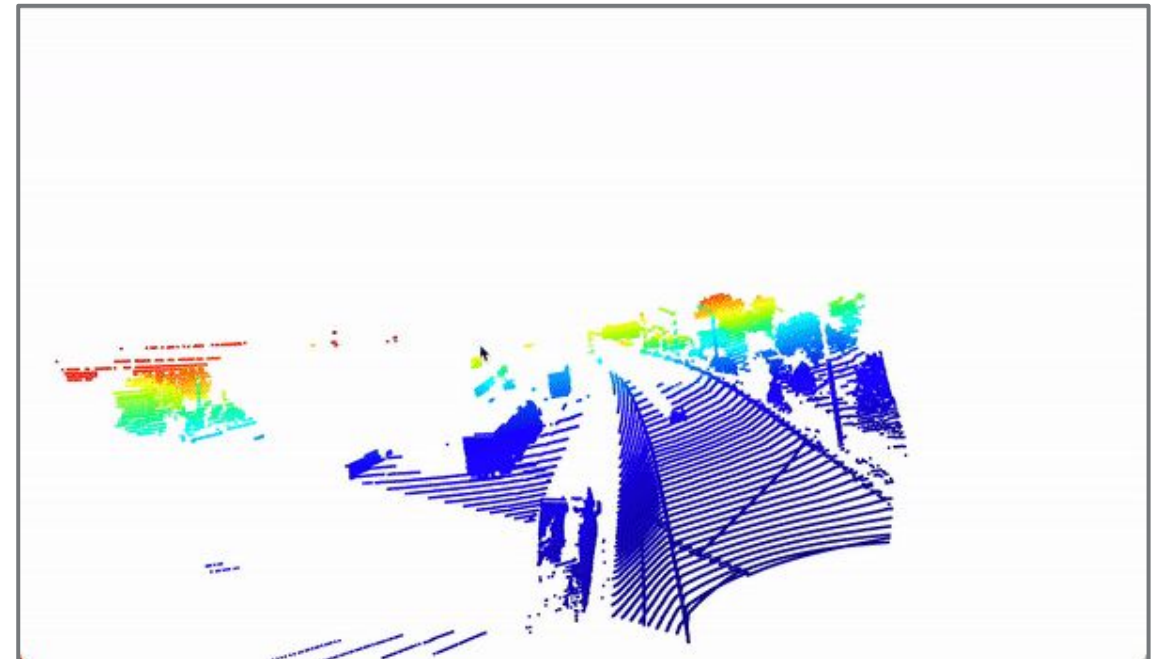
CNN Image Classifier

- Image classifier utilizing a Convolutional Neural Network (CNN) implemented with PyTorch



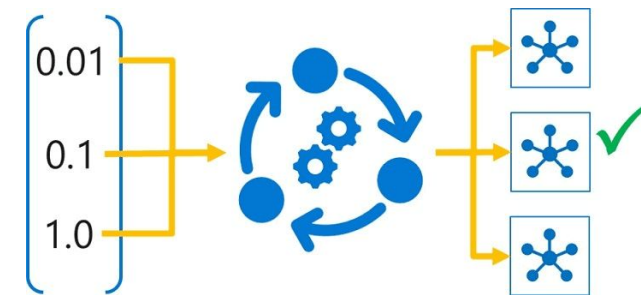
LiDAR Anomaly Detection

- Detection algorithm processing LiDAR spatial data to identify anomalies in the scans that are indicative of rain conditions using recent studies



Unique Features

- **GPU Acceleration:** Leverage PyTorch's GPU support for faster, efficient computation
- **ROS Environment Integration:** Solution operates within ROS environment, reading & writing from/to ROS nodes
- **Grid Search Hyperparameter Tuning:** Made design decision based on accuracy and other performance based statistics

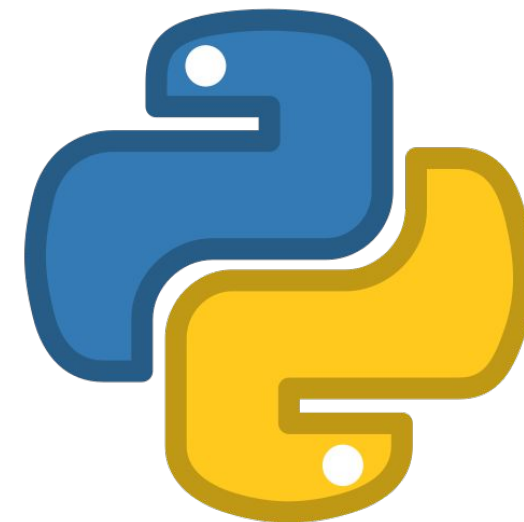


System Build Process

System Build Process

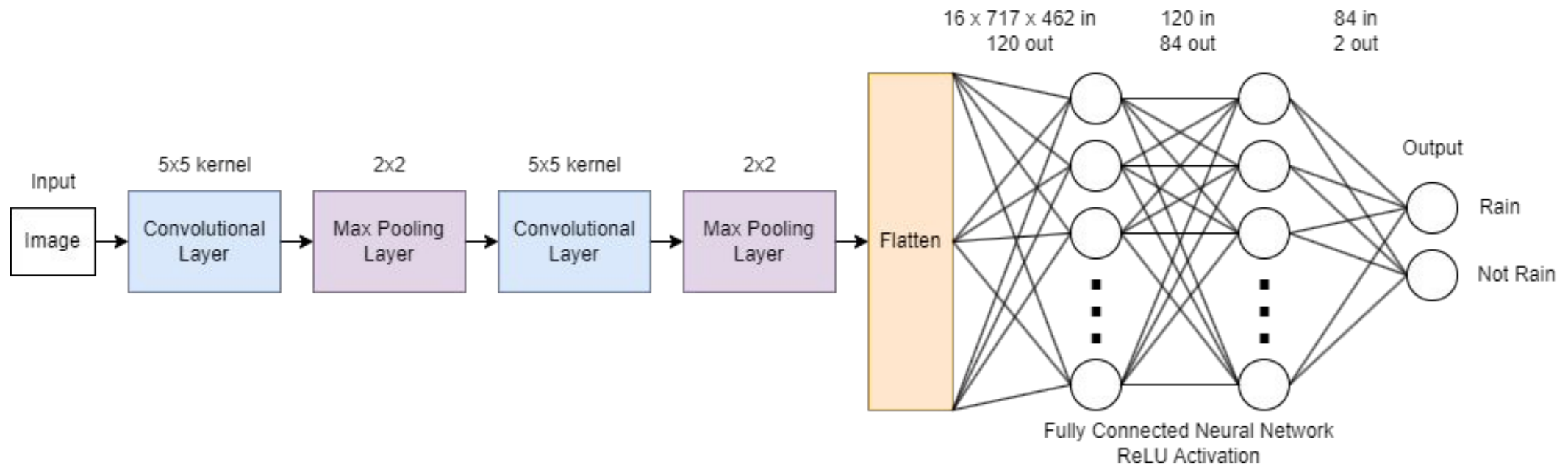
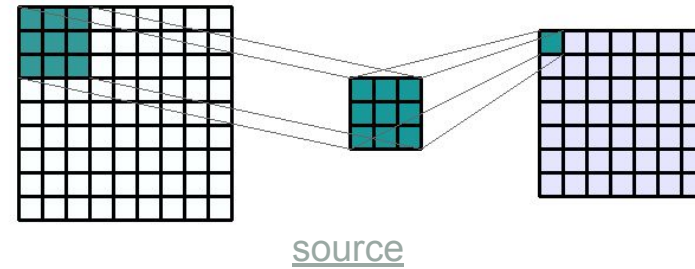
Tools

- ROS environment
- Python
- Google Colab
- PyTorch ML framework



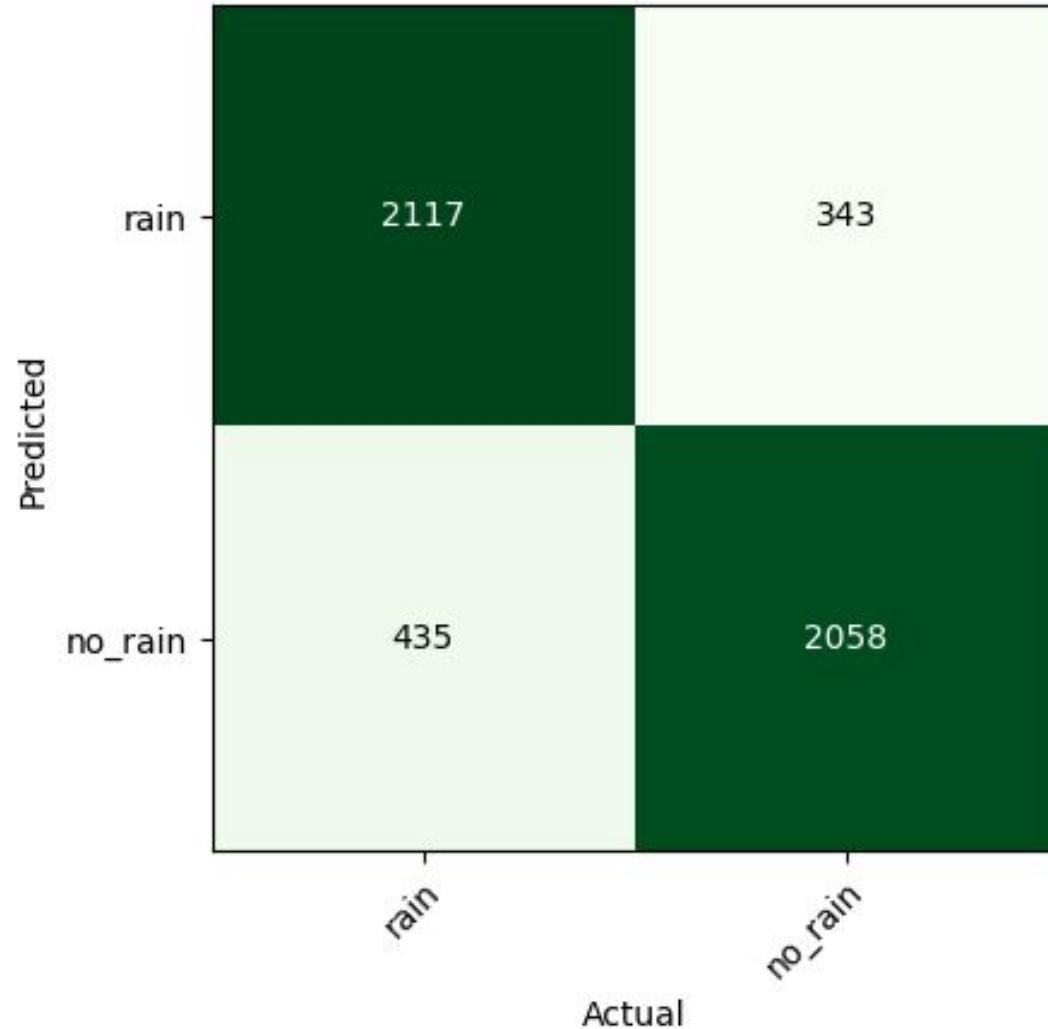
CNN Classifier

- Convolutional Layer
 - Breaks image into sections (5 px \times 5 px)
- Max Pooling Layer
 - Reduces dimensionality by taking only the max value of each section (2 px \times 2 px)
- Flattened & fed into 3 fully connected layer outputting the class value



CNN Classifier Results

CNN Model Confusion Matrix - Test Set

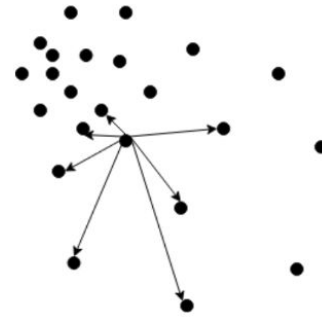


Accuracy	84%
Sensitivity	83%
Specificity	86%
Precision	86%

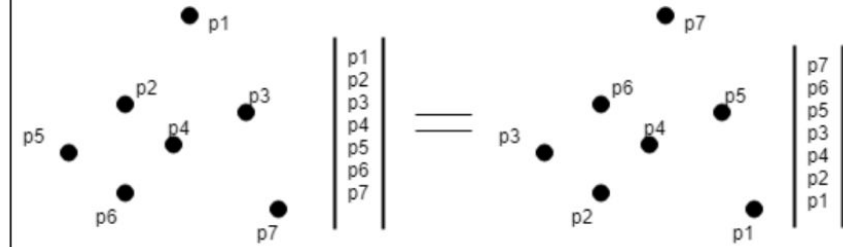
LiDAR



(a) Irregular. Sparse and dense regions



(b) Unstructured. No grid, each point is independent and distance between neighboring points is not fixed



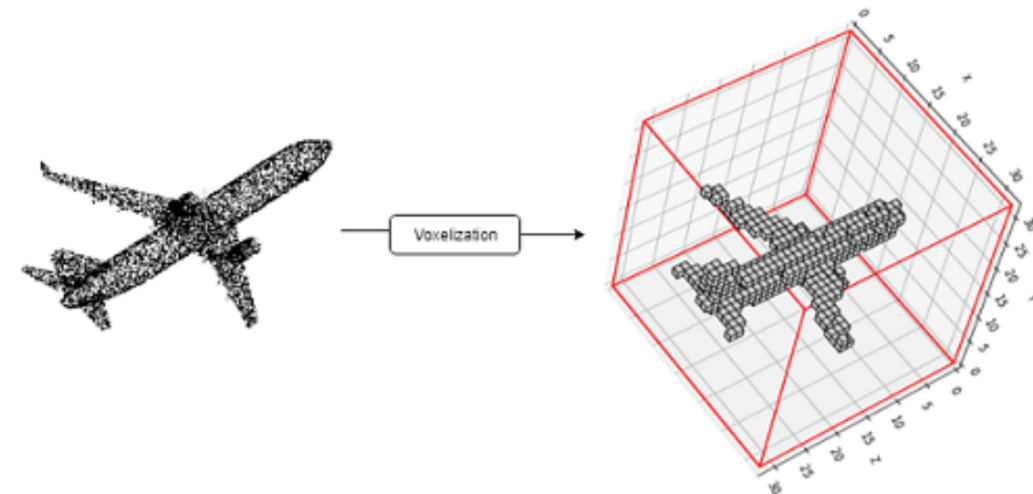
(c) Unordered. As a set, point cloud is invariant to permutation

Voxelization

Converts continuous spatial data into a regular grid of volumetric pixels

Common Approaches

- Binary Voxelization
- Mean/Center Voxelization
- Density-based Voxelization



LiDAR Anomaly Detection

Spatial indices for measuring three-dimensional patterns in a voxel-based space

Jjumba, A. et al (2016) [source](#)

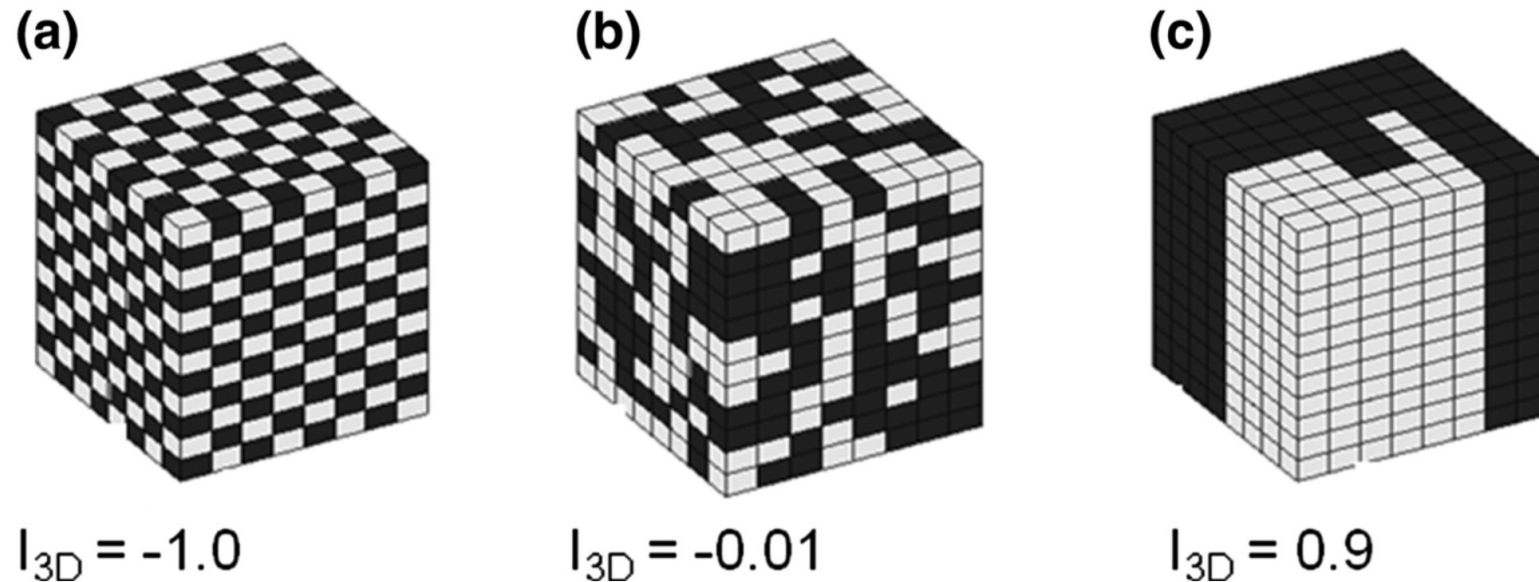


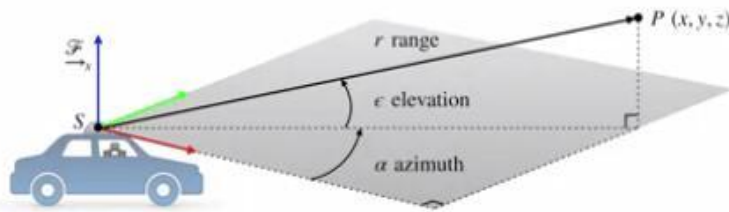
Fig. 1 Moran's I values for the different 3D voxel state configurations depicting **a** negative spatial autocorrelation for a perfectly uniform pattern, **b** no spatial autocorrelation for a random pattern and **c** a strong positive correlation for a clustered pattern

LiDAR Anomaly Detection

Detecting the Anomalies in LiDAR Pointcloud

TuSimple (2023) [source](#)

3D LIDAR sensors report range, azimuth angle and elevation angle (+ return intensity)



$P = \{p_i = (r_i, \theta_i, \phi_i, \gamma_i) | i = 1, 2, \dots, N\}$
 where r_i : distance, θ_i : azimuth, ϕ_i :
 elevation, γ_i : intensity

$$I = \begin{cases} \frac{N}{W} \frac{(\sum_{i=1}^N \sum_{j=1}^N w_{ij} (r_i - r)(r_j - r))}{\sum_{i=1}^N (r_i - r)^2} & \text{for } N > 1 \\ -1 & \text{for } N = 1 \end{cases}$$

where r = average distance,
 w_{ij} predefined weights

Spatial Autocorrelation

$$K_\gamma = \exp\left(k \cdot \frac{\max(0, \gamma_{ref} - \bar{\gamma})}{\gamma_{ref}}\right)$$

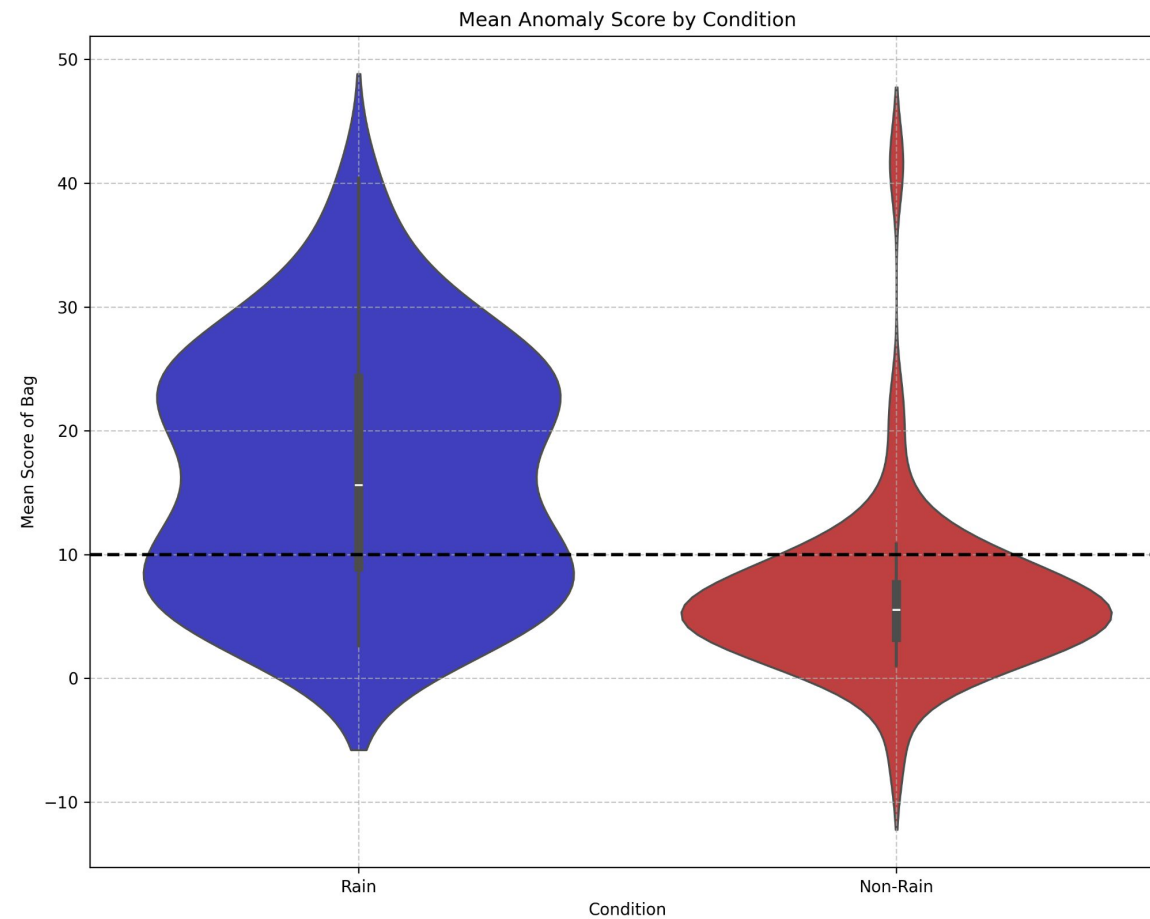
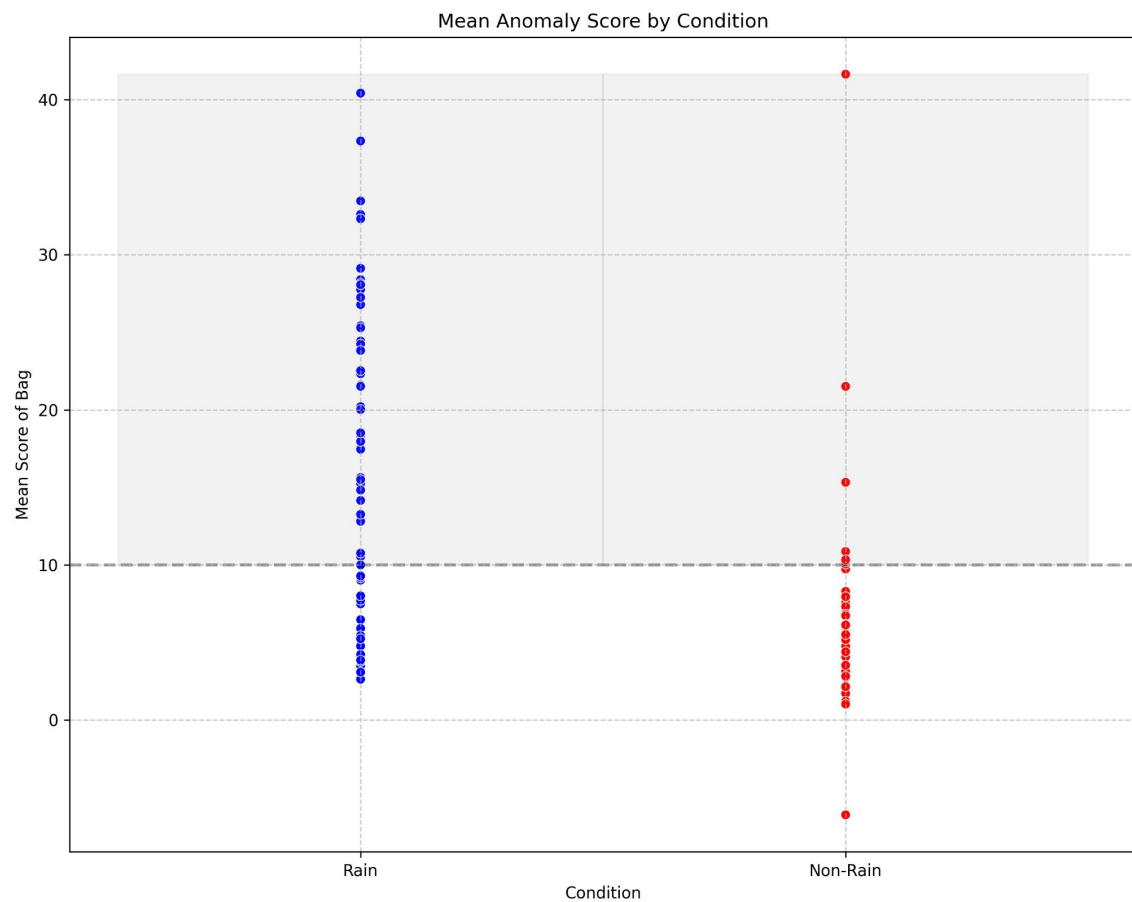
where γ_{ref} : 'Normal' Intensity,
 $\bar{\gamma}$: Avg. Intensity

Intensity Measure

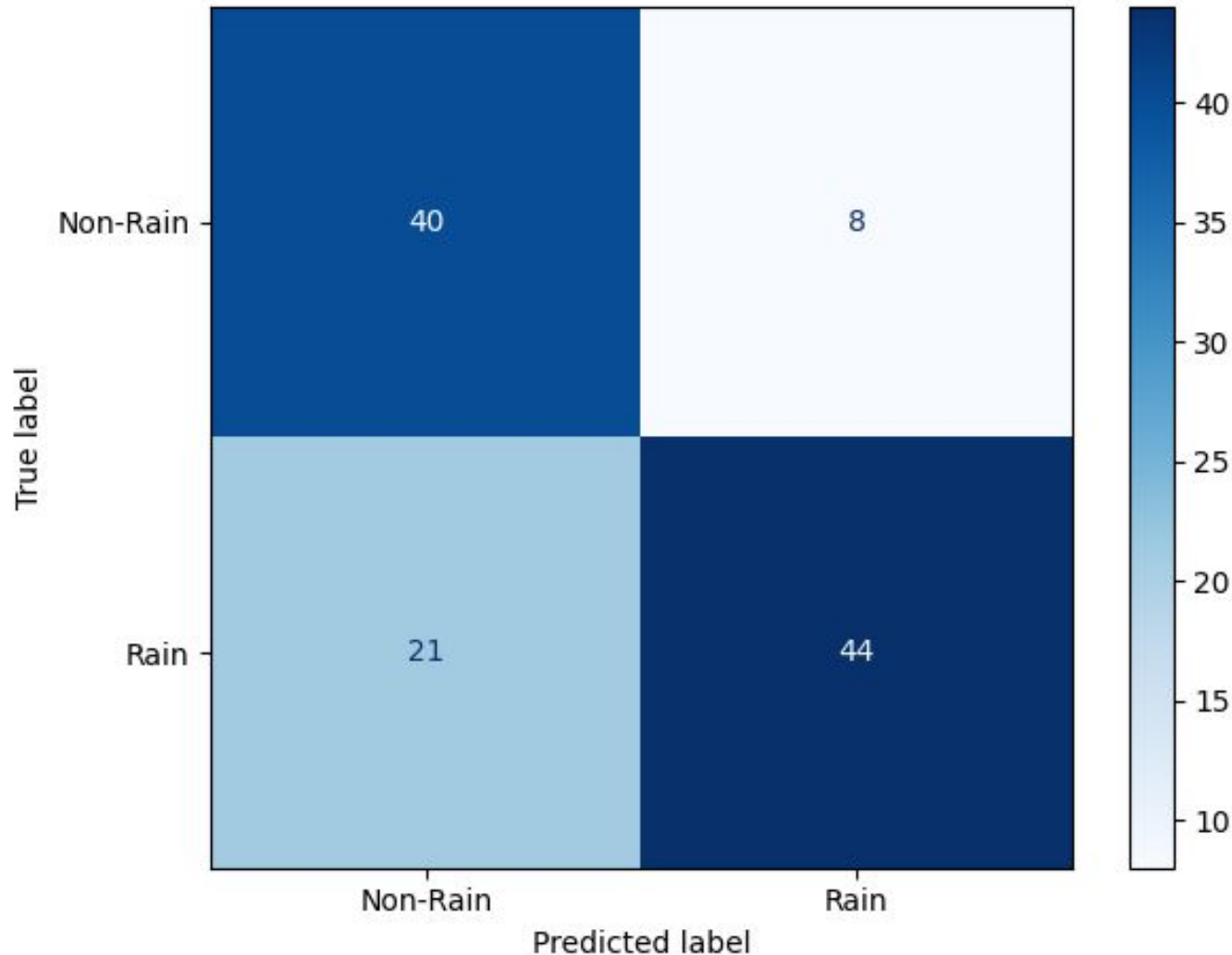
$$S = \frac{1}{VH} \sum_i^V \sum_j^H K_{\gamma,ij} \cdot I_{ij}$$

Anomaly Score

LiDAR Anomaly Detection Results



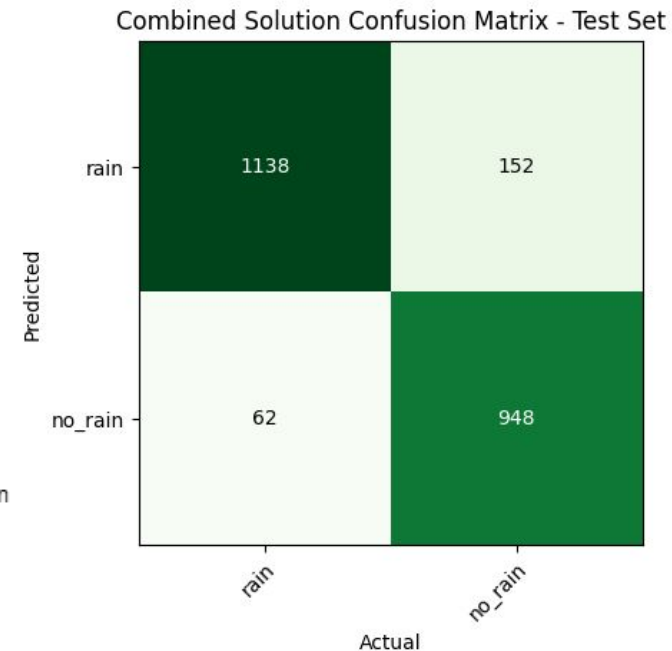
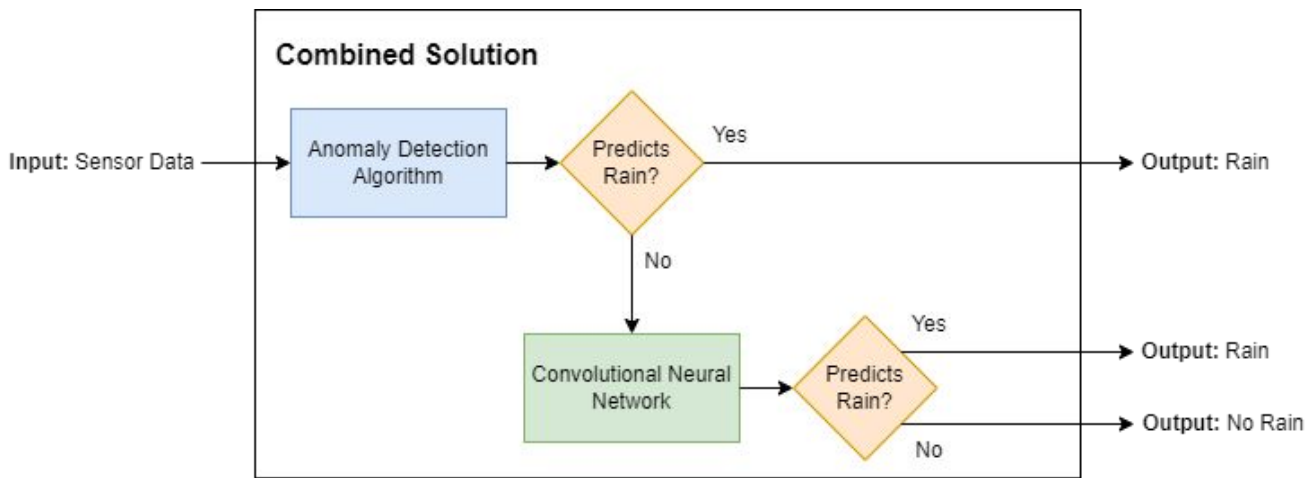
LiDAR Anomaly Detection Results



Accuracy	74%
Sensitivity	67%
Specificity	83%
Precision	84%

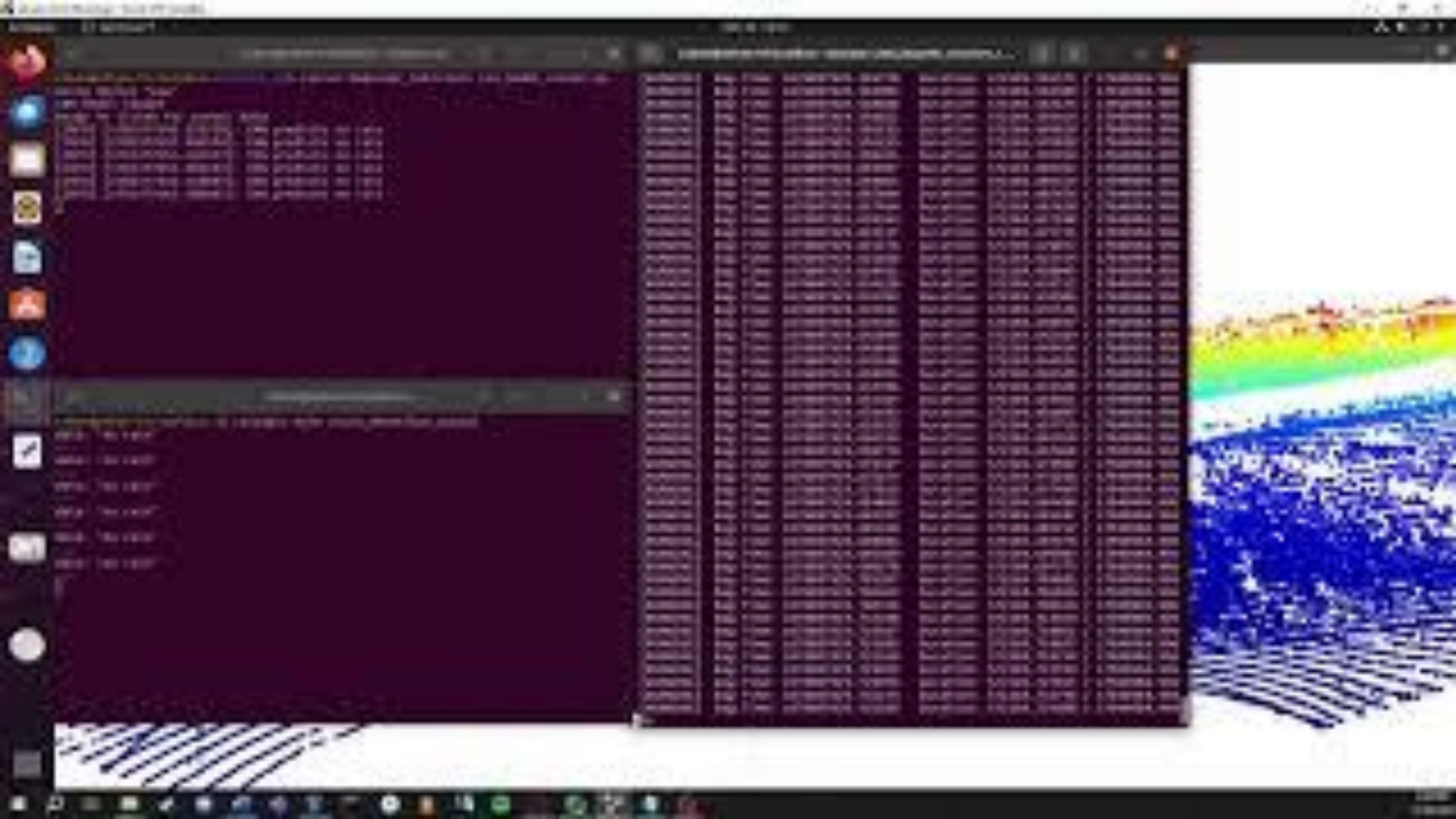
Final Product: Ensemble Learning

- Benefit from utilizing both modalities, image & LiDAR
- Anomaly detection produces few false positives but higher false negative
- Image prediction produces vice versa
- Use both weak learners in a voting system



Accuracy	91%
Sensitivity	95%
Specificity	86%
Precision	88%

Demonstration



Lessons Learned & Reflections

Reflection and Retrospective

- Future direction for model specific improvements “Down the Road”:
 - Increase data variety, potential performance improvements through data augmentation, incorporating more extensive external datasets.
 - Overall: MORE DATA
- Project Accomplishments:
 - Minimal viable accuracy for image classification of over 80%.
 - Anomaly detection of about 75% accuracy.
 - Combined solution accuracy of over 90%

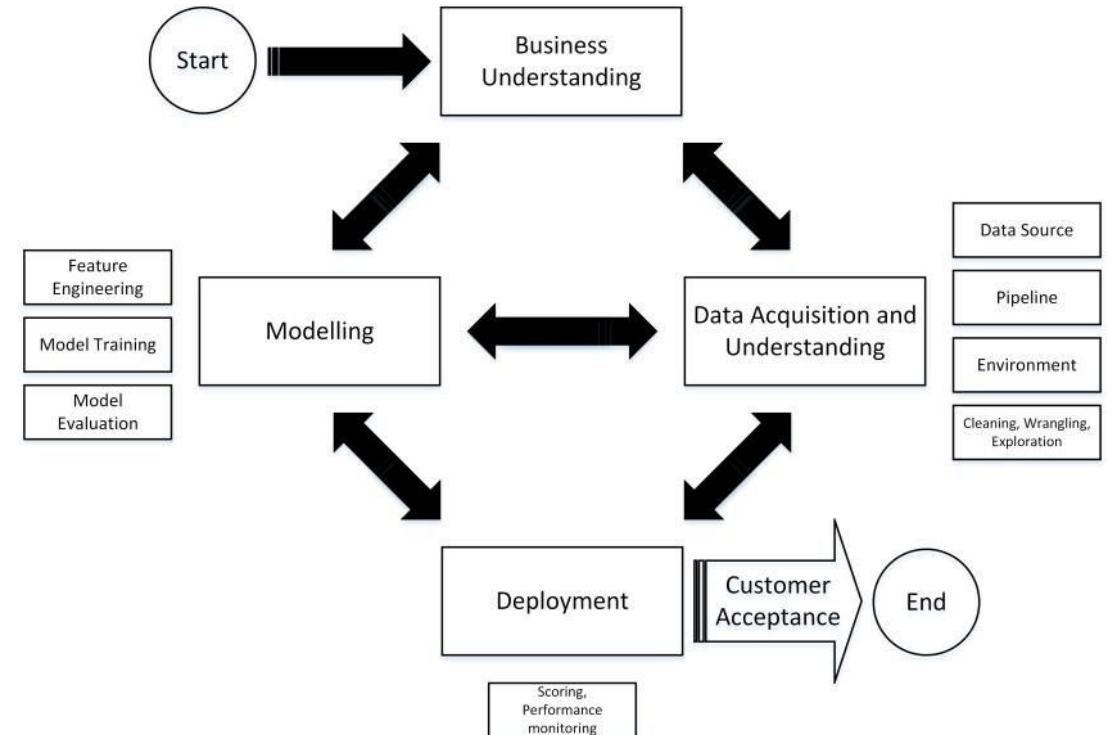
Technical Difficulties

- Image & LiDAR Data
 - Delays granting access to TORC data
 - Data may not be robust enough to test on different situations
 - Images & LiDAR data of *rain* & *no-rain* situations are extremely similar
 - Data is large leading to significant computation, training and forecast time
- CNN Classifier
 - Large learning curve
- LiDAR Anomaly Detector
 - Research area in its infancy
 - Large learning curve
 - Challenging programming research findings to functional programs
 - Tedious transitioning from NumPy (CPU) to PyTorch (GPU) implementation

Project Management Related Difficulties

Misalignment between ML project & traditional agile approach for Software Development Life Cycle (SDLC)

- Course project requirements structured for web development
- Continuous communication with professor diverging from SDLC to data science lifecycle



Room for Improvement

Challenges and Resolutions

Data Variety and Quality

- **Challenge:** Limited variety in initial datasets
- **Resolution:** Collaborated with Torc for diverse data from more varied driving scenarios, including increased nighttime scenarios
- **Next Steps:** Continue exploring ways to enhance dataset diversity

Overfitting in Initial Model

- **Challenge:** Sprint 1 model showed high accuracy but overfitted to similarities between images
- **Resolution:** Developed a new model using a larger dataset, incorporating validation sets, and optimizing the way data was split into train/validation/test sets
- **Learning:** Emphasize real rain detection over image similarity

Future Possibilities

Image classifier & algorithm false positives

- Not Raining (Gray Sky, Wet Ground)
 - But both the LiDAR algorithm and the CNN predict rain

Possible solution to enhance capabilities
in future real-world scenarios:

- Add other sensors to this solution
 - Infrared (IR) sensor commonly
used for automatic windshield wipers



Thank You

Questions?