IDOT Sprinternship 2023

Hasti Mehta, Samuel Effendy, Vira Kasprova, Srijita Banerjee, Rebekah Cheung, Habeebah Abdulbaseer

BREAK THROUGH TECH

UIC UNIVERSITY OF ILLINOIS CHICAGO

College of Engineering

Illinois Department of Transportation

Introduction

- Hasti: Computer Science + Nature walks
- Samuel: Computer Science + likes to play the ukulele
- Rebekah: Computer Science + reading
- Vira: Data Science + watching Netflix
- Srijita: Computer Science + mostly painting and singing
- Habeebah: Information Decision Science + playing volleyball or reading

Break Through Tech Chicago and IDOT

What is a Sprinternship?
A Sprinternship is a 3-week internship for women and nonbinary students at partner companies to help them gain exposure and experience in the tech world.

What is Break Through Tech and what do they do?
Break Through Tech works at the intersection of academia and industry to propel women and non-binary students into computing degrees and tech careers.

BREAK THROUGH TECH CHI

Outline

1. Problem Overview
2. Term Definitions
3. Timeline
4. Image Tagging
5. Computer Vision Models
6. Functions
7. Legacy Project Statistics
8. What’s Left To Do
9. Applications + Recommendations
10. Lessons Learned
11. Appreciation
12. Q&A

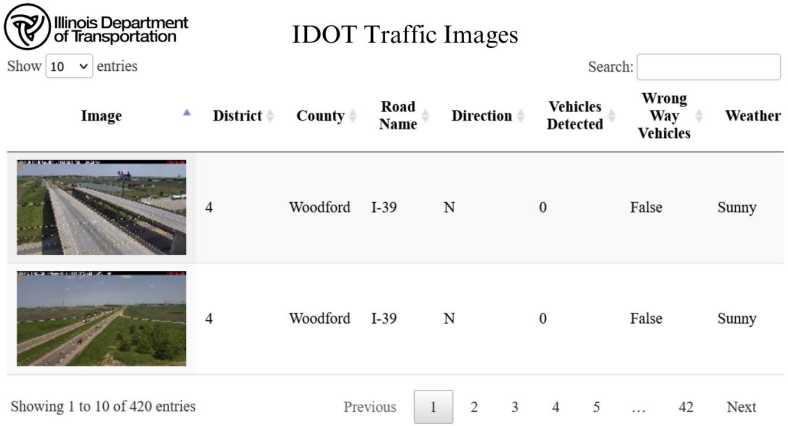
Challenge Problem Overview

Problem: Using a computer vision model to recognize and blur houses from view on roads monitored by IDOT cameras in real-time, and to create a website that displays the dataset as shown below.

Additionally, we could also detect other objects like weather and emergency vehicles.

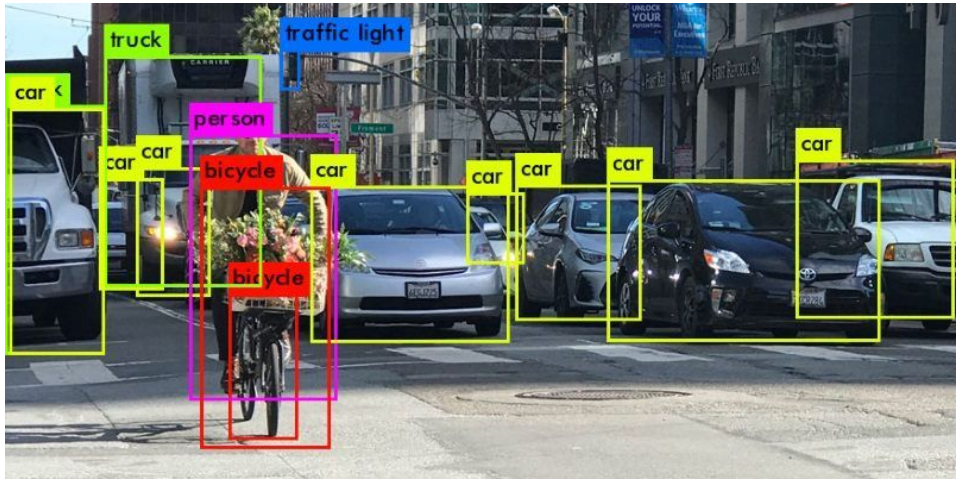
Tools used:

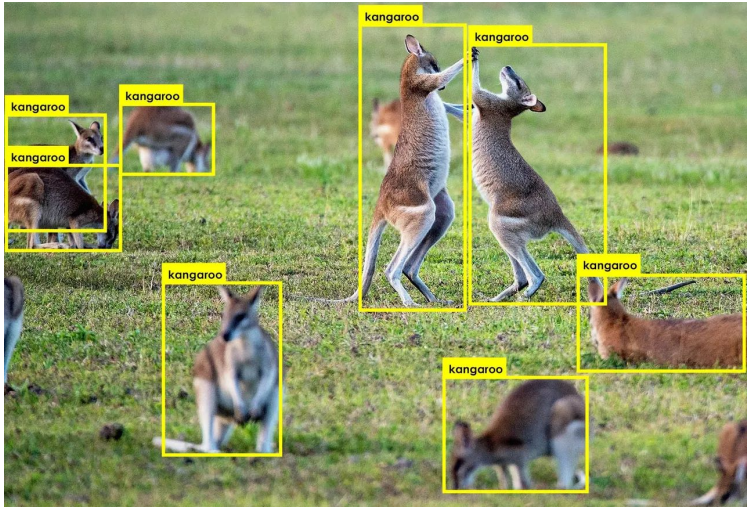
- Programming Languages/Frameworks: Django, Python
- Platforms: Jupyterlab, Roboflow
- Libraries: Pytorch, OpenCV
- Model: YOLOv5, Detectron 2



Machine Vision

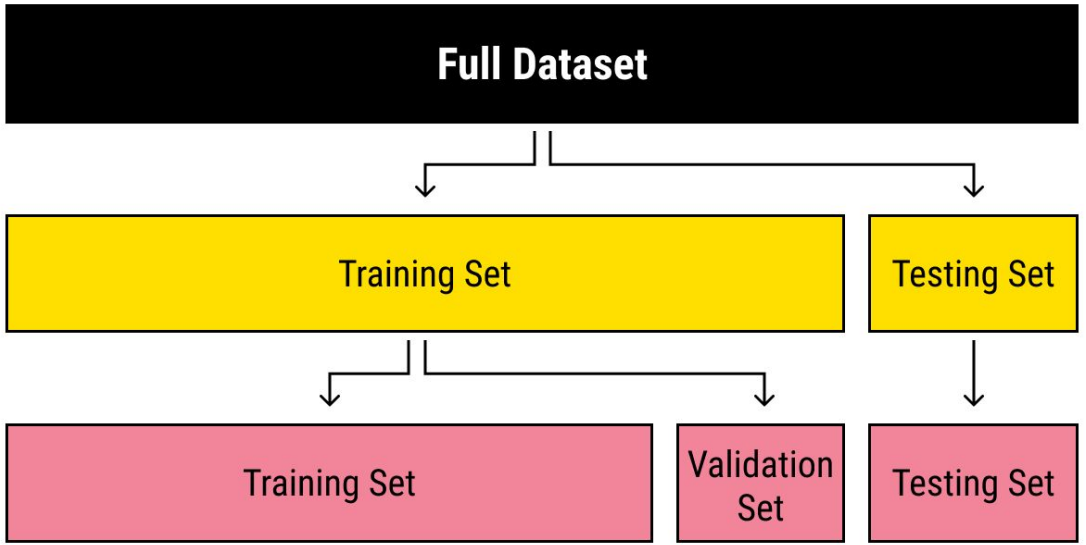
Machine vision is the technology and methods used to provide imaging-based automatic inspection and analysis for such applications like automatic inspection.





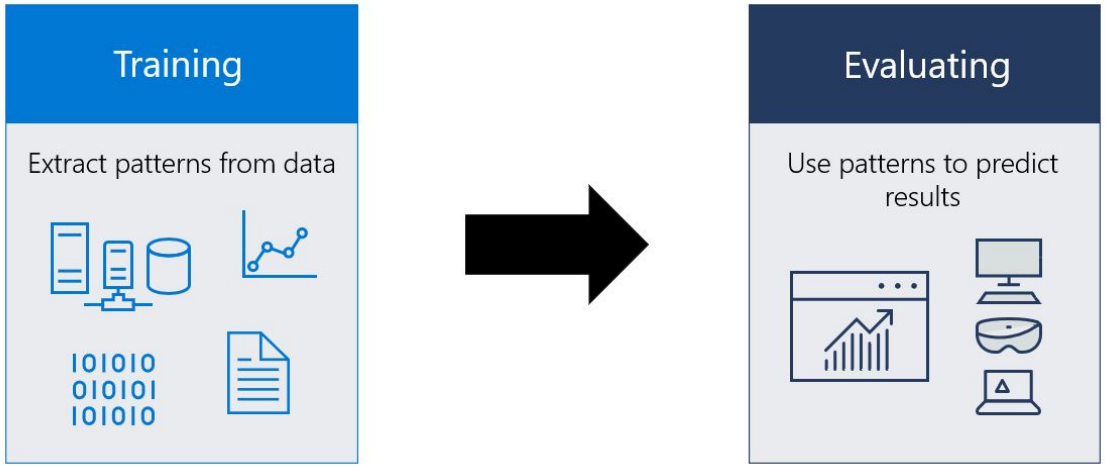
Datasets

Datasets are collections of data pieces that can be treated by a computer as a single unit for analytic and prediction purposes. ‘

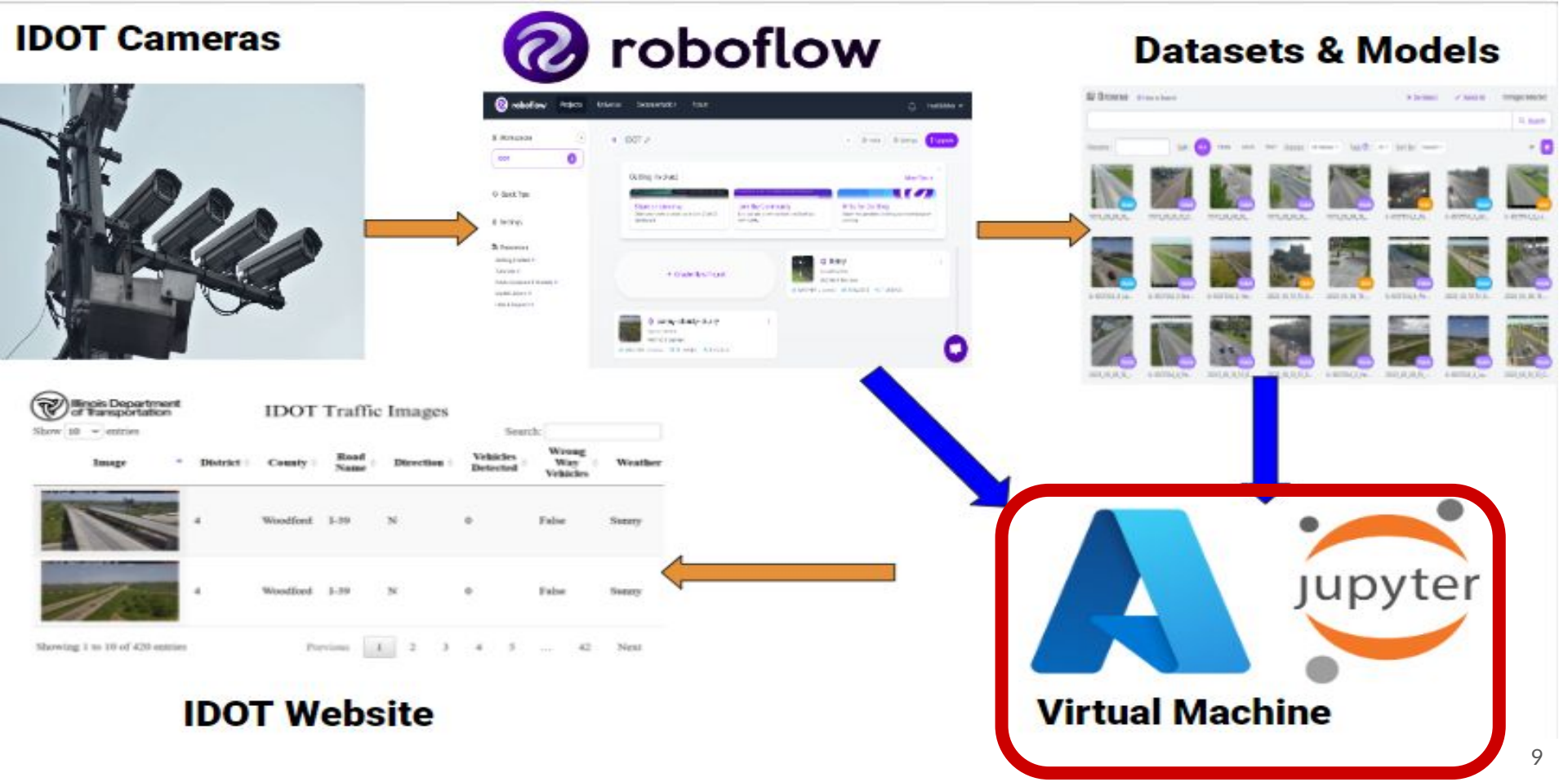


Models

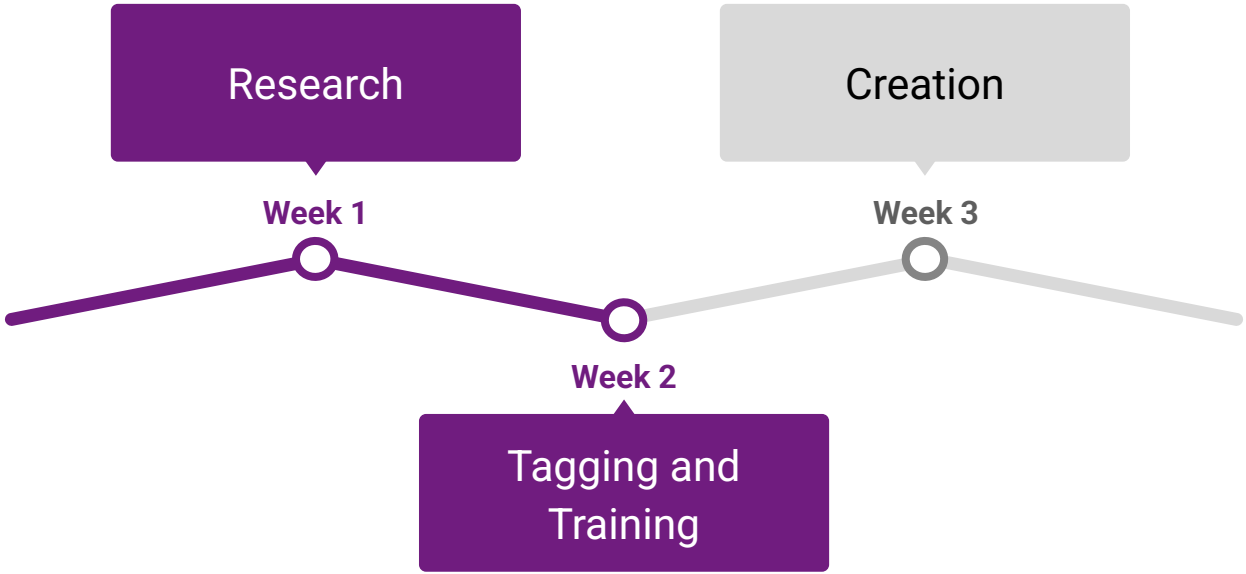
Machine learning models are created by training algorithms with either labeled or unlabeled data, or a mix of both.



Data Flow Chart



Timeline



Initial Research

Srijita: Vehicles
Rebekah: Buildings
Hasti: Emergency vehicles
Habeebah: Weather conditions (snow, rain, etc)
Vira: Lane
Samuel: Wrong-way

Image Tagging Program

```
def load_images(folder_path):
    image_files = [f for f in os.listdir(folder_path) if f.lower().endswith(('.png', '.jpg', '.jpeg', '.gif', '.bmp'))]
    image_files.sort()
    return image_files

def display_image(folder_path, image_files, index):
    return Image(os.path.join(folder_path, image_files[index]))

def tag_image(source_folder, destination_folder, image_files, index, tag):
    source_path = os.path.join(source_folder, image_files[index[0]])
    dt = datetime.fromtimestamp(os.stat(source_path).st_ctime)
    # add timestamp to filename to prevent same camera image from different dates from overwriting each other
    destination_file = f'{dt.year}_{dt.month:02d}_{dt.day:02d}_{dt.hour:02d}_{dt.minute:02d}_' + image_files[index[0]]
    destination_path = os.path.join(destination_folder, tag, destination_file)
    print(f'tag_image({source_path},{destination_path}')
    if not os.path.exists(destination_path):
        shutil.copyfile(source_path, destination_path)
        print(f"Image '{image_files[index[0]]}' tagged with '{tag}' and copied to '{destination_path}'.")
    else:
        print(f"Image '{image_files[index[0]]}' was already tagged with '{tag}'.")

def next_image(image_files, index, display_image_widget, folder_path, image_number_widget):
    index[0] = (index[0] + 1) % len(image_files)
    display_image_widget.value = display_image(folder_path, image_files, index[0]).data
    image_number_widget.value = f"Image {index[0]+1} of {len(image_files)}"
```

```
def previous_image(image_files, index, display_image_widget, folder_path, image_number_widget):
    index[0] = (index[0] - 1) % len(image_files)
    display_image_widget.value = display_image(folder_path, image_files, index[0]).data
    image_number_widget.value = f"Image {index[0]+1} of {len(image_files)}"

def load_tags(tag_folder):
    return [tag for tag in os.listdir(tag_folder) if os.path.isdir(os.path.join(tag_folder,tag))]


def main():
    folder_path = '/home/dillenbu/storage/idot_camera_images/2023/05/12/10/00'
    destination_folder = '/home/dillenbu/storage/datasets'

    image_files = load_images(folder_path)
    index = [0]  # Using a list to make it mutable within the event handler

    previous_button = widgets.Button(description="Previous")
    next_button = widgets.Button(description="Next")
    tag_options = load_tags(destination_folder)
    tag_dropdown = widgets.Dropdown(options=tag_options, description='Select tag:')
    tag_button = widgets.Button(description="Tag")
    image_number_widget = widgets.Label(value=f"Image {index[0]+1} of {len(image_files)}")
```

- Looked over & sorted ~10,000 images individually

```
    previous_button.on_click(lambda x: previous_image(image_files, index, display_image_widget, folder_path, image_number_widget)
    next_button.on_click(lambda x: next_image(image_files, index, display_image_widget, folder_path, image_number_widget))
    tag_button.on_click(lambda x: tag_image(folder_path, destination_folder, image_files, index, tag_dropdown.value))

    display(widgets.HBox([previous_button, image_number_widget, next_button, tag_dropdown, tag_button]))
    display_image_widget = widgets.Image(value=display_image(folder_path, image_files, index[0]).data)
    display(display_image_widget)

    return display_image_widget

isplay_image_widget = main()
```

Previous | Image 13 of 1006 | Next | Select tag: other_vehicles | Tag

Tag Options:

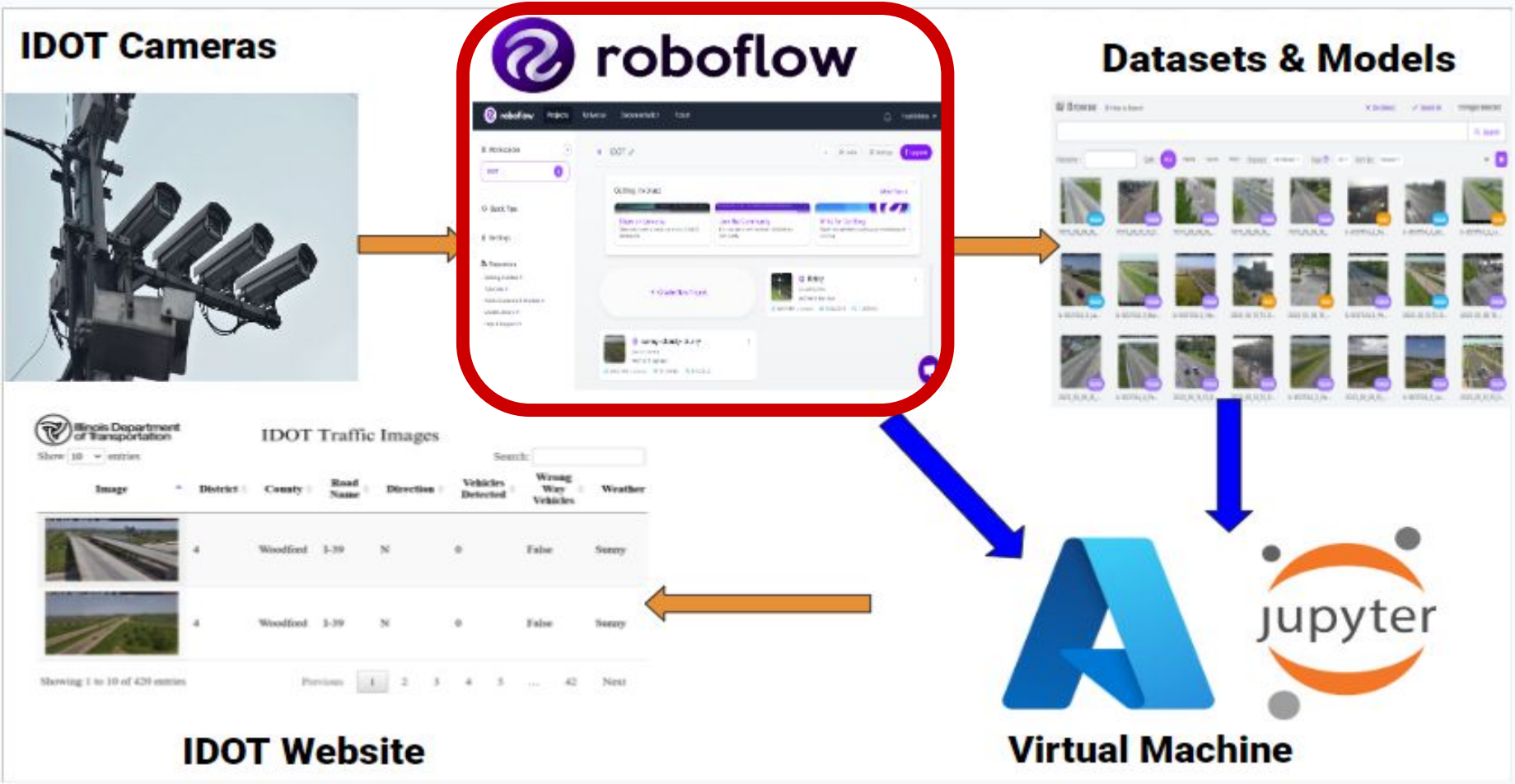
ambiguous	fire_truck	potholes
ambulance	flooding	other_vehicles
animals	fog	cloudy
back_of_vehicle	front-vehicle	rain
blurry	glare	squad_car
car_on_shoulder	house	sunny
tow_truck	train	water_on_lens
water_on_pavement	wrong_way	

Example Images from Tagging

houses tag

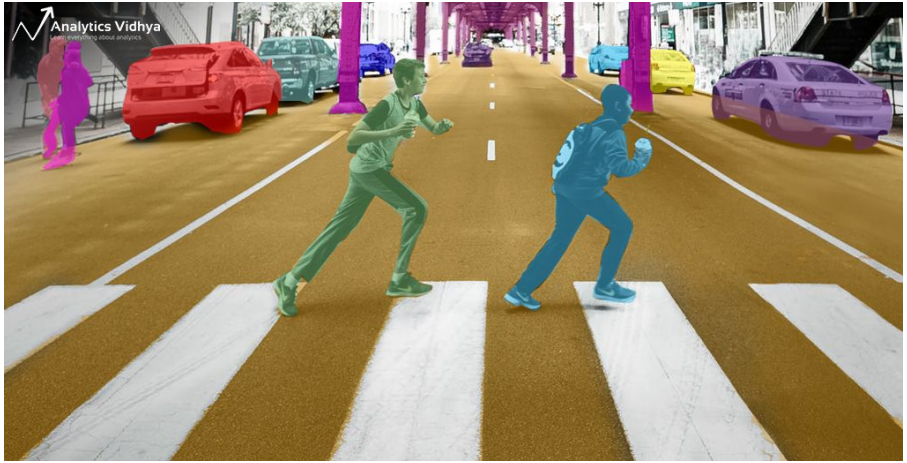
front-vehicle, back_of_vehicle, and cloudy tags

Roboflow


IDOT Cameras
roboflow
Datasets & Models
Illinois Department of Transportation
IDOT Traffic Images
Search:
Image
District
County
Road Name
Direction
Vehicles Detected
Wrong Way Vehicles
Weather
4
Woodford
I-39
N
0
False
Sunny
4
Woodford
I-39
N
0
False
Sunny
Showing 1 to 10 of 420 entries
Previous
1
2
3
4
5
...
42
Next
jupyter
IDOT Website
Virtual Machine

Object Detection

Object Detection/Image segmentation is termed as categorizing each pixel value of an image to a particular class.



Semantic Segmentation

Object segmentation, but doesn’t distinguish between different instances in the same category.

Fig: 1

Fig: 2

Instance Segmentation

Object segmentation, where all instances in each class will be differentiated in an image.

Fig: 1

Fig: 2

Panoptic Segmentation

Semantic segmentation + instance segmentation

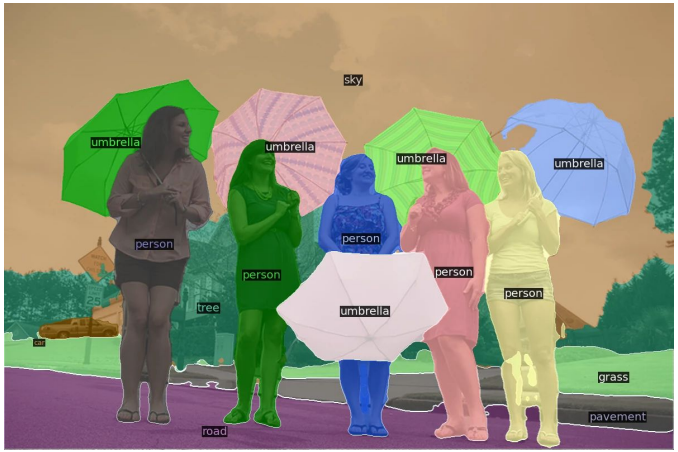


Fig: 1

Fig: 2

Precision

Precision

Of all positive predictions, how many are really positive?

$$\frac{TP}{TP + FP}$$

		Real Class: Positive	Real Class: Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

Recall

Recall

Of all real positive cases, how many are predicted positive?

$$\frac{TP}{TP + FN}$$

		Real Class: Positive	Real Class: Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

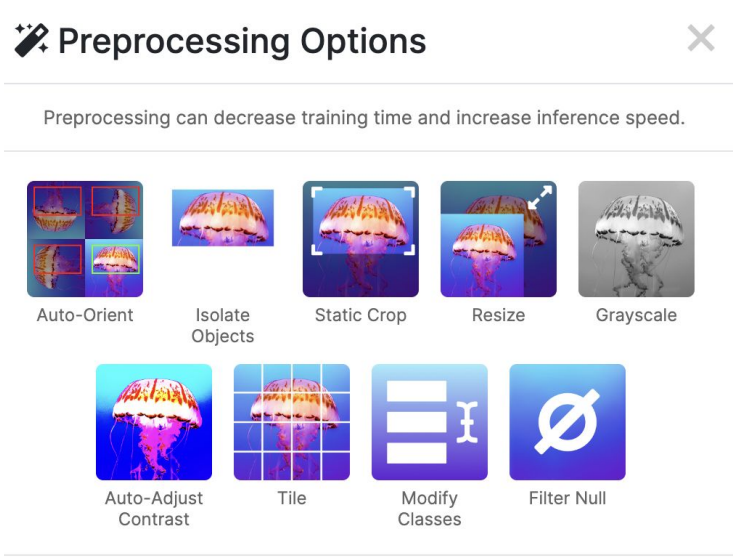
Intersection over Union (IoU)

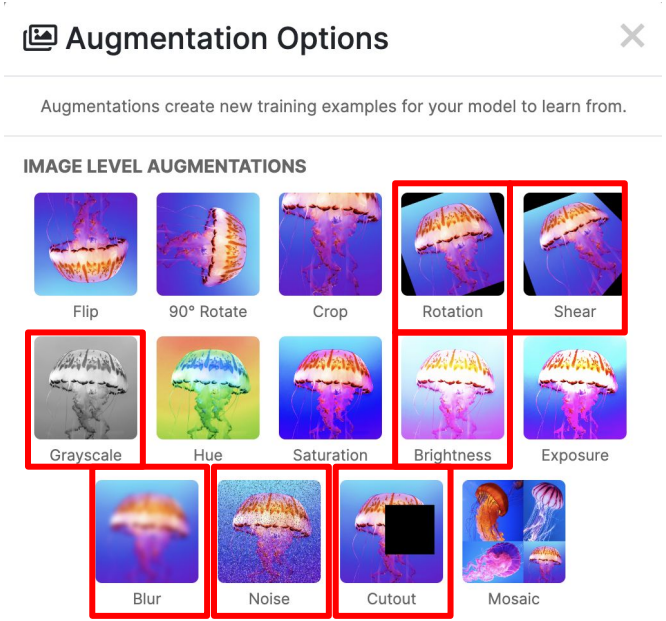
Ground truth

Prediction

$$IoU = \frac{\text{area of overlap}}{\text{area of union}}$$

Preprocessing and augmentation





Houses Using Semantic Segmentation

IoU = 82.4%

Annotated

Model Prediction

Houses Using Object Detection

Precision: 75.6% Recall: 47.3%

Annotated

Model Prediction

All Buildings Using Semantic Segmentation

IoU = 73.8%

Annotated

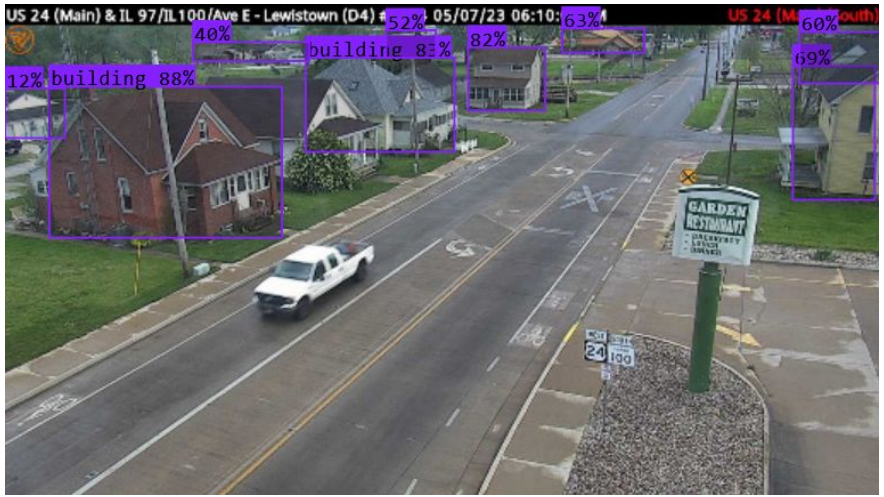
Model Prediction

All Buildings Using Object Detection

Precision: 67.8%

Recall: 51.3%

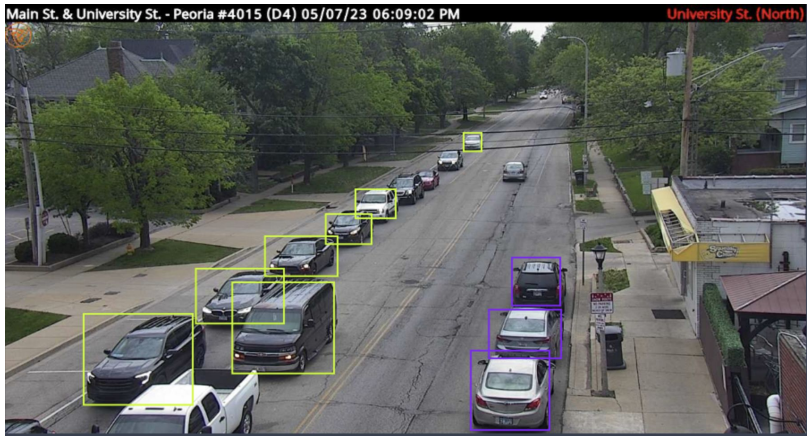
Annotated



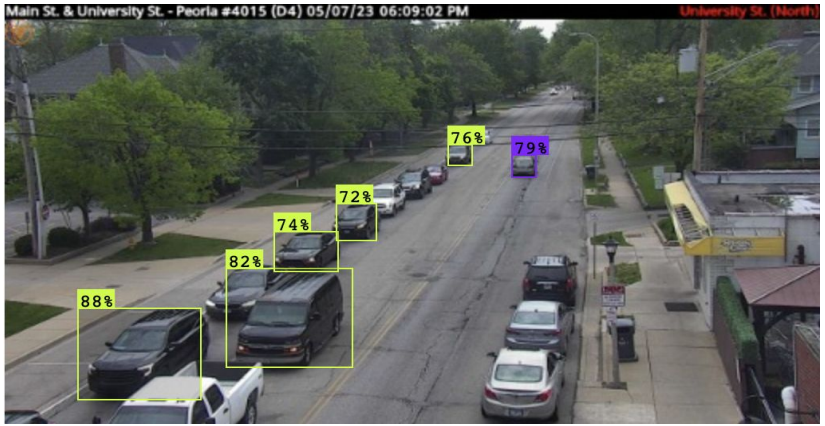
Model Prediction

Front & back of vehicles (instance segmentation)

Precision: 88.2%
Recall: 59.6%



Annotated



Model Prediction

Weather

Validation Accuracy: 96%

Model Prediction (Rain)

Validation Accuracy: 71.4%

Model Prediction (Sunny)

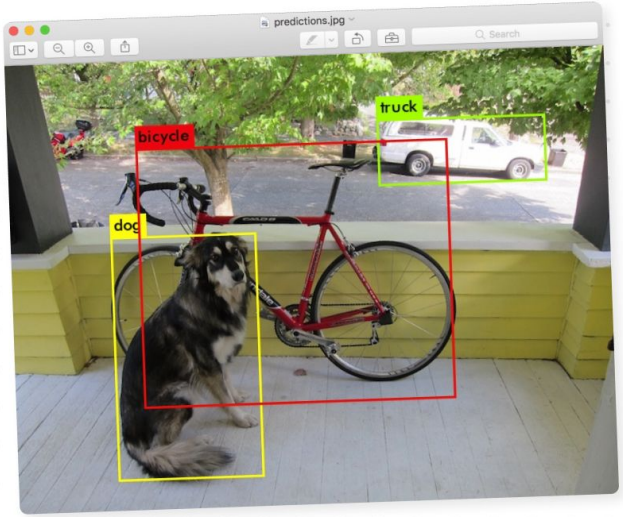
What is YOLO?

YOLO (You Only Look Once) is a computer vision algorithm that uses end-to-end neural networks that makes predictions of bounding boxes and class probabilities all at once.

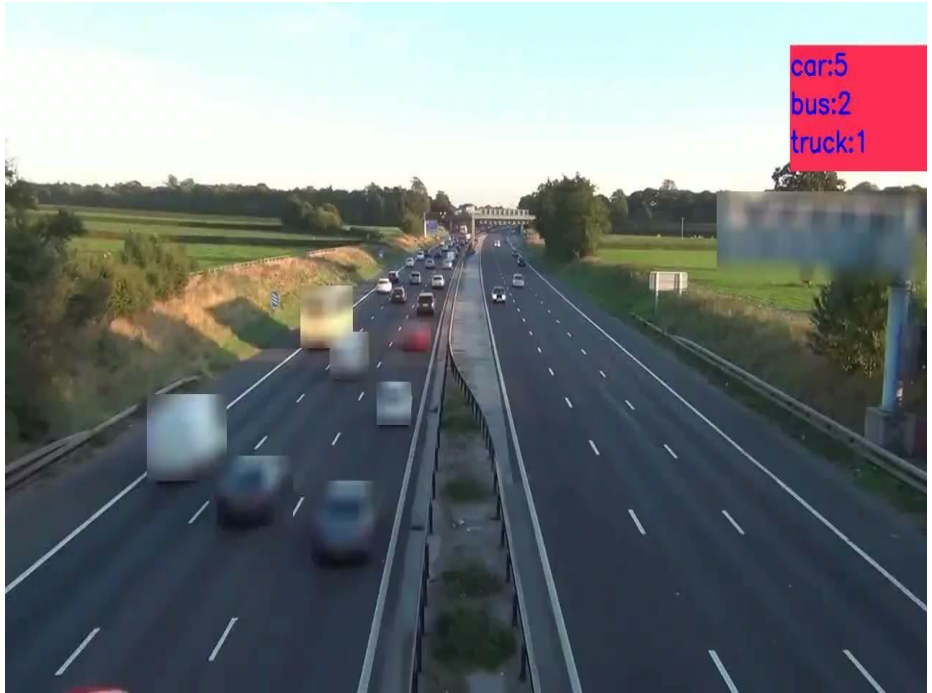
This means that it processes an entire image in a single pass,

making it computationally efficient.

Why YOLO over other algorithms?



What is YOLO?



Functions - Inference

We wrote an inference function to apply our trained model to a dataset and generating an output/prediction.

Before processing, labelling

```python
def preprocess_image(image):
    # Resize the image to the expected input size of the YOLOv5 model
    size = (640, 640)  # Adjust this size according to the model's input requirements
    image = image.resize(size)

    # Convert the image to a NumPy array
    image = np.array(image)

    # Normalize the image
    image = image / 255.0

    # Add a batch dimension
    image = np.expand_dims(image, axis=0)

    # Convert the image array to a PyTorch tensor
    image = torch.from_numpy(image).float()

    # Move the tensor to the appropriate device (CPU or GPU)
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    image = image.to(device)

    return image
```

Functions - Inference

This is the main inference function. Here, the model processes every image in the dataset and then makes predictions. Then, every image with bounding boxes are saved to a directory containing the outputs.


```python
def inference_on_dataset(model, dataset_path, output_path):
    # Set the model to evaluation mode
    model.eval()

    # Create the output directory if it doesn't exist
    Path(output_path).mkdir(parents=True, exist_ok=True)

    # Get a list of image files in the dataset directory
    image_files = Path(dataset_path).glob('*.jpg')

    # Process each image in the dataset
    # for image_file in image_files:
        # Load the image
    image = Image.open(image_file).convert('RGB')

    # Preprocess the image
    image_tensor = F.to_tensor(image)
    image_tensor = image_tensor.unsqueeze(0)  # Add batch dimension

    # Perform inference
    with torch.no_grad():
        detections = model(image_tensor)

    # Post-process the detections
    # TODO: Implement your post-processing logic here
    # ...

    # Save the image with bounding boxes
    output_image_path = Path(output_path) / image_file.name
    image.save(output_image_path)

    print(f"Inference completed for {image_file.name}. Result saved at {output_image_path}")
```

Functions - Blur

```python
def blur_boxes(image, boxes):
    image = np.copy(image)
    for box in boxes:
        x, y, w, h = box
        roi = image[y:y+h, x:x+w]
        roi = cv2.GaussianBlur(roi, (23, 23), 30)
        image[y:y+h, x:x+w] = roi
    return image
```

Before Blurring

After Blurring

Website

Illinois Department of Transportation

IDOT Traffic Images

Show 10 entries

Search:

Image	District	County	Road Name	Direction	Vehicles Detected	Wrong Way Vehicles	Weather
	4	Woodford	I-39	N	0	False	Sunny
	4	Woodford	I-39	N	0	False	Sunny

Showing 1 to 10 of 420 entries

Previous 1 2 3 4 5 … 42 Next

Mackinaw Dells Parking lot

Eastbound Mackinaw Dells Rest Stop

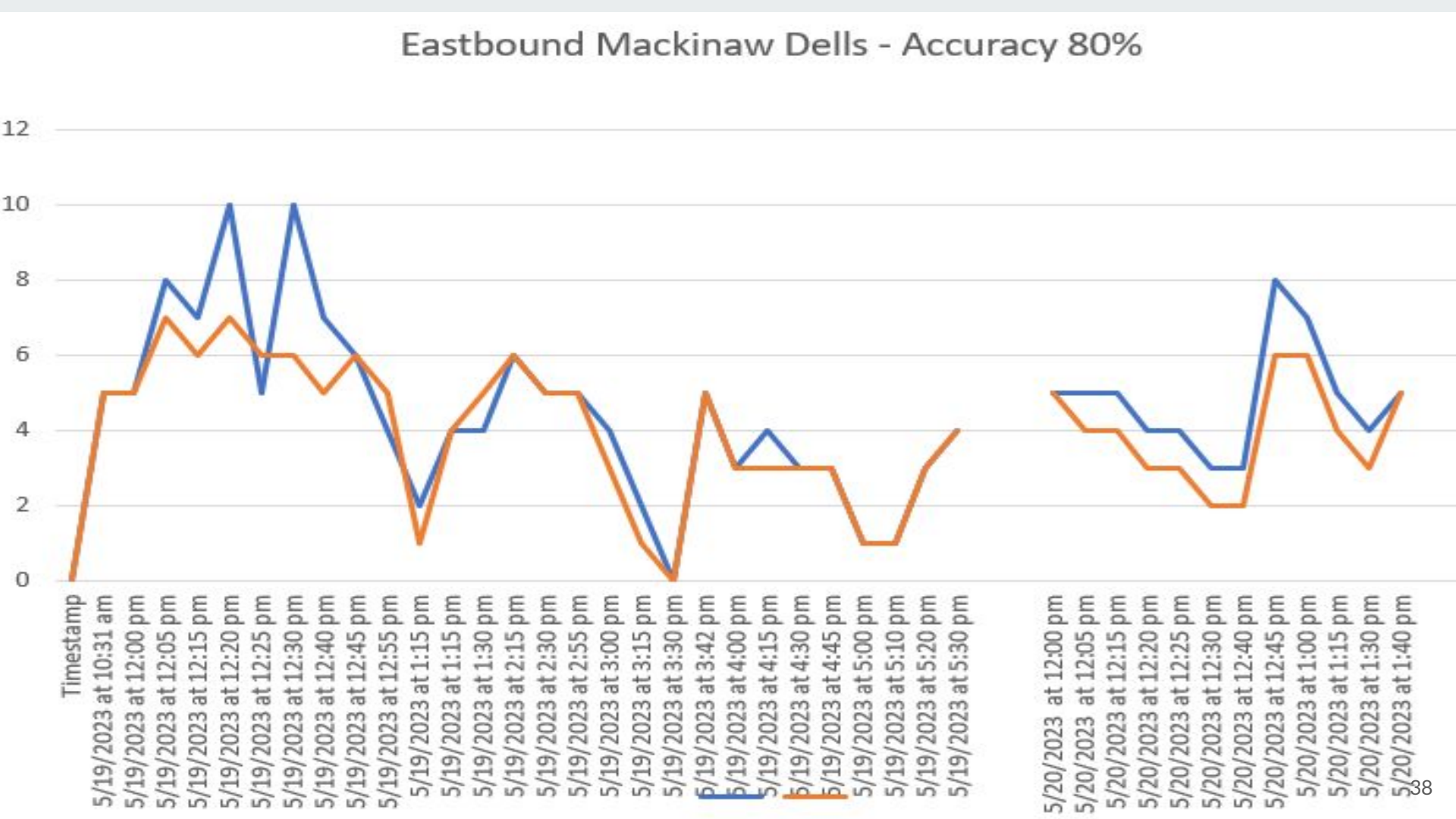
Live Image

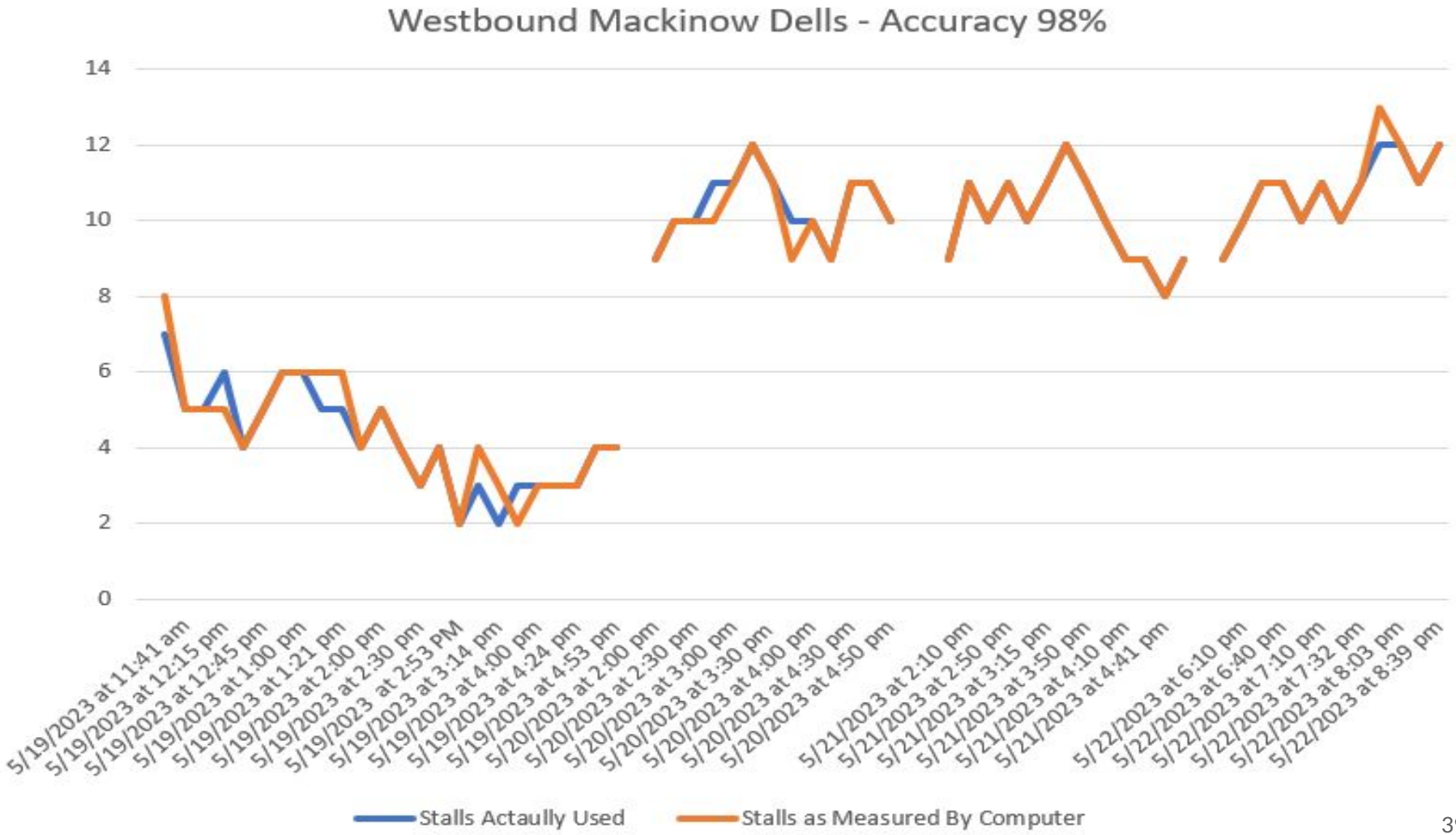
Westbound Mackinaw Dells Rest Stop

Live Image

Stalls Used

EB - 7th and 19th Stalls not detected


Eastbound Mackinaw Dells - Accuracy 80%
12
10
8
6
4
2
0
Timestamp
5/19/2023 at 10:31 am
5/19/2023 at 12:00 pm
5/19/2023 at 12:05 pm
5/19/2023 at 12:15 pm
5/19/2023 at 12:20 pm
5/19/2023 at 12:25 pm
5/19/2023 at 12:30 pm
5/19/2023 at 12:40 pm
5/19/2023 at 12:45 pm
5/19/2023 at 12:55 pm
5/19/2023 at 1:15 pm
5/19/2023 at 1:15 pm
5/19/2023 at 1:30 pm
5/19/2023 at 2:15 pm
5/19/2023 at 2:30 pm
5/19/2023 at 2:55 pm
5/19/2023 at 3:00 pm
5/19/2023 at 3:15 pm
5/19/2023 at 3:30 pm
6/19/2023 at 3:42 pm
6/19/2023 at 4:00 pm
5/19/2023 at 4:15 pm
6/19/2023 at 4:30 pm
6/19/2023 at 4:45 pm
5/19/2023 at 5:00 pm
5/19/2023 at 5:10 pm
5/19/2023 at 5:20 pm
5/19/2023 at 5:30 pm
5/20/2023 at 12:00 pm
5/20/2023 at 12:05 pm
5/20/2023 at 12:15 pm
5/20/2023 at 12:20 pm
5/20/2023 at 12:25 pm
5/20/2023 at 12:30 pm
5/20/2023 at 12:40 pm
5/20/2023 at 12:45 pm
5/20/2023 at 1:00 pm
5/20/2023 at 1:15 pm
5/20/2023 at 1:30 pm
5/20/2023 at 1:40 pm


Westbound Mackinow Dells - Accuracy 98%
14
12
10
8
6
4
2
0
5/19/2023 at 11:41 am
5/19/2023 at 12:15 pm
5/19/2023 at 12:45 pm
5/19/2023 at 1:00 pm
5/19/2023 at 1:21 pm
5/19/2023 at 2:00 pm
5/19/2023 at 2:30 pm
5/19/2023 at 2:53 PM
5/19/2023 at 3:14 pm
5/19/2023 at 4:00 pm
5/19/2023 at 4:24 pm
5/19/2023 at 4:53 pm
5/20/2023 at 2:00 pm
5/20/2023 at 2:30 pm
5/20/2023 at 3:00 pm
5/20/2023 at 3:30 pm
5/20/2023 at 4:00 pm
5/20/2023 at 4:30 pm
5/20/2023 at 4:50 pm
5/21/2023 at 2:10 pm
5/21/2023 at 2:50 pm
5/21/2023 at 3:15 pm
5/21/2023 at 3:50 pm
5/21/2023 at 4:10 pm
5/21/2023 at 4:41 pm
5/22/2023 at 6:10 pm
5/22/2023 at 6:40 pm
5/22/2023 at 7:10 pm
5/22/2023 at 7:32 pm
5/22/2023 at 8:03 pm
5/22/2023 at 8:39 pm
Stalls Actaully Used
Stalls as Measured By Computer

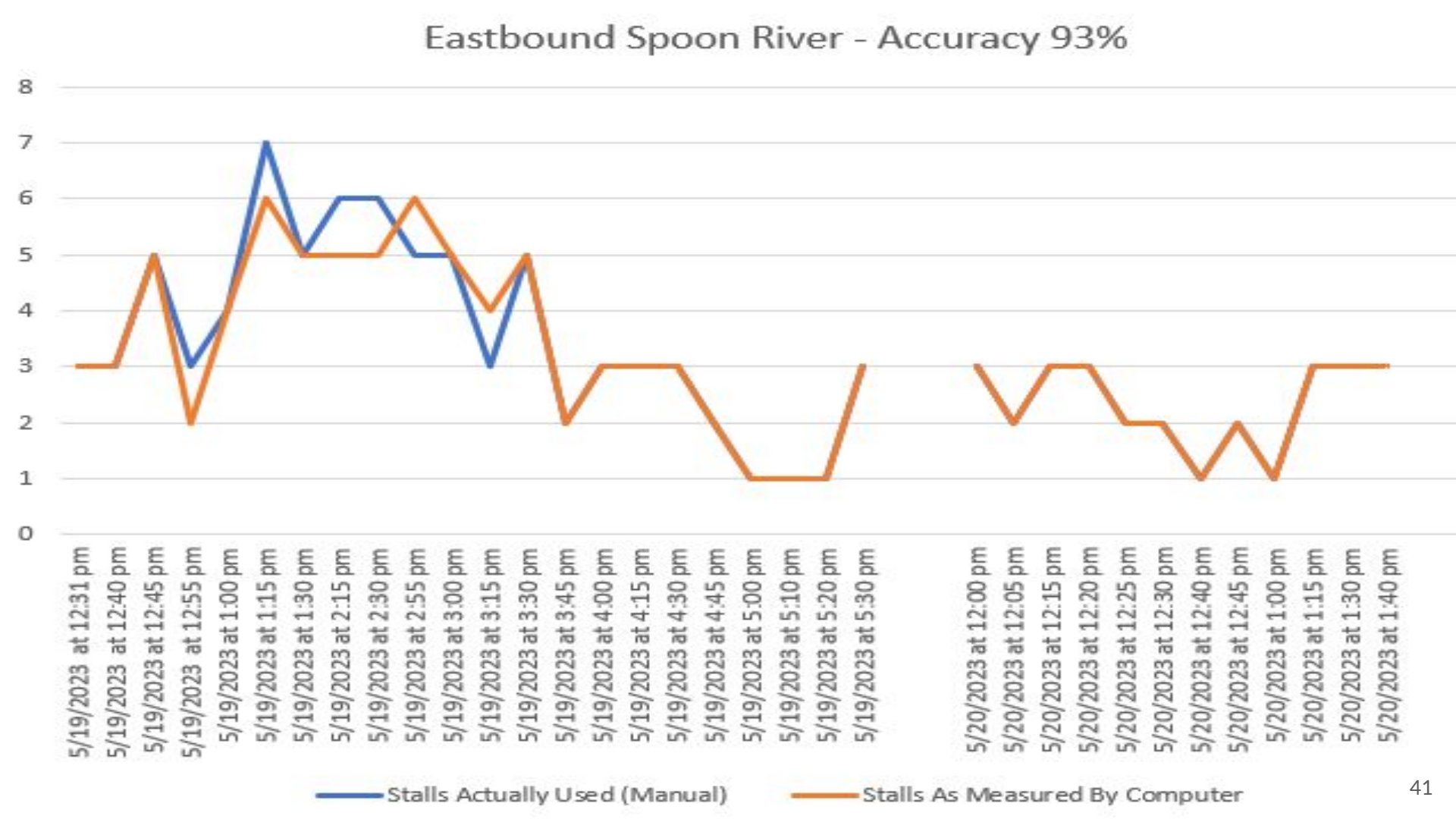
Spoon River Parking lot

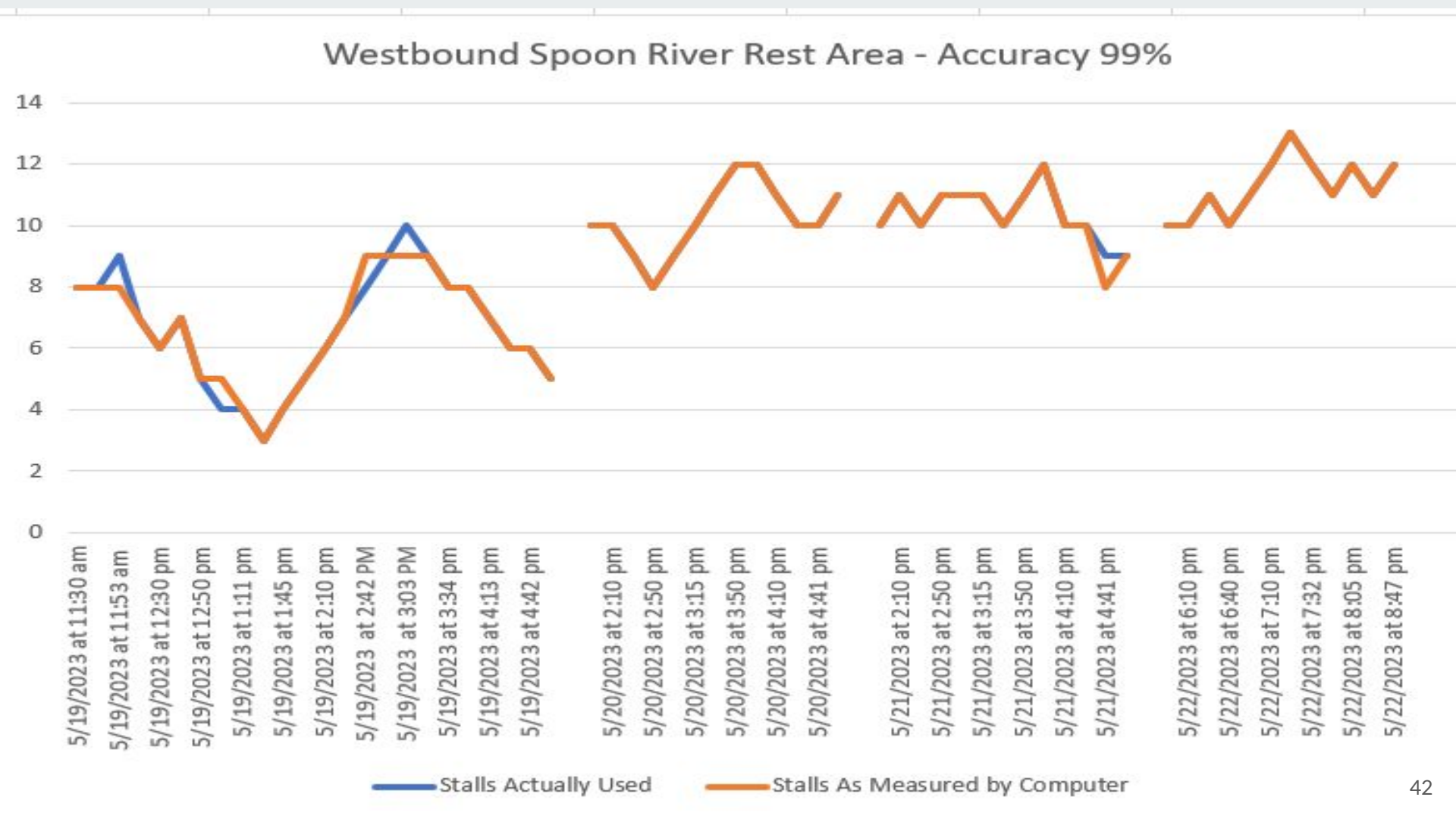
Eastbound Spoon River Rest Area

Live Image

Westbound Spoon River Rest Area

Live Image


Eastbound Spoon River - Accuracy 93%
8
7
6
5
4
3
2
1
0
5/19/2023 at 12:31 pm
5/19/2023 at 12:40 pm
5/19/2023 at 12:45 pm
5/19/2023 at 12:55 pm
5/19/2023 at 1:00 pm
5/19/2023 at 1:15 pm
5/19/2023 at 1:30 pm
5/19/2023 at 2:15 pm
5/19/2023 at 2:30 pm
5/19/2023 at 2:55 pm
5/19/2023 at 3:00 pm
5/19/2023 at 3:15 pm
5/19/2023 at 3:30 pm
5/19/2023 at 3:45 pm
5/19/2023 at 4:00 pm
5/19/2023 at 4:15 pm
5/19/2023 at 4:30 pm
5/19/2023 at 4:45 pm
5/19/2023 at 5:00 pm
5/19/2023 at 5:10 pm
5/19/2023 at 5:20 pm
5/19/2023 at 5:30 pm
5/20/2023 at 12:00 pm
5/20/2023 at 12:05 pm
5/20/2023 at 12:15 pm
5/20/2023 at 12:20 pm
5/20/2023 at 12:25 pm
5/20/2023 at 12:30 pm
5/20/2023 at 12:40 pm
5/20/2023 at 12:45 pm
5/20/2023 at 1:00 pm
5/20/2023 at 1:15 pm
5/20/2023 at 1:30 pm
5/20/2023 at 1:40 pm
Stalls Actually Used (Manual)
Stalls As Measured By Computer


Westbound Spoon River Rest Area - Accuracy 99%
14
12
10
8
6
4
2
0
5/19/2023 at 11:30 am
5/19/2023 at 11:53 am
5/19/2023 at 12:30 pm
5/19/2023 at 12:50 pm
5/19/2023 at 1:11 pm
5/19/2023 at 1:45 pm
5/19/2023 at 2:10 pm
5/19/2023 at 2:42 PM
5/19/2023 at 3:03 PM
5/19/2023 at 3:34 pm
5/19/2023 at 4:13 pm
5/19/2023 at 4:42 pm
5/20/2023 at 2:10 pm
5/20/2023 at 2:50 pm
5/20/2023 at 3:15 pm
5/20/2023 at 3:50 pm
5/20/2023 at 4:10 pm
5/20/2023 at 4:41 pm
5/21/2023 at 2:10 pm
5/21/2023 at 2:50 pm
5/21/2023 at 3:15 pm
5/21/2023 at 3:50 pm
5/21/2023 at 4:10 pm
5/21/2023 at 4:41 pm
5/22/2023 at 6:10 pm
5/22/2023 at 6:40 pm
5/22/2023 at 7:10 pm
5/22/2023 at 7:32 pm
5/22/2023 at 8:05 pm
5/22/2023 at 8:47 pm
Stalls Actually Used
Stalls As Measured by Computer

Accuracy Table

Location	Accuracy
Eastbound Mackinaw Dells	80%
Westbound Mackinaw Dells	98%
Eastbound Spoon River	93%
Westbound Spoon River	99%

What’s Left To Do

- Continue with image tagging on the live images coming in.
- Continue training the object detection models to achieve higher precision and recall rates
- Complete the website with IDOT input for future development of the website
- Continue monitoring the parking lots from legacy Sprintern projects
    - Having a per stall percentage accuracy graph

Applications + Recommendations

- Privacy issue prevention
- Detecting possible road damage
- Detecting crashes
- Detecting high traffic areas to decrease congestion

Recommendations

- Find a free, reliable, open-source platform for creating and labelling our datasets as an alternative to Roboflow.
- Use a subscriber-based ML cloud environment to improve onboarding and development of the app.

Technical Skills Learned

- Learned about Machine Vision
- Learned how to use Roboflow to label images
- Learned how to use JupyterLab and cloud environments
- Learned how to use Python libraries like Pythorch and wrote Python scripts for using them
- Learned about AI models like Detectron and YOLOv5

Non-Technical Lessons Learned

- Don’t be afraid to ask for help
- Ask Questions
- Be open-minded
- Persistence in finding new solutions
- Teamwork

Appreciation

- John Dillenburg
- IDOT
- Break Through Tech Chicago

Complete Outline

Time spent
-green = short
-yellow = medium
- red = long

1. Welcome **(John)**
2. Introductions (**Everyone**)
    a. Name, major, hobby, where are you from?
3. Project Overview **(Sam)**
4. Explaining what Machine Vision, datasets, models (**Habeebah**)
5. Tools we used (**Hasti**)
    a. Virtual machine (jupyterlab, azure, google colab)
    b. Have some sort of dataflow chart: show travel midwest.com website and then arrow to microsoft azure machines with label of 3k images and at other end is the machine v
    website. To left od website would be a box that says idot travel cameras
        i. Backend? - due to the time constraints that we had, John assisted in creating a Jupyterlab environment on Azure. **(Srijita)**
6. Timeline (**Rebekah**)
7. Initial Research (**Vira**)
8. Explaining the image tagging we did in jupyterlab with program. (**Rebekah**)
    a. Describe different categories like sunny, firetruck, ambulance, rainy, glare
        i. Include example images, scatter them on page
9. Explaining Roboflow and difference between models (**Srijita**)
    a. Object, semantic, instance, panotropic, image recognition (5 simple slides - 2 max bullet points, mostly pictures)
10. Roboflow (Split up based on responsibilities, data type, model trained) - slides per person/team containing roboflow screenshot
    a. Explanation of precision, recall, and IoU (**Vira**)
    b. Explain lack of pre-processing and augmentation methods - why we did those things (**Vira**)
    c. Explain the house, weather, front-back vehicle
11. What is Yolo and why we had to custom train (**Sam**)
12. Functions (**Sam** and **Vira**)
    a. Sam starts his function then vira
    b. Show some code and maybe make a cohesion slide
13. Website (**Rebekah**)
    a. Put mockup or redraw
14. Accuracy for the parking lots, east bound, west bound Add all 4 graphs (**Hasti**) and (**John**)
    a. Mention that stall #19 isn't working
        i. Include a screenshot of it not working
15. What's Left to do for completion of this project (**Sam**)
    a. Mention doing more tagging
    b. Complete the website, train the models better, not a bad thing we didn't complete the website, wanted to have more of idiots input so we can have their input for future development of it, continue mon
    the parking lots, have a per stall accuracy graph (percentage accuracy),
16. How does this apply to IDOT (**Habeebah**) (skip if short on time)
17. Lessons learned from the past three weeks (**Habeebah**) discuss some setbacks
18. Appreciation and shout outs (IDOT for sponsoring internship (**Habeebah**)
19. Questions