Group 9

# Drowsy Driver Detection

Sameera Boppana

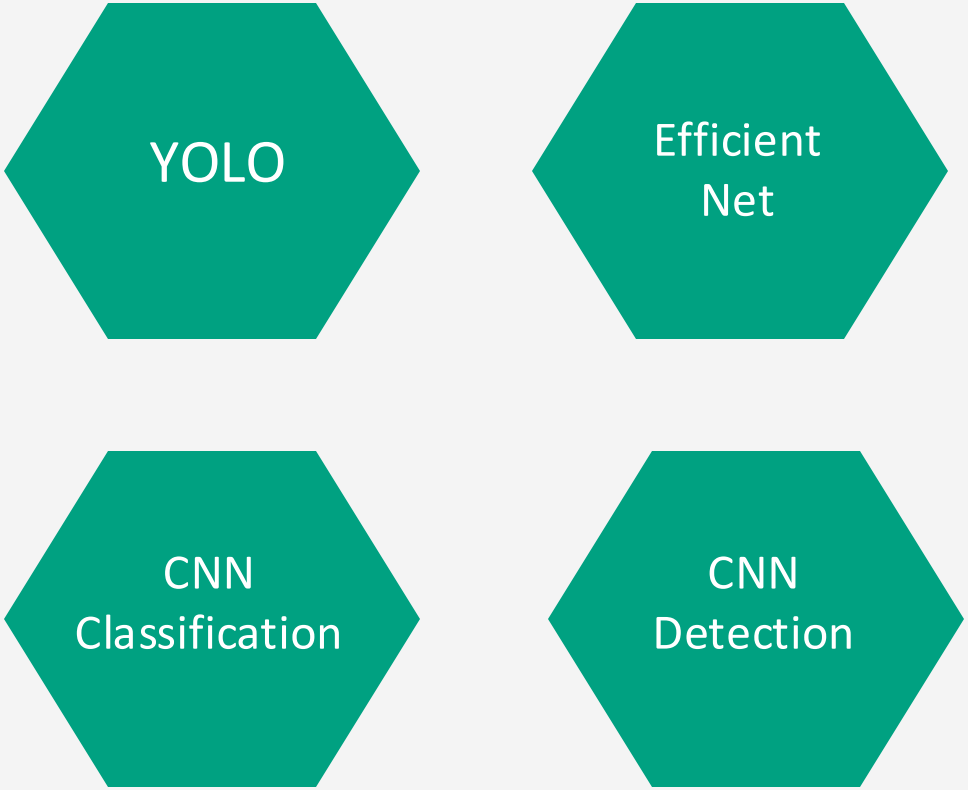
# Content

- Problem Objective
- Data Gathering
- Exploratory Data Analysis
- Image Preprocessing
- Models + Results

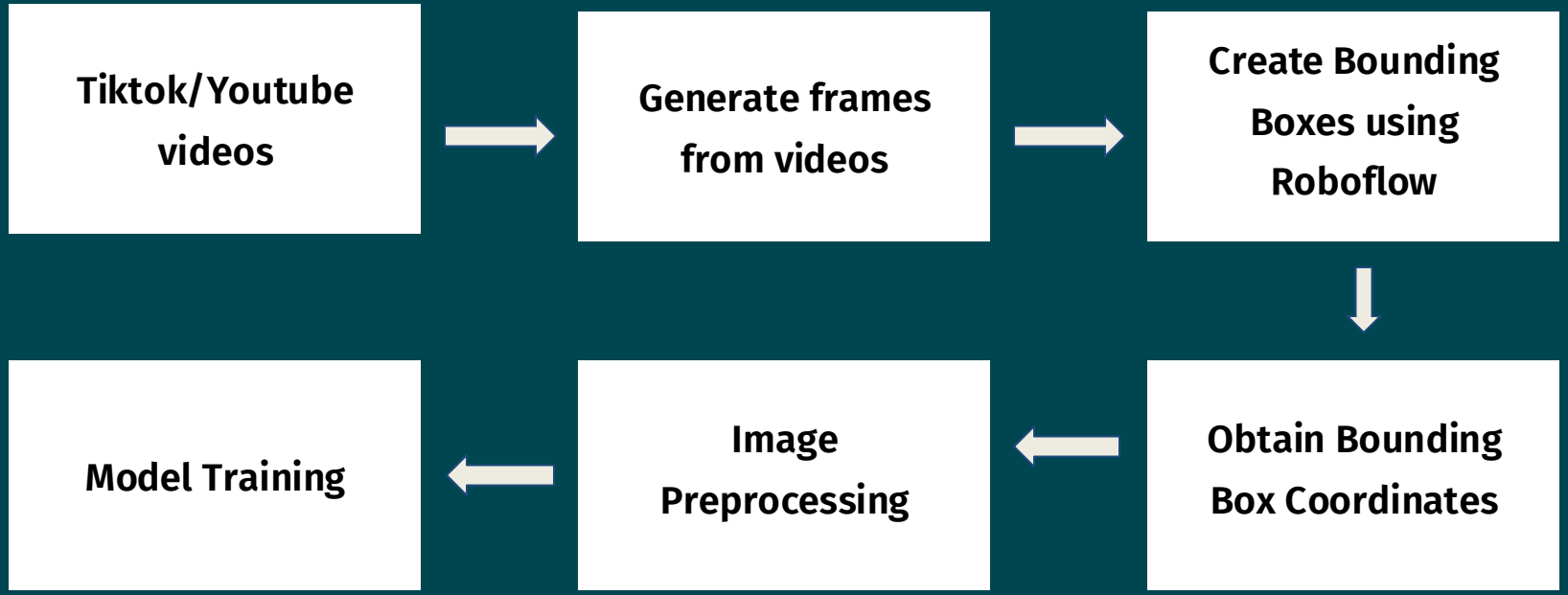
Group 9

# Problem Objective

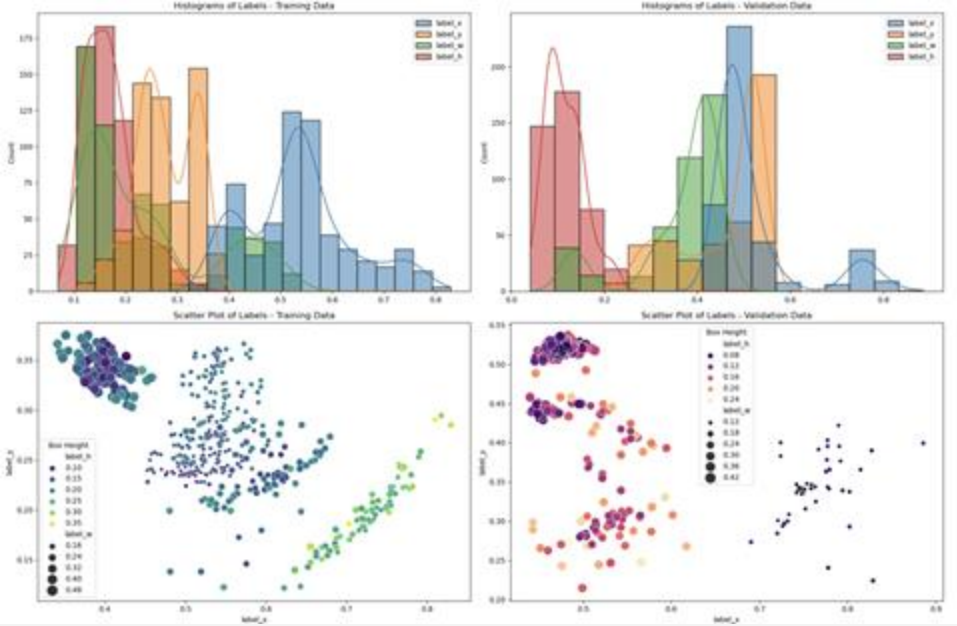
Utilizing deep learning models to detect if a driver appears to be drowsy when driving



# Data Gathering



# Exploratory Data Analysis




**Dataset Overview:**

- Training Data: 590 images
- Validation Data: 418 images
- No Missing Values

**Histogram Analysis:**

- Histogram shows distribution of bounding box position and size on images
- The distribution of the image size varies but is slightly right skewed - most bounding boxes are smaller

**Scatterplot Analysis:**

- Scatterplots show the relationship between x and y coordinates of the bounding boxes - colored by width and height

# Image + Data Preprocessing

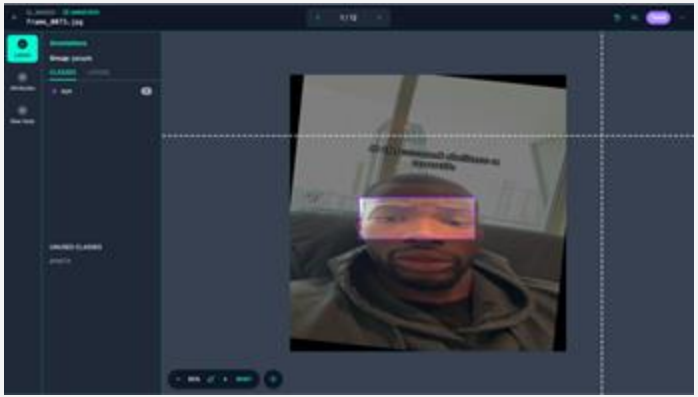
Object Classification:

- Load and convert images to RGB
- Resize images to 244x244 pixels
- Convert class labels to integer format

Object Detection:

- Images are resized to fit within a given input size while maintaining the original aspect ratio
- A new image matrix is created to ensure that the image fits a standard size
- Bounding boxes are recalculated to correspond to the resized image dimensions.

Using RoboFlow to Create Boundary Box Example:



# IOU Metric

Intersection over Union (IoU) is a performance metric that measures the overlap between the predicted bounding box and the ground truth bounding box.

IOU = (Area of Overlap) / (Area of Union)

IOU Values range from 0 - 1:

- 0: There is no overlap between the predicted bounding box and the ground truth
- 1: Perfect overlap between the predicted bounding box and the ground truth

# Object Detection - CNN

## Model Architecture:

- Input Layer: 244x244 pixel images
- Convolutional Layers: 64 filters, 3x3 kernel size, ReLU activation
- Output Layers: Fully connected layers output bounding box coordinates

## Regularization:

- Dropout Rate: Around 0.5 in specific layers to reduce overfitting
- L2 Regularization: Applied to penalize large weights and control overfitting

## Results:

- Validation IoU: 0.7474900484085083
- Test IOU: 0.09928718954324722

# Object Detection - Pretrained YOLOv8

```
!yolo task=detect mode=train model=yolov8m.pt data={dataset.location}/data.yaml epochs=30 imgsz=640
```

```
names:
- eyes
nc: 1
roboflow:
  license: CC BY 4.0
  project: drowsy_driver-i2wpd
  url: https://universe.roboflow.com/luka-24jet/drowsy_driver-i2wpd/dataset/8
  version: 8
  workspace: luka-24jet
test: ../test/images
train: drowsy_driver-8/train/images
val: drowsy_driver-8/valid/images
```

# Object Detection - Pretrained YOLOv8

```
module
ultralytics.nn.modules.conv.Conv
ultralytics.nn.modules.conv.Conv
ultralytics.nn.modules.block.C2f
ultralytics.nn.modules.conv.Conv
ultralytics.nn.modules.block.C2f
ultralytics.nn.modules.conv.Conv
ultralytics.nn.modules.block.C2f
ultralytics.nn.modules.conv.Conv
ultralytics.nn.modules.block.C2f
ultralytics.nn.modules.block.SPPF
torch.nn.modules.upsampling.Upsample
ultralytics.nn.modules.conv.Concat
ultralytics.nn.modules.block.C2f
torch.nn.modules.upsampling.Upsample
ultralytics.nn.modules.conv.Concat
ultralytics.nn.modules.block.C2f
ultralytics.nn.modules.conv.Conv
ultralytics.nn.modules.conv.Concat
ultralytics.nn.modules.block.C2f
ultralytics.nn.modules.conv.Conv
ultralytics.nn.modules.conv.Concat
ultralytics.nn.modules.block.C2f
ultralytics.nn.modules.head.Detect
```

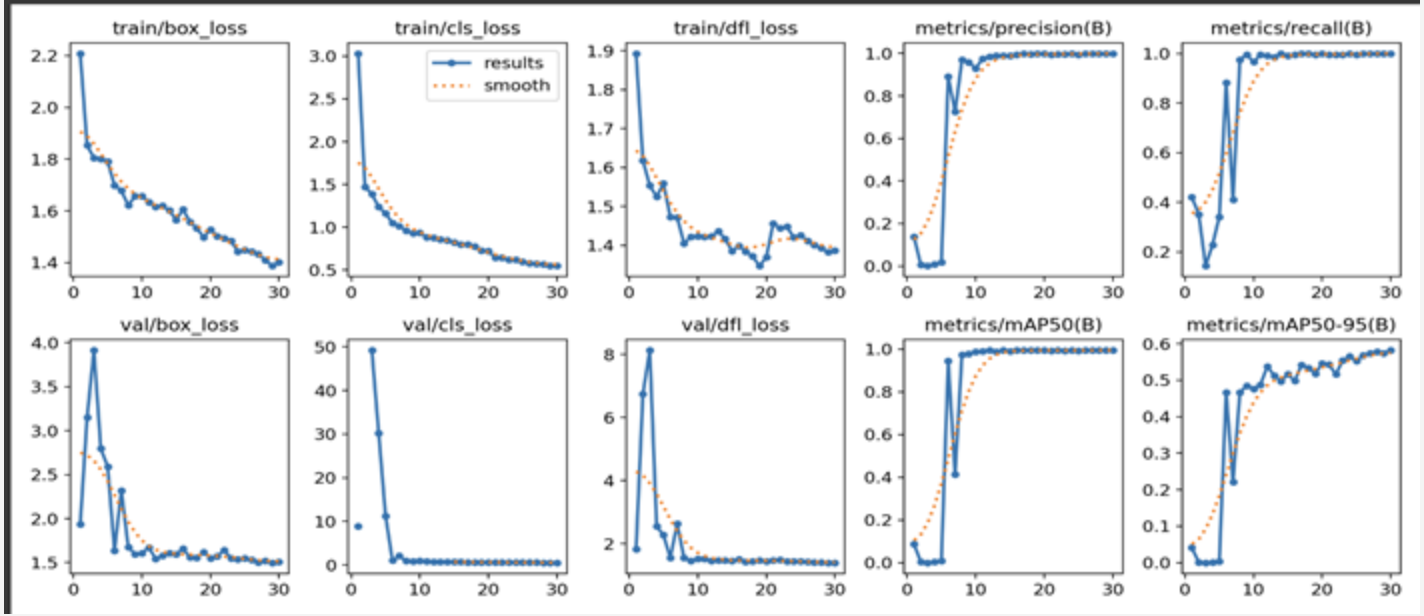
- Architecture
  - Layers:
    - **Convolution**
    - **C2f** (self-defined, with Batch-Norm etc.)
    - **Upsampling** (increase resolution)
    - **SPPF** (Spatial Pyramid Pooling)
    - **Concat** (combine feature maps from layers)
    - **Detect** (object detection layer)

```
train: Scanning /content/drowsy_driver-8/train/labels.cache... 712 images,
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_li
/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeWarning: os.fo
  self.pid = os.fork()
val: Scanning /content/drowsy_driver-8/valid/labels.cache... 202 images, 0
Plotting labels to runs/detect/train3/labels.jpg...

optimizer: 'optimizer=auto' found, ignoring 'lr0=0.01' and 'momentum=0.937'
optimizer: AdamW(lr=0.002, momentum=0.9) with parameter groups 77 weight(de
```

# Object Detection - Pretrained YOLOV8

```
# plot the model training results
Image(filename='/content/runs/detect/train/results.png', width=800)
```




- **AP:** Computes the area under the precision-recall curve
- **MAP50:** Mean average precision when the IOU overlap is greater than 50%
- **MAP50-95:** Averaging the AP values from threshold 0.5 to 0.95 by 0.05
- Quick Demo!

# Object Classification - CNN

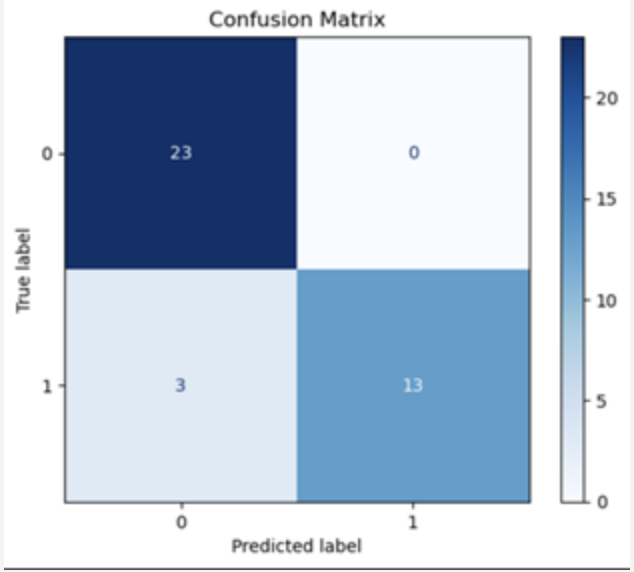
Model Architecture:

- Convolutional Layers:
  - Three layers with filter sizes escalating from 32 to 256.
  - Each layer features kernel size options of 3 or 5, max pooling, and L2 regularization.
- Dense and Dropout Layers:
  - One fully connected layer with 64 to 256 units, L2 regularization.
  - Dropout layer with rates from 0.2 to 0.5 to reduce overfitting.
- Output Layer:
  - Single dense layer with sigmoid activation for binary classification.

# Object Classification - CNN

Results:

- Validation Accuracy: 0.9659090638160706
- Test Accuracy: 0.9230769276618958




**0 = Eyes Open; 1 = Eyes Closed**

Accuracy = 0.92

Precision for "Eyes Closed" = 1

Recall for "Eyes Closed" = 0.81

F1 Score for "Eyes Closed" = 0.9

# Object Classification - Efficient Net

Model Architecture:

- EfficientNetB0:
  - Pre-trained on ImageNet: A large dataset with over 14 million labeled images.
  - Used as the base model without the top layers: Utilizes learned features from pre-training but customizes the final classification layers.

- Custom Layers:
  - GlobalAveragePooling2D
  - Dense Layer (128 units): Fully connected layer with ReLU activation for feature extraction.
  - Output Layer (2 units): Fully connected layer with Softmax activation for binary classification.

- Model Compilation:
  - Optimizer: Adam
  - Loss Function: Sparse Categorical Cross-entropy
  - Metrics: Accuracy

# Object Classification - Efficient Net

Results:

- Train Accuracy: 0.9974
- Test Accuracy:  0.9487

| | Predicted Eyes Open | Predicted Eyes Closed |
|---|---|---|
| True Eyes Open | 21 | 2 |
| True Eyes Closed | 0 | 16 |

Accuracy = 0.94
Precision for "Eyes Closed" = 0.89
Recall for "Eyes Closed" = 1.00

# Thank you!

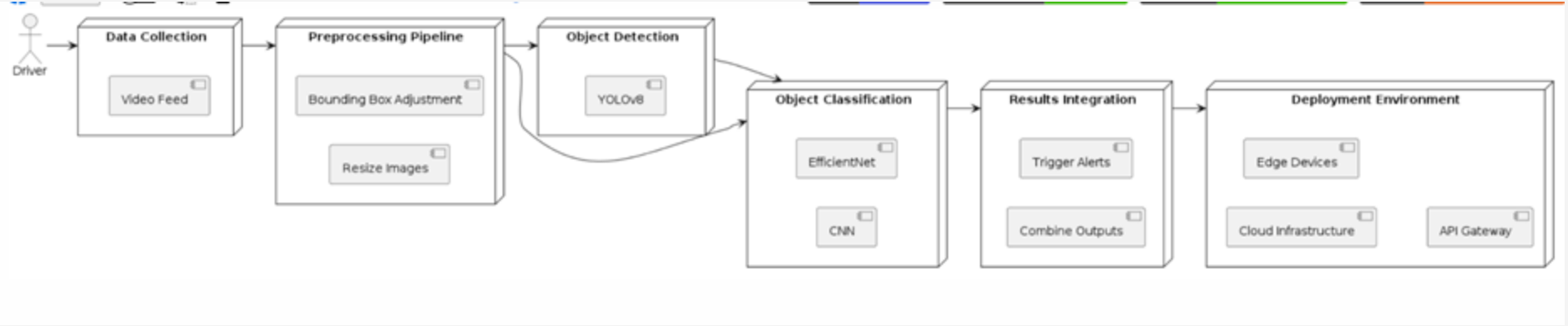
# Appendix

# Challenges

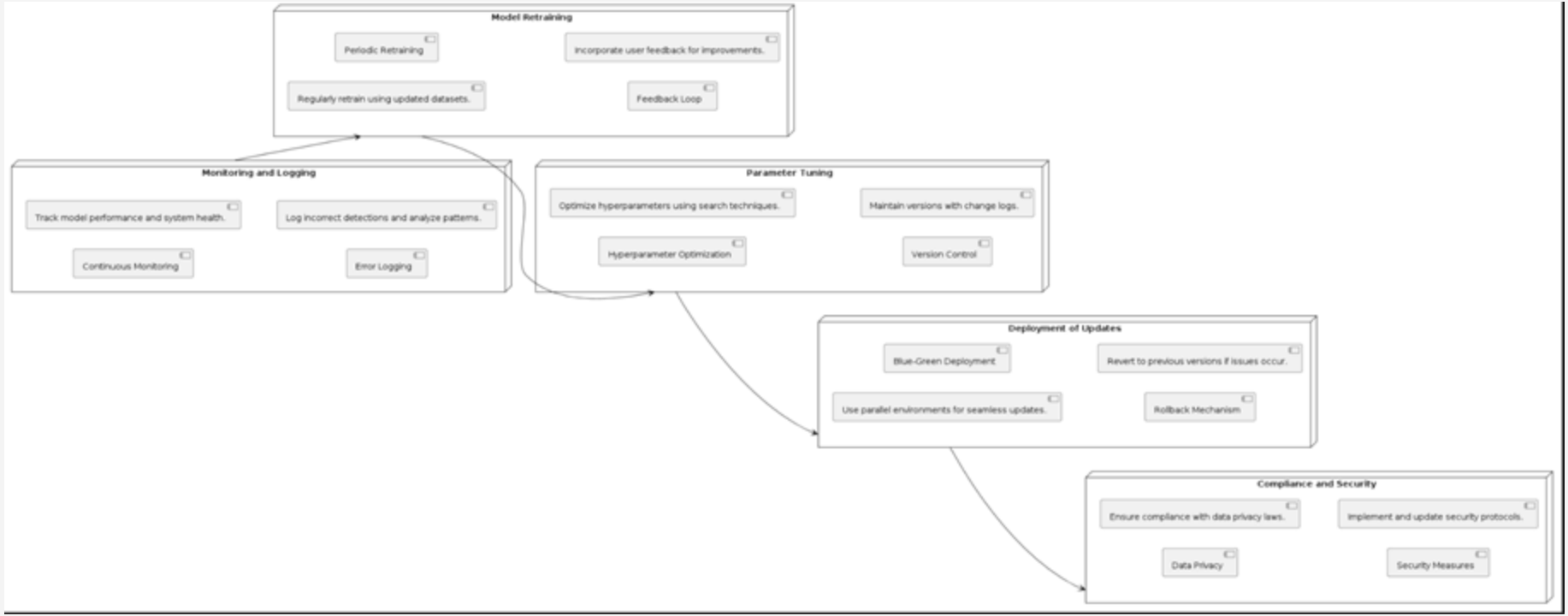
Class Naming Mismatch in Roboflow

Combine the pre-trained YOLO model and classification for cam streaming

# Model Operations - Deployment



# Model Operations - Maintenance



# Interesting Resources.

Youtube series for creating transfer-learning model using YOLO:

- https://www.youtube.com/watch?v=QtsI0TnwDZS

Building your own object detection model from scratch:

- https://pub.towardsai.net/building-your-own-object-detector-from-scratch-with-tensorflow-bfeadfaddad8