

Model Development Phase Template

Date	8 November 2024
Team ID	team - 740058
Project Title	Virtual Eye - Life Guard for Swimming Pools to Detect Active Drowning
Maximum Marks	10 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for the model, presented through respective screenshots.

Initial Model Training Code (5 marks):

```

[8] yolov5n.pt      100%[=====] 3.77M --.-KB/s  in 0.06s
2024-11-05 12:31:52 (64.5 MB/s) - 'yolov5n.pt' saved [3952441/3952441]

from ultralytics import YOLO

# Load the model using the correct filename
model = YOLO("yolov5n.pt") # Ensure this file is in the current directory

# Step 3: Verify successful loading
print("Model loaded successfully.")
# Train the model
results = model.train(data="/content/swimming-and-drowning-dataset/data.yaml", epochs=70, imgsz=640)

Class      Images  Instances  Box(P   R   mAP50  mAP50-95): 100%|██████████| 62/62 [00:17:00:00, 3.51it/s]

Epoch  GPU_mem  box_loss  cls_loss  dfl_loss  Instances  Size
63/70   2.22G    1.284    0.8229    1.397      25         640: 100%|██████████| 376/376 [01:50:00:00, 3.42it/s]
Class      Images  Instances  Box(P   R   mAP50  mAP50-95): 100%|██████████| 62/62 [00:17:00:00, 3.52it/s]

```

Model	Summary	Training and Validation Performance Metrics
--------------	----------------	--

Model Validation and Evaluation Report (5 marks):

YOLO (v5)

YOLOv5 is a fast and accurate object detection model that identifies and locates objects in images in real-time. It predicts both bounding boxes and class labels in a single pass, making it efficient for tasks like surveillance and drowning detection.

```
[8] yolo5n.pt 100%[=====] 3.77M --.-KB/s in 0.86s
2024-11-05 12:31:52 (64.5 MB/s) - 'yolo5n.pt' saved [3952441/3952441]

from ultralytics import YOLO

# Load the model using the correct filename
model = YOLO('yolo5n.pt') # Ensure this file is in the current directory

# Step 3: Verify successful loading
print('Model loaded successfully.')
# Train the model
results = model.train(data='/content/swimming-and-drowning-dataset/data.yaml', epochs=70, imgsz=640)

Class Images Instances Box(P R mAP50 mAP50-95): 100% [62/62 [00:17:00:00, 3.51it/s]

Epoch GPU_mem box_loss cls_loss dfl_loss Instances Size
63/70 2.22G 1.284 0.8229 1.397 25 640: 100% [376/376 [01:50:00:00, 3.42it/s]
Class Images Instances Box(P R mAP50 mAP50-95): 100% [62/62 [00:17:00:00, 3.51it/s]

70 epochs completed in 2.600 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 5.39B
Optimizer stripped from runs/detect/train/weights/best.pt, 5.39B

Validating runs/detect/train/weights/best.pt...
Ultralytics 8.3.27 Python 3.10.12 torch 2.5.0+cu121 CUDA:0 (Tesla T4, 15102MB)
YOLOv5n summary (fused): 193 layers, 2,590,539 parameters, 0 gradients, 7.1 BFLOPs
Class Images Instances Box(P R mAP50 mAP50-95): 100% [62/62 [00:20:00:00, 2.98it/s]
all 1977 3624 0.833 0.781 0.838 0.489
Drowning 1957 3138 0.856 0.847 0.901 0.567
Swimming 555 1377 0.886 0.838 0.85 0.471
out of water 60 117 0.778 0.658 0.762 0.448
Speed: 0.3ms preprocess, 2.2ms inference, 0.0ms loss, 2.1ms postprocess per image
Results saved to runs/detect/train

[12] model.val(data = '/content/swimming-and-drowning-dataset/data.yaml')
0.64565, 0.64665, 0.64785, 0.63865, 0.65165, 0.65265, 0.65365, 0.65465, 0.65566, 0.65666, 0.65766,

names: ['Drowning', 'Swimming', 'out of water']
results_dict: {'metrics/precision(B)': 0.8123269156632354, 'metrics/recall(B)': 0.7835984545583999, 'metrics/mAP50(B)': 0.8368448733853998,
'metrics/mAP50-95(B)': 0.4881379061238458, 'fitness': 0.5230806518508712}
save_dir: /path/to/runs/detect/all
speed: {'preprocess': 0.30839897831380333, 'inference': 4.432705597981215, 'loss': 0.0013960857462379869, 'postprocess': 1.486679088646827}
task: 'detect'

from ultralytics import YOLO

# Load the model using the correct filename
model = YOLO('/content/runs/detect/train/weights/best.pt') # Ensure this file is in the current directory

model.predict('/content/test/images/-Clipchamp-mp4-11.jpg.rf.8c2b48fcd8dc1741baded48d49546814.jpg', save=True, imgsz=320, conf=0.5)

keypoints: None
masks: None
names: {'Drowning': 1, 'Swimming': 2, 'out of water': 3}
obb: None
orig_img: array([[[[186, 185, 187],
[185, 184, 186],
[180, 182, 184],
...,
[157, 149, 112],
[155, 147, 118],
[169, 161, 94],
[169, 161, 94],
[169, 161, 94]],
[[[166, 159, 90],
[166, 159, 90],
[166, 159, 90],
...,
[169, 161, 94],
[169, 161, 94],
[169, 161, 94]],
[[[166, 159, 90],
[166, 159, 90],
[166, 159, 90],
...,
[169, 161, 94],
[169, 161, 94],
[169, 161, 94]],
[[[169, 161, 94],
[169, 161, 94],
[169, 161, 94]], dtype=uint8)
orig_shape: (640, 640)
path: '/content/test/images/-Clipchamp-mp4-11.jpg.rf.8c2b48fcd8dc1741baded48d49546814.jpg'
probs: None
save_dir: 'runs/detect/predict'
speed: {'preprocess': 1.3511188877685547, 'inference': 17.46511459356986, 'postprocess': 4.098176956176758}]
```