

## Model Optimization and Tuning 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

### Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining the YOLOv5 model to improve its performance for detecting drowning incidents in swimming pools. This phase focuses on adjusting hyperparameters, fine-tuning the model architecture, and enhancing the dataset to achieve the best accuracy, speed, and generalization.

### Hyperparameter Tuning Documentation (8 Marks):

Model	Tuned Hyperparameters(Epochs)																																																																																																		
YOLOv5	<p>In YOLOv5, <b>epochs</b> refer to the number of times the entire training dataset is passed through the model. The right number of epochs is crucial to avoid underfitting (too few epochs) or overfitting (too many epochs). Typically, YOLOv5 training runs for 100 to 300 epochs, but this depends on the dataset and hardware. The model can use <b>early stopping</b> to halt training when performance on the validation set stops improving, preventing overfitting. The optimal number of epochs helps achieve the best balance between training time and model accuracy.</p> <pre>## 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)  Logging results to runs/detect/train Starting training for 70 epochs...</pre> <table><thead><tr><th>Epoch</th><th>GPU_mem</th><th>box_loss</th><th>cls_loss</th><th>dfl_loss</th><th>Instances</th><th>Size</th></tr></thead><tbody><tr><td>1/70</td><td>2.28G</td><td>1.777</td><td>2.79</td><td>1.757</td><td>57</td><td>640: 100% [0:15:00:00, 2.77it/s]</td></tr><tr><td>Class</td><td>Images</td><td>Instances</td><td>Box(P</td><td>R</td><td>mAP50</td><td>mAP50-95): 100% [0:21:00:00, 2.86it/s]</td></tr><tr><td>all</td><td>1977</td><td>3624</td><td>0.65</td><td>0.202</td><td>0.235</td><td>0.0944</td></tr></tbody></table> <table><thead><tr><th>Epoch</th><th>GPU_mem</th><th>box_loss</th><th>cls_loss</th><th>dfl_loss</th><th>Instances</th><th>Size</th></tr></thead><tbody><tr><td>2/70</td><td>2.27G</td><td>1.784</td><td>2.287</td><td>1.784</td><td>63</td><td>640: 100% [0:08:00:00, 2.94it/s]</td></tr><tr><td>Class</td><td>Images</td><td>Instances</td><td>Box(P</td><td>R</td><td>mAP50</td><td>mAP50-95): 100% [0:19:00:00, 3.21it/s]</td></tr><tr><td>all</td><td>1977</td><td>3624</td><td>0.649</td><td>0.253</td><td>0.27</td><td>0.109</td></tr></tbody></table> <table><thead><tr><th>Epoch</th><th>GPU_mem</th><th>box_loss</th><th>cls_loss</th><th>dfl_loss</th><th>Instances</th><th>Size</th></tr></thead><tbody><tr><td>3/70</td><td>2.28G</td><td>1.81</td><td>2.185</td><td>1.813</td><td>30</td><td>640: 100% [0:09:00:00, 2.90it/s]</td></tr><tr><td>Class</td><td>Images</td><td>Instances</td><td>Box(P</td><td>R</td><td>mAP50</td><td>mAP50-95): 100% [0:19:00:00, 3.12it/s]</td></tr><tr><td>all</td><td>1977</td><td>3624</td><td>0.625</td><td>0.239</td><td>0.265</td><td>0.105</td></tr></tbody></table> <table><thead><tr><th>Epoch</th><th>GPU_mem</th><th>box_loss</th><th>cls_loss</th><th>dfl_loss</th><th>Instances</th><th>Size</th></tr></thead><tbody><tr><td>4/70</td><td>2.24G</td><td>1.780</td><td>2.192</td><td>1.780</td><td>32</td><td>640: 100% [0:09:00:00, 2.93it/s]</td></tr></tbody></table>	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	1/70	2.28G	1.777	2.79	1.757	57	640: 100% [0:15:00:00, 2.77it/s]	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% [0:21:00:00, 2.86it/s]	all	1977	3624	0.65	0.202	0.235	0.0944	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	2/70	2.27G	1.784	2.287	1.784	63	640: 100% [0:08:00:00, 2.94it/s]	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% [0:19:00:00, 3.21it/s]	all	1977	3624	0.649	0.253	0.27	0.109	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	3/70	2.28G	1.81	2.185	1.813	30	640: 100% [0:09:00:00, 2.90it/s]	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% [0:19:00:00, 3.12it/s]	all	1977	3624	0.625	0.239	0.265	0.105	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	4/70	2.24G	1.780	2.192	1.780	32	640: 100% [0:09:00:00, 2.93it/s]
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size																																																																																													
1/70	2.28G	1.777	2.79	1.757	57	640: 100% [0:15:00:00, 2.77it/s]																																																																																													
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% [0:21:00:00, 2.86it/s]																																																																																													
all	1977	3624	0.65	0.202	0.235	0.0944																																																																																													
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size																																																																																													
2/70	2.27G	1.784	2.287	1.784	63	640: 100% [0:08:00:00, 2.94it/s]																																																																																													
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% [0:19:00:00, 3.21it/s]																																																																																													
all	1977	3624	0.649	0.253	0.27	0.109																																																																																													
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size																																																																																													
3/70	2.28G	1.81	2.185	1.813	30	640: 100% [0:09:00:00, 2.90it/s]																																																																																													
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% [0:19:00:00, 3.12it/s]																																																																																													
all	1977	3624	0.625	0.239	0.265	0.105																																																																																													
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size																																																																																													
4/70	2.24G	1.780	2.192	1.780	32	640: 100% [0:09:00:00, 2.93it/s]																																																																																													

### Final Model Selection Justification (2 Marks):

Final Model	Reasoning
YOLOv5	<p>The <b>YOLOv5 model</b> was chosen as the final optimized model for the Virtual Eye Lifeguard project due to several key advantages that make it highly suitable for real-time drowning detection in swimming pools:</p> <ol style="list-style-type: none"> <li><b>Real-Time Object Detection:</b> YOLOv5 is designed for real-time object detection, enabling the system to quickly and accurately identify drowning incidents as they occur, which is crucial for timely intervention.</li> <li><b>Speed and Efficiency:</b> YOLOv5 processes images in a single pass, making it significantly faster compared to other models, such as CNNs or RNNs. This efficiency is vital for the project, where quick detection and response are necessary for safety.</li> <li><b>High Accuracy:</b> YOLOv5 consistently provides high detection accuracy by predicting both bounding boxes and class labels simultaneously. This is important for accurately identifying people and potential drowning events in complex environments like swimming pools.</li> <li><b>Scalability and Flexibility:</b> YOLOv5 can be easily trained on custom datasets (such as swimming and drowning images) and adapted to various object detection scenarios. It also supports multiple model sizes (small, medium, large) to balance between speed and accuracy based on system requirements.</li> <li><b>Pre-Trained Weights and Transfer Learning:</b> YOLOv5 provides pre-trained weights, which significantly reduce training time and enhance the model's ability to generalize, especially with limited data like that in your project.</li> <li><b>Ease of Use:</b> YOLOv5 is user-friendly, with robust community support, well-documented code, and integration with popular frameworks like PyTorch, making it easier to implement and finetune for the specific task of drowning detection.</li> </ol> <p>In conclusion, YOLOv5 was selected as the final model because it meets the project's requirements for <b>real-time detection, speed, accuracy, and ease of implementation</b>, making it the most suitable choice for the Virtual Eye Lifeguard system.</p>