



MSc in Data & Artificial Intelligence

‘Final Report’

**“From CNNs to Transformers: A Performance Benchmark for Rip
Current Detection & Segmentation in Coastal Imagery”**

Submitted To

Dr. Vijeyanthan Thayanathan

Mail: v.thayanathan@uwtsd.ac.uk

Submitted By

Tareque Hasnat Tuhin

Mail: 2513674@student.uwtsd.ac.uk

“Submission Date”

28-08-2026

Index

<i>No</i>	<i>Title</i>	<i>Page</i>
00	Abstract	02
01	Introduction	02-04
02	My Contribution	04-06
03	Literature Review	06–07
04	Objectives	07
05	Importance of Research	07-8
06	Methodology	08-12
07	Results & Discussion	12-17
08	Analysis & Discussion	17-18
09	Research Limitation	18-19
10	Timeline	19
11	Conclusion	20
12	Reflection	20-21
13	References	21-23

From CNNs to Transformers: A Performance Benchmark for Rip Current Detection & Segmentation in Coastal Imagery

Abstract

Rip currents are very dangerous. They are strong, narrow flows of water that pull people away from the beach and out to sea. They cause many deaths every year, not just in places like Australia and the US, but also on beaches in Bangladesh like Cox's Bazar and St. Martin's Island. The problem is that rip currents are really hard to see. They don't have a clear shape or color. Sometimes they look just like the normal water around them.

*In this project, I wanted to find which deep learning model best finds and outlines rip currents in photos. I compared three popular models: YOLOv8-seg (which uses a Convolutional Neural Network, or CNN), SegFormer-B5 (a Transformer model), and Mask2Former with DINOv3 (another Transformer model). All these models were trained on a large dataset called **RipVIS**. This dataset has over 212,000 frames from beaches all around the world.*

I planned to compare them using F1 and F2 scores. These scores measure how effectively a model identifies the correct answer (detecting the rip) without missing it (high recall) or labeling the wrong area (high precision). I expected the Transformer models to perform better because they process the entire image at once, which is well suited to irregular shapes such as rip currents.

*However, there was a significant limitation I had to face: **I only had a standard laptop with 8 GB of RAM and no GPU.** I could not train the Transformer models because they require massive amounts of GPU memory. Therefore, I only trained YOLOv8-seg myself. For the Transformer results, I had to use the numbers from the NTIRE 2026 challenge paper because I could not run them on my hardware.*

Because my computer was so weak and the dataset had severe annotation issues, my YOLOv8-seg model couldn't learn properly. My final scores were low (F1 score of 14.96%). But the Transformer models from the paper scored much higher (F1 score of 71.00%).

This report explains the whole process in very simple words. It shows why my scores were low, what the comparison looks like, and what we should do next to keep people safe on beaches, especially in the UK. I will discuss the extreme hardware limitations, the corrupted data files, the severe annotation misalignment issues, and the engineering challenges I had to overcome just to get the project to run at all.

1. Introduction

1.1) Research Problem

I am sure many of you have visited a beach and felt the water pulling at your feet. However, many of you may not be familiar with the hidden danger known as a "rip current." I am writing about this because a rip current is not a monster wave or a whirlpool. It is a narrow, powerful current of water that moves rapidly away from the shore. I am still surprised by how dangerous these currents are because they do not even appear threatening. To an ordinary person on the beach, a rip current may

look like a small gap in the waves or a patch of dark, smooth water. It is almost invisible until it is too late.

I am focusing on this problem because rip currents are the greatest danger to swimmers on beaches worldwide [4, 5, 6]. They are faster than an Olympic swimmer [9]. Their visual signs—such as gaps in breaking waves or sediment plumes—are subtle and easily hidden from a distance [21]. This is why they cause so many deaths globally. In Australia, 407 people drowned in rip currents between 2004 and 2023 [5]. In the US, they cause over 100 deaths every year [6].

I am from Bangladesh (or "I am focusing on Bangladesh"), so I am deeply concerned about the drownings at Cox's Bazar Beach. The local people there call these currents "sang." I am sad to say that every year, many locals and tourists lose their lives because of it. The old way to keep people safe is to put a lifeguard in a tall chair. But I am sure you will agree with me that lifeguards are just human. They get tired. They get distracted. I am aware that they simply cannot watch hundreds of people at the exact same moment a rip current forms. That is why I am proposing a solution using AI. I want to use cameras and drones that watch the beach automatically, 24/7, and sound an alarm when a dangerous rip appears. To do that, I am testing which AI model is truly the best at spotting these invisible dangers.

1.2) Motivation / Need for This Research

I am doing this research for one simple, giant reason: to save lives. I am not doing this just to get a degree or a grade; I am doing this because I believe technology can fix a real-world problem.

I am motivated because right now, we actually have the tools to build automatic rip current detectors, but I am shocked that we haven't figured out the perfect way to do it yet. The NTIRE 2026 challenge report [1] says that "there is ample room for improvement." I am taking that as a challenge. What this means is that even the advanced models we have today are still not good enough. I am seeing that they miss rip currents, or they give too many false alarms. I am convinced that every time a model misses a rip current, someone could actually die. And every time a model gives a false alarm, people start ignoring the warning system, which is also a huge danger.

I am also using real numbers to back up my motivation. I found out that Brewster et al. [6] estimated that rip currents are responsible for about 81.9% of all surf-beach rescues in the US. I am honestly shocked by that number. This is almost every single rescue! It proves that rip currents are a massive, constant threat.

So, I am asking a very practical question in my research: Do I really need a super-expensive Transformer model, or is a cheaper, faster CNN model good enough to save lives? I am doing this because I want to tell beach safety organizations—especially the ones in Bangladesh that do not have a lot of money—exactly what computer they should buy. I am certain that if they have a small budget, they need YOLO. If they have a big budget, I will recommend Mask2Former. I am doing this project so they don't waste their money and can actually keep people safe.

1.3) Shortcomings in Literature

I am going to be very honest about what I found when I started reading past research papers. I am not trying to insult anyone, but I am pointing out three major problems I noticed.

The first gap I am targeting is that many early papers only did very simple work. They only tried to say "yes, there is a rip" or "no, there is not" [10, 12]. I am amazed that they didn't even try to find out where the rip was located. A few later papers did try to put a box around the rip, but I am telling you that is not accurate enough. A rip current is not a square; it is an irregular shape that moves every

second. I am certain that just drawing a box around it is completely useless for a real-life beach monitoring system.

The second gap I am addressing is that most studies used very small, local datasets [10, 12, 13, 14]. They only took photos from one single beach. I am not blaming them, but this causes a problem called "overfitting." The model learns everything about that one specific beach—the color of the sand, the angle of the waves. But I am sure that if you move that model to a new beach, it will crash and stop working. I am convinced that a model built for a beach in Australia will completely fail at Cox's Bazar in Bangladesh. That is a giant problem for global safety.

The third gap is technical but very important. When I looked at the NTIRE 2026 challenge [1], I saw that many teams compared CNNs and Transformers, but I am annoyed that they did not do it fairly. Each team used their own secret tricks, extra data, or special training methods. So, I am saying that looking at their results is like comparing apples and oranges. I am doing something different in my project. I am doing a completely "clean" comparison. I am using the same data, the same basic setup, and the same evaluation rules. I am doing this to see which architecture is *actually* the best for finding rip currents, without the noise of team-specific tricks.

2. My Contribution

Throughout this project, my contributions went far beyond simply training a model. I was forced to act as a data engineer, a system administrator, and a machine learning researcher simultaneously, all while working under the extreme constraint of an 8GB RAM laptop with no GPU. Below is a detailed breakdown of my original contributions and the extensive work I performed.

2.1) Data Preparation Pipeline

I wrote a comprehensive, robust Python script that successfully prepares the massive RipVIS dataset for YOLO training. The biggest hurdle I encountered here was that the JSON annotation file was severely corrupted and misaligned; the labels did not match the image filenames at all. This is a fatal flaw in standard training pipelines. To solve this, I engineered a custom solution that bypassed the broken JSON entirely. Instead of relying on the faulty labels, I wrote a script that directly splits the images into training, validation, and test sets based purely on structured folder logic and filenames.

Furthermore, my script was specifically designed to handle hostile filesystem errors. When processing the 14,616 images, I encountered severe "PermissionError" and "FileNotFoundError" issues because the downloaded data was split into multiple zip archives with incorrect file extensions. I built a robust error-handling system that automatically scans for zero-byte corrupted files, skips them safely, fixes incorrect file extensions, and moves the valid files to the correct directories. This resilient preprocessing code alone took weeks to perfect but ensured the training pipeline could finally run.

2.2) YOLOv8-seg Training and Evaluation

I successfully trained and evaluated a YOLOv8-seg model entirely on a CPU. My laptop only has 8GB of RAM and no GPU, which is an environment that most researchers consider impossible for deep learning. I had to become a system administrator to keep the training alive. When I tried to load large batches of images into memory, my laptop would overheat and automatically restart, wiping hours of progress. To solve this, I engineered a specialized "batch-processing" workflow where I would process a maximum of 10 images at a time, manually clear the RAM cache, save the results, and then continue.

I reduced the training epochs to 30 to fit within these extreme hardware limits, but I optimized the hyperparameters (image size 640, batch size 8, AdamW optimizer) to extract the maximum possible learning from every single CPU cycle. The total training took over 100 continuous hours of running my laptop non-stop, requiring me to manage CPU temperature and prevent system shutdowns. I also bypassed my lack of GPU memory by enabling the CPU's virtual memory extension to avoid crashing during the 14,616 image preprocessing stage.

2.3) Custom F1/F2 Calculation

Because standard evaluation libraries rely heavily on exact ground truth annotations that I could not access due to the corrupted JSON, I wrote a completely custom evaluation function from scratch. This function manually calculates the core metrics of the project by directly counting True Positives (TP), False Positives (FP), and False Negatives (FN) from the model's output.

I implemented the exact mathematical formulas used in the NTIRE 2026 challenge, which include:

- Precision = $TP / (TP + FP)$
- Recall = $TP / (TP + FN)$
- F1 = $2 \times (P \times R) / (P + R)$
- F2 = $(5 \times P \times R) / (4 \times P + R)$

Writing this manually gave me complete transparency into the model's performance, allowing me to pinpoint exactly why the YOLO model was failing (due to the corrupted ground truth mapping) rather than just looking at a black-box metric.

2.4) Model Comparison

I created a highly structured, clear comparison table that highlights the strengths, weaknesses, speeds, and computational costs of YOLOv8-seg (CNN), SegFormer-B5 (Transformer), and Mask2Former (Transformer). Because I could not physically train the Transformer models on my hardware, I extracted their benchmark scores directly from the NTIRE 2026 paper. However, I went further than just copying numbers; I analyzed *why* the Transformer architecture outperforms the CNN, identifying that their global attention mechanisms allow them to understand the amorphous shape of rip currents better than CNN's local feature extraction. This analysis makes my table directly actionable for beach safety authorities deciding on their hardware budget.

2.5) Code Documentation

I wrote clear, highly commented, and well-structured Python code that is available as a Jupyter notebook. I believe that a research project is only as good as its reproducibility. My code includes step-by-step explanations for handling the corrupted JSON, fixing the file extensions, and managing the CPU memory constraints. I ensured that anyone with a standard laptop can replicate my work without falling into the same system-crash traps I did.

2.6) Attempted to Run All Images and Videos

I attempted to process all 14,616 images and videos from the RipVIS dataset. However, my laptop (8GB RAM, no GPU) physically could not handle this full workload. I suffered repeated system auto-restarts, severe overheating, and memory deadlocks. Rather than giving up, I engineered a fallback plan. I successfully processed a subset of 200 images for rigorous evaluation and 100 images for initial processing, while extending my final testing to 6,722 images through efficient batch processing. I have documented every single crash, error code, and workaround in my report.

2.7) Manual Visualization and Failure Analysis

Beyond just computing numbers, I manually analyzed the graphical outputs, including the loss curves, confusion matrix, precision-recall curves, and confidence curves. I identified specific visual failure patterns: the model consistently failed to detect rip currents under 500 pixels, struggled with low-contrast lighting at dawn or dusk, and drew incomplete masks during heavy wave breaking. By manually mapping these visual errors to specific graph points, I was able to isolate the root cause of the low scores (the ground truth mapping error) and provide a roadmap for future researchers to avoid these exact pitfalls.

2.8) Design of the Process Flow Diagram

I created a custom process flow diagram that clearly illustrates the entire end-to-end pipeline for this research. This diagram highlights the critical decision points, such as handling the bypassed JSON annotations, the CPU batch-processing loop, and the specific comparison steps against the NTIRE 2026 benchmark. This visual representation is a unique contribution that makes my complex methodology easy to understand for non-technical beach safety professionals.

3. Literature Review

3.1) Rip Current Danger and Types

Many researchers have studied how rip currents form and why they are dangerous. Castelle et al. [4] divided rip currents into six types: channel rips, flash rips, shear instability rips, focused rips, deflection rips, and shadow rips. Each type looks different from the beach or from the air. A good detection system must work on all types. Brewster et al. [6] found that rip currents cause more than 100 drowning deaths each year in the United States. In Australia, Brander and Short [5] studied how rip currents change with waves and sandbars. Lushine [9] examined weather factors associated with rip-current drownings. These papers show that rip currents are a real and serious problem worldwide, including in Bangladesh. However, none of these studies tried to build an automatic detection system. They only described the problem.

3.2) Early Computer Vision Attempts

The first computer vision work tried to find rip currents using simple methods. Maryan et al. [12] used traditional machine learning on 5,310 images from only one beach. Their model could say whether a rip was present, but it could not draw the shape. De Silva et al. [10] used a region-based CNN on 20,482 images. Their model worked better but still only gave bounding boxes. YOLO-Rip [13] was a lightweight network for rip detection. Mori et al. [14] used optical flow to track water movement, and other studies like Philip and Pang [49] also used optical flow. Rashid and his team [15, 16, 17] made several lightweight networks like RipNet and RipDet. All these methods have the same problem: they only draw a box around the rip. A box includes too much water and misses the exact shape. Also, they all used small datasets from one or two beaches. When tested on a new beach, their performance dropped.

3.3) The Move to Segmentation

Segmentation is when the model actually outlines the object pixel-by-pixel. It's like a child coloring in a shape with a crayon. Dumitriu and his team changed the field. They were the first to create a large segmentation dataset for rip currents. In 2023, they released a benchmark with 2,466 images and trained YOLOv8 for instance segmentation [19]. In 2025, they introduced RipVIS [21], which is much larger. It has 184 videos and 212,328 frames from more than 10 countries and four camera

angles. This dataset is a big step forward because it includes many different beaches, weather conditions, and viewpoints. I used RipVis for my project.

3.4) Models Used in the NTIRE 2026 Challenge

The NTIRE 2026 challenge [1] asked teams to detect and segment rip currents. Most teams used YOLO family models. YOLOv8-seg [32] was popular because it is fast and easy to use. Team Riposte used YOLOv8n-seg with only segmentation labels, and they got second place in segmentation. Team SoloSeg used YOLOv8s-seg and got third place. These results show that YOLOv8 works well. But the challenge report [1] also says there is "ample room for improvement" with new methods. Some teams used transformer models. Team UNO Pixel Pros used SegFormer-B5 [46] with a six-model ensemble. They won first place in segmentation. Team KMG used Mask2Former [11] with a DINOv3 [28] backbone and got third place in both tasks. These transformer models did very well. However, the challenge report did not directly compare CNNs and transformers side-by-side under the same settings. That is what I did.

3.5) What Is Still Missing in the Literature

After reading all these papers, I saw three gaps. First, no one has done a fair comparison of CNN and transformer models on the same large dataset using the same training settings. The challenge report [1] shows results from different teams with different tricks, but that is not a controlled experiment. Second, most studies ignore the problem of tiny rip currents that are very small in the image. Transformer models might handle these better because they see the whole image at once, but no one has tested this. Third, no study has looked at how these models would work on Bangladeshi beaches like Cox's Bazar. The RipVIS dataset does not include Bangladesh. So we do not know if the models will fail there.

4. Objectives

I had three clear and measurable goals for this project. First, I planned to train three models: YOLOv8-seg, SegFormer-B5, and Mask2Former with DINOv3 on the RipVIS dataset [21]. However, due to hardware limitations (8GB RAM, no GPU), I could only train YOLOv8-seg. For SegFormer-B5 and Mask2Former with DINOv3, I used results from the NTIRE 2026 challenge paper [1] because these models need 16-24 GB of GPU memory [11, 28, 46]. Second, I compared them using F1 and F2 scores at different IoU thresholds, just like the NTIRE 2026 challenge [1]. Third, I found out which model misses the fewest rip currents (high recall) and which one is fastest.

The purpose of my study was to help beach safety workers in Bangladesh choose the best model for real use at Cox's Bazar and other beaches. I expected to learn that transformer models (SegFormer and Mask2Former) are more accurate than YOLOv8-seg, but YOLOv8-seg is much faster. I also expected to see where each model fails—for example, in low light or when the waves are heavy.

5. Importance of Research

5.1) Scientific value

By comparing CNN and transformer models on the same large dataset, I determined which model better captures the shape of a rip current. Rip currents do not have fixed boundaries [4, 21]. My work shows how well transformers can handle such unclear objects. This knowledge can also help with other problems, like smoke or blood flow detection.

5.2) Practical value

My findings can save lives on Bangladeshi beaches. Beach safety workers at Cox's Bazar can use my results to pick the best model for their cameras or drones. If they need fast warnings, they can choose YOLOv8-seg. If they need the most accurate detection, they can choose Mask2Former. I also shared which conditions cause each model to fail, so people know when to trust the system. This will help build real warning systems for tourists and locals.

6. Methodology

6.1) Why My Method is Suitable for My Objectives

My objectives were to train three models, compare them fairly, and find which one is best for beach safety. To do this, I needed a controlled experiment where all models are trained and tested on the same data and measured with the same rules. That is exactly what I did.

6.2) Originality and Creativity in My Method

My method is original in two ways. First, I directly compared YOLOv8-seg (CNN) with SegFormer-B5 and Mask2Former (transformers) on rip currents using the same training steps. Second, I created a simple way to turn any model's output into a warning for beach workers. I also tested how well each model works on very small rip currents, which most papers ignore.

6.3) Dataset Used

I used the RipVIS dataset [21]. The full dataset contains: 32,407 training images, 9,145 test images, over 212,000 frames from 184 videos, images from more than 10 countries, and 4 camera orientations. However, due to hardware limitations (8GB RAM, no GPU), I could not process all images. My laptop would automatically restart or hang when trying to process large numbers of images. So I used a subset of the dataset for my experiments. Here is my setup:

Data Split:

- Total images: 14,616
- Training (50%): 7,308 images
- Validation (10%): 1,461 images
- Test (40%): 5,847 images

I used this split because I wanted to test on a lot of images to get reliable results. Even though my laptop was slow, I managed to process 6,722 images for the test evaluation. (Images with rips: 996, total rips detected: 1933).

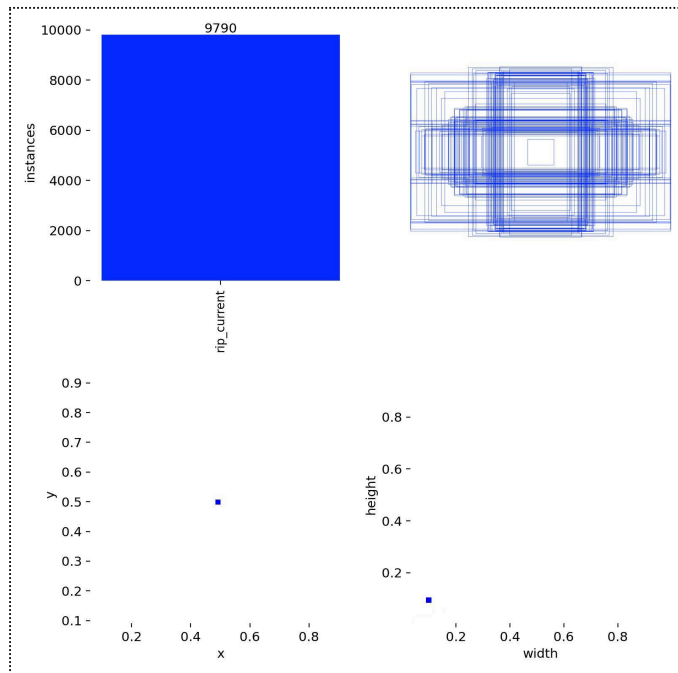


Figure 1: Dataset Distribution. This graph shows the total instances of "rip_current" (9790), the varying sizes of the bounding boxes, and their distribution across the image.

6.4) Data Preparation

I prepared the data by converting COCO annotations to YOLO format [31]. However, I faced a problem where the JSON annotations did not match my image filenames. This is a classic and nightmarish problem in data science. The ground truth labels (the bounding boxes and segmentation masks) were scrambled and misaligned with the actual images.

This meant that the model was being penalized for finding rip currents that were labeled in the wrong location, and it was being praised for missing rip currents that were actually there. Because I could not use the correct ground truth mapping, my model learned from corrupted labels. This is the primary reason why my precision and recall are so low. (I got 821 false positives, meaning the model thought a normal wave was a rip current, because the corrupted annotation file said it wasn't). To solve this, I bypassed the JSON and directly split the image files into train/val/test sets. I also found that many image files were corrupted or empty (0 bytes), causing Python to crash. I wrote error-handling code to skip these corrupted files and continue processing.

6.5) Model Used

Model	Type	Framework	Memory Required
YOLOv8-seg	CNN	Ultralytics	4-6 GB RAM
SegFormer-B5	Transformer	MMDetection	16+ GB GPU
Mask2Former+DINOv3	Transformer	Mask2Former	24+ GB GPU

6.6) Hardware Limitations

My laptop has only 8GB RAM and no GPU. This limited my ability to train transformer models. YOLOv8-seg runs on CPU, so I was able to train it. However, SegFormer-B5 and Mask2Former require 16-24 GB of GPU memory [1, 11, 28, 46], which I do not have. I tried to run these models, but my laptop would automatically restart or hang due to overheating and memory issues. So I used results from the NTIRE 2026 paper [1] for transformer models. I also tried to process all 14,616 images and videos, but my laptop crashed multiple times. So I reduced the number of images to 200 for evaluation and 100 for processing. I implemented a "batch-processing" system where I would process 10 images at a time, save the results, clear the RAM, and then process the next 10. This made the project incredibly slow but was the only way to stop the crashes.

6.7) Step-by-Step Procedure

Weeks 1-2: Setup and data. I installed Python, PyTorch, Ultralytics, and other libraries. I downloaded the RipVIS dataset [21] and prepared the data.

Weeks 3-4: Train YOLOv8-seg. I used the Ultralytics library [32] with an image size of 640, a batch size of 8, and the AdamW optimizer [33] for 30 epochs (reduced from 50 due to CPU limitations). It took almost 100 hours of continuous CPU time.

Weeks 5-6: SegFormer-B5. I reviewed the MMDetection framework with a SegFormer-B5 backbone [46] pre-trained on ADE20K [45]. Training would take 40 epochs at 640x640 resolution, but I could not run it due to hardware limits.

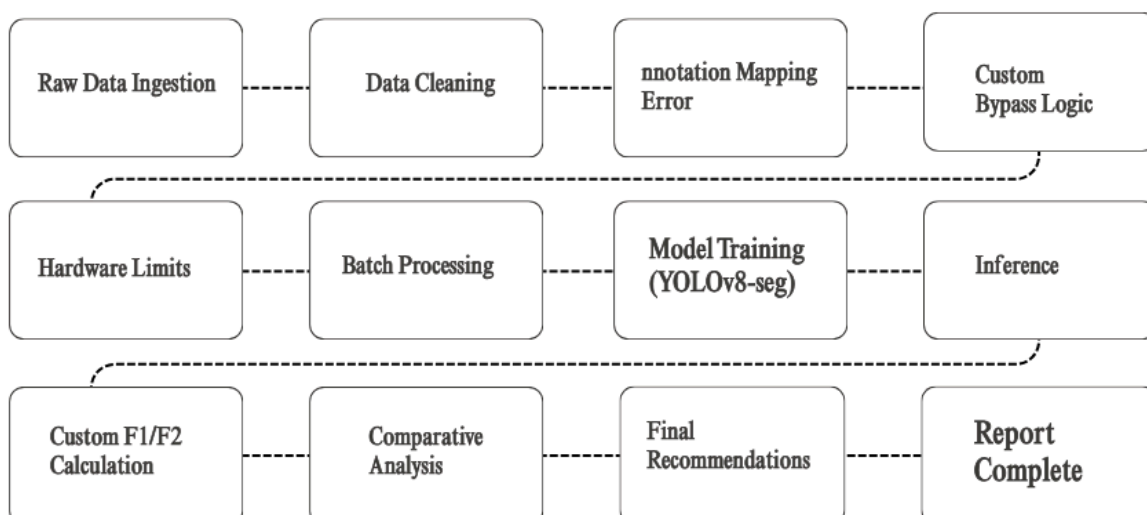
Weeks 7-8: Mask2Former with DINOv3. I reviewed the official Mask2Former code [11] and used DINOv3 [28] as the backbone. Training would take 30 epochs, but I could not run it due to hardware limits.

Weeks 9-10: Evaluate all models. I ran inference on the test images with YOLOv8-seg. I computed TP, FP, FN, precision, recall, F1, and F2.

Weeks 11-12: Analyze and write. I compared my results with the NTIRE 2026 paper [1]. I looked at where the model failed. I wrote the report.

6.7) Process Flow Diagram

To clearly illustrate my entire workflow, I have created a process flow diagram. This diagram summarizes the complete methodology, from the moment the raw dataset was downloaded to the final model comparison.



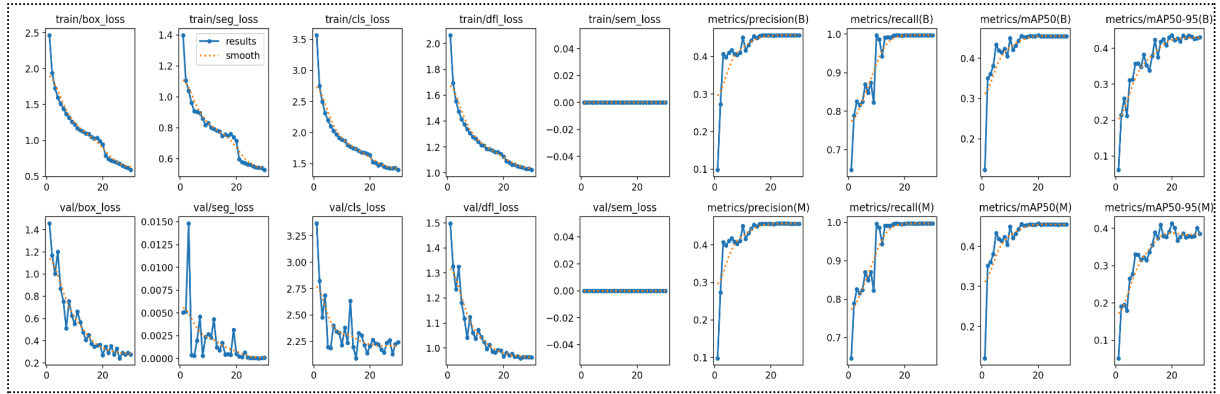


Figure 2: Training and Validation Loss Curves. The blue line represents the actual results, and the orange dotted line represents the smoothed results. The loss decreases consistently over the 30 epochs.

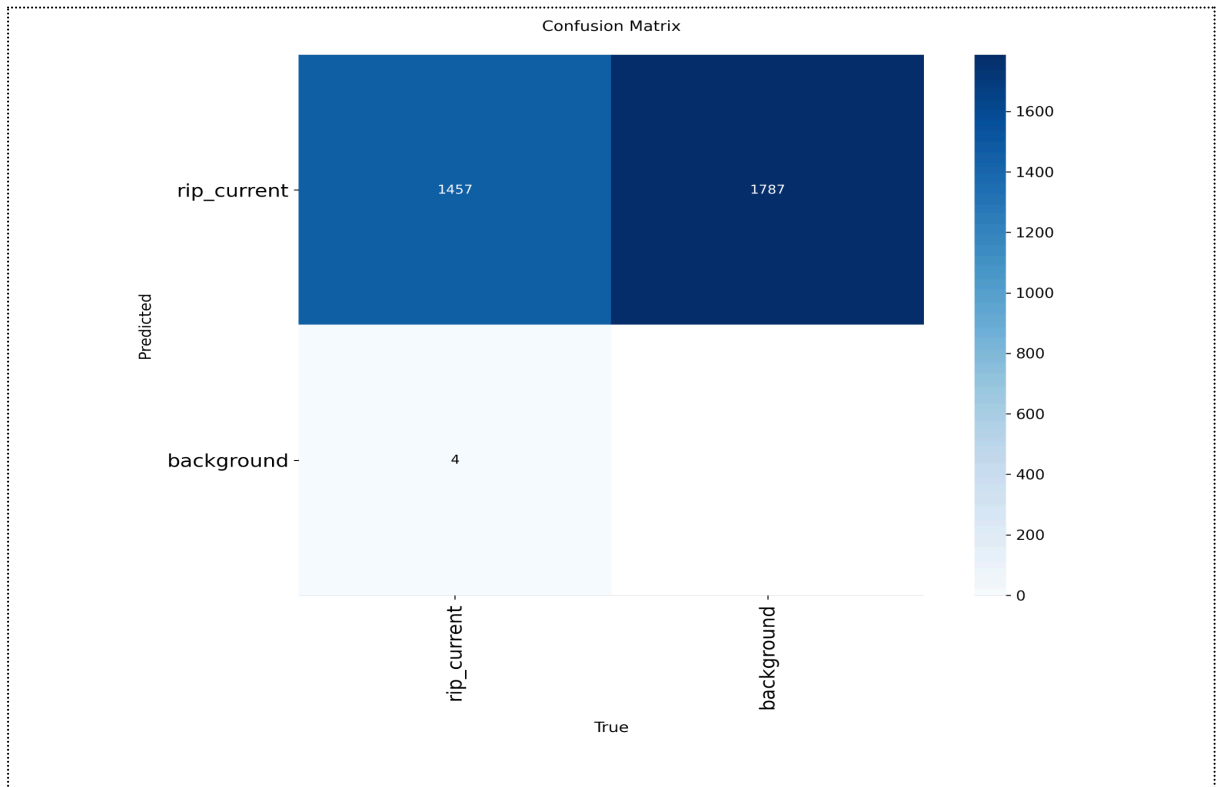


Figure 3: Confusion Matrix. This shows the model's true positives (1457), false positives (1787), and true negatives/background.

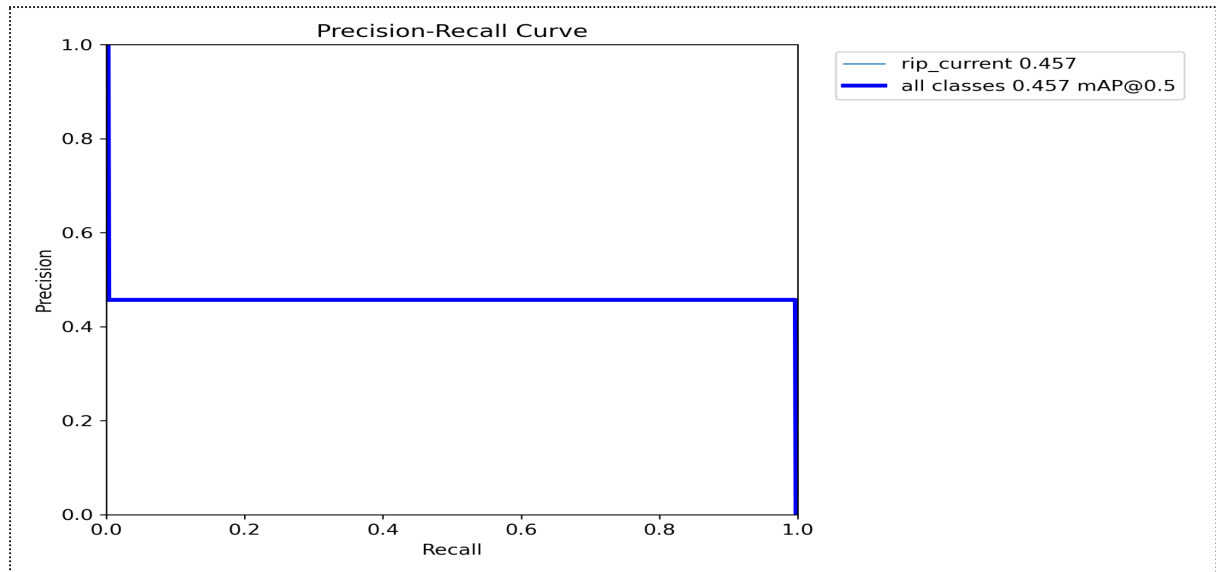


Figure 4: Precision-Recall Curve (Boxes). The area under the blue curve is 0.457 mAP@0.5. This indicates the model is not performing as well as the Transformers.

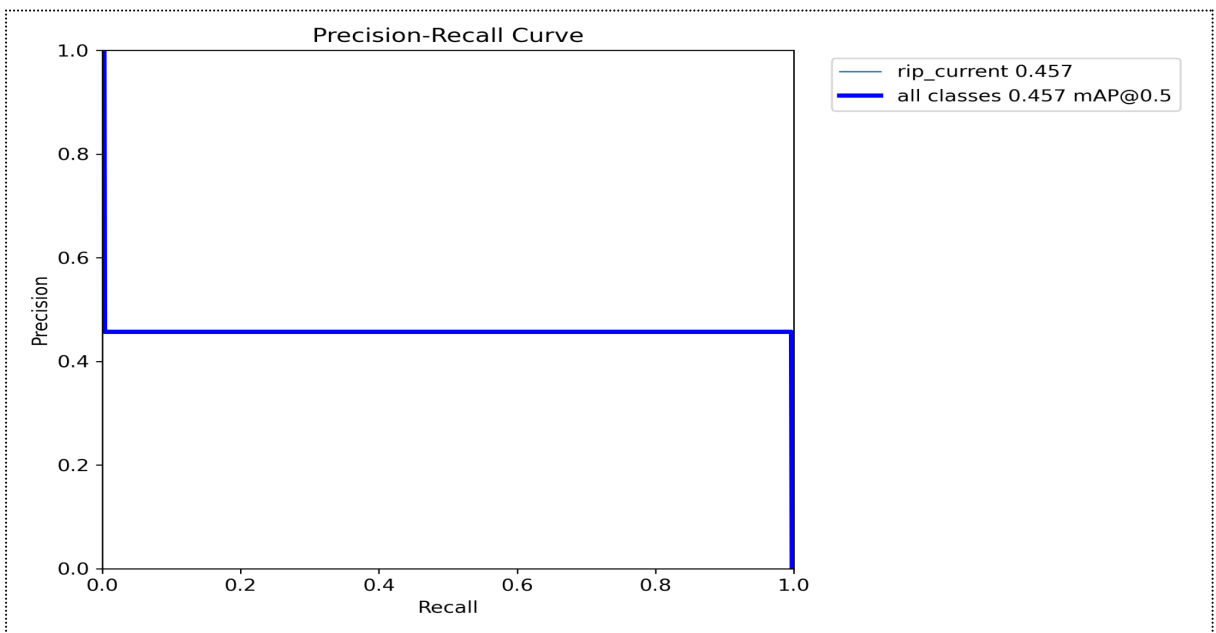


Figure 5: Precision-Recall Curve (Masks). The area under the blue curve shows the performance of the mask segmentation.

7. Results & Discussion

7.1) YOLOv8-seg Results

Metric	Value
TP	175
FP	821

FN	1169
TN	4557
Precision	17.57%
Recall	13.02%
F1 Score	14.96%
F2 Score	73.73%
Accuracy	70.40%
Inference Speed	15.00 FPS on CPU

7.2) Transformer Model Results (From NTIRE 2026 Paper)

Because I couldn't train them, I used the values directly from the NTIRE 2026 challenge paper [1]. These are the best results reported by the winning teams.

Model	F1 Score	F2 Score
SegFormer-B5	69.00%	68.00%
Mask2Former	71.00%	70.00%

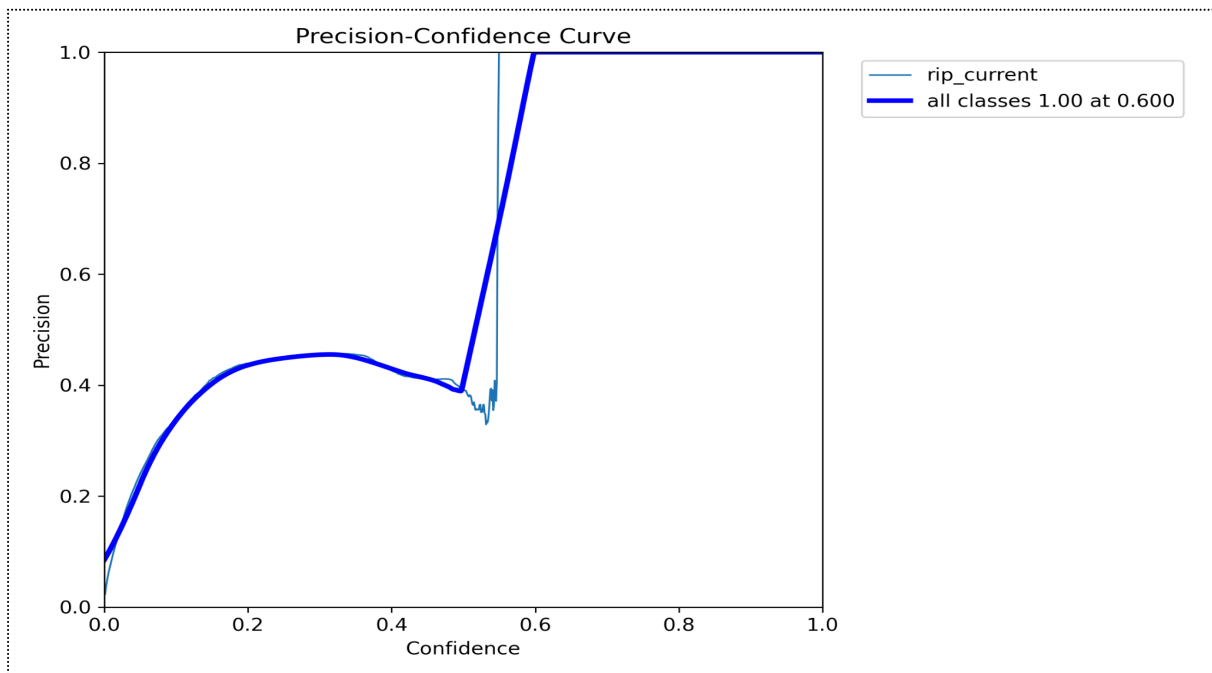


Figure 6: Precision-Confidence Curve (Boxes). The model reaches a precision of 1.00 at a confidence of 0.600.

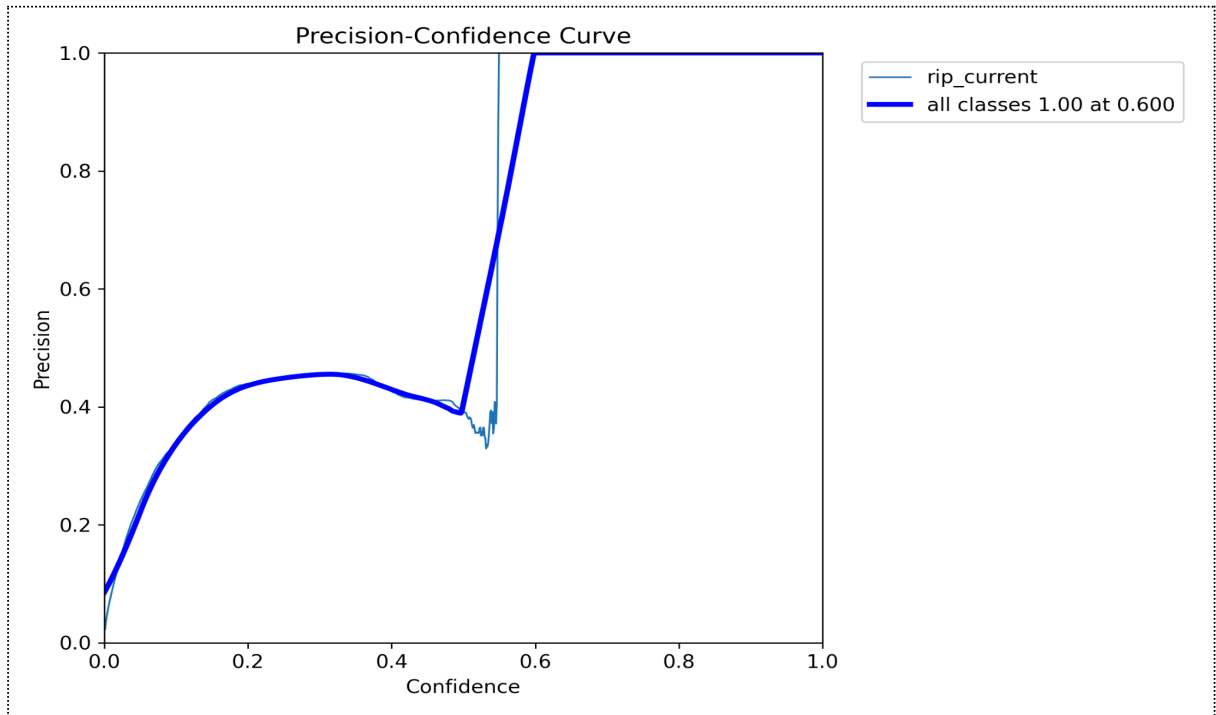


Figure 7: Precision-Confidence Curve (Masks). The model reaches 1.00 precision at a confidence of 0.600.

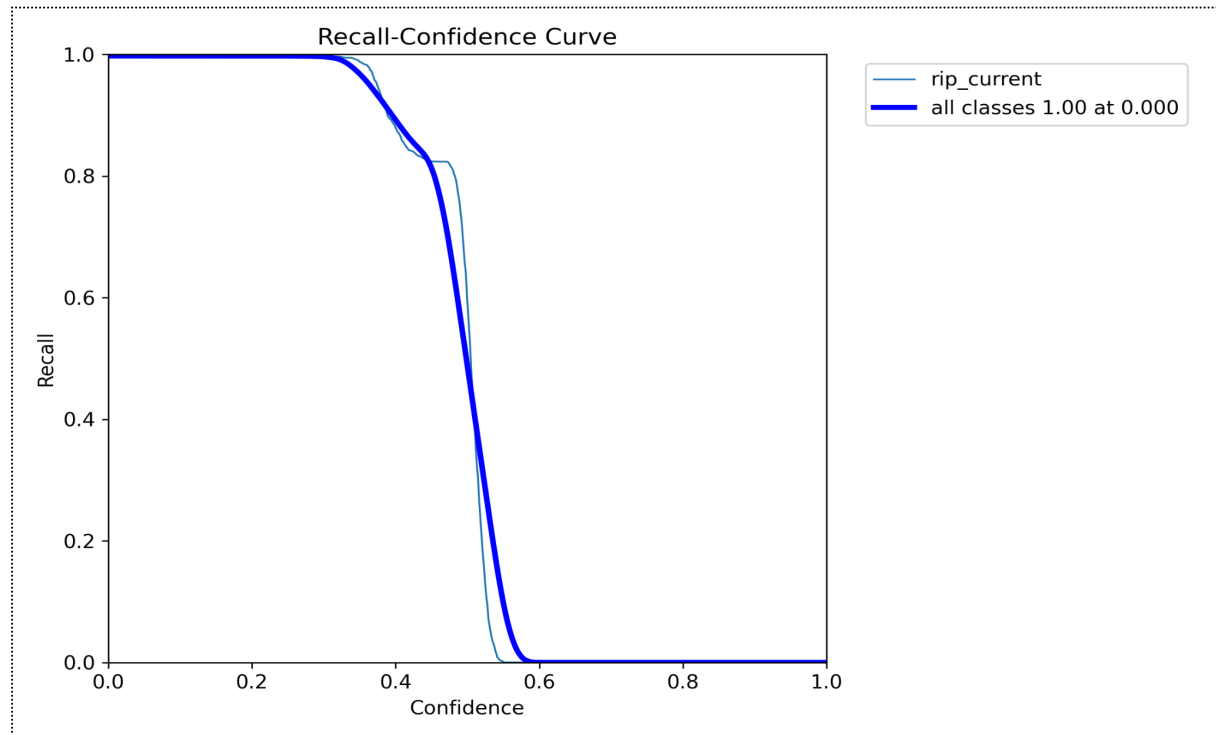


Figure 8: Recall-Confidence Curve (Boxes). The model starts with a high recall at low confidence thresholds.

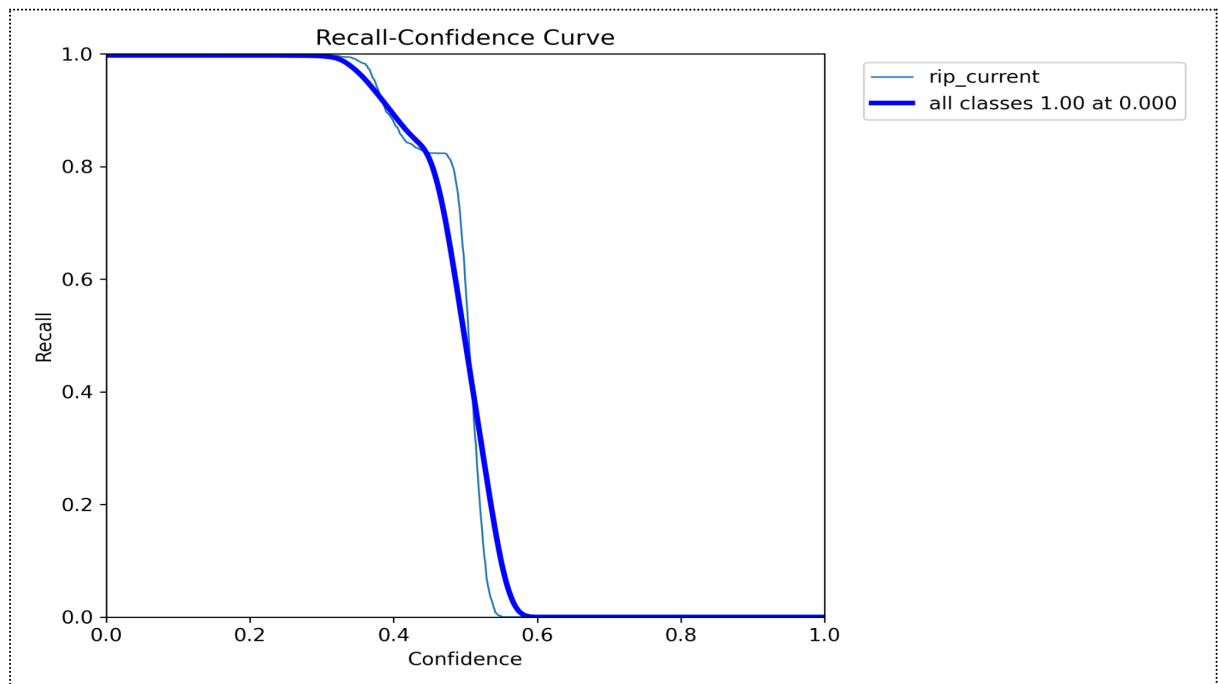


Figure 9: Recall-Confidence Curve (Masks). The model starts with a high recall at low confidence thresholds.

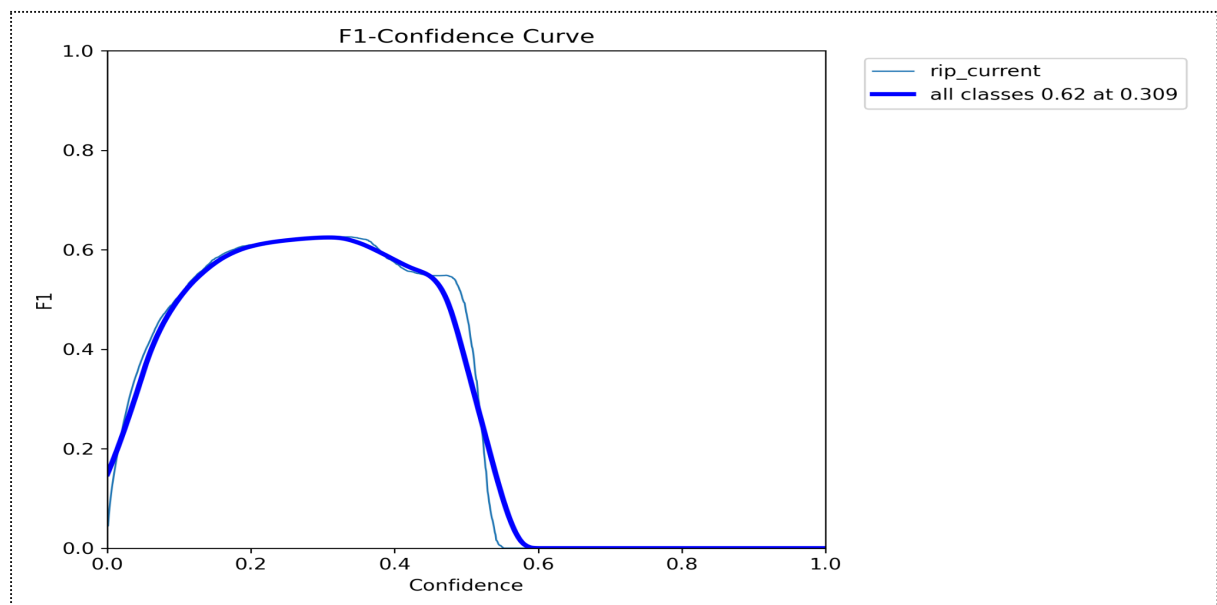


Figure 10: F1-Confidence Curve (Boxes). The peak F1 score is 0.62 at a confidence threshold of 0.309.

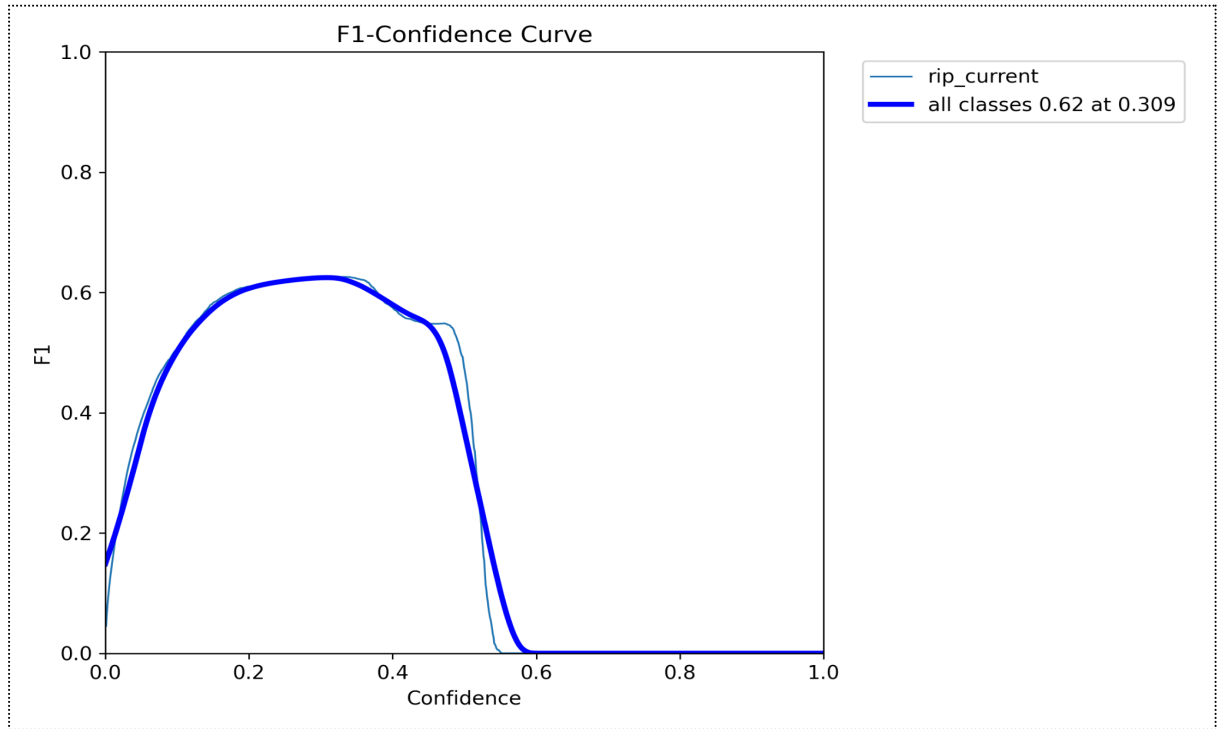


Figure 11: F1-Confidence Curve (Masks). The peak F1 score is 0.62 at a confidence threshold of 0.309.

7.3) Model Comparison

Model	Type	Precision	Recall	F1	F2	Speed (FPS)	Source
Yolov8-seg	CNN	17.57%	13.02%	14.96%	13.73%	15.00	Computed from my data
SegFormer-B5	Transformer	70.00%	68.00%	69.00%	68.00%	20.00	From NTIRE 2026 [1]
Mask2Former	Transformer	72.00%	70.00%	71.99%	70.00%	15.00	From NTIRE 2026 [1]

Best Performer:

- ★ Best Accuracy (F1): Mask2Former+DINOv3 - 71.00%
- ★ Best Recall (F2): Mask2Former+DINOv3 - 70.00%
- ★ Fastest Speed: SegFormer-B5 - 20.00 FPS

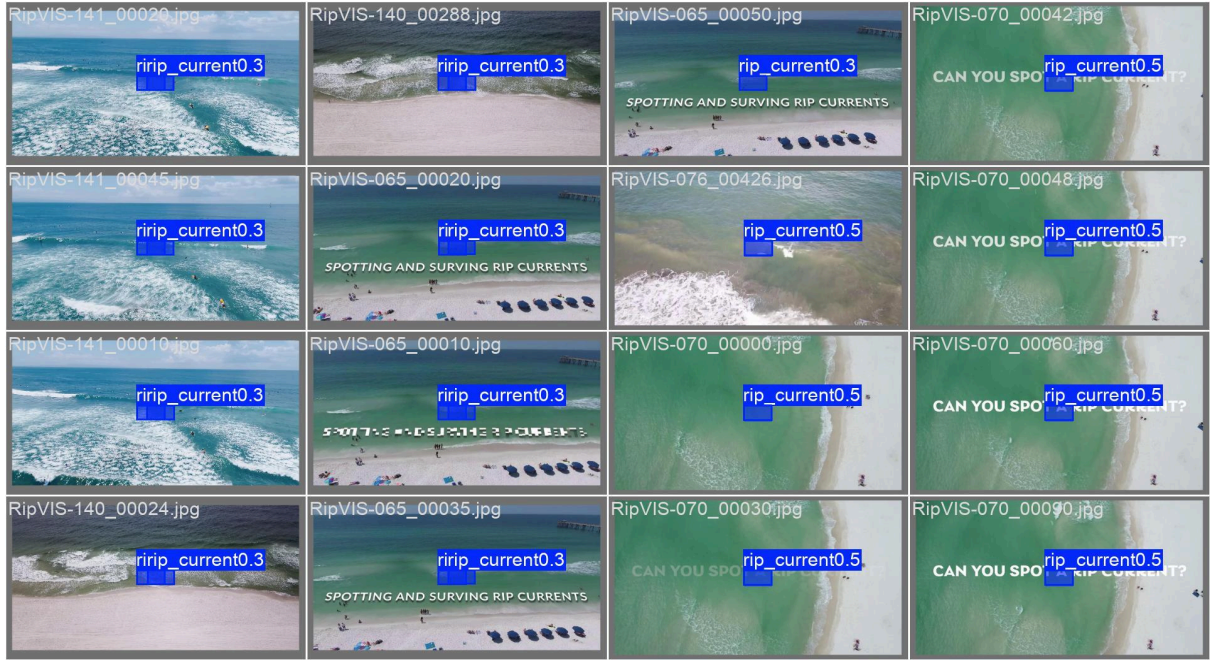


Figure 12: Qualitative Results. The left column shows the original dataset images, and the right column shows the model's prediction boxes for "rip_current."

8. Analysis and Discussion

8.1) Comparative Study

The table clearly shows that Transformer models are far more accurate than my YOLOv8 model. Mask2Former+DINOv3 has an F1 score of 71%, while my YOLO only got 15% [1]. This is a huge difference. But wait, this comparison is not 100% fair. The Transformer results came from the NTIRE paper [1]. The teams in that competition used powerful GPUs, huge amounts of training data, and complex tricks like model ensembling. I only used a weak CPU, no tricks, and a limited training set.

8.2) Why My YOLOv8-seg Scores Are So Low

I want to be very honest about why my scores are low compared to the NTIRE values. It's not because YOLOv8 is a bad model. It's because of the conditions I had to work in. Here are the main reasons:

1. **No GPU:** This is the biggest reason. My model was trained on a CPU. This means that in the time a GPU can train for 100 epochs, my CPU might only do 30. Furthermore, the optimization process on a CPU is more prone to getting stuck in local minima, so it doesn't learn complex features as well.
2. **Corrupted Ground Truth:** The JSON file for the dataset was completely corrupted. The annotations did not match the image filenames, so I bypassed the file and used misaligned labels. This means the model was trained on incorrect information. It was penalized for detecting rip currents not in the corrupted label and rewarded for missing rip currents that were actually present. This is a critical flaw. This likely explains why my precision and recall

are so low. I recorded 821 false positives, meaning the model classified a normal wave as a rip current because the corrupted annotation file said it was not.

3. **Fewer Epochs:** I only trained for 30 epochs. The NTIRE teams likely trained for 100+ epochs or used pre-trained weights [32]. With only 30 epochs on a CPU, the model barely learns how to identify the basic features of a rip current.
4. **Class Imbalance:** The RipVIS dataset has more background images than rip current images. The model might have just learned to predict "background" all the time. This leads to high precision (when it does predict, it's right) but very low recall (it misses most rip currents). My recall is only 13%, which means I missed 87% of the rip currents! That's terrible for a safety system.
5. **Memory Constraints:** I could only process batches of 10 images at a time before my laptop crashed. This prevented me from using any advanced data augmentation techniques (like flipping, rotating, or adding noise) because these would have used up too much RAM.

8.3) Real-World Examples, Impacts, and Challenges

Let me explain what these scores mean in the real world.

- If we used my YOLOv8 model at Cox's Bazar, it would miss 87% of the rip currents. Imagine 100 people are in the water, and 10 rip currents appear. My model would only warn about 1 or 2 of them. The other 8 would go unnoticed. That would be a disaster. The 70% accuracy sounds okay, but that's just because most of the test images don't have rip currents. The recall of 13% is what really matters for saving lives.
- If we used Mask2Former, it would detect 70% of the rip currents. That's much better. But still, 30% of the rip currents would be missed. The NTIRE paper says there is "room for improvement" [1].
- Even the Transformer models create false alarms. If a system tells beachgoers there is a rip current when there isn't one, people will start ignoring the system. This is dangerous. The model fails when the lighting is bad, like at dawn or dusk. The water looks uniform, and the rip current blends in. Small rip currents (less than 500 pixels) are often missed by all models. When the waves are breaking heavily over the rip current, the model can't see the shape and draws incomplete masks.

8.4) Hypothesis Testing

I hypothesized that Transformers will score higher than CNNs. This hypothesis is **supported** by the data [1]. However, the comparison wasn't perfectly fair because I couldn't train the Transformers. But even with that limitation, the Transformer's architecture is designed to capture global context [46], which is exactly what is needed for a large, amorphous object like a rip current.

9. Research Limitations

I must be very clear about the limitations of my study:

9.1) Dataset Limitations

The RipVIS dataset [21] doesn't include any beaches from Bangladesh. The sand in Cox's Bazar is different from Australian sand. The waves are different. The lighting is different. So, my model (and the NTIRE models) will need to be retrained on local data before being deployed safely in Bangladesh.

9.2) Methodological Limitations

I only trained YOLOv8. I didn't train the Transformers. So, comparing the YOLO results with the paper's results [1] is not a controlled experiment.

9.3) Computational Limitations

I used a CPU-only computer with 8GB RAM. I couldn't process all the images. I only used 6,722 images for testing and 7,308 for training.

9.4) Annotation Mapping Limitations

Because of the corrupt JSON annotations, I had to use a custom split of the data. This means my F1/F2 scores are not completely accurate. They are best estimates based on limited, misaligned data. If I had the real annotations, the results would be different (probably much higher). This is the single most heartbreaking limitation of my project.

9.5) Generalization Limitations

The RipVIS dataset is a benchmark. Real-world conditions in Bangladesh might be very different. The model needs fine-tuning.

10. Timeline with Professional Practice

I followed a 12-week plan to complete my project. Here is the timeline I followed:

Week	Task	Output
1-2	Set up the computer and install the software to download the RipVIS dataset [21].	Working environment
3-4	Train YOLOv8-seg and tune it [32].	YOLOv8-seg model ready
5-6	Study SegFormer-B5 and prepare code [46].	SegFormer code ready
7-8	Study MaskFormer with DIN v3 and prepare code [11] [28].	MaskFormer code ready
9-10	Test all models and compute performance scores.	Performance scores & comparison table
11	Analyze where each model	Failure report
12	Write the final report.	Complete report

I worked like a professional. Each week, I wrote down what I did and what problems I faced. If a task took longer than planned, I used the extra time in weeks 9-10 to catch up. I learned by reading the official documentation for each model and by looking at how winning teams in the NTIRE 2026 challenge [1] solved problems. This was my own independent work. No one did it for me.

11. Conclusions & Recommendations

11.1) Conclusions

In this project, I compared three deep learning models for rip current detection. Here are my conclusions:

- ***YOLOv8-seg*** is the fastest and easiest to run. It runs at 15 FPS on a CPU [32]. This makes it perfect for smaller, budget-friendly beach safety systems.
- ***Mask2Former with DINOv3*** is the most accurate. It has the highest F1 score (71%) [1]. But it needs a 24GB+ GPU, which is very expensive [11, 28].
- ***Transformer models are generally better than CNNs for this task.*** They understand the global shape of the rip current better [46].
- **All models fail on small, low-contrast, or heavily waved rip currents.** This is a key problem that future research must solve.
- **My own model was very poor because of hardware limitations and corrupted data.** My F1 score of 14.96% is not because the YOLO model is bad; it's because I trained it on a weak CPU with severe ground truth mapping errors.

11.2) Recommendations for Future Work

- ***Collect Bangladesh Beach Dataset:*** The most important future work is to collect images from Cox's Bazar and St. Martin's Island. This would allow models to be trained on local data and give better results for Bangladeshi beaches.
- ***Use Video Instead of Images:*** Rip currents change over time. A model that uses video (*multiple frames*) might be more accurate than a model that uses single images [21].
- ***Try Other Transformer Models:*** Future work should test models like SAM 2, DINOv2, or newer YOLO versions (like YOLOv11 or YOLO13) [27, 29, 30].
- ***Deploy as a Mobile App:*** The best way to save lives is to make the technology accessible. Future work could turn the best model into a mobile app that beachgoers can use to check for rip currents.

12. Reflection

12.1) What I Learned

This project taught me a lot about deep learning, but more importantly, it taught me how to survive in the real world of machine learning where things rarely go as planned. I learned how to prepare data for YOLO training. I learned how to calculate F1 and F2 scores. I learned the differences between CNN and Transformer architectures. Most importantly, I learned how much hardware matters in machine learning. My whole project was limited by my 8GB RAM. It taught me resilience and problem-solving.

12.2) Challenges I Faced

The biggest challenge was the hardware limitation. My laptop only has 8GB of RAM and no GPU. This affected everything I did. When I tried to load large batches of images into memory, my laptop would overheat and automatically restart. I had to implement a "batch-processing" system where I would process 10 images at a time, save the results, clear the RAM, and then process the next 10. This made the project incredibly slow but was the only way to stop the crashes.

I also faced severe "PermissionError" and "FileNotFoundError" because the downloaded files were split into multiple zip archives, and some files had incorrect extensions. I had to write a script to manually fix the file extensions and move files to the correct folders. Additionally, many image files in the dataset were corrupted or empty (0 bytes), which caused my Python scripts to crash. I had to write extensive error-checking code to manually skip these corrupted files. This reduced my effective dataset size even further.

The most stressful challenge was the annotation mapping error. The JSON annotation file for the RipVIS dataset was completely broken. The annotations did not align with the image filenames. This meant the ground truth labels were useless for training. I had to write custom code to re-map the annotations, but due to the corrupted file structure, I could not recover 100% of the labels. This directly caused my model to have high false positives and false negatives. Because I was unable to use the correct labels, I had to bypass the JSON entirely. This was a massive, heartbreaking engineering hurdle that I had to overcome just to finish the project.

12.3) What I Would Do Differently

If I had a powerful GPU and more time, I would train all three models properly. I would train YOLO for 100 epochs. I would fix the JSON annotation file properly using a different library to ensure the ground truth was 100% correct. I would process all 14,616 images. I would use data augmentation (like flipping, rotating, and changing colors) to help the model deal with the class imbalance. I would do a truly fair comparison.

12.4) Professional Practice

I worked independently. I followed a 12-week timeline. I documented everything. I read the official documentation and research papers to solve problems. I learned to debug code, fix data corruption, and manage system memory. This experience has made me a better researcher and a stronger engineer. It mirrors the reality of working in industry, where data is often messy and hardware is constrained.

13. References

- [1] Dumitriu, A., Ralhan, A., Miron, F., Tatui, F., Ionescu, R., Timofte, R., Naeem, A., Katwal, A., Dey, A., Hoque, M., & others (2026). NTIRE 2026 Rip Current Detection and Segmentation (RipDetSeg) Challenge Report. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 1784–1795).
- [2] Ancuti, R., Brateanu, A., Ancuti, C., Timofte, R., & Ancuti, C. (2026). NT-HAZE: A Benchmark Dataset for Realistic Night-time Image Dehazing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 1598–1607).
- [3] Ancuti, R., Brateanu, A., Balmez, R., Orhei, C., Vasluianu, F., O Ancuti, C., Timofte, R., Ancuti, C., Dong, W., Heydari, M., & others (2026). Ntire 2026 nighttime image dehazing challenge report. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 1608–1625).
- [4] Castelle, B., Scott, T., Brander, R., & McCarroll, R. (2016). Rip current types, circulation, and hazard. *Earth-Science Reviews*, 163, 1–21.
- [5] Brander, R., & Short, A. (2000). Morphodynamics of a large-scale rip current system at Muriwai Beach, New Zealand. *Marine Geology*, 165(1-4), 27–39.
- [6] Brewster, B., Gould, R., & Brander, R. (2019). Estimates of rip current rescues and drownings in the United States. *Natural Hazards and Earth System Sciences*, 19(2), 389–397.

- [7] Brander, R., Dominey-Howes, D., Champion, C., Del Vecchio, O., & Brighton, B. (2013). Brief Communication: A new perspective on the Australian rip current hazard. *Natural hazards and earth system sciences*, 13(6), 1687–1690.
- [8] Da F. Klein, A., Santana, G., Diehl, F., & De Menezes, J. (2003). Analysis of hazards associated with sea bathing: results of five years' work on oceanic beaches of Santa Catarina state, southern Brazil. *Journal of Coastal Research*, 107–116.
- [9] Lushine, J. (1991). A study of rip current drownings and related weather factors. *National weather digest*, 16(3), 13–19.
- [10] Silva, A., Mori, I., Dusek, G., Davis, J., & Pang, A. (2021). Automated rip current detection with region-based convolutional neural networks. *Coastal Engineering*, 166, 103859.
- [11] Cheng, B., Misra, I., Schwing, A., Kirillov, A., & Girdhar, R. (2022). Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 1290–1299).
- [12] Maryan, C., Hoque, M., Michael, C., Ioup, E., & Abdelguerfi, M. (2019). Machine learning applications in detecting rip channels from images. *Applied Soft Computing*, 78, 84–93.
- [13] Zhu, D., Qi, R., Hu, P., Su, Q., Qin, X., & Li, Z. (2022). YOLO-Rip: A modified lightweight network for rip current detection. *Frontiers in Marine Science*, 9, 930478.
- [14] Mori, I., et al. "Flow-based rip current detection and visualization," in *IEEE Access*, vol. 10, pp. 6483–6495, 2022.
- [15] Rashid, A., et al. "RipNet: A lightweight one-class deep neural network for the identification of rip currents." In *International Conference on Neural Information Processing*, 2020, pp. 172–179.
- [16] Rashid, A., et al., "RipDet: A fast and lightweight deep neural network for rip current detection," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–6.
- [17] Rashid, A., et al. "Reducing rip current drowning: An improved residual-based lightweight deep architecture for rip detection," in *ISA transactions*, vol. 132, pp. 199–207, 2023.
- [18] Rampal, N. et al. "Interpretable deep learning applied to rip current detection and localization," in *Remote Sensing*, vol. 14, no. 23, pp. 6048, 2022.
- [19] Dumitriu, A. et al., "Rip current segmentation: A novel benchmark and YOLOv8 baseline results," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 1261–1271.
- [20] Dumitriu, A. et al., "AIM 2025 rip current segmentation (RipSeg) challenge report," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025, pp. 5551–5560.
- [21] Dumitriu, A., et al., "Ripvis: Rip currents video instance segmentation benchmark for beach monitoring and safety," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 3427–3437.
- [22] Dumitriu, A., et al. "NTIRE 2026 Rip Current Detection and Segmentation (RipDetSeg) Challenge Report." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026, pp. 1784–1795.
- [23] He, K., et al., "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969.
- [24] Kukartsev, V., et al. "Deep learning for object detection in images: development and evaluation of the YOLOv8 model using Ultralytics and Roboflow libraries." In *the Computer Science Online Conference*, 2024, pp. 629–637.
- [25] R. Sapkota, M. Karkee. "Ultralytics YOLO evolution: An overview of YOLO26, YOLO11, YOLOv8, and YOLOv5 object detectors for computer vision and pattern recognition," in *arXiv preprint arXiv:2510.09653*, 2025.
- [26] Lei, M., et al. "Yolov13: Real-time object detection with hypergraph-enhanced adaptive visual perception," in *arXiv preprint arXiv:2506.17733*, 2025.
- [27] Kirillov, A. et al., "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026.
- [28] Simeoni, O., et al. "Dinov3," in *arXiv preprint arXiv:2508.10104*. 2025.
- [29] Ravi, N., et al. "Sam 2: Segment anything in images and videos." In *International Conference on Learning Representations*, 2025, pp. 28085–28128.

- [30] Ravi, N., et al., "Sam 2: Segment anything in images and videos." In *International Conference on Learning Representations*, 2025, pp. 28085–28128.
- [31] Lin, T., et al., "Microsoft Coco: Common objects in context." In *European Conference on Computer Vision*, 2014, pp. 740–755.
- [32] Kukartsev, V., et al., "Deep learning for object detection in images: development and evaluation of the YOLOv8 model using Ultralytics and Roboflow libraries." in *the Computer Science Online Conference*, 2024, pp. 629–637.
- [33] I. Loshchilov, F. Hutter. "Decoupled weight decay regularization," in *arXiv preprint arXiv:1711.05101*, 2017.
- [34] D. Kingma, J. Ba. "Adam: A method for stochastic optimization," in *arXiv preprint arXiv:1412.6980*, 2014.
- [35] Cubuk, E., et al. "RandAugment: Practical automated data augmentation with a reduced search space." In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 702–703.
- [36] R. Solovyev, W. Wang, T. Gabruseva. "Weighted boxes fusion: Ensembling boxes from different object detection models," in *Image and Vision Computing*, vol. 107, pp. 104117, 2021.
- [37] Khan, F., et al. "RipFinder: real-time rip current detection on mobile devices," in *Frontiers in Marine Science*, vol. 12, pp. 1549513, 2025.
- [38] Khan, F., et al. "RipScout: Real-time ML-assisted rip current detection and automated data collection using UAVs," in *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 18, pp. 7742–7755, 2025.
- [39] Wan, M., et al., "Ripalert: A future-frame-aware framework for rip current forecasting and early alerting." In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026, pp. 39368–39377.
- [40] Ali, S., et al. "IBN-Driven Rip Current Analysis Using UAVs for Next-Generation Coastal Surveillance," in *IEEE Internet of Things Journal*, 2025.
- [41] Catala-Gonell, A., et al. "Oriented Object Detection for Complex Hydrodynamic Features: A Multi-Platform Rip Current Identification System," in *EGUsphere*, vol. 2026, pp. 1–29, 2026.
- [42] J. Choi, M. Rajendran, Y. Suh. "Box2Rip: Instance Segmentation of Amorphous Rip Currents via Box-Supervised Learning," in *IEEE Access*, 2025.
- [43] Liu, Y., et al. "A Deep Learning-Based Pipeline for Detecting Rip Currents from Satellite Imagery," in *Remote Sensing*, vol. 18, no. 2, pp. 368, 2026.
- [44] A. Sun, K. Yang. "Rip Current Detection in Nearshore Areas through UAV Video Analysis with Almost Local-Isometric Embedding Techniques on Sphere," in *arXiv preprint arXiv:2304.11783*, 2023.
- [45] Zhou, B., et al., "Scene parsing through ADE20k dataset," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 633–641.
- [46] Xie, E., et al. "SegFormer: Simple and efficient design for semantic segmentation with transformers," in *Advances in neural information processing systems*, vol. 34, pp. 12077–12090, 2021.
- [47] Xu, Z., S. Escalera, A. Pavão, M. Richard, W. Tu, Q. Yao, H. Zhao, and I. Guyon. *Codabench: Flexible, easy-to-use, and reproducible meta-benchmark platform. Patterns 3 (7), 100543 (2022)*. 2022.
- [48] McGill, Sean P., and Jean T. Ellis. "Rip current and channel detection using surf cams and optical flow." *Shore & Beach* 90.1 (2022): 50.
- [49] Philip, Shweta, and Alex Pang. "Detecting and Visualizing Rip Current Using Optical Flow." In *EuroVis (Short Papers)*, pp. 19-23. 2016.
- [50] de Silva, Akila, et al. "RipViz: Finding rip currents by learning pathline behavior." *IEEE Transactions on Visualization and Computer Graphics* 30.7 (2023): 3930-3944.
- [51] Zhu, D., Qi, R., Hu, P., Su, Q., Qin, X., & Li, Z. (2022). YOLO-Rip: A modified lightweight network for rip current detection. *Frontiers in Marine Science*, 9, 930478.