Cichlid Computer Vision Project – Weekly Progress

Week ending Friday, February 21st, 2025

Time Log

Charlie Clark

What progress did you make?

- Attended a quick temporal pipeline overview with Kailey Monday afternoon.
- Attended weekly BioBoost meeting on Monday afternoon.
- Attended weekly higher-ed meeting on Monday evening.
- Attended a quick huddle with Eric to go over some aspects of PACE Tuesday afternoon.
- Attended weekly publication seminar Tuesday evening.
- Attended bi-weekly Freeman comp advisor peer meeting Wednesday morning.
- Attended weekly admin meeting Thursday afternoon.
- Attended monthly Freeman comp advisor – faculty meeting Friday afternoon.
- Attended bi-weekly Freeman researcher – faculty meeting Friday afternoon.
- Updated meeting recording procedures to make adoption more feasible (given the download issues we've been having).
- Implemented a few simple MLP-based temporal networks to use in post-processing.
- Looked into finding a Pythonic YOLOv5-cls implementation I could use in-code (instead of via a CLI); was unable to find any.

What are you planning on working on next?

- Attend required weekly meetings.
- Attend optional weekly meetings.
- Look into the classification-centric code in the YOLOv5 repo to see if we can extract the logits or the model implementation from it directly.
- Work with Eric on implementing the framework necessary for our temporal training needs.

Is anything blocking you from getting work done?

- **None**

Researcher

What did you do this week?

(1) I attended/watched the following 3 meetings/recordings:

(a) Logistics Meeting with Charlie on February 17th. I gave an overview of the temporal part of the pipeline and suggested replacing the simple DT with tree-based classifiers like random forests, etc. The meeting lasted 30 min.

(b) Cichlid Team Meeting on February 17th. I gave an overview of the model that I developed this week. I was at the meeting for about 1 hr.

(c) HAAG Publication Seminar on February 18th. We discussed the feedback from the WACV submission and general goals and questions for our rewrite. The meeting lasted about 1 hr.

(2) I worked on the meeting manager role requirements for the weekly meeting. Per the new requirements, I added a permanent meeting recording, meeting slides, and meeting notes to the Teams folder at Projects / Cichlid-CV / ReID / For Microsoft Planner. I updated the Teams attendance sheet at Projects / CichlidCV / ReID / For Microsoft Planner / Meeting Attendance Tracking.xlsx. I sent the meeting notes (that has links to all required materials) to the meeting-managers, time-log, and bioboost Slack channels.

(3) Building on my code for finding fish contours from last week, I found a way to actually classify the fish based on those contours. I used PCA to find the length of the fish, projected the contours onto the primary axis to find the head and tail, took the HSV hue difference, and applied a threshold to classify.

(4) Based on guidance given during the weekly team meeting, I applied clustering. Specifically, I found the hue for all points, clustered the HSV hue value using k-means, found the intra-cluster and between cluster distances, and applied thresholds to the distances in order to classify.

(5) To improve the contours, I created an aspect ratio filtering section to filter out sand detections, which tend to be more square. I attempted to write code to prevent the head and tail from being confused so often, but it made things worse, so I abandoned the attempt.

(6) When looking at HSV hue plots while doing clustering, I noticed a difference in range and standard deviation between male and female frames. Therefore, I also wrote some code to take the hues of the points of a contour and find the range and standard deviation. I then applied a non-tuned threshold to classify.

What are you going to do next week?

(1) I need to attend required meetings.

(2) I need to fulfill my meeting manager responsibilities.

(3) Other tasks as assigned after the weekly meeting on Monday.

(4) Choose a training and testing set. Using train, choose threshold values for head/tail hue difference, hue range, hue std, between cluster distance, and intra-cluster distance.

(5) If I have time, sort intra-cluster distances by size and try thresholding both or try new clustering algorithm.

(6) If I have time, choose the 2 or 3 best methods and run through neural net or something simple that can be trained with around 100 samples.

(7) If I have time, integrate with Eric's existing method for the contours. This may take some time. Note that the errors appear to be mostly from the coloration being difficult, not tank detections.

(8) If I have time, tune contour filtering, contour smoothing, and contour aspect ratio elimination.

Is anything blocking you from getting work done?

(1) I still need an email from Dr. Lytle for CS 8903. Doesn't look like he sent it.

Eric Iamarino

What did you do this week?

- Attended Cichlid Team Meeting on Monday
- Attended Weekly HAAG Publication Meeting on Tuesday
- Worked on Image Segmentation alternative for BioBoost o Fixed an open bug in the FastSAM library that prevented it from initially working
    - Setup script to iterate through each cichlid video frame, segment the frame, and store the results as a new video
    - Manually tuned the IoU and Confidence thresholds
    - Working on custom annotation function that won't fill in contours with a color
- Small amount of work towards temporal/post-processing model
    - Read and took notes on GeoAware Paper
    - Created diagram of this model but with our workflow
    - Discussed this with Charlie to understand how we are integrating the postprocessing model into our pipeline

- Adding Weekly Reports to Cichlid CV Website
- Adding Weekly Meeting to Cichlid CV Website

What are you going to do next week?

- Attend BioBoost meeting
- Attend publication meeting
- Integrate new segmentation approach with KQ's current workflow
- Help Charlie with post-processing and temporal model

Blockers, things you want to flag, problems, etc.

- None

Abstracts

Charlie Clark

"DeepFins: Capturing dynamics in underwater videos for fish detection", Jalal et al. (2025; Ecological Informatics).

- Abstract: "The monitoring of fish in their natural habitat plays a crucial role in anticipating changes within marine ecosystems. Marine scientists have a preference for automated, unrestricted underwater video-based sampling due to its non-invasive nature and its ability to yield desired outcomes more rapidly compared to manual sampling. Generally, research on automated video-based detection using computer vision and machine learning has been confined to controlled environments. Additionally, these solutions encounter difficulties when applied in real-world settings characterized by substantial environmental variability, including issues like poor visibility in unregulated underwater videos, challenges in capturing fish-related visual characteristics, and background interference. In response, we propose a hybrid solution that merges YOLOv11, a popular deep learning based static object detector, with a custom designed lightweight motion-based segmentation model. This approach allows us to simultaneously capture fish dynamics and suppress background interference. The proposed model i.e., DeepFins attains 90.0% F1 Score for fish detection on the OzFish dataset (collected by the Australian Institute of Marine Science). To the best of our knowledge, these results are the most accurate yet, showing about 11% increase over the closest competitor in fish detection tasks on this demanding benchmark OzFish dataset.

Moreover, DeepFins achieves an F1 Score of 83.7% on the Fish4Knowledge LifeCLEF 2015 dataset, marking an approximate 4% improvement over the baseline YOLOv11. This positions the proposed model as a highly practical solution for tasks like automated fish sampling and estimating their relative abundance."

- AI Summary by ChatGPT (4o): "This paper presents DeepFins, a hybrid deep learningbased system for fish detection in underwater videos. Unlike conventional methods that rely solely on static image-based object detection, DeepFins combines spatial and motion-based features to improve accuracy in complex underwater environments. The system integrates a modified YOLOv11 object detector with a custom lightweight motionbased segmentation model. This approach enhances detection performance by capturing fish movement while suppressing background noise from dynamic aquatic environments."
    - Key Contributions
        - "Hybrid Motion-Segmentation and Deep Learning Approach: The paper introduces a novel motion-based segmentation module that extracts moving fish from videos while reducing false positives caused by background motion (e.g., swaying plants and water currents). This module is combined with YOLOv11, which classifies detected objects as fish or non-fish."
        - "Performance Gains on Benchmark Datasets: Achieves 90.0% F1 Score on the OzFish dataset, an 11% improvement over the best existing fish detection methods. Demonstrates a 4% improvement over baseline YOLOv11 on the Fish4Knowledge LifeCLEF 2015 dataset."
        - "Superior Generalization and Real-Time Feasibility: The model is tested across two challenging datasets, showing robust performance even when trained on one dataset and tested on another. DeepFins processes frames in real time (~16-33ms per frame), making it suitable for real-world ecological monitoring."
    - Contributions to Knowledge:
        - "Addresses Environmental Variability: Unlike previous work that struggles with occlusions, turbidity, and lighting inconsistencies, DeepFins explicitly models fish motion, making it more robust in real-world underwater conditions."
        - "Bridges the Gap Between Static and Temporal Features: Many prior studies rely only on spatial object detection (e.g., YOLO variants, CNNs), while others use motion-based approaches (optical flow,

        GMM). This paper integrates both approaches, achieving state-of-the-art results."
        - "Advances Fish Monitoring Applications: The proposed method enables automated fish sampling and relative abundance estimation, which are critical for ecological research and conservation efforts."
    - Future Research Directions:
        - "Fish Species Classification: While DeepFins detects fish, it does not yet classify species effectively. Future research should explore integrating fine-grained classification models."
        - "Handling Occlusions and Long-Term Tracking: The system performs well on single-frame detection, but tracking fish across multiple frames remains an open challenge. Future work could incorporate temporal deep learning models (e.g., LSTMs, Transformers) to improve fish tracking."
        - "Data Augmentation for Low-Resource Species: Some fish species in the dataset have very few samples. Semi-supervised or few-shot learning techniques could help improve performance on rare species."
        - "Deployment in Real-World Scenarios: While the model is real-time capable, further optimizations could make it feasible for large-scale, realtime ocean monitoring systems."
- Link: https://linkinghub.elsevier.com/retrieve/pii/S1574954125000226

Researcher

Hu, J., et al. "Fish species classification by color, texture and multi-class support vector machine using computer vision." Computers and Electronics in Agriculture, 2012. https://www.sciencedirect.com/sc ience/article/abs/pii/S0168169912001937.

This week, I was looking for papers that talked about machine learning and fish coloration. I was looking for insight into how other researchers have used hue differences to classify fish. The study focuses on a new method for classifying fish species using images taken with smartphones, which is important for diagnosing fish diseases in rural China. By analyzing the color and texture of fish skin, the researchers extracted specific features from the images and used a machine learning technique called multi-class support vector machine (MSVM) to classify the species. The results showed that a particular wavelet filter provided the best accuracy for identifying fish species, making it a useful tool for farmers to detect diseases early. The study has some limitations, particularly with the classification of silver carp, which can be confused with other fish species, leading to inaccurate results. This misclassification could hinder the effectiveness of the fish disease diagnostic system. Additionally, the images used for classification were manually selected, so creating a computer program to automatically choose the best images would improve the testing and accuracy of the classification method. No future work is discussed.

Eric Iamarino

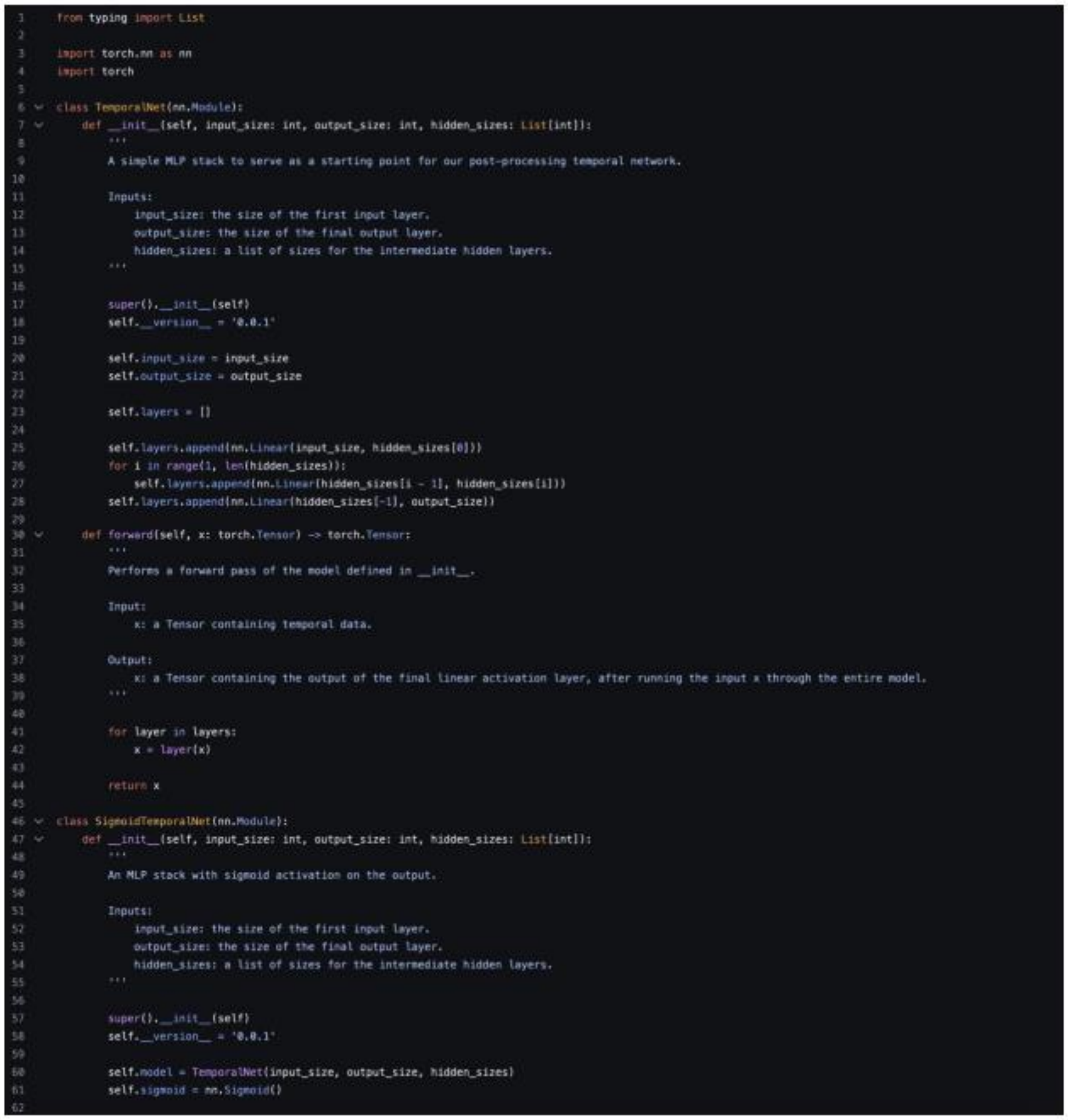
Zhao, X., Ding, W., An, Y., Du, Y., Yu, T., Li, M., Tang, M., & Wang, J. (2023, June 21). Fast segment anything. arXiv.org. https://doi.org/10.48550/arXiv.2306.12156

The recently proposed segment anything model (SAM) has made a significant influence in many computer vision tasks. It is becoming a foundation step for many high-level tasks, like image segmentation, image caption, and image editing. However, its huge computation costs prevent it from wider applications in industry scenarios. The computation mainly comes from the Transformer architecture at high-resolution inputs. In this paper, we propose a speed-up alternative method for this fundamental task with comparable performance. By reformulating the task as segments-generation and prompting, we find that a regular CNN detector with an instance segmentation branch can also accomplish this task well. Specifically, we convert this task to the well-studied instance segmentation task and directly train the existing instance segmentation method using only 1/50 of the SA-1B dataset published by SAM authors. With our method, we achieve a comparable performance with the SAM method at 50 times higher run-time speed. We give sufficient experimental results to demonstrate its effectiveness. The codes and demos will be released at this https URL.

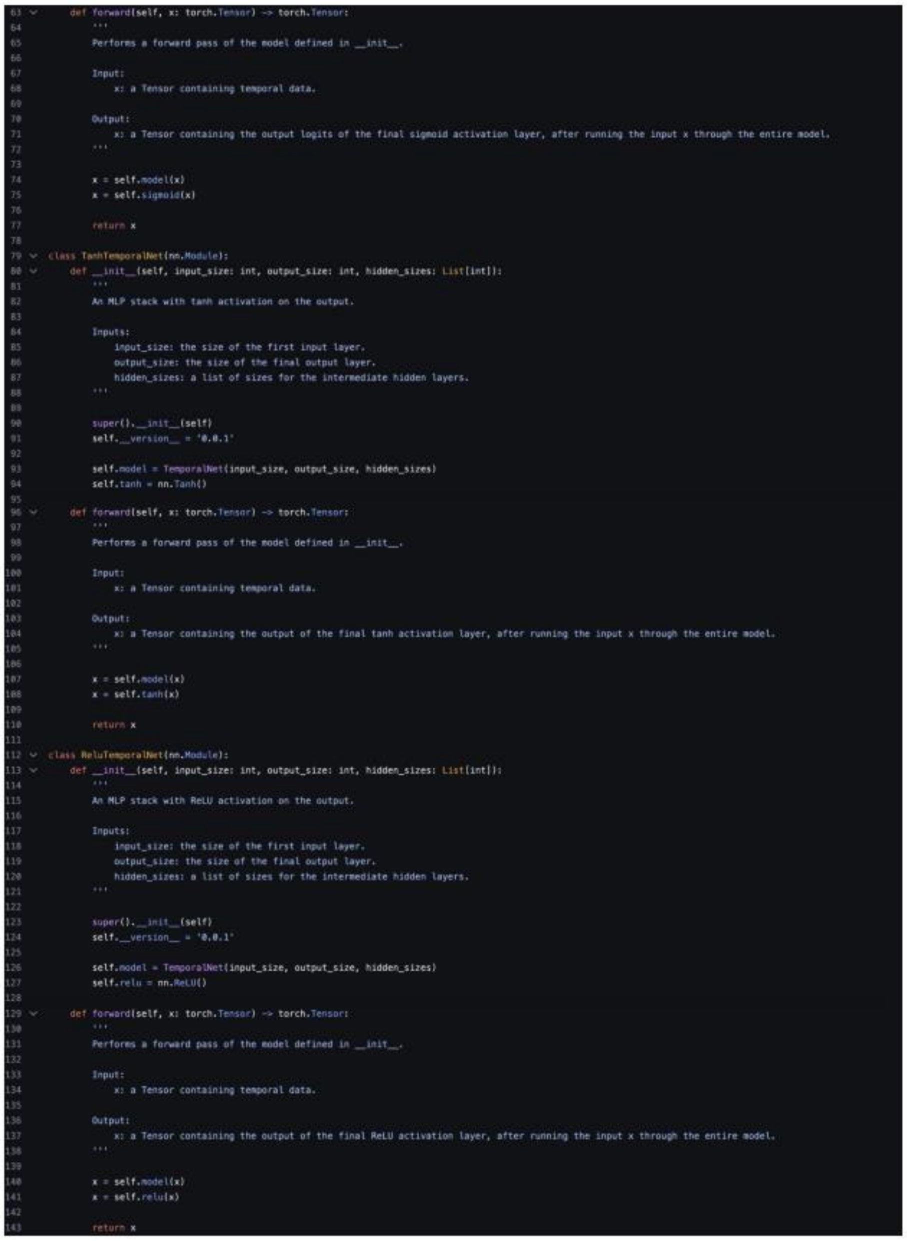
Documentation of Work

Charlie Clark

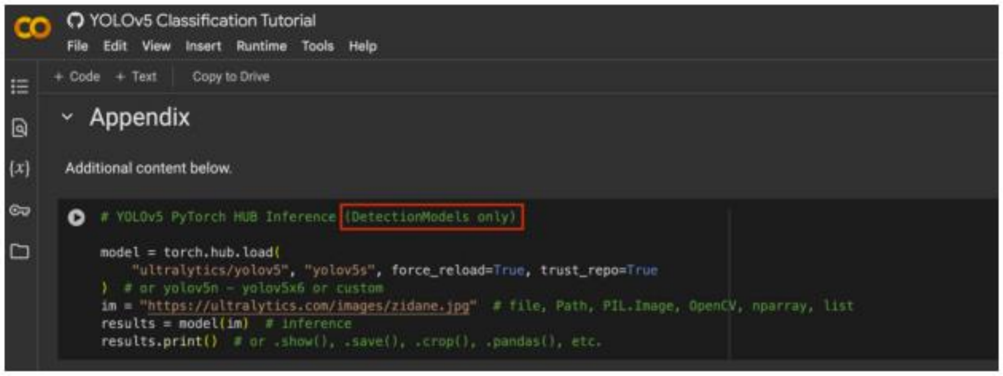
- Attended a quick temporal pipeline overview with Kailey Monday afternoon.
- Attended weekly BioBoost meeting on Monday afternoon.
- Attended weekly higher-ed meeting on Monday evening.
- Attended a quick huddle with Eric to go over some aspects of PACE Tuesday afternoon.
- Attended weekly publication seminar Tuesday evening.
- Attended bi-weekly Freeman comp advisor peer meeting Wednesday morning.
- Attended weekly admin meeting Thursday afternoon.
- Attended monthly Freeman comp advisor – faculty meeting Friday afternoon.
- Attended bi-weekly Freeman researcher – faculty meeting Friday afternoon.
- Updated meeting recording procedures to make adoption more feasible (given the download issues we've been having).
- Implemented a few simple MLP-based temporal networks to use in post-processing.
- Looked into finding a Pythonic YOLOv5-cls implementation I could use in-code (instead of via a CLI); was unable to find any.



```python
from typing import List

import torch.nn as nn
import torch

class TemporalNet(nn.Module):
    def __init__(self, input_size: int, output_size: int, hidden_sizes: List[int]):
        '''
        A simple MLP stack to serve as a starting point for our post-processing temporal network.

        Inputs:
            input_size: the size of the first input layer.
            output_size: the size of the final output layer.
            hidden_sizes: a list of sizes for the intermediate hidden layers.
        '''

        super().__init__(self)
        self.__version__ = '0.0.1'

        self.input_size = input_size
        self.output_size = output_size

        self.layers = []

        self.layers.append(nn.Linear(input_size, hidden_sizes[0]))
        for i in range(1, len(hidden_sizes)):
            self.layers.append(nn.Linear(hidden_sizes[i - 1], hidden_sizes[i]))
        self.layers.append(nn.Linear(hidden_sizes[-1], output_size))

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        '''
        Performs a forward pass of the model defined in __init__.

        Input:
            x: a Tensor containing temporal data.

        Output:
            x: a Tensor containing the output of the final linear activation layer, after running the input x through the entire model.
        '''

        for layer in layers:
            x = layer(x)

        return x

class SigmoidTemporalNet(nn.Module):
    def __init__(self, input_size: int, output_size: int, hidden_sizes: List[int]):
        '''
        An MLP stack with sigmoid activation on the output.

        Inputs:
            input_size: the size of the first input layer.
            output_size: the size of the final output layer.
            hidden_sizes: a list of sizes for the intermediate hidden layers.
        '''

        super().__init__(self)
        self.__version__ = '0.0.1'

        self.model = TemporalNet(input_size, output_size, hidden_sizes)
        self.sigmoid = nn.Sigmoid()

```



```python
    def forward(self, x: torch.Tensor) -> torch.Tensor:
        '''
        Performs a forward pass of the model defined in __init__.

        Input:
            x: a Tensor containing temporal data.

        Output:
            x: a Tensor containing the output logits of the final sigmoid activation layer, after running the input x through the entire model.
        '''

        x = self.model(x)
        x = self.sigmoid(x)

        return x

class TanhTemporalNet(nn.Module):
    def __init__(self, input_size: int, output_size: int, hidden_sizes: List[int]):
        '''
        An MLP stack with tanh activation on the output.

        Inputs:
            input_size: the size of the first input layer.
            output_size: the size of the final output layer.
            hidden_sizes: a list of sizes for the intermediate hidden layers.
        '''

        super().__init__(self)
        self.__version__ = '0.0.1'

        self.model = TemporalNet(input_size, output_size, hidden_sizes)
        self.tanh = nn.Tanh()

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        '''
        Performs a forward pass of the model defined in __init__.

        Input:
            x: a Tensor containing temporal data.

        Output:
            x: a Tensor containing the output of the final tanh activation layer, after running the input x through the entire model.
        '''

        x = self.model(x)
        x = self.tanh(x)

        return x

class ReluTemporalNet(nn.Module):
    def __init__(self, input_size: int, output_size: int, hidden_sizes: List[int]):
        '''
        An MLP stack with ReLU activation on the output.

        Inputs:
            input_size: the size of the first input layer.
            output_size: the size of the final output layer.
            hidden_sizes: a list of sizes for the intermediate hidden layers.
        '''

        super().__init__(self)
        self.__version__ = '0.0.1'

        self.model = TemporalNet(input_size, output_size, hidden_sizes)
        self.relu = nn.ReLU()

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        '''
        Performs a forward pass of the model defined in __init__.

        Input:
            x: a Tensor containing temporal data.

        Output:
            x: a Tensor containing the output of the final ReLU activation layer, after running the input x through the entire model.
        '''

        x = self.model(x)
        x = self.relu(x)

        return x
```

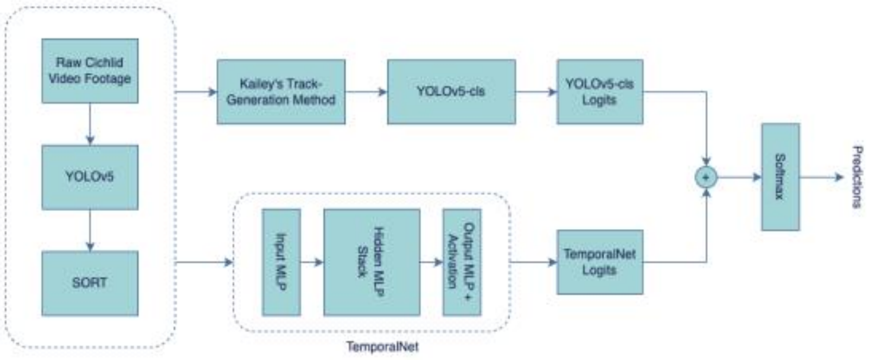

YOLOv5 Classification Tutorial
File Edit View Insert Runtime Tools Help
+ Code + Text Copy to Drive

Appendix

Additional content below.

```
# YOLOv5 PyTorch HUB Inference (DetectionModels only)

model = torch.hub.load(
    "ultralytics/yolov5", "yolov5s", force_reload=True, trust_repo=True
)  # or yolov5n - yolov5x6 or custom
im = "https://ultralytics.com/images/zidane.jpg"  # file, Path, PIL.Image, OpenCV, nparray, list
results = model(im)  # inference
results.print()  # or .show(), .save(), .crop(), .pandas(), etc.
```

2. *Results Visualization:* Find below a diagram depicting our current idea for a temporal-net post-processing pipeline.



Researcher

For a full list of what I did, see the time log above. The first big thing I did this week was to find a way to actually classify fish based on the contours I created last week. I used PCA to find the length of the fish, projected the contours onto the primary axis to find the head and tail, took the HSV hue difference, and applied a threshold to classify. In Figure 1, a screenshot of one of the videos with the head and tail points automatically labeled is shown.

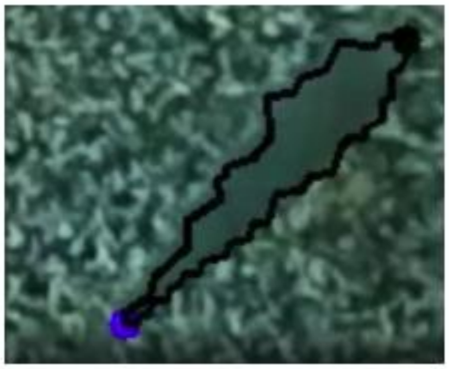


Figure 1: Determining Head and Tail on a Contour

Using my guessed threshold, I got an accuracy of 82.14% across all 140 tracks, where 60 tracks were male and 80 tracks were female. For reference, the strategy of always guessing that the fish is female results in an accuracy of 57.14%.

```python
# Find and Filter Contours
...
# Compute Eigenvectors, Project to Principal Axis
mean, eigenvectors = cv2.PCACompute(points, mean=None)
principal_axis = eigenvectors[0]
projections = np.dot(points - mean, principal_axis)

# Get Head
max_idx = np.argmax(projections)
head_point = tuple(points[max_idx].astype(int))

# Get Tail
min_idx = np.argmin(projections)
tail_point = tuple(points[min_idx].astype(int))

# Get Hues of Head and Tail
head_hue = hsv_frame[head_point[1], head_point[0], 0]
tail_hue = hsv_frame[tail_point[1], tail_point[0], 0]

# Calculate Abs Value of Difference in Hue btwn Head and Tail
# HSV has Circular Range 0-179!! Must Check Forward and Backward
hue_dif_1 = head_hue - tail_hue
hue_dif_2 = tail_hue - head_hue
if (hue_dif_1 < hue_dif_2):
    hue_dif = abs(hue_dif_1)
else:
    hue_dif = abs(hue_dif_2)
...
# Use Hue Difference to Classify
```

Listing 1: Using PCA and HSV Hue Difference between Head and Tail to Classify

The next step was to apply the k-means clustering algorithm to all of the points' hues in each contour. I forced k-means to take two clusters, since that is the expected outcome of male fish. In contrast, the female fish will more closely resemble a single cluster. Because of this, I believe that the intra-cluster differences of males and females will vary. I added the two cluster's intra-cluster distances together. Then, I applied a threshold to that distance in order to classify. The threshold was guessed and is not tuned as of now. With my initial threshold value, an accuracy of 75% was achieved. I also took a between cluster distance with an initial, untuned threshold value and got an accuracy of 72.86%.

```python
# Cluster
kmeans = KMeans(n_clusters=2, random_state=42)
kmeans.fit(hue_all_pts)
labels = kmeans.labels_
centroids = kmeans.cluster_centers_

# Find Intra and Between Cluster Distances
intra_cluster_distances = []
for i in range(2):
    cluster_points = hue_all_pts[labels == i]
    centroid = centroids[i]
    distances = np.linalg.norm(cluster_points - centroid, axis=1)
    intra_cluster_distances.append(np.mean(distances))
between_cluster_distance = np.linalg.norm(centroids[0] - centroids[1])
```

Listing 2: Cluster Points and Find Distances

When looking at HSV hue plots while doing clustering, I noticed a difference in range and standard deviation between male and female frames. Therefore, I also wrote some code to take the hues of the points of a contour and find the range and standard deviation. I then applied a non-tuned threshold to classify. The hue range accuracy across all tracks was 72.14%. The hue standard deviation accuracy was 73.57%.

```
hue_range = np.max(hue_all_pts) - np.min(hue_all_pts)
hue_std_dev = np.std(hue_all_pts)
```

Listing 3: Find the Range and Standard Deviation of Points

To improve the contours, I created an aspect ratio filtering section to filter out sand detections, which tend to be more square. I attempted to write code to prevent the head and tail from being confused so often, but it made things worse, so I abandoned the attempt. A summary of the performances of the different methods used is shown in Figure 2. Note that the ranking of these performances will likely change after tuning.

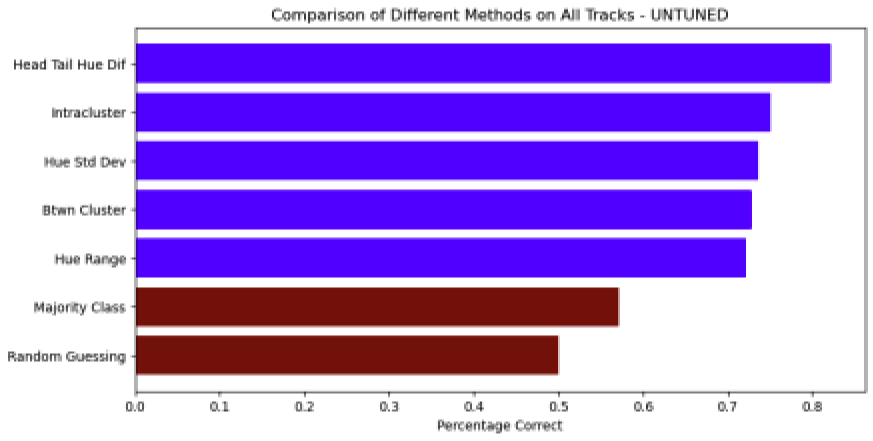
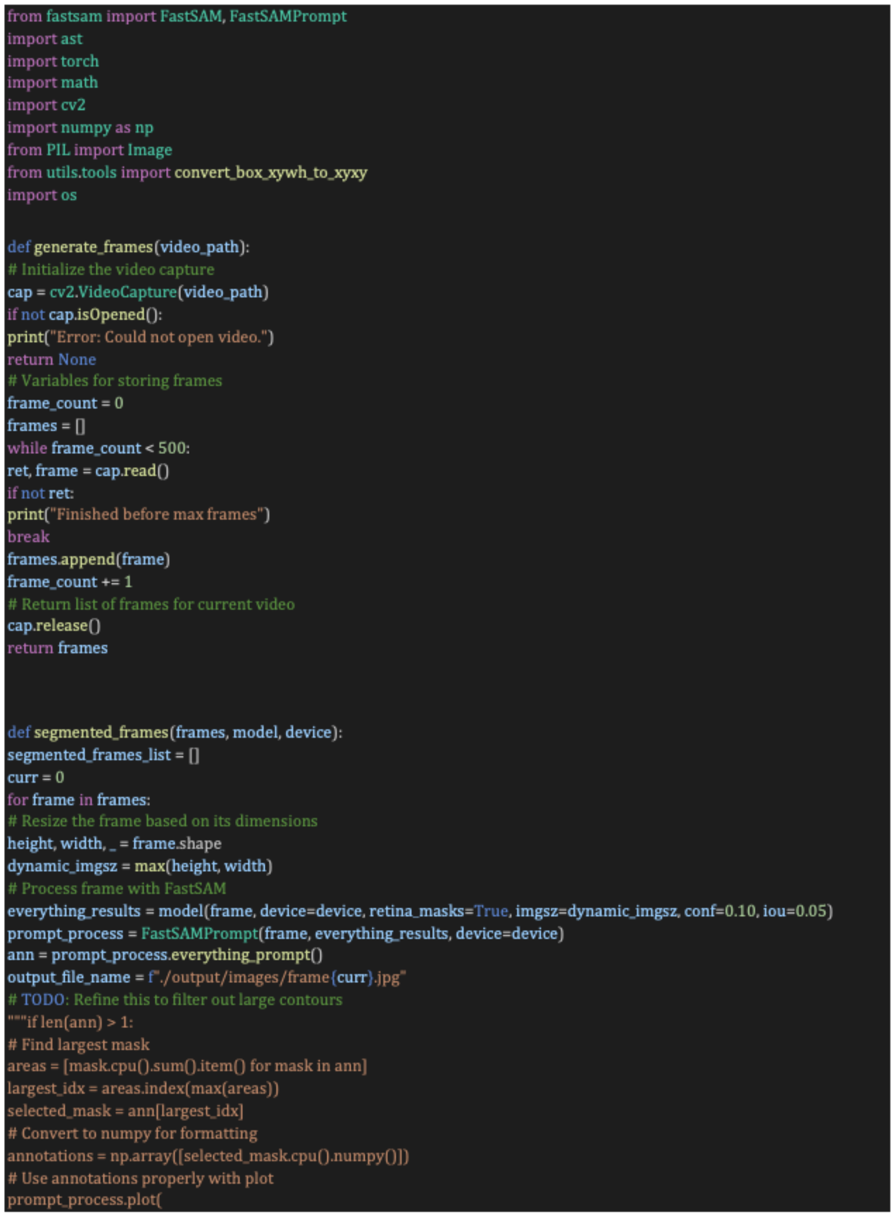


Figure 2: Comparison of Performance for Untuned Thresholds

And that's it! The finished scripts will be uploaded to BioBoost upon completion: https://github.c om/Human-Augment-Analytics/Bio-Boost. Specifically, my current version is here: https://github.com /Human-Augment-Analytics/Bio-Boost/blob/main/scripts/modeling/img_seg/img-seg-cleaned.ipy nb.

Eric Iamarino

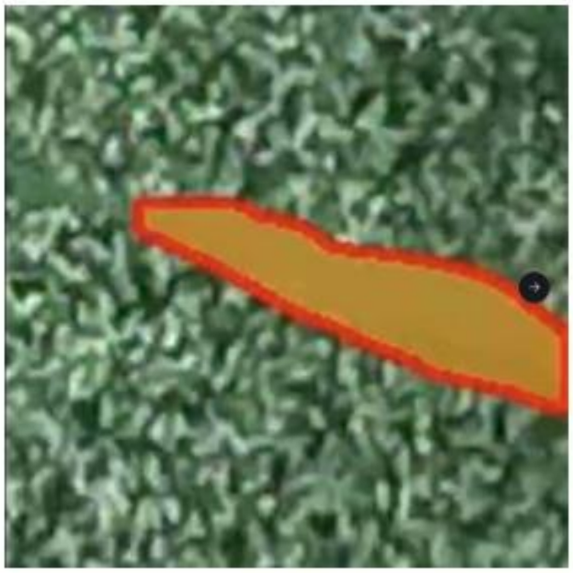
FastSAM Code:



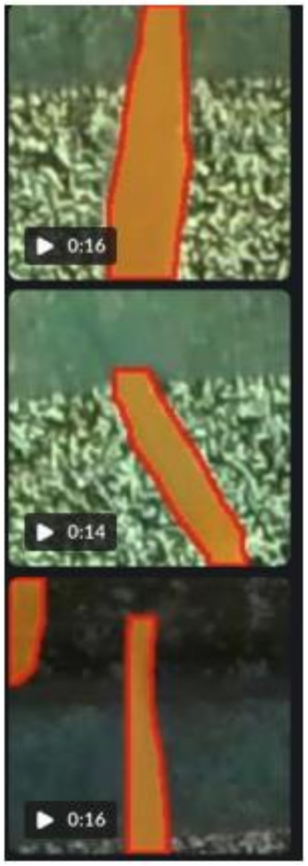
```
from fastsam import FastSAM, FastSAMPrompt
import ast
import torch
import math
import cv2
import numpy as np
from PIL import Image
from utils.tools import convert_box_xywh_to_xyxy
import os

def generate_frames(video_path):
# Initialize the video capture
cap = cv2.VideoCapture(video_path)
if not cap.isOpened():
print("Error: Could not open video.")
return None
# Variables for storing frames
frame_count = 0
frames = []
while frame_count < 500:
ret, frame = cap.read()
if not ret:
print("Finished before max frames")
break
frames.append(frame)
frame_count += 1
# Return list of frames for current video
cap.release()
return frames


def segmented_frames(frames, model, device):
segmented_frames_list = []
curr = 0
for frame in frames:
# Resize the frame based on its dimensions
height, width, _ = frame.shape
dynamic_imgsz = max(height, width)
# Process frame with FastSAM
everything_results = model(frame, device=device, retina_masks=True, imgsz=dynamic_imgsz, conf=0.10, iou=0.05)
prompt_process = FastSAMPrompt(frame, everything_results, device=device)
ann = prompt_process.everything_prompt()
output_file_name = f"./output/images/frame{curr}.jpg"
# TODO: Refine this to filter out large contours
"""if len(ann) > 1:
# Find largest mask
areas = [mask.cpu().sum().item() for mask in ann]
largest_idx = areas.index(max(areas))
selected_mask = ann[largest_idx]
# Convert to numpy for formatting
annotations = np.array([selected_mask.cpu().numpy()])
# Use annotations properly with plot
prompt_process.plot(
annotations=annotations
output_path=output_file_name,
mask_random_color=False,
withContours=True,
retina=True
)
"""
# Generate the segmented frame
prompt_process.plot(annotations=ann, output_path=output_file_name, mask_random_color=False, withContours=True,
retina=True)
# Write segmented frame to new video
if not os.path.exists(output_file_name):
cv2.imwrite(output_file_name, frame)
print("Wrote frame")
segmented_frames_list.append(output_file_name)
curr += 1
return segmented_frames_list

def cleanup_segmented_images(segmented_frame_paths):
# Delete all the temporarily stored frames
for frame_path in segmented_frame_paths:
try:
os.remove(frame_path)
print(f"Removed {frame_path}")
except Exception as e:
print(f"Error removing {frame_path}: {e}")
def write_video(segmented_frame_paths, output_video_path, fps=30):
# Get image size from frames
first_frame = cv2.imread(segmented_frame_paths[0])
frame_height, frame_width, _ = first_frame.shape
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
out = cv2.VideoWriter(output_video_path, fourcc, fps, (frame_width, frame_height))

# Write out frames
for frame_path in segmented_frame_paths:
frame = cv2.imread(frame_path)
out.write(frame)
out.release()
print(f"Video saved to {output_video_path}")



def main():
# Define input/output paths
video_path = "MC_singlenuc24_4_Tk47_030320__0001_vid__454_male_tricky.mp4"
output_video_path = "./output/maleTricky_highConf_lowIoU.mp4"

# Load model
model = FastSAM('./weights/FastSAM-x.pt')
DEVICE = 'cuda'

# Generate blank frames
frames = generate_frames(video_path)
```

```
if frames is None:
print("Unable to generate frames from video")
return
# Segment those frames
segmented_frames_list = segmented_frames(frames, model, DEVICE)

# Write out those frames
write_video(segmented_frames_list, output_video_path)

# Cleanup
cleanup_segmented_images(segmented_frames_list)

if __name__ == "__main__":
main()
```

Initial Results:



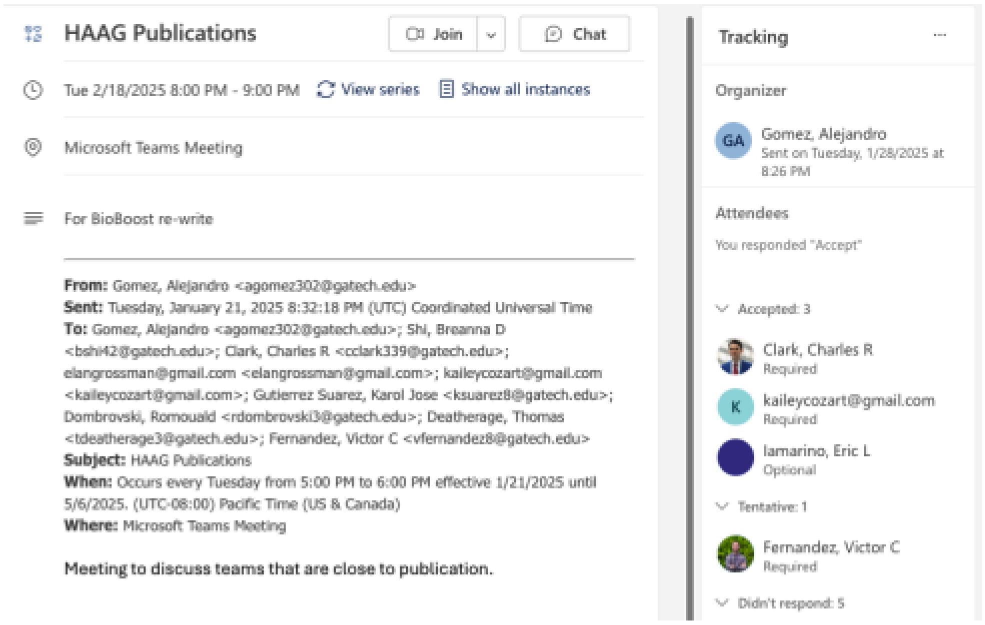
Issue I ran into and resolved: https://github.com/CASIA-IVA-Lab/FastSAM/issues/261



Cichlid CV Website for updates: https://sites.gatech.edu/cichlid-computer-vision-project/

BioBoost Weekly Meeting: Cichlid CV Weekly Meeting-20250217_170453-Meeting Recording.mp4

Publication Meeting:



HAAG Publications

Join | Chat

Tue 2/18/2025 8:00 PM - 9:00 PM View series Show all instances

Microsoft Teams Meeting

For BioBoost re-write

From: Gomez, Alejandro <agomez302@gatech.edu>
Sent: Tuesday, January 21, 2025 8:32:18 PM (UTC) Coordinated Universal Time
To: Gomez, Alejandro <agomez302@gatech.edu>; Shi, Breanna D <bshi42@gatech.edu>; Clark, Charles R <cclark339@gatech.edu>; elangrossman@gmail.com <elangrossman@gmail.com>; kaileycozart@gmail.com <kaileycozart@gmail.com>; Gutierrez Suarez, Karol Jose <ksuarez8@gatech.edu>; Dombrovski, Romouald <rdombrovski3@gatech.edu>; Deatherage, Thomas <tdeatherage3@gatech.edu>; Fernandez, Victor C <vfernandez8@gatech.edu>
Subject: HAAG Publications
When: Occurs every Tuesday from 5:00 PM to 6:00 PM effective 1/21/2025 until 5/6/2025. (UTC-08:00) Pacific Time (US & Canada)
Where: Microsoft Teams Meeting

Meeting to discuss teams that are close to publication.

Tracking

Organizer

GA Gomez, Alejandro
Sent on Tuesday, 1/28/2025 at 8:26 PM

Attendees

You responded "Accept"

Accepted: 3

Clark, Charles R
Required

K kaileycozart@gmail.com
Required

Iamarino, Eric L
Optional

Tentative: 1

Fernandez, Victor C
Required

Didn't respond: 5