Project Writeup

Background

This submission for the MATE 2024 Computer Coding Challenge uses a fine-tuned yolov8 model to identify and locate individual brittle stars (*O. ophiura*), a type of starfish, in images and video.

The yolov8 family of models are pretrained on internet scale image datasets to develop generalized object detection capabilities and can be easily fine-tuned to develop proficiency in downstream tasks.

Thanks to the availability of reliable pretrained object detection models and architectures, the primary challenges faced in this project were two-fold: creating a high quality dataset for fine-tuning and empirically tuning model hyperparameters.

Due to the high number of starfish in each frame (roughly 200 instances) and the statement that the model will be validated against other data similar in distribution to the example clip, we decided that it made the most sense to fine-tune the model exclusively on the provided test video. Due to the short nature of the provided video, this decision created a concern about the model overfitting to the training data.

Model Development

Training data

Due to the large quantity of starfish across all the frames of the video (roughly 300,000 total instances), manual data annotation was considered infeasible. Creation of training data was originally attempted using a zero-shot object detection model called Grounding DINO. The results of running this model with the prompt "starfish" yielded a dataset of mediocre quality, which was sufficient to start experimenting with model training.

Training data generation was revisited upon realizing that dataset quality was the main limiting factor of performance in our first experiments. We developed tooling to manually bulk edit image annotations, and manually labeled a handful of video frames ($n = 4$) to use as input to the recently released promptable video tracking model Segment Anything 2 (SAM 2). This model allowed us to extrapolate our annotations into past and future frames in the video, reducing the amount of annotations that needed to be created manually by a factor of 450 times. However, we encountered hardware resource limitations when attempting to run SAM 2 on the full video clip. The large number of starfish instances caused the program to encounter both insufficient RAM and VRAM issues. This was overcome by running SAM in batches of 30 instances at a time, and merging the results of all the runs into a single dataset. After reviewing the dataset created by SAM 2 and manually removing spurious annotations, we deemed the data set to be of high enough quality proceed further with model training.

We considered exploring additional techniques to generate and filter our training data, such as using a more modern alternative to DINO, but we were unable to pursue these other ideas due to limited time constraint as well as perceived diminishing returns.

Model Training

During our experiments, we conducted eight training runs, and evaluated 831 combinations of models checkpoints and inference parameters. We trained three different sizes of models, nano, small, and medium, ranging from 3.2M parameters to 25.9M parameters in size. Although the largest model achieved the best scores on our evaluation metrics, we observed evidence of overfitting in this model, meaning that the model was memorizing our training dataset and would likely fail to generalize well to unseen images. Because of this, we choose to submit one of our "small" models with a parameter count of 11.2M parameters. We found this model size to be well-balanced in terms of performance and compute cost, and its smaller size reduced its tendency to overfit.

Due to how we generated our training data using SAM 2, annotations on some starfish near the left and right edges of certain frames were missing from the dataset, to prevent the model from incorrectly learning this, we decided to train it in two phases: first on a cropped data set that excluded the edges of the frame where annotations could be missing, and second on a data set of cherry-picked full-size frames without annotation errors. We observed that this training curriculum outperformed our other experiments which were trained only on cropped, cherry-picked, or unaltered frames.

Other than size and data quality, we also experimented with different values for image resolution and "dropout". We found that training models at an image size of 960x960px allowed them to outperform models trained with the default for yolov8 models of 640x640px. We hypothesize that extra performance comes from the extra information in the larger size and the reduced scaling artifacts from the larger size evenly dividing the source video resolution. As for dropout, this is the value determines what fraction of the model's weights, if any, are randomly zeroed or dropped out during each training epoch. Counterintuitively, this can improve a model's performance by limiting the model's ability to overfit and memorize its training data, therefore improving its ability to generalize to unseen data. For each of our training runs, we somewhat arbitrarily a value for dropout in the range of 0.1-0.5 depending on the size of the model and the diversity of its training data.

Model Evaluation

We evaluated our models using a mix of subjective and quantitative measures. For each epoch we saved from our training runs, we calculated a metric for its inter-frame consistency and accuracy as compared with manually annotated frames ($n = 4$). Using these metrics, we identified our top five best preforming

models, and used subjective review to pick the best model. During this review, we primarily focused on picking the model with the least false positives.

Custom Tooling

In the process of completing this project, several pieces of related tooling were developed, including an interactive dataset editor with cross-frame editing and a cli program for bulk manipulation and metric computation. The cli program's features include comparing, merging, and diffing datasets, matching annotations across frames (to power the editors cross-frame editing), deduplication of annotations, and computing loss and score metrics. The gui editor served as our main tool for visualizing the model's outputs, and allowed us to add, edit, and bulk delete annotations in our datasets. The bulk delete feature, which allowed us to delete occurrences of an annotation in past or future frames was particularly useful when manually cleaning the data generated by SAM 2 as the model had a tendency to produce incorrect annotations when a tracked object went out of view or became too distant.

User Guide / Software Usage

Dependencies

The inference script is written for python 3 and its dependencies can be installed with the following command

```
# Install the dependencies from PyPI
pip install ultralytics openpyxl pandas numpy argparse
```

Note 1: Additional dependencies such as pytorch and certain system libraries may be required depending on your software and hardware setup. Please refer to the documentation of each dependency for further details if installation difficulties are encountered.
Note 2: We recommend using a python venv to prevent possible dependency conflicts with other programs

Model Inference

With all the relevant files downloaded, the model can be run against video using the following command (changing file names as needed):

```
python inference.py --model model.pt --input input_video.mp4 --output output_dir/
```

Click here to view supported video/image formats.

After running the script, the output folder will contain an image sequence and a spreadsheet titled "frame_data.xlsx"

The image sequence can be converted to a video using ffmpeg with the following commands:

```
cd output_dir/
ffmpeg -framerate 30 -i out%04d.jpg -c:v libx264 -crf 18 output.mp4
```

The coordinate system used for the spreadsheet has its origin set at the image's upper left corner with positive x pointing right and positive y pointing downward where the coordinates in units of pixels. The spreadsheet structure follows the template provided. The coordinates of each bounding boxes' min and max point are stored across the four respective columns in a comma separated list where the nth item in each list corresponds to the nth bounding box.

Future Research

We believe that our submission in its current state is limited by both training data quality and the fact that the model only "sees" one frame at a time. We suspect it is possible to generate higher quality training data by incorporating more modern zero-shot object detection models into the labeling workflow such as Grounding DINO 1.5 or Florence-2. As for the model architecture, yolov8 does not support receiving context from other frames in the video. If a video native model architecture was chosen, it would likely be able to output annotations that are far more temporally consistent and may be able to detect some of the more distant starfish more reliably.