Preliminary Results of Chinook and Sockeye Counting Using Object Detection and Tracking

Moses Lurbur

December 15, 2020

1 Introduction

The Columbia River is dotted with over 400 dams that impede the migration of salmonids and other species like the Pacific Lamprey [1]. On many hydroelectric dams on the lower Columbia, fish are counted through special viewing windows as they bypass the dam.

The Chelan County Public Utility District (PUD), which manages Rocky Reach and Rock Island dam, employs people to manually count fish. Footage is recorded as fish pass through the viewing area and then reviewed the next day by fish counters who document species and whether the fish has an adipose fin. Adipose fins are commonly clipped off of hatchery Chinook salmon when they are released as juveniles, making hatchery fish more easily distinguishable.

This report reviews the preliminary results of Sockeye (*Oncorhynchus nerka*) and Chinook salmon (*O. tshawytscha*) counting with computer vision techniques, using the same footage recorded for fish counters. It is meant as a proof of concept, illustrating the feasibility of computer aided fish counting for fish passages in hydroelectric dams and other applications.

Computer aided fish counting on dams would save money, allowing more funding to be directed towards other remediation projects, like habitat restoration. It would also allow access to real-time data on fish passages, a more standardized method of data collection across dams, and the potential for more advanced analysis, like tracking the upstream migration of individual fish by identifying unique markings on individuals, sex determination and size estimation.

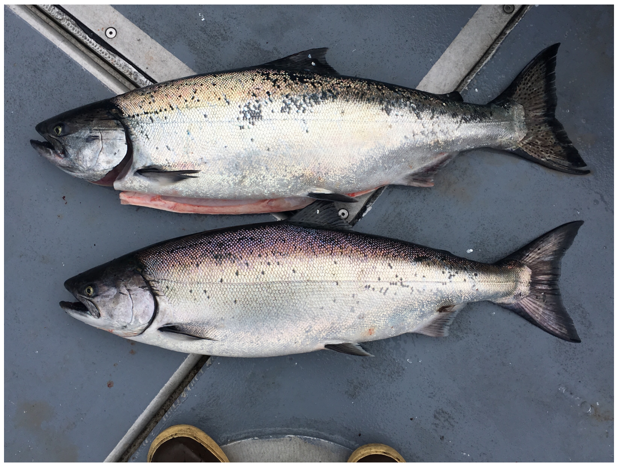

Figure 1: Chinook salmon with adipose (top) and without (bottom). Image from Wikimedia commons.

2 Related Work

The TensorFlow object detection library [3] as well as OpenCV's object tracking library were integral to this project [4]. The GitHub car counting project, IVY, provided a helpful reference for integrating object detection models, object tracking, and counting into a single application [5].

3 Method

Training and testing data were generated from recordings of fish passages obtained from the Chelan County PUD. The data was from fish passage on July 9th, 2020 and showed Sockeye and Chinook salmon. The data annotation tool MakeSense.ai was used to manually annotate video stills to build a data set of png and xml files. Images not containing fish were manually removed. Initially there were 3 classes: Chinook - adipose present, Chinook - no adipose, and Sockeye.

The pre-trained ssd resnet50 v1 fpn model was trained on the data set for 7000 training steps according to TensorFlow's documentation. Initial observations of performance on test images showed issues differentiating between wild (adipose present) and hatchery (no adipose) Chinook. Because there is no difference between the two classes (Figure 1), apart from the absence of the adipose fin, this was not a particularly surprising result.

After consulting with the Brown University Computer Vision professor, James Tompkins, the object classification problem was divided into two parts:

- Species classification
- Adipose classification

Beginning with the species classification, another training data set was generated, this time with only 2 classes: Chinook and Sockeye. The new data set contains 866 images taken from the video as stills. Ninety percent (779) were used for training and 10% (87) were used for testing. Each image, on average, contained more than one fish. It should be noted that this is a relatively small data set for training a robust object detection model, and represents less than an hour of actual fish passage footage from a single day.

The same pre-trained model was retrained on the newly formatted data set and trained for 120000 steps.

The trained model was then exported for use on a video feed. Using the rough framework provided by the IVY car counting project, the model was deployed to count Chinook and Sockeye passage in a 10 minute video clip.

3.1 Counting Method

Initially, fish were counted when their bounding box intersected the counting line (Figure 2). This method makes it difficult to detect when fish are moving ”backwards” and should be subtracted from the count of fish swimming upstream.

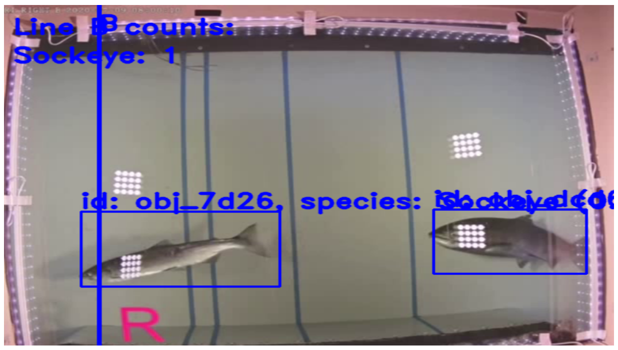

Figure 2: Sockeye being counted as bounding box crosses counting line. The count is visible in the upper left-hand corner of the frame.

To enable subtraction of fish a different counting method was created. Instead of a counting line, a counting region and object centroid were used. The centroid is the center of the rectangular bounding box that contains a fish (Figure 3).

With this new method, if a fish object's centroid is in the counting region, its direction is calculated using the past and present saved centroid values. If the fish is moving forward the fish is counted. If the fish is moving backward it is subtracted. Note that forward and backward are entirely dependent on the perspective of the camera.

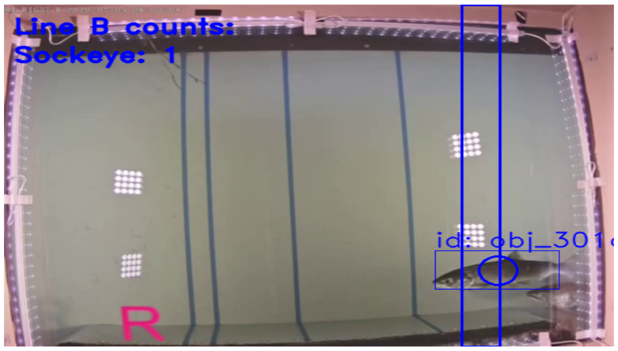

Figure 3: Sockeye being counted as centroid (show as circle) enters the counting region (defined by vertical lines). Note the difference in position of the counting line in prior image and the counting region show here. These were determined by testing accuracy of positioning across the frame.

4 Results

In the 10 minute clip used to test the performance of the model and counting program, 41 Chinook and 129 Sockeye salmon passed through the viewing window. (This ground truth value is a result of manually tallying Chinook and Sockeye.) Accuracy was evaluated based on how close the results of the counting program were to the ground truth (percent error). The best performance was a Sockeye percent error of 0.78% and a Chinook percent error of 7.32% and a total fish count percent error of 2.35%. The system counted 130 Sockeye and 44 Chinook, and 174 total fish. This result was achieved using an accuracy threshold of 0.8, a detection frequency of 10 frames, using the CRST object tracker from OpenCV and the centroid counting method (see Counting Method).

Performance was evaluated against a variety of variables, including counting method, confidence threshold of the model and the detection interval (the frequency that the object detection model was run during the video clip).

4.1 Confidence Threshold

Video analysis was performed at a range of confidence thresholds for the object detection model. Accuracy peaked at a threshold of 0.8 (Figure 4). As would be expected, at lower confidence thresholds the model tended to over-count the total number of fish and at higher thresholds the total number of fish was under-counted.

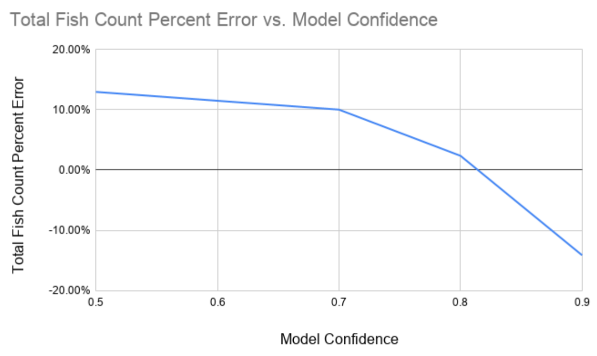

Figure 4: Total fish count percent error vs. model confidence.

4.2 Detection Interval

The optimal detection interval was found to be around 10 frames.

The main impact of varying the detection interval is time. If the object detection model is run using a detection interval of 1, that means that the model would be run on every frame of the video. The most time consuming and computationally expensive part of the counting system is running the object detection model. Object tracking, which is performed on all frames, is not as computationally expensive. Thus, the fish counter becomes faster with longer detection intervals. However, as the object detection model is run less frequently, accuracy decreases because the object tracking algorithms can't reliably follow identified fish as they are occluded or change direction quickly (Figure 5).

4.3 Counting Method

The centroid counting method should theoretically be more robust, although results don't show significant differences in accuracy between the two methods. Using the centroid method with subtraction, the fish counter reached 0.0% error for total fish count, but over counted Chinook (21.95% error) and under counted Sockeye (-6.98% error).

The lowest percent error for Chinook and Sockeye was reached using the centroid counting method without the implementation of subtraction, with 7.32% error for Chinook, 0.78% error for Sockeye and 2.35% error for the total fish count.

5 Discussion

Overall, the model appears to over-count Sockeye and under-count Chinook. Based on anecdotal evidence, the model is more likely to mistake smaller Chinook as Sockeye. Because Sockeye outnumber Chinook in the training data set by a factor of roughly 3:1, it is not surprising that the model tends to misidentify Chinook as Sockeye. This problem has a simple solution: more data.

With more data and the continued refinement of the fish counting system, a sufficiently accurate computer vision powered fish counting system is possible. That being said, a couple important challenges still remain in the fine tuning of this system.

5.1 Counting Method

The fact that the ”more robust” and theoretically more accurate counting method performed worse than the method that didn’t account for fish swimming downstream could have a variety of explanations. I suspect the primary source of error lies in the object detection model and object tracking. The model regularly misidentifies Chinook as Sockeye. This would account for some of the over counting of Sockeye and under counting of Chinook.

Double counting may also be an issue. Double counting the same fish is avoided by assigning each fish a unique id when the fish is first identified by the detection model. This id stays with the fish until it is lost by the object tracker. Ideally, the fish is lost when it leaves the frame. But the fish can also be lost if it is occluded by another fish or makes a sudden and unpredictable movement. Double counting may be occurring when a fish is counted, then lost by the object tracker, then detected again by the model and assigned a new id, all while remaining in the counting region.

This issue lies less in the model’s ability to correctly classify species, but in the quality of the object tracker. Further research and experimentation with trackers is necessary. Fortunately, the counting region is a relatively small region and tracking failures only impact fish counting accuracy if they occur within the counting region.

5.2 The Dreaded Adipose Fin

Determining the presence of an adipose fin is the most challenging part of fish counting in this application. The adipose fin is easily occluded and difficult to distinguish from other fins. Part of the classification challenge is that there must be three classes in an adipose identification model: adipose present, adipose unknown and adipose not present.

If species classification and adipose classification were to happen all in a single object detection model, for each species where adipose status is important, there must be three classes for each possible adipose situation (present, absent, unknown). This adds complexity to the model and the data set.

This was tried with a limited data set and proved to be very inaccurate based on preliminary observations. Species classification was not an issue, but classifying by adipose was very inaccurate. This isn’t very surprising because we are essentially asking a model to notice a very small difference between fish, focusing on the presence of a single, tiny fin, while also classifying the fish as a whole. Perhaps with a much larger data set and lots of training it would be possible to classify species and adipose in a single model.

Another approach would be to perform adipose classification separate from the object detection model that performs species classification. This approach would require a separate data set for training.

This second approach was attempted with a very small data set of fish images that were cropped out of the data set used to train the species classification model. Results from training were encouraging, with accuracy approaching 80%. However, a larger data set is necessary for a more robust analysis.

A more serious challenge beyond identifying the presence of the adipose fin is knowing when to count a fish.

Let’s begin with the assumption that we have a perfect object detection model, adipose classifier and object tracker. To count a fish, the adipose must not be occluded to allow the adipose classifier to identify the adipose fin as either present or absent. This means we would have to track a fish until its adipose status is determined and then count it. This could occur at any point in the video frame (or never in the worst case scenario), so the counting region would have to be very large to account for long periods of occlusion. This isn’t an issue if our object tracker is perfect, but in any other case this raises serious challenges when a fish is lost by the tracker and then re-identified with a new id. As mentioned in the Counting Method section, this would cause over counting. Any error in the adipose classifier or object detector would also result in a drop in accuracy.

5.3 Run time

All models and programs were trained and evaluated on a 2017 MacBook Air, which has never been mistaken for a high performance machine.

It took about an hour and a half to perform a fish count on about 10 minutes of footage. This time would be dramatically improved by running the system on a machine with a GPU. It's also probably possible to use a less complex model with fewer trainable parameters, which would help with speed. Run time should not be a serious issue. The current results are simply a symptom of using a machine built for browsing the web, not running machine learning models.

5.4 Real World Implementation

At least initially, an automated fish counting system lends itself to analyzing footage with low fish densities, like winter fish passage on the Columbia, because higher fish densities are more challenging to count using computer vision techniques.

According to Dave Duvall of the Grant County Public Utility District, winter footage of fish passage is usually not processed by fish counters until the Spring. This footage often includes long periods with no fish and takes days for the fish counting team to process. With a computer vision based fish counter, winter data could be analyzed and made available in real-time over the winter months, saving time for fish counters and giving biologist access to fish passage data months earlier.

Using the system on winter fish counts would also be a convenient way to test functionality and fine-tune the model, ultimately preparing it for use during times with higher fish density.

5.5 Accommodating More Species

This report describes a fish counting system that only counts two species: Chinook and Sockeye. In reality, there are many more species of fish that are counted passing through fish ladders. These include Steelhead, Coho, Pike Minnow, Lamprey and Bull Trout. Fish counters also differentiate between adult and sub-adult individuals as well as adipose presence.

This extra layer of complexity would translate to a more complex data set and model. Differentiating between adult and sub-adult individuals also raises an interesting challenge, likely solved with the incorporation of stereo cameras and length estimation.

6 Conclusion

Preliminary experimentation showed that a computer vision based fish counter for fish passage at dams is feasible. The main challenge lies in collecting more data, which would significantly improve species classification and adipose identification.

Rather than using minutes of footage, months worth of video will need to be annotated. With a larger data set, training the object detection model also becomes an issue. An investment in computing power (physical or virtual machines with GPUs) would be required.

The benefits of transitioning to a computer vision based counting technique are numerous. Aside from saving resources that can be put towards other conservation projects, these techniques would also allow for real-time data analysis by biologists, enable standardization across fish counting programs and lay the groundwork for more advanced scientific data collection, like tracking individual fish up the Columbia river based on unique markings.

References

[1] Northwest Power and Conservation Council, Fish Passage, `https://www.nwcouncil.org/reports/columbia-river`

[2] numpy. *NumPy - a fundamental package for scientific computing with Python*. 2020. 1

[3] Google Brain Team. *TensorFlow - a free and open-source software library for dataflow and differential programming*. 2020.

1

[4] OpenCV Object Detection. *https://www.pyimagesearch.com/2018/07/30/opencv-object-tracking/*. 2020. 1

[5] IVY. *https://github.com/nicholaskajoh/ivy*. 2019. 1

[6] Make Sense. *https://www.makesense.ai*. 2020.

Appendix

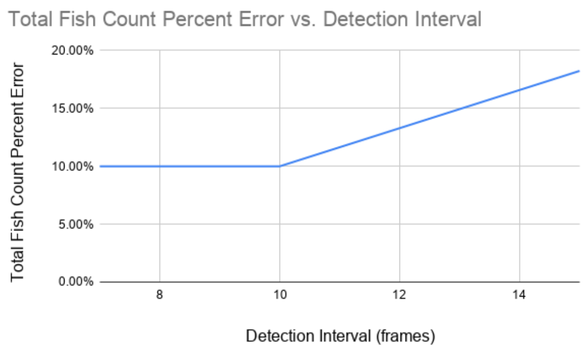

Figure 5: