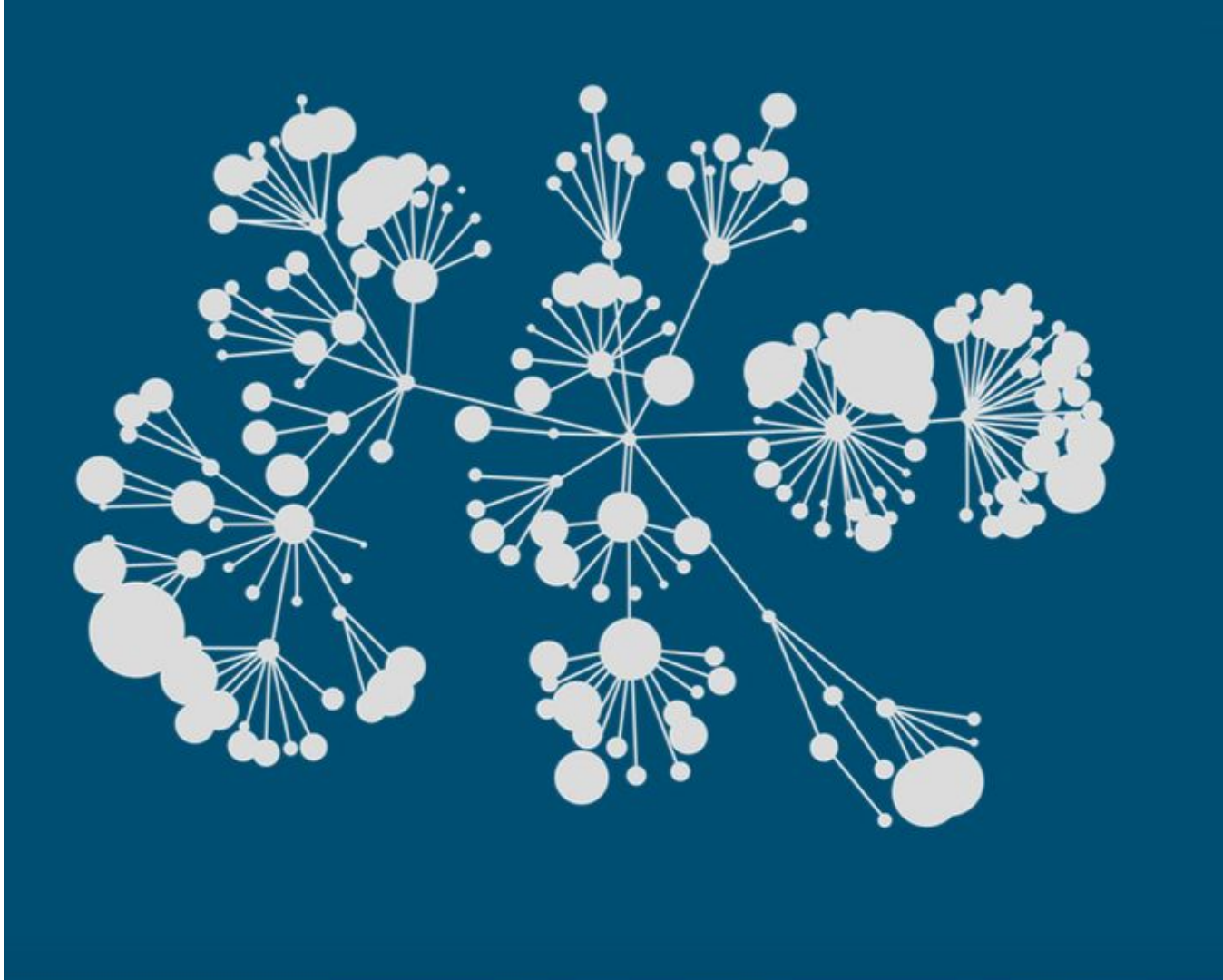
The Nature Conservancy Fisheries Monitoring

Winner Presentation

Ignas Namajūnas
Jonas Bialopetravičius
Gediminas Pekšys

kaggle



Agenda

1. Background
2. Summary
3. Training methods
4. Important findings
5. Simple model

Background

Gediminas P.: BA Mathematics (University of Cambridge), about 2 years of experience as a data scientist/consultant, about 1.5 years a software engineer, about 1.5 years of experience with object detection research and frameworks as a research engineer working on surveillance applications.

Ignas N.: Mathematics BS, Computer Science MS and nearly 3 years of R&D work, including around 7 months of being the research lead for a surveillance project.

Jonas B.: Software Engineering BS, Computer Science MS, 6 years of professional experience in computer vision and ML, currently studying astrophysics where I also apply deep learning methods.

Summary

We mostly used Faster R-CNN with VGG-16 as the feature extractor (9 models), but one of the models was R-FCN with ResNet-101.

The challenge was to train it well and not overfit the scarce training data. For this we used heavy data augmentation (rescaling, blurring, rotating by a multiple of 30 degrees, flipping, making the image more as in night-vision, varying the contrast and so on). Additionally, we selectively chose some of the external data posted on the forum.

Each of the models needs several hours of GPU training.

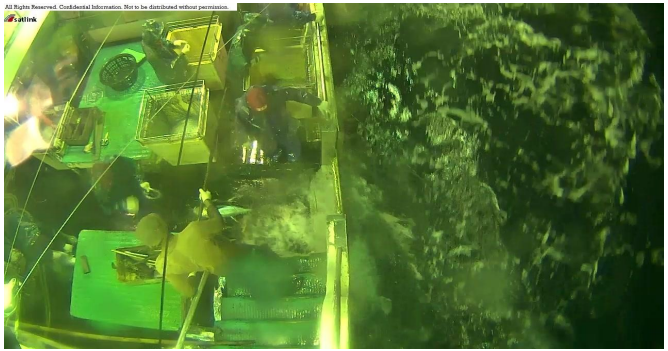
We also used some heuristics during the testing phase.

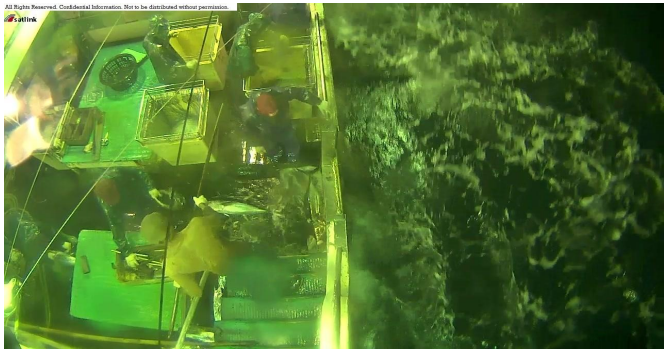
Training and Inference Methods

We trained R-CNN like detectors to detect the bounding boxes of fishes (no models were trained on the whole images). Fish confidence was usually the maximum of the detection confidence.

The predictions generated by each of the detectors were averaged using a simple arithmetic mean.

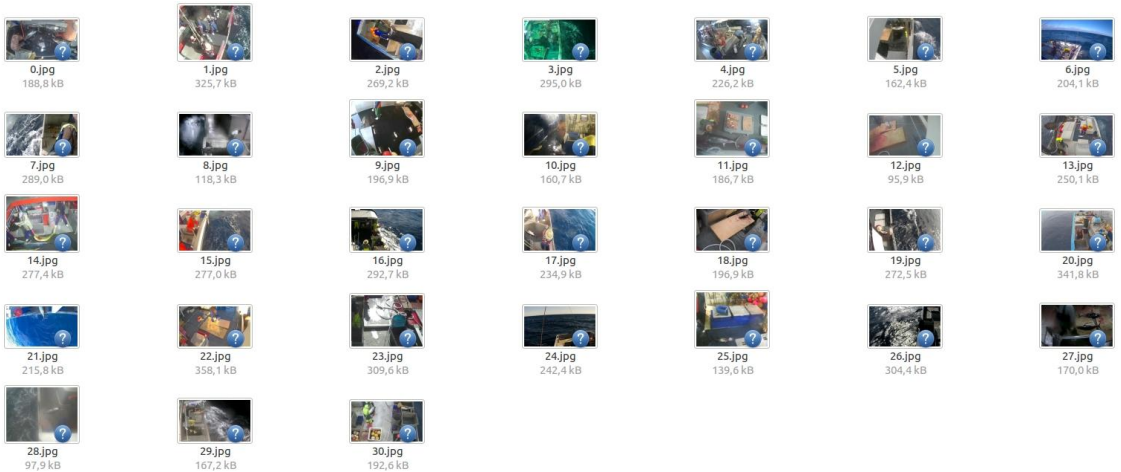
The predictions on a particular frame were averaged with predictions made on similarly looking (color histogram distance) frames.





Validation

Validation was constructed by clustering photos automatically and then fixing any mismatches by hand. This method converged to having 31 separate camera angles. Each of us selected a different subset of clusters with approximately similar cumulative distribution of classes. This helped us to focus on the ability of our models to generalise to previously unseen data relatively early. This should explain the significant jump from stage 1 performance on the public leaderboard.

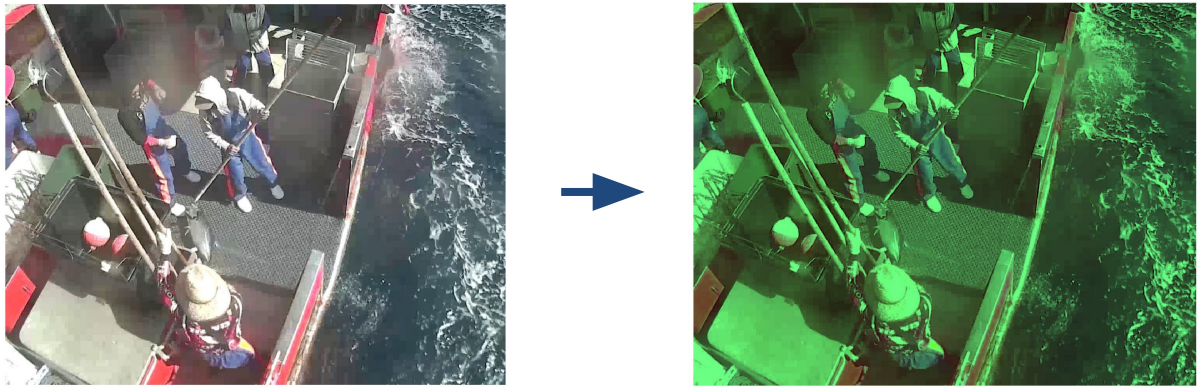


Important and Interesting Findings

The size of the fishes were of drastically different sizes. To handle this, some of the detections were made on a relatively small image, if a fish was found with high confidence - detections made on larger side lengths were given a small weight (as they included a lot of detected background noise which becomes more pronounced at higher resolutions), if not - more weight was given to probabilities generated at higher resolutions.

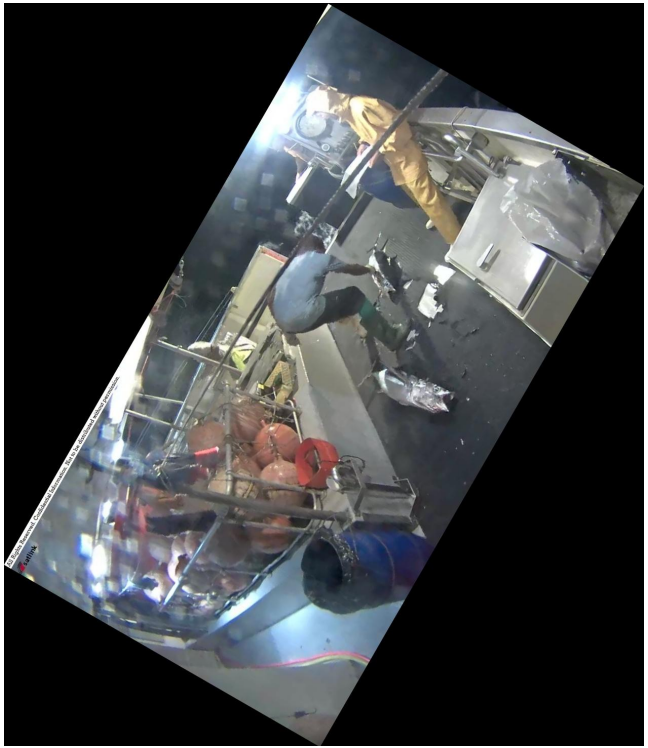
Important and Interesting Findings

A lot of the images were taken in night-vision mode. We handled this by generating more of them at training time or changing the pixel distribution of night-vision images at testing time.



Important and Interesting Findings

Lastly, we had polygonal annotations for the original training images, which we believe helped us achieve more accurate bounding boxes on rotated images, as they would have included a lot of background otherwise.



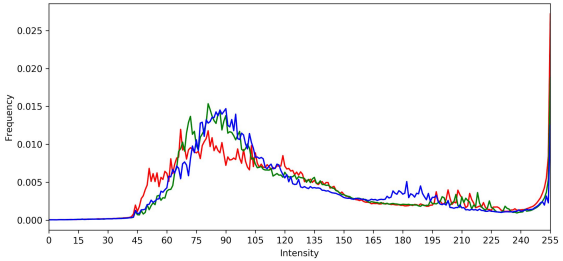
Simple Model

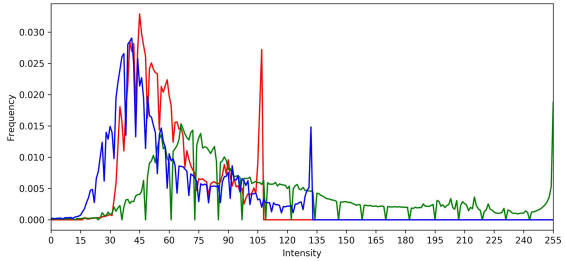
- The simplified model uses heavy night-vision augmentation: making regular photos look like they’ve been filmed in low light conditions

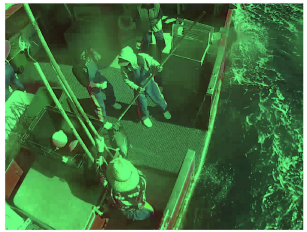
- After the competition we’ve been able to reach 1.049 log-loss with this model alone (instead of 1.062, which was our winning submission)

Simple Model

- While training randomly pick 30% of “normal” photos (these are easy to identify from histograms)
- Stretch their histograms to approximately match real night-vision photos (let’s pretend they’re Gaussian)
- Add a random stretch factor for each color (up to 50% in the best model)







Final word

Our proposed solution is probably not practical enough – the log loss is equivalent to always giving the correct class around 34%.

We hope we at least made a proof of concept – in principle, machine learning can solve this problem.

Much better performance could be achieved with more training data, however, it is important to ensure the diversity of the samples, so as not to have many near-identically looking fishes.

Thank you for organizing this competition, we definitely learned a lot!

Jonas, Gediminas, Ignas

kaggle™