CS230

Final Report: Identifying Active Surfers from Surf Camera Footage to Enable Better Surf Forecasting Tools

Joseph L. Mann
Stanford University
jlmann@stanford.edu

Daniel K. Angell
Stanford University
dkangell@stanford.edu

Nihat Cem Inan
Stanford University
ncinan@stanford.edu

1 Introduction to Problem

The ultimate goal of surf forecasting is to transform data (wave height, period, direction) gathered from a complex network of buoys across the ocean into a spot-by-spot temporal prediction of the quality of breaking waves. Excitingly, there exists a system of equations to model swell as it moves through deep-water ocean systems, and can be used to, very successfully, track the time frames of swell arrival and disappearance.[1] However, the quality of the wave (speed of wave breaking, homogeneity of speed, length of ride able portion, size of wave, availability for tricks, availability for barrels *etc.*) are largely determined by the underwater topography (bathymetry) and the amount and direction of local wind at near the surf spot at the time of breaking.[2] Thus the quality of the wave becomes a complex and difficult to model function of the swell that generates it (information available from buoy systems), the current tide conditions (information available from tide charts), and the wind conditions (information available from anemometer data). A much easier route to predicting wave quality from a multitude of input parameters is to use machine learning algorithms. However, for this to be implicated, there needs to exist an accurate way to measure the quality of waves.

Currently, wave quality measurement is a subjective approach in which a surf forecaster watches a camera of breaking waves for a short duration of time and makes an immediate classification of the conditions (for example, the leading surf forecasting service, Surfline, employs a rating system of poor, fair, good, and the always coveted, epic).[3] Not only is this classification system subjective to the forecasters own human bias, but it is formed from a relatively short visual analysis of an environment that is rapidly changing. We envision that a less subjective, robust, system of wave classification is needed to assess the quality of waves to enable machine learning algorithms to successfully and reliably match ocean and atmospheric conditions to the quality of wave. The development of this tool would enable future wave classification (size, speed, homogeneity, *etc.*) into a robust analysis for machine learning assisted forecasting. To do this we will set to implement a video classifier that can identify a surfer who is currently surfing on a wave, separate from a surfer who is sitting in the water. By detecting the surfer on the wave over time and converting the surfer's position over time into a vector we can begin to quantitatively describe the movement of a wave (because a surfer can only surf a surfable wave).**The project goal is to accurately be able to detect a surfer surfing at 1 location with high enough accuracy and precision to enable object tracking. A supplementary goal of this project (beyond the scope of this report) is a location agnostic surf detector. The ultimate goal of this project (outside the scope of this report) is to train a separate model to predict wave conditions with less bias by using the information from this model**.

2 Methods

2.1 Dataset and Methods

We established a pipeline to generate our dataset for training. We downloaded videos manually from Surfline.com, and cut the video at 2 frames per second regardless of if a surfer is surfing or not. We manually labeled the images using Labelbox. The output from Labelbox is a JSON file which

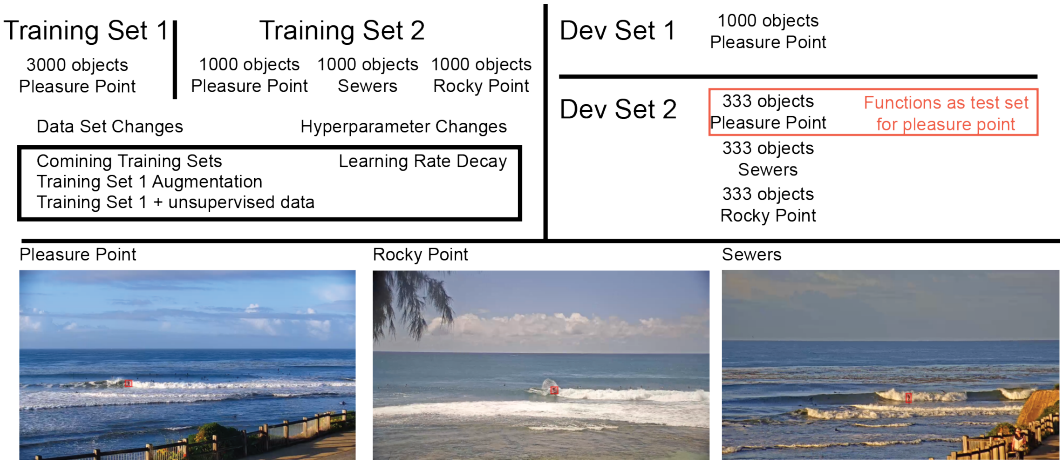


Figure 1: **Training and Dev image description with dataset and hyperparameter modifications**

contains URLs to the stored images (AWS S3 storage) and the bounding box coordinates associated with each image. This JSON is then transferred into labels (COCO format) for implementation with the Github repository.

We only label instances of surfers actively surfing. Our dataset is composed of images from Pleasure Point (California), Rocky Point (Hawaii), and Sewers (California), and was chosen to include footage from different weather conditions (sunrise, sunset, mid day sunny, mid day cloudy). We generated two training sets: (1) 3000 objects (surfers) from pleasure point in 2022 images and (2) 1000 objects from (each) pleasure point, sewers and rocky point in 2538 images (Figure 1). We similarly generated two dev sets: (1) 1000 objects from pleasure point in 674 images and (2) 333 objects from (each) pleasure point, sewers and rocky point in 786 images. The separation of the training sets and dev sets allow us to explore which data collection strategy is most effective at training a model (with a small amount of data) for (1) a location-specific model and (2) a location agnostic model. We use the 333 objects at Pleasure Point as a test set for a camera specific model, as the training only allows the epoch-dependent training of 1 dev set. To further explore the creation of a camera agnostic trained model, we include 361 objects (329 images) and 341 objects (321 images) from Snapper Rocks (Australia) and Kirra (Australia) as a test set

2.2 Model Architecture

The computer vision network we chose to implement is a derivative of You Only Look Once (YOLO). [4] We chose the YOLO-V3 implementation with a spatial pyramid pooling layer because it demonstrated the highest performance on the COCO dataset of the available, built YOLO networks[8,9]. We are using an implementation in pytorch that has modular control over hyper-parameters, detection parameters, and yolo configurations.[6] Due to the limited size of our datasest we chose to approach the problem using transfer learning, where we could leverage pretrained low-level feature knowledge that is common to many photos, and focus on training the final detector. In our transfer learning implementation, all layers contain weights pretrained on the COCO dataset. The weights of all but 3 layers are fixed in the training phase, which correspond to the 3 different classification layers within the CNN (Appendix Figure 6 shows an example of this). Pretrained weights should be able to detect a surfer, surfing, and classify it with a label. This would indicate that the lower and higher level features embedded in the network are already apt to detect the features of a surfer. An ideal pretrained weight would also not detect a surfer waiting for a wave, as these are not objects we want our detector to detect. We chose to implement weights from this repository because it was able to detect, a surfer surfing as a human, albeit with extremely poor accuracy. Moreover, it would not classify surfers sitting in the water as persons. Weights trained on the COCO dataset achieve these criteria (Appendix Figure 7).

The loss function employed in this model is a Generalized Intersection over Union (GIoU) loss function, which incorporates both the intersection over union and a penalty for the smallest

convex hull between the ground truth and detected object, useful in differentiating when there is no intersection between the prediction and ground truth.[10] The model employs an adam optimizer for gradient descent. The standard learning rate decay for this model is $\alpha_0 : \alpha_f$ 0.01 : 0.0005. The model employs hue saturation value augmentation, which randomly makes minor alterations to the hue, saturation, and color value, to prevent over-fitting. Our minibatch size was maximized such that the optomizer could run with our available memory (either 8 or 16 images per minibatch).

2.3 Proposed Tuning

In the exploration of training (1) a model that functions at 1 location (pleasure point) and (2) a location agnostic model we propose to (i) compare training set 1 and 2 (ii) combine training set 1 and 2 and compare to individual training set 1 and 2 data (iii) augment training set 1 (image rotation (random, less than 5 degrees) and image scaling (random, less than 5% increase and decrease) (iv) generate a larger training set 1 by introducing 10,000 new "dirty" images from pleasure point labeled using the training set 1 model (unsupervised) and (v) diminishing the learning rate decay ($\alpha_0 : \alpha_f$ 0.01 : 0.0005 (standard) to 0.01 : 0.005 (new)).

3 Results and discussion

3.1 Selection of Dataset

Comparing two different training datasets. Purpose: We explore which training set (1,2) performs best at an individual location, or multiple locations. **Training Regimen:** 200 epochs, start with pretrained weights on COCO dataset, select best performing epoch (F1 score) on dev set 1 for analysis (training set 1 - epoch 133, training set 2 - epoch 197). **Results** Training set 1 performed better (F1 + mAP@0.5) on dev set 1 (Figure 2**A**). Both training sets seemed to realize only small improvements after epoch 100. Training set 2 performed better on dev set 2 (F1) (Figure 2**B**), because it performed drastically better at locations that training set 1 hadn't seen (Figure 2**C**).

Figure 2: **Comparing Models trained with Training set 1 and 2. A** Performance metrics on Dev Set 1 from training set 1 and training set 2 as a function of epoch. **B** F1 performance metric on Dev set 1 and Dev Set 2 from training set 1 (epoch 133) and training set 2 (epoch 197). **C** F1 performance metric on images (categorized by location) in Dev Set 2 from training set 1 (epoch 133) and training set 2 (epoch 197).

Exploration of combined dataset training. Purpose: We explore if we can improve training set 1 on dev set 1 and 2 by including training data from different locations. **Training Regimen:** Start with weights from best epoch from training set 1 on dev set 1 (epoch 133), train until 200 epochs (67 more) with 5000 objects (3000 from training set 1, 2000 from training set 2). **Results:** Introduction of new data increases training loss value , which asymptotes higher than without new training data (Figure 3**A**). Introduction of new training data does not make dramatic improvement (F1 metric) on dev set 1 (Figure 3**B**) but improves performance on dev 2 locations to the original performance of training set 2 (Figure 3**C**).

3.2 Application of Image Augmentation

Purpose: Dataset 1 under-performs when surfer is performing maneuvers (compressed and rotated body); artificially rotate and compress surfers to improve data. **Training Regimen:** Start with

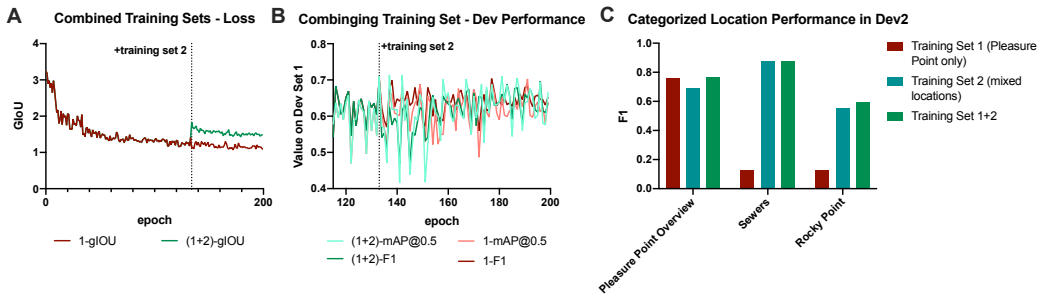


Figure 3: **Combined training data performance.** **A** GIoU loss on the training set 1 and training set 1 after introduction of images from training set 2 (Rocky Point (1000 objects), Sewers(1000 objects)) at epoch 133. **B** Performance metrics on Dev Set 1 from training set 1 and training set 1 after introduction of training data from training set 2 at epoch 133 as a function of epoch. **C** F1 performance metric on images (categorized by location) in Dev Set 2 from training set 1 (epoch 133), training set 2 (epoch 197), and training set 1 after introduction of training set 2 data at epoch 133 (epoch 200).

weights trained on COCO, train until 104 epochs, select best performing epoch (F1 score) on dev set 1 for analysis (epoch 104), compare to training set 1 and 2. **Results: Failed Hypothesis.** Introduction of augmentation lowers performance on dev set 1 (F1, mAP@0.5) on every epoch compared to training set 1 (Figure 4**A**). However, augmentation improves performance (F1) on location data (Dev set 2: Rocky Point, Sewers) not seen in the training set(Figure 4**B**).

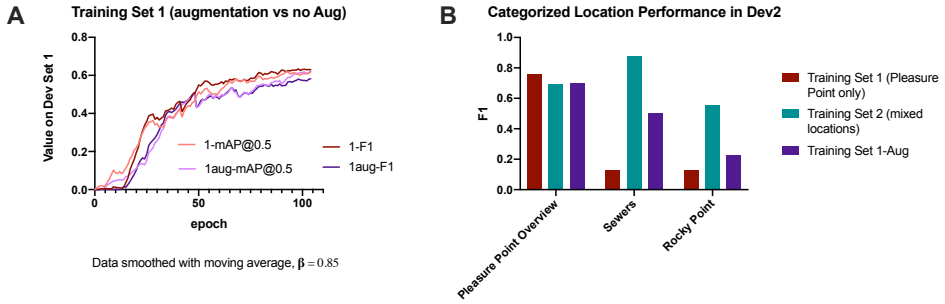


Figure 4: **Image augmentation in Training Set 1 performance.** **A** Performance metrics on Dev Set 1 from training set 1 and training set 1 with image augmentation ($<= |5|^\circ$ degree rotation, $<= 5\%$ scale) (purple) as a function of epoch. **B** Categorized location performance from model trained on training set 1 (epoch 133), training set 2 (epoch 197), and training set 1 with image augmentation (epoch 104).

3.3 Unsupervised Dataset Supplementation

Purpose: Improve performance on pleasure point location (dev set 1) through more instances of surfers that the model has not seen without hand labeling data. New labels are generated from best performing training set 1 model on new images. **Training Regimen:** Start with weights from best epoch from training set 1 on dev set 1 (epoch 133), train until 160 epochs (27 more, epochs are longer, include more data) with 10,000 new "dirty" labeled images in the training set. Perform analysis with final epoch (160). New images are from Pleasure Point, labels generated from model trained on training set 1 (epoch 133) at confidence interval p > 0.2 (selected to maximize F1 on dev set 1). **Results: Failed hypothesis.** While new training data lowers training loss function (Figure 5**A**), it does not substantially improve performance on dev set 1 (Figure 5**B**) or images at Pleasure Point in dev set 2 (pseudo-test set, Figure 5**C**).

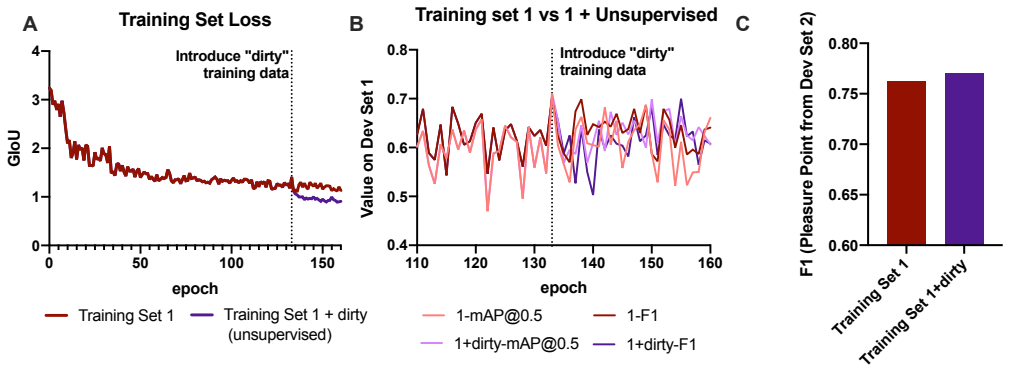


Figure 5: **Unsupervised data supplementation performance. A** GIoU loss on the training set 1 and training set 1 after introduction of "dirty" (unsupervised) training data at epoch 133. **B** Performance metrics on Dev Set 1 from training set 1 and training set 1 after introduction of "dirty" (unsupervised) training data at epoch 133 as a function of epoch. F1 performance metric on images from Pleasure Point in Dev Set 2 (test set) from training set 1 (epoch 133) and training set 1 after introduction of "dirty" (unsupervised) training data at epoch 133 (epoch 160).

3.4 Changing Learning Rate Decay

Purpose: We are worried that asymptotic performance behavior (training set 1 on dev set 1) around epoch 100 is due to learning rate decay, not a global minimum. We investigate this by diminishing the effect of learning rate decay ($\alpha_0 : \alpha_f$ 0.01 : 0.0005 (standard) to 0.01 : 0.005 (new)). **Training Regimen:** Start with weights trained on COCO dataset. Train until 135 epochs with training set 1. Compare training set loss and dev set 1 performance (F1) to standard learning rate decay. **Results: Failed Hypothesis.** Qualitative curve of the loss function and F1 metric as a function of epoch are very similar (Figure 8), indicating that they are both approaching the same global minimum for the training set - that the asymptote is not due to too low of a learning rate missing the minimum.

3.5 Testing on Unseen Locations

Purpose: Explore the extent to which our different trained models function as a location agnostic surfer detector by testing on images from locations not encountered (Figure 9**A**). **Results:** Our models perform worse on images from locations they have not seen (Figure 9**B**). Models trained with diverse data (Training set 2 and Training set 1+2) perform the best. Models perform better at Snapper Rocks than Kirra, likely due to the camera angle; the camera in Snapper Rocks is positioned similar to cameras from other locations.

4 Conclusion

Our model works well (this is it functioning while Joseph Surfs a wave - `https://photos.app.goo.gl/wrxe2SuoK8PMpfKC6`). However, it can still be improved. Through this process we have gained insight into transfer learning using small datasets. To detect surfers from a location, it is important to have adequate training data from that location (Training set 1 performs better on dev set 1 than training set 2, no trained models performed particularly well on locations they hadn't seen). Including training data from other locations does not hinder performance on a given location, although it doesn't particularly help (addition of training set 2 to training set 1 did very little). Unsupervised trained data and augmented data do not improve or detract from the performance at a given location. Not all cameras are created equal; Sewers represents an easier location to train on, Rocky Point represents a more difficult location to train on - this is likely due to the homogeneity in size of surfers that can be detected at sewers, as opposed to rocky point. We envision this model can be improved with error analysis and more training data. However, while training we noticed that, even as trained camera watchers, it was difficult to detect an instance of a surfer surfing in a still frame. However, when a surfer was surfing in video, there was no doubt about the surfers activity. We hypothesize a four dimensional CNN would better be able to detect surfers through learning easier to recognize patterns of temporal motion, but present a greater labeling challenge.

References

[1] Knauss, J.A. (1997) *Introduction to Physical Oceanography, 2nd Edition.* Illinois.

[2] Scarfe, B. E.; Alwany, M. H. S.; Mead, S. T. (2003) The Science of Surfing Waves and Surfing Breaks - A Review *UC San Diego: Library – Scripps Digital Collection*.

[3] Agent, G. (2020) How does Surfline determine surf heights and quality ratings? *https://support.surfline.com/hc/en-us/articles/360020048531-How-does-Surfline-determine-surf-heights-and-quality-ratings-*.

[4] Redmon J.; Divvala S.; Girshick R.; Farhadi, A. "You Only Look Once: Unified, Real-Time Object Detection" 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[5] Redmon J.; Farhadi, A. "Yolov3: An incremental improvement" arXiv preprint arXiv:1804.02767, 2018

[6] Jocher, G. YoloV3. https://github.com/ultralytics/yolov3

[7] Freeston, B. Intelligent Surf Cameras. https://medium.com/surfline-labs/intelligent-surf-cameras-fce6e7e3d03e

[8] Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." European conference on computer vision. Springer, Cham, 2014.

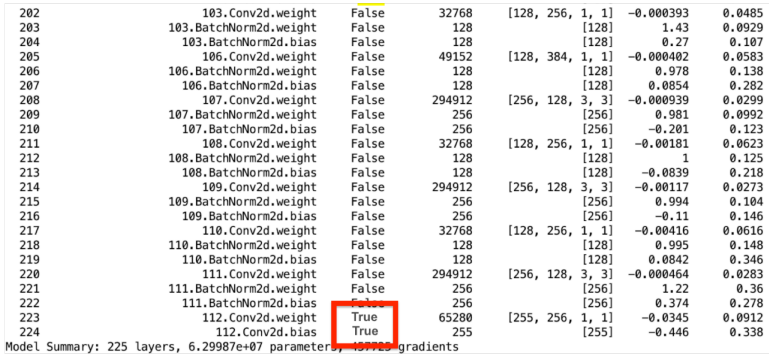
[9] Huang, Zhanchao, et al. "DC-SPP-YOLO: Dense connection and spatial pyramid pooling based YOLO for object detection." Information Sciences (2020).

[10] Rezatofighi, Hamid et al. "Generalized Intersection over Union" The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019).

5 Contributions

J.L.M., D.K.A, N.C.I formulated project idea, labeling protocol, and labeled data. J.L.M. and D.K.A. worked out data pipeline for training. D.K.A installed training architecture and layout. J.L.M., D.K.A trained data. D.K.A generated unsupervised labeled data. J.L.M. analyzed and worked up data. J.L.M., D.K.A, N.C.I discussed results. J.L.M. (introduction, methods, results, conclusion,appendix), D.K.A. (introduction, methods, conclusion, appendix), and N.C.I.(introduction) contributed to writing.

6 Appendix Figures



202	103.Conv2d.weight	False	32768	[128, 256, 1, 1]	-0.000393	0.0485
203	103.BatchNorm2d.weight	False	128	[128]	1.43	0.0929
204	103.BatchNorm2d.bias	False	128	[128]	0.27	0.107
205	106.Conv2d.weight	False	49152	[128, 384, 1, 1]	-0.000402	0.0583
206	106.BatchNorm2d.weight	False	128	[128]	0.978	0.138
207	106.BatchNorm2d.bias	False	128	[128]	0.0854	0.282
208	107.Conv2d.weight	False	294912	[256, 128, 3, 3]	-0.000939	0.0299
209	107.BatchNorm2d.weight	False	256	[256]	0.981	0.0992
210	107.BatchNorm2d.bias	False	256	[256]	-0.201	0.123
211	108.Conv2d.weight	False	32768	[128, 256, 1, 1]	-0.00181	0.0623
212	108.BatchNorm2d.weight	False	128	[128]	1	0.125
213	108.BatchNorm2d.bias	False	128	[128]	-0.0839	0.218
214	109.Conv2d.weight	False	294912	[256, 128, 3, 3]	-0.00117	0.0273
215	109.BatchNorm2d.weight	False	256	[256]	0.994	0.104
216	109.BatchNorm2d.bias	False	256	[256]	-0.11	0.146
217	110.Conv2d.weight	False	32768	[128, 256, 1, 1]	-0.00416	0.0616
218	110.BatchNorm2d.weight	False	128	[128]	0.995	0.148
219	110.BatchNorm2d.bias	False	128	[128]	0.0842	0.346
220	111.Conv2d.weight	False	294912	[256, 128, 3, 3]	-0.000464	0.0283
221	111.BatchNorm2d.weight	False	256	[256]	1.22	0.36
222	111.BatchNorm2d.bias	False	256	[256]	0.374	0.278
223	112.Conv2d.weight	True	65280	[255, 256, 1, 1]	-0.0345	0.0912
224	112.Conv2d.bias	True	255	[255]	-0.446	0.338

Model Summary: 225 layers, 6.29987e+07 parameters, [illegible] gradients

Figure 6: **Training Model using transfer learning. A** Here we show how we set the final classification layer to be updated via training. "True" indicates we are updating the weights, while where the rest of the layer weights are "False". Image used from github repository.[6]

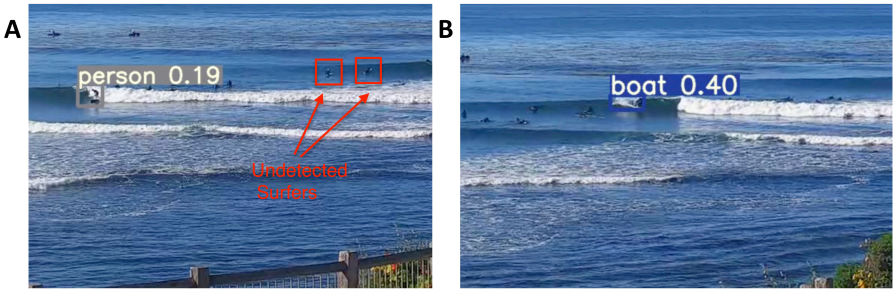


Figure 7: **Yolo out of the box weights A** Example of yolo detecting surfer with pretrained weights, while ignoring surfers sitting in the water. **B** Incorrect detection of boat instead of surfer

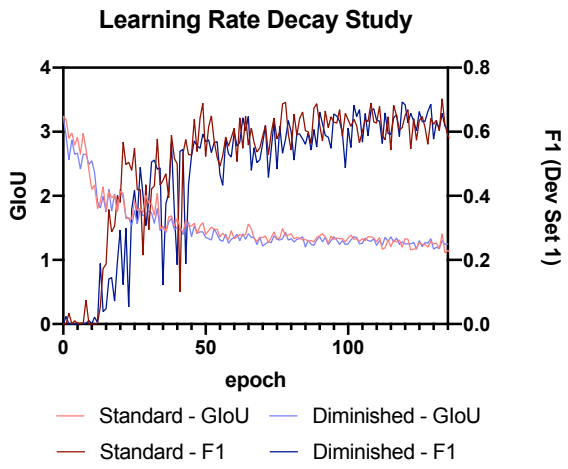


Figure 8: **Testing effects of diminished learning rate decay.** GIoU (left axis) and F1 on Dev Set 1 (Right axis) plotted as a function of epoch for standard and diminished learning rate decay.

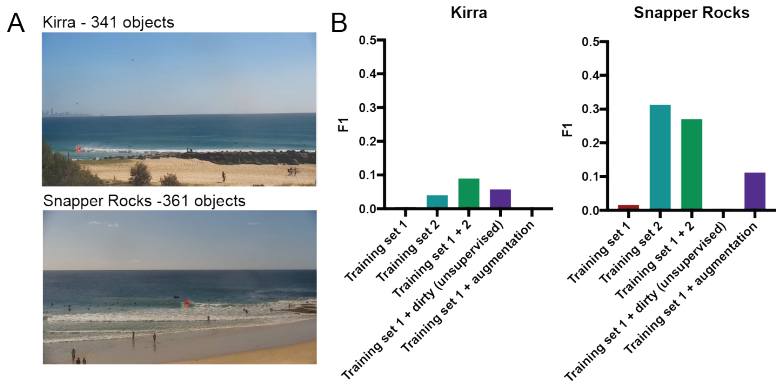


Figure 9: **Testing trained models on unseen locations. A** Example labeled frames from Kirra (341 objects in test set) and Snapper Rocks (361 objects in test set). **B** F1 performance metrics from each of the trained models on Kirra and Snapper Rocks.