Quick overview my solution: UNet + regression on masks

posted in NOAA Fisheries Steller Sea Lion Population Count 8 months ago

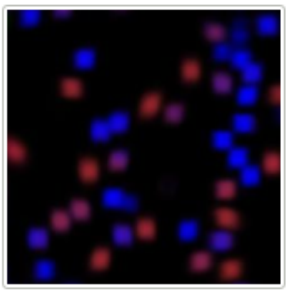
55

Hi, congrats to @outrunner for an amazing performance, and congrats to everyone who participated! Special thanks to @threeplusone for the lion coordinates!

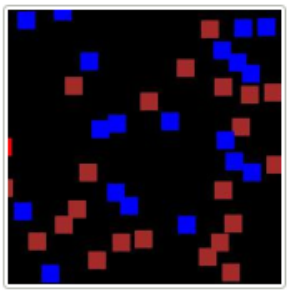
Quick overview of my solution:

UNet with batch normalization and upsampling instead of deconvolution, trained on 256x256 patches with scale (about 0.8 -- 1.6) and rotation augmentation. With probability 0.2 patches were sampled close to some random sea lion. The task was to predict fixed sized squares centered at lion coordinates, softmax output with log loss was used. Here is what it looked like:

00-13-input.jpg

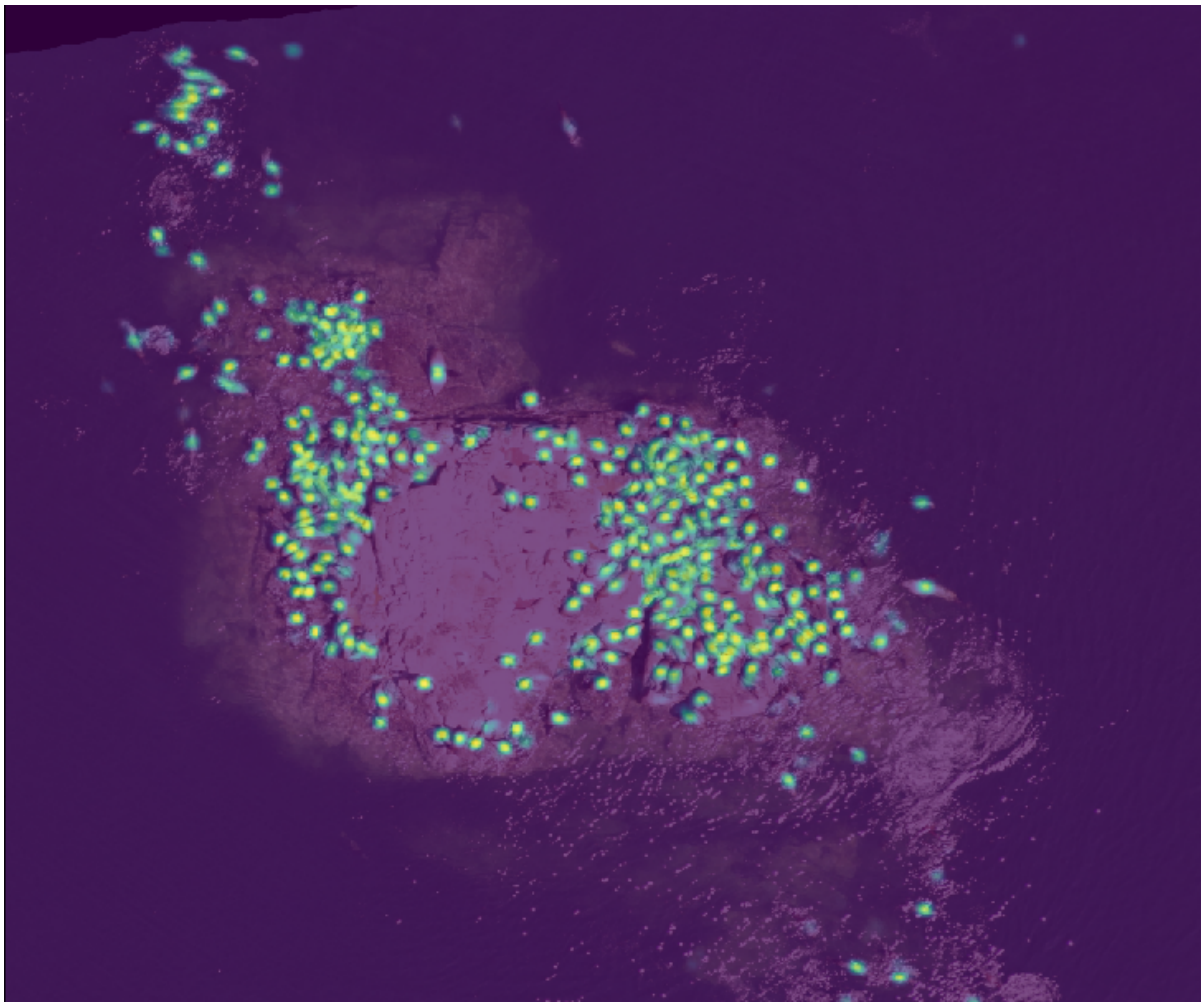


00-13-output.jpg



00-13-target.jpg

And here is what predictions (here all classes are combined) looked like:



In order to **predict the lion count**, patches of about the same size (240x240) were extracted from predicted masks, and several features were extracted: mostly sum of outputs (with several thresholds) and also blobs were detected and counted. This features were extracted from predictions of each class separately, and for each class a separate regressor was trained (an ensemble of two extra trees regressors from sklearn and one xgboost regressor).

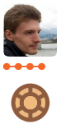
The only **trick for Test** was to predict at 0.5 scale (so I downscaled test 2x) - I determined this value by just looking at the images.

All in all I was very unsure of the leaderboard score as it often did not correlate with my local validation, so I didn't do any per-class LB tuning. For the final submission I averaged several models that were better on the LB (mid-13 public LB / mid-12 private LB), and some models that were better on my validation (they gave about 14 on public LB / 13 on private LB).

Training took about 12-24 hours on a single 1070, and prediction took another 12 hours. Plus about 3 hours for training and running count prediction.

Russ W • (5th in this Competition) • 8 months ago • Options • Reply ▲ 3 ▼

Excellent work Konstantin, congratulations and thanks also for sharing a lot on forum throughout the competition. Wondering if you might be willing to try outrunner's simple and elegant postprocessing trick: add 50% juveniles, subtract adult_females with the same amount, and add 20% pups. Does this improve your public/private scores to his amazing levels? For us it improved public by 2.3 and private by 1.8. You can likely tune the percentages even better given the powerful properties of your nice UNet model.



Konstantin Lopu... • (2nd in this Competition) • 7 months ago • Options • Reply ▲ 2 ▼

Thanks Russ, I really liked your solution, I think it's the most principled and robust of the ones shared due to explicit accounting for scale.

Finally got to check this clever hack, a milder version of it (just increasing pups by 20%) also gives a moderate boost: 12.5 -> 12.0 on private (and 13.2 -> 12.9 on public), but the optimal settings are probably different for my model.

Artem.Sanakoev • (4th in this Competition) • 8 months ago • Options • Reply ▲ 1 ▼

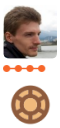
Kostia, Which UNet specifically did you use? Based on VGG ?

Konstantin Lopu... • (2nd in this Competition) • 8 months ago • Options • Reply ▲ 3 ▼

Yes, I would say it's a classical UNet, and it's similar to bn-VGG, just conv3s everywhere. It's almost the same as this https://github.com/lopuhin/kaggle-dstl/blob/master/models.py#L188 with a small twist: it has 4x pooling instead of 2x in the "deepest" part - this save a bit of memory and in theory gives a larger receptive field.

Artem.Sanakoev • (4th in this Competition) • 8 months ago • Options • Reply ▲ 1 ▼

Could you please elaborate a little bit on blob detection and counting?

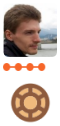


Konstantin Lopu... • (2nd in this Competition) • 8 months ago • Options • Reply ▲ 2 ▼

Sure - I used blob_log from skimage to detect blobs http://scikit-image.org/docs/dev/api/skimage.feature.html#skimage.feature.blob_log and then generated two kinds of features from them: "blob-sum" is the sum of probabilities at blob centers (so if we have two blobs with max probability 0.3 and 0.05 the value would be 0.35), and "blob-count" is just the number of blobs (so here the value of the feature would be 2). I also did this for several blob thresholds (minimal blob intensities). In order to speed things up and to save space, I saved 4x downscaled prediction, and ran blob detection at this scale too, so it was not so slow.

GarethJones • (67th in this Competition) • 8 months ago • Options • Reply ▲ 0 ▼

Nice work. Congrats on getting UNET to work so successfully. I've tried it out on a couple of projects and have found it to be a pain in the arse to train :)



Konstantin Lopu... • (2nd in this Competition) • 8 months ago • Options • Reply 3

Thanks! Yeah, training UNet can be painful!

Forgot to mention one more difference from vanilla UNet: I'm using upsampling instead of deconvolutions, on other problems it was much more robust and produced better looking masks.

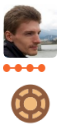
I also could not make UNet to work properly with dice loss (which should work better in case of class imbalance), and also UNet that predicted 4x smaller output also didn't work so well (which is a pity because it was much faster and more convenient).

KapilYadav • (359th in this Competition) • 8 months ago • Options • Reply 0

Congratulations. Thanks for sharing the details both before and after the competition deadline :). It helped us a lot. I was working a similar approach but was not getting good accuracy. Would you mind answering a few questions -

1. How was the performance with deconvolution layers instead of upsampling?
2. I am guessing, you used 6 layer masks as Unet output with cross entropy without weighs for your Unet.
3. Did you encounter problems with misclassification/ crowded sea lions ? (I was using Unet and most of the sea-lions were being misclassified as adult-females).
4. How was the training done ? I mean, how many epochs were trained to get these results ? Also, what kind of optimizer and lr was used? In my case, after 1-2 epochs, the log loss stops decreasing and remains constant. The output masks at this point remain inaccurate. The output masks had sea-lion shaped predictions but they were being misclassified and some other structures were also classified as sea-lions.
5. Is the purple background in the output image coming from the background label?

Thanks again and congrats.



Konstantin Lopu... • (2nd in this Competition) • 8 months ago • Options • Reply 3

Thanks Kapil!

> How was the performance with deconvolution layers instead of upsampling?

I didn't check on this task, but on DSTL Satellite image segmentation I just could not get good results with deconvolutions at all - maybe it was a problem with my implementation (because others used them just fine), or bad init - I was using PyTorch. So here I just went with what I knew worked.

> I am guessing, you used 6 layer masks as Unet output with cross entropy without weighs for your Unet.

Right. I tried applying a different weight to the background, but it resulted in more false positives and lower quality masks. On the other hand, oversampling of images with lions helped - which makes sense in retrospect.

> Did you encounter problems with misclassification/ crowded sea lions ? (I was using Unet and most of the sea-lions were being misclassified as adult-females).

Yes to both, and to be honest, I didn't manage to implement a good solution to any of those. For crowded areas maybe xgboost regressor + custom features helped a little. For misclassificatoin, I added a VGG-based classifier that tried to re-classify blobs, but I implemented it only two days before the deadline and didn't manage to get any gain from it - still curious if this would help.

> How was the training done ? I mean, how many epochs were trained to get these results ? Also, what kind of optimizer and lr was used? In my case, after 1-2 epochs, the log loss stops decreasing and remains constant. The output masks at this point remain inaccurate.

For me one epoch covered entire training set (but I sampled randomly and did all augmentations on the fly), and I got best results with training for 10-20 epochs. I used Adam with lr 0.0001 and batch size 32 (but I multiplied loss by batch size) - it was much better than SGD. I divided lr by 5 when validation loss stopped

decreasing for more than two epochs (just once usually). Both training and validation losses continued to go down, but it turned out it's better to stop training even before the best validation loss is reached - probably due to difference between train and test.

> The output masks had sea-lion shaped predictions but they were being misclassified and some other structures were also classified as sea-lions.

I've seen such shapes early in training, maybe after the first few epochs. I also saw some misclassifications and some false positives (especially on the test set), didn't get to solve it.

> Is the purple background in the output image coming from the background label?

Yes - it's just some default way of visualising masks with matplotlib, definitely not the best way to show it.

KapilYadav • (359th in this Competition) • 8 months ago • Options • Reply 0

Thanks for the detailed insights of your approach.

Just one more question and then I will stop bugging you :) -

By "With probability 0.2 patches were sampled close to some random sea lion", do you mean that a sea-lion in train images was selected randomly and then several patches were cropped within some specified distance from that sea-lion ? But then where does the 0.2 probability come into play ?

Konstantin Lopu... • (2nd in this Competition) • 8 months ago • Options • Reply 1

I'm happy to answer the questions :)

I sampled patches that formed the batch completely independently. With probability 0.8, I sampled a random image, a random location in that image, a random scale and random angle, and created a patch from it. With probability 0.2, I sampled a random lion (uniformly across all lions, so crowded images got more samples), and then sampled a random location within a pre-defined distance (~patch size) from that lion, and then again the angle and scale.

gbhalla • (169th in this Competition) • 8 months ago • Options • Reply 0

Thanks Konstantin and great work. Quick question, did you learn the categories separately? In other words your log loss was over 5 categories?

Konstantin Lopu... • (2nd in this Competition) • 8 months ago • Options • Reply 1

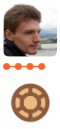
Thanks @gbhalla! I used a log loss with 6 classes, one for background and the other for different lions - so it's the same loss you normally use for classification problems, but applied per-pixel on the output mask.

Charles Jansen • 8 months ago • Options • Reply 0

Congratulation!! and thanks for sharing!

So you resized the test images by half. Did it not make all the sea lions smaller compare to training? (in pixels). Or did you resize the train images too?

How did you deal with half cut sea lions on the patches (like the tail in on patch and the other half with the head on another patch)?



Konstantin Lopu... • (2nd in this Competition) • 8 months ago • Options • Reply 1

Thanks!

> So you resized the test images by half. Did it not make all the sea lions smaller compare to training? (in pixels). Or did you resize the train images too?

I think that images in test are about 2x bigger than what we have in the training set. Probably they used a different camera, or didn't resize them, or flew lower, or something like this :)

Still the images in test had different scales, so I applied scale augmentation during training to make the model more robust. Also I was not sure if 2x was the precise difference, and didn't really want to fit it using the public LB.

> How did you deal with half cut sea lions on the patches (like the tail in on patch and the other half with the head on another patch)?

I averaged overlapping predictions at test time. At train time I didn't do anything special.

Artem.Sanakoev • (4th in this Competition) • 8 months ago • Options • Reply 1

@Charles Jansen, You can check our solution. https://www.kaggle.com/c/noaa-fisheries-steller-sea-lion-population-count/discussion/35442
To mitigate the effect of cutting a lion on half I placed gaussians on top of each lion and regressed the sum over all Gaussians in the current tile.

mrgloom • (104th in this Competition) • 8 months ago • Options • Reply 0

It's very high level overview, so maybe it's better to look at the code after, but I have one question: I haven't tried semantic segmentation losses yet, but I was able to train density map regression with U-net only for all classes collapsed to one class and only for 'positive' tiles, so I wonder is it so hard to train U-net or maybe it's due to class imbalance? do you balance your data?

P.S. thanks for your comments in Discusssion threads.

Artem.Sanakoev • (4th in this Competition) • 8 months ago • Options • Reply 1

I tried the density map regression as well and had classes collapsing too. I think this is just a limitation of the density map regression. It doesn't work with more than one class.