Which whale is it, anyway? Face recognition for right whales using deep learning

by Robert Bogucki (http://deepsense.io/author/robert-bogucki/) | Jan 16, 2016 | Blog (http://deepsense.io/category/blog/) | 4 comments (http://deepsense.io/deep-learning-right-whale-recognition-kaggle/#respond)

(http://deepsense.io/wp-content/uploads/2016/02/noaa_fisheries_small.png)
Right Whale Recognition (https://www.kaggle.com/c/noaa-right-whale-recognition) was a computer vision competition organized by the NOAA Fisheries on the Kaggle.com data science platform. Our machine learning team at deepsense.io has finished 1st! In this post we describe our solution.

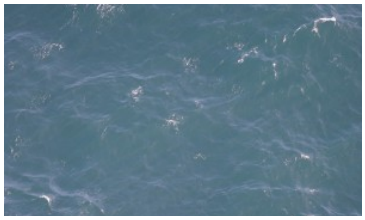
The challenge

The goal of the competition was to recognize individual right whales in photographs taken during aerial surveys. When visualizing the scenario, do not forget that these giants grow up to more than 18 metres, and weigh up to 91 tons. There were 447 different right whales in the data set (and this is likely the overall number of living specimens). Even though that number is (terrifyingly) small for a species, uniquely identifying a whale poses a significant challenge for a person. Automating at least some parts of the process would be immensely beneficial to right whales' chances of survival.

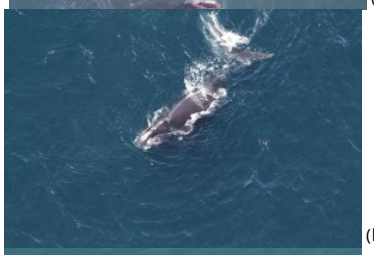
> "Recognizing a whale in real-time would also give researchers on the water access to potentially life-saving historical health and entanglement records as they struggle to free a whale that has been accidentally caught up in fishing gear,"
>
> — excerpt from competition's description page.

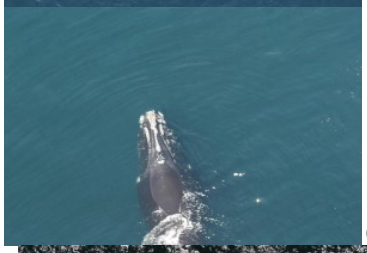
The photographs had been taken at different times of day, and with various equipment. They ranged from very clear and centered on the animal to taken from afar and badly focused.

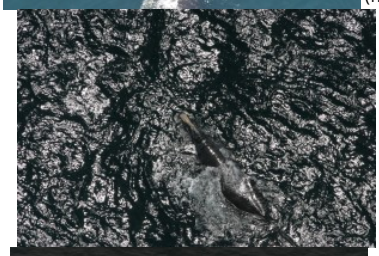

(http://deepsense.io/wp-content/uploads/2016/01/w_4615.jpg)

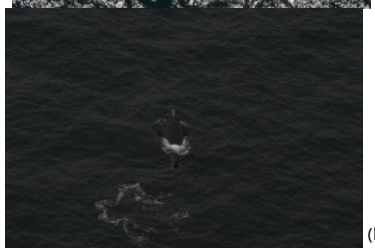



(http://deepsense.io/wp-content/uploads/2016/01/w_4661.jpg)


(http://deepsense.io/wp-content/uploads/2016/02/w_10578.jpg)


(http://deepsense.io/wp-content/uploads/2016/02/w_10579.jpg)


(http://deepsense.io/wp-content/uploads/2016/01/w_81.jpg)


(http://deepsense.io/wp-content/uploads/2016/01/w_4771.jpg)

A non-random sample from the dataset

The teams were expected to build a model that recognizes which one of the 447 whales is captured in the photograph. More technically, for each photo, we were asked to provide a probability distribution over all 447 whales. The solutions were judged by multiclass log loss, also known as cross-entropy loss.

It is worth noting that the dataset was not balanced. The number of pictures per whale varied a lot: there were some "celebrities" having around forty, a great deal with over a dozen, and around twenty with just a single photo! Another challenge, is that images representing different classes (i.e. different whales) were very similar to each other. This is somewhat different than the situation where we try to discriminate between, say, dogs, cats, wombats and planes. This posed some difficulties for the neural networks that we had been training – the unique characteristics that make up an individual whale, or that set this particular whale apart from others, occupy only a small portion of an image and are not very apparent. Helping out our classifiers to focus on the correct features, i.e. the whale's heads and their callosity patterns, turned out to be crucial.

The backbone of our solution

Convolutional neural networks (CNNs) (https://en.wikipedia.org/wiki/Convolutional_neural_network) have proven to do extraordinarily well in image recognition tasks, so it was natural for us to base our solution on them. In fact, to our knowledge, all top contestants have used them almost at every step. Even though some say that those techniques require huge amounts of data (and we only had 4,544 training images available with some of the whales appearing only once in the whole training set), we were still able to produce a well-performing model, proving that CNNs are a powerful tool even on limited data.

In fact our solution consisted of multiple steps:
– head localizer (using CNNs),
– head aligner (using CNN),
– training several CNNs on passport-like photos of whales (obtained from previous steps),
– averaging and tuning the predictions (not using CNNs).

One additional trick that has served us well is providing the networks with some additional targets, even though those additional targets were not necessarily used afterwards or required in any way. If the additional targets depend on the part of image that is of particular interest for us (i.e. head and callosity pattern in this case), this trick should force the network to focus on this area. Also, the networks have more stimuli to learn on, and thus have to develop more robust features (sensible for more than one task) which should limit overfitting.

Software and hardware

We used Python, NumPy and Theano to implement our solution. To create manual annotations (and do not lose sanity during these long moments of questionable joy) we employed Sloth (a universal labeling tool), as well as an ad-hoc Julia script.

To train our models we used two types of Nvidia GPUs: Tesla K80 and GRID K520.

Domain knowledge

Even though it seems significantly harder for humans to identify whales than other people, neural networks do not suffer from such problem for obvious reasons. Even then, a bit of Wikipedia research on right whales reveals that "the most distinguishing feature of a right whale is the rough patches of skin on its head which appear white due to parasitism by whale lice." Turns out that it not only distinguishes them from other species of whales, but also serves as an excellent way to differentiate between specimens. Our solution uses this fact to great lengths. Besides this somewhat trivial hint (provided by the organizers) about what to look at, we (sadly) posses no domain knowledge about right whales.

Preparing for the take-off

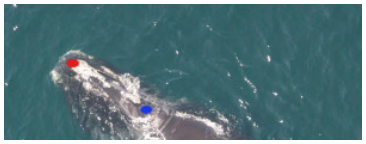
Before going further into the details, a disclaimer is needed. During a competition, one is (or should be) more tempted to test new approaches, than to fine tune and clean up existing ones. Hence, soon after an idea had proved to work well enough, we usually settled on it and left it "as is". As a side-effect, we expect that some parts of the solution may possess unnecessary artifacts or complexity that could (and ought to) be eliminated. Nevertheless, we have not done so during the competition, and decided not to do so here either.

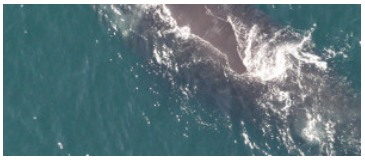
One does not need to see many images from the dataset in order to realize that whales do not pose very well (or at least were reluctant to do so in this particular case).


(http://deepsense.io/wp-content/uploads/2016/01/w_12.jpg)


(http://deepsense.io/wp-content/uploads/2016/01/w_7.jpg)
Not very cooperative whales

Therefore, before training final classifiers we spent some time (and energy) to account for this fact. The general idea of this approach can be thought of as obtaining a passport photo from a random picture where the subject is in some arbitrary position. This boiled down to training a head localizer and a head aligner. The first one takes a photo and produces a bounding box around the head, still the head may be arbitrarily rotated and not necessarily in the middle of the photo. The second one takes a photo of the head and aligns and rescales it, so that the blowhead and bonnet-tip are always in the same place, and the distance between them is constant. Both of these steps were done by training a neural network on manual annotations for the training data.





(http://deepsense.io/wp-content/uploads/2016/02/w_0_indygo2-e1452887273604.jpg)
Bonnet-tip (red) and blowhead (blue)

Localizing the whale

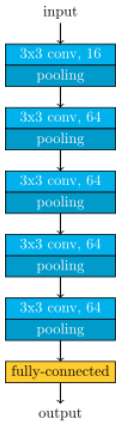
This was the first step in order to achieve good quality, passport-like photos. In order to obtain the training data, we have manually annotated all the whales in the training data with bounding boxes around their heads (a special thanks goes to our HR department for helping out!).

(http://deepsense.io/wp-content/uploads/2016/02/w_0_bbox.jpg)
Bounding box produced by the head localizer

These annotations amounted to providing four numbers for each image in the training set: coordinates of the bottom-left and the top-right points of the rectangle. We then proceeded to train a CNN which takes an original image (resized to 256×256) and outputs the two coordinates of the bounding box. Although this is clearly a regression task, instead of using L2 loss, we had more success with quantizing the output into bins and using Softmax together with cross-entropy loss. We have also tried several different approaches, including training a CNN to discriminate between head photos and non-head photos or even some unsupervised approaches. Nevertheless, their results were inferior.

In addition, the head localizing network also had to predict the coordinates of the blowhead and bonnet-tip (in the same, quantized manner), however it was less successful in this task, so we ignored this output.

We trained 5 different networks, all of them had almost the same architecture.



(http://deepsense.io/wp-content/uploads/2016/02/gscp_smaller.png)
Head localizer's architecture

Where they differed is the number of buckets used to quantize the coordinates. We have tried 20, 40, 60, 128, and also another (a little bit smaller) network using 20 buckets.

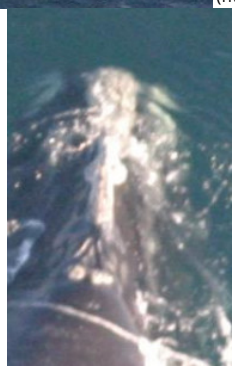
Before feeding the image into the network, we had used the following data augmentation (after resizing it to 256×256):
– rotation: up to 10° (note that if you allow bigger angles, it won't be enough to simply rotate the points – you'll have to implement some logic to recalculate the bounding box – we were too lazy to do this),
– rescaling: random ratio between 1/1.2 and 1.2,
– colour perturbation (as in Krizhevsky et al. 2012 (http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf)), with scale 0.01.

Although we have not used test-time augmentation here, we combined the outputs from all the 5 networks. Each time a crop was passed to the next step (head alignment), a random one from all 5 variants was chosen. As can be seen above, the crops provided by these networks were quite satisfactory. To be honest – we did not really "physically" crop the images (i.e. produce a bunch of smaller images). What we've done instead, and what turned out to be really handy, is producing a json file with the coordinates of bounding boxes. This may seem trivial, but doing so encouraged and allowed us to easily experiment with them.

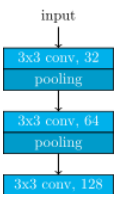
"Passport photos" of whales

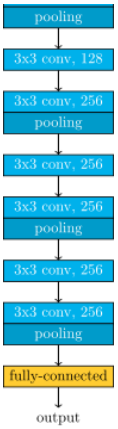
The final step of this "make the classifier's life easier" pipeline was to align the photos so that they all conform to the same standards. The idea was to train a CNN that estimates the coordinates of blowhead and bonnet-tip. Having them, one can easily come up with a transformation that maps the original image into one where these two points are always in the same position. Thanks to the annotations made by Anil Thomas (https://www.kaggle.com/c/noaa-right-whale-recognition/forums/t/17555/try-this), we had the coordinates for the training set. So, once again we proceeded to train CNN to predict quantized coordinates. Although we do not claim that it was impossible to pinpoint these points using the whole image (i.e. avoid the previous step), we now face a possibly easier task – we know the approximate location of the head.

In addition, the network had some additional tasks to solve! First of all it needed to predict which whale is on the image (i.e. solve the original task). Moreover, it needed to tell whether the callosity pattern on the whale's head is continuous or not (training on manual annotations once again, although there was much less work this time since looking at 2-3 images per whale was enough).


(http://deepsense.io/wp-content/uploads/2016/02/w_8734_non_continuous.jpg)

(http://deepsense.io/wp-content/uploads/2016/02/w_1000_continuous.jpg)
Broken (left) and continuous (right) callosity patterns

We used the following architecture.





(http://deepsense.io/wp-content/uploads/2016/02/new_gsc.png)
Head aligner's architecture

As an input to head aligner we took the crops from the head localizer. This time we dared to use a more bold augmentation:
– translation: random up to 4 pixels,
– rotation: up to 360°,

– rescaling: random ratio between 1.0 and 1.5,
– random flip,
– colour perturbation with scale 0.01.

Also, we used test-time augmentation – the results were averaged over 5 random augmentations (not much, but still!). This network achieved surprisingly good results. To be more (or less) accurate – during manual inspection, the blowhead and bonnet-tip positions were predicted almost perfectly for almost all images. Besides, as a by-product (or was it the other way?), we started to see some promising results for the original task of identifying the whales. The (quite pessimistic) validation loss was around 2.2.

Putting everything together

When chaining multiple machine learning algorithms in the pipeline, some caution is needed. If we train a head localizer and head aligner on the training data and use them to produce "passport photos" for both the training and testing set, we may end up with photos that vastly differ in quality (and possibly some other properties). This can turn out to be a serious difficulty when training a classifier on top of these crops – if at test time it faces inputs that don't bear any similarity to those it was accustomed to, it does not perform.

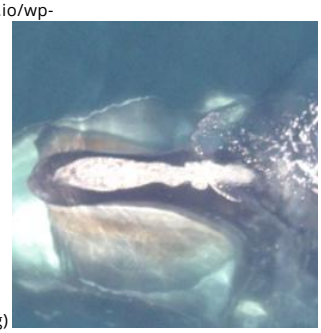
Although we were aware of this fact, when inspecting the crops we discovered that the quality did not differ much between training and testing data. Moreover, we've seen a huge improvement in the log loss. Following the mindset of "more ideas, less dwelling", we decided to settle on this impure approach and press on.


(http://deepsense.io/wp-content/uploads/2016/02/img_1_w_8604_class_idx_349.jpg)

(http://deepsense.io/wp-content/uploads/2016/02/img_2_w_1354_class_idx_322.jpg)


(http://deepsense.io/wp-content/uploads/2016/02/img_1_w_7356_class_idx_67.jpg)

(http://deepsense.io/wp-content/uploads/2016/02/img_1_w_3178_class_idx_332.jpg)

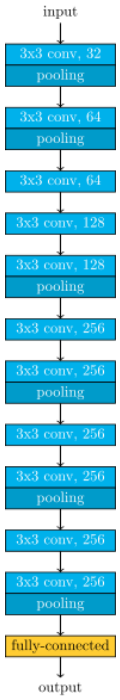
(http://deepsense.io/wp-content/uploads/2016/02/img_1_w_929_class_idx_51.jpg)

(http://deepsense.io/wp-content/uploads/2016/02/img_1_w_427_class_idx_372.jpg)

"Passport photos" of whales

Final classifier

Network architecture

Almost all of our models worked on 256×256 images and shared the same architecture. All convolutional layers had 3×3 filters and did not change the size of the image, all pooling layers were 3×3 with 2 stride (they halved the size). In addition all convolutional layers were followed by batch normalization and ReLU nonlinearity.



(http://deepsense.io/wp-content/uploads/2016/02/new_gsc3.png)

Main net's architecture

In the final blend, we have also used a few networks with an analogous architecture, but working on 512×512 images, and one network with a "double" version of this architecture (with an independent copy of the stacked convolutional layers, merged just before the fully-connected layer).

Once again, we've violated the networks' comfort zone by adding an additional target – determining the continuity of the callosity pattern (same as in the head aligner case). We've also tried adding more targets coming from other manual annotations, one such target was "how much was the face symmetric". Unfortunately, it did not improve the results any further.

Data augmentation

At this point data augmentation is a little bit trickier than usual. The reason behind this, is that you have to balance between enriching the dataset enough, and not ruining the alignment and normalization obtained from the previous step. We ended up using quite mild augmentation. Although the exact parameters were not constant among all models, the most common were:
– translation: random up to 4 pixels,
– rotation: up to 8°,
– rescaling: random ratio between 1.0 and 1.3,
– random flip,
– colour perturbation with scale 0.01.

We also used test-time augmentation, and averaged over 20+ random augmentations.

Initialization

We used a very simple initialization rule, a zero-centred normal distribution with std 0.01 for convolutional layers and std 0.001 for fully connected layers. This has worked well enough.

Regularization

For all models, we have used only L2 regularization. However, separate hyperparameters were used for convolutional and fully connected layers. For convolutional layers, we used a smaller regularization, around 0.0005, while for fully connected ones we preferred higher values, usually around 0.01 – 0.05.

One should also recall that we were adding supplementary targets to the networks. This imposes additional constraints on weights, enforces more focus on the head of the whale (instead of, say, some random patterns on the water), i.e. it works against overfitting.

Training

We trained almost all of our models with stochastic gradient descent (SGD) combined with 0.9 momentum. Usually we settled after around 500-1000 epochs (the exact moment didn't really matter much, because the lack of overfitting). During the training process we used quite slow exponential decay on the learning rate (0.9955) and also manually adjusted the learning rate from time to time. After a rough kick – increasing the learning rate (yet still leaving the decay) the network's error (on both: training and validation) went up a lot. Yet it tended to settle on a lower error after a few epochs to get back on a good track. Our best model also uses another idea – it was first trained with Nesterov momentum for over a hundred epochs, and then we switched to using Adam (adaptive moment estimation). We couldn't achieve similar loss when using Adam all the way from the start. The initial learning rate may not have mattered much, but we used values around 0.0005.

Validation

We used a random 10% of the training data for validation. Thanks to our favourite seed 7300, we kept it the same for all models. Although, we were aware that this method caused some whales to be absent in the training set, it has worked good enough. The validation loss was quite pessimistic, and correlated well with the leaderboard.

Giving up 10% of a relatively small dataset is not something one would decide to do without any hesitation. Therefore, after we had decided that a model is good enough to use it, we proceeded to reuse the validation set for training (it was straightforward because we didn't have any problems with overfitting). This was done either by a time-consuming process of rerunning everything from scratch on the whole training set (rarely), or (more often) adding the validation set to the training one and running 50-100 additional epochs. This was also meant to curate the issue of single-photo-whales appearing only in our validation set.

Combining the predictions

We have ended up with a number of models scoring in a range of 0.97 to 1.3 according to our validation (actual test scores were actually better). Thanks to keeping a consistent validation set we were able to test some blending techniques as well as basic transformations. We have concluded that using more complex ensembling methods would not be plausible due to the fact that our validation set did not include all the distinct whales, and was, in fact, quite small. Having no time left for some fancy cross-validation like ensembling, we settled on something ad-hoc and simple.

Even though simple weighted average of the predictions did not provide a better score than the best model alone (unless we increased the best model's weight significantly), it performed better when combined with a simple transformation of raising the predictions to a moderate power (in the final solution we used 1.45). This translated into roughly ~0.1 improvement in the log loss. We have not scrutinized this enough, yet one may hypothesize that this trick worked because our fully connected layers were strongly regularized, and were reluctant to produce more extreme probabilities.

We also used some "safety" transformations, namely increasing all the predictions by a certain epsilon as well as slightly skewing them to the whale distribution found in the training set. They had not impacted our score much.

Loading images

At the latest, and middle stages of our solution the images we loaded from the disk were original ones. They are quite big. To efficiently use the gpu we had to load them in parallel. We thought the main cost was loading the images from the disk. It turned out not to be true. Conversely, it was decoding JPEG file to a numpy array. We did a quick and dirty benchmark, with 111 random original images from the dataset, 85Mb total. Reading them, when they are not cached in RAM takes ~420 ms. Whereas reading, and decoding to numpy arrays takes ~10 seconds. That's a huge difference. The possible solutions to mitigate this, might be using other image formats, that offer better encoding speeds (at the cost of image sizes), decoding at GPU, etc.

What might have made it tick

Although, it's hard to pinpoint a single trick or technique that overshadows everything else (and to do so, one needs a more careful analysis), we have hypothesized about the key tricks.

Producing good quality crops with well aligned heads

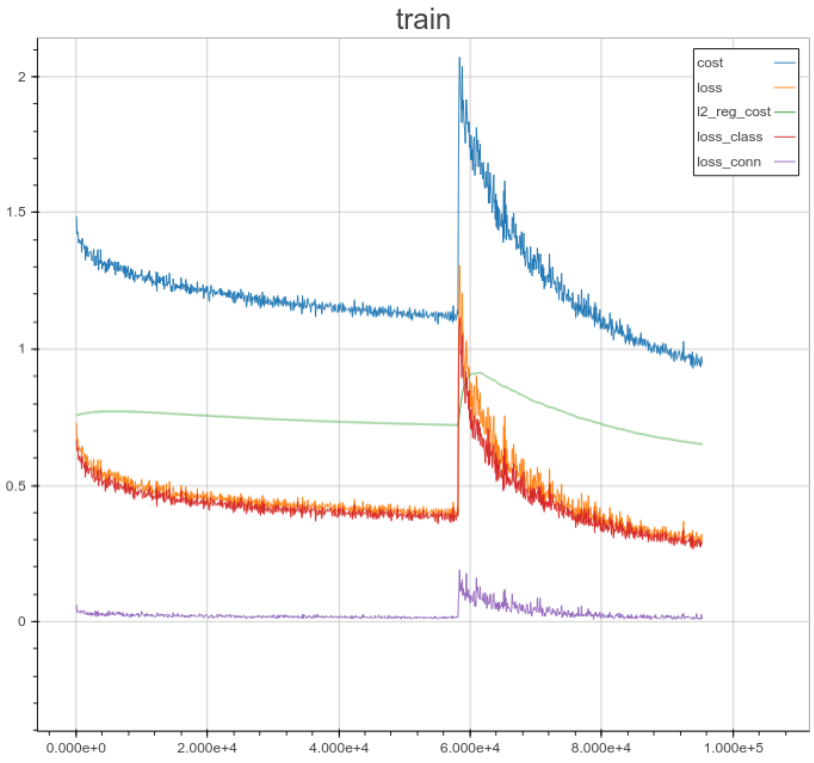
We have achieved this by doing this in two stages, and the results were vastly superior than what we observed with just a single stage.

Manual labour and providing the networks with additional targets

This was based on adding additional annotations to the original images. It required manual inspection of thousands of whales images, but we ended up using them in all stages (head localizing, head aligning, and classification).

Kicking the learning rate

Although it might have been the combination of a little bit careless initialization, poor learning rate decay schedule, and not fancy enough SGD, it turned out, that kicking (increasing) the learning rate did a great job.



(http://deepsense.io/wp-content/uploads/2016/02/Screenshot-from-2016-01-15-153358.png)
Loss functions after kicking the learning rate

Calibrating the probabilities

Almost all our models, and blends of models benefitted from raising the predictions to a moderate power in the range [1.1 – 1.6]. This trick, discovered at the end of the competition, improved the score by ~0.1.

Reusing the validation set

Merging validation and training sets and running for additional 50-100 epochs gave us steady improvements on the score.

Conclusion

Overall, this was an amazing problem to tackle and an amazing competition to take part in. We have learned a lot during the process, and are actually quite amazed with the super-powers of deep learning!

A special thanks to Christin Khan and NOAA for presenting the data science community with this exceptional challenge. We hope that this will inspire others. Of course, all this would be much harder without Kaggle.com, which does a great job with their platform. We would also like to thank Nvidia for giving us access to Nvidia K80 GPUs – they've done a great job.

Robert Bogucki
Marek Cygan
Maciej Klimek
Jan Kanty Milczek

Marcin Mucha

Share this on your favorite social media platform:

(/#facebook) (/#twitter) (/#linkedin) (/#google_plus)

4 Comments

Chris Farrell (http://www.fazshot.com.au) on January 19, 2016 at 5:04 am

Chi it's great to see the work on the Northern Right Whales, I have master aerial photography to help ID the Southern Right Whales on the Victorian coast of Australia. I send my aerials to Mandy Watson for ID. Please feel free to check out my images on http://www.fazshot.com.au (http://www.fazshot.com.au) under Southern Right Whale. Note I also taken aerial of the entire coast all funded by myself. Note all my photos are donated to the world wide fund for nature Audtralia.

Regards Chris

Xu Dong on January 18, 2016 at 12:34 pm

Thank you for sharing !

Christian S. Perone (http://blog.christianperone.com) on January 17, 2016 at 7:13 pm

Congatulations, hard work and well deserved place.

Adam on January 17, 2016 at 3:27 am

Well done!

Submit a Comment

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

SUBMIT COMMENT

Recent Posts

Which whale is it, anyway? Face recognition for right whales using deep learning (http://deepsense.io/deep-learning-right-whale-recognition-kaggle/)

deepsense.io's Data Scientists Help to Save Endangered Right Whales with Deep Learning (http://deepsense.io/deepsense-ios-data-scientists-help-to-save-endangered-right-whales-with-deep-learning/)

Exploration of data from iPhone motion coprocessor (http://deepsense.io/exploration-of-data-from-iphone-motion-coprocessor/)

Top 5 deepsense.io BIG Data Predictions for 2016 (http://deepsense.io/top-5-deepsense-io-big-data-predictions-for-2016/)

deepsense.io Announces Marathon24 Hackathon Winners (http://deepsense.io/deepsense-io-announces-marathon24-hackathon-winners/)

Archives

January 2016 (http://deepsense.io/2016/01/)
December 2015 (http://deepsense.io/2015/12/)
November 2015 (http://deepsense.io/2015/11/)
October 2015 (http://deepsense.io/2015/10/)
September 2015 (http://deepsense.io/2015/09/)
August 2015 (http://deepsense.io/2015/08/)
July 2015 (http://deepsense.io/2015/07/)
June 2015 (http://deepsense.io/2015/06/)
May 2015 (http://deepsense.io/2015/05/)
April 2015 (http://deepsense.io/2015/04/)
March 2015 (http://deepsense.io/2015/03/)

February 2015
(http://deepsense.io/2015/02/)
January 2015
(http://deepsense.io/2015/01/)

SEARCH

NEWS & PRESS RELEASES

- deepsense.io's Data Scientists Help to Save Endangered Right Whales with Deep Learning (http://deepsense.io/deepsense-ios-data-scientists-help-to-save-endangered-right-whales-with-deep-learning/) *January 14, 2016*
- deepsense.io Announces Marathon24 Hackathon Winners (http://deepsense.io/deepsense-io-announces-marathon24-hackathon-winners/) *December 21, 2015*
- CodiLime, Inc. Commits to Spend $4 million as Reinvestment in deepsense.io (http://deepsense.io/codilime-inc-commits-to-spend-4-million-as-reinvestment-in-deepsense-io/) *December 16, 2015*

more... (/about-us/press-center/)

OUR BLOG

- Which whale is it, anyway? Face recognition for right whales using deep learning (http://deepsense.io/deep-learning-right-whale-recognition-kaggle/) *January 16, 2016*
- Exploration of data from iPhone motion coprocessor (http://deepsense.io/exploration-of-data-from-iphone-motion-coprocessor/) *January 13, 2016*
- Top 5 deepsense.io BIG Data Predictions for 2016 (http://deepsense.io/top-5-deepsense-io-big-data-predictions-for-2016/) *December 30, 2015*
- How to create a new geom for ggplot2 (http://deepsense.io/how-to-create-a-new-geom-for-ggplot2/) *December 17, 2015*

more... (/our-blog/)

WHERE TO MEET US

Spark Summit East
Visit our booth!
(https://spark-summit.org/east-2016/)
Feb 16-18, 2016, New York, USA

Strata+Hadoop World
Meet us at our booth!
(http://conferences.oreilly.com/hadoop-big-data-ca)
March 29 - 31, 2016, San Jose, USA

Hadoop Summit
Come and visit our booth!
(http://2015.hadoopsummit.org/brussels/)
April 13-14, 2016, Dublin, Ireland

WORKSHOPS & WEBINARS

Spark & Machine Learning workshops
(http://workshops.deepsense.io/workshops/new-york-usa-feb-18th-19th-2016/)
Feb 18-19, 2016, New York, USA

Python in Data Science – hands-on workshops
(http://workshops.deepsense.io/workshops/python-in-data-science-hands-on-workshops/)
March 22, 2016, London, UK

more details soon at workshops.deepsense.io
(http://workshops.deepsense.io/)

PRODUCTS

Seahorse overview (/products/spark-applications/)

Seahorse CE (/products/seahorse-community-edition/)

Seahorse Enterprise (/products/seahorse-enterprise/)

Functionality (/products/seahorse-functionality/)

SERVICES

Spark Services (/services/spark-services/)

Machine Learning (/services/machine-learning/)

WORKSHOPS

Sign up (http://workshops.deepsense.io/)

Testimonials (http://workshops.deepsense.io/#testimonials)

Video (http://workshops.deepsense.io/)

ABOUT US

Company (/about-us/company/)

Executive Team (/about-us/senior-executive-team/)

Advisory Board (/about-us/scientific-advisory-board/)

Press Center (/about-us/press-center/)

Careers (/about-us/join-us/)

CONTACT

+1 415 900 9741 (tel:+14159009741)

contact@deepsense.io (mailto:contact@deepsense.io)

(https://www.facebook.com/deepsense.io) (https://twitter.com/deepsense_io) (https://www.linkedin.com/company/deepsense-io) (/our-blog/)