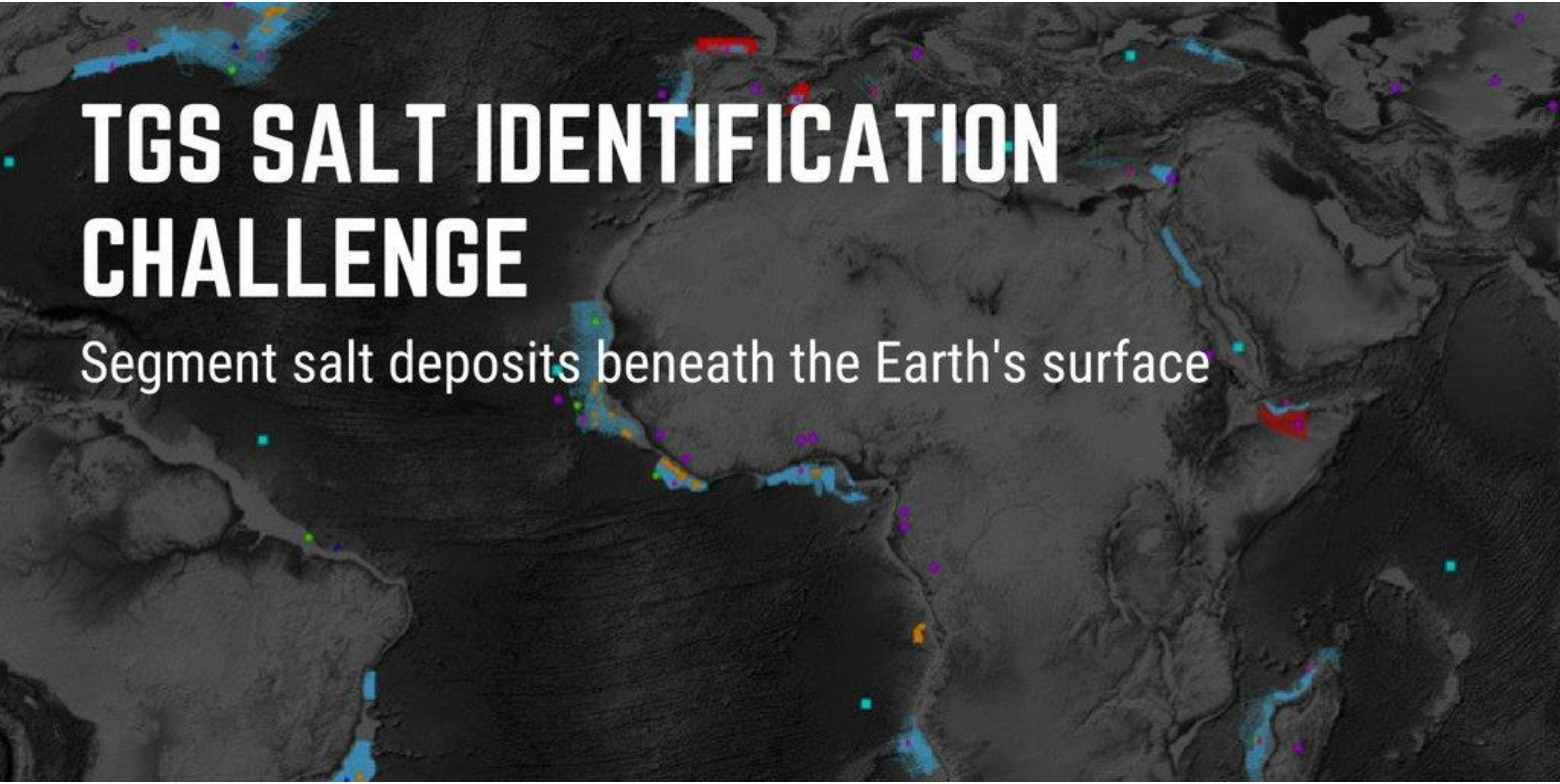
Kaggle TGS Salt Identification Challenge

Sosin Ivan
Kaspersky Lab



TGS SALT IDENTIFICATION CHALLENGE

Segment salt deposits beneath the Earth's surface

Salts and Spices: 11 / 3234

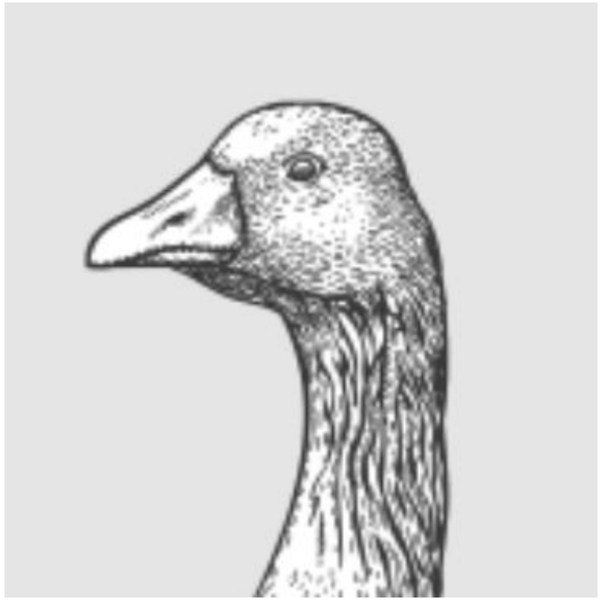
Team members:

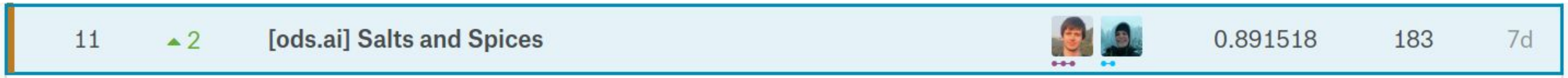
- Ivan Sosin

  Kaspersky Lab, ML Engineer

- Alexey Rozhkov

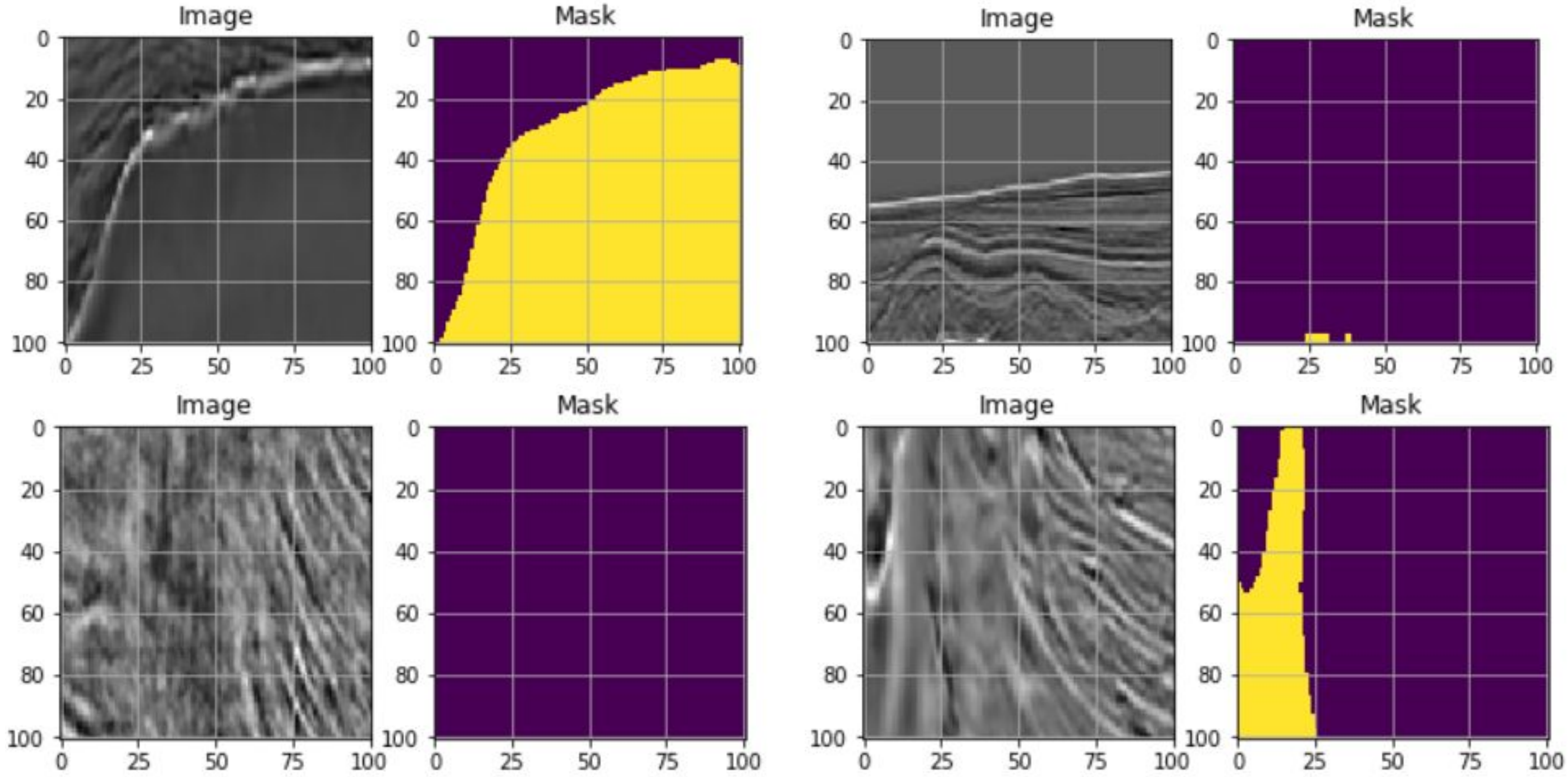
  Mati, Computer Vision Engineer





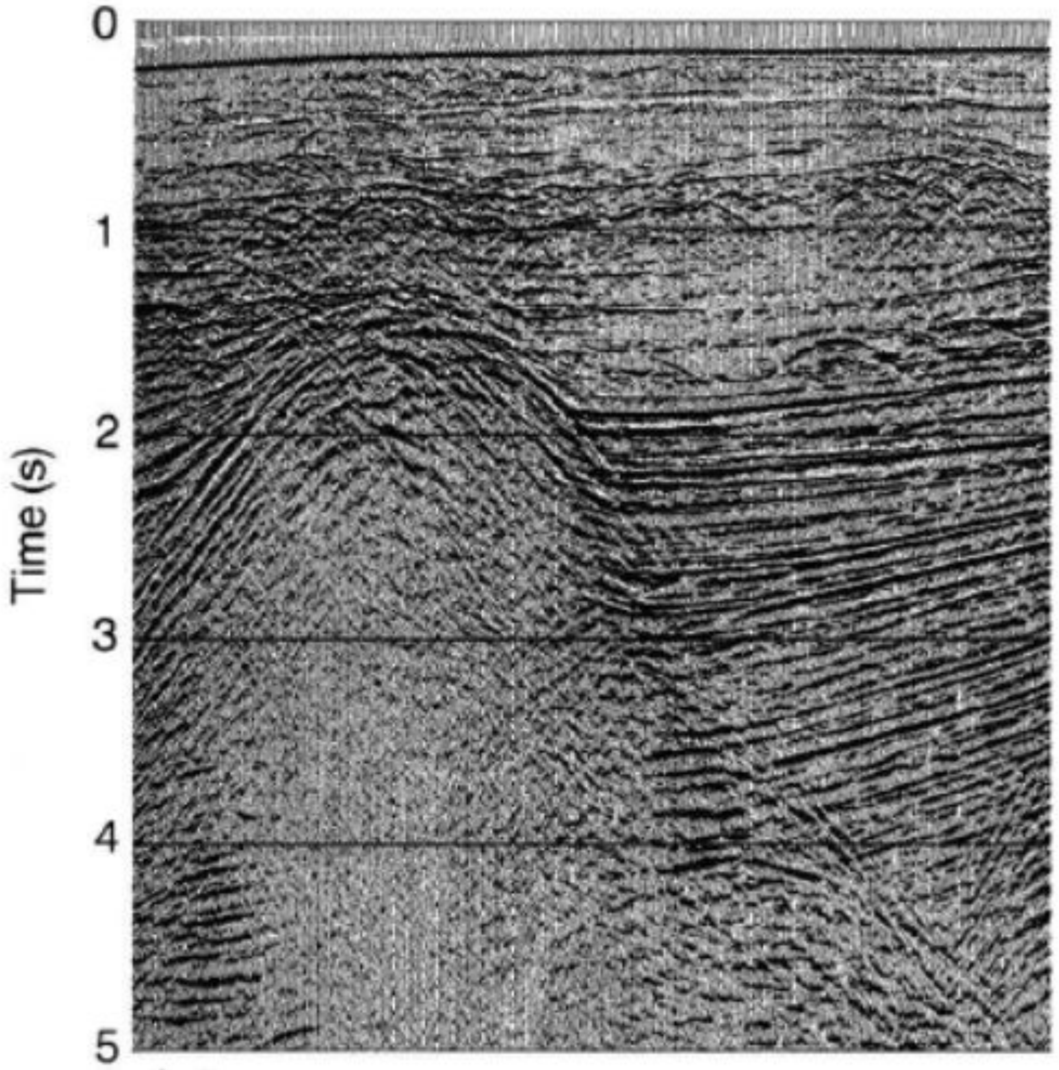
11	▲2	[ods.ai] Salts and Spices	0.891518	183	7d

Task - binary segmentation



Data orientation

This kind of axes orientation plays a great role in data augmentation



Metric

$$IoU(A, B) = \frac{A \cap B}{A \cup B}.$$

thresholds = (0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95)

$$\frac{1}{|thresholds|} \sum_t \frac{TP(t)}{TP(t) + FP(t) + FN(t)}.$$

Dataset

Train: 4000 images with masks

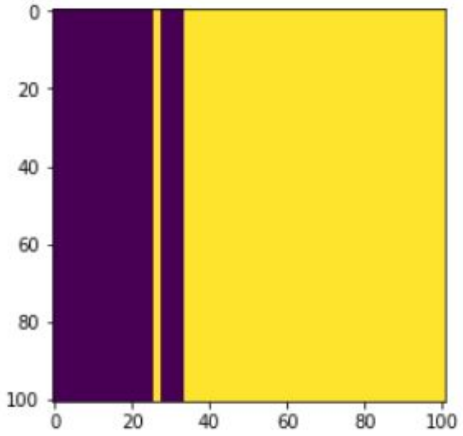
Test: 18000 images

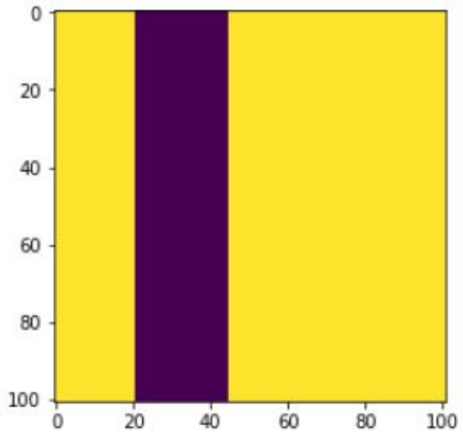
Train and Test image depths

Image resolution: 101 * 101

Facts:

- About of 39% of train masks were empty
- There was no full salt masks
- Some masks were just straight lines



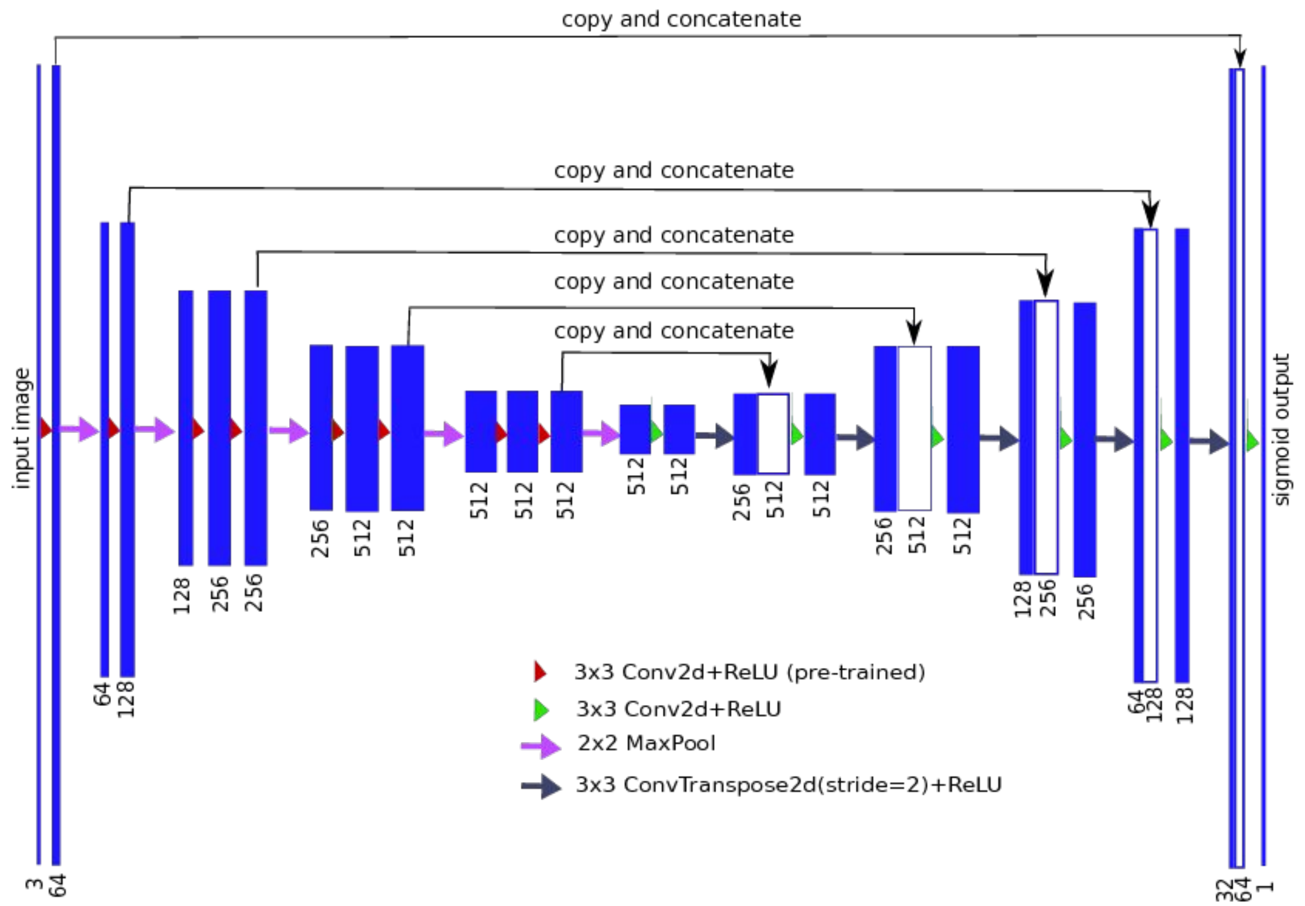


Augmentations

1) Random horizontal flip
2) Random color invert
3) Random cutout
4) Random gamma-correction
5) Random fixed-size crop
6) Random crop and scale
7) Random rotation up to 10 degrees

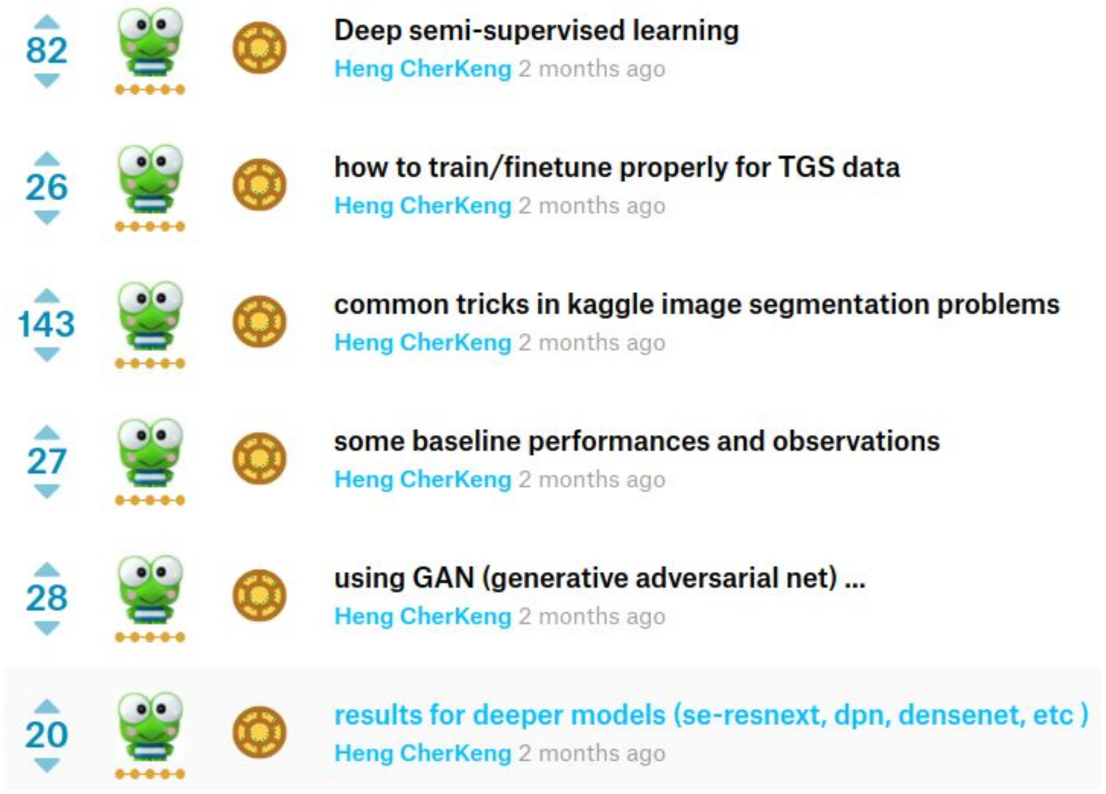
Model

Baseline: TernausNet (variation of UNet architecture)



https://github.com/ternaus/TernausNet

Tribute to Heng CherKeng



82 Deep semi-supervised learning
Heng CherKeng 2 months ago

26 how to train/finetune properly for TGS data
Heng CherKeng 2 months ago

143 common tricks in kaggle image segmentation problems
Heng CherKeng 2 months ago

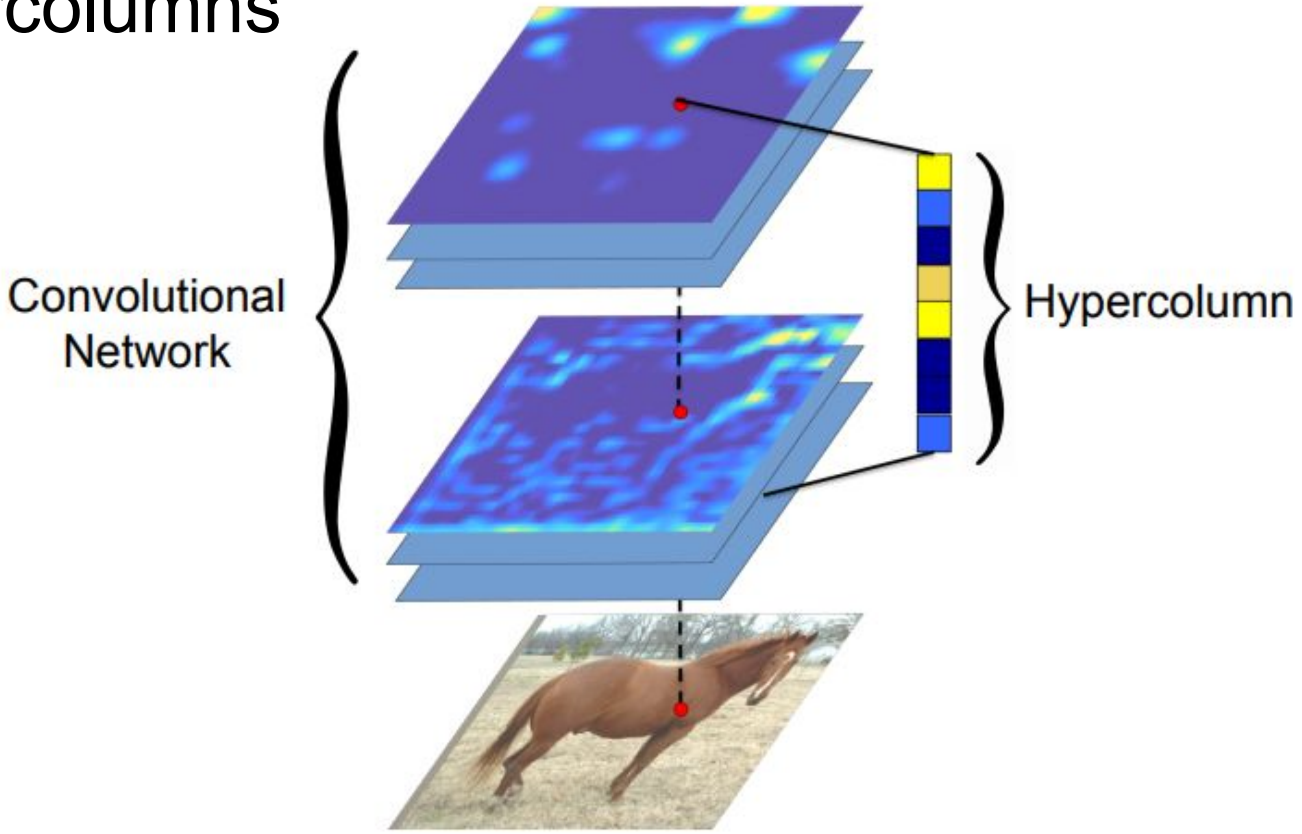
27 some baseline performances and observations
Heng CherKeng 2 months ago

28 using GAN (generative adversarial net) ...
Heng CherKeng 2 months ago

20 results for deeper models (se-resnext, dpn, densenet, etc)
Heng CherKeng 2 months ago

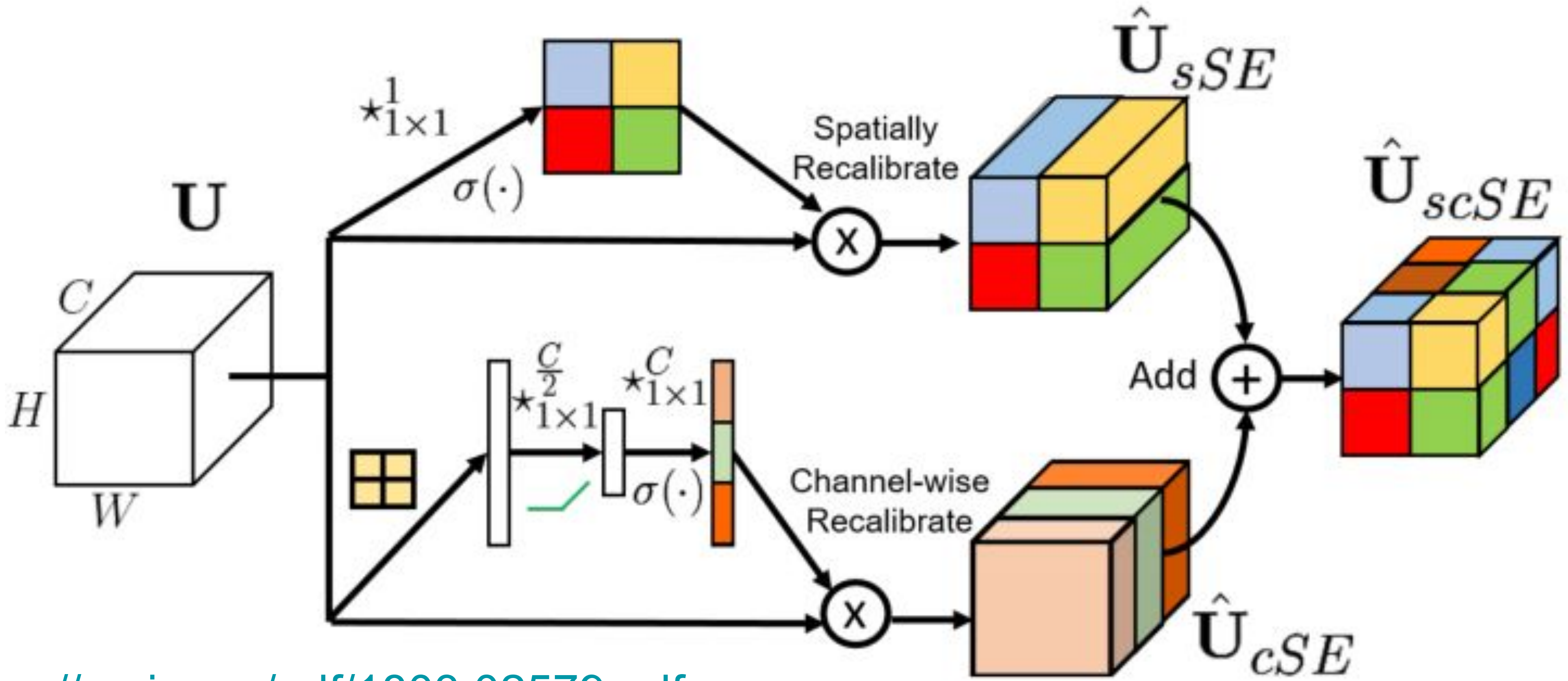
https://www.kaggle.com/hengck23

Hypercolumns



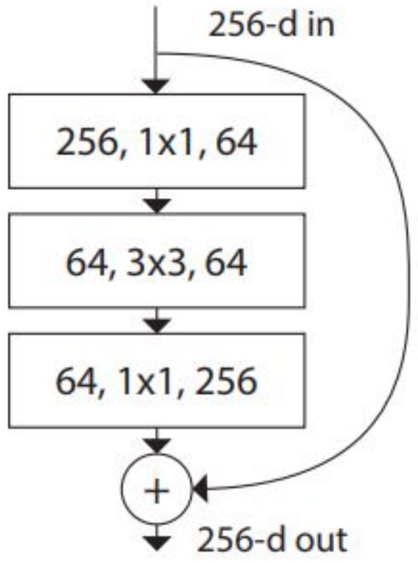
https://arxiv.org/abs/1411.5752

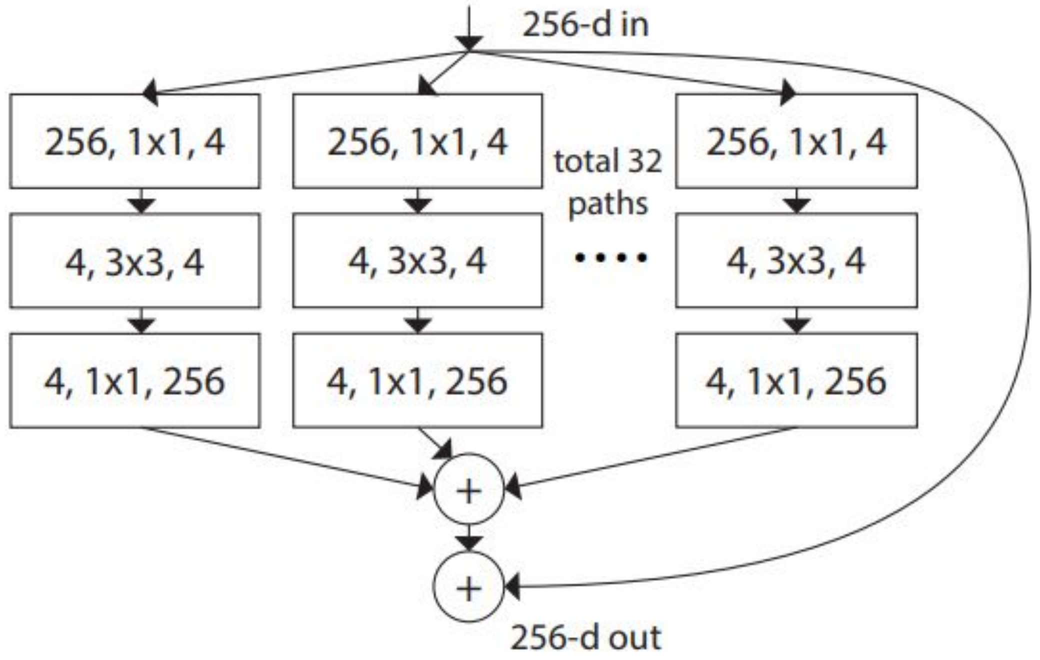
Concurrent Spatial and Channel Squeeze and Channel Excitation - scSE



https://arxiv.org/pdf/1803.02579.pdf

Resnext backbone architecture



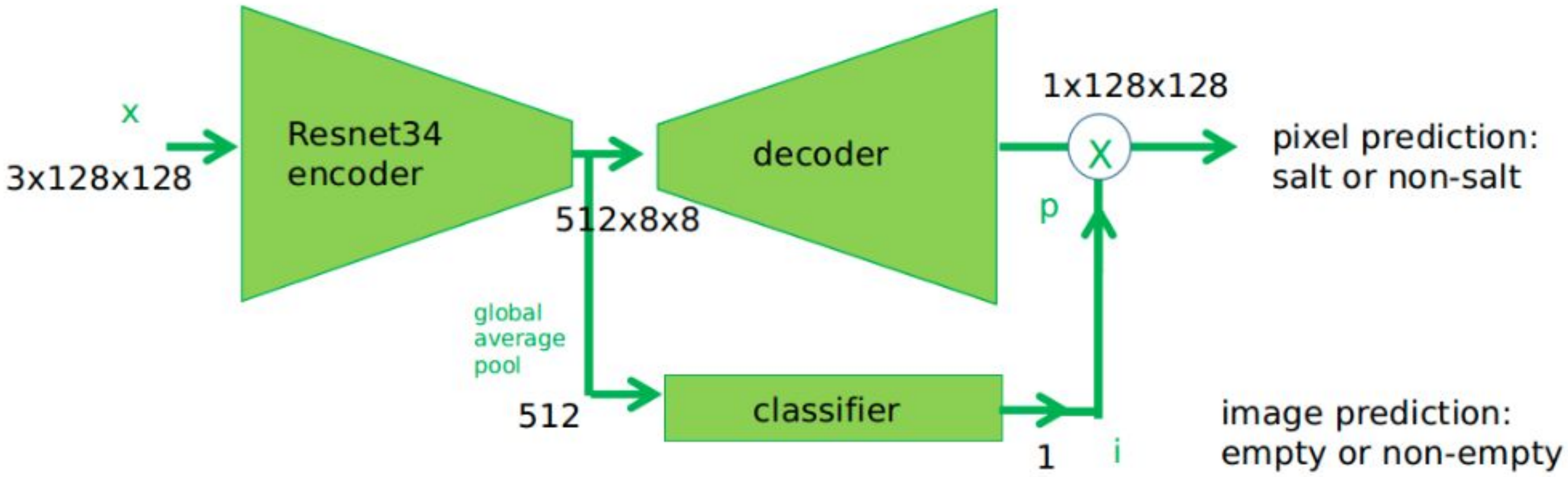


Resnet block

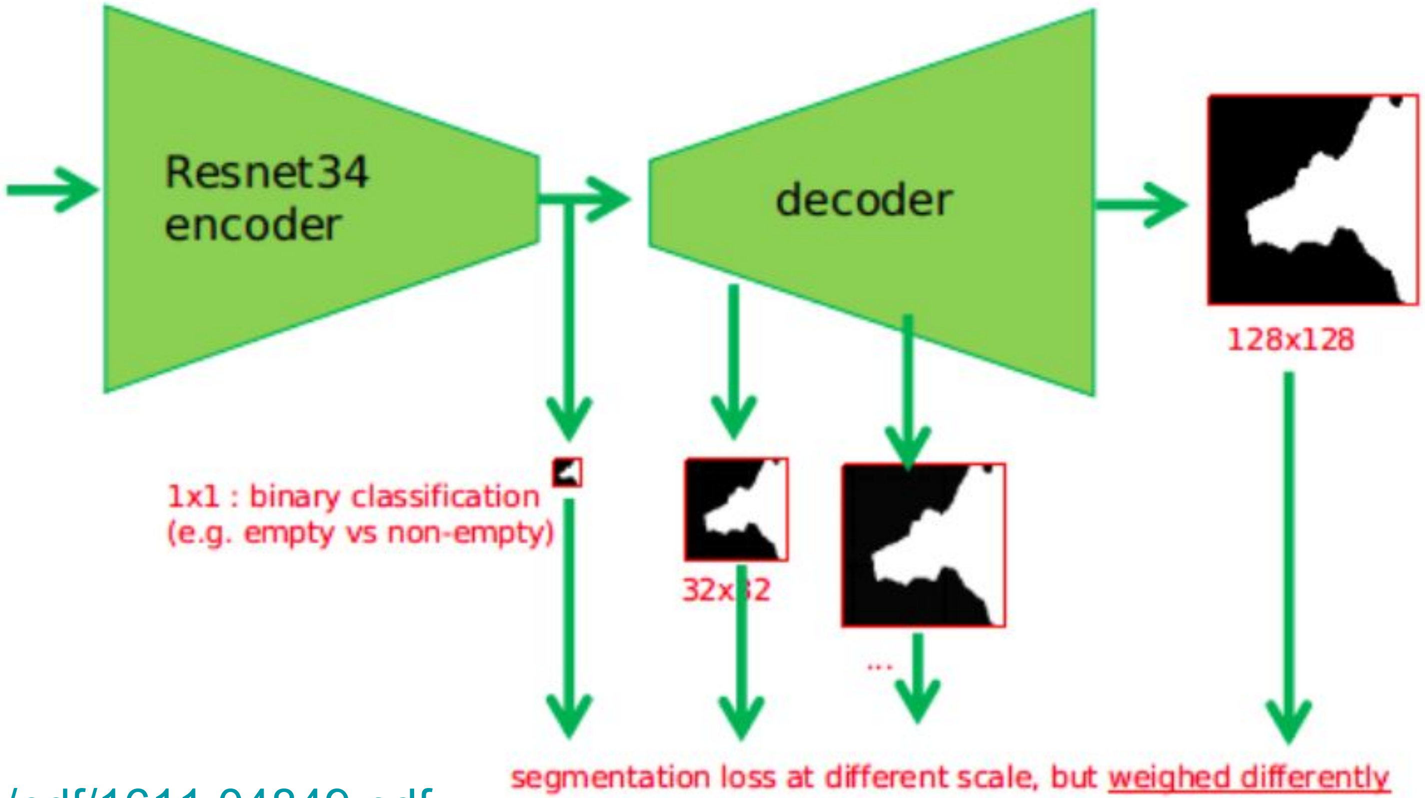
Resnext block with cardinality of 32

https://arxiv.org/pdf/1611.05431.pdf

Empty / non-empty mask classification

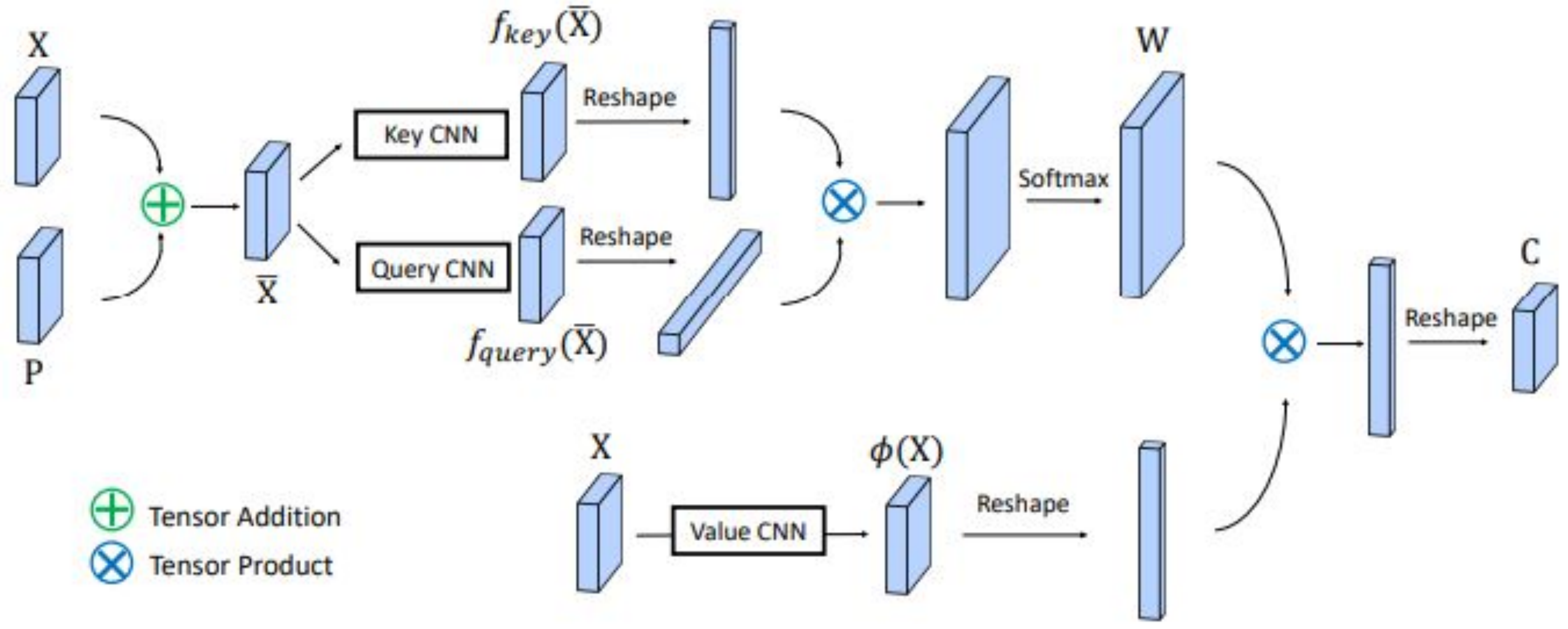


Deeply supervised network


Resnet34 encoder
decoder
128x128
1x1 : binary classification (e.g. empty vs non-empty)
32x32
...
segmentation loss at different scale, but weighed differently

https://arxiv.org/pdf/1611.04849.pdf

Object Context Network: Self-attention module (SA)

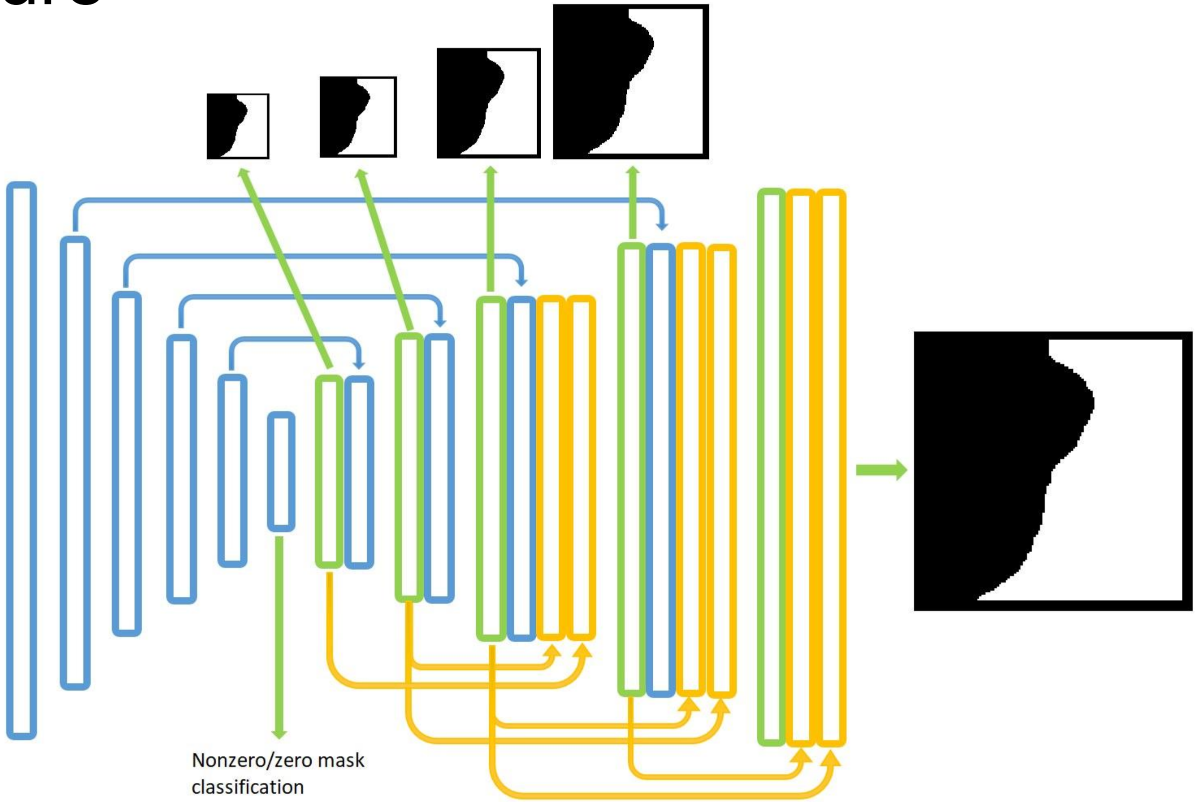


https://arxiv.org/pdf/1809.00916.pdf

My model architecture

Almost all ideas blended together

Encoder: pretrained Se-Resnext-50_32x4d



Decoder features

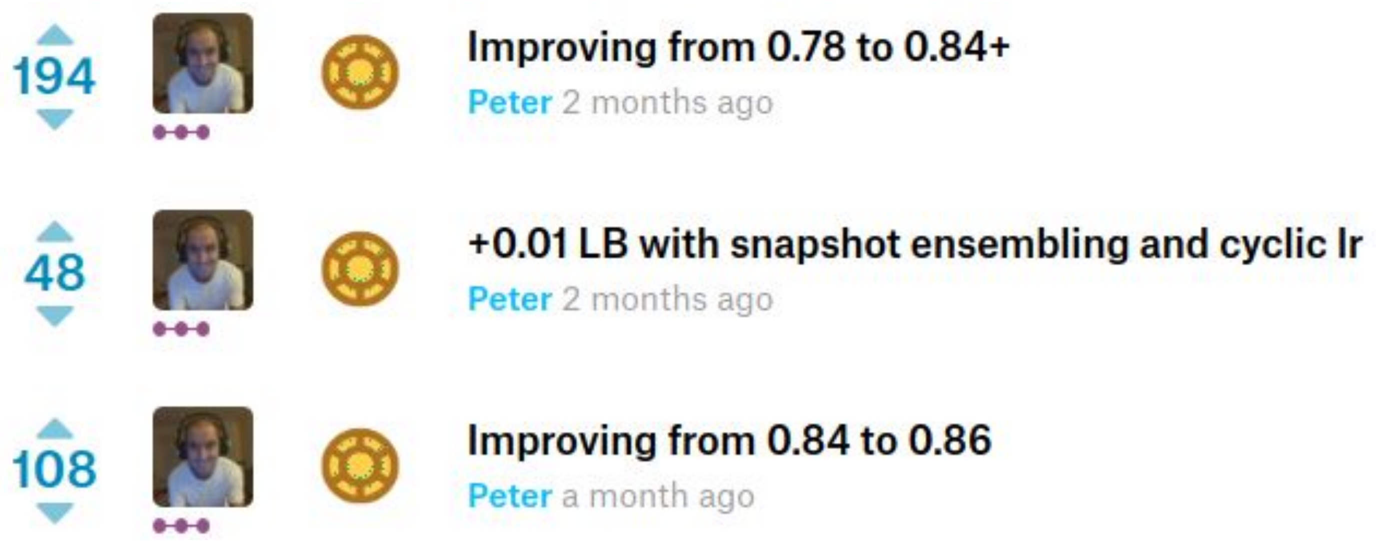
Ivan:

- Dilated convolutions with dilation from 1 to 5
- Hypercolumns
- OC Module before last Convolution layer
- Deep supervision (zero/nonzero mask, nonzero mask segmentation)

Alexey:

- Spatial and Channel Squeeze Excitation
- Hypercolumns
- Deep supervision (zero/nonzero mask)

Tribute to Peter



194

Improving from 0.78 to 0.84+

Peter 2 months ago

48

+0.01 LB with snapshot ensembling and cyclic lr

Peter 2 months ago

108

Improving from 0.84 to 0.86

Peter a month ago

https://www.kaggle.com/pestipeti

Losses

1) BCE
2) Focal Loss (https://arxiv.org/pdf/1708.02002.pdf)
3) Dice Loss
4) Lovasz loss - a tractable surrogate for the optimization of the ´ intersection-over-union measure in neural networks(https://arxiv.org/pdf/1705.08790.pdf)

Training

General:

- SGD: momentum 0.9, weight decay 0.0001
- Batch size: 16
- 5 Fold without stratification

Ivan

- Pretrain for 32 epochs with lr = 0.01 (0.9 * BCE + 0.1 * lovasz)
- SGDR with cosine annealing (0.1 * BCE + 0.9 * lovasz):
    - 4 cycles, lr = 0.01, min_lr = 0.0001, cycle_len = 64

Alexey

- SGDR with cosine annealing (1.0 * BCE + 0.5 * lovasz):
    - 10 cycles, lr = 0.05, min_lr = 0.0, cycle_len = 50

Equipment

Ivan

- Setup: 2x 1080Ti (~11 Gb memory consumption)
- Training time: 8 - 12 hours (per single fold) depending on model modification

Alexey

- Setup: 3x 1080 (~6.6 Gb memory consumption)
- Training time: 16 hours (per single fold)

Ensembling

Ivan:

- LB 0.859 single model, public LB 0.865 5 fold

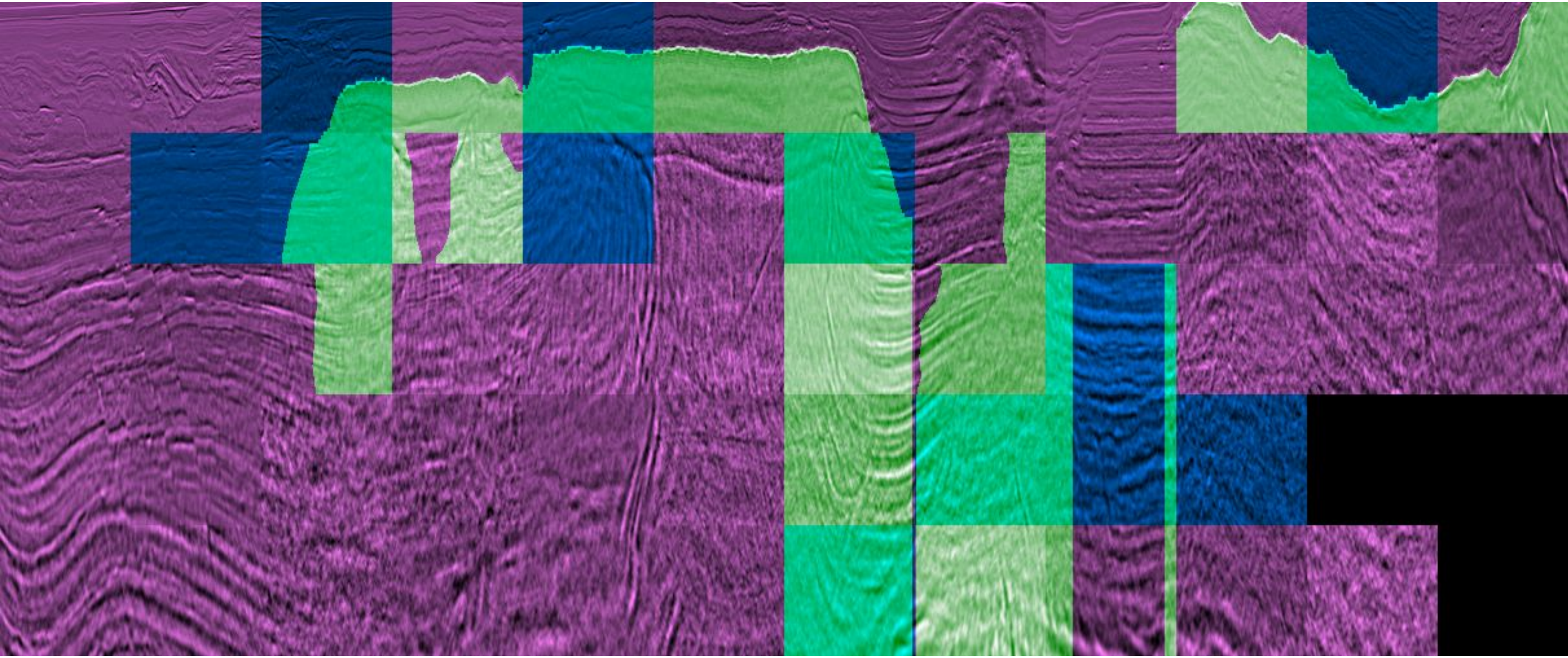
Alexey:

- LB 0.865 single model, public LB 0.871 5 fold

Ensemble:

- Without post processing: public LB 0.875
- With post processing: public LB 0.881

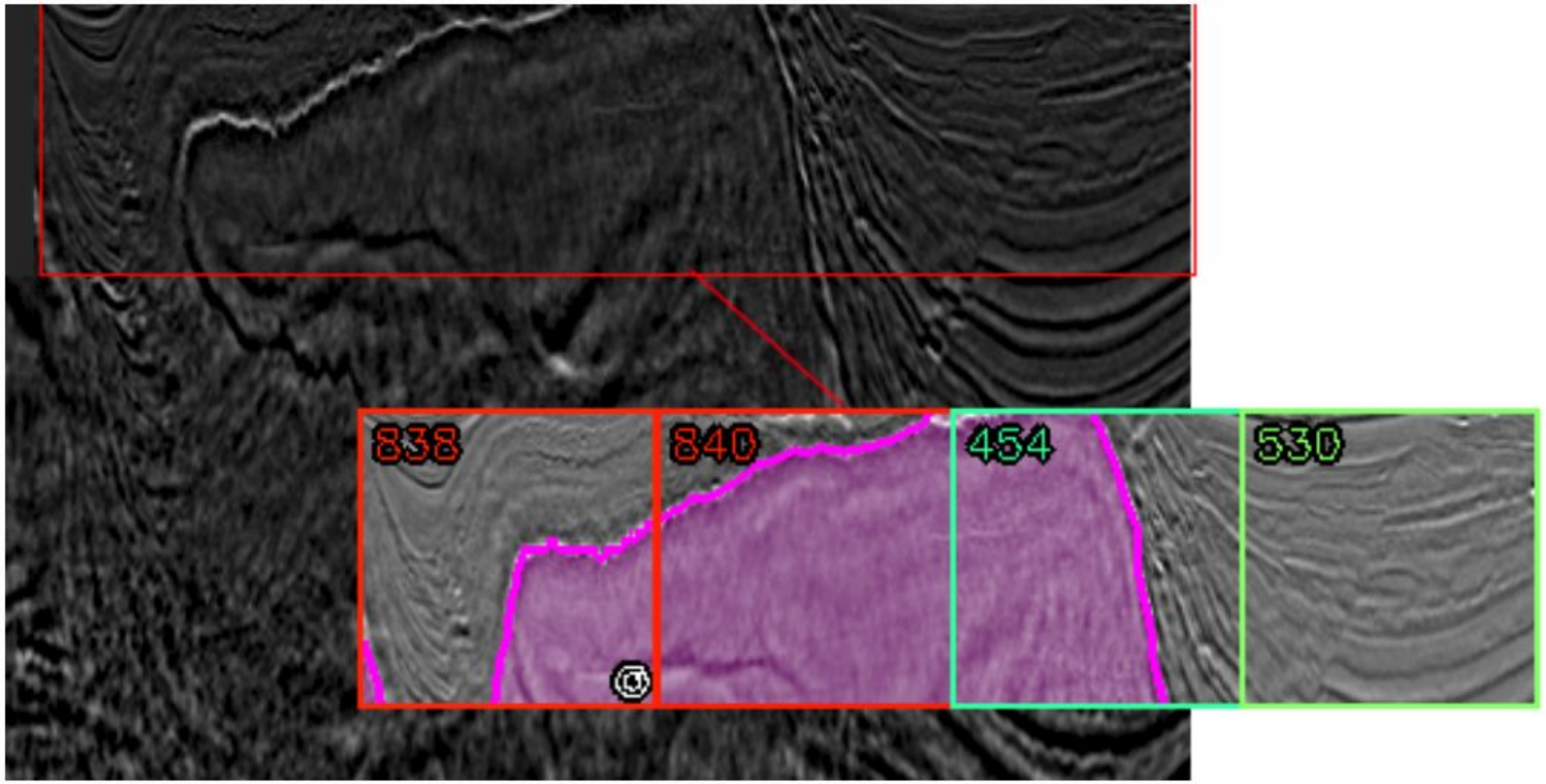
Mosaics: ~5000 images out of 18000



Post processing

- For almost-vertical ground truth tiles - extrapolate last row downwards till the end of mosaic
- For tiles below almost-vertical predicted tiles - extract downward
- For “caps” in ground truth tiles - zero all tiles below
- For tiles below predicted “caps” - do the same as for ground truth case
- For tiles vertically “sandwiched” between predicted or ground truth masks - replace mask with last row of tile above

Depth feature - 0

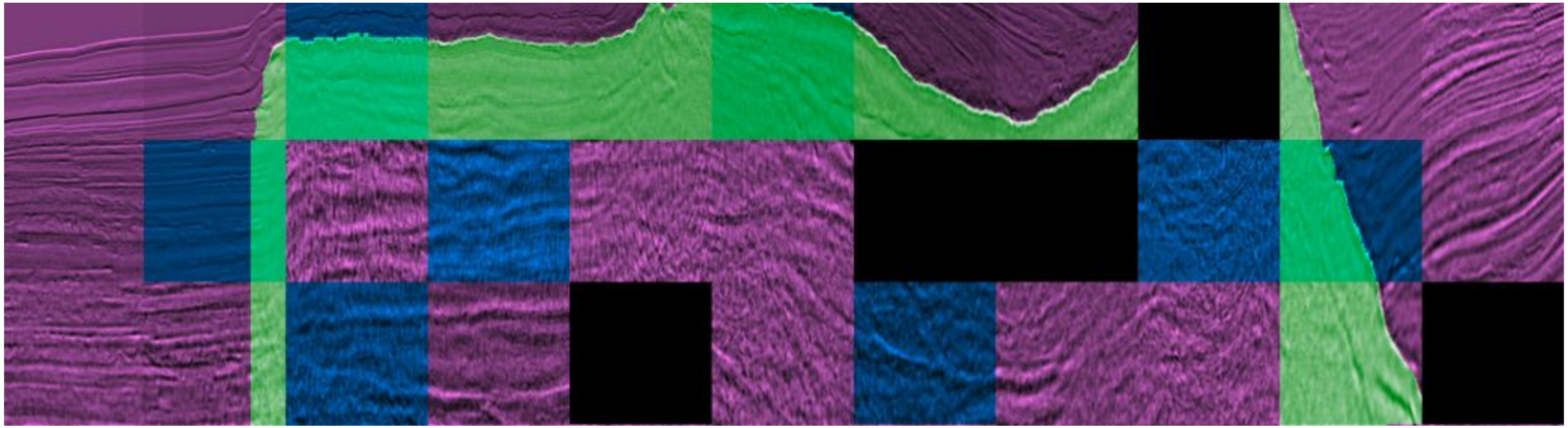


Depth feature - 1

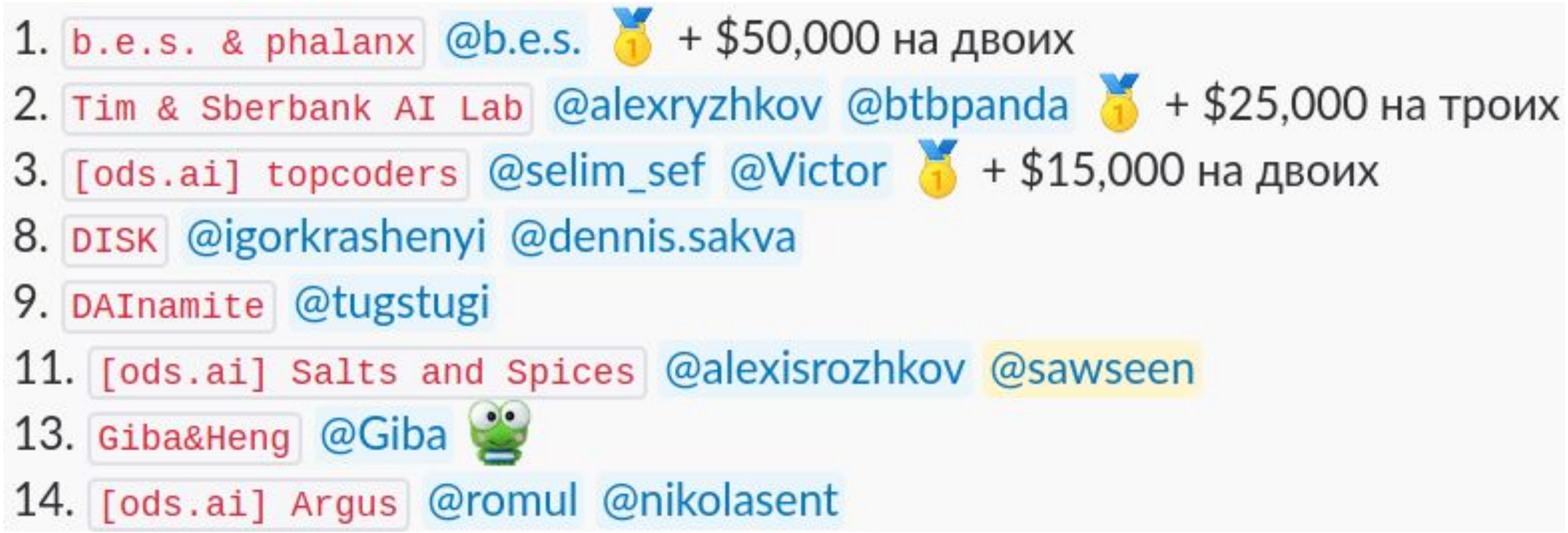
“The idea of jigsaw puzzles work great for natural images for the reason that it implicitly exploits continuity of edges within images. Unfortunately, seismic images don't share such properties. Two images from different areas of the world (with some constraints that it is a topic for another post!) can be made to look adjacent/similar under a similar metric. This is because they don't have pronounced discontinuities that make it seem adjacent.”

Questions for organizators

1) Why did train and test data come from the same places? Or did it?
2) Why there was no full salt tiles?
3) Why tiles were so small?



ODS achievements: half of gold



1. `b.e.s. & phalanx` @b.e.s. 🥇 + $50,000 на двоих
2. `Tim & Sberbank AI Lab` @alexryzhkov @btbpanda 🥇 + $25,000 на троих
3. `[ods.ai] topcoders` @selim_sef @Victor 🥇 + $15,000 на двоих
8. `DISK` @igorkrashenyi @dennis.sakva
9. `DAInamite` @tugstugi
11. `[ods.ai] Salts and Spices` @alexisrozhkov @sawseen
13. `Giba&Heng` @Giba 🐸
14. `[ods.ai] Argus` @romul @nikolasent

1st place solution: the first stage

b.e.s.

- Input: 101 -> resize to 192 -> pad to 224
- Encoder: ResNeXt50

Phalanx

- Input: 101 -> resize to 202 -> pad to 256
- Encoder: **Resnet34**, ResNext50

Memes - 0

oyaroshevskiy 12:08 AM
Screenshot from 2018-10-12 00-07-49.png

19 8 6

n01z3 12:04 AM
mem++.jpg

SpaceDoge 11:13 PM

перед шейкапом

resnet34.png

ЕБАНЁТ
НЕ ДОЛЖНО...
34

49 20 10 8 20 3 19 3

1st place solution: the second stage

Confident pseudo labeling: confidence as a percentage of confident pixel predictions ($0.2 < \text{probability} < 0.8$)

b.e.s.

- ResNeXt50 was pretrained on confident pseudolabels; and 5 folds were trained on top of them

Phalanx

- Added 1580 pseudolabels to each of 5 folds and trained the model from scratch

1st place solution: the third stage

All pseudo labels from the second stage were taken

Phalanx

- Added 1580 pseudo labels to each of 5 folds and trained the model from scratch
- 2 different models with:
    - Feature Pyramid Attention (https://arxiv.org/pdf/1805.10180.pdf)
    - Global Attention Upsample (https://arxiv.org/pdf/1805.10180.pdf)
    - Global Convolutional Network (https://arxiv.org/pdf/1703.02719.pdf)

Final model is a blend of 2nd stage and 3rd stage

https://www.kaggle.com/c/tgs-salt-identification-challenge/discussion/69291

Other things that worked

1) SENet154 and other fat encoders, extremely slow
2) Symmetric lovasz loss (with minus sign)
3) A lot of models (up to 200)
4) Simulated shakeup
5) SGDR with lr update per each iteration instead of each epoch

Other things that sometimes worked

1) Adam
2) Swa
3) Mosaic based folds
4) Depth stratification
5) Upsample instead of Deconvolution in decoder
6) Deeply supervised network with mask edge prediction
7) Best thresholding by KL divergence between train and test distributions
8) CBAM: Convolutional Block Attention Module (https://arxiv.org/pdf/1807.06521.pdf)

My solution

https://github.com/iasawseen/Kaggle-TGS-salt-solution

Thank you for your attention!