Salt Identification Challenge

Kenneth Emeka Odoh

15th Nov 2018 (Kaggle Data Science Meetup | SFU Ventures Lab)

Vancouver, BC

Table of Content

Bio

Education

- Masters in Computer Science (University of Regina) 2013 - 2016
- A number of MOOCs in statistics, algorithms, and machine learning

Work

- Research assistant in the field of Visual Analytics 2013 - 2016
  - { https://scholar.google.com/citations?user=3G2ncgwAAAAJ&hl=en }
- Software Engineer in Vancouver, Canada 2017 -
- Regular speaker at a number of Vancouver AI meetups, organizer of distributed systems meetup, and Haskell study group and organizer of Applied Cryptography study group 2017 -

22
NOV

Thursday, November 22, 2018

Study Group: Applied Cryptography For Engineers

Hosted by Kenneth Emeka Odoh
From Vancouver Tech Meetup
Public group

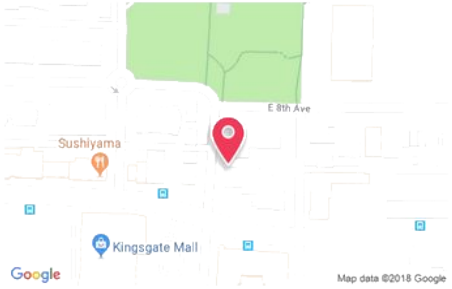
You're going 8 people going

Share:

Organizer tools

Thursday, November 22, 2018
6:00 PM to 8:00 PM
Add to calendar

7GATE VENTURES
#401 68 Water Street · Vancouver, bc



Details

Our goal is to learn the fundamentals of modern cryptography. A solid foundation in cryptography is important for software developers that want to design secure modern computer systems. If you are interested in topics such as Distributed Systems, Cloud Computing, Blockchain and of course, Security, this is the study group for you.

Competition

- **Geology** is the earth science that deals with the study of the properties of the earth crust.
- This competition is related to **geology**.
- Competition focuses on **structured prediction**.
- Salt occurs when drilling for crude oil or gas in a number of oil fields around the world.
- Given a **radar** image, find the patches of salt in the images. This is a classic image segmentation problem.

The prizes consist of

- 1st Place - $ 50,000
- 2nd Place - $25,000
- 3rd Place - $ 15,000
- 4th Place - $ 10,000

Competition rules

- 5 submissions per day.
- External data is not allowed.

Leaderboard

#	△pub	Team Name	Score	Entries	Last
1	—	b.e.s. & phalanx	0.896469	316	11d
2	▲ 6	Tim & Sberbank AI Lab	0.895906	251	11d
3	▲ 6	[ods.ai] topcoders	0.895456	270	11d
4	▼ 2	SeuTao@CHAN&Venn&Kele@Co...	0.895439	403	11d
5	▼ 1	Kent AI Lab | Ding Han | Renan	0.894755	452	11d
6	▼ 3	bestfitting	0.894738	214	11d
7	▲ 16	aki.	0.894461	29	12d
8	▼ 2	DISK	0.892384	337	11d
9	▼ 4	DAInamite	0.892245	202	11d
10	▲ 7	无人工不智能	0.891562	268	11d
11	▲ 2	[ods.ai] Salts and Spices	0.891518	183	11d
12	—	Learning the Future	0.891371	441	11d
13	▼ 6	Giba&Heng	0.891198	240	11d
14	▲ 5	[ods.ai] Argus	0.891060	68	11d
15	▲ 3	adam and atom	0.891051	361	11d
16	▼ 6	bird@cortexlabs|Guido	0.890973	200	11d
17	▼ 2	Less is More	0.890835	15	11d
18	▲ 18	Mii	0.889909	228	11d
19	▲ 19	ZZZ	0.889857	141	11d
20	▲ 26	yunhai	0.889710	140	11d

#	△1w	Team Name	Score	Entries	Last
1	▲ 3	b.e.s. & phalanx	0.888333	316	11d
2	—	SeuTao@CHAN&Venn&Kele@Co...	0.887216	403	11d
3	▲ 15	bestfitting	0.886449	214	11d
4	▼ 3	Kent AI Lab | Ding Han | Renan	0.885866	452	11d
5	▲ 4	DAInamite	0.885849	202	11d
6	▼ 1	DISK	0.884683	337	11d
7	▼ 4	Giba&Heng	0.884249	240	11d
8	—	Tim & Sberbank AI Lab	0.884216	251	11d
9	▲ 2	[ods.ai] topcoders	0.883733	270	11d
10	▲ 3	bird@cortexlabs|Guido	0.883666	200	11d
11	▲ 3	[ods.ai] Power Fist	0.882916	269	11d
12	▼ 5	Learning the Future	0.881849	441	11d
13	▲ 31	[ods.ai] Salts and Spices	0.881466	183	11d
14	▲ 18	im_crm	0.879699	248	11d
15	new	Less is More	0.878816	15	11d
16	▲ 1	Dmitriy, Terence & Takato	0.878449	363	11d
17	▲ 22	无人工不智能	0.878266	268	11d
18	▼ 2	adam and atom	0.878266	361	11d
19	▲ 34	[ods.ai] Argus	0.877966	68	11d
20	▲ 2	Kyle	0.877933	112	11d

Private and public leaderboard (side by side comparison)

Types of Segmentation

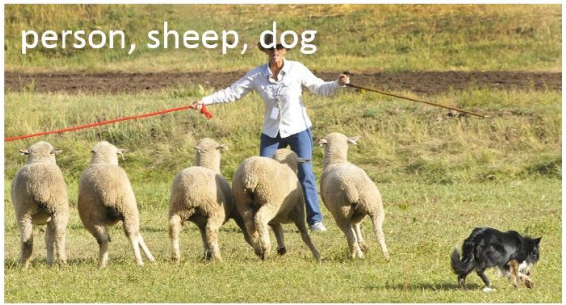
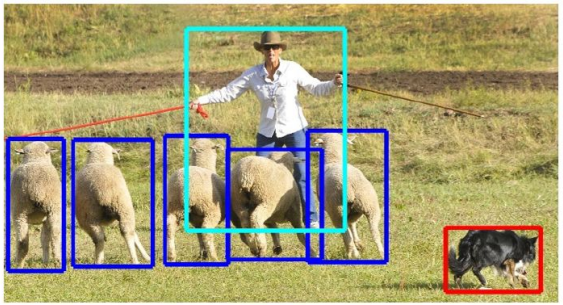
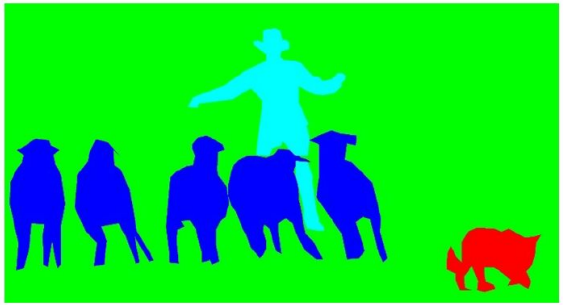


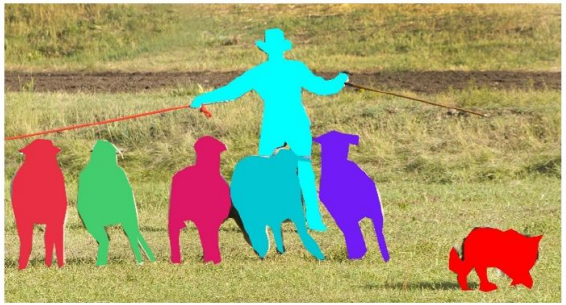
Image classification



Object localization



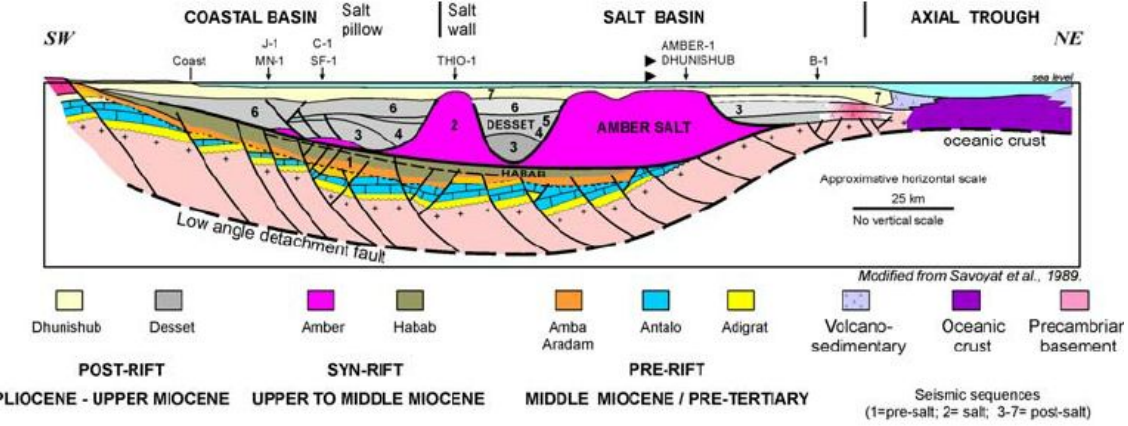
Semantic segmentation

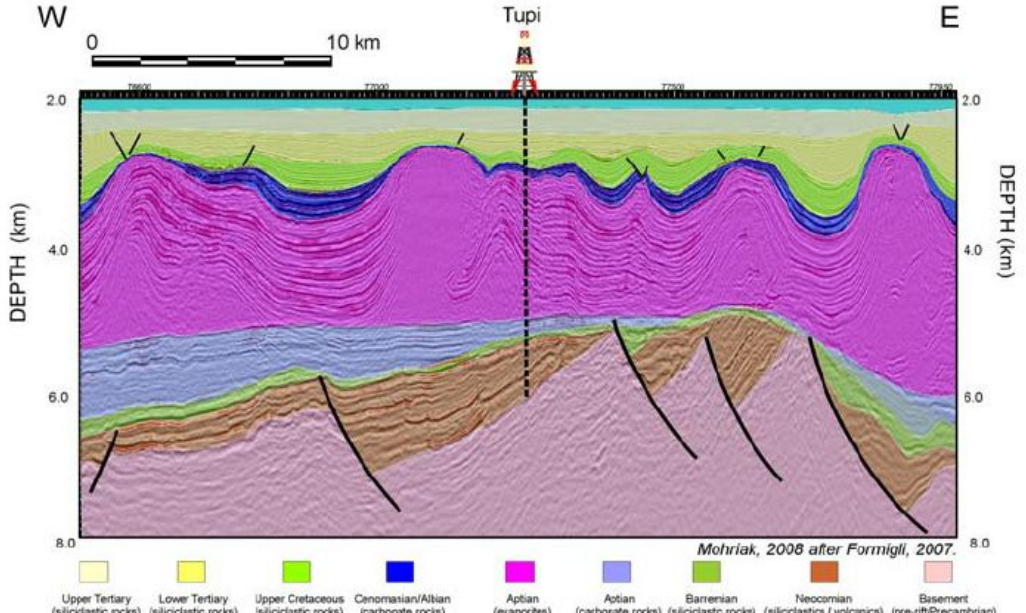


Instance segmentation (this competition)

https://www.meetup.com/LearnDataScience/events/253121103/

@matt & @bruce





This knowledge from the figure was exploited

- Show the impact of depth on the plates
- Create a local CV validation by stratifying using the depth. This matches the public leaderboard to a few decimal places.

[https://hal.archives-ouvertes.fr/file/index/docid/730844/filename/Mohriak_LeroyGSL2012.pdf]

Data

- Number of training set: 4000 images
- Number of testing set: 18000 images
- Original image size (101 x 101) pixels

Each image has a mask in training set.

Only images are given in the testing set, then you have to predict the mask.

Competition lasted for **3 months**.

Submission format is a compression format of **Run length encoding**

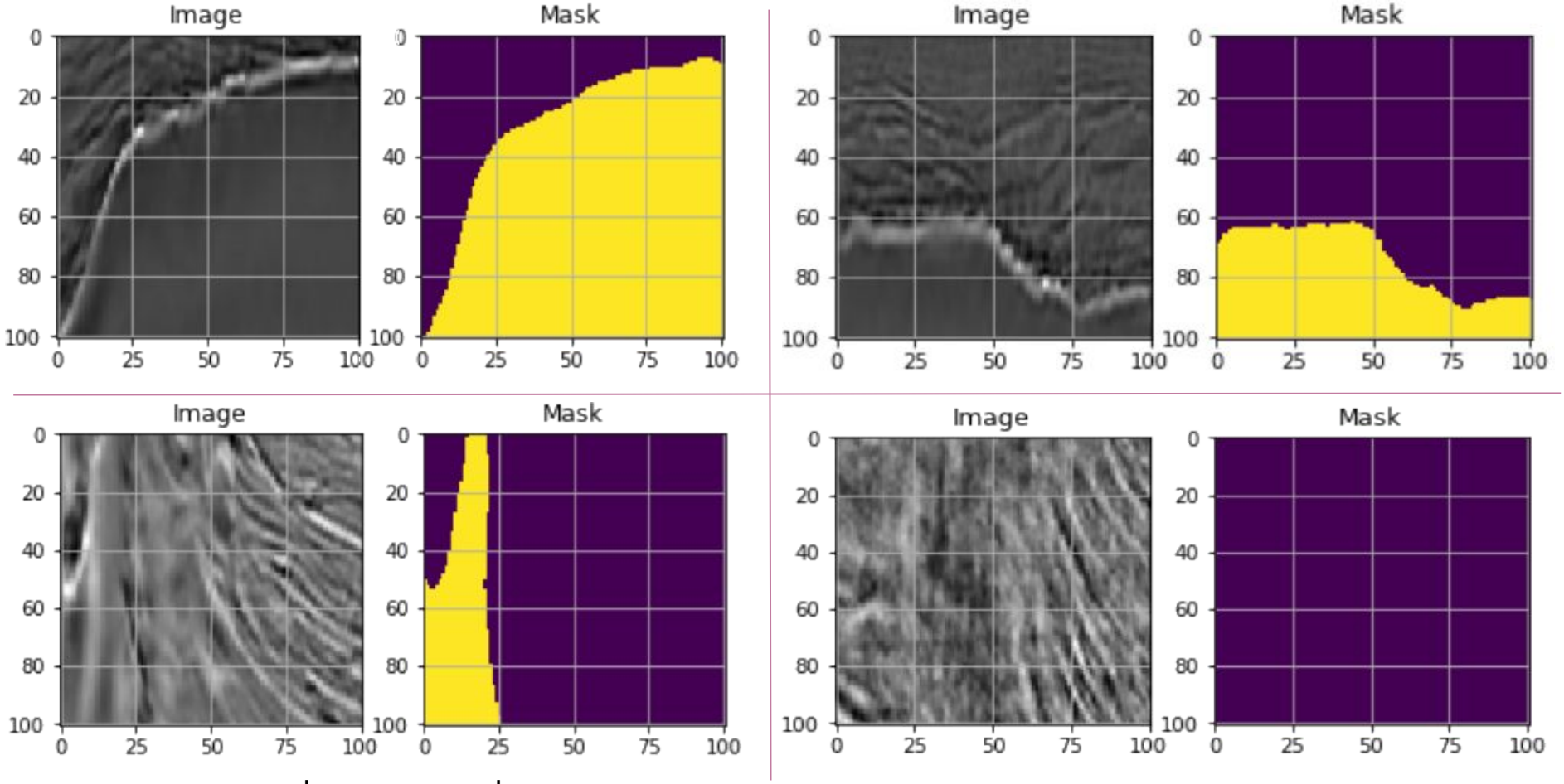
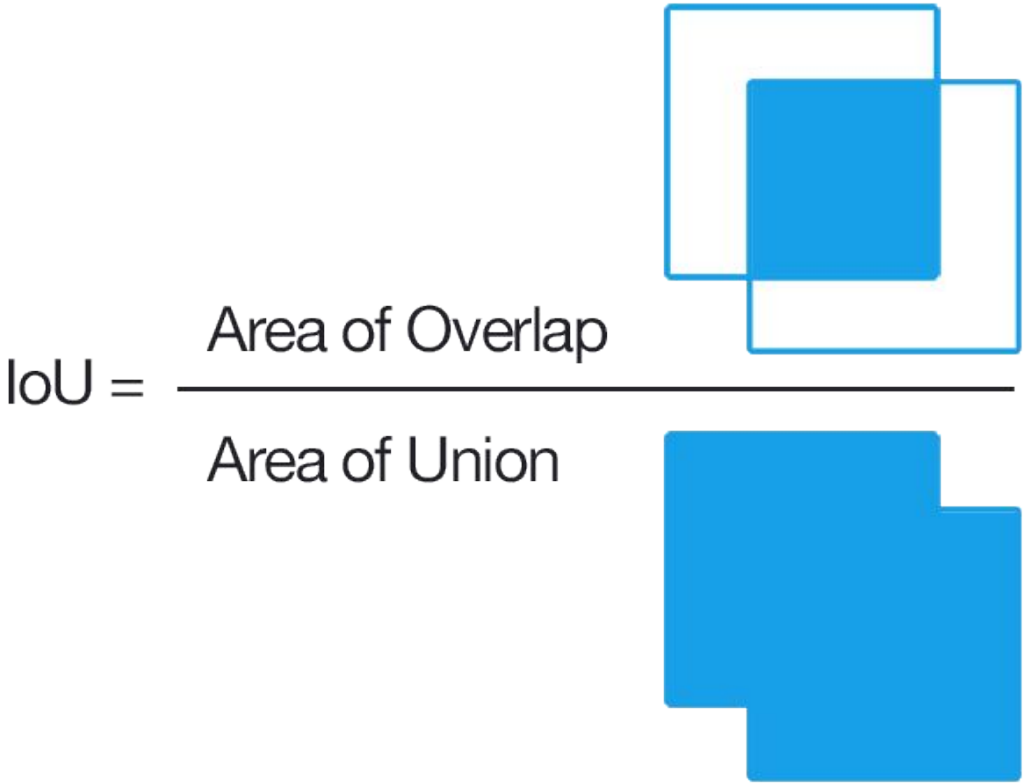


Image vs mask
{https://www.kaggle.com/skainkaryam/basic-data-visualization-using-pytorch-dataset#}

Evaluation metric



Average IOU as a mean of IOUs by varying the thresholds {0.5, 0.55, … , 0.95}.

[https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/]

Winning Solution: 1st place

Summary can be found in {https://www.kaggle.com/c/tgs-salt-identification-challenge/discussion/69291 }.

- encoder: ResNeXt50 / ResNet34
- center: Feature Pyramid Attention
- decoder: conv3x3, T conv, scSE + hyper columns
- loss: BCE, Lovász - 2-level pseudo-labeling
- 2 model final ensemble

It is good to note the following:

- Loss: BCE for classification and Lovasz for segmentation
- Created a local CV by stratifying by depth which correlated with LB.
- segmentation zoo {https://github.com/qubvel/segmentation_models }
- for more information on hypercolumn {https://en.wikipedia.org/wiki/Cortical_column }.
- scSE, Spatial-Channel Sequeeze & Excitation {https://arxiv.org/abs/1803.02579 }

1st stage training

Input: 101 -> resize to 192 -> pad to 224
Encoder: ResNeXt50 pretrained on ImageNet
Decoder: conv3x3 + BN, Upsampling, scSE
Optimizer: RMSprop. Batch size: 24

Loss: BCE+Dice. Reduce LR on plateau starting from 0.0001

Loss: Lovasz. Reduce LR on plateau starting from 0.00005

Loss: Lovasz. 4 snapshots with cosine annealing LR, 80 epochs each, LR starting from 0.0001

Some attempts at ensembling

5-fold ResNeXt50 had 0.864 Public LB (0.878 Private LB)

5-fold ResNet34 had 0.863 (0.880 Private)

Their ensemble scored 0.867 (0.885 Private)

2nd Stage Training

The first stage provides pseudolabels using probability. Confidence was measured as a measure of percentage count of pixel predictions (probability < 0.2 or probability > 0.8)

They used ResNeXt50 was pretrained on confident pseudolabels; and 5 folds were trained on top of them. 0.871 (0.890 Private)

3rd Stage Training

We took all the pseudolabels from the 2nd stage ensemble, and phalanx trained 2 models:
resnet_34_pad_128

Input: 101 -> pad to 128
Encoder: ResNet34 + scSE (conv7x7 -> conv3x3 and remove first max pooling)
Center Block: Feature Pyramid Attention (remove 7x7)
Decoder: conv3x3, transposed convolution, scSE + hyper columns
Loss: Lovasz

resnet_34_resize_128
Input: 101 -> resize to 128
Encoder: ResNet34 + scSE (remove first max pooling)
Center Block: conv3x3, Global Convolutional Network
Decoder: Global Attention Upsample (implemented like senet -> like scSE, conv3x3 -> GCN) + deep supervision
Loss: BCE for classification and Lovasz for segmentation

Training overview:

Optimizer: SGD. Batch size: 32.

Pretrain on pseudolabels for 150 epochs (50 epochs per cycle with cosine annealing, LR 0.01 -> 0.001)

Finetune on train data. 5 folds, 4 snapshots with cosine annealing LR, 50 epochs each, LR 0.01 -> 0.001

resnet_34_pad_128 had 0.874 (0.895 Private)

resnet_34_resize_128 had 0.872 (0.892 Private)

Final Model

Final model is a blend of ResNeXt50 from the 2nd stage and resnet_34_pad_128 from the 3rd stage with horizontal flip TTA: 0.876 Public LB (0.896 Private LB).

Augmentations

The list of augmentations

HorizontalFlip(p=0.5)

RandomBrightness(p=0.2,limit=0.2)

RandomContrast(p=0.1,limit=0.2)

ShiftScaleRotate(shift_limit=0.1625, scale_limit=0.6, rotate_limit=0, p=0.7)

Code: https://www.kaggle.com/youhanlee/1st-solution-reproducing-1-unet-resnet34-se/notebook#

Winning Solution: 9th place

The single fold score of SENet154 was 0.882/0.869 and with 10 folds reflective padding + resizing 0.890/0.875.

- AdamW with the Noam scheduler
- cutout {https://arxiv.org/abs/1708.04552 }
- Stochastic Weight Averaging (SWA) after the training on the best loss, pixel accuracy (+0.004).
- They used gitlab to manage experiments.
- Using a symmetric version of Lovasz which gave better result

```
def symmetric_lovasz(outputs, targets):
    return (lovasz_hinge(outputs, targets) + lovasz_hinge(-outputs, 1 - targets)) / 2
```

Summary available {https://www.kaggle.com/c/tgs-salt-identification-challenge/discussion/69053 }

Winning Solution: 11th place

Summary can be found
{https://www.kaggle.com/c/tgs-salt-identification-challenge/discussion/69093}
Model: UNet-like architecture
Backbone: SE ResNeXt-50, pretrained on ImageNet

Decoder features:

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

Tile size adaptation:

- Pad 101 -> 128

Augmentations:

- Random invert
- Random cutout
- Random gamma-correction
- Random fixed-size crop
- Random horizontal flip

Optimizer:

- SGD: momentum 0.9, weight decay 0.0001
- Batch size: 16
- Finding LR find using fast.ai course - 5e-2
- Cosine annealing for training
- used **snapshots** for ensembling.

Loss:

deep supervision it’s a combination of 3 losses - classification loss, pure segmentation loss , total loss

Classification loss - BCE

Segmentation losses - BCE + Lavasz Hinge*0.5

Segmentation loss was evaluated after cropping masks and predictions back to 101 -

Cross-validation

5-fold random split

Only depth stratification worked

My Solution

- A number of neural architectures like dilated unet, ‘vanilla’ unets with varying depths with batchnorm or dropout.
- Used loss function like lovasz_loss in second stage of training gives boost after binary cross entropy was used in the first stage of training.
- Standardizing the image gives better results than just dividing by 255.
- Cyclical learning rate policy (CLR) with restart gave some boost in training {https://arxiv.org/abs/1506.01186}.
- Don’t use ReduceLROnPlateau with CLR at the same time. You can **confuse** your network.
- Training with reduce on plateau in the spirit of simulated annealing tends to restore learning after relatively long epochs of degrading performance in some cases.

- Be careful with **early stopping**. It may be a blessing or a curse.
- Pre-trained networks in my experimentation were not better than training from scratch.
- Did Test-time argumentation with only horizontal flipping.
- Got best 5 models with scores from **0.74** - **0.79**.
- Perform CRF is local CV is <0.75. Under this condition, I get some 0.2 boost in LB score.
- Ensemble using weighted average using the LB scores as weight.See more information in **conclusions** section.

Position: 1482 / 3267 teams

Private LB : **0.824**

My System Setup

- I used a GTX 1080 with 32GB Ram and ITB hard disk. Luckily, I did not have any need for **heaters** in my room.
- I create a hypothesis and try to verify it in the code. I will out a number of Python scripts.
- Run the Python script with a bash script. Each bash script runs a list of Python script which I consider as an experiment.
    - The benefit of this setup, if a script files, the other scripts with still runs. This will reduce time of development.
- Jupiter was better if I had to visualize.
- Ran over 100 experiments, validating a number of hypotheses.
- Extensive diagnostics to figure out when experiments go badly and end experiments to save time.

What didn’t work

- Performing fft on the image and using the amplitude gives very bad performance. This made me **suspect** that the phase angles were important.
- Focal loss, lovasc loss seem to give participant boost if used in first stage training. However, it gives output in many cases that is meaningless. Although, someone claimed on the forum that it required a special kind of gradient flow but logit on the softmax layer of the network
- Training-time argumentation could not work for me.
- Techniques like DANet, OCNet, pixel shuffling, transposing did not work for some people.
- Pseudo-labeling worked for some and not for others.

Discussions

- Private LB score was higher for many teams as it has been hypothesized that organizers cherry picked “few-pixel masks (which were hard to predict, incurred high penalty and probably weren’t important from business standpoint) from private set”.
- Rule of thumb for using pseudo-labeling: it is used when the training set is small. However, it is difficult to determine sample size and to tune properly making hard to use in production.
- Treat as a classification + segmentation problem instead of a straightforward segmentation.
- Don't trust CV, trust stability {https://www.kaggle.com/c/tgs-salt-identification-challenge/discussion/69051 }
- There was a well-written post about the unethical nature of the competition based on a climate change activist.This person was against the oil industry and managed to clearly articulate his point, urging people to abandon the competition.

Common Approaches

- Data preprocessing
  - Data augmentation
  - Upsampling
- Build the model
  - Variations of neural architectures (encoder-decoder architecture)
  - Creating more models
    - Perturb the model (change configuration of network)
    - Perturb the data (fft, cumsum, glcm, watershed e.t.c)
- Post-processing
  - Conditional random field
  - Thresholding
  - Downsampling
- Ensembling
  - Averaging of rle blend (weighted average)

Data Augmentation

- The goal is to perturb the data to **intensify** the signal.
- Caveat: using the wrong argumentation can worsen performance.
- During the competition, I came up with a rule of thumb (Intuition) behind data argumentation
  - Crop if the majority of the data on a 2D spectrogram are concentrated in an area in the image. Taking the crop in the concentrated area would intensify the signal.
  - Flip is if you observe that the patterns in the data tend to be symmetric in either horizontal and vertical axis.
  - Augmentation requires some **domain knowledge** of how the distribution of data may be in the **wild**.
  - imgaug library { https://github.com/aleju/imgaug }
  - Albumentations {https://github.com/albu/albumentations/releases/tag/v0.1.1}

The best argumentation was **horizontal flip** in this competition.

Conditional Random Field (CRF)

- CRF is a **Markov network** suitable for structured prediction.
- Imagine that the features of the data set are highly correlated. As a result there are lots of overlaps and redundancy. For example, pixels in an image are similar in local neighbourhood regions.
- **Naive Bayes** would ignore all the correlations between attributes which is very informative due to strong independence assumption leading to skewed probability distribution.
- Adding edges to capture correlations forms a densely connected network and even harder to figure out the connections.
- Model **p(Y|X)**, instead of **p(X,Y)**. Makes us to care less about correlation between features.

- Uses Gibbs (Boltzmann) distribution. See url {https://en.wikipedia.org/wiki/Boltzmann_distribution } for more information.

conditional random field

CRF Representation

$$\Phi = \{\phi_1(\boldsymbol{D}_1), \ldots, \phi_k(\boldsymbol{D}_k)\}$$

Gibbs dist.

factors ← scope

unnormalized measure

$$\tilde{P}_\Phi(\boldsymbol{X}, \boldsymbol{Y}) = \prod_{i=1}^{k} \phi_i(\boldsymbol{D}_i)$$

← multiply factors

← partition function of X

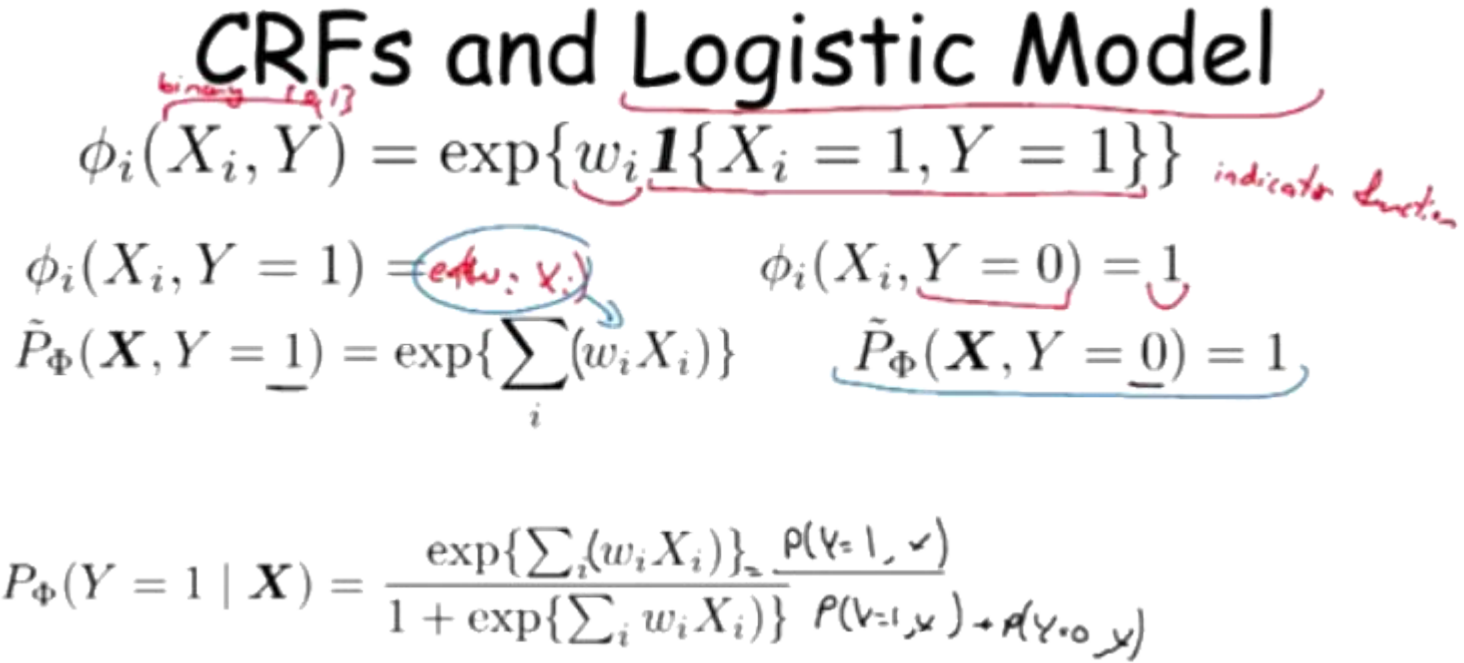
$$Z_\Phi(\boldsymbol{X}) = \sum_{\boldsymbol{Y}} \tilde{P}_\Phi(\boldsymbol{X}, \boldsymbol{Y})$$

← sum over Y

$$\tilde{P}_\Phi(\boldsymbol{Y}|\boldsymbol{X}) = \frac{1}{Z_\Phi(\boldsymbol{X})} \tilde{P}_\Phi(\boldsymbol{X}, \boldsymbol{Y})$$

[@Daphne Koller, Coursera]

- CRF is parametrized as a **Gibbs distribution**, but normalizalized differently.



- CRF can be a **logistic regression** under some conditions.
- No need to model over distributions that we don't care about.
- Allows for very expressive features, without the limiting **independence** assumption.[@Daphne Koller, Coursera]

Applications of **CRF** consist of the following, but not limited to these

- image segmentation
  - input: is a set of pixels with values
  - output: class for every pixel
- Text processing
  - input: word in a sentences
  - output: labels of the word (named entity recognition)

CRF can exploit neighbourhood information when classifying thereby making it look like **sequence modeling** of some sort.

For more information on using CRF with image segmentation, let us **visit**

{**https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/classes/cs294_f99/notes/lec7/lec7.htm** }.

The version of CRF used by a number of participants is {https://github.com/lucasb-eyer/pydensecrf} based on the paper {https://www.philkr.net/papers/2013-06-01-icml/2013-06-01-icml.pdf }

Handling large data set

- Reduce image size.
- Use smaller batch size.
- Use Python generators. A number of deep learning packages provides a fit-generator method. This allows for careful batch processing. See tutorial on generator and coroutines {https://www.dabeaz.com/usenix2009/generators/GeneratorUSENIX.pdf }, {https://www.dabeaz.com/coroutines/Coroutines.pdf }
- Taking a sliding window to crop the image into manageable sizes and pass through the network and aggregate at the test time.
- Empirically, I found **batchnorm** to be more computationally expensive than **dropout** in keras.

Conclusions

- If you are using save-best model, then it is better to train for longer epoches.
- The trick to **improving results** is in identifying images without salt patches.
  - Participant tried to create a network to identify empty mask. This helped to intensify the **signal**.
  - I tried to do so at the point of ensembling. Note, this will only work if your models are as **diverse** as possible.
- Avoid using batchnorm with dropout in the same network. This leads to bad performance {https://arxiv.org/pdf/1801.05134.pdf }
- Systematically reducing the dropout rate as information flows through the encoder-decoder network leads to improved performance.
- Upsampling during training especially for the dilated u-nets tends to improve performance.

Thanks for listening

https://github.com/kenluck2001?tab=repositories

@kenluck2001

https://www.linkedin.com/in/kenluck2001/

kenneth.odoh@gmail.com

References

1. Daphne Koller, Probabilistic graphical models, Coursera, {https://www.coursera.org/lecture/probabilistic-graphical-models/conditional-random-fields-UJ1Ke }

Past Talks

Some of my past talks

- ★ Compressed sensing using generative models, https://www.slideshare.net/kenluck2001/compressend-sensing-using-generative-model-122976230 , 2018
- ★ Tutorial on Cryptography, slide: https://www.slideshare.net/kenluck2001/crypto-bootcamp-108671356 , 2018
- ★ Landmark Retrieval & Recognition, slide: https://www.slideshare.net/kenluck2001/landmark-retrieval-recognition-105605174 , video: https://youtu.be/YD6ihpBMyso , 2018
- ★ Tracking the tracker: Time Series Analysis in Python from First Principles, slide: https://www.slideshare.net/kenluck2001/tracking-the-tracker-time-series-analysis-in-python-from-first-principles-101506045 , 2018
- ★ WSDM Recommender System, slide: https://www.slideshare.net/kenluck2001/kaggle-kenneth , video: https://youtu.be/exwJmQzDBag , 2018