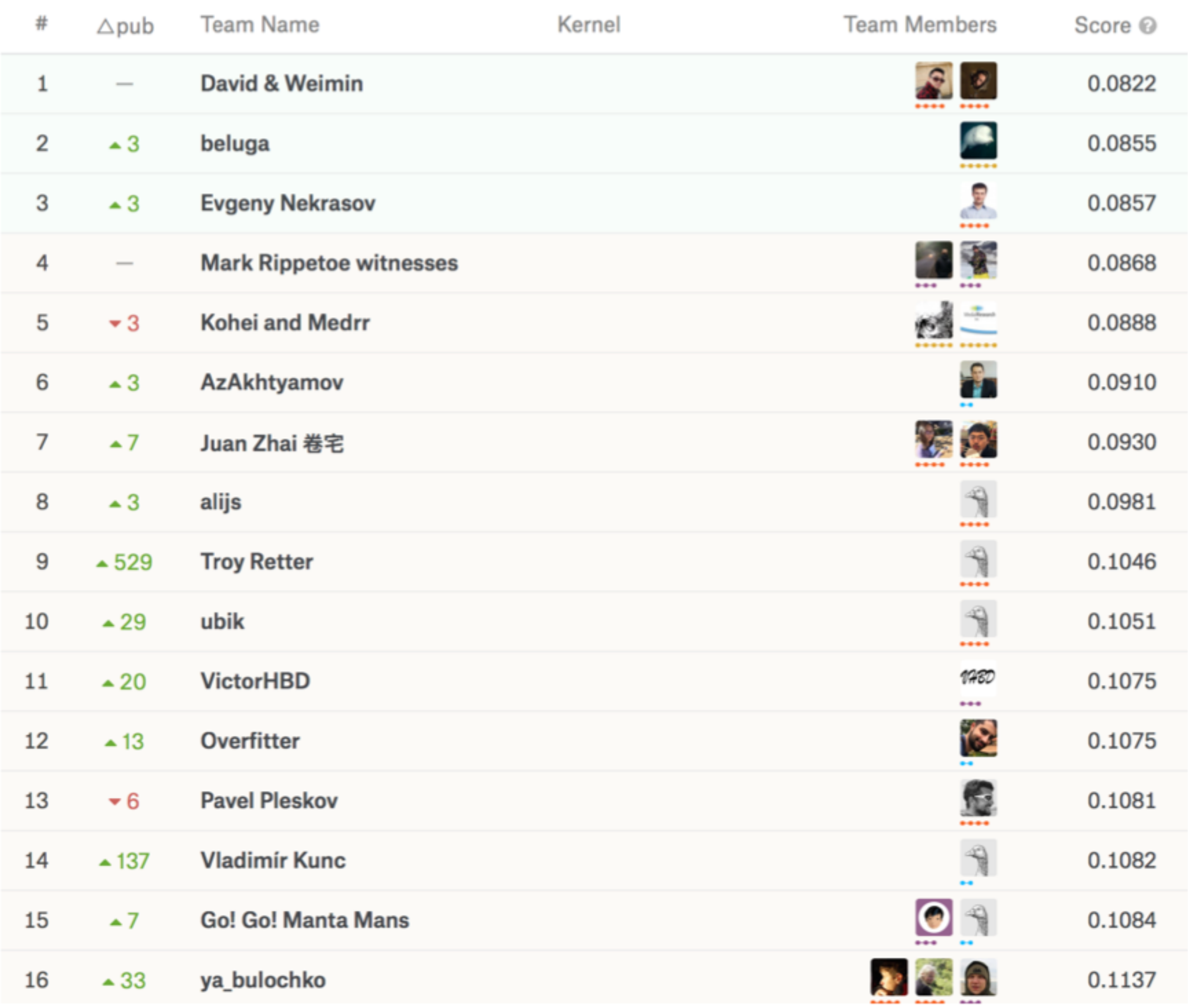
Ship or iceberg?

Kaggle Statoil/C-CORE Iceberg Classifier Challenge

ODS: @azzy
Azat Akhtyamov



#	△pub	Team Name	Kernel	Team Members	Score
1	—	David & Weimin			0.0822
2	▲3	beluga			0.0855
3	▲3	Evgeny Nekrasov			0.0857
4	—	Mark Rippetoe witnesses			0.0868
5	▼3	Kohei and Medrr			0.0888
6	▲3	AzAkhtyamov			0.0910
7	▲7	Juan Zhai 卷宅			0.0930
8	▲3	alijs			0.0981
9	▲529	Troy Retter			0.1046
10	▲29	ubik			0.1051
11	▲20	VictorHBD			0.1075
12	▲13	Overfitter			0.1075
13	▼6	Pavel Pleskov			0.1081
14	▲137	Vladimír Kunc			0.1082
15	▲7	Go! Go! Manta Mans			0.1084
16	▲33	ya_bulochko			0.1137

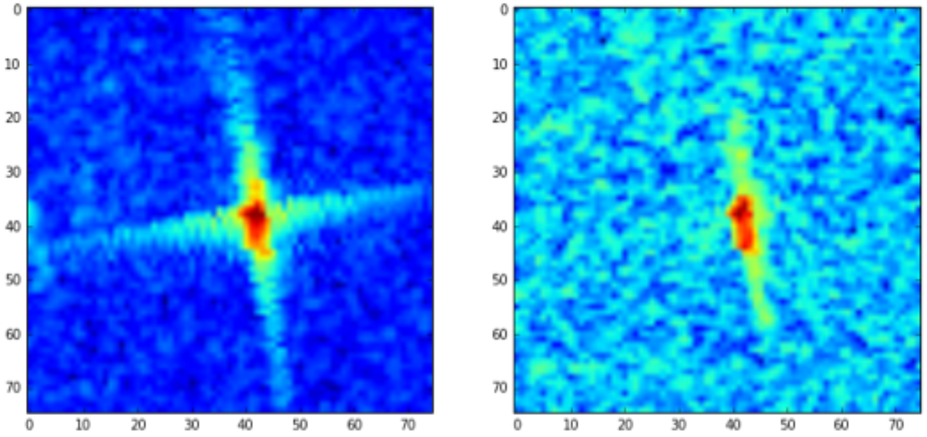
/3343

Description

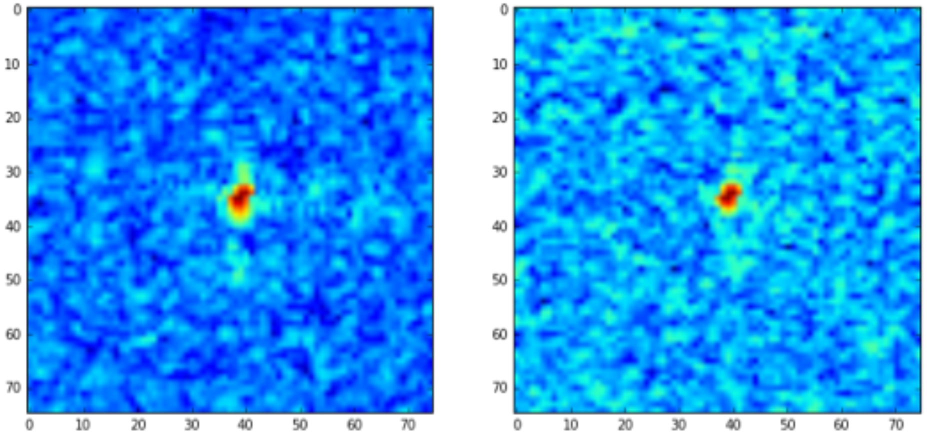
- 2 bands 75x75 + incidence angle
- Binary classification
- 1604 samples in train, 133 with NaN incidence angle
- 8424 samples in test, 5000 generated, no NaN incidence angle
- Evaluation metric: logloss
- Generated images excluded from private/public scoring
- Only 2 submission per day

Examples

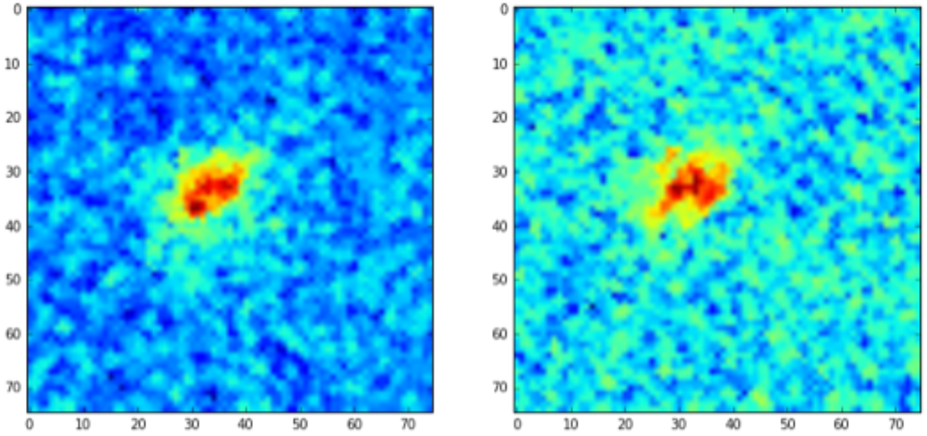
Ship:



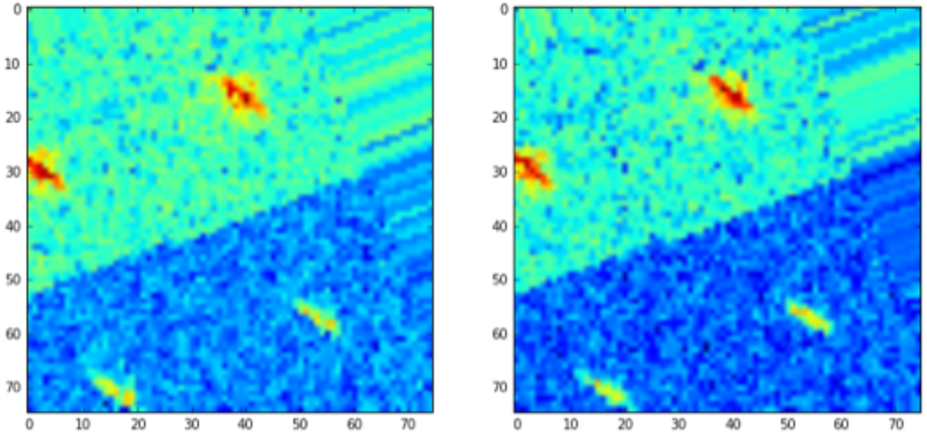
Hard case:



Iceberg:

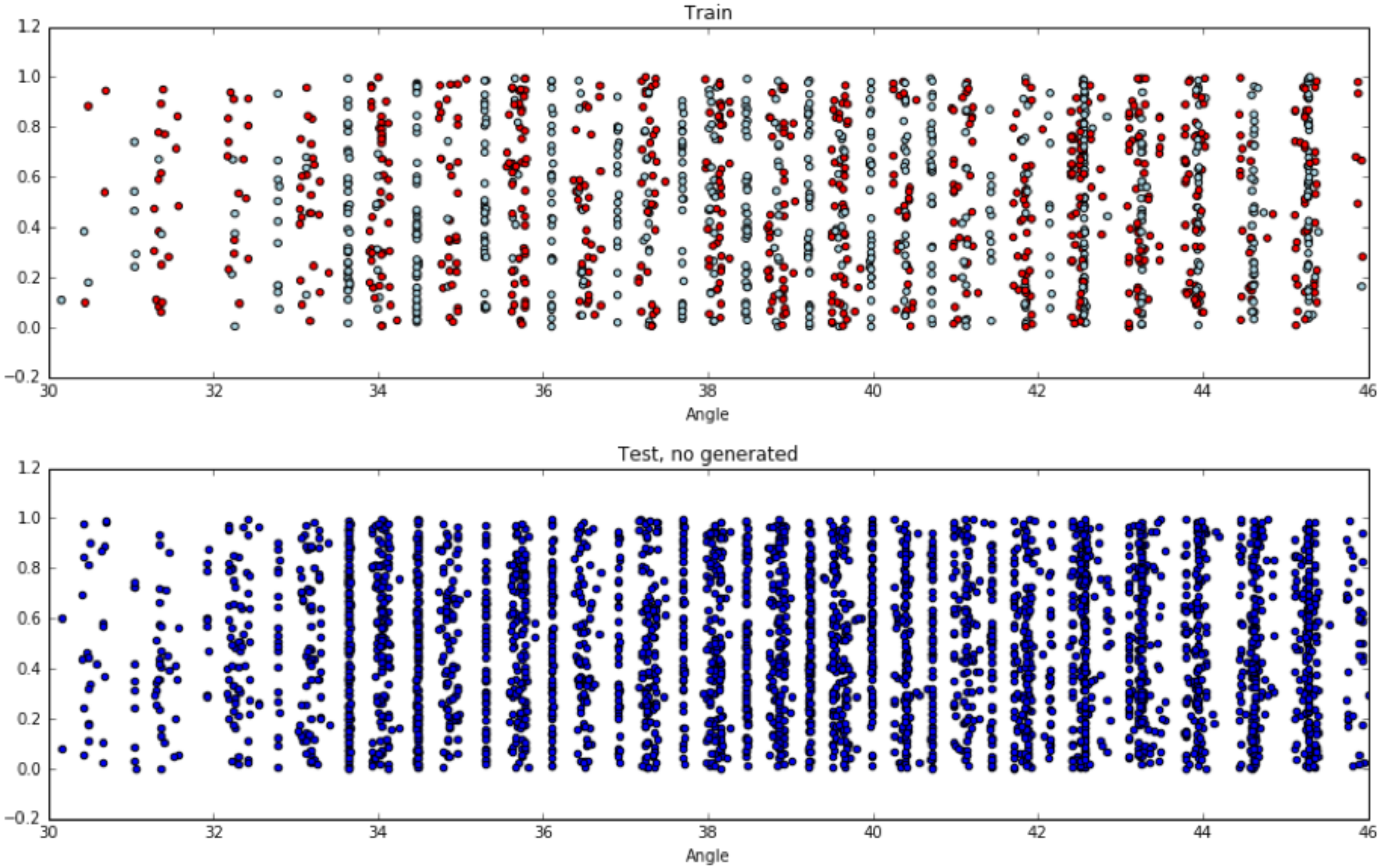


Generated:



Leak or feature?

Idea: icebergs with icebergs, ships with ships (remember Titanic?)

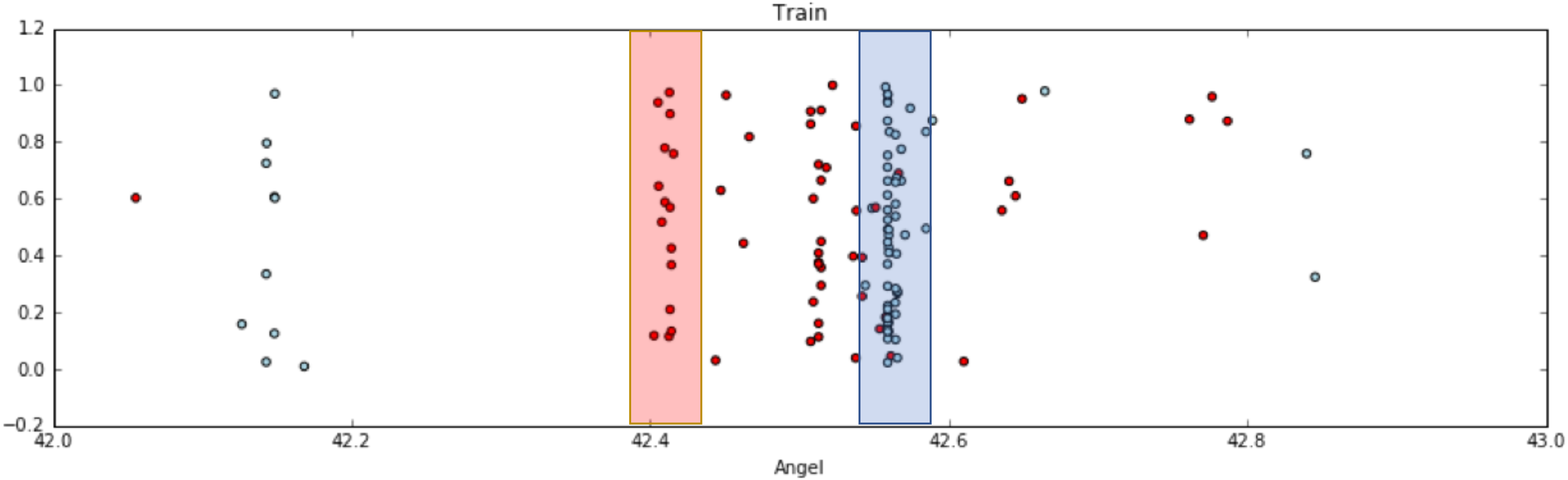


Leaky features

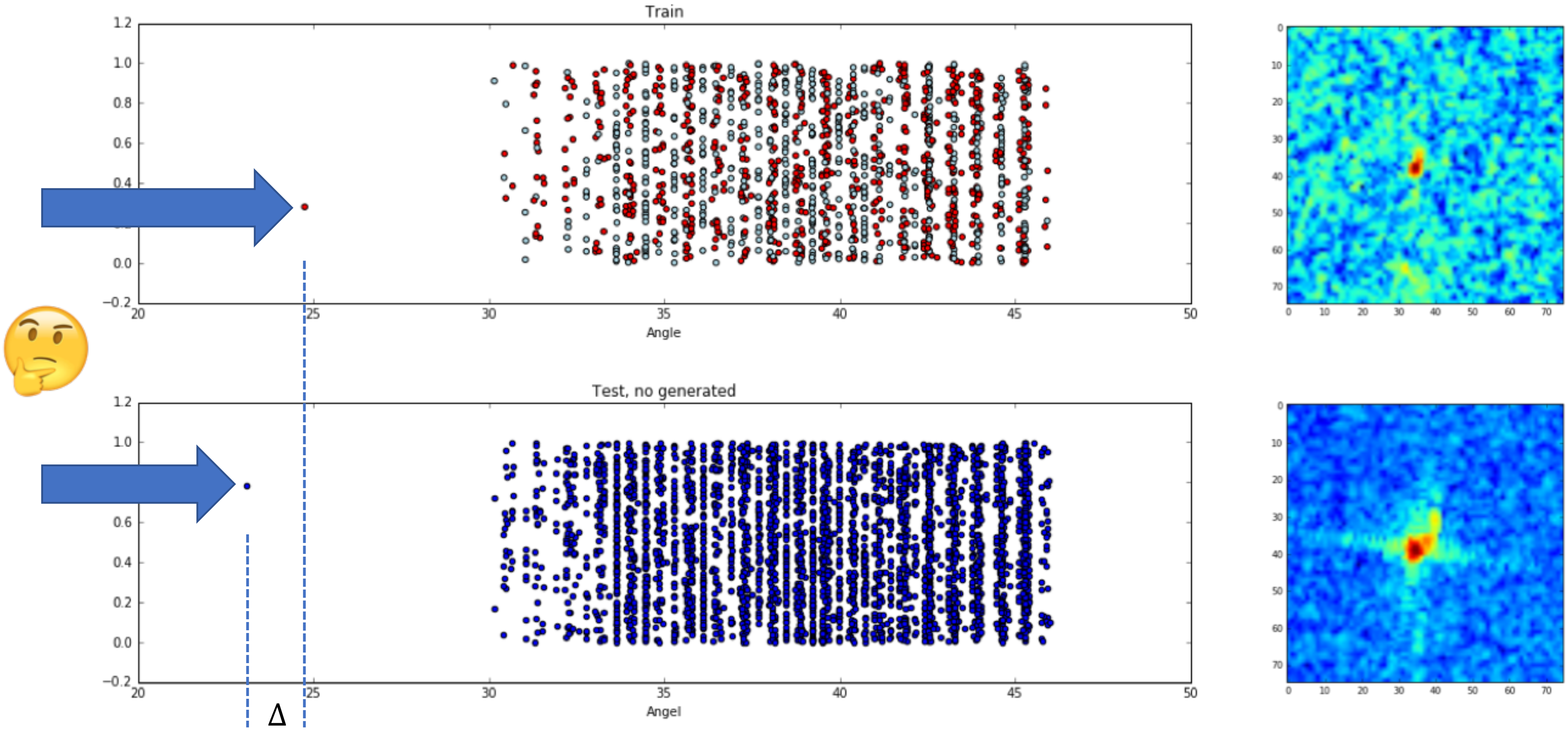
For every unique angle:

- Mean target
- Total count (include test set)
- Mean target in the neighborhood

Public 0.2106
Private **0.1965**



MORE LEAKS


Train
1.2
1.0
0.8
0.6
0.4
0.0
−0.2
20
25
30
35
40
45
50
Angle
Test, no generated
1.2
1.0
0.6
0.4
0.2
0.0
−0.2
Angel
Δ
0
10
20
30
40
50
60
70

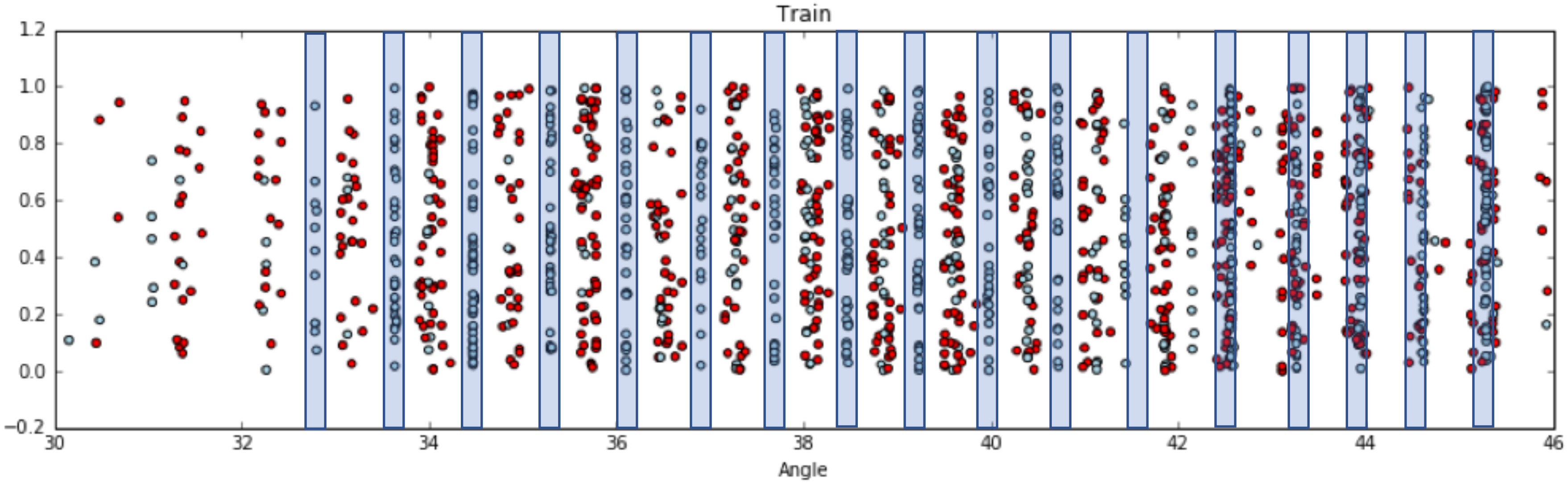
MORE LEAKS

$\Delta = 1.6741$, real step: $\frac{\Delta}{2} = 0.83705$

For every angle a:

- Mean target over $\beta \in [a - 10\Delta \pm \varepsilon, a - 9\Delta \pm \varepsilon, \dots, a + 9\Delta \pm \varepsilon, a + 10\Delta \pm \varepsilon]$
- Count samples over $\beta \in [a - 10\Delta \pm \varepsilon, a - 9\Delta \pm \varepsilon, \dots, a + 9\Delta \pm \varepsilon, a + 10\Delta \pm \varepsilon]$
- Mean target over area with center in $\beta \in [a - 10\Delta, a - 9\Delta, \dots, a + 9\Delta, a + 10\Delta]$

Here ε = 0.00005



2D model and pseudo-labeling

- Label images with $p < 0.01$ or $p > 0.99$ (nearly 3000 images)
- Train also on train images (ships with NaN angle)
- Augmentation: rotations, flips
- Trained 200 models, median prediction of top 100 models
- If previous model predicts $p \in [0.1, 0.9]$ then average, else use previous model

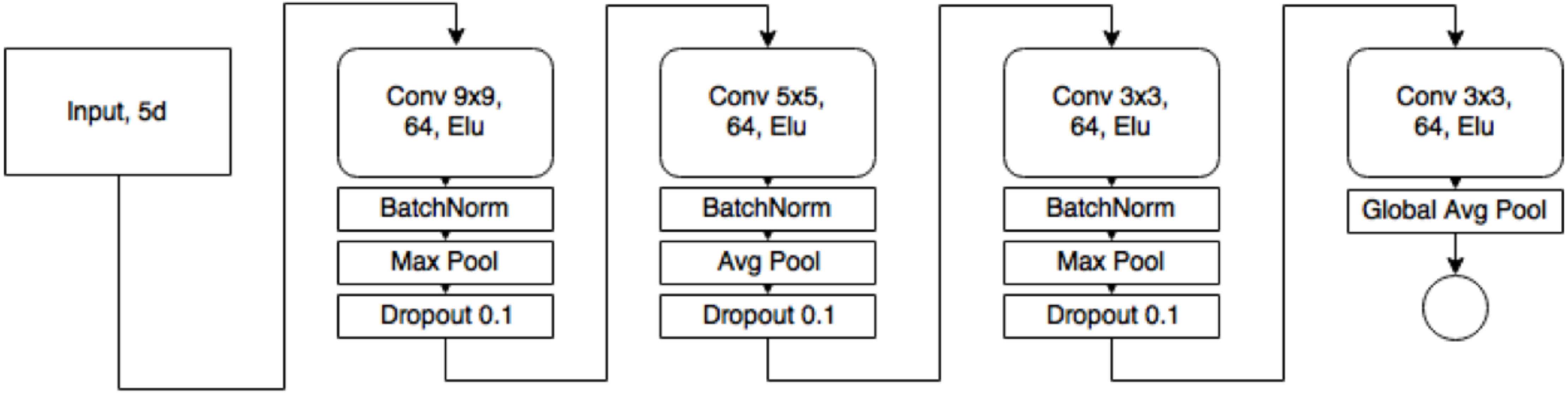
Public 0.0940
Private **0.0910**

- Better strategy: if the mean target for the angle is near 0.5 then average, else use previous model

Public 0.0984
Private **0.0873**

But this didn’t work…

5D CNN Architecture



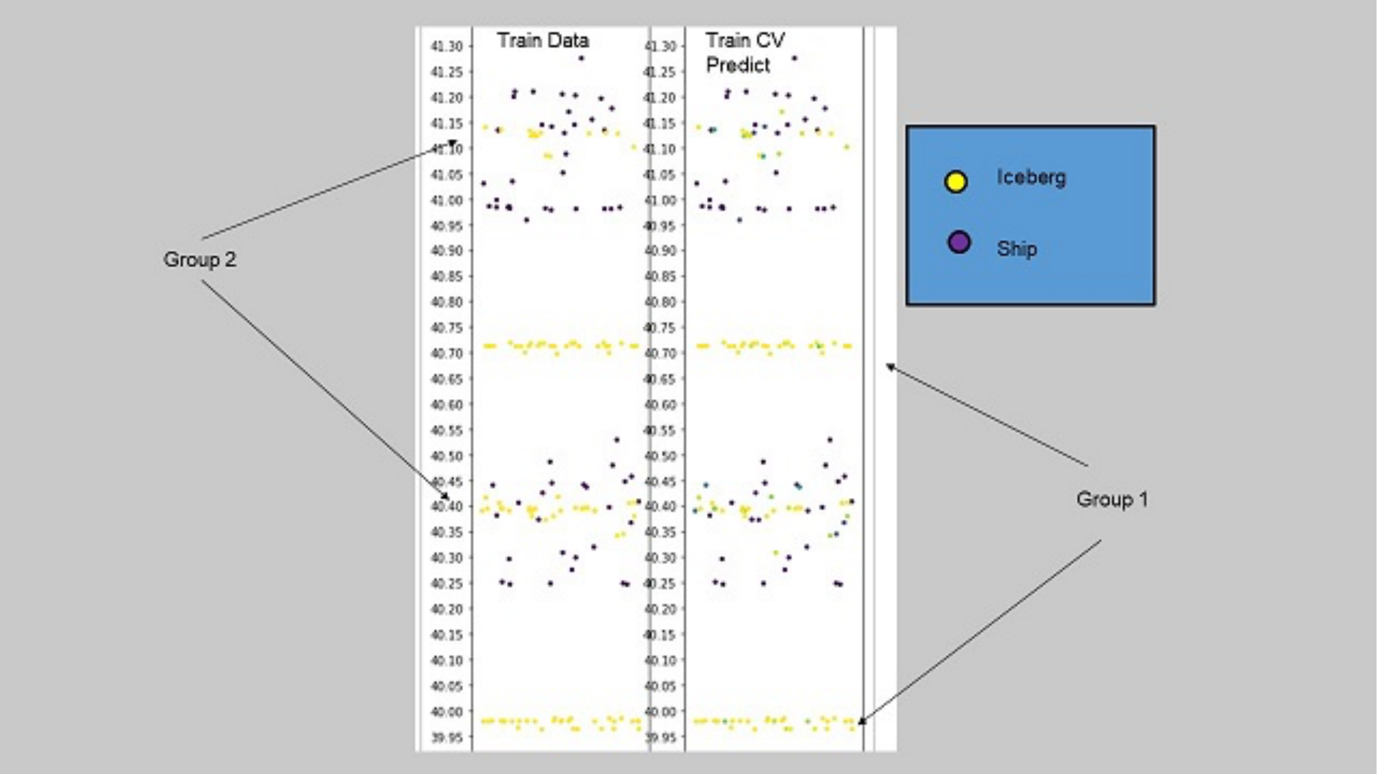
- Total parameters: 203,137
- 5 channels: 2 bands and 3 leaky features (mean, count, mean over area)
- Global Average Pooling on top
- No preprocessing for data
- No augmentation
- Trained 200 models, median prediction of top 100 models

Public 0.1065
Private **0.0946**

Other solutions

First place (**Weimin Wang and David**), 0.0822:

1) Found groups
2) Different models for group 1 and group 2
3) Ensembling and stacking



Other solutions

Second place (**beluga**), 0.0855:

1) ”Hundreds of CNN with different random parameters” + augmentation + pseudo
2) Xgboost over group features and previous models

Averaging:

- 95% model average of the 100 best xgb models.
- 5% model average of the 100 best xgb model without using inc_angle

Other solutions

Third place (**Evgeny Nekrasov**), 0.0857 :

1) 7 NNs, 5 folds and 30 repeats – no angle information
2) Mixed NNs with XGBoost, 7 folds and 1000 repeats.
3) Spatial model using neighborhood mean target variable
4) Mixing model without spatial information with the spatial model
5) Retraining models with pseudo-labeling
6) Mixing again

Other solutions

Fourth place (**Kirill Zhdanovich, Andrii Sydorchuk**), 0.0868 :

1) 5 NN with incidence angle
2) Take NNs with best validation score
3) For each angle calculate mean prediction, median prediction, total number of samples in each group
4) Stacking KNN, LightGBM

Conclusions

- Do not use public kernels at least for the first time
- Do not stack public kernels
- Do not stack stacked public kernels
- Make EDA before training
- Try to connect samples with each other if possible
- Do not spend to much time on hyperparameter tuning
- Clip if the metric is logloss
- Hardware is not always the key

Thank you!