**Model Explanations**

*Local Validation*

5 common folds stratified by depth. Score on local validation had pretty solid correlation with the LeaderBoard (LB).

*1st* Stage Training

BES & Phalanx both developed single model based on training data:

*BES model*

*Input:* 101 -> resize to 192 -> pad to 224

*Encoder:* ResNeXt50 pretrained on ImageNet

*Decoder:* conv3x3 + BN, Upsampling, scSE

*Training overview:*

Optimizer: RMSprop. Batch size: 24

1. Loss: BCE+Dice. Reduce LR on plateau starting from 0.0001
2. Loss: Lovasz. Reduce LR on plateau starting from 0.00005
3. Loss: Lovasz. 4 snapshots with cosine annealing LR, 80 epochs each, LR starting from 0.0001

[*Phalanx*](https://www.kaggle.com/phalanx)*model*

It was ResNet34 (architecture is similar to resnet\_34\_pad\_128 described below) with input: 101 -> resize to 202 -> pad to 256

* 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*

Based on the ensemble from the 1st stage, they created a set of confident pseudolabels. The confidence was measured as percentage of confident pixel predictions (probability < 0.2 or probability > 0.8).

Then, again, we had 2 models:

1. BES ResNeXt50 was pretrained on confident pseudolabels; and 5 folds were trained on top of them. 0.871 (0.890 Private) of Accuracy
2. [phalanx](https://www.kaggle.com/phalanx) added 1580 pseudolabels to each of 5 folds and trained the model from scratch. 0.861 (0.883 Private) of Accuracy
3. Their ensemble scored 0.870 (0.891 Private) Accuracy

*3rd Stage Training*

We took all the pseudolabels from the 2nd stage ensemble, and [phalanx](https://www.kaggle.com/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](https://arxiv.org/abs/1805.10180) (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](https://arxiv.org/abs/1703.02719)

*Decoder:* [Global Attention Upsample](https://arxiv.org/abs/1805.10180) (implemented like senet -> like scSE, conv3x3 -> GCN) + deep supervision

*Loss:* BCE for classification and Lovasz for segmentation

*Training overview:*

Optimizer: SGD. Batch size: 32.

1. Pretrain on pseudolabels for 150 epochs (50 epochs per cycle with cosine annealing, LR 0.01 -> 0.001)
2. 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*

We were using pretty similar list of augmentations. My augmentations were based on the great [albumentations](https://github.com/albu/albumentations) library:

* 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)

Postprocessing

We developed postprocessing based on jigsaw mosaics. Here is an idea:

1. Find all vertical and half-vertical (bottom half of the image is vertical) images in train data
2. All test images below them in mosaics get the same mask
3. Only one test image above them get the same mask, and only if its depth in mosaic >= 3

Unfortunately, it gave huge boost on Public LB and no boost on Private:

0.876 -> 0.884 on Public LB and 0.896 -> 0.896 on Private LB

GPU ressources

* I used only a single 1060Ti

Frameworks

* BES was using Keras.
* [Phalanx](https://www.kaggle.com/phalanx) was using PyTorch