Severstal Steel Defect Detection

1~14등 솔루션

1. Overview

- (1) Overview

Overview

문제
- 철판(steel sheet)의 표면 결함을 찾고 4종류로 분류

평가
- Mean Dice Coefficient

제출
- Run-length encoding

데이터
- 12K 학습 이미지 (1600 x 256)
- 5k 평가 이미지

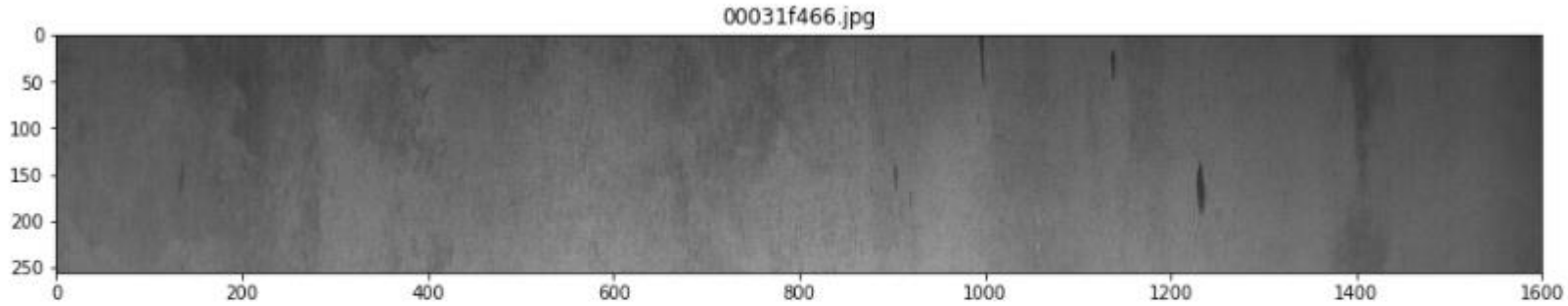
출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/

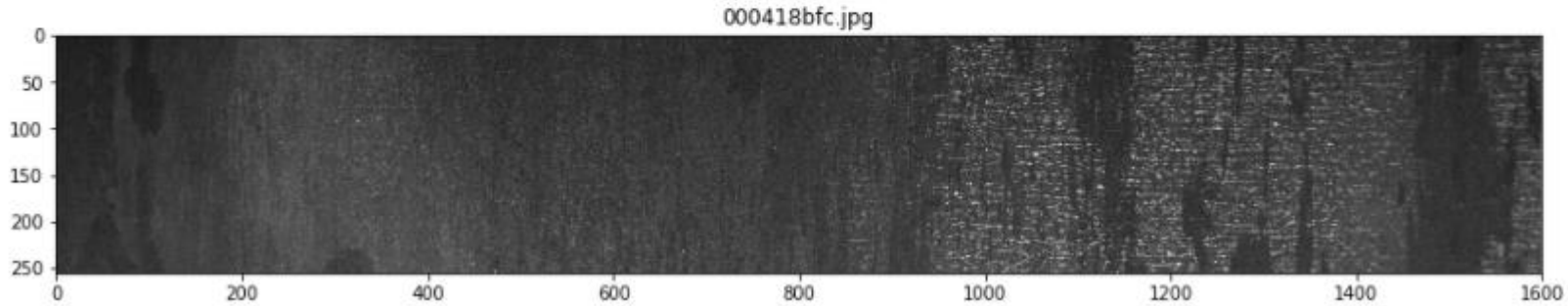
1 Overview

• (2) Data Exploration Analysis

Data Exploration Analysis

1. Images with no defect (결함이 없는 이미지)

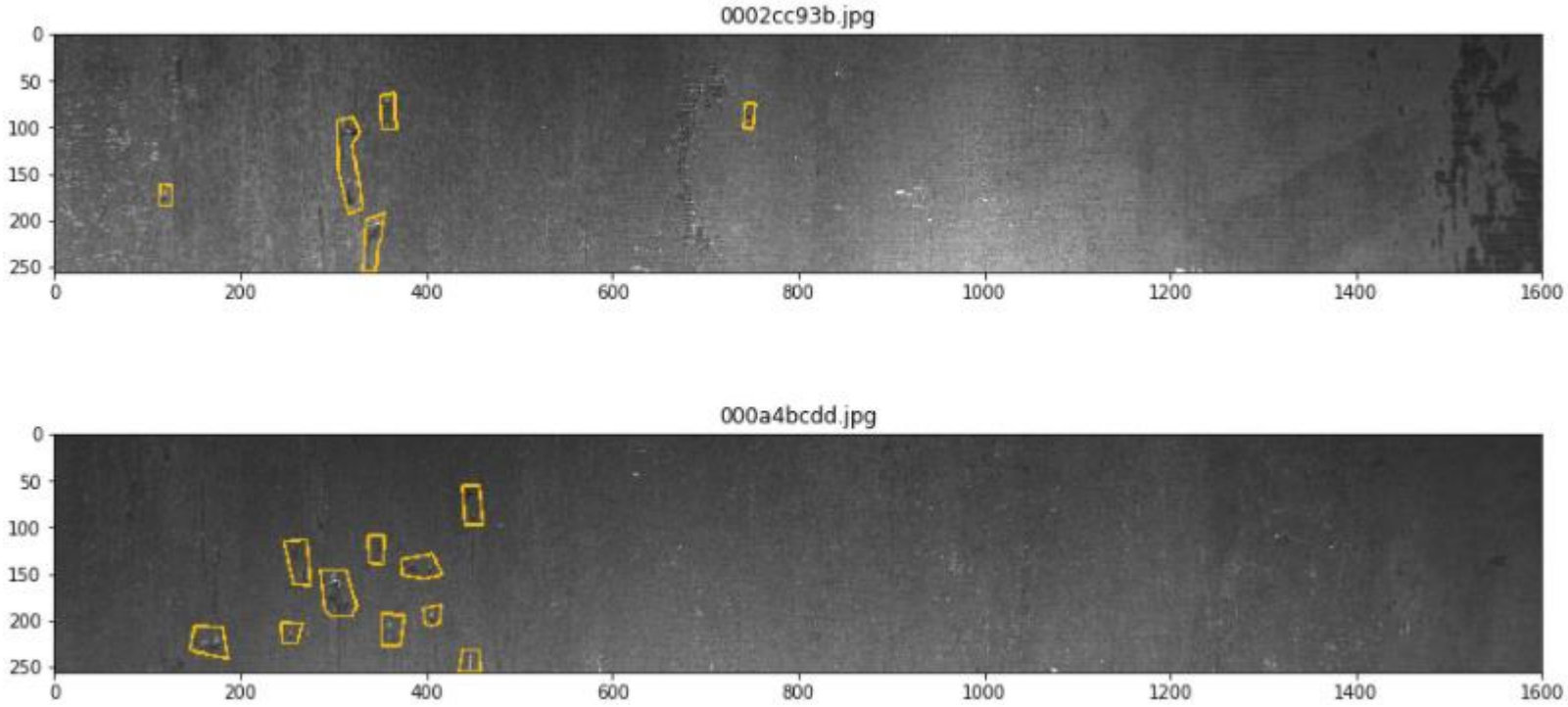




출처 : https://www.kaggle.com/go1dfish/clear-mask-visualization-and-simple-eda

Data Exploration Analysis

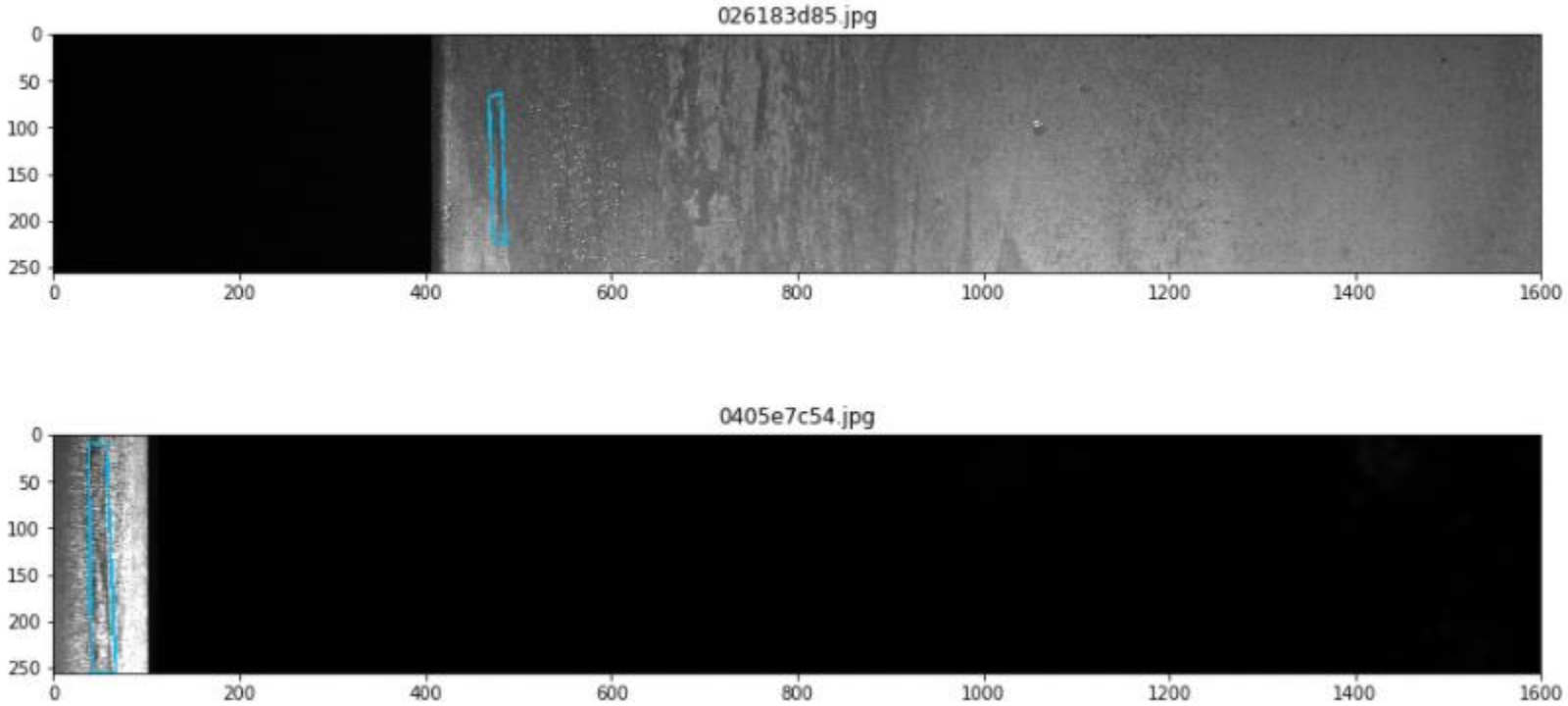
2. Images with 1 defect (1번 결함이 있는 이미지)



출처 : https://www.kaggle.com/go1dfish/clear-mask-visualization-and-simple-eda

Data Exploration Analysis

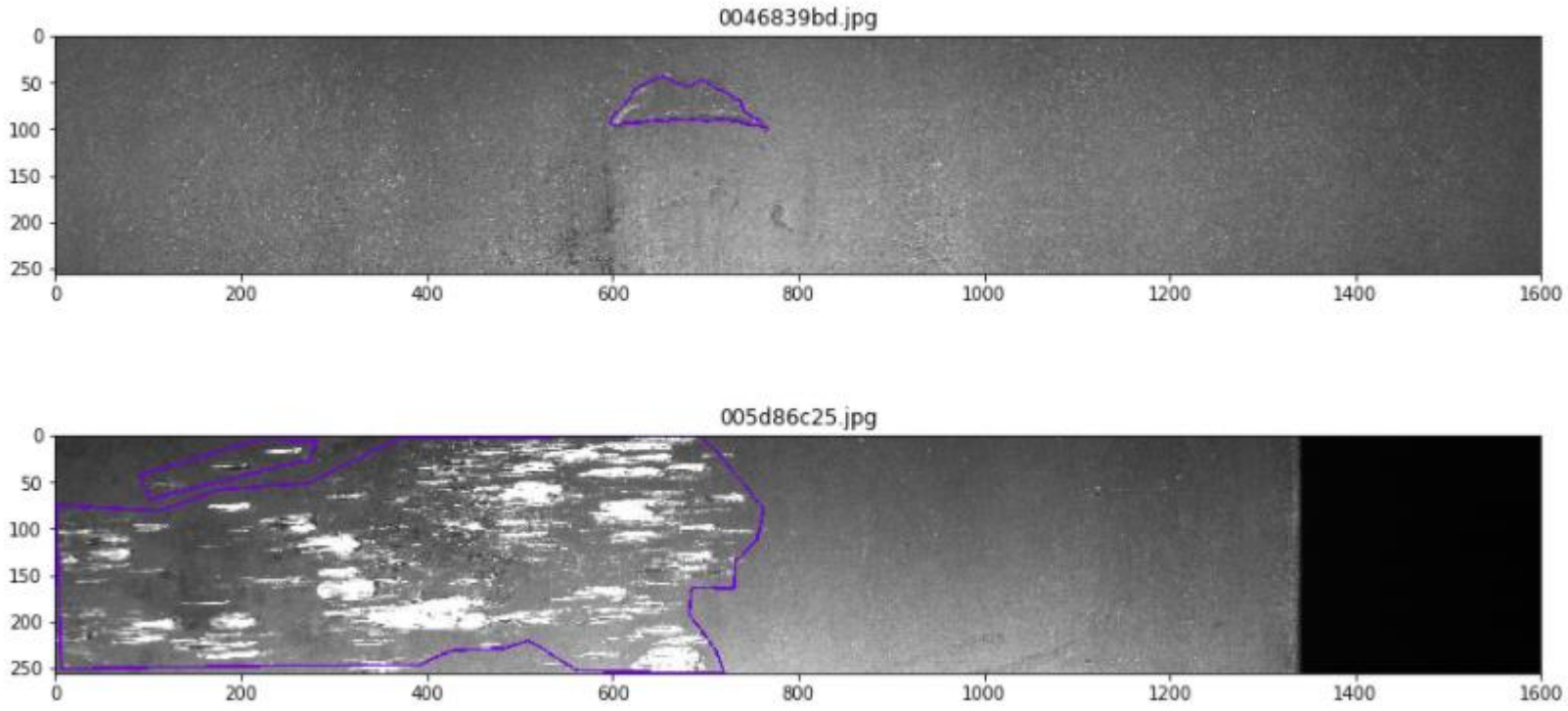
3. Images with 2 defect (2번 결함이 있는 이미지)



출처 : https://www.kaggle.com/go1dfish/clear-mask-visualization-and-simple-eda

Data Exploration Analysis

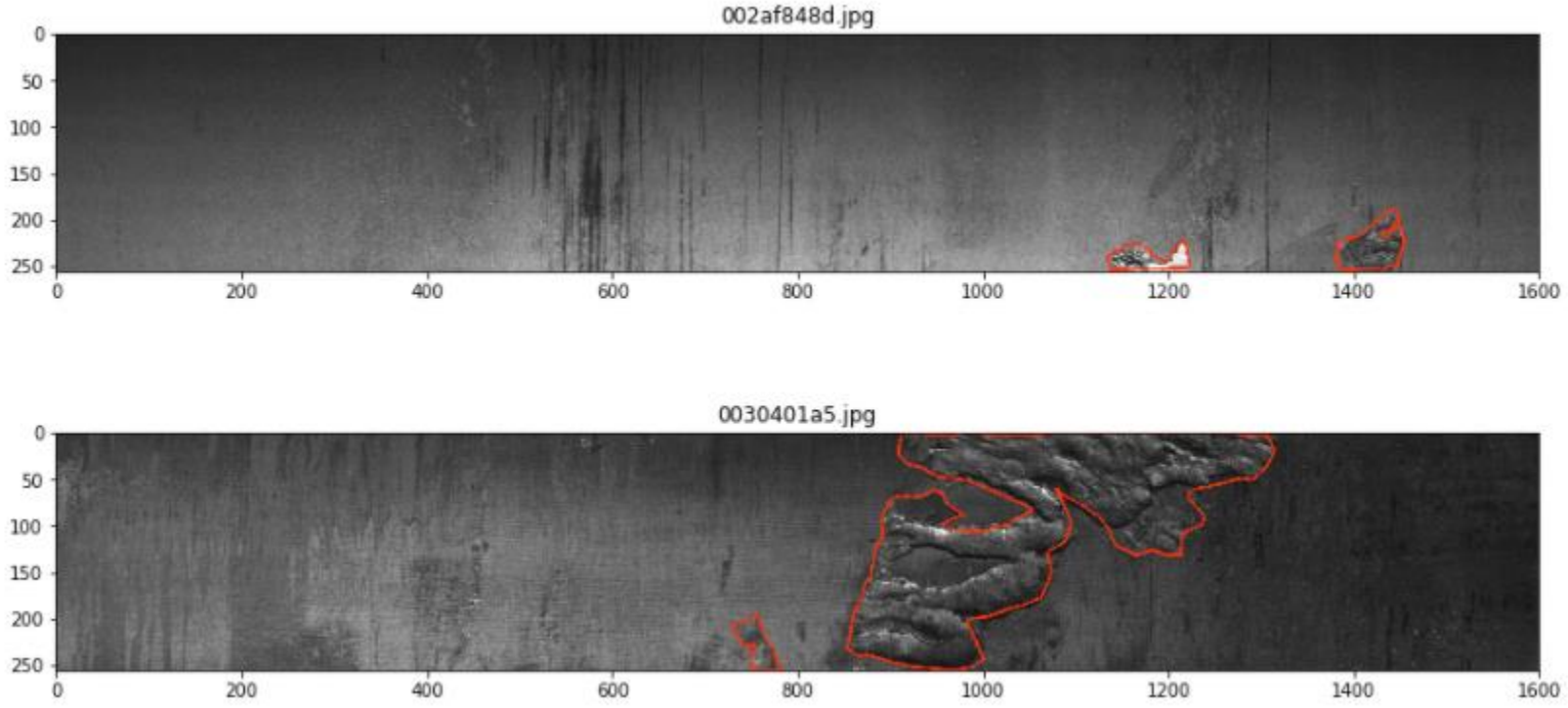
4. Images with 3 defect (3번 결함이 있는 이미지)



출처 : https://www.kaggle.com/go1dfish/clear-mask-visualization-and-simple-eda

Data Exploration Analysis

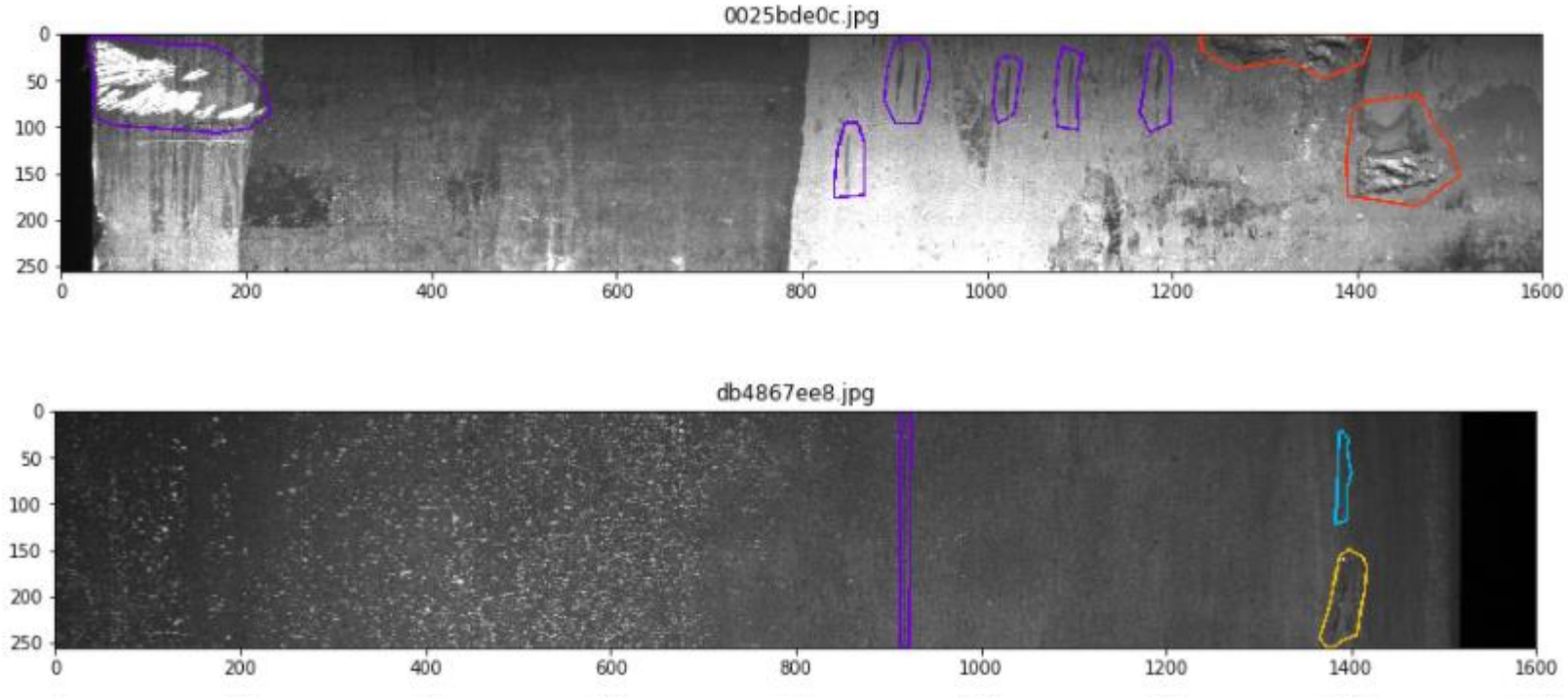
5. Images with 4 defect (4번 결함이 있는 이미지)



출처 : https://www.kaggle.com/go1dfish/clear-mask-visualization-and-simple-eda

Data Exploration Analysis

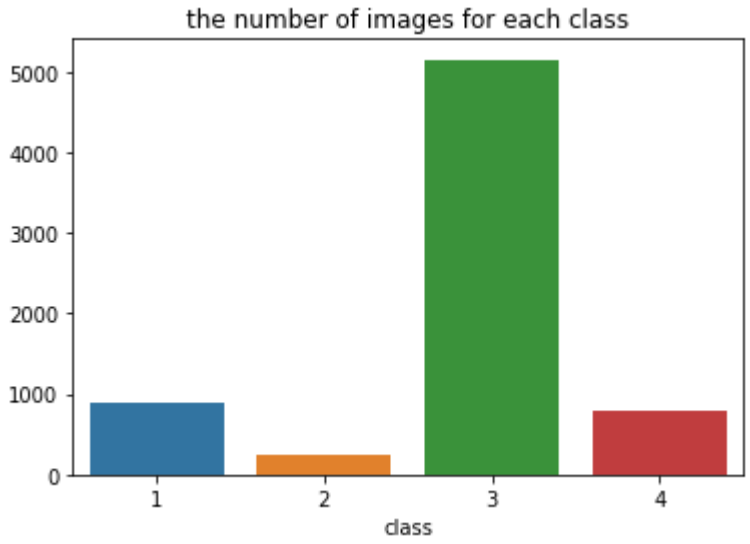
6. Images with multi defect (여러 개의 결함이 있는 이미지)

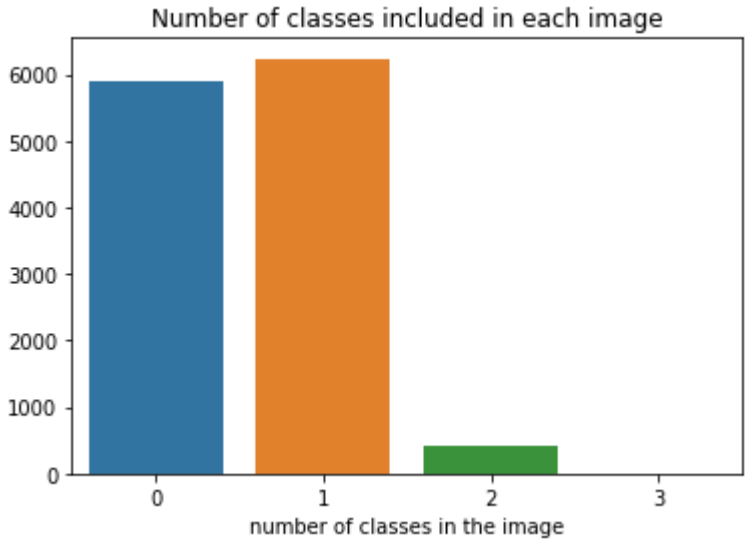


출처 : https://www.kaggle.com/go1dfish/clear-mask-visualization-and-simple-eda

Data Exploration Analysis

7. Others



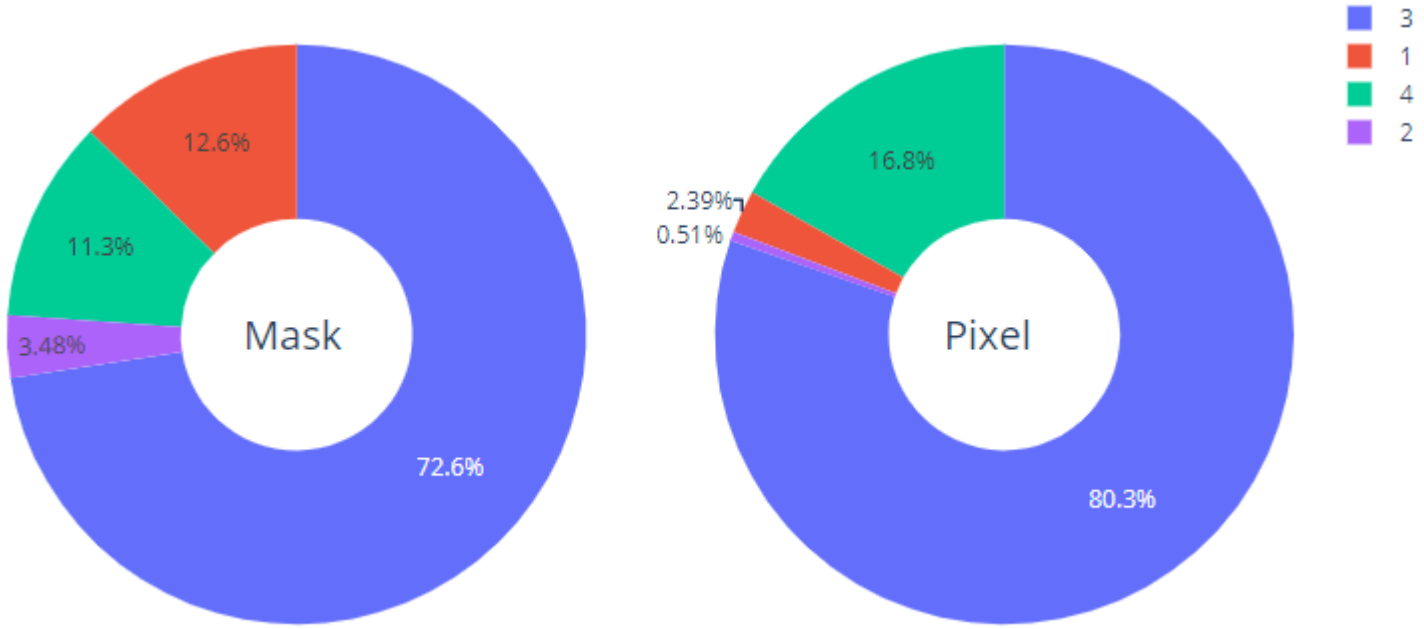


- 3번 클래스는 많이 나오는 반면 다른 클래스는 적은 편
- 대부분의 이미지가 클래스가 없고 1개의 클래스만 나온 이미지가 대부분

출처 : https://www.kaggle.com/go1dfish/clear-mask-visualization-and-simple-eda

Data Exploration Analysis

7. Others



- 1, 2번 클래스의 경우 Mask 대비 Pixel의 비율이 굉장히 적음
- 3번 클래스의 경우 Mask 대비 Pixel의 비율이 굉장히 큼

출처 : https://www.kaggle.com/ekhtiar/defect-area-segments-eda-with-plotly-fp-mining

Data Exploration Analysis

7. Others

```
for col in tqdm(range(0, len(train_df), 4)):
    name, mask = name_and_mask(col)
    if (mask.sum(axis=2) >= 2).any():
        show_mask_image(col)
```

```
100%|██████████| 12568/12568 [02:24<00:00, 87.19it/s]
```

- 하나의 픽셀에 대해서 2개 이상의 클래스가 중복된 경우는 없음

출처 : https://www.kaggle.com/go1dfish/clear-mask-visualization-and-simple-eda

2. Solution

(1) 1st Solution

Solution

1st solution

Network

- ResNet34
- EfficientNet b1

1) Classification - 결함 유무 분류

: public LB의 score에 많은 영향을 주지는 않았지만 50% 이상의 결함이 없는 이미지를 분류해낼 수 있기 때문에 이러한 프로세스 선택함. 랜덤 크롭된 데이터로 학습하였고, inference시에는 전체 이미지를 사용, accuracy를 올릴 수 있었음

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114254

2. Solution

(1) 1st Solution

Solution

1st solution

1568x224로 Crop해서 활용

Augmentations:

- Randomcrop, Hflip, Vflip, RandomBrightnessContrast (from albumentations) and a customized defect blackout

불량이 있는 부분을 cutout 적용

Batchsize

- 8 for efficientnet-b1, 16 for resnet34 (both accumulate gradients for 32 samples)

Optimizer : SGD

Model Ensemble: 3 x efficientnet-b1+1 x resnet34

TTA: None, Hflip, Vflip

Threshold: 0.6,0.6,0.6,0.6

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114254

2. Solution

(1) 1st Solution

Solution

1st solution

Network

- Unet (EfficientNet-b3)
- FPN (EfficientNet-b3)

2) Segmentation

Train data: 256x512 crop images

Augmentations: Hflip, Vflip, RandomBrightnessContrast (from albumentations)

Batchsize: 12 or 24 (both accumulate gradients for 24 samples)

Optimizer: Rectified Adam

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114254

2. Solution

(1) 1st Solution

Solution

1st solution

Loss : BCE (with pos_weight = (2.0,2.0,1.0,1.5))

- 0.75BCE+0.25DICE (with pos_weight = (2.0,2.0,1.0,1.5))

Model Ensemble :
1 x Unet(BCE loss) + 3 x FPN(first trained with BCE loss then finetuned with BCEDice loss) +2 x FPN(BCEloss)+ 3 x Unet from mlcomp+catalyst infer

TTA : None, Hflip, Vflip

Label Thresholds: 0.7, 0.7, 0.6, 0.6

Pixel Thresholds: 0.55,0.55,0.55,0.55

Postprocessing :
Remove whole mask if total pixel < threshold (600,600,900,2000) + remove small components with size <150

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114254

2. Solution

(1) 1st Solution

Solution

1st solution

Pesudo Label

We did 2 rounds of pseudo labels in this competition.

- The first round is generated from a submission with 0.916 public LB, maybe it is too early? The second round was done several days before the end of this competition, generated from a submission with 0.91985 public LB. With pseudo label and public models, we finally improved from 0.91985 to 0.92124 on public LB and from 0.90663 to 0.90883 on private LB.

Classifier와 Segmentation가 똑같이 분류하였고, Classifier의 분류 score가 0.95 이상, 0.05인 데이터만 사용함. 이 방법을 통해 Train set에 1135개의 이미지 추가하였음.

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114254

2. Solution
(2) 3rd Solution

Solution

3rd solution

Basic Model : Unet, Feature Pyramid Network (FPN)

Encoder : efficientnet-b3, efficientnet-b4, efficientnet-b5, se-resnext50

Loss : Focal Loss

Optimizer : Adam, init lr = 0.0005

Learning Rate Scheduler : ReduceLROnPlateau (factor=0.5, patience=3,cooldown=3, min_lr=1e-8)

Image Size : 256x800 for training, 256x1600 for inference

Image Augmentation : horizontal flip, vertical flip

Sampler : Weighted Sampler

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/117377

2. Solution

(2) 3rd Solution

Solution

3rd solution

Ensemble Model :
I simply average 9 model output probability to achieve the final mask probability without TTA

- FPN + efficientnet-b5 + concatenation of feature maps
- FPN + efficientnet-b4
- Unet + efficientnet-b4 , add pseudo labeling data in training data
- Unet + efficientnet-b4, training with heavy image augmentation
- Unet + efficientnet-b4 +SCSE layer
- Unet + efficientnet-b4 +SCSE layer, add pseudo labeling data in training data
- Unet + efficientnet-b4 + Mish layer
- Unet + efficientnet-b3
- Unet + se-resnext50

Threshold
- Label Thresholds: 0.7, 0.7, 0.6, 0.6
- Pixel Thresholds: 0.4, 0.4, 0.4, 0.4

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/117377

2. Solution

(3) 4th Solution

Solution

4th solution

Network

- DenseNet201
- EfficientNet b5
- Resnet 34
- Some of them with FPN decoders, some with UNet

Training two-headed NN for segmentation and classification.

Combine heads at inference time as with soft gating (mask.sigmoid() * classifier.sigmoid())

Focal loss / BCE + Focal loss

Training with grayscale instead of gray-RGB

FP16 with usage of Catalyst and Apex

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114716

2. Solution

(3) 4th Solution

Solution

4th solution

Our best (and final) ensemble consisted of 9 models with densenet201, efficientnetb5, resnet34, seresnext50 encoders, some of them with FPN decoders, and some with UNet.

We added 3-flip TTA and averaged logits of the models, and soft gating applied

Others

- Postprocessing

: 0.55를 기준으로 mask binarize하였고, 256 픽셀 이하로 차지하는 마스크는 제거함

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114716

2. Solution

(3) 4th Solution

Solution

4th solution

[Tried But Not Work]

- Adding hard negative mining by **resampling** dataset every epoch. **Weights** were chosen as the inverse of the Dice score per image. Absolutely no difference;
- Adding **ArcFace** to embeddings. It should help to distinguish classes better. One more nope;
- Training **multi-stage network** (inspired by pose detectors). It should help mimic bad markings. Performed worse than single-stage;
- Adding **label smoothing**. Nope again.
- Adding **mixup** and **Poisson blending** to increase the number of images with defects;
- Trainig on **double-sized crops**: 256x1600 -> crop(256x512) -> resize(512x1024). Nope;
- Adding result of **anisotropic segmentation** to the input of the neural network, no gain;
- Training **HRNetV2** with full resolution. 22 hours with 8xV100 and no better than ResNet34.

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114716

2. Solution

(4) 7th Solution

Solution

7th solution

Network

- Unet (efficientnet-b7, efficientnet-b4, efficientnet-b0)

Loss

- Cross entropy loss

Others

- Mix up and label smoothing

  : Hard target (One hot representation)을 Soft target으로 바꾸는 방식.

  ex) [0,0,1,0] → [0.025, 0.025, 0.925, 0.025]

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114215

2. Solution

(4) 7th Solution

Solution

7th solution

[Worked]

- AdamW with the Noam scheduler
- gitlab pipeline to manage training experiments
- mixup and label smoothing
- fine tuning on the full resolution
- random sampler
- cross entropy loss

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114215

2. Solution

(4) 7th Solution

Solution

7th solution

[Not Worked]

- SGD
- SWA worked really good for the salt competition, but for this competition it didn't work at all
- pseudo labeling (trained on last 2, 3 days)
- training a classifier
- balanced sampler

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114215

2. Solution

(5) 9th Solution

Solution

9th solution

Network

- Resnet34
- FPN EfficientNet b2

Classification - 결함 유무 분류

: 특이하게 4개(= class 개수)의 binary classifier 사용함. 결함이 없다고 판단된 이미지에 대해 Segmentation 단계를 생략하기 위함이었으나 Class imbalance 해결에도 사용됨.

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114297

2. Solution

(5) 9th Solution

Solution

9th solution

Loss

- BCE → Lovasz hinge

: BCE가 초반에는 잘 작동하나, 후반에선 loss가 줄어듦에도 metric에 큰 개선이 없었음. 따라서 초반에는 BCE를 사용해 빠르게 학습하고, 이후 Lovasz hinge loss를 사용해 학습할 수 있게함. (Segmentation)

Others

- Class Inbalance

: Defect가 있는 이미지의 숫자가 매우 적었음. 따라서 전체 데이터가 아닌 1) Mask가 있는 모든 이미지와 Classifier에서 defect가 있을 것으로 판단된(threshold 0.01) 일부 빈 이미지만을 이용해 Segmentation 모델을 학습함.

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114297

2. Solution

(6) 10th Solution

Solution

10th solution

1) Separate models for different defects (since the number of classes is small - it is feasible)

+ can use completely different architectures/pipelines/samplers etc

+ can combine trained models for different defects instead of retraining a single model

+ encourages to decompose the problem (4-defect detection) into smaller subproblems (4 single-defect detections) – it allows to focus on perfecting them separately, without the fear of spoiling results of a different subproblem

+ easier to reuse code for binary segmentation

+ no need to balance defect class losses

- wastes valuable submission time → smaller size/number of models can be ensembled

- more training time

- no potential for synergy between detectors of different defect types

위의 과정은 greedy하게 찾음

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114465

2. Solution

(6) 10th Solution

Solution

10th solution

2) Not predicting defect2

+ less models → less training / research / inference time

+ trying to predict defect2 was always decreasing the pub LB for me, so not predicting it → higher score

- if private test set contains more instances of images with defect2 – score would degrade

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114465

2. Solution

(6) 10th Solution

Solution

10th solution

3) Training on full size images

+ model has more context → easier to make a correct decision (avoid a false positive/negative)

+ no need for an extra fine-tuning stage on full images / less domain mismatch if no fine-tuning is done

- more time and GPU RAM is required → less experiments, smaller batch size

- less “unique” samples seen by model → more potential overfitting

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114465

2. Solution

(6) 10th Solution

Solution

10th solution

[FPN 모델에 대한 비교]

- FPN : Vanila FPN
- FPNB: 인코더의 최상위 계층 4개가 아니라 최하위 계층 4개를 사용합니다. 직관은 결함이 로컬 아티팩트인 경우 더 강한 의미 정보를 사용할 필요가 없으며, 대신 고해상도 레이어를 사용하면 작은 결함을 더 잘 찾을 수 있다는 것이었습니다. 내 실험에서 128x800 해상도에서 FPNA를 능가했지만, 최대 해상도로는 저조한 성능을 보였다.

- FPNC: (5)레벨의 인코더를 모두 사용합니다. 이렇게 하면 다양한 스케일 레벨을 올바르게 처리할 수 있습니다. 불행히도 GPU RAM 사용 증가와 교육 시간은 테스트의 성능 향상으로 이어지지 않았습니다.

- FPND: 인코더 레이어의 출력을 합하는 대신, 인코더 레이어의 출력에 있는 채널 수를 줄여 최종 채널 수를 동일하게 유지하니다. 이는 TGS 경쟁에서 잘 작동한 하이퍼 칼럼과 유사합니다. 이 버전은 다른 모델들에 비해 유의미하게 큰 스케일 변동을 가지고 있었기 때문인지 가장 좋은 성능을 보였습니다.

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114465

2. Solution
(6) 10th Solution

Solution

10th solution

[Augmentation]

a) only horizontal flip
b) RandAugment - up to 2 augmentations chose randomly from the list below (albumentation names):

- HorizontalFlip
- VerticalFlip
- ISONoise
- IAAAdditiveGaussianNoise
- CoarseDropout
- RandomBrightness
- RandomGamma
- IAASharpen
- Blur
- MotionBlur
- RandomContrast

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114465

2. Solution

(7) 12th Solution

Solution

12th solution

Stage 1. Multilabel classification

: On the Stage 1 there was a multilabel classifier to detect images with at least one type of defect.

Model:

- Best 3 of 5 folds Senet154 trained on resized images (128x800) with simple BCE loss

Augmentations:

- Normalization
- Resize
- h-flip, v-flip
- no TTA

After training we have found thresholds for binarization with maximizing f1-score for the given class, so each class have its own threshold. It allowed us to exclude almost half of the images and speed up inference.

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114309

2. Solution

(7) 12th Solution

Solution

12th solution

Stage 2. Multilabel segmentation

: On the Stage 2 there was some overfit magic. There we have mean ensemble of the multilabel segmentation nets:

- 4 folds of PSPNet on se_resnext101_32x4d (Awesome Qubvel implementation), trained on full images with BCE + Jaccard loss;
- Custom FPN on senet154 trained on crops (256x256) with simple BCE loss
- Custom PSPNet on senet154 trained on crops (256x256) with simple BCE loss

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114309

2. Solution

(7) 12th Solution

Solution

12th solution

[Training]

- Pretrain on crops (256x256), batch size 32, BCE + Jaccard
- Fine-tune on full size with 0.1 * lr, batch size 4 with gradients accumulation up to 20 images

Augmentations:

- Normalization
- CropNonEmptyMaskIfExists: crop image with defect area if one exists, else make random crop (we have contributed this transformation to Albumentations during competition)
- h-flip, v-flip
- only h-flip on TTA

[Trick]

There was a trick in the custom FPN and PSPNet training. First of all, we have trained a multilabel Senet154 classifier on the crops. After that we use this classifier as a backbone for the segmentators. It speed up training significantly with the same quality.

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114309

2. Solution

(7) 12th Solution

Solution

12th solution

First of all, we removed small objects and holes from the image. Then we assume that there is no objects if the sum of positive pixels is less than a minimum threshold (unique for the each class). It boosted our score on the public leaderboard, but could cause overfit. On this stage we have no thresholds optimization for the ensemble of models, since the default thresholds give us better result.

Second stage allows us to remove many of the False Positive images.

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114309

2. Solution

(7) 12th Solution

Solution

12th solution

Stage 3. Binary segmentation

: On the Stage 3 we use binary segmentation models, trained on non-empty masks for each class: 2 Unet(seresnext50), 2 FPN(seresnext50), 1 Unet(SeNet154)

[Training]

- loss: SoftDice + BinaryFocal(gamma=2.)
- optimizer: RAdam
- sheduler: reduce on plateau
- sampling: non-empty
- crop: 256x768

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114309

2. Solution

(7) 12th Solution

Solution

12th solution

[Augmentations]

- Normalization
- h-flip, v-flip
- rotate180
- random brightness/contrast
- jpeg compression
- random scale (limit=0.1)

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114309

2. Solution

(7) 12th Solution

Solution

12th solution

[Tips]

- Pseudolabels. For the classification step we use the most confident predicts from the previous submission. The confidence score for each image: np.mean(np.abs(np.subtract(prob_cls, 0.5))) where prob_cls - probability for each class on the image

- Use torch.jit to serialize your models (it helps a lot to transfer models to kaggle without pain)

- Trick that helps to improve models quality for about 0.3-0.5 % points for all models - best checkpoints weights average (weights!, not predictions). During traning 5 best checkpoints have been saved and then differnet combinations of them evaluated to find the best candidates to average

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114309

2. Solution
(8) 14th Solution

Solution

14th solution

[Augmentation]

```
def aug_medium(prob=1):
 return aug.Compose([
     aug.Flip(),
     aug.OneOf([
            aug.CLAHE(clip_limit=2, p=.5),
            aug.IAASharpen(p=.25),
            ], p=0.35),
     aug.OneOf([
         aug.RandomContrast(),
         aug.RandomGamma(),
         aug.RandomBrightness(),
         ], p=0.3),
     aug.OneOf([
         aug.ElasticTransform(alpha=120, sigma=120 * 0.05, alpha_affine=120 *
0.03),
         aug.GridDistortion(),
         aug.OpticalDistortion(distort_limit=2, shift_limit=0.5),
         ], p=0.3),
     aug.ShiftScaleRotate(rotate_limit=12),
     aug.OneOf([
            aug.GaussNoise(p=.35),
            SaltPepperNoise(level_limit=0.0002, p=.7),
            aug.ISONoise(p=.7),
            ], p=.5),
     aug.Cutout(num_holes=3, p=.25),
 ], p=prob)
```

Then I took 256x512 crops from the augmented image. Note that many images contain large black area, and there is always steel part shown on at least one end of an image. Simple random crops would make training inefficient, so here is what I did:

Pick a random crop

If the proportion of pixels with values < 10 is greater than 85% or the average pixel values of the crop < 15, pick the crop on either left or right end of image depending on which end satisfying the criterion. If nothing works, pick the crop on the right end

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114367

2. Solution

(8) 14th Solution

Solution

14th solution

[Balanced Sampling]

- To fight against imbalanced classes, I used balanced sampling. I don't like the approach that random picking defect type (or non-defect) under the same probability. It would make the concept of epoch arbitrary. I prefer deterministic approach, so in each epoch, I did something like 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 ... till every single non-defect (class 0) was sampled once.

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114367

2. Solution

(8) 14th Solution

Solution

14th solution

[Model]

- End-to-end classification + segmentation asymmetric U-Net on training set only.
- Backbone: se-resnext50 32x4d
- Decoder: CBAM attention and hyper-columns. I thought that the last decoder (and maybe the upsample in the second last decoder as well) was kind of redundant, so I removed it. In this way, the output stride of my model became 2. This saved a lot of my GPU memory
- Note that, unlike other classes, I fed all images to the defect 3 branch instead of defect 3 images only.

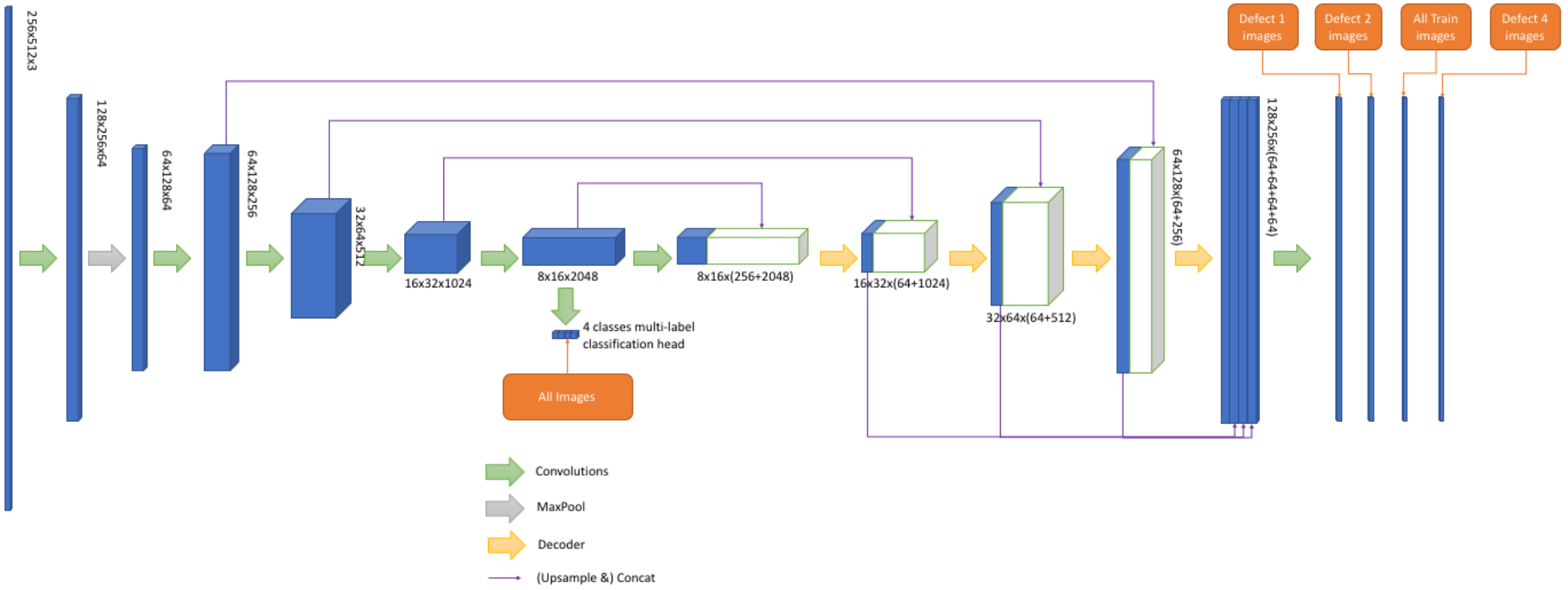
출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114367

2. Solution

(8) 14th Solution

Solution

14th solution



출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114367

2. Solution

(8) 14th Solution

Solution

14th solution

[Model 2]

Same as model 1, except using deep-stem and replacing every ReLU with Mish. Trained on training set and pseudo-labels on public test set

[Training]

- Loss: BCE loss for classification, symmetric Lovasz-Hinge for segmentation
- Optimizer: Adam
- Scheduler: Warmup+Flat+Cosine. 0.5 epoch linear warmup, 49.5 epochs flat at 1e-4, and 50 epochs cosine to 0
- Batch size: 6
- No fine-tuning on full images

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114367

2. Solution

(8) 14th Solution

Solution

14th solution

[Post-Processing]

Thresholds: since I was using hinge loss, I didn't tune segmentation thresholds

Classification: [0.55, 0.99, 0.25, 0.5]

Minimum pixels: [0, 0, 1200, 0]

These values were determined by one fold of model 1.

After the competition ended, I tried 0.5 for classification and 0 for minimum pixels for all four defects. It gave me 0.90625 on private test.

TTA: original, hflip and vflip

The masks are actually polygonal bounding boxes. It means that no holes in each part of defect. I randomly saw this page . I decided to gave Edge-based segmentation in that page a try. This post-processing consistently gave me 0.00005-0.0001 boost on local validation, public test and private test.

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114367

2. Solution
(8) 14th Solution

Solution

14th solution

[Pseudo-labelling]

Obtained pseudo-labels from 3 folds ensemble of model 1. Removed defect 1 with classification output < 0.85 and defect 3 with classification output < 0.75. I kept the amount of test images around 35% train images for each epoch.

[Ensemble]

2 folds of model 1 and 1 fold of model 2. Simple average before thresholding.

My best single fold of model 1 gave me 0.90310 on private test.

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114367

2. Solution

(8) 14th Solution

Solution

14th solution

[Things didn't work]

For classifier:

- Unsupervised Data Augmentation. I also tried to just use a subset of techniques in it such as RandAugment and TSA. They didn't improve either.
- Ring loss
- soft f-beta loss

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114367

2. Solution

(8) 14th Solution

Solution

14th solution

[Things didn't work]

For segmentation:

- Mini-Deeplab from Seamseg
- Surface Loss
- EMA
- Softmax on cropped images (with all kinds of multi-class losses / attention, including CrossEntropy, Focal Loss, Lovasz-Softmax, OCR

Things I didn't go deeper but worked at early stage:

Softmax on full or resized (128 x 800) images. HRNet + OCR with OHEM-softmax was the best. Didn't go deeper since sigmoid on cropped (256 x 512) images performed consistently better

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/114367

2. Solution
(9) Others

Solution

70th solution

Not average in ensemble, use temperature shaping

(https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/107716)

- Temperature sharpen

```
normal ensemble:
p = sum( p0,p1,p2 ...)/N

temperature sharpen:
t = 0.5
p = sum( p0**t,p1**t,p2**t ...)/N
```

Train on Crop, Predict on Full (https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/107716)

- Crop된 이미지에 대해 train 후, Full 이미지에 대해 예측할 모델로 transfer함.

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/107716

2. Solution
(9) Others

Solution

70th solution

Adversarial validation

: Train 데이터와 Test 데이터 병합 후, 데이터가 어떤 그룹에 속하는지 예측하는 모델을 만듦. 만약 두 데이터가 같은 분포에서 나온 데이터라면 어떤 그룹에 속하는지 구분하기가 어려워지기 때문에, Train 데이터와 Test 데이터가 얼마나 다른지 알 수 있음. 이 방법을 이용해 Test 데이터와 유사한 Train 데이터를 select, Validation 세트를 구성할 수 있음.

출처 : https://www.kaggle.com/c/severstal-steel-defect-detection/discussion/106834