AUTOMATED DETECTION OF STEEL DEFECTS VIA DEEP LEARNING

NEERAJ KATIYAR, MOHAMMAD FARZANULLAH, ARISH YASEEN, FARHAN BISHE

GROUP 15

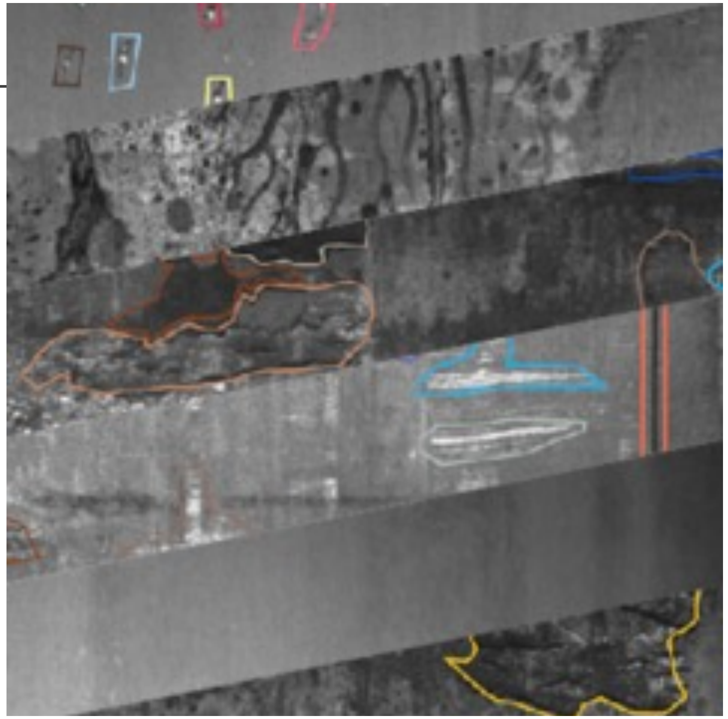
Introduction

- Steel is one of the most important building materials for modern architecture.

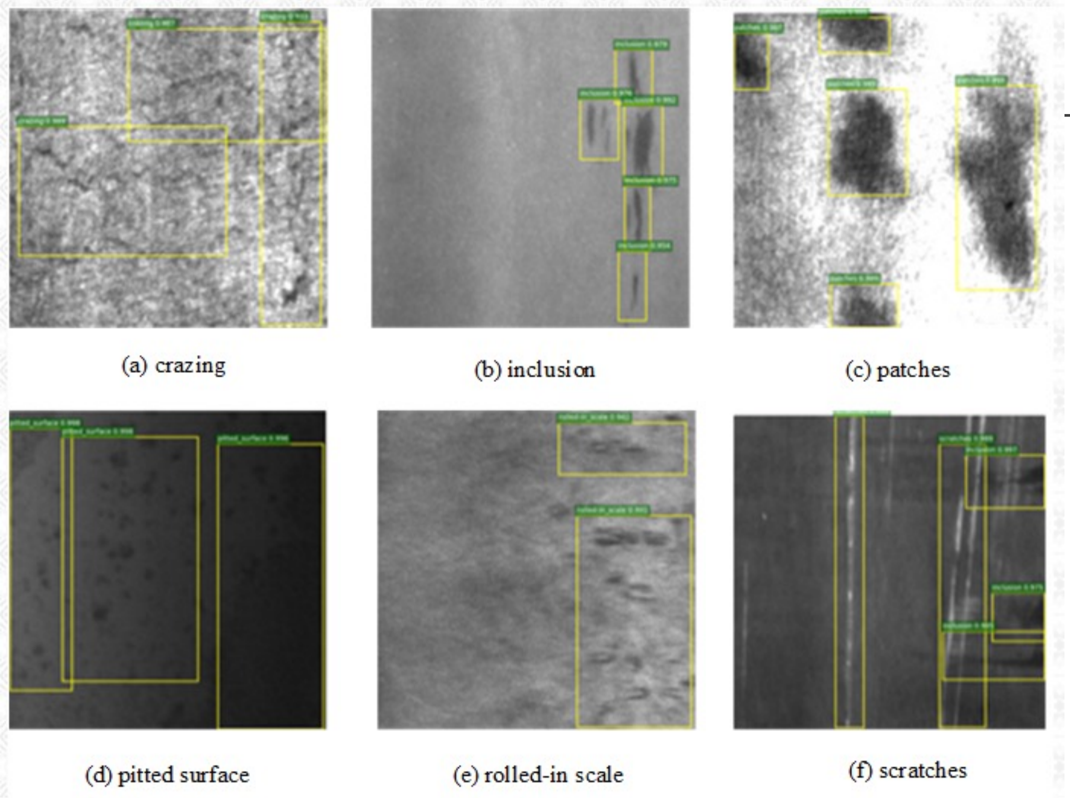
- Steel buildings are resistant to natural and manmade wear, which has made the steel material widely used around the world.

- However, despite its tough tensile strength, steel manufacturing is a very complex and delicate process, in particular, the production process of flat sheets is highly delicate.

Problem Statement



- As a result of this delicate process, the steel sheets are prone to several kind of defects,
- It is of high importance that these defects are identified, localized and classified swiftly and accurately.
- However, detecting the defects manually is inefficient and error-prone, sometimes dangerous when performing detection tasks on the fly.
- Many big Steel Producer are looking for an efficient, safer and accurate solution



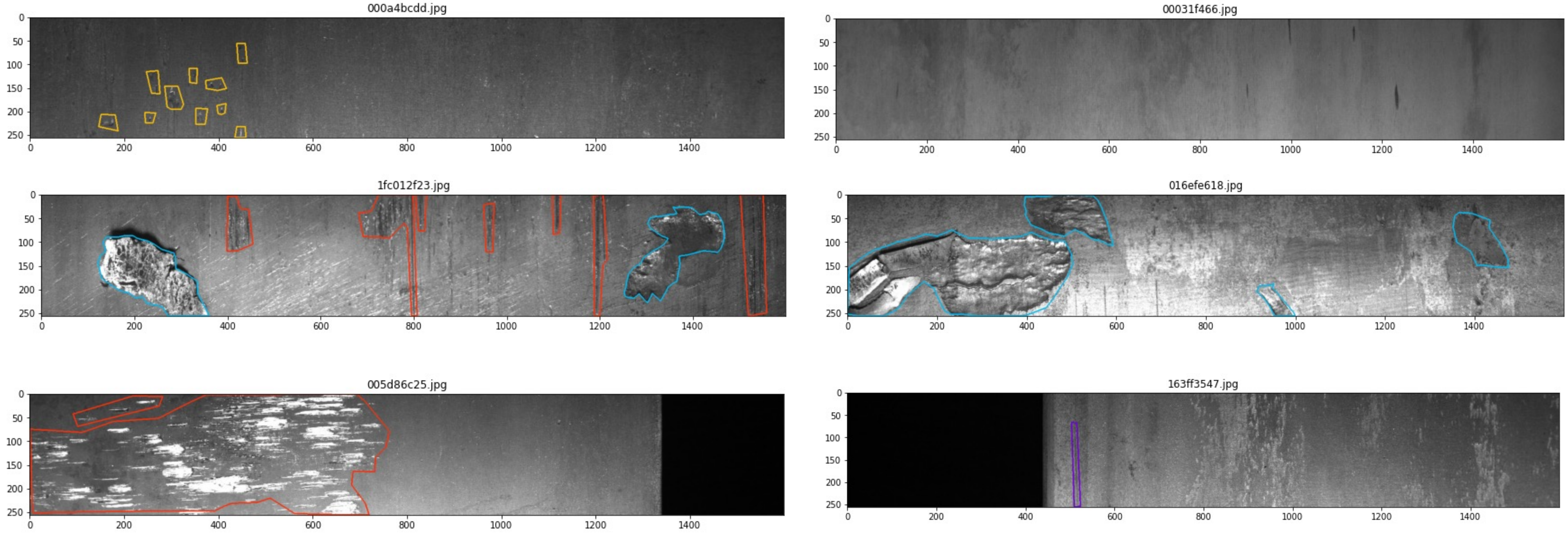
(a) crazing (b) inclusion (c) patches

(d) pitted surface (e) rolled-in scale (f) scratches

Goals

- The solution lies within the use of AI and Machine Learning
- Research has been carried out to identify and classify steel sheet defects through image processing models
- In this project, we design a model for classifying defects in steel sheet through image segmentation with aim to
  - Increasing the efficiency and enhancing automation in detecting steel defects to maintain high quality in steel production
  - Isolating the location and identifying the type of defect in steel sheet
  - Classifying and segmenting the steel defects in four distinct classes by implementing Deep Learning CNN approach.
  - Achieving accuracy above the baseline models and closer to the state-of-the-art algorithm in this field.

Kaggle Dataset (Severstal Steel Defect Detection)

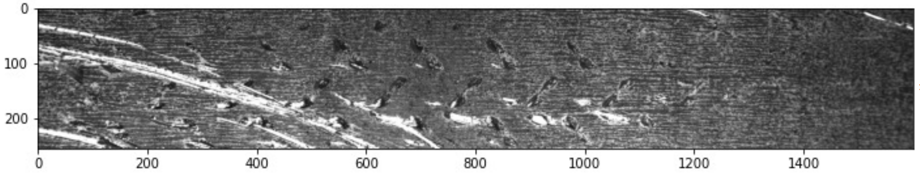


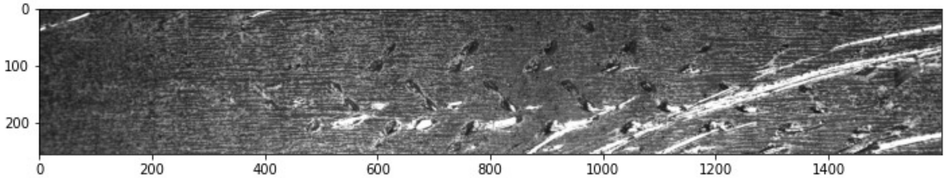
Method

Data Pre-processing

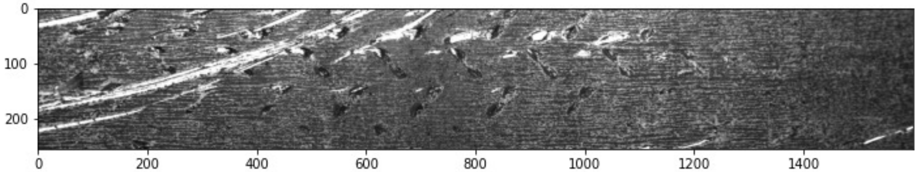
- Data Normalization (using ImageNet mean and std)
- Data Augmentation

Actual image

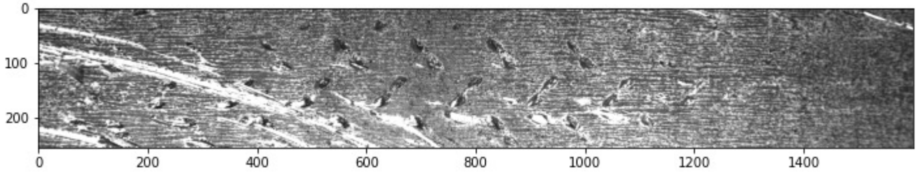




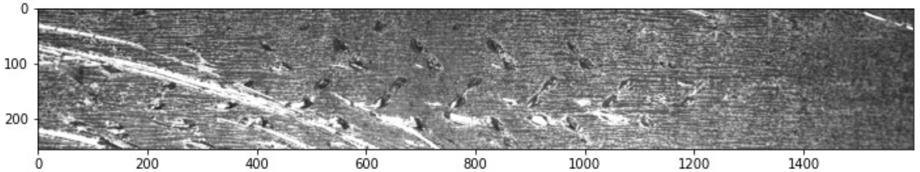
Horizontal flip (with probability 0.5)



Vertical flip (with probability 0.5)



20% change in brightness, contrast, hue



Gaussian Noise (with probability 0.5)

Training Methodology

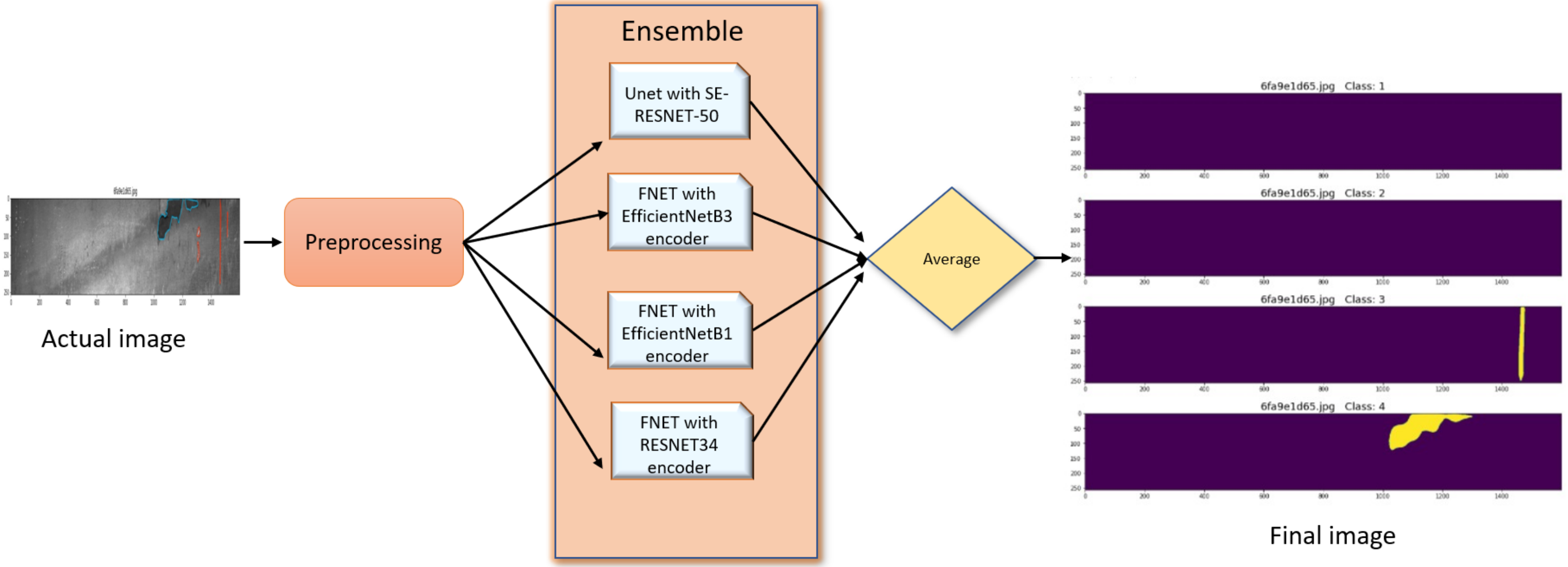
Architecture	Encoder	Loss	Training data	Optimizer	LR	Threshold	Minsize	Epochs	Public	Private
Unet	se_resnet_50	BCE	All	Adam	5.00E-04	0.50	3500	20	0.88918	0.89103
FPN	EfficientNet-b3	BCE	All	Adam	5.00E-04	0.50	3500	20	0.88777	0.8891
FPN	EfficientNet-b1	BCE	All	Adam	5.00E-04	0.50	3500	20	0.89629	0.88865
DeepLabV3Plus	ResNet34	BCE	All	Adam	5.00E-04	0.50	3500	20	0.88629	0.88851
FPN	ResNet34	BCE	All	Adam	5.00E-04	0.50	3500	20	0.8877	0.88825
Unet	se_resnet_50	BCE	All	RMSProp	5.00E-04	0.50	3500	20	0.88493	0.88797
Unet	se_resnext_32	BCE	All	Adam	5.00E-04	0.50	3500	16	0.88695	0.88711
Unet	ResNet34	BCE	All	Adam	5.00E-04	0.50	3500	20	0.88312	0.88701
Unet	EfficientNet-b2	BCE	All	Adam	5.00E-04	0.50	3500	15	0.88868	0.88663
Unet	ResNet34	BCEDice	All	Adam	5.00E-04	0.50	3500	20	0.88644	0.88605
UnetPlusPlus	ResNet34	BCE	All	Adam	5.00E-04	0.50	3500	20	0.88386	0.8854
Unet	DenseNet121	BCE	All	Adam	5.00E-04	0.50	3500	20	0.88494	0.88529
FPN	ResNet18	BCE	All	Adam	5.00E-04	0.50	3500	20	0.88508	0.88393
Unet_SingleClass	ResNet34	BCE	All	Adam	5.00E-04	0.50	3500	20	0.88073	0.88373
Unet	ResNet18	BCE	All	Adam	5.00E-04	0.50	3500	20	0.88689	0.8836
Unet	VGG11	BCE	All	Adam	5.00E-04	0.50	3500	20	0.88434	0.88259
Unet	EfficientNet-b1	BCE	All	Adam	5.00E-04	0.50	3500	20	0.87904	0.88211
Unet	EfficientNet-b3	BCE	All	Adam	5.00E-04	0.50	3500	15	0.88274	0.88168
FPN	se_resnet_50	BCE	All	Adam	5.00E-04	0.50	3500	20	0.87911	0.88159
PAN	ResNet34	BCE	All	Adam	5.00E-04	0.50	3500	20	0.88066	0.8801
LinkNet	ResNet34	BCE	All	Adam	5.00E-04	0.50	3500	20	0.88416	0.87972
FPN	EfficientNet-b2	BCE	All	Adam	5.00E-04	0.50	3500	20	0.88502	0.87894
Unet	ResNet18	BCE	Only +ve	Adam	5.00E-04	0.50	3500	20	0.86126	0.87855
MANet	ResNet34	BCE	All	Adam	5.00E-04	0.50	3500	20	0.87473	0.87764
Unet	Inceptionv4	BCE	All	Adam	5.00E-04	0.50	3500	15	0.88234	0.87509
Unet	ResNet18	Focal	All	Adam	5.00E-03	0.50	3500	20	0.85636	0.861
Unet	se_resnet_50	BCE	All	SGD	5.00E-04	0.50	3500	20	0.85674	0.8556

- Tried 21 different combinations of semantic segmentation architectures + encoders.
- We used transfer learning for all models. The encoders pretrained on ImageNet were used.
- Train-Validation split of 80-20%.
- Learning rate decreases by a factor of 10 when validation loss does not decrease for 3 epochs.
- Used the same postprocessing for all models (remove a defected mask if less than 3500 pixels).
- The output shape is 4x256x1600. Sigmoid is applied to find the probability of each defect on each pixel.

Ensemble Model

- Used the four models that gave the best results:
  - U-Net with se-resnet-50 encoder
  - FPN with EfficientNetB3 encoder
  - FPN with EfficientNetB1 encoder
  - FPN with ResNet34 encoder
- Predicted the original test image, horizontally flipped image and vertically flipped image for all 4 models and took the average of all predictions.
- Improved the post-processing step via extensive trial and error. The best result was obtained when defect masks of size less than [1400,1200,1800,1300] was removed for 4 classes. The pixel threshold was set as [0.5,0.6,0.45,0.4] for the 4 defect classes.

High level process flow



Within each model we make predictions for actual image, horizontal flipped and vertical flipped image

Results

Evaluation metric (Dice coefficient)

$$Dice = 2 \times \frac{|X \cap Y|}{|X| + |Y|}$$

Hyperparameter effects (Unet with SE-ResNet-50 encoder)

Optimizer	**Learning Rate**	**Weight Decay**	**Public**	**Private**
Adam	5.00E-04	0.00E+00	0.88918	0.89103
RMSProp	5.00E-04	0.00E+00	0.88493	0.88797
SGD	5.00E-04	0.00E+00	0.85674	0.8556
Adam	1.00E-03	0.00E+00	0.88741	0.88873
Adam	5.00E-04	1.00E-04	0.87667	0.87462

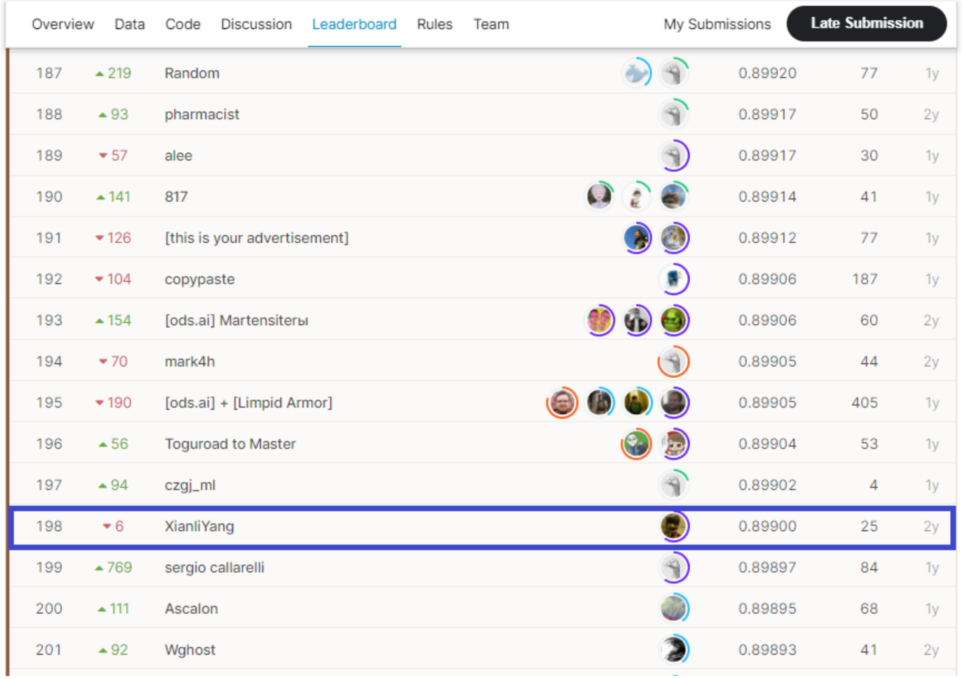
Winner/Base model comparison

Model	Private
Our ensemble model	0.89901
Winner model	0.90883
No defects prediction	0.8556

Results

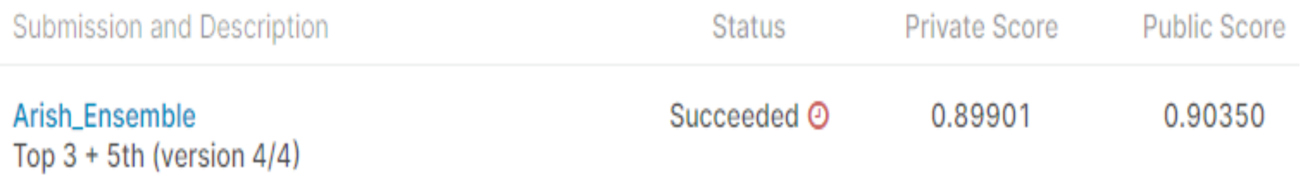
Kaggle performance

1. We stand at 198th position out of 2427 submissions in Kaggle competition. Which is under top 10% best performance.



Overview Data Code Discussion Leaderboard Rules Team My Submissions Late Submission

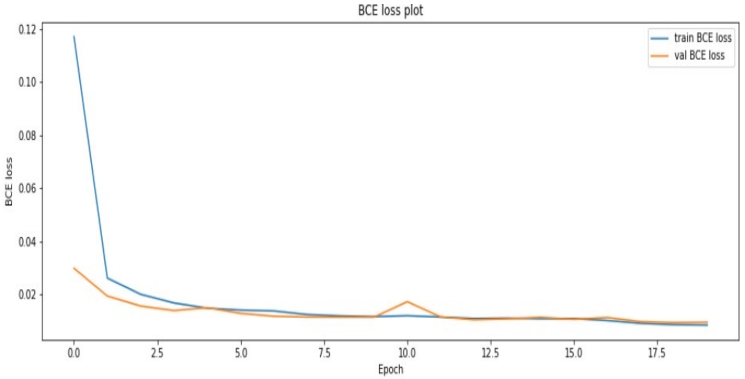
187	▲219	Random	0.89920	77	1y
188	▲93	pharmacist	0.89917	50	2y
189	▼57	alee	0.89917	30	1y
190	▲141	817	0.89914	41	1y
191	▼126	[this is your advertisement]	0.89912	77	1y
192	▼104	copypaste	0.89906	187	1y
193	▲154	[ods.ai] Martensiterы	0.89906	60	2y
194	▼70	mark4h	0.89905	44	2y
195	▼190	[ods.ai] + [Limpid Armor]	0.89905	405	1y
196	▲56	Toguroad to Master	0.89904	53	1y
197	▲94	czgj_ml	0.89902	4	1y
198	▼6	XianliYang	0.89900	25	2y
199	▲769	sergio callarelli	0.89897	84	1y
200	▲111	Ascalon	0.89895	68	1y
201	▲92	Wghost	0.89893	41	2y



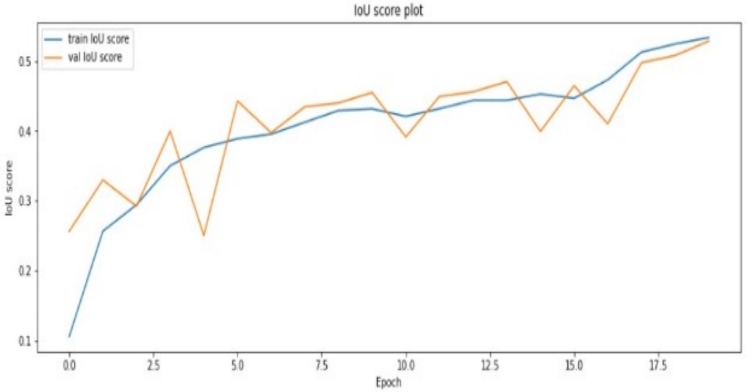
Submission and Description	Status	Private Score	Public Score
Arish_Ensemble Top 3 + 5th (version 4/4)	Succeeded	0.89901	0.90350

Results

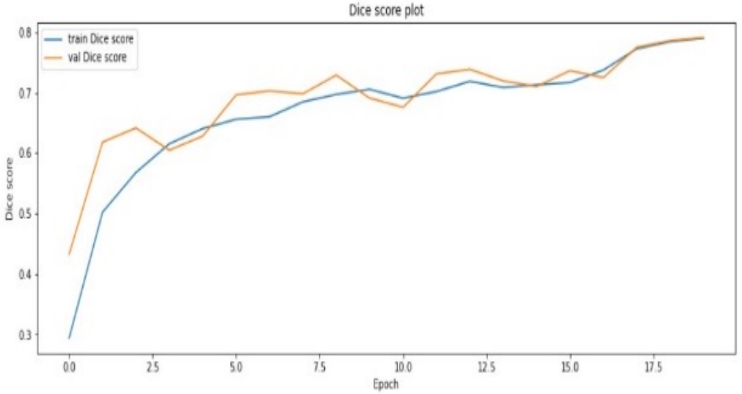
Performance curves



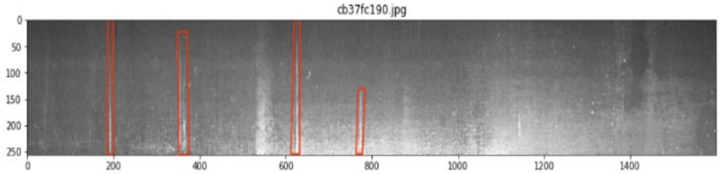
BCE loss over epochs

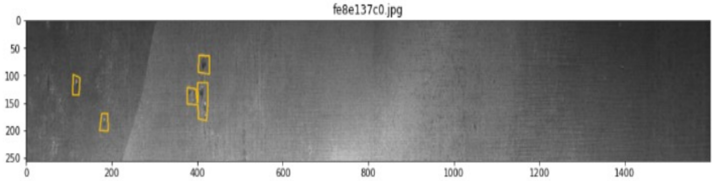


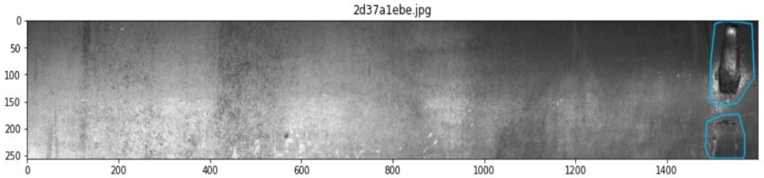
IoU score over epochs

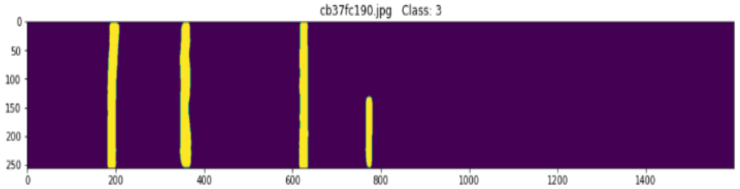


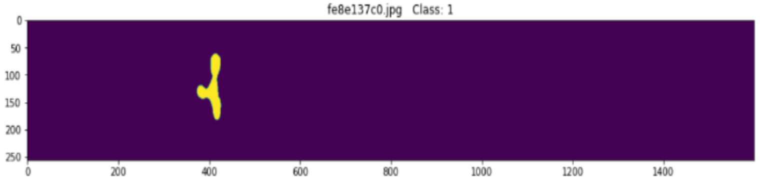
Dice score over epochs

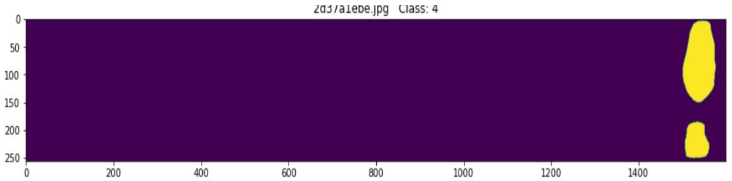












Conclusion & Future work

- An efficient model has been trained in terms of accuracy and meeting the desired goals
- Ensembling technique proved to be the better approach in achieving better accuracy, whereas individual we found SE-ResNet and FPN to be the best models
- The model was ranked within the top 10% of Kaggle competitors
- As a future work pseudo labeling can be investigated for improving the score
- Exploring faster techniques as the existing model requires considerable time for training and inference

Thank you!