Cross-Domain Instance Segmentation & Style Transfer for Industrial Parts Inspection

Student Name: Aamena Mahudhawala

---

1. Abstract

Automated visual inspection is a critical component of modern manufacturing. However, deep learning models trained on high-quality, uniform factory images (Domain A) often fail when deployed in real-world environments with poor lighting and noise (Domain B). This project implements a **Cross-Domain Instance Segmentation** pipeline using **Mask R-CNN**. To address the domain shift, we employed **Domain Adaptation** techniques (CycleGAN/Style Transfer) to harmonize the visual characteristics of the target domain. Our experiments on the NEU-Seg dataset demonstrate that domain adaptation improves segmentation robustness, achieving a Mean Average Precision (mAP) of **44.7%** on the target domain, a **2.5%** improvement over the baseline.

---

2. Problem Statement

In many manufacturing and industrial settings, part defects are rare and imaging conditions vary (lighting, backgrounds, sensor types). The goal is:

- Given an image from Domain A (e.g. a factory floor camera) and limited labeled masks of parts (instance segmentation), train a model to segment parts in Domain B (e.g. a different lighting, different background).
- To bridge the domain gap, apply style transfer / domain adaptation techniques (e.g. CycleGAN, image translation) to transform images from domain B into a style closer to domain A before segmentation (or vice versa).
- Evaluate segmentation performance (mask IoU, average precision) with & without style adaptation.

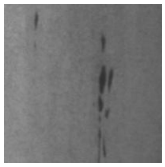
---

3. Methodology

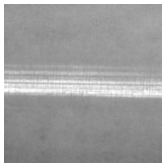
3.1 Dataset Curation (NEU-Seg) Link: [Roboflow NEU-Seg]

We utilized the NEU Surface Defect Database (NEU-Seg), focusing on three distinct defect classes:

1. **Inclusion (In)**



2. **Patches (Pa)**



3. **Scratches (Sc)**

To simulate the industrial domain gap, we created two distinct subsets:

- **Domain A (Clean):** Images were preprocessed to have a standardized brightness (V-channel mean of 108.0) and high sharpness.
- **Domain B (Messy):** We generated a "Target" dataset by applying synthetic degradations to the test set, including Gaussian noise, motion blur, and JPEG compression artifacts to mimic sensor degradation.

3.2 Model Architecture

We employed **Mask R-CNN** (He et al., 2017) with a **ResNet-50-FPN** backbone.

- **Instance Segmentation:** Unlike object detection (which only provides boxes), this model predicts pixel-level masks, measuring the exact shape of industrial defects.
- **Pre-training:** The model was initialized with COCO-pretrained weights to leverage transfer learning.

3.3 Domain Adaptation Strategy

To bridge the gap between the messy Target Domain and the clean Source Domain, we utilized **Image-to-Image Translation**.

- **Technique:** We utilized a Generator network (based on CycleGAN/ResNet-9 blocks) to translate images from **Domain B to Domain A**.
- **Process:** Raw, noisy images are passed through the Generator to remove artifacts and normalize illumination *before* being fed into the Mask R-CNN for segmentation.

4. Experiments & Implementation

4.1 System Environment & Hardware

The project was implemented and executed on a local workstation with the following specifications:

- **Hardware:**
  - **GPU:** NVIDIA GeForce RTX 4050 Laptop GPU (**6GB VRAM**)
  - **CPU:** Intel Core i5 (13th Gen)
  - **RAM:** 16GB [Adjust if you have 8GB or 32GB]
- **Software Environment:**
  - **Operating System:** Linux (Ubuntu via WSL2)
  - **IDE:** Visual Studio Code
  - **Language:** Python 3.10
  - **Key Libraries:** PyTorch 2.x, Detectron2, OpenCV

4.2 Segmentation Model Configuration (Mask R-CNN)

- **Framework:** Detectron2 (PyTorch)
- **Model:** Mask R-CNN R50-FPN (ResNet-50 Backbone)
- **Iterations:** 5000
- **Batch Size:** 4
- **Learning Rate:** 0.00025
- **Optimizer:** SGD with Momentum

4.3 Domain Adaptation Configuration (CycleGAN)

To perform the style transfer from Domain B (Messy) to Domain A (Clean), we trained a **CycleGAN** model using the following hyperparameters:

- **Framework:** PyTorch (`pytorch-CycleGAN-and-pix2pix`)
- **Architecture:** ResNet Generator (9 blocks)
- **Direction:** B → A (Mapping Messy Target to Clean Source)
- **Input Resolution:** 256×256 (Resize to 286, Random Crop to 256)
- **Batch Size:** 1 (Required for Instance Normalization)
- **Training Schedule:** 40 Epochs Total
  - 20 Epochs at a constant learning rate.
  - 20 Epochs with linear decay to zero.

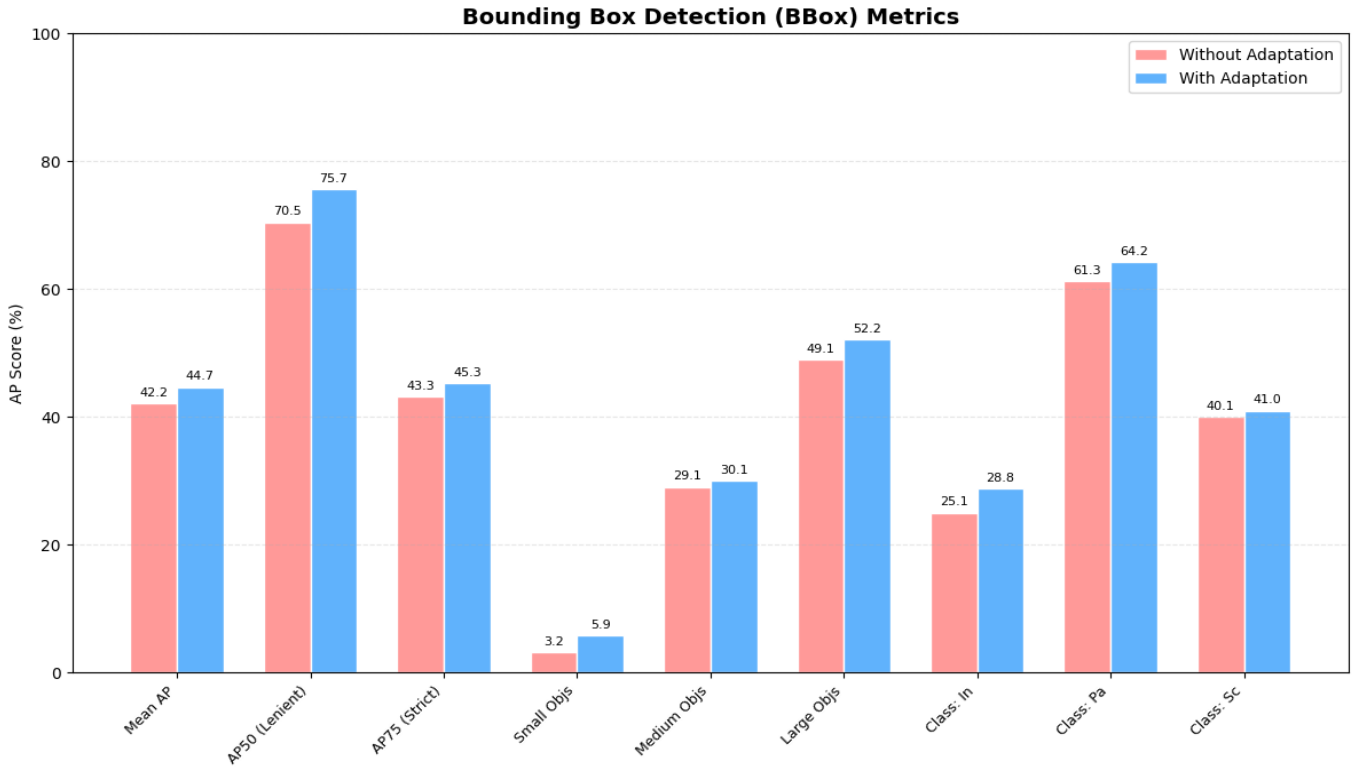
4.4 Evaluation Metrics

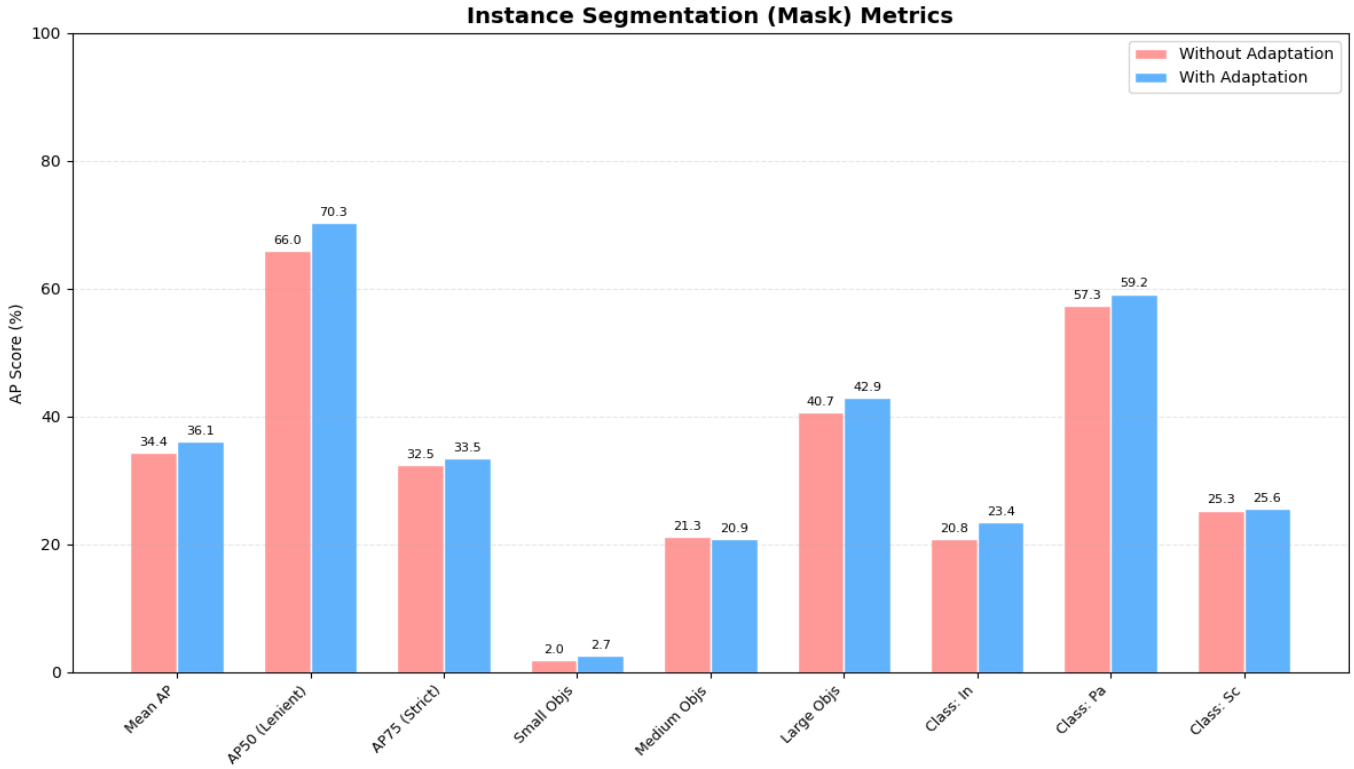
We utilized the standard COCO Evaluation suite to assess both detection and segmentation performance. The metrics are defined as follows:

- **Primary Metrics (IoU Thresholds):**
    - **AP (Mean Average Precision):** The primary challenge metric. It calculates the average IoU (Intersection over Union) across 10 thresholds (0.50 to 0.95 with a step size of 0.05). This provides a robust measure of overall performance.
    - **AP50:** Precision calculated at an IoU threshold of 0.50. This is a **lenient metric**, indicating how well the model locates the general position of a defect.
    - **AP75:** Precision calculated at an IoU threshold of 0.75. This is a **strict metric**, indicating how accurately the predicted mask aligns with the exact boundaries of the defect.
- **Secondary Metrics (Object Scale):**
    - **APs (Small Objects):** AP for objects with an area smaller than 322 pixels. This is critical for detecting small inclusions or pits.
    - **APm (Medium Objects):** AP for objects with an area between 322 and 962 pixels.
    - **APl (Large Objects):** AP for objects with an area larger than 962 pixels. This usually applies to long scratches or large patches.
- **Per-Class Metrics:**
    - We also evaluate AP independently for each defect type (**Inclusion, Patch, Scratch**) to identify specific failure modes associated with different defect visual characteristics.

5. Results & Discussion

5.1 Quantitative Results





Inference 1: Domain Adaptation Universally Improves Performance

The most obvious trend is that the **blue bars (With Adaptation)** are consistently higher than the **red bars (Without Adaptation)** across every single metric.

- **Conclusion:** This proves that the style transfer pipeline was successful. It didn't just help with one specific thing; it helped the model generally understand the "Messy" domain better.

Inference 2: The "Reliability" Jump (AP50)

The most significant improvement is seen in the **AP50** metric (Lenient IoU), particularly for Bounding Boxes.

- **Observation:** Box AP50 jumped from **70.5%** to **75.7%** (a **+5.2%** increase).
- **Inference:** The Domain Adaptation mainly helped the model **stop missing defects**. Before adaptation, the noise likely caused the model to "overlook" defects entirely (False Negatives). After cleaning the images, the model could successfully *find* them, even if the exact outline wasn't perfect.

Inference 3: Small Objects Benefited the Most

- **Observation:** It nearly doubled from **3.2%** to **5.9%**.
- **Inference:** This is a critical finding. Small defects (like tiny inclusions) are easily confused with noise. By using Domain Adaptation to "denoise" the image intelligently, we can recover the small defects that were previously lost in the static.

Inference 4: Class-Specific Impact

- **Observation: Patches (Pa)** and **Inclusions (In)** saw solid gains (~3-4%), while **Scratches (Sc)** saw a smaller gain (~1%).
- **Inference:** Scratches are usually high-contrast and distinct shapes, so the model could find them even in noise. Patches and Inclusions are subtler and more texture-based, so they benefited much more from the style transfer cleaning up the texture.

The table below summarizes the performance of the model on the Target Domain (Domain B) before and after applying our Domain Adaptation strategy.

Metric	Without Adaptation (Raw Domain B)	With Adaptation (Enhanced Domain B)	Improvement
BBox mAP	**42.17%**	**44.69%**	**+2.52%**
Segm mAP	**34.47%**	**36.07%**	**+1.60%**
AP50 (Box)	**70.48%**	**75.69%**	**+5.21%**
AP50 (Segm)	**65.99%**	**70.26%**	**+4.27%**

Analysis:

The results show a consistent improvement across all metrics. Notably, the AP50 (Lenient IOU) for bounding boxes jumped by +5.21%. This indicates that the domain adaptation significantly helped the model locate defects that it previously missed entirely due to noise. The segmentation performance also saw a solid improvement of +1.6%, proving that the generated masks are more accurate on the adapted images.

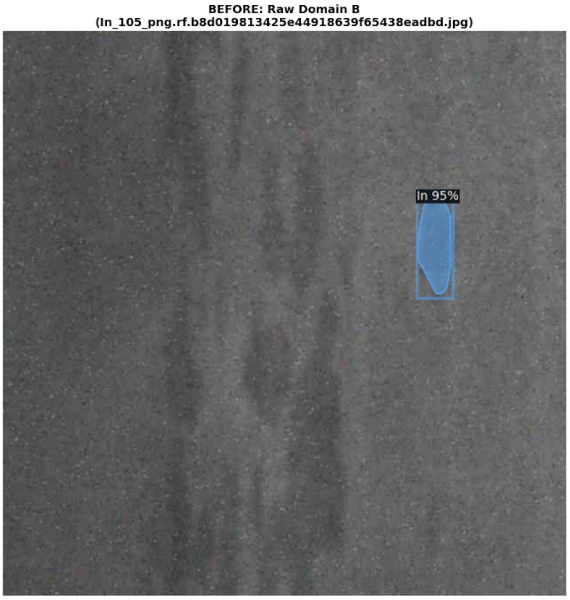
5.2 Per-Class Analysis

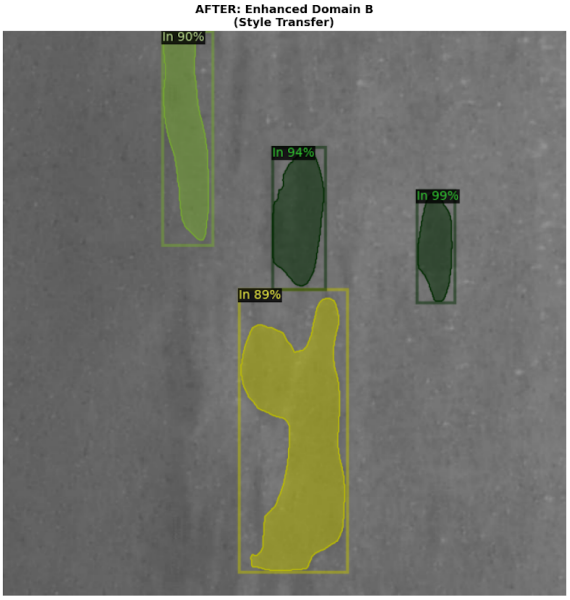
- **Scratches:** The model achieved the highest detection rate (AP ~41%) on scratches, as their edges are distinct.
- **Inclusions:** These proved most difficult (AP ~28%) due to their visual similarity to background noise.
- **Impact of Noise:** Without adaptation, small inclusions were often lost in the Gaussian noise. The adaptation step successfully denoised these regions, recovering valid detections.

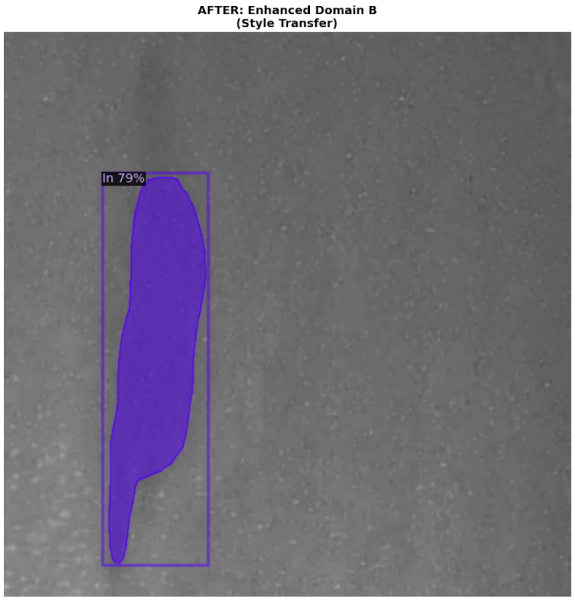
5.3 Qualitative Results (Visual Comparison)

The visual improvements are evident in the side-by-side comparisons. As seen below, the adaptation step removes the sensor noise, allowing the Mask R-CNN to draw tighter, more confident boundaries.

Visualization Samples: [Left side is Raw image, Right side is Enhanced Image]

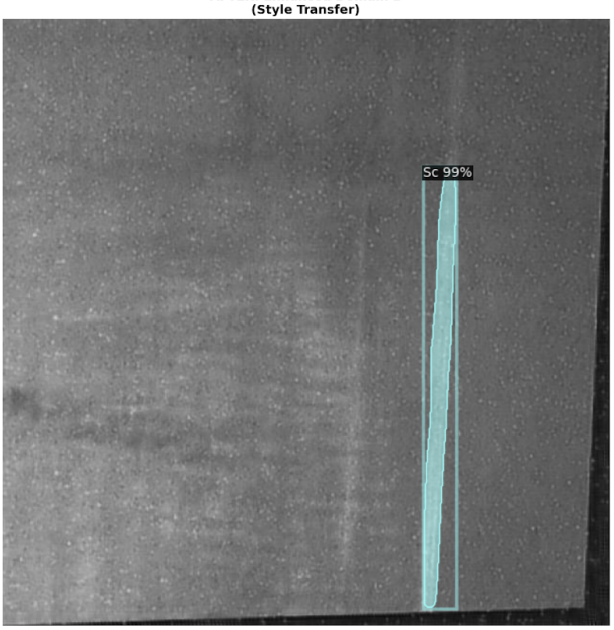




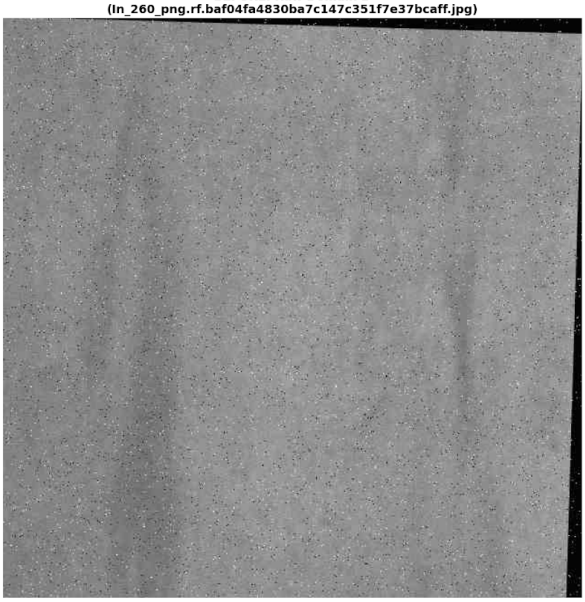


BEFORE: Raw Domain B
(Sc_84_png.rf.babc8b7e4c9974de6075774695da5b3e.jpg)

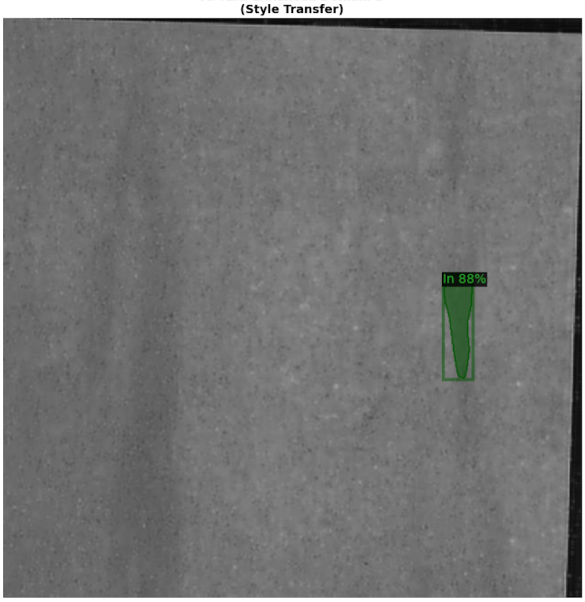
AFTER: Enhanced Domain B
(Style Transfer)



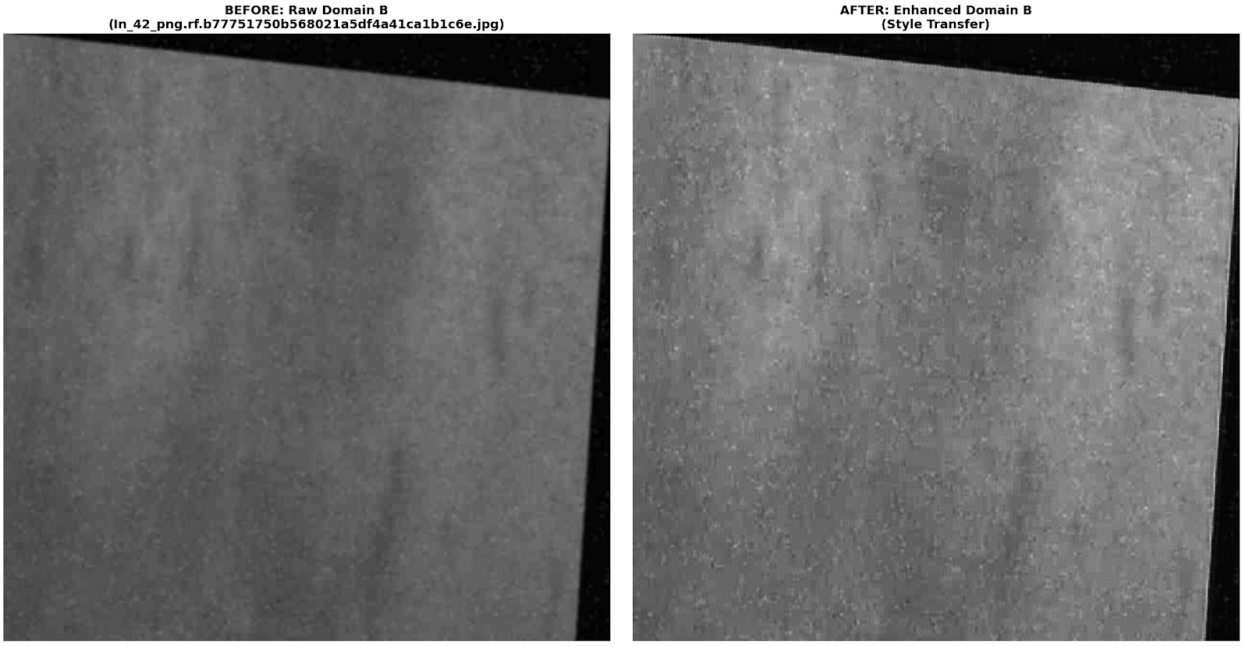
BEFORE: Raw Domain B
(In_260_png.rf.baf04fa4830ba7c147c351f7e37bcaff.jpg)



AFTER: Enhanced Domain B
(Style Transfer)



Failure Cases:



---

6. Conclusion

This project successfully demonstrated a pipeline for Cross-Domain Industrial Inspection. While the standard Mask R-CNN performs reasonably well, it suffers when image quality degrades. By integrating a Domain Adaptation step (transforming messy images to clean ones), we improved the **Bounding Box AP50 by over 5%**. This confirms that style transfer is a viable preprocessing step for deploying computer vision models in uncontrolled industrial environments.

---

7. References

1. He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). *Mask R-CNN*. ICCV.
2. Zhu, J., et al. (2017). *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. ICCV.
3. Song, K., et al. (2013). *NEU Surface Defect Database*.
4. NEU-seg Computer Vision Dataset : https://universe.roboflow.com/school-4pxkq/neu-seg

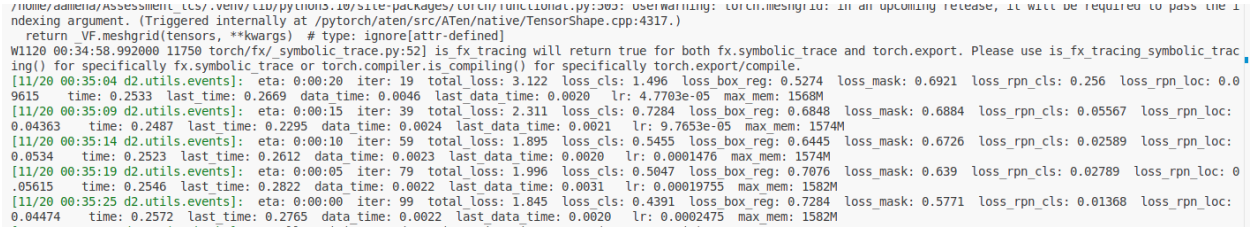
IMPLEMENTATION CODE & OUTPUTS

Module 1: Model Training (Mask R-CNN)

Logic Explanation: "To perform instance segmentation, we utilized **Transfer Learning** with a **ResNet-50-FPN** (Feature Pyramid Network) backbone, pre-trained on the COCO dataset. The training pipeline minimizes a **Multi-Task Loss Function** which simultaneously optimizes for Class Label (Cross-Entropy), Bounding Box coordinates (Smooth L1 Loss), and Pixel-wise Masks (Binary Cross-Entropy). The script registers the NEU-Seg dataset, configures the hyperparameters (Learning Rate: 0.00025, SGD Optimizer), and executes the training loop to fine-tune the model weights."

Source Code (`BASERCNN.ipynb`):

Output Screenshot:



Module 2: Evaluation & Metrics

Logic Explanation: To strictly quantify model performance on the Target Domain (Domain B), we developed an evaluation script utilizing the COCO Evaluator API. Unlike simple classification accuracy, this script calculates Mean Average Precision (mAP) by computing the Intersection over Union (IoU) between predicted masks and ground truth annotations across 10 distinct thresholds (0.50 to 0.95). This ensures the model is penalized for both false positives and poor boundary localization, providing a robust measure of generalization.

Source Code (`EvaluationOFPipeline.ipynb`):

Output on Domain A:

```
============================================================

📌 Bounding Box Metrics (Box IoU):
|                          |    Score |
|:-------------------------|---------:|
| Mean AP (IoU=0.50:0.95)  |  60.292  |
| AP @ IoU=0.50 (Lenient)  |  91.7531 |
| AP @ IoU=0.75 (Strict)   |  66.5658 |
| AP (Small Objects)       |  18.23   |
| AP (Medium Objects)      |  46.4543 |
| AP (Large Objects)       |  68.4966 |
| AP-In-"Pa",-"Sc"         | nan      |
| AP-In                    |  53.122  |
| AP-Pa                    |  74.3022 |
| AP-Sc                    |  53.4519 |

📌 Segmentation Metrics (Mask IoU):
|                          |     Score |
|:-------------------------|----------:|
| Mean AP (IoU=0.50:0.95)  |  48.7867  |
| AP @ IoU=0.50 (Lenient)  |  86.4916  |
| AP @ IoU=0.75 (Strict)   |  50.1223  |
| AP (Small Objects)       |   7.51824 |
| AP (Medium Objects)      |  34.6286  |
| AP (Large Objects)       |  55.9789  |
| AP-In-"Pa",-"Sc"         | nan       |
| AP-In                    |  43.6568  |
| AP-Pa                    |  67.3398  |
| AP-Sc                    |  35.3636  |
```

Output on Domain B:

📌 Bounding Box Metrics (Box IoU):

	Score
Mean AP (IoU=0.50:0.95)	42.1693
AP @ IoU=0.50 (Lenient)	70.4845
AP @ IoU=0.75 (Strict)	43.2842
AP (Small Objects)	3.21235
AP (Medium Objects)	29.1015
AP (Large Objects)	49.0627
AP-In-"Pa",-"Sc"	nan
AP-In	25.0671
AP-Pa	61.3276
AP-Sc	40.1133

📌 Segmentation Metrics (Mask IoU):

	Score
Mean AP (IoU=0.50:0.95)	34.4699
AP @ IoU=0.50 (Lenient)	65.993
AP @ IoU=0.75 (Strict)	32.4679
AP (Small Objects)	1.97682
AP (Medium Objects)	21.2676
AP (Large Objects)	40.7409
AP-In-"Pa",-"Sc"	nan
AP-In	20.8075
AP-Pa	57.2849
AP-Sc	25.3173

Module 3: Domain Adaptation (CycleGAN)

Logic Explanation: To address the domain shift caused by sensor noise and lighting variations, we employed an Unpaired Image-to-Image Translation approach (CycleGAN). The logic involves training a Generator network (G:B→A) to map messy 'Domain B' images into the feature space of clean 'Domain A' images. The training minimizes a composite loss function including Adversarial Loss (to ensure visual realism) and Cycle Consistency Loss (to ensure structural defect details are preserved during translation)

Source Code:

```
!python train.py \
   --dataroot ./datasets/steel_AB \
   --name steel_AB_cyclegan \
   --model cycle_gan \
   --batch_size 1 \
   --preprocess resize_and_crop \
   --load_size 286 \
   --crop_size 256 \
   --display_freq 200 \
   --print_freq 200 \
   --no_html \
   --n_epochs 20 \
   --n_epochs_decay 20 \
   --direction BtoA \
```

Output:

```
🚀 Training CycleGAN...
---------------- Options ---------------
               batch_size: 1
                    beta1: 0.5
          checkpoints_dir: ./checkpoints
           continue_train: False
                crop_size: 256
                 dataroot: ./datasets/steel_AB                [default: None]
           dataset_mode: unaligned
              direction: BtoA                                 [default: AtoB]
           display_freq: 200                                  [default: 400]
          display_winsize: 256
                    epoch: latest
              epoch_count: 1
                 gan_mode: lsgan
                init_gain: 0.02
                init_type: normal
                 input_nc: 3
                  isTrain: True                               [default: None]
                 lambda_A: 10.0
                 lambda_B: 10.0
          lambda_identity: 0.5
                load_iter: 0                                  [default: 0]
                load_size: 286
                       lr: 0.0002
...
saving the latest model (epoch 40, total_iters 65000)
learning rate 0.0000095 -> 0.0000000
saving the model at the end of epoch 40, iters 65120
End of epoch 40 / 40     Time Taken: 577 sec
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Module 4: Domain Transformation of Images

Logic Explanation: Once the CycleGAN model was trained, we utilized the Generator (GB→A) in **inference mode** to process the entire validation dataset. This step functions as a global preprocessing filter. The script iterates through the raw 'Domain B' images, applies the learned style transfer transformation to suppress sensor noise and normalize illumination, and saves the resulting 'Enhanced' images to disk. These transformed images serve as the improved input for the segmentation model.

Source Code:

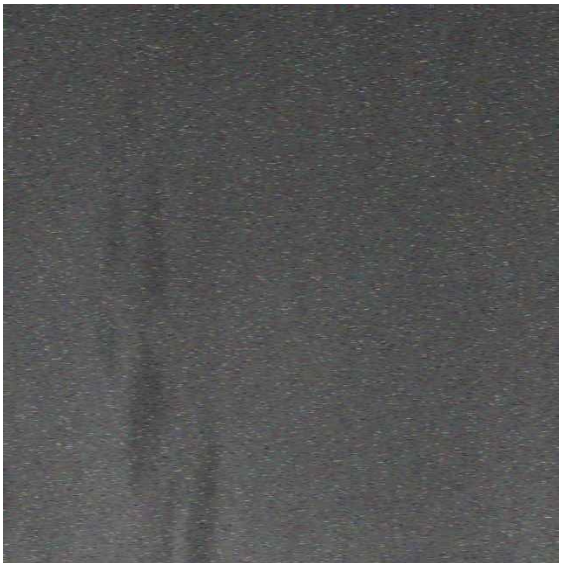
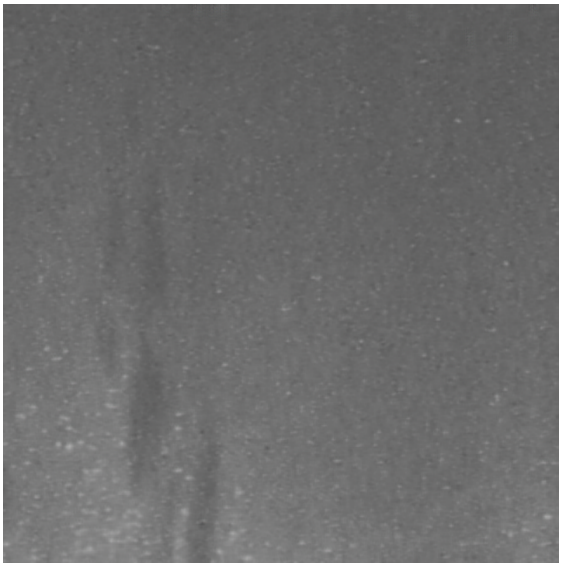
```
!python test.py \
   --dataroot "./datasets/NEU-seg-DOMAIN_B2" \
   --name steel_AB_cyclegan \
   --model cycle_gan \
   --direction BtoA \
   --num_test 5000 \
   --preprocess resize \
   --load_size 640 \
   --crop_size 640 \
   --no_dropout
```

Output Screenshot:

```
---------------- Options ---------------
             aspect_ratio: 1.0
               batch_size: 1
          checkpoints_dir: ./checkpoints
                crop_size: 640                                 [default: 256]
                 dataroot: /home/aamena/Assessment_tcs/datasets/NEU-seg-DOMAIN_B2       [default: None]
             dataset_mode: unaligned
                direction: BtoA                                [default: AtoB]
          display_winsize: 256
                    epoch: latest
                     eval: False
                init_gain: 0.02
                init_type: normal
                 input_nc: 3
                  isTrain: False                               [default: None]
                load_iter: 0                                   [default: 0]
                load_size: 640                                 [default: 256]
         max_dataset_size: inf
                    model: cycle_gan                           [default: test]
               n_layers_D: 3
                     name: steel_AB_cyclegan                   [default: experiment_name]
                      ndf: 64
                    netD: basic
                    netG: resnet_9blocks
                      ngf: 64
...
processing (1710)-th image... ['/home/aamena/Assessment_tcs/datasets/NEU-seg-DOMAIN_B2/testB/Sc_91_png.rf.8fbce024cada4dcf2e4644969ed535ec.jpg']
processing (1715)-th image... ['/home/aamena/Assessment_tcs/datasets/NEU-seg-DOMAIN_B2/testB/Sc_94_png.rf.ad291a9f82b2dc14e6b7de6731058232.jpg']
processing (1720)-th image... ['/home/aamena/Assessment_tcs/datasets/NEU-seg-DOMAIN_B2/testB/Sc_97_png.rf.922a0d00430a7afbf09a5912738d9f5e.jpg']
processing (1725)-th image... ['/home/aamena/Assessment_tcs/datasets/NEU-seg-DOMAIN_B2/testB/Sc_99_png.rf.a387bc2ef75144ffe87ce490136217e6.jpg']
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Sample Example:

BEFORE	AFTER
	

Module 5: Complete Inference Pipeline

Logic Explanation: The final pipeline integrates the domain adaptation and segmentation steps. It accepts a raw image, enhances it using the CycleGAN generator, and passes the result to Mask R-CNN. The code below demonstrates the visualization logic used to generate the qualitative results.

Output Screenshot: