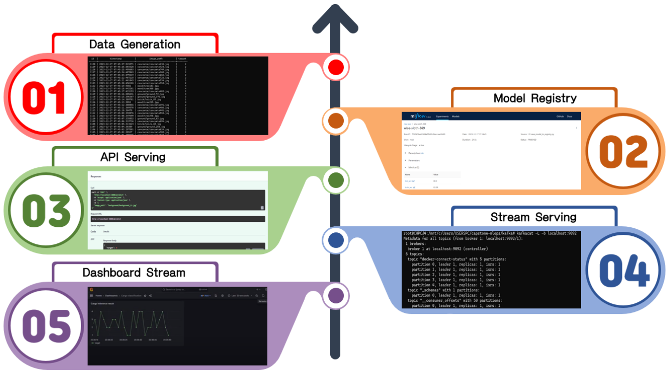
Supplementary material

BCCP: An MLOps Framework for Self-cleansing Real-Time Data Noise via Bayesian Cut-off-based Closest Pair Sampling

I. Method

A. Dataset & Model



```
root@CHPCJ4:/mnt/c/Users/USERSPC/capstone-mlops/kafka# kafkacat -L -b localhost:9092
Metadata for all topics (from broker 1: localhost:9092/1):
 1 brokers:
  broker 1 at localhost:9092 (controller)
 6 topics:
  topic "docker-connect-status" with 5 partitions:
    partition 0, leader 1, replicas: 1, isrs: 1
    partition 1, leader 1, replicas: 1, isrs: 1
    partition 2, leader 1, replicas: 1, isrs: 1
    partition 3, leader 1, replicas: 1, isrs: 1
    partition 4, leader 1, replicas: 1, isrs: 1
  topic "_schemas" with 1 partitions:
    partition 0, leader 1, replicas: 1, isrs: 1
  topic "__consumer_offsets" with 50 partitions:
    partition 0, leader 1, replicas: 1, isrs: 1
```

Fig. 1. An operational process of the MLOps pipeline

TABLE I
DATASET CONFIGURATION

Dataset	Class	Monday	Tuesday
Cargo	Background	43	42
	Brick	145	144
	Concrete	378	378
	Ground	983	978
	Wood	191	190
Bag	Garbage bag	500	500
	Paper bag	500	500
	Plastic bag	500	500
Sugarcane Leaf Disease	Healthy	261	261
	Mosaic	231	231
	Redrot	259	259
	Rust	257	257
	Yellow	252	253

The first dataset involves images of cargo entering a factory captured by cameras [1]. It encompasses background, brick, concrete, ground, and wood images. The second dataset entails images of plastic, paper, and garbage bags [2]. The third dataset consists of images of sugarcane leaf diseases collected in Maharashtra, India [3]. It possesses healthy, mosaic, red rot, rust, and yellow disease images. The dataset composition is as follows, as illustrated in Table I.

For detection and segmentation, we utilize Detectron2 [4]. Detectron2 is an advanced platform for object detection and segmentation. It offers a wide range of models, including traditional ones like Faster and Mask R-CNN, as well as newer models like Cascade R-CNN. The dataset for detection and segmentation includes bounding boxes and masks corresponding to images featuring background, car, wheel, light, and windows [5].

B. Reuse Buffer & BCCP Sampling

Eq. 1, 2, and 3 refer to L1-norm, L2-norm, and cosine similarity, respectively.

$$||A - X||_1 = \sum |A - X| \tag{1}$$

$$||A - X||_2 = \sum (|A - X|^2)^{\frac{1}{2}} \tag{2}$$

$$cosine\, similarity(A, X) = \frac{A \cdot X}{||A||_2 \times ||X||_2} \tag{3}$$

II. Experiment

A. Experimental Setup

The experiments are conducted using CUDA 12.1 on a single NVIDIA GeForce RTX 4070 graphics card, with model implementations using PyTorch 2.1.2. For image classification tasks, the objective is to minimize cross entropy loss with a learning rate of 0.001, a batch size of 64, and 15 epochs. For object detection and segmentation, a learning rate of 0.001, a batch size of 2, and 10 epochs are applied. The experiments are structured to maintain an 8:2 ratio between the train and test datasets. Considering the real-time and fast operation of MLOps, we set the number of epochs to a modest value.

B. Preprocessing

To generate images with noise for data corruption scenarios, three methods, including random boxes, mosaic, and random lines, are applied. For a detailed illustration, please refer to Fig. 2. Firstly, the random boxes method is designed to presume scenarios where snowfall obscures the image. It introduces image data by adding 10 to 20 white rectangles with sizes ranging from 1/100 to 1/25 of the image. Second, the mosaic method simulates situations where water droplets on the camera lens cause blurring. It generates image data by adding 20 to 30 mosaics with sizes between 1/100 and 1/25 of the image. Finally, the random line method assumes cases where parts of images are corrputed due to camera malfunctions. It generates image data by adding randomly oriented lines crossing the image, with thicknesses ranging from 1/10 to 1/5 of the image's diagonal length.

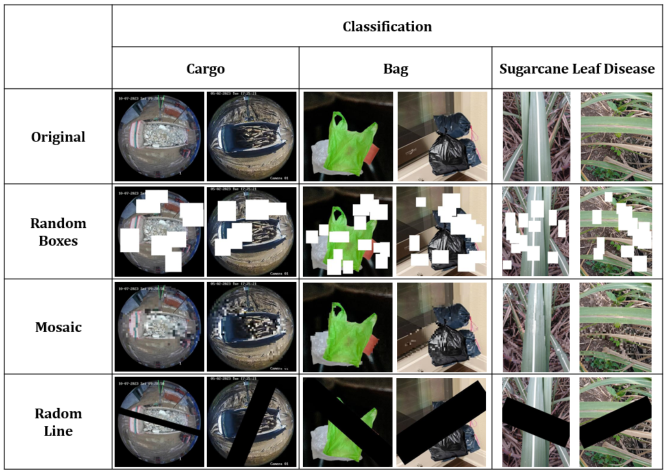


Fig. 2. Image Dataset with Noise

C. Detection and Segmentation Results

We construct a integrated framework for detection and segmentation. It shows proper AP_{50} of 0.700 and 0.677, respectively.

TABLE II
DETECTION AND SEGMENTATION RESULTS

Task	AP	$\mathbf{AP_{50}}$	$\mathbf{AP_{75}}$
Detection	0.490 ± 0.007	0.700 ± 0.010	0.569 ± 0.016
Segmentation	0.488 ± 0.004	0.677 ± 0.013	0.535 ± 0.005

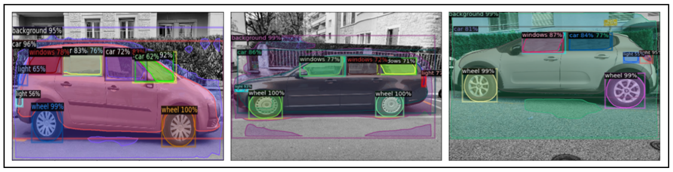


Fig. 3. Car Detection and Segmentation Results

REFERENCES

[1] Kaggle Homepage for Cargo Images, https://www.kaggle.com/datasets/morph1max/definition-of-cargo-transportation. Last accessed 27 August 2024

[2] Kaggle Homepage for Plastic, Paper, and Garbage Bag Synthetic Images, https://www.kaggle.com/datasets/vencerlanz09/plastic-paper-garbage-bag-synthetic-images/data. Last accessed 27 August 2024

[3] Kaggle Homepage for Sugarcane Leaf Disease Dataset, https://www.kaggle.com/datasets/nirmalsankalana/sugarcane-leaf-disease-dataset?select=Mosaic. Last accessed 27 August 2024

[4] Detectron2 Homepage, https://detectron2.readthedocs.io/en/latest/. Last accessed 27 August 2024

[5] Kaggle Homepage for Car Segmentation Images, https://www.kaggle.com/datasets/intelecai/car-segmentation. Last accessed 27 August 2024