Yunbo’s tasks (ZL proposed)

1.

(1) Evaluate the Open-source anomaly detection algorithms library "anomalib" from the OpenVINO toolkit (<https://github.com/openvinotoolkit/anomalib>) for industrial quality inspection.

1.1 Develop an adaptive quality inspection system in the robot lab, integrating the "anomalib" to detect anomalies in manufacturing processes. (baseline)

1.2 Create reusable building blocks for anomaly detection using the "anomalib" to facilitate easier integration into various industrial quality inspection applications.

1.3 Investigate the application of RGBD data for anomaly detection in industrial settings, utilizing the "anomalib" to analyze multi-modal data. (optional)

1.4 Explore the use of Large Language Models (LLMs), such as GPT-3 or similar, for semantic anomaly detection in conjunction with the "anomalib" to enhance the interpretability of anomalies and improve the detection accuracy. (optional)

2.

(2) Characterize low-cost vision systems in comparison with commercial industrial vision systems.

In this task, compare and contrast low-cost vision systems with established commercial industrial vision systems to identify their strengths, limitations, and potential use cases in different industrial scenarios. Consider factors such as image resolution, processing speed, accuracy, ease of integration, and cost-effectiveness in the characterization process. Additionally, explore ways to optimize and improve the performance of low-cost vision systems to bridge the gap between them and high-end commercial systems.

Phase1:

1. Minimum data for training
2. Faults types (color,shape,surface,position,rotation)
3. Multi-type products
4. Background changes
5. Model deploy in Raseberrry Pi (where computing)
6. Computing resources cost

Variables:

4-5 models

2 products

Environment: two lighting-setting

Output: machine learning methods + computing system

Star map for each model

Adaptability---

Training resources----time

Robustness---

Output1: Best practice (setup)

Phase2:

1. System development
2. Function development(Multi-type products) generability and individuality
3. Location（including position and robot scrapping）

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Week0  (8.8-8.11) | Week1 | Week2 | Week3 | Week4 | Week5 | Week6 |
| Task1 | 1.1 Anomalib example learning | 1.2. data collection and define the data size | 1.2. Deploy the system in the real environment | 1.3.Try the RGBD data with anomalib | 1.4.try to use LLM models to do description | 1.4.try to deploy the LLM system | 2./ |
| Task2 | 1.1 Anomalib test on custom datasets | 1.2. Building the cv system and do multiple test to find the best model | 1.2. Testing with the Real Camera | 1.3. quickly employ the new system | 1.4. Build the system by using the open resources | 1.4.try to deploy the LLM system | 2./ |
| Task3 | 1.4 explore the possibility of LLMs |  |  |  |  |  |  |

**Week0**

Task1. Anomalib has anomaly detection models available in `anomalib` library, which need to explore further

- [CFA](https://arxiv.org/abs/2206.04325)

- [CS-Flow](https://arxiv.org/abs/2110.02855v1)

- [CFlow](https://arxiv.org/pdf/2107.12571v1.pdf)

[DFKDE](https://github.com/openvinotoolkit/anomalib/tree/main/anomalib/models/dfkde)

- [DFM](https://arxiv.org/pdf/1909.11786.pdf)

- [DRAEM](https://arxiv.org/abs/2108.07610)

- [FastFlow](https://arxiv.org/abs/2111.07677)

- [Ganomaly](https://arxiv.org/abs/1805.06725)

- [Padim](https://arxiv.org/pdf/2011.08785.pdf)

- [Patchcore](https://arxiv.org/pdf/2106.08265.pdf)

- [Reverse Distillation](https://arxiv.org/abs/2201.10703)

- [R-KDE](https://ieeexplore.ieee.org/document/8999287)

- [STFPM](<https://arxiv.org/pdf/2103.04257.pdf>)

And the anomaly detection results for the bottle bottom are shown below.

A close-up of a bottle

Description automatically generated A blue and green light

Description automatically generated with medium confidence

Figure 1 bottle bottom detection process 1

A close-up of a blue circle

Description automatically generated A yellow object on a purple background

Description automatically generated

Figure 2 bottle bottom detection process 2

A close up of a bottle

Description automatically generated

Figure 3 bottle bottom detection final result

2. Demo test on Custom dataset

A black and white grid

Description automatically generated

Figure 4 demo detection final result

The Dataset only has 8 photos including 6 good ones and 2 bad ones

A close-up of several round objects

Description automatically generated

A close-up of several circular objects

Description automatically generated

A collage of images of a blue wall

Description automatically generated

Figure 5 demo detection results

3. LLM solution

Tried to use several plugins in Chatgpt4 to recognize the images, the results were poor and **the model could not recognize the gears**, so the future goal is to find the LLM model trained on industrial image data and test it further

23/08/2023

Intel 的Anomalib

一个端到端的无监督异常检测---数据不均衡，正常多，异常少

边缘设备检测

A screenshot of a computer

Description automatically generated

A close-up of several different types of software

Description automatically generated

收集图片

训练

测试

OpenVINO优化与部署

高清模型不适合AI训练

后处理：归一化打分，可视化

A screenshot of a computer

Description automatically generated

OPENVINO优化，压缩

at the [2022 Conference on Computer Vision and Pattern Recognition](https://paularamo.github.io/cvpr-2022/)

人工智能要真正增强质量控制和质量保证体验，它必须能够利用平衡的数据集。尽管现在有大量可用的良好数据样本，但它并不总是足以在工业和医疗行业做出准确有效的预测。此外，大规模制造和工业自动化的发展使得质量检验员处理大量产品变得越来越困难。

基于监督学习的方法利用足够的带注释的异常样本来实现足够的异常检测结果。但是，如果行业规范是一个不平衡的数据集，缺乏异常类中的代表性样本怎么办？当缺陷可以是任何形状时，如何定义异常的边界？

对于这种场景下的异常检测，我们没有硬件加速器可以在边缘训练模型。我们也不能假设已经收集了数千张图像，尤其是有缺陷的图像，用于边缘训练。此外，预计不会出现很多缺陷，这在现实制造场景中是常见的。

考虑到这些初始条件，我们的目标之一是在边缘实现更快的训练过程，并以高精度和高效率执行异常检测。重要的是要记住，如果有任何外部条件变化（例如照明、相机或异常），我们将不得不重新训练我们的模型，因此不需要太多努力的重新训练过程是必要的。最后，为了使模型在实际制造用例中有用，我们必须保证异常检测模型的推理结果准确。

模型

在您所描述的上下文中，"预训练 CNN "一词指的是在大型、多样化数据集（如 ImageNet）上训练过的卷积神经网络。通过在 ImageNet 上的初始训练，CNN 可以学会识别图像中的各种特征、模式和结构。ImageNet 是一个数据集，包含数以百万计的标注图像，涉及数千个类别，是学习一般视觉表征的合适来源。

一旦在 ImageNet 上对 CNN 进行了预训练，它就会开发出一套学习参数，用于捕捉图像中不同的抽象层次。这些参数基本上就是网络各层的权重和偏置。这些学习到的表征可以作为其他任务的起点。

在 PaDiM 和 PatchCore 等算法中，预训练 CNN 被用作特征提取器。图像通过预训练的 CNN，网络各层的激活被视为描述输入图像不同方面的特征或嵌入。然后，这些嵌入信息将用于进一步处理，例如 PaDiM 和 PatchCore 的异常检测。

总之，这里的 "预训练 "是指在 ImageNet 等大型数据集上训练神经网络，学习一般的视觉表征，然后再针对特定任务进行调整或微调，例如上述算法中的异常检测。与针对特定任务从头开始训练模型相比，这种方法可以节省时间和计算资源。