HW3_113034572_王奕鈞

1.

A. Number of defect classes：1 class (All defects are classified as defective；
In MVTec's design, toothbrush's database does not further subdivide defect types).

B. Types of defect classes：Indicates that the toothbrush is abnormal, such as: Bristles are skewed、Defect or deformation、Surface damage, etc.

C. Number of images used in your dataset：

Split	Class	Number of Images
Training	good	60
Test	good	12
Test	defective	30
Ground truth	defective (mask)	30

D. Distribution of training and test data：

- train/good：60 normal (non-defective) images for training
- test/good：12 normal images for testing
- test/defective：30 defective images for testing
- ground_truth/defective：30 binary mask images that highlight the defect regions.

E. Image dimensions：All images: 1024 × 1024 pixels, RGB (3 channels).

2.

Method 1: Partial Fine-Tuning of Pretrained Weights

- Adjustment:：Unfreezes the later layers (e.g., layer4) and the fully connected (fc) layer, allowing the model to adapt high-level features to the new dataset.
- Result：Test Accuracy: 75.00%；Helped improve feature extraction but showed instability and signs of overfitting on limited data.
- Pros: Leverages pretrained features and further fine-tunes them to extract representations more suitable for the new data；Overall good performance with the ability to distinguish between the two classes..
- Cons: Training can be unstable without careful learning rate tuning；Risk of overfitting when the dataset is small, requiring additional techniques like regularization or data balancing..

Method 2: Changing Optimizer to SGD with Momentum

- Adjustment：Replaced the optimizer with SGD and added momentum (0.9).
- Result：Test Accuracy: 28.57%；Predictions are heavily biased toward a single class.

- Pros：In theory, SGD provides better generalization and convergence over time.
- Cons：In this setup, it failed to update the model effectively. Training accuracy was high, but validation/test performance was poor—indicating severe overfitting and class bias；Likely caused by inappropriate learning rate or missing hyperparameter tuning.

Method 3：Enhanced Data Augmentation

- Adjustment:：Applied RandomCrop, RandomHorizontalFlip, ColorJitter, and RandomRotation to the training data.
- Result：Test Accuracy: 28.57%；Predictions remain biased toward one class (all class 0), indicating no real improvement.
- Pros：Data augmentation is generally helpful to reduce overfitting and improve robustness.
- Cons：In this case, augmentation alone wasn't sufficient to resolve the core issue—likely due to data imbalance or lack of complementary improvements like model architecture tuning or loss weighting.

Method 4: Tuning Learning Rate and Adding Scheduler

- Adjustment：Reduced learning rate from 1e-3 to 5e-4；Applied StepLR scheduler to halve the learning rate every 10 epochs.
- Result：Test Accuracy: 66.67%；Predictions were more balanced compared to Methods 2 and 3, though still imperfect.
- Pros：A smaller learning rate and scheduler helped the model converge more stably, with visible improvement in validation and test accuracy.
- Cons：While improved, performance still lags behind Method 1, and some overfitting was observed in later epochs.

Final Recommendation

- Method 1 (Partial Fine-Tuning) currently delivers the best results with 75.00% test accuracy and is the most reliable at distinguishing between classes.
- Method 4 also performed well (66.67%) and highlights the importance of proper learning rate tuning and scheduler use.

Future Improvements

- Combine Strategies：Combine Method 1 and Method 4: fine-tune specific layers while using a learning rate scheduler to stabilize training.
- Address Class Imbalance：Apply class weighting, weighted samplers, or oversampling to reduce class

prediction bias.

- Regularization Techniques：Introduce dropout, early stopping, or L2 regularization to mitigate overfitting.
- Expand Dataset：If possible, collect more data to help the model generalize better and capture more diverse patterns.

3.

(i) A long-tail distribution refers to the phenomenon where a small number of classes (head classes) have many samples, while the majority of classes (tail classes) have very few samples. This imbalance can cause models to be biased towards head classes during training, resulting in poor performance on the underrepresented tail classes.

(ii) They proposed a method called Balanced Group Softmax (BAGS), which divides classes into groups and applies softmax within each group separately. This approach helps balance the classifier training and improves detection performance on long-tail distributions.

To apply this method to the MVTec AD dataset, we could group the 'Good' class separately from the defect classes and apply softmax within each group. This would prevent the model from being overly biased toward the 'Good' class, thereby enhancing its ability to detect defects.

4. Since the training set of the MVTec AD dataset mainly contains 'Good' images and lacks defective samples, it is suitable to adopt unsupervised or self-supervised anomaly detection methods. A common strategy is to train a reconstruction-based model such as an Autoencoder or Variational Autoencoder (VAE) using only normal samples. The model learns to reconstruct normal images, and during testing, defective images yield higher reconstruction errors, which can be used to detect anomalies. Another approach involves using pre-trained feature extractors (e.g., CNNs) and applying distance-based metrics like the Mahalanobis distance to identify abnormal patterns.

5.

(i) To fine-tune an object detection model like YOLO-World, the dataset must include images with bounding box annotations for each defect, along with class labels. For segmentation tasks using models like SAM, pixel-wise annotations are required, where each pixel is labeled as either normal or a specific defect type. These annotations are usually provided in the form of masks.

(ii) These models are well-suited for fine-tuning on our custom dataset because they offer strong feature extraction capabilities and high transferability. YOLO-World can efficiently detect and localize custom defect categories, while SAM excels at precise segmenting regions of interest, which is critical for identifying subtle or irregular defect patterns. Additionally, both are open-source and highly adaptable, making them practical choices for improving anomaly detection accuracy and detail.