Part 2: Reasoning-Based Questions

Q1: Choosing the Right Approach

To find out if a product is missing its label, I would choose object detection. Detection is better than classification because classification only tells us “Yes or no,” but it doesn’t show *where* the label is. Detection, on the other hand, can draw a box around the label, so we know exactly if and where it exists. If the label is very small or hard to see, we may even use segmentation, which highlights the label pixel by pixel. This is more detailed but also more complex and requires more data. I would start with detection since it gives a good balance of accuracy and simplicity. If detection fails due to tricky cases like overlapping labels or small sizes, then segmentation would be my backup plan. This way, we get both efficiency and accuracy depending on the situation.

Q2: Debugging a Poorly Performing Model

If my model performs poorly on new factory images, the first thing I’d check is data mismatch. Maybe the training images had clear lighting or plain backgrounds, but the real factory images are darker, blurrier, or busier. Then I’d check the annotations to make sure they were correct, because bad labels confuse the model. I would also visualize predictions to see whether the model is consistently missing certain types of images. Another experiment I’d try is training on a smaller but cleaner dataset, just to confirm if the issue is with data or the model itself. I might also test with data augmentation (like rotation, brightness change) to see if that helps the model adapt. Lastly, I’d review the model’s confidence scores to check if it is unsure or overconfident on wrong predictions. By following this checklist, I can figure out whether the problem is poor data, poor labels, or the wrong model choice.

Q3: Accuracy vs Real Risk

Accuracy sounds impressive at 98%, but it is misleading in this case. Even though the accuracy is high, the model is still missing 1 out of 10 defective products, which is a serious issue for factory quality control. If even one faulty item goes through, it can cause big financial loss, safety hazards, or damage to the company’s reputation. Instead of accuracy, I would look at recall, which measures how many of the defective products the model actually catches. I’d also check the precision-recall balance, because we don’t want too many false alarms either. In critical applications, recall is more important because missing a defective product is riskier than flagging a good one by mistake. I might also monitor the confusion matrix to see where errors are happening. In short, the right metric is not accuracy but recall, since that directly affects safety and reliability.

Q4: Annotation Edge Cases

When labelling data, we often get blurry or half-visible products. These images are important because real-life factory cameras will never give us perfect pictures every time. If we throw these cases away, the model may only learn from “perfect” examples and fail badly in real-world usage. However, if we keep them, they need to be labelled carefully so the model learns how to handle such imperfect situations. The trade-off here is between having clean data that makes training easier versus realistic data that makes the model stronger. A smart approach is to include them but maybe tag them separately, so the model can learn that blurry or partial objects are still valid cases. This makes the system more robust and reliable when deployed. In real-world AI, it’s usually better to prepare the model for messy, imperfect data than to train only on ideal situations.