Part 2: Reasoning-Based Questions - Answers

Q1: Choosing the Right Approach

Question: You are tasked with identifying whether a product is missing its label on an assembly line. The products are visually similar except for the label. Would you use classification, detection, or segmentation? Why? What would be your fallback if the first approach doesn't work?

Answer:

I would start with **object detection** as the primary approach for identifying missing labels on an assembly line. Detection is ideal because it not only determines if a label is present (classification) but also locates where the label should be positioned on the product, which is crucial for quality control systems. The bounding box output provides spatial information that can be used to verify proper label placement and orientation. Detection models like YOLOv8 can efficiently process images in real-time manufacturing environments while providing confidence scores for each detection.

If detection doesn't work well initially, my fallback would be **binary classification** combined with region-of-interest (ROI) analysis. I would crop the expected label area from each product image and train a simple CNN classifier to distinguish between "label present" and "label absent" classes. This approach reduces complexity by focusing only on the relevant image region and can be more robust when label variations are minimal. Additionally, I might implement a hybrid approach using template matching for consistent label designs, followed by classification for verification, which would be computationally efficient and highly accurate for standardized manufacturing processes.

Q2: Debugging a Poorly Performing Model

Question: You trained a model on 1000 images, but it performs poorly on new images from the factory. Design a small experiment or checklist to debug the issue. What would you test or visualize?

Answer:

I would design a systematic debugging experiment starting with **data distribution analysis**. First, I'd visualize sample images from both training and factory datasets side-by-side to identify domain shift issues like different lighting conditions, camera angles, or product variations that weren't represented in training. I would create confusion matrices and examine misclassified examples to understand specific failure patterns. Next, I'd check for **overfitting** by plotting training vs. validation loss curves and testing the model on a held-out subset of the original 1000 images to establish a baseline performance.

My debugging checklist would include: (1) **Image preprocessing consistency** - ensuring factory images undergo identical preprocessing (resizing, normalization) as training data, (2) **Class distribution verification** - confirming that factory data contains the same class proportions as training data, (3) **Augmentation testing** - applying data augmentation techniques like rotation, brightness adjustment, and noise addition to simulate factory conditions, and (4) **Gradient visualization** using techniques like Grad-CAM to verify the model focuses on relevant image regions rather than spurious correlations. I would also implement cross-validation on the original dataset to ensure the model wasn't simply memorizing specific image artifacts rather than learning generalizable features.

Q3: Accuracy vs Real Risk

Question: Your model has 98% accuracy but still misses 1 out of 10 defective products. Is accuracy the right metric in this case? What would you look at instead and why?

Answer:

Accuracy is not the right metric for this defect detection scenario because it doesn't adequately reflect the business risk and safety implications of missing defective products. The 98% accuracy could be misleading if the dataset is imbalanced - for example, if only 2% of products are actually defective, a model could achieve 98% accuracy by simply predicting "non-defective" for everything while missing all actual defects. In manufacturing and safety-critical applications, the cost of missing a defective product (false negative) is typically much higher than incorrectly flagging a good product as defective (false positive).

Instead, I would focus on **recall (sensitivity)** as the primary metric, which measures the percentage of actual defective products correctly identified. A recall of 90% means we're catching 9 out of 10 defective products, which directly addresses the stated problem. I would also examine **precision** to understand false positive rates and calculate **F1-score** for a balanced view. Most importantly, I'd implement **cost-sensitive metrics** that weight false negatives much more heavily than false positives, reflecting the real-world consequence that missing a defective product could lead to customer safety issues, warranty claims, or brand damage. Additionally, I'd establish a **minimum recall threshold** (e.g., 95%) as a hard constraint, accepting lower overall accuracy if necessary to ensure safety and quality standards.

Q4: Annotation Edge Cases

Question: You're labeling data, but many images contain blurry or partially visible objects. Should these be kept in the dataset? Why or why not? What trade-offs are you considering?

Answer:

Yes, these challenging images should generally be kept in the dataset, but with careful consideration and potentially special handling. Blurry and partially visible objects reflect real-world conditions that the model will encounter in production, so including them improves model robustness and generalization. Excluding such images creates an artificially clean dataset that doesn't represent the full spectrum of input conditions, leading to poor performance when the model faces similar challenging scenarios in deployment. These edge cases help the model learn to handle uncertainty and make more reliable predictions across varying image quality conditions.

However, the key trade-off is **annotation quality versus dataset representativeness**. I would implement a tiered approach: (1) **Keep high-confidence partial objects** where the visible portion clearly indicates the object class, (2) **Use "difficult" flags** to mark ambiguous cases during training, allowing the model to learn from them without penalizing uncertain predictions, (3) **Create separate validation sets** with only clear examples to get accurate performance metrics, and (4) **Consider confidence thresholding** during inference to handle uncertain predictions appropriately. For severely blurred images where even human annotators disagree, I might exclude them or create a separate "uncertain" class. The goal is maintaining dataset realism while ensuring annotation consistency, as inconsistent labels on ambiguous images can confuse the model and degrade overall performance.

Submitted by: [Ayaan Shaheer]
Date: September 24, 2025
Course: Computer Vision Assignment - Part 2

Bibliography:

1 2 3 4 5 6 7 8 9