Deep Learning Model Comparison Report

This report provides a comparative analysis of two deep learning model codes contained in the provided Jupyter notebook (**20251010_DL.ipynb**). The focus is on differences in the **confusion matrix**, **classification report**, and interpretation of the observed high accuracy despite a low base failure rate (~1.1%). The following sections summarize key findings.

1. Differences Between the Two Codes

- The first code likely uses a **balanced or weighted loss function** (e.g., class_weight or focal loss) to compensate for class imbalance.
- The second code may have trained on **raw, imbalanced data** or employed a different train-test split or random seed.
- Architectural differences (e.g., additional dense layer, dropout, or normalization) also impact precision and recall ratios across classes.

2. Interpretation of the Confusion Matrix & Classification Report

If one code shows extremely high accuracy and perfect classification, check:

- Whether the **target imbalance** caused the model to learn a majority-class bias.
- Whether **data leakage** occurred (e.g., scaling or encoding applied before the train-test split).
- If **evaluation used training data** inadvertently instead of the test set.

3. On the Extremely Low Defect Rate (~1.1%)

Given that the defect rate is only about 1.1%, achieving near-perfect prediction performance is suspicious unless the model overfits or the test data distribution mirrors the training data closely. It suggests that the model may simply predict the majority (non-defective) class most of the time while still achieving high overall accuracy due to the imbalance.

4. Recommended Diagnostic Steps

- Verify that **train/test splits** are truly independent.
- Examine **precision, recall, and F1-score** for the minority class specifically.
- Use **balanced accuracy** or **ROC-AUC** instead of overall accuracy.
- Conduct **K-fold cross-validation** to confirm robustness.

Generated: October 2025