DeepGD3: Unknown-Aware Deep Generative/Discriminative Hybrid Defect Detector for PCB Soldering Inspection (Supplementary Material)

Ching-Wen Ma[1] **Yanwei Liu**[1]

[1]College of Artificial Intelligence, National Yang Ming Chiao Tung University, Tainan, Taiwan

A VISUAL EXPLANATION OF THE PREDICTION CONVERTER

τ_0 controls the variance of Gaussians. If it is small, the Gaussian looks shaper. If it is large, the Gaussian looks wider. Figure. 1 explain this concept. Bayesian optimization then optimizes the shape of Gaussians, making them proper for thresholding.

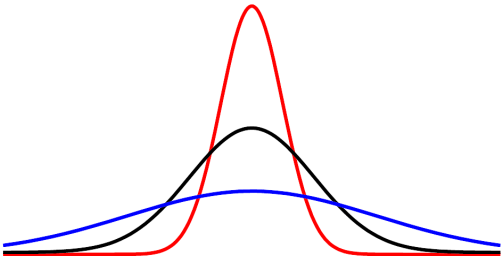

Figure 1: The effect of τ_0 on the shape of probability density function. Black curve: the baseline shape of the probability density function. Red curve: smaller τ_0 makes the shape of the probability density function concentrated. Blue curve: larger τ_0 makes the shape of the probability density function wider.

τ_1 and τ_2 adjust the thresholds $h^1_{j,n}$ and $h^1_{j,n}$ by adjusting the variance of the corresponding Gaussians. A larger τ_1 makes $h^1_{j,n}$ smaller, resulting in more samples being classified as good samples. A larger τ_2 makes $h^2_{j,n}$ smaller, resulting in fewer

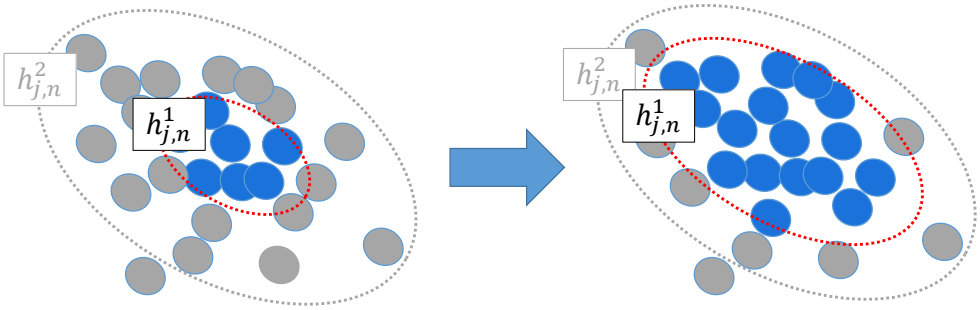

Figure 2: The effect of τ_1 and $h^1_{j,n}$. The larger the τ_1 is, or similarly, the smaller the $h^1_{j,n}$ is, the more test samples will be classified as good samples.

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

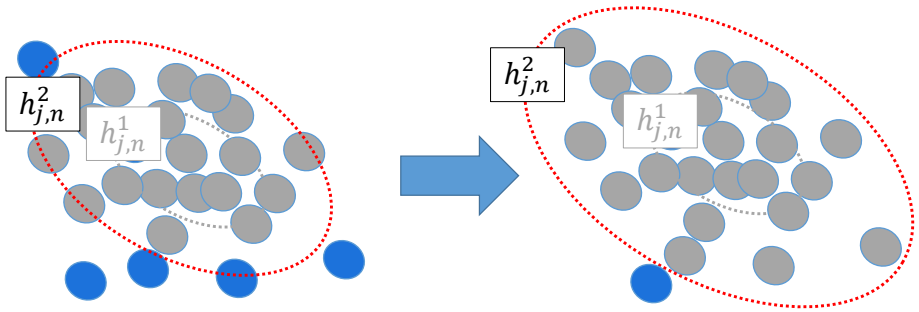

Figure 3: The effect of τ_2 and $h^2_{j,n}$. The larger the τ_2 is, or similarly, the smaller the $h^2_{j,n}$ is, the fewer test samples will be classified as unknown samples.

samples being classified as unknown samples. Figure 2 and Figure 3 explain this concept. Bayesian optimization then optimizes τ_1 and τ_2 for maximizing harmonic score H in main paper's Equation 5.

B ADDITIONAL EXPERIMENTAL DETAILS

We fine-tuned the MobileNetV3 large model pre-trained on ImageNet [Deng et al., 2009] using PyTorch and trained the defect and component classifiers.

The default hyperparameters were as follows: The number of training epochs was 30. The initial learning rate was 0.05. We used linear warm-up for the first 5 epochs and decay the learning rate with the cosine annealing scheduler; For the gradient descent optimization algorithm, stochastic gradient descent (SGD) was used; the momentum and weight decay were set at 0.9 and 0.0001. The input image was 224 $\times$ 224 color images.

In addition, the early stop technique was applied to prevent overfitting. We applied random horizontal flip, random vertical flip, color jitter, and random grayscale to the data loader of the training set. We did not apply any image augmentation to the validation and test set. A single NVIDIA Tesla V100 GPU was used for all experiments.

C ADDITIONAL SIMULATION RESULTS FOR ABLATION STUDY

We investigated the effects of (a) not including the good new compoent samples in the training data and (b) not using the combiner.

Comparisons between not including and including good new component samples in the training data. Table 1 shows the effects of including and not including good new component samples in the training data for all test samples, i.e. both old and new componet test samples. Table 2 and Table 3 show the performances for the old and new component test samples, respectively. The $(\cdot)^-$ symbol indicates training without including good new component samples.

Observing all the three tables, we notice that Expert 1 shows a lower overkill rate with including good new component samples in the training data, suggesting that the good new component samples help improve performance. Expert 2's leakage rates are greatly reduced when good new component samples are available during training. For the Hybrid Expert, if there are no good new component samples during the training period, the shared fully connected network $f_{\theta 2}$ may not be able to form proper clusters for these new components. As a result, the GMM model will not be able to correctly classify the new component samples, resulting in a higher unknown rate. However, both the overkill rates and the leakage rates still remains low. This implies that even if good new component samples are not available during training, the Hybrid Expert remains trustworthy.

Comparisions between not using and using the prediction combiner Table 4 shows the performance of $\hat{y}^1_{def}$, $\hat{y}^2_{def}$, and $\hat{y}_{def}$. The results show that $\hat{y}^1_{def}$ produces a high leakage rate with a high standard deviation, and $\hat{y}^2_{def}$ achieves 0.691% $\pm$ 0.974% in terms of leakage rate, but also produces a high overkill rate. The prediction combiner combines $\hat{y}^1_{def}$ and $\hat{y}^2_{def}$, achieving an overkill rate of 0.108% $\pm$ 0.033% and a leakage rate of 0.063% $\pm$ 0.075%, with an unknown rate of 3.706% $\pm$ 2.270%.

Table 1: Comparisons between not including $(\cdot)^-$ and including $(\cdot)$ good new compoent samples as additional training data. The test performances for all test samlpes, i.e. both old and new component test samples.

	Overkill rate (%)	Leakage rate (%)	Unknown rate (%)
Expert 1^-	1.773 ± 1.459	0.327 ± 0.463	-
Expert 1	0.015 ± 0.008	1.827 ± 3.063	-
Expert 2^-	3.650 ± 2.902	11.419 ± 19.321	0.000 ± 0.000
Expert 2	1.954 ± 0.724	1.942 ± 1.337	0.000 ± 0.000
Hybrid Expert$^-$	0.189 ± 0.266	0.074 ± 0.114	26.089 ± 40.843
Hybrid Expert	0.108 ± 0.033	$\mathbf{0.063 \pm 0.075}$	3.706 ± 2.270

Table 2: Comparisons between not including $(\cdot)^-$ and including $(\cdot)$ good new compoent samples as additional training data. The test performances for old component test samples.

	Overkill rate (%)	Leakage rate (%)	Unknown rate (%)
Expert 1^-	0.097 ± 0.146	0.028 ± 0.018	-
Expert 1	0.017 ± 0.007	0.021 ± 0.011	-
Expert 2^-	3.293 ± 5.306	9.268 ± 15.373	0.000 ± 0.000
Expert 2	1.282 ± 0.192	2.257 ± 1.495	0.000 ± 0.000
Hybrid Expert$^-$	0.007 ± 0.010	0.042 ± 0.055	18.695 ± 31.605
Hybrid Expert	0.129 ± 0.110	$\mathbf{0.019 \pm 0.013}$	3.529 ± 3.002

Table 5 and Table 6 show the performance of $\hat{y}^1_{def}$, $\hat{y}^2_{def}$, and $\hat{y}_{def}$ for the old component test set and new component test set, respectively. For the old component settings, $\hat{y}_{def}$ yields the best overkill and leakage rate performance. For the new component setting, $\hat{y}_{def}$ achieves $0.063\% \pm 0.075\%$ in terms of leakage rate, indicating excellent performance for detecting new component defects.

D ALGORITHMS

Training procedure: The detailed training procedure of Figure 3 is described in Algorithm 1.

Inference procedure: The Hybrid Expert inference procedure is described in Algorithm 2.

References

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Table 3: Comparisions between not including $(\cdot)^-$ and including $(\cdot)$ good new component samples as additional training data. The test performances for new component test samples.

	Overkill rate (%)	Leakage rate (%)	Unknown rate (%)
Expert 1^-	4.591 ± 3.687	0.611 ± 0.824	-
Expert 1	0.010 ± 0.010	3.380 ± 5.540	-
Expert 2^-	3.368 ± 3.112	17.666 ± 30.599	0.000 ± 0.000
Expert 2	3.739 ± 2.459	0.989 ± 1.713	0.000 ± 0.000
Hybrid Expert$^-$	0.713 ± 1.033	0.096 ± 0.167	35.055 ± 46.395
Hybrid Expert	0.126 ± 0.062	$\mathbf{0.090 \pm 0.156}$	3.324 ± 1.553

Table 4: Performance of $\hat{y}^1_{def}$, $\hat{y}^2_{def}$, and $\hat{y}_{def}$ for all test samples, i.e. both old and new component test samples

	Overkill rate (%)	Leakage rate (%)	Unknown rate (%)
$\hat{y}^1_{def}$	0.390 ± 0.320	0.681 ± 1.114	-
$\hat{y}^2_{def}$	2.287 ± 1.424	0.691 ± 0.974	0.000 ± 0.000
$\hat{y}_{def}$	$\mathbf{0.108 \pm 0.033}$	$\mathbf{0.063 \pm 0.075}$	3.706 ± 2.270

Table 5: Performance of $\hat{y}^1_{def}$, $\hat{y}^2_{def}$, and $\hat{y}_{def}$ for old component test samples.

	Overkill rate (%)	Leakage rate (%)	Unknown rate (%)
$\hat{y}^1_{def}$	0.671 ± 0.794	0.034 ± 0.032	-
$\hat{y}^2_{def}$	2.227 ± 1.768	0.891 ± 1.343	0.000 ± 0.000
$\hat{y}_{def}$	$\mathbf{0.129 \pm 0.110}$	$\mathbf{0.019 \pm 0.013}$	3.529 ± 3.002

Table 6: Performance of $\hat{y}^1_{def}$, $\hat{y}^2_{def}$, and $\hat{y}_{def}$ for new component test samlpes.

	Overkill rate (%)	Leakage rate (%)	Unknown rate (%)
$\hat{y}^1_{def}$	0.150 ± 0.073	1.202 ± 2.064	-
$\hat{y}^2_{def}$	2.315 ± 0.592	0.090 ± 0.156	0.000 ± 0.000
$\hat{y}_{def}$	$\mathbf{0.126 \pm 0.062}$	$\mathbf{0.090 \pm 0.156}$	3.324 ± 1.553

Algorithm 1 Learning algorithm of the proposed method

Input: A mini batch of defect training samples (x_{def}, y_{def}), and component training samples $(x_{com}, y_{com}) \in D_{train}$, A mini batch of validation samples $(x_{val}, y'_{def}, y'_{com}) \in D_{val}$, Max number of training epochs E
Output: Feature extractor $f_{\theta 1}$, Encoder network $f_{\theta 2}$, Defect classifier ψ, Projection network ϕ, Gaussian mixture model γ_j where $j \in \{1, 2, \ldots, 23\}$.
Initialize $f_{\theta 1}$, $f_{\theta 2}$, ψ, ϕ, model selection loss $\ell_\omega = \infty$
for $e \leftarrow 1$ to E **do**
    $\ell_{def} \leftarrow$ CrossEntropy($\psi(f_{\theta 2}(f_{\theta 1}(x_{def}))), y_{def}$) // Train the upper branch, i.e. $f_{\theta 1}$, $f_{\theta 2}$, and Classifier ψ.
    Update ψ, $f_{\theta 2}$, $f_{\theta 1}$ using ℓ_{def}
    $\ell_{com} \leftarrow$ MultiSimilarity($\phi(f_{\theta 2}(f_{\theta 1}(x_{com}))), y_{com}$) // Train the lower branch, i.e. $f_{\theta 1}$, $f_{\theta 2}$, and Projector ϕ.
    Update ϕ, $f_{\theta 2}$, $f_{\theta 1}$ using ℓ_{com}
    $\ell_{def'} \leftarrow$ CrossEntropy($\psi(f_{\theta 2}(f_{\theta 1}(x_{val}))), y'_{def}$) // Evaluate the upper branch
    $\ell_{com'} \leftarrow$ MultiSimilarity($\phi(f_{\theta 2}(f_{\theta 1}(x_{val}))), y'_{com}$) // Evaluate the lower branch
    $\ell_{eval} \leftarrow \ell_{def'} + \ell_{com'}$ // Procedure for model selection
    if $\ell_\omega > \ell_{eval}$ **then**
        $\ell_\omega \leftarrow \ell_{eval}$
        saveModel($f_{\theta 1}$, $f_{\theta 2}$, ψ, ϕ)
    end if;
end for;
for $j \leftarrow 1$ to 23 **do**
    $\gamma_j \leftarrow \gamma_j(f_{\theta 2}(f_{\theta 1}(x_{com,j}), y_{com})$ // Gaussian mixture model fitting on $f_{\theta 2}(f_{\theta 1}(x_{com,j})$ guided by y_{com}.
end for

Algorithm 2 Inference Procedure of the Proposed Method

Input: A mini batch of test samples $x_{test} \in D_{test}$
Output: Defect prediction $\hat{y}_{def}$
Require Feature extractor $f_{\theta 1}$, Shared fully connected network $f_{\theta 2}$, Defect classifier ψ, Gaussian mixture model γ_j where $j \in \{1, 2, \ldots, 23\}$, and prediction converter Λ;
$\hat{y}^1_{def} \leftarrow \psi(f_{\theta 2}(f_{\theta 1}(x_{test})))$ // Defect classification.
for $j \leftarrow 1$ to 23 **do**
    $P(\cdot|y_{com} = j) \leftarrow \gamma_j(f_{\theta 2}(f_{\theta 1}(x_{test})))$ // Probabilistic component type prediction.
end for
$\hat{y}^2_{def} \leftarrow \Lambda(\hat{y}^{j=1,2,\ldots,23})$ // Prediction convertion.
if $\hat{y}^1_{def}$ == $\hat{y}^2_{def}$ **then**
    $\hat{y}_{def} \leftarrow \hat{y}^1_{def}, \hat{y}^2_{def}$ // If the predictions match, $\hat{y}_{def}$ will be the same as $\hat{y}^1_{def}$ and $\hat{y}^2_{def}$.
else
    $\hat{y}_{def} \leftarrow$ unknown // If the predictions do not match, $\hat{y}_{def}$ is unknown.
end if
return $\hat{y}_{def}$