Supplementary Information

Contrastive Learning for Data-Efficient Substrate Deoxidation Monitoring in Edge-Side Adaptive Molecular Beam Epitaxy Systems

1. The typical deoxidized and oxidized RHEED data of different substrate

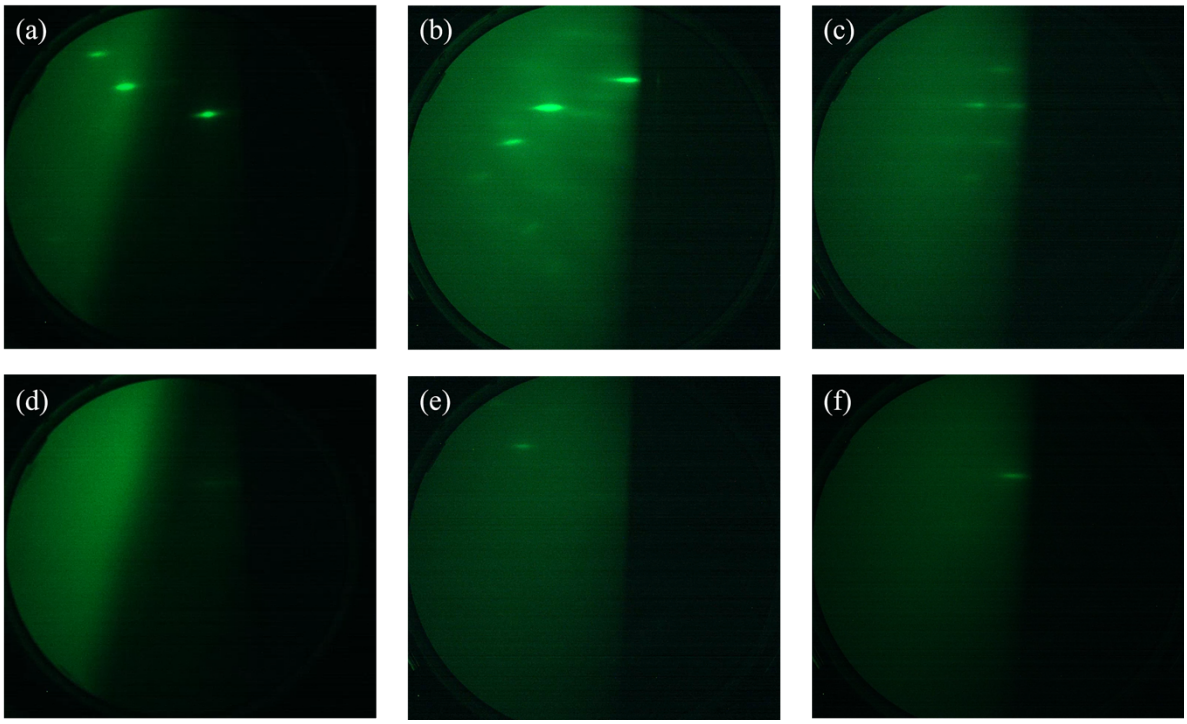

Figure S1. Typical deoxidized RHEED data of (a) GaAs, (b) Ge, and (c) InAs substrate. Typical oxidized RHEED data of (a) GaAs, (b) Ge, and (c) InAs substrate.

2. The image augmentation results

A systematic image augmentation pipeline was implemented to simulate the visual degradation patterns associated with fluorescent screen ageing and varying camera configurations, enhancing the model's generalization ability. As shown in Figure S2a, the preprocessing stage involves converting input images to grayscale and center-cropping them to 900×900 pixels to ensure spatial consistency. Following this, five augmentation transformations—contrast modification, brightness adjustment, rotation, Gaussian blur, and horizontal flipping—are randomly applied according to predefined probabilities. The effects of these transformations are illustrated in Figure S2b-S2f, respectively. This augmentation strategy effectively addresses the domain gap between controlled laboratory conditions and real-world deployment environments, where hardware ageing and different imaging setups introduce systematic visual variations. By simulating these conditions, this approach enables the development of more robust computer vision models that can maintain reliable performance across various operational scenarios and stages of equipment degradation.

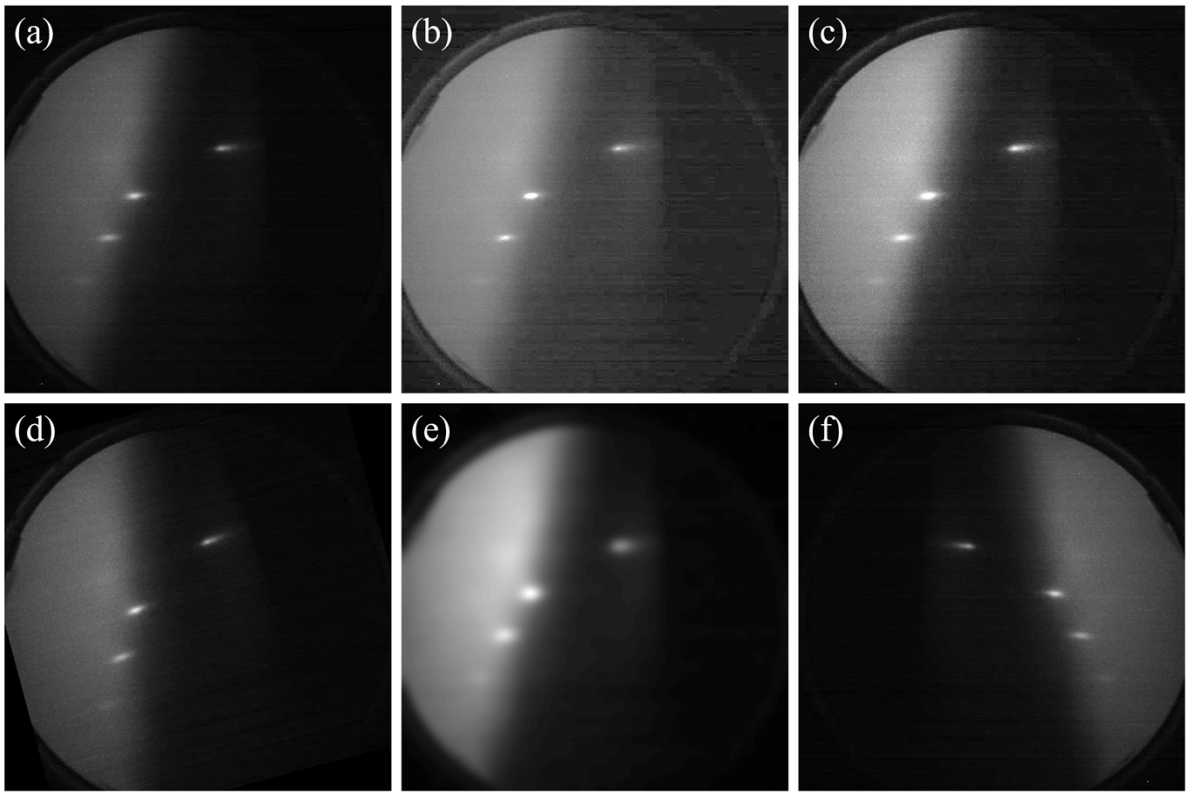

Figure S2. The image augmentation results. (a) The grayscale image cropped to 900×900 pixels at center. The result of randomly applying augmentation transformations: (b) contrast correction, (c) brightness adjustment, (d) rotation, (e) Gaussian blur, and (f) horizontal flip.

3. The model construction environment

All the models were developed and trained on a system running Ubuntu 22.04 LTS with Python version 3.8.19. We utilized PyTorch 2.4.0, which was compiled with CUDA to enable GPU acceleration. The computer had an Intel i7-12700F CPU, 64 GB of RAM, and two NVIDIA RTX A6000 GPUs. The model is deployed on an NVIDIA Jetson AGX Orin 32 GB running JetPack 6.2 (Ubuntu 22.04) with a Python 3.10.12 runtime. Inference is executed via TensorRT 10.3.0 (FP16 precision), and PyCUDA is used to query GPU properties and manage CUDA contexts. Our

approach runs on a multiprocessing architecture processes video acquisition and inference tasks in parallel, ensuring end-to-end latency below 50 ms.

4. The Fine-tuning accuracy and loss data under different sample sizes

Figure S3a illustrates the validation accuracy achieved with different sizes of fine-tuning samples, showing that models trained on larger datasets consistently outperformed those with fewer samples. Specifically, the model fine-tuned with 500 samples achieved the highest accuracy of 98.31%. The 50-sample model followed this at 96.81% and the 5-sample model at 94.52%. In addition to achieving higher peak accuracy, models trained with larger datasets also demonstrated faster convergence toward optimal performance. The horizontal dashed lines in Figure S3a represent the maximum validation accuracy obtained for each dataset size. Notably, even with only 5 samples, the model could still converge, provided there were enough training iterations. The corresponding validation loss curves shown in Figure S3b reflect the same trend, further emphasizing the impact of data quantity on model performance and learning efficiency.

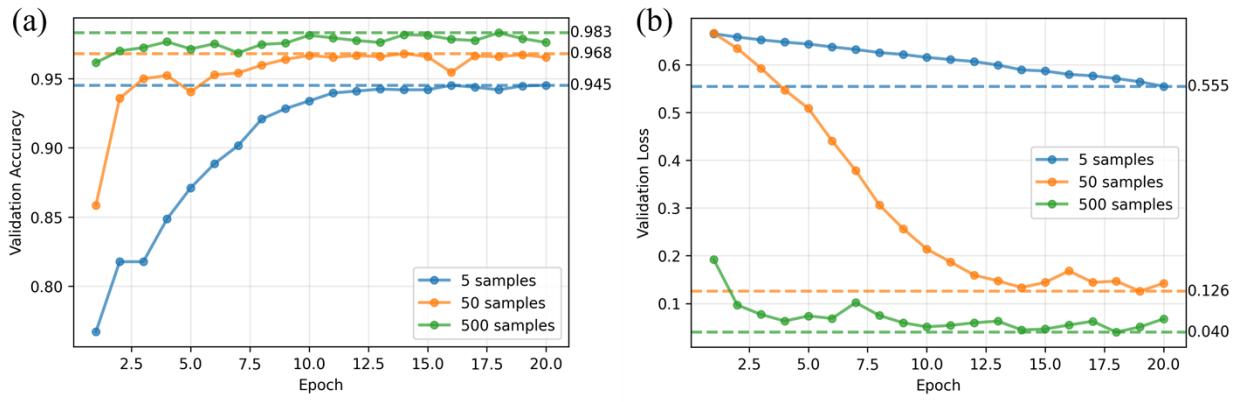

Figure S3. The influence of fine-tuning dataset amounts on (a) validation accuracy and (b) validation loss.

5. The program deployment environment

As illustrated in Figure S4, the camera was connected to the hardware via USB 3.0, allowing it to continuously read images at a resolution of 1920×1080 pixels with a frame rate of 8 frames per second and an exposure time of 100 milliseconds. The temperature controller was connected through USB 2.0 using the Modbus RTU protocol, enabling it to read furnace temperature data continuously.

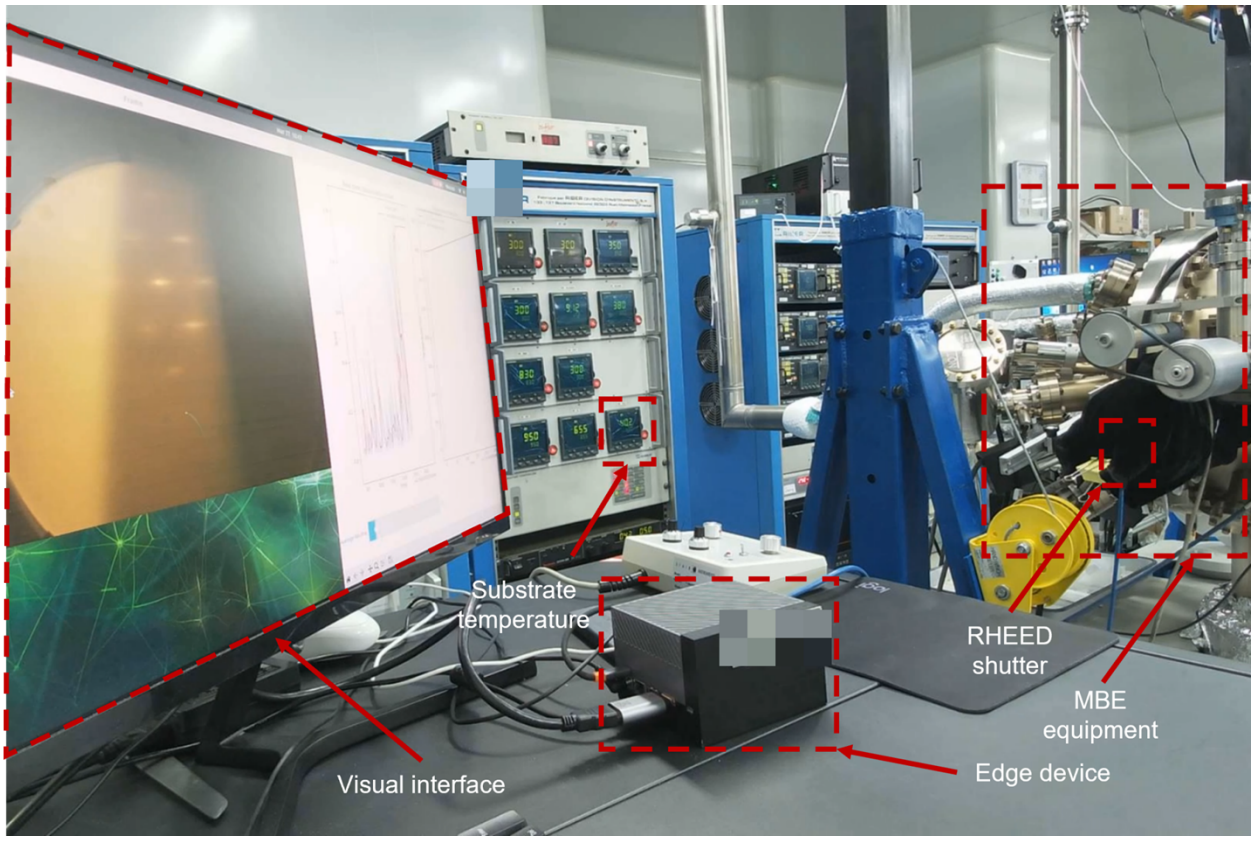

Figure S4. The program deployment environment.

6. The program interfaces

The developed software interface provides real-time monitoring and analysis capabilities for RHEED-based material deoxidation processes. RHEED Image Display Panel on the left displays live RHEED diffraction patterns with real-time classification scoring. The current deoxidation confidence score is prominently shown, indicating the system's assessment of the deoxidation process completion status. Real-time Classification Score Monitor features a dynamic time-series plot tracking the deoxidation classification score evolution. The graph displays both raw score data in blue line and smoothed data in red line to reduce noise and provide clearer trend visualization. The y-axis represents the classification score between 0 to 1.0, while the x-axis shows the temporal progression during the deoxidation process. It should be noted that RHEED images are displayed in BGR format, thus resulting in red-blue channel inversion. This does not affect the algorithm output, as the images are converted to grayscale during the preprocessing stage. Temperature Control and Monitoring integrates substrate temperature monitoring with real-time feedback control. The panel displays current temperature readings, target set points, and temperature evolution over time, enabling correlation between thermal conditions and deoxidation progress. Control Interface at the bottom provides adjustable parameters including an averaging window length for the deoxidation score smoothing filter, allowing users to optimize the balance between responsiveness and noise reduction based on specific experimental requirements.

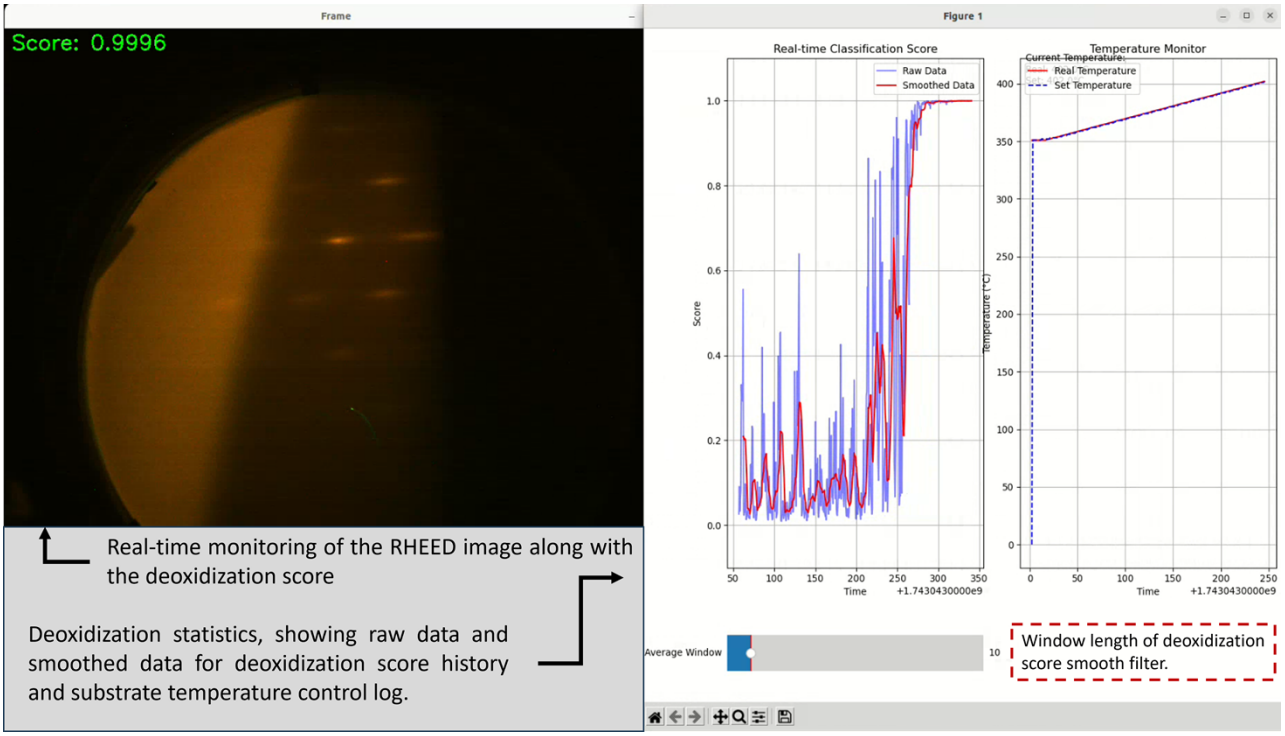

Figure S5. The program interfaces.