Portfolio

AI & Selected Data Engineering

Tianlang Liu

Vision Algorithm Engineer | CATL, Germany

📍 Germany | 💼 Open to relocation

🔗 linkedin.com/in/tianlang-liu

✉️ tlliu_liesmars@outlook.com

TranAD-Based Current Pattern Mining

Tech Stack: PyTorch, TranAD, NumPy

Problem:

Manual inspection failed to **capture** early anomalies in high-frequency current signals from automated equipment.

Solution:

Applied TranAD to **learn normal power curve behavior** and identify deviations indicating stepper signal faults. Focused on **unsupervised detection** using latent representation dynamics.

Results:

Detected pre-failure signals during pilot testing and **improved interpretability** of current anomalies in early-phase diagnostics.

My Contribution:

Built **preprocessing pipeline** for current signal normalization, **implemented TranAD** model **training**, and **visualized output scores for internal validation**.

New Defect Filtering using YOLO + DBSCAN

Tech Stack: YOLO, OpenCV, Scikit-learn (DBSCAN), Python

Problem:

Electrodes **misaligned** in orientation were difficult to detect **using angle thresholds alone**, causing **defects to slip through** automated QA filters, resulting in **extra visual inspection**.

Solution:

Used **YOLO** to detect electrode **masks** and evaluated their alignment via overlap with its ROI. Applied DBSCAN clustering on overlap scores to **dynamically determine threshold** boundaries and filter out outliers.

Results:

Enabled reliable detection of orientation anomalies under varying production configurations. **Reduced human checking workload** and stabilized process quality.

My Contribution:

Modified the orientation filtering algorithm, tuned **DBSCAN parameters** for dynamic thresholding, and implemented the full postprocessing logic in Python.

CV Model Evaluation Platform

Tech Stack: PyQt, Python, Pandas

Problem:

Manual evaluation of classification outputs (e.g., FP/FN tagging) was **inefficient**, **error-prone**, and **lacked standardization**, especially across batches of hundreds of samples.

Solution:

Built a GUI desktop tool for efficient model evaluation with **keyboard-based tagging**, **undo operations**, and **real-time feedback**. Supported TP/FN/FP/TN tagging and automatic **export** of performance reports **to database or .csv**.

Results:

Accelerated validation process **by 85%** in model tuning cycles and enabled consistent human verification across multiple teams by structured output results.

My Contribution:

Designed UI/UX flow, implemented batch logic and deployed the tool internally for users during model iteration cycles.

Real-time Defect Data Pipeline

Tech Stack: PyQt, Pandas, Matplotlib

Problem:

Analyzing log files for error distribution was **slow** and **manual**. Teams **lacked visibility** into patterns such as detection delays, upload latency, or NG type diversity.

Solution:

Developed a GUI tool to **batch-process** production logs files concurrently, **extracting metrics** like critical error counts, detection/upload time intervals, and NG category distributions. Enabled **time-series and histogram-based visualization for QA review**.

Results:

Improved log analysis **efficiency** significantly and **enabled** engineers to **proactively identify** system bottlenecks and shift-related trends across large production datasets.

My Contribution:

Designed the data extraction logic, **implemented batch log loader and GUI layout**, and integrated visualization modules for detailed analysis output.