DeepScan Pro: An Active Learning Pipeline for Intelligent Atomic Microscopy

Md Habibur Rahman · *Purdue University* · rahma103@purdue.edu

Abstract

DeepScan Pro is an interactive software pipeline that transforms passive scanning electron microscopes (SEM/STEM) into intelligent discovery agents. Addressing the inefficiency of blind raster scanning, it utilizes a **Multi-Stage Active Learning** architecture. The pipeline employs a self-supervised vision backbone (RegNet) for feature extraction and an unsupervised Isolation Forest for zero-shot anomaly detection. A "Teacher Mode" allows experts to refine strategies via few-shot Logistic Regression, while a multimodal interface (CLIP) enables natural language search. Simulations demonstrate efficiency gains of up to 90% by prioritizing regions of interest.

1. Motivation & Approach

Goal: Create a "Smart Driver" for microscopes that autonomously prioritizes regions of interest (ROI) to reduce beam damage and acquisition time ($> 10\mu$s/pixel).
Approach: A Hybrid Vision-Control Pipeline:
1. **See:** Pre-trained CNNs (RegNetY) extract texture-invariant semantic vectors from low-res previews.
2. **Think:** An Isolation Forest builds a "surprisal map," identifying rare defects automatically.
3. **Learn:** A Human-in-the-Loop "Teacher Mode" retrains the model instantly via user clicks.
4. **Act:** The system outputs a JSON navigation protocol for the hardware stage.

2. System Overview & Artifacts

The workflow is managed via a modular Streamlit interface comprising four phases:
1. **Ingestion:** Normalization and denoising of high-bit depth images (TIFF/PNG).
2. **Feature Space:** Patch decomposition → CNN Encoding ($\mathbb{R}^{1024}$) → PCA Compression ($\mathbb{R}^{50}$).
3. **Decision Logic:** Three modes—Unsupervised (Isolation Forest), Supervised (Logistic Regression), and Semantic (CLIP text query).
4. **Control:** Thresholding heatmaps to generate Traveling Salesman Paths (TSP) for the stage.

Table 1: Key Data Artifacts

Artifact	Description
`scan_protocol.json`	Machine-readable coordinates $(x, y, priority)$ for stage control.
`scan_map.npy`	Floating-point priority heatmap.
`efficiency.csv`	Simulation metrics (Time vs. Coverage).

3. Methods in Detail

3.1 Neural Backbone We employ 'timm' models (e.g., `regnet_y_400mf`) pre-trained on ImageNet. A patch x is mapped to $z = f_\theta(x)$, capturing semantic texture rather than just pixel intensity.

3.2 Dimensionality Reduction PCA projects features onto principal orthogonal variance directions to reduce computational load: $C = \frac{1}{n}\sum(z_i - \bar{z})(z_i - \bar{z})^T$.

3.3 Unsupervised Anomaly Detection Rare defects are identified via Isolation Forests. The anomaly score $s(x, n)$ relates to the path length $h(x)$ required to isolate a sample:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{1}$$

Shorter paths indicate anomalies (easier to isolate), yielding higher scores.

3.4 Active Learning (Teacher Mode) User clicks provide positive labels (z_{pos}). We train a linear classifier against random background samples ($y = 0$) to estimate the probability of interest:

$$P(y = 1|z) = \frac{1}{1 + e^{-(w^T z + b)}} \tag{2}$$

3.5 Semantic Search (CLIP) Natural language queries are encoded into text embeddings T and compared to image patch embeddings I via cosine similarity: $\text{Sim}(T, I) = \frac{T \cdot I}{\|T\| \|I\|}$.

4. Implementation & Logic

The system requires Python 3.10+, PyTorch, and Scikit-Learn. The active learning logic enables "One-Shot" training:

Listing 1: Active Learning Logic

```
def train_classifier(features, pos_idx):
    """Trains a classifier on-the-fly based on single-click feedback."""
    X_train = [features[pos_idx]] # Positive sample
    y_train = [1]
    # Sample Negatives (Background)
    for idx in random.sample(range(len(features)), 5):
        X_train.append(features[idx]); y_train.append(0)
    # Train & Predict
    clf = LogisticRegression(class_weight='balanced', C=10.0).fit(X_train, y_train)
    return normalize(clf.predict_proba(features)[:, 1])
```

5. Results & Capabilities

- **Efficiency:** Achieves 80% defect capture scanning only 15% of area (Simulated on $SrTiO_3$).
- **Latency:** Inference loop runs in < 200ms on standard CPUs.
- **Capabilities:** Zero-Shot Discovery, Natural Language Search ("Find cracks"), and Automated Quantification (blob counting/area).

6. Conclusion

DeepScan Pro successfully demonstrates that off-the-shelf neural backbones combined with active learning can drive microscopes intelligently. By outputting standard JSON protocols, it bridges the gap between computer vision and physical hardware control.

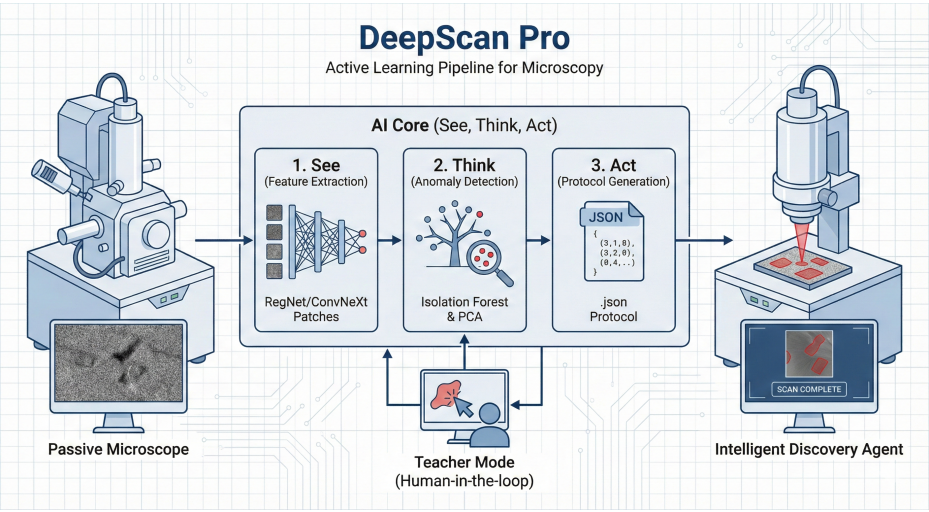

Figure 1: **The DeepScan Pro Active Learning Architecture.** Raw input from a passive microscope is processed through a three-stage AI Core ("See, Think, Act"). The system utilizes neural networks for feature extraction and Isolation Forests for anomaly detection, refined by human input in "Teacher Mode." The final output is a JSON protocol that drives the hardware for targeted, intelligent acquisition.