ScreenShieldAI: Sensitive Data Detection in Screenshots

Building vision-based ML systems to prevent accidental data leakage in enterprise screenshots before they're shared across communication platforms.

Students:

- Hen Golyan
- Afik Suissa
- Aviv Heller

The Problem: Screenshots Are Everywhere

Modern teams constantly share screenshots containing code snippets, dashboards, terminal outputs, CRM data, and chat conversations. These images frequently expose critical secrets that slip past traditional text-based DLP systems.

Our mission is to catch these leaks before screenshots reach Slack channels, support tickets, or documentation repositories.

API Keys & Tokens

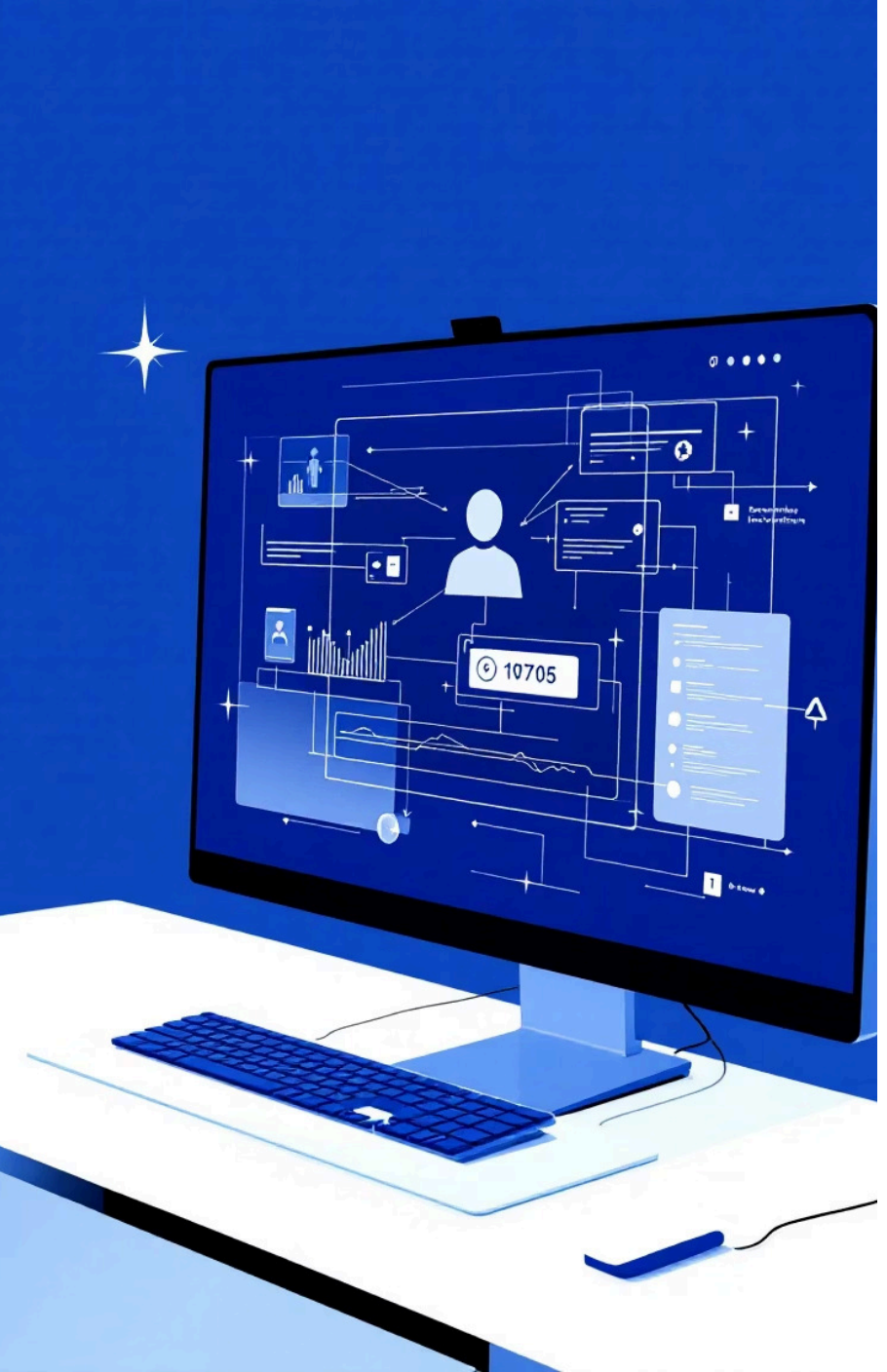
Authentication credentials embedded in code or config files

Passwords & Connection Strings

Database URLs and plaintext authentication

Personal Identifiers

ID numbers, credit cards, phone numbers, email addresses



Dual ML Task Architecture

Binary Classification

Input: Screenshot image

Output: Safe (no leaks) or Leak (sensitive data detected)

Image-level decision for fast screening

Segmentation & Localization

Input: Screenshot flagged as leak

Output: Pixel masks or bounding boxes highlighting exact locations

Pinpoint the API key line, ID column, or token string

Example: VSCode screenshot with `API_KEY = "sk_12345..."` → Classification: **leak**, Segmentation: mask over the credential string

Synthetic Data Generation Strategy

Why Synthetic Data?

Real screenshots containing sensitive information are dangerous to collect, store, and label. We generate training data programmatically with perfect ground truth annotations.

01

Template-Based Rendering

Programmatically create fake IDEs, terminals, dashboards, and forms with synthetic secrets. Auto-generate ground truth masks since we control placement.

02

GenAI Inpainting

Use Stable Diffusion inpainting to replace text regions in realistic screenshots with sensitive patterns while preserving UI authenticity.

03

Diversity Control

Vary fonts, themes (dark/light), languages, layouts, application types, and secret formats to build robust models.

Models & Evaluation Framework

Model Architectures

Classification: CNN and Vision Transformer (ViT) baselines

Segmentation: U-Net, DeepLab, and modern architectures

Comparison: OCR-based pipelines vs. end-to-end vision models

Experimental Approach

Compare baseline models on simple synthetic data against enhanced models trained on rich GenAI-generated datasets

Explore zero/few-shot capabilities with large vision-language models

Success Metrics

Classification: Precision, Recall, F1 for leak detection

Segmentation: IoU and Dice scores for sensitive region localization