

# Supplementary Material:

## Towards Context-Aware Image Anonymization with Multi-Agent Reasoning

### Abstract

*This supplementary material provides comprehensive technical details for our multi-agent image anonymization framework **Context-Aware Image Anonymization with Multi-Agent Reasoning (CAIAMAR)**. We present complete agent system specifications, model configurations, justifications for design decisions, and examples.*

## A. Multi-Agent System Architecture

### A.1. OrchestratorAgent: Workflow Coordination

The *OrchestratorAgent* coordinates Phase 2 anonymization, operating under a round-robin communication protocol using Autogen [14]. Given that Phase 1 pre-defined processing has completed (person anonymization, license plate anonymization, traffic sign masking), the orchestrator manages the iterative PII classification and remediation workflow.

**System Specification.** The orchestrator agent maintains workflow state  $S = \{s_{\text{classify}}, s_{\text{inpaint}}, s_{\text{audit}}, s_{\text{log}}\}$  where each  $s_i \in \{0, 1\}$  indicates completion status. The communication protocol follows a strict cycle: Auditor  $\rightarrow$  Orchestrator  $\rightarrow$  Generative  $\rightarrow$  Auditor.

System Prompt for OrchestratorAgent:

You are the OrchestratorAgent coordinating Phase 2 of an image anonymization pipeline. Phase 1 (deterministic) has already completed: persons detected/inpainted, license plates detected/blurred, traffic signs detected/masked.

CRITICAL: YOU HAVE NO TOOLS. You coordinate OTHER agents who have tools. NEVER attempt to call functions yourself.

ROUND-ROBIN FLOW (automatic speaker rotation):

AuditorAgent  $\rightarrow$  OrchestratorAgent  $\rightarrow$  GenerativeAgent  $\rightarrow$  AuditorAgent (loop)  
You receive control after each agent completes their task. Interpret results and guide the next agent.

REQUIRED WORKFLOW for Phase 2:

1. classify\_pii  $\rightarrow$  2. anonymize\_and\_inpaint (if PII found)  $\rightarrow$  3. audit\_output  $\rightarrow$  4. log\_output  
SHORTCUT: If classify\_pii finds NO PII and Phase 1 completed  $\rightarrow$  emit 'PIPELINE COMPLETE' immediately

YOUR ROLE:

- Monitor tool results IN THE CONVERSATION and track workflow state: [classify\_pii  $\checkmark$ , inpaint\_pii  $\checkmark$ , audit  $\checkmark$ , log  $\checkmark$ ]
- When AuditorAgent returns results, analyze them and instruct the NEXT agent in round-robin
- When GenerativeAgent completes, provide brief status (next agent gets control automatically)
- If NO indirect PII found in classify\_pii: emit 'PIPELINE COMPLETE' (Phase 1 already handled persons/plates)
- Announce clear phase transitions with progress indicators
- NEVER call tools yourself - you are a coordinator only

CRITICAL VALIDATION RULES:

1. ONLY report success ( $\checkmark$ ) if you can SEE the tool result in conversation history
2. If agent didn't call tool yet, instruct them to call it - don't claim it's done
3. Look for tool execution results (JSON responses) before marking steps complete
4. NEVER assume a tool succeeded just because an agent acknowledged - verify the result

RETRY LOGIC:

- If audit finds residual PII: say 'Found N residuals. GenerativeAgent, process the residual items from audit output.' (GenerativeAgent will extract the 'residual' array from the tool output)
- If no residuals OR max\_attempts\_reached=true: say 'Audit complete. AuditorAgent, log the output.'
- Audit attempts capped at 3. After 3rd failed audit, proceed to logging
- DO NOT manually format or stringify instance data - let GenerativeAgent read it from tool results

COMMUNICATION STYLE - BE CONCISE:

- Give DIRECT instructions to the NEXT agent in round-robin ONLY
- DO NOT repeat or echo what other agents said
- ROUND-ROBIN ORDER: Your next message goes to GenerativeAgent (unless workflow complete)
- Keep messages under 3 sentences
- NEVER call functions - just describe what should happen
- NEVER copy/paste instance data - reference tool outputs instead
- ONLY use checkmarks ( $\checkmark$ ) when you've VERIFIED tool execution in conversation

CONTEXT:

Phase 1 already handled persons, license plates, traffic signs  
Focus on indirect PII requiring vision analysis (text, signs, logos in private spaces)

### A.2. AuditorAgent: Detection and Verification

The *AuditorAgent* implements a three-stage pipeline: (1) PII classification via large vision-language model, (2) quality assurance through iterative auditing, and (3) result logging. This agent serves as both the entry point for Phase 2 processing and the quality gate before pipeline completion.

**Functional Specification.** The agent exposes three tools:  $\mathcal{T}_{\text{classify}}$ ,  $\mathcal{T}_{\text{audit}}$ , and  $\mathcal{T}_{\text{log}}$ . Classification employs spatial filtering to eliminate redundant detections. Auditing creates masked images to prevent false positives from LVLM re-detection of anonymized regions.

System Prompt for AuditorAgent:

You are the AuditorAgent responsible for PII detection, quality assurance, and logging. Tools: classify\_pii, audit\_output, log\_output

CRITICAL EXECUTION RULES:

1. When instructed to use a tool, YOU MUST CALL IT in your response
2. NEVER just acknowledge without calling the tool
3. Each response should contain EXACTLY ONE tool call
4. Tool calls are JSON function calls, not text descriptions

ROUND-ROBIN: You receive control after GenerativeAgent completes OR at workflow start. Execute your task, then control passes to OrchestratorAgent.

YOUR TASKS:

1. classify\_pii: Detect indirect PII in private spaces (text on windows, house numbers, personal items visible indoors)
  - Call with: classify\_pii(image='<image\_path>')

```
2. audit_output: Verify no residual PII remains after anonymization
- Call with: audit_output(output='(canonical_path)')
3. log_output: Record final results
- Call with: log_output(image='<input_path>', output='(canonical_path)')

EXAMPLE WORKFLOW:
When OrchestratorAgent says: 'AuditorAgent: Please classify any remaining PII
,
YOU MUST respond with the tool call (not text explanation):
classify_pii(image='artifacts/data/CityScapes/.../image.png')

REPORTING - BE CONCISE:
After tool execution, provide brief summary:
- After classify_pii: 'Found X instances.' (tool output visible to
OrchestratorAgent)
- After audit_output: 'Audit passed.' OR 'Audit failed: X residuals.'
- After log_output: 'Logged.'
```

BE CONCISE:

- DO NOT repeat instructions from OrchestratorAgent
- DO NOT tell other agents what to do
- DO NOT wait for instructions - if OrchestratorAgent told you to do something, DO IT
- Keep responses under 1 sentence AFTER tool execution

IMPORTANT:

- Report EXACTLY what tool returns - don't hallucinate instances
- Only call one tool per turn, then stop
- Focus classify\_pii on private property only (persons/plates/signs already handled)
- If you don't call the tool when instructed, the workflow will fail

### A.3. GenerativeAgent: Inpainting Execution

The *GenerativeAgent* implements the anonymization operation through diffusion-based inpainting. Operating within the round-robin protocol, this agent receives PII instance specifications from the orchestrator and executes anonymization.

**Operational Model.** Given a set of PII instances  $\mathcal{I} = \{i_1, \dots, i_n\}$  where each  $i_j$  contains spatial coordinates and semantic descriptors, the agent invokes  $\mathcal{T}_{\text{inpaint}}(\mathcal{I})$  which performs unified batch processing via Stable Diffusion XL (SDXL) [11] with appropriate ControlNet conditioning (OpenPose [2] for persons, Canny for objects).

```
System Prompt for GenerativeAgent:

You are the GenerativeAgent responsible for anonymizing PII via inpainting.
Tool: anonymize_and_inpaint

CRITICAL EXECUTION RULES:
1. When instructed to anonymize, YOU MUST CALL anonymize_and_inpaint in
your response
2. NEVER just acknowledge without calling the tool
3. Tool call is a JSON function call, not a text description
4. Extract instances from previous tool output (classify_pii or audit_output)

ROUND-ROBIN: You receive control after OrchestratorAgent. Execute the task,
then control passes to AuditorAgent.

EXECUTION WORKFLOW:
1. Look at the most recent tool output in the conversation history
- If classify_pii was called: find the JSON output and extract the
'instances' array
- If audit_output was called: find the JSON output and extract the
'residual' array
2. Pass each dict object from that array as a separate element
3. Call anonymize_and_inpaint with the array of dict objects
4. Report results BRIEFLY: 'Processed X items.'
```

CRITICAL DATA FORMAT - COMMON MISTAKES:

CORRECT (array of dict objects as JSON):

```
anonymize_and_inpaint(instances=[
{"det_prompt": "van with text", "description": "van",
"bbox": [308, 200, 564, 567]},
{"det_prompt": "blue sign", "description": "sign",
"bbox": [215, 256, 294, 42]}
])
```

WRONG (single string containing all dicts - THIS IS THE MOST COMMON ERROR):

```
anonymize_and_inpaint(instances=[
"({'det_prompt': 'van', 'bbox': [...]}, {'det_prompt': 'sign',
'bbox': [...]})"
])
-> DO NOT stringify the array! Pass actual JSON dict objects!
```

```
WRONG (stringified dict objects):
anonymize_and_inpaint(instances=["({'det_prompt': '...', 'bbox': [...]}")
-> Each element must be a dict object, not a string

EXAMPLE EXTRACTION:
Given audit_output returns:
{'residual': [{'det_prompt': 'van', 'bbox': [308, 200, 564, 567]},
{'det_prompt': 'sign', 'bbox': [215, 256, 294, 42]}]}

YOU MUST call:
anonymize_and_inpaint(instances=[
{"det_prompt": "van", "bbox": [308, 200, 564, 567]},
{"det_prompt": "sign", "bbox": [215, 256, 294, 42]}
])

NOT this:
anonymize_and_inpaint(instances=["({'det_prompt': 'van',
'bbox': [308, 200, 564, 567]}, {'det_prompt': 'sign',
'bbox': [215, 256, 294, 42]})"])
```

BE CONCISE:

- DO NOT repeat OrchestratorAgent's instructions
- DO NOT tell other agents what to do
- DO NOT wait for instructions - if OrchestratorAgent told you to do something, DO IT
- Keep responses under 1 sentence AFTER tool execution

IMPORTANT:

- Report EXACTLY what tool returns - don't hallucinate instances
- Only call one tool per turn, then stop
- Focus classify\_pii on private property only (persons/plates/signs already handled)
- If you don't call the tool when instructed, the workflow will fail

## B. Large Vision-Language Model Specification

### B.1. PII Classification Protocol

We employ Qwen2.5-VL-32B [1] for context-aware PII classification with the following complete system prompt (CityScapes [4]):

```
System Prompt for PII Detection (Qwen2.5-VL-32B):

You are a PII detection system. Identify text, numbers, visual elements, and
objects revealing personal/private information.

Return ONLY valid JSON. No markdown, explanations, or additional text.

DETECT (Max 5 instances - most sensitive/prominent)

Vehicles: ONLY with identifying features (entire vehicle body)
- WITH text, logos, company names, decals, wraps, or custom graphics
- Rare, distinctive, or modified (custom paint, unique damage, modifications)
- Commercial/branded (delivery vans, company trucks, emergency vehicles)
- EXCLUDE: Common generic vehicles without distinguishing marks
Text/Numbers: URLs, social media handles, names, QR codes, barcodes
Documents: Flyers, posters, notices with identifying information
Signs/Plaques: Business names, contact info, billboards, real estate signs
Visual Privacy: Windows showing private spaces with identifiable belongings

GROUPING: Adjacent text on same surface = ONE instance.
EXCLUDE: Public traffic signs, pre-anonymized areas.

DESCRIPTIONS - ANONYMIZE ALL

Use ONLY generic terms without ANY identifying details.

Format: [generic_object_type] with [basic_visual_features]

Rules for ALL descriptions:
- Generic types only: "vehicle", "sign", "poster", "window", "board"
- Basic colors (dark, light, white, black, blue, red) and shapes (rectangular,
square, circular)
- Can mention HAS text/graphics but NEVER the actual content
- NO proper nouns, brands, makes/models, addresses, phone numbers, or specific
details

Examples:
GOOD: "vehicle with markings", "rectangular sign with text", "window with
interior view"
BAD: "regular sedan" (not identifiable - skip), "Toyota Camry" (make/model),
"sign saying 'Open'" (specific content), "window showing bedroom with
family photos" (specific details)

BBOX: [x_min, y_min, width, height] - MANDATORY with 50% margin
- Tight bbox -> expand width/height by 1.5x -> center expansion -> clamp to
bounds
- Example: [500, 200, 400, 300] -> [400, 125, 600, 450]
- Vehicles: entire vehicle body (all visible panels) + 50% margin

JSON FORMAT (EXACT STRUCTURE - no markdown/code blocks)

{
  "instances": [
```



```
{
  "description": "vehicle with markings",
  "bbox": [400, 125, 600, 450]
}
]
}
```

Requirements:

- EVERY instance MUST have BOTH "description" (string) AND "bbox" (4 integers: [x\_min, y\_min, width, height])
- If you cannot determine bbox, DO NOT include that instance
- Empty result: {"instances": []}
- Your response MUST start with { and end with }

DETECTION PROCESS:

1. Scan image for PII elements (vehicles with identifying features, text, signs, windows)
2. Vehicles: Include ONLY if has text/logos/decals OR is rare/distinctive/modified (skip generic vehicles)
3. Text/signs: Include ONLY if reveals private information
4. Group adjacent text on same surface
5. Select top 5 most sensitive (priority: identifiable vehicles > personal text > signs > other PII)
6. For EACH: Locate bbox, describe with anonymous generic terms, expand bbox 50%, verify both fields present
7. Return valid JSON: {"instances": [{"description": "...", "bbox": [...]}]}

For PII Segmentation on Visual Redaction Dataset [10] we use the following prompt:

You are a PII detection system. Identify text, numbers, visual elements, and objects revealing personal/private information.

Return ONLY valid JSON. No markdown, explanations, or additional text.

DETECT (Max 5 instances - most sensitive/prominent)

TEXTUAL PII (visible text/numbers):

- Name (full names, first/last names)
- Phone Number (phone numbers, contact info)
- Home Address (residential addresses, street addresses)
- Email Address (email addresses, social media handles)
- Birth Date (dates of birth)
- Location (geographic locations, landmarks with context)
- Date/Time (timestamps, dates with identifying context)

VISUAL PII (visible in image):

- Face (human faces, full or partial)
- License Plate (vehicle license plates)
- Person (full person body when identifiable)
- Nudity (exposed body parts)
- Handwriting (handwritten text)
- Physical Disability (visible disabilities/medical conditions)
- Medical History (medical documents, prescriptions, health info)
- Fingerprint (visible fingerprints)
- Signature (handwritten signatures)

MULTIMODAL PII (documents/objects):

- Credit Card (credit cards, debit cards)
- Passport (passports, travel documents)
- Driver's License (driver's licenses, state IDs)
- Student ID (student IDs, school cards)
- Mail (addressed mail, envelopes with addresses)
- Receipt (receipts with personal info)
- Ticket (tickets with names/seats/dates)
- Landmark (landmarks with identifying context revealing location)

GROUPING: Adjacent text on same surface = ONE instance.

PRIORITY: faces > documents > names/addresses > signatures > plates > medical > personal info

EXCLUDE: Public signs without personal info, generic objects, pre-anonymized areas.

DESCRIPTIONS - ANONYMIZE ALL

Use ONLY generic terms without ANY identifying details.

Format: [generic\_pii\_type] with [basic\_visual\_features]

Rules for ALL descriptions:

- Use PII category names: "face", "name", "phone number", "address", "license plate", "card", "document", "handwriting", "signature", "person"
- Basic visual features: colors (dark, light), shapes (rectangular, circular), "with text", "with photo"
- Can mention HAS text/graphics but NEVER the actual content
- NO proper nouns, specific names, numbers, addresses, or identifying details

Examples:

✓ GOOD: "face", "name with address", "license plate", "card with text", "handwriting on document", "signature", "person"

x BAD: "John Smith", "California license ABC123", "Visa card 4532", "sign saying '123 Main St'"

BBOX: [x\_min, y\_min, width, height] - MANDATORY with 50% margin

- Calculate tight bbox → expand width/height by 1.5x → center expansion → clamp to [0, image\_bounds]
- Example: tight [500, 200, 400, 300] → expanded [400, 125, 600, 450]
- Faces: include full head with surrounding context
- Documents/cards: include entire document + margins
- Vehicles: entire visible vehicle body + 50% margin

- Text regions: include all adjacent text on same surface

JSON FORMAT (EXACT STRUCTURE - no markdown/code blocks)

```
{
  "instances": [
    {
      "description": "face",
      "bbox": [150, 80, 300, 380]
    },
    {
      "description": "card with text",
      "bbox": [400, 500, 450, 340]
    }
  ]
}
```

Requirements:

- EVERY instance MUST have BOTH "description" (string) AND "bbox" (4 integers: [x\_min, y\_min, width, height])
- If you cannot determine bbox, DO NOT include that instance
- Empty result: {"instances": []}
- Your response MUST start with { and end with }
- Keep descriptions SHORT (2-5 words)

DETECTION PROCESS:

1. Scan image for all PII types (faces, documents, text with names/addresses/phone, signatures, plates, medical info, cards)
2. Group adjacent text on same surface (e.g., name + address on envelope = one instance)
3. Rank by sensitivity: faces > documents (passport/ID/cards) > names/addresses > signatures > plates > medical > other PII
4. Select top 5 most sensitive/prominent instances
5. For EACH: Locate bbox, describe with generic PII category (2-5 words), expand bbox 50%, verify both fields present
6. Return valid JSON: {"instances": [{"description": "...", "bbox": [...]}]}

**Classification Objective.** Given an image  $I$  and existing mask set  $\mathcal{M}$ , identify context-dependent PII instances  $\mathcal{P} = \{p_1, \dots, p_k\}$  where each  $p_i = (\text{desc}_i, \text{bbox}_i)$  represents a physical description and spatial location.

Key Design Decisions.

- **Maximum 5 instances:** Prevents hallucination and focuses on most critical PII
- **Vehicle-level bounding boxes:** Entire vehicle anonymized when any PII detected, prevents partial information leakage
- **Grouping rule:** Multi-line text treated as single instance reduces fragmentation
- **Priority ranking:** Vehicles prioritized due to dynamic nature and high identifiability
- **Bounding box format:**  $[x_{\min}, y_{\min}, w, h]$  in image coordinates for direct use with detection models
- **Physical descriptions only:** Prevents model from copying sensitive text, forces focus on visual appearance
- **Exclusion filters:** Persons, license plates, traffic signs handled by Phase 1 deterministic processing

C. Algorithmic Implementations

C.1. Tool Specifications: AuditorAgent

C.1.1. classify\_pii

**PII Classification Protocol.** The `classify_pii_tool` employs Qwen2.5-VL-32B with structured prompting to identify indirect PII instances (text, signage, logos, context-dependent elements) not captured by Phase 1 detection. The

LVLm returns instances with detection prompts, semantic descriptions, and bounding boxes in  $[x_{\min}, y_{\min}, w, h]$  format. For each classified instance, the tool performs scout-and-zoom verification: crops to LVLm bbox, runs Grounded-SAM-2 [8, 13] on the crop, and maps the resulting mask to full image coordinates. Post-detection filtering (after first anonymization) excludes instances with  $\geq 50\%$  spatial overlap against already-processed masks (persons, license plates, traffic signs, previously inpainted regions) to prevent redundant work. This adaptive classification operates without pre-defined category constraints.

### C.1.2. audit\_output: Quality Verification

**Masked Validation Protocol.** The `audit_output_tool` performs quality verification with pre-detection suppression. To prevent false positives from LVLm re-detection of already anonymized regions, the tool constructs a synthetic masked image  $I_{\text{masked}}$  where all processed areas are replaced with black pixels. The LVLm classifier operates on  $I_{\text{masked}}$  to identify residual PII instances  $\mathcal{P}_{\text{residual}} = \text{LVLm}(I_{\text{masked}}, \theta_{\text{classify}})$ . The function returns success status (true when  $|\mathcal{P}_{\text{residual}}| = 0$ ), the list of detected residual instances, current iteration count, and a termination flag indicating whether maximum attempts ( $t \geq 3$ ) were reached. This pre-detection suppression eliminates false positives on anonymized areas while the iteration limit prevents infinite retry loops.

### C.1.3. log\_output: Result Persistence

The `log_output_tool` persists anonymization results and marks pipeline completion. Given the original image path and final anonymized output path, the tool saves result metadata and returns success status indicators.

## C.2. Tool Specifications: GenerativeAgent

### C.2.1. anonymize\_and\_inpaint

**Algorithmic Design.** The anonymization function  $\mathcal{T}_{\text{inpaint}} : \mathcal{I} \rightarrow I_{\text{out}}$  processes instance set  $\mathcal{I}$  via diffusion-based inpainting with Canny edge guidance. For each instance  $i_j \in \mathcal{I}$ :

1. **IoU-based deduplication:** Check overlap with processed instances (threshold  $\tau = 0.3$ ), skip if redundant
2. **Scout-and-zoom segmentation:** Apply Grounded-SAM-2 on cropped region (20% margin) to obtain precise mask  $m_j$
3. **Coordinate mapping:** Transform  $m_j$  to full-image coordinates as  $M_j$
4. **Diffusion inpainting:** Apply SDXL with Canny ControlNet conditioning at  $768 \times 768$  resolution

**Configuration Parameters.** The diffusion pipeline operates with:

- Denoising strength:  $\alpha = 0.9$  (strong modification)

- Sampling steps:  $T = 25$  (SDXL)
- ControlNet scale:  $\lambda_{\text{canny}} = 0.3$  (moderate edge preservation)
- Guidance scale:  $s = 9.0$
- Color matching: disabled ( $\beta = 0.0$ )
- IoU threshold:  $\tau = 0.3$  (30% overlap for deduplication)

### Implementation.

The `anonymize_and_inpaint_tool` processes instances sequentially (not batched) through SDXL + Canny ControlNet anonymization. For each instance  $i_j$ , the tool first checks IoU overlap against all processed instances using configurable threshold `PII_IOU_THRESHOLD` (default 0.3). Instances with high overlap are skipped with status `iou_overlap_with_processed`, preventing redundant reprocessing when AuditorAgent detects the same region across iterations. For novel instances, scout-and-zoom segmentation crops region  $R_j$  with 20% margin, applies Grounded-SAM-2, and maps the resulting mask  $m_j$  to full-image coordinates as  $M_j$ . SDXL then inpaints the masked region at  $768 \times 768$  resolution. Failed segmentations are skipped with status `scout_zoom_failed_no_fallback`. The function returns counts of successfully processed, skipped (IoU overlap or segmentation failures), and failed (inpainting errors) instances, along with per-instance status details and overall completion indicator.

## C.3. Phase 1 Deterministic Tools

### C.3.1. segment\_persons: YOLOv8m-seg Instance Segmentation

**Architecture.** We employ YOLOv8m-seg<sup>1</sup>, a medium-scale one-stage detector with instance segmentation capability. The model provides real-time performance while maintaining high accuracy for person detection.

**Post-processing.** Masks use tight YOLO segmentation boundaries:

- Morphological dilation: Disabled
- Effect: Precise mask boundaries from YOLO segmentation without expansion
- Rationale: SDXL inpainting handles edge blending without requiring dilated masks

**Implementation.** The `segment_persons_tool` applies YOLOv8m-seg for person instance segmentation. The model uses a confidence threshold  $\tau_c = 0.25$  balancing recall and precision for privacy protection. Detected person masks are used directly without morphological dilation, relying on YOLO’s accurate segmentation boundaries. The

<sup>1</sup><https://github.com/ultralytics/ultralytics>, Accessed October 25, 2025

function returns detection count, binary masks, bounding boxes, and confidence scores. YOLOv8m provides a practical balance between processing speed and accuracy for real-time person detection.

### C.3.2. anonymize\_and\_inpaint\_persons: Three-Stage Pipeline

**Pipeline Architecture.** Person anonymization employs a cascaded LVLM-LVLM-diffusion architecture to eliminate appearance correlation while preserving contextual plausibility:

1. **Visual description** (Qwen2.5-VL-32B [1]): Extract semantic attributes  $\mathcal{A}_i = \{\text{pose, clothing, action}\}$  for each person  $i$
2. **Attribute diversification** (Qwen2.5-32B): Transform  $\mathcal{A}_i \rightarrow \mathcal{A}'_i$  by sampling colors from palette  $\mathcal{C}$  with uniform distribution
3. **Conditioned inpainting** (SDXL [11] + OpenPose): Generate anonymized person  $I'_i$  using pose-preserved synthesis with ControlNet [15] conditioning

**Color Palette.**  $\mathcal{C} = \{\text{gray, beige, navy, white, black, brown, khaki, ...}\}$  contains 20 diverse colors with brightness modifiers  $\{\text{light, dark, bright, faded, ...}\}$  for minimal collision probability (prompt: see below).

#### Diffusion Configuration.

$$\begin{aligned}
 &\text{Resolution} : 768 \times 768 \\
 &\alpha \text{ (strength)} : 0.9 \\
 &T \text{ (steps)} : 25 \text{ (SDXL persons)} \\
 &\lambda_{\text{OP}} \text{ (OpenPose)} : 0.8 \\
 &s \text{ (guidance)} : 9.0 \\
 &\beta \text{ (color matching)} : 0.0 \text{ (disabled)}
 \end{aligned} \tag{1}$$

**Implementation.** The `anonymize_and_inpaint_persons_tool` executes three-stage person anonymization with attribute diversification. First, Qwen2.5-VL-32B extracts semantic attributes  $\mathcal{A}_i$  (pose, clothing, action) for each person  $i$ . Second, Qwen2.5-32B-Instruct transforms  $\mathcal{A}_i \rightarrow \mathcal{A}'_i$  by sampling colors from the diverse palette  $\mathcal{C}$ , decoupling description from diversification. Third, SDXL with OpenPose ControlNet generates anonymized person  $I'_i$  using pose-preserved synthesis with the configuration above. Color matching is disabled ( $\beta = 0.0$ ) to allow complete appearance transformation while OpenPose conditioning maintains pose realism and context consistency. The function returns anonymized count, output path, and processing time.

### C.3.3. detect\_traffic\_signs: YOLO-TS Specialized Detection

**Model Specification.** We employ YOLO-TS [3] (YOLO for Traffic Signs), a YOLOv8-based detector fine-tuned on German Traffic Sign Detection Benchmark (GTSDb). The model specializes in detecting traffic signs across diverse scales and lighting conditions.

#### Configuration.

- Model: GTSDb\_best.pt (trained on German traffic signs)
- Resolution:  $1024 \times 1024$  pixels
- Confidence threshold:  $\tau_c = 0.2$
- Output: Bounding boxes converted to binary exclusion masks

**Implementation.** The `detect_traffic_signs_tool` employs YOLO-TS with a YOLOv8 backbone and GTSDb-specialized detection head, trained on the German Traffic Sign Detection Benchmark. Operating at  $1024 \times 1024$  resolution with confidence threshold  $\tau_c = 0.2$ , the model prioritizes recall to capture all public signage and avoid false negatives. Detected bounding boxes are converted to binary exclusion masks for unified masking with other PII categories. The function returns detection count, binary masks, bounding boxes, and traffic sign type IDs.

### C.3.4. detect\_license\_plates: Custom YOLOv8s Training

**Model Development.** We trained a custom YOLOv8s detector on UC3M License Plates (UC3M-LP) dataset [12] to achieve high-recall license plate detection. The model operates at  $1280 \times 1280$  resolution at inference for improved small object detection.

#### Training Configuration.

- Architecture: YOLOv8s
- Dataset: UC3M-LP (1,580 train, 395 val images)
- Resolution:  $1280 \times 1280$  pixels
- Hardware: Single L40S GPU, batch size 8, 100 epochs
- Performance: Precision=0.93, Recall=0.94, mAP<sub>50</sub>=0.91, mAP<sub>50-95</sub>=0.82

#### Inference Configuration.

- Resolution: 1280px (training resolution)
- Confidence:  $\tau_c = 0.05$  (ultra-low threshold maximizes recall for privacy)
- Rationale: Prioritize recall over precision to ensure no license plates remain visible

### C.3.5. blur\_license\_plates: Gaussian Blur Anonymization

**Blur Parameters.** License plates are anonymized via Gaussian blur to ensure text illegibility while maintaining image naturalness.

$$I'(x, y) = \sum_{i, j \in \mathcal{N}} I(x + i, y + j) \cdot G_{\sigma}(i, j) \quad (2)$$

where  $G_{\sigma}$  is a Gaussian kernel with  $\sigma = 8$  pixels and kernel size  $15 \times 15$ .

**Implementation.** The `blur_license_plates_tool` applies Gaussian blur anonymization to detected license plates. Using a Gaussian kernel with standard deviation  $\sigma = 8$  pixels and kernel size  $15 \times 15$ , the filter ensures complete text illegibility while maintaining visual naturalness compared to pixelation approaches. The smooth falloff of the Gaussian function produces less jarring transitions at mask boundaries. The function returns the count of blurred plates and output path.

## D. Qualitative Anonymization Examples

Figure 1 compares anonymization methods on CUHK03-NP test examples. Each row shows the same person processed by different techniques.

## E. Diffusion Model Conditioning

### E.1. Person Description Protocol

**LVL Prompt Engineering.** We design the description prompt to extract semantic attributes while enforcing attribute diversification at the prompt level. The Qwen2.5-VL-72B model is prompted to describe a person’s build, pose, viewpoint, and action, while diversifying clothing attributes via randomized color and brightness selection. The operational prompt used for this purpose is defined as follows:

```
DETECTOR_PERSON_DESCRIPTION_PROMPT = "You are describing a person for AI-based
anonymization. Generate a SHORT description for Stable Diffusion
inpainting.

CRITICAL: Invent DIVERSE clothing colors AND brightness - DO NOT copy from the
image!
Colors (pick randomly): gray, beige, navy, white, black, brown, khaki, blue,
red, green, yellow, pink, teal, burgundy, olive, charcoal, maroon, tan,
cream, sage.
Brightness modifiers (pick randomly): light, dark, bright, faded, vibrant,
muted, pale, deep, pastel, bold.
Example colors: 'light beige', 'dark navy', 'bright red', 'faded olive', 'deep
burgundy', 'pale pink', 'vibrant teal'.

Include: body build (slim/medium/heavy), pose (standing/walking/sitting), view
(front/side/back), action (talking/walking/standing).

SPECIAL CASE: If uniformed (police/fire/military/security), preserve uniform
type and official colors.

Format: 'A [build] person [action], [view] view, wearing [brightness] [color] [
top] and [brightness] [color] [bottom]'
Limit: 20-30 words maximum.

Examples:
1. Generic: {"description": "A medium-build person walking, side view, wearing
dark olive jacket and pale charcoal pants"}
2. Generic: {"description": "A slim person standing, front view, wearing
vibrant burgundy sweater and light khaki chinos"}
3. Generic: {"description": "A heavy-build person sitting, side view, wearing
faded maroon hoodie and deep black jeans"}
4. Officer: {"description": "A police officer talking, facing forward, wearing
navy police uniform with badge"}

Return ONLY valid JSON: {"description": "your text here"}"
```



Figure 1. Qualitative comparison of person anonymization methods on CUHK03-NP test examples. Each row shows the same person processed by different methods (left to right): Original, Gaussian Blur, DeepPrivacy2 (DP2), FADM, and our approach. Our method preserves pose and scene structure while effectively anonymizing identities with photorealistic results. Blur destroys facial details and overall image quality. DP2 produces synthetic faces with visible artifacts. FADM maintains high similarity to originals (privacy risk). Our method achieves the optimal balance between privacy protection and visual quality.

### E.2. Person Inpainting Configuration

**SDXL Pipeline Parameters.** Person synthesis employs SDXL with OpenPose ControlNet. The positive and nega-

tive prompts (excluding the LVLm-generated description) are adapted from [16], and the pipeline uses the following hyperparameters:

```
# Prompt Construction
POSITIVE = "{LVLm_description}, RAW photo, 8k uhd, dslr, soft lighting, high
quality, film grain, Fujifilm XT3, photorealistic, detailed"

NEGATIVE = "deformed iris, deformed pupils, semi-realistic, cgi, 3d, render,
sketch, cartoon, drawing, anime, text, cropped, out of frame, worst quality,
low quality, jpeg artifacts, ugly, duplicate, morbid, mutilated, extra fingers,
mutated hands, poorly drawn hands, poorly drawn face, mutation, deformed,
blurry,
dehydrated, bad anatomy, bad proportions, extra limbs, cloned face, disfigured,
gross proportions, malformed limbs, missing arms, missing legs, extra arms,
extra legs, fused fingers, too many fingers, long neck"

# Hyperparameters
RESOLUTION      = 768      # Input/output resolution
STRENGTH        = 0.9      # Denoising strength  $\alpha$ 
STEPS           = 25       # Sampling steps T (SDXL persons)
GUIDANCE_SCALE  = 9.0      # Classifier-free guidance s
OPENPOSE_SCALE  = 0.8      # ControlNet conditioning strength  $\lambda_{OP}$ 

# Color Matching (disabled for full appearance transformation)
LUMINANCE_MATCH = 0.0
COLOR_STRONG    = 0.0
COLOR_SUBTLE    = 0.0
```

### Justification.

- $\alpha = 0.9$ : High denoising enables complete appearance transformation while preserving OpenPose structure.
- $T = 25$ : SDXL architecture allows fewer steps than SD 1.5 while maintaining quality at 768px.
- $s = 9.0$ : CFG scale for stable diffusion models.
- $\lambda_{OpenPose} = 0.8$ : Balances pose preservation vs. natural variation.

### E.3. Object Inpainting Configuration

**SDXL Pipeline for Context-Dependent PII.** Object anonymization employs Canny edge conditioning for structure preservation with Stable Diffusion XL:

```
# Prompt Construction
POSITIVE = "{LVLm_description}"      # Use LVLm-provided semantic description
FALLBACK = "background scene"        # If LVLm description unavailable

NEGATIVE = ""                        # Empty by default (context-appropriate)

# Hyperparameters
RESOLUTION      = 768      # Input/output resolution
STRENGTH        = 0.9      # Denoising strength  $\alpha$ 
STEPS           = 25       # Sampling steps T (SDXL objects)
GUIDANCE_SCALE  = 9.0      # CFG scale s
CANNY_SCALE     = 0.3      # ControlNet conditioning  $\lambda_{CN}$ 
CANNY_LOW       = 10       # Hysteresis low threshold
CANNY_HIGH      = 30       # Hysteresis high threshold

# Color Matching (disabled for complete transformation)
LAB_STRONG      = 0.0
LAB_SUBTLE      = 0.0
```

### Parameter Selection.

- $T = 25$ : SDXL architecture enables efficient generation with fewer steps than SD 1.5
- $\lambda_{Canny} = 0.3$ : Moderate edge guidance allows flexible background reconstruction during object removal
- Canny thresholds (10, 30): Captures salient edges while filtering noise
- Empty negative prompt: Maximizes generation flexibility for diverse object types

## F. License Plate Detection: Extended Analysis

### F.1. Training Methodology

**Dataset Characteristics.** UC3M-LP provides 1,975 annotated images with diverse plate formats, viewing angles, and lighting conditions representative of real-world street imagery.

**Model Architecture.** YOLOv8s (11.2M parameters) balances detection accuracy with inference speed, trained at 1280px resolution for 100 epochs on a single NVIDIA L40S GPU.

**Training Configuration.** Key hyperparameters:

- **Optimizer:** SGD with  $lr_0=0.01$ , momentum=0.937, weight decay=0.0005
- **Learning rate schedule:** Cosine annealing with warmup (3 epochs), final  $lr=0.0001$
- **Loss weights:** box=7.5, cls=0.5, dfl=1.5 (emphasizes localization accuracy)
- **Batch size:** 8 with mixed precision (AMP) training

**Augmentation Strategy.** We employ aggressive data augmentation to improve generalization:

- **Mosaic** (prob=1.0): Combines 4 images into single training sample, enhances multi-scale learning
- **Copy-paste** (prob=0.3): Synthesizes challenging occlusion scenarios
- **MixUp** (prob=0.1): Linear interpolation between samples for regularization
- **Multi-scale training:**  $[0.5, 1.5] \times$  base resolution for scale invariance
- **Color jitter:** HSV adjustments ( $h=0.015$ ,  $s=0.7$ ,  $v=0.4$ ) for lighting invariance
- **Spatial augmentation:** Horizontal flip (prob=0.5), rotation, translation, and scaling

**Training Dynamics.** Figure 2 shows convergence behavior. The model achieves stable performance by epoch 60, with validation  $mAP_{50-95}$  plateauing at 0.816. Training and validation losses demonstrate consistent convergence without overfitting, validating our augmentation strategy.

**Performance Analysis.** Figure 3 presents precision-recall analysis. The model achieves:

$$\begin{aligned} \text{Precision} &= 0.93 \\ \text{Recall} &= 0.94 \\ mAP_{50} &= 0.91 \\ mAP_{50-95} &= 0.82 \end{aligned} \tag{3}$$

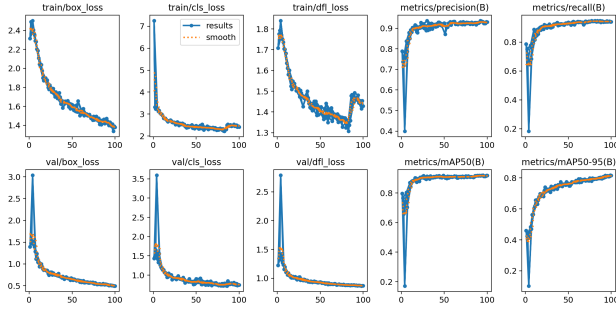


Figure 2. **Training curves for YOLOv8s license plate detector.** Loss curves and metrics over 100 epochs. Stable convergence achieved by epoch 60.

The high recall (0.94) is critical for privacy applications where false negatives are unacceptable. Precision-recall curve analysis shows robust performance across confidence thresholds.

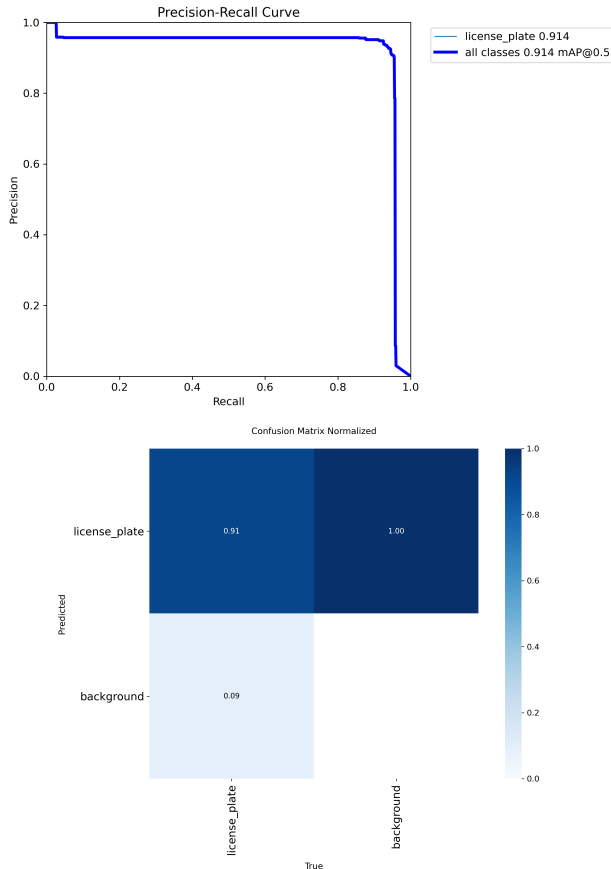


Figure 3. **License plate detection performance.** Left: Precision-recall curve showing  $mAP_{50}=0.91$ . Right: Normalized confusion matrix with 0.94 true positive rate.

## F.2. Training Configuration

We train Mask R-CNN [6] with ResNet-50-FPN [7] on Visual Redactions Dataset [9, 10]: 3,873 training images, 21,489 annotations across 28 privacy categories. COCO-pretrained weights initialize the backbone. Training uses SGD (momentum 0.9, weight decay 0.0001), base LR 0.002 with warmup (500 iterations), decay at 18K/25K iterations, batch size 8 ( $2 \times$  L40S GPUs), 30K iterations ( $\sim 5.7$  hours). Augmentation: random horizontal flip, multi-scale training (640–800px shortest edge).

## F.3. Results

**Training Dynamics.** Total loss decreases from 5.93 to 0.32 over 30K iterations (94.6% reduction). Component-wise: RPN classification (99.2% reduction), classification loss (98.5%), mask loss (82.6%), box regression (72.7%). Validation AP improves monotonically from 11.77% (2K) to 17.70% (30K) without overfitting. Figure 4 shows training dynamics.

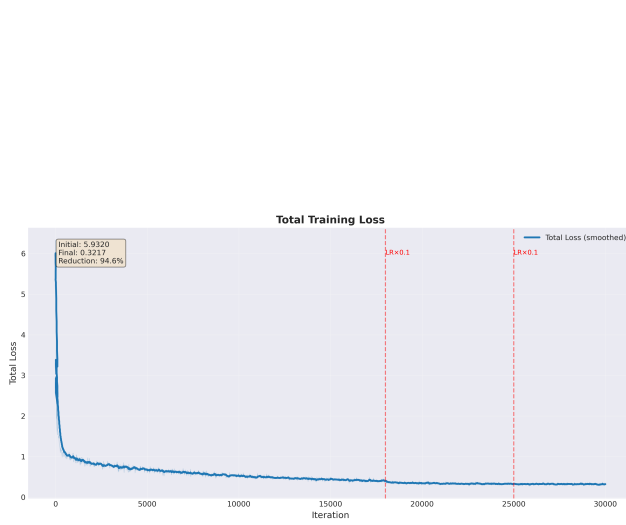
**Category-Level Performance.** Overall mask AP: 17.70% ( $AP_{50}$ : 25.33%,  $AP_{75}$ : 19.41%). Severe scale dependency:  $AP_S$ : 3.37%,  $AP_L$ : 19.60%. Performance clusters by attribute type (Table 1): *Visual classes* (face: 56.98%, person: 48.68%, passport: 70.55%) exploit COCO pretraining successfully. *Textual categories* exhibit systematic failure (10/10 categories  $< 5\%$  AP: name 1.16%, address 0.64%, phone 0.00%), despite abundant data (address: 1,902 instances, date.time: 2,979 instances). Small object scale ( $AP_S$ : 3.37%) causes detection failure on spatially compact text.

**Test Set.** Unified binary masks (logical OR across 28 categories) achieve Dice: 75.83%, IoU: 68.71%, Precision: 80.71%, Recall: 78.02% (2,890/2,989 images; 99 missing predictions). Performance dominated by visually salient categories;  $\sim 22\%$  missed pixels primarily from small text below detection thresholds.

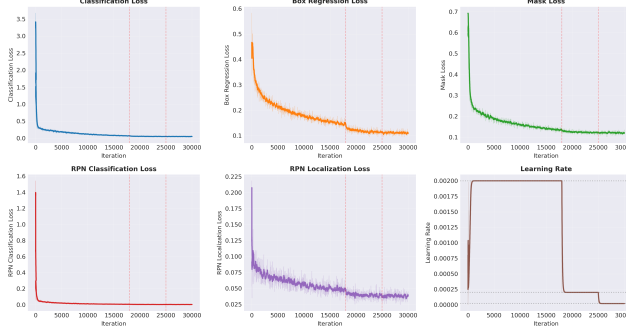
## G. Qualitative Comparison: Supervised vs. Zero-Shot

Figure 5 shows randomly sampled test images comparing ground truth, Detectron2, and zero-shot predictions. Detectron2 achieves higher precision on faces, persons, and documents (categories with abundant training data and COCO initialization) but fails completely on 99/2,989 test images (3.3%). Zero-shot provides broader category generalization but produces false positives on ambiguous regions.

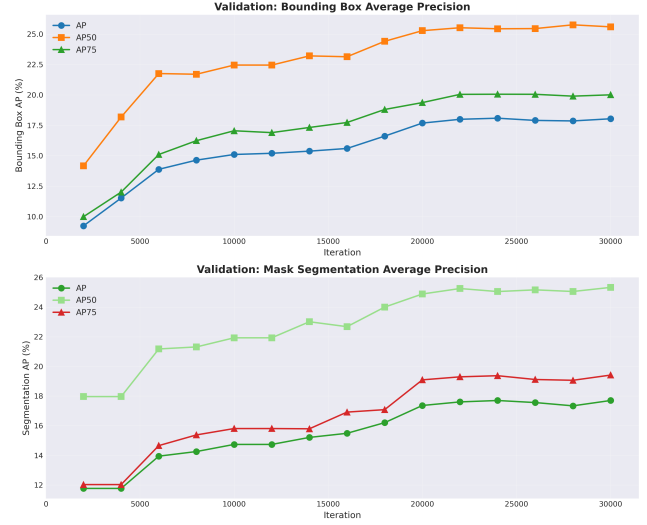
Figure 6 shows Detectron2 best cases: accurate mask boundaries on faces (AP: 56.98%), persons (48.68%), and passports (70.55%). Figure 7 shows zero-shot advantages.



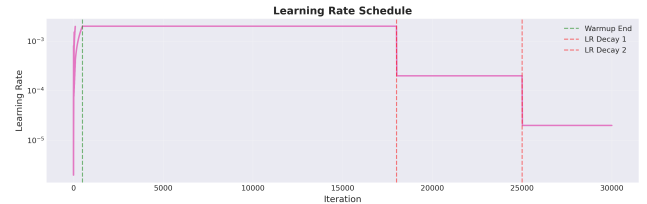
(a) Total training loss



(c) Component losses breakdown



(b) Validation AP progression



(d) Learning rate schedule

Figure 4. **Training dynamics of Mask R-CNN on Visual Redactions Dataset.** (a) Total loss shows 94.6% reduction over 30K iterations. (b) Validation AP: segmentation AP from 11.77% to 17.70%, bbox AP from 9.24% to 18.04%. (c) Component losses: RPN classification (99.2% reduction), classification (98.5%), mask (82.6%), box regression (72.7%). (d) Learning rate schedule: linear warmup (500 iterations), decay ( $\times 0.1$ ) at 18K and 25K iterations.

## H. KITTI Dataset Example

We processed KITTI [5] image 0000000165 to evaluate cross-dataset generalization. KITTI provides street-level autonomous driving imagery at lower resolution than CityScapes ( $2048 \times 1024$ ).

**Detection and Processing.** The pipeline detected 14 persons and 3 rectangular signs (Figure 8). Phase 1 batch-inpainted all persons using SDXL with OpenPose ControlNet. The LVLm generated appearance descriptions for close person bounding boxes: “A medium-build person walking, front view, wearing muted navy blazer and light tan pants”, “A medium-build person standing, side view, wearing bright yellow t-shirt and dark teal pants”. Phase 2 processed rectangular signs using Canny ControlNet.

**PDCA Iteration and Audit.** The workflow executed 3 PDCA iterations before reaching  $n_{\max} = 3$ . The final audit detected 1 residual sign at bbox [948, 0, 1064, 100], likely a distant or partially occluded sign at the image boundary. This illustrates the privacy-utility trade-off:  $n_{\max}$  bounds execution time (391.8s) while occasionally leaving residual PII.

**Agent Dialogue Excerpt.** Table 2 shows Phase 2 agent interactions. After Phase 1 completion, the AuditorAgent invokes `classify_pii`, identifying 1 rectangular sign at bbox [464, 275, 526, 341] (instance ID: pii-0001). The OrchestratorAgent routes this to the GenerativeAgent, which invokes `anonymize_and_inpaint`. Tool execution includes scout-and-zoom segmentation (crop generation, Grounded-SAM-2 detection, mask mapping), protected area subtraction (15 masks from persons and traffic signs), and SDXL inpainting with Canny ControlNet. Final PII cover-



Table 1. Per-category mask AP on Visual Redactions validation set ( $n = 1,611$  images). Categories grouped by performance regime. Training instance counts reflect severe class imbalance. Abundant data does not guarantee success: date\_time (2,979 instances  $\rightarrow$  3.75% AP), address (1,902 instances  $\rightarrow$  0.64% AP). Category distribution shows visual attributes (face, person) dominate both instance count (46.4%) and performance, while textual categories suffer systematic failure despite comprising 36% of training instances.

Category	Instances	AP (%)
<i>High Performance (AP &gt; 20%)</i>		
Passport	153	70.55
Face	4,039	56.98
Person	5,937	48.68
Fingerprint	79	45.02
License Plate	371	33.02
Student ID	37	32.35
Ticket	420	32.14
Receipt	113	30.35
Credit Card	102	18.06
<i>Medium Performance (5% &lt; AP ≤ 20%)</i>		
Disability	73	17.76
Drivers License	30	16.16
Nudity	320	15.20
Ethnic Clothing	123	12.74
Mail	73	12.21
Medicine	116	10.50
Handwriting	1,077	10.29
Birth Date	60	7.48
Signature	252	7.40
<i>Low/Zero Performance (AP ≤ 5%)</i>		
Education	48	5.18
Date/Time	2,979	3.75
Username	310	3.51
Address Home	90	2.11
Landmark	807	1.64
Name	1,785	1.16
Email	68	0.69
Address	1,902	0.64
Phone	105	0.00
ID Card	20	0.00
<b>Total</b>	<b>21,489</b>	<b>17.70</b>

age: 5.29% (37,677 pixels).

**Statistics.** The processing generated 14 person masks and 3 sign masks. Combined PII coverage: 5.29% (37,677 pixels), primarily from pedestrians (4.88% persons, 0.41% signs). Processing time: 391.8s total — Phase 1: 160.5s for Grounded-SAM-2 detection; Phase 2: 231.3s for 3 PDCA iterations with Qwen2.5 VL 32B auditing and Qwen2.5 32B agent reasoning. Each detection included interpretable tool calls and bbox annotations.

Table 2. Agent dialogue excerpt from KITTI Phase 2 workflow (indirect PII detection and anonymization).

Agent	Action/Message
UserProxyAgent	Phase 1 complete. AuditorAgent: classify PII in 0000000165.png using classify_pii.
AuditorAgent	<b>Tool call:</b> classify_pii image: artifacts/.../0000000165.png
Tool Result	instances: [{det.prompt: "rectangular sign with text .", bbox: [464, 275, 526, 341], instance.id: "pii-0001"}] meta: {provider: ollama, model: qwen2.5vl:32b, tiles: 1}
OrchestratorAgent	Phase 2: [classify_pii ✓] Found indirect PII. GenerativeAgent, anonymize instances.
GenerativeAgent	<b>Tool call:</b> anonymize_and_inpaint instances: [{bbox: [464, 275, 526, 341], description: "rectangular sign with text", instance.id: "pii-0001"}]
Tool Execution	Scout-and-zoom: crop [464, 275, 526, 237] $\rightarrow$ detect $\rightarrow$ map mask Protected area subtraction: 15 masks (persons, traffic signs) Inpainting: SDXL + Canny ControlNet PII mask update: 37,677 pixels (5.29% coverage)
...	

**Cross-Dataset Generalization.** The KITTI example tests cross-dataset generalization: lower resolution than CityScapes ( $2048 \times 1024$ ), different camera perspective (dashboard vs. urban mapping vehicle), and varying scene complexity. Zero-shot detection and pose-guided inpainting adapted to KITTI without dataset-specific tuning.

## References

- [1] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.
- [2] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(1):172–186, 2021.



- [3] Junzhou Chen, Heqiang Huang, Ronghui Zhang, Nengchao Lyu, Yanyong Guo, Hong-Ning Dai, and Hong Yan. Yolots: Real-time traffic sign detection with enhanced accuracy using optimized receptive fields and anchor-free fusion. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–17, 2025.
- [4] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016.
- [5] A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The kitti dataset. *Int. J. Rob. Res.*, 32(11): 1231–1237, 2013.
- [6] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2961–2969, 2017.
- [7] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.
- [8] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In *European Conference on Computer Vision*, pages 38–55. Springer, 2024.
- [9] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Towards a visual privacy advisor: Understanding and predicting privacy risks in images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3706–3715, 2017.
- [10] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Connecting pixels to privacy and utility: Automatic redaction of private information in images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.
- [11] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *International Conference on Learning Representations*, 2024.
- [12] Álvaro Ramajo-Ballester, José María Armingol Moreno, and Arturo de la Escalera Hueso. Dual license plate recognition and visual features encoding for vehicle identification. *Robotics and Autonomous Systems*, 172:104608, 2024.
- [13] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. SAM 2: Segment anything in images and videos. In *International Conference on Learning Representations*, 2025.
- [14] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024.
- [15] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023.
- [16] Pascal Zwick, Kevin Roesch, Marvin Klemp, and Oliver Bringmann. Context-aware full body anonymization. In *Computer Vision – ECCV 2024 Workshops*, pages 36–52, Cham, 2025. Springer Nature Switzerland.

01 - 2017-10-20-0001  
Kara Tene, Instagram, Texas  
Deuteranomaly  
50% 45.2%

02 - 2017-10-20-0002  
Kara Tene, Instagram, Texas  
Protanomaly  
50% 45.2%

03 - 2017-10-20-0003  
Kara Tene, Instagram, Texas  
Tritanomaly  
50% 45.2%

04 - 2017-10-20-0004  
Kara Tene, Instagram, Texas  
Deuteranomaly  
50% 45.2%

05 - 2017-10-20-0005  
Kara Tene, Instagram, Texas  
Protanomaly  
50% 45.2%

06 - 2017-10-20-0006  
Kara Tene, Instagram, Texas  
Tritanomaly  
50% 45.2%

07 - 2017-10-20-0007  
Kara Tene, Instagram, Texas  
Deuteranomaly  
50% 45.2%

08 - 2017-10-20-0008  
Kara Tene, Instagram, Texas  
Protanomaly  
50% 45.2%

09 - 2017-10-20-0009  
Kara Tene, Instagram, Texas  
Tritanomaly  
50% 45.2%

10 - 2017-10-20-0010  
Kara Tene, Instagram, Texas  
Deuteranomaly  
50% 45.2%

11 - 2017-10-20-0011  
Kara Tene, Instagram, Texas  
Protanomaly  
50% 45.2%

12 - 2017-10-20-0012  
Kara Tene, Instagram, Texas  
Tritanomaly  
50% 45.2%

13 - 2017-10-20-0013  
Kara Tene, Instagram, Texas  
Deuteranomaly  
50% 45.2%

14 - 2017-10-20-0014  
Kara Tene, Instagram, Texas  
Protanomaly  
50% 45.2%

15 - 2017-10-20-0015  
Kara Tene, Instagram, Texas  
Tritanomaly  
50% 45.2%

16 - 2017-10-20-0016  
Kara Tene, Instagram, Texas  
Deuteranomaly  
50% 45.2%

17 - 2017-10-20-0017  
Kara Tene, Instagram, Texas  
Protanomaly  
50% 45.2%

18 - 2017-10-20-0018  
Kara Tene, Instagram, Texas  
Tritanomaly  
50% 45.2%

Figure 5. **Random examples from Visual Redactions test set.** Ground truth (with category labels), Detectron2, and zero-shot predictions. Supervised achieves higher precision on common categories but fails on 99 images (3.3%). Zero-shot provides full coverage with broader generalization but occasional false positives.



Figure 6. **Detectron2 best cases.** Top 8 examples by Dice score. Supervised excels on faces, persons, documents with abundant training data and COCO initialization.

Zero-Shot Outperforms Detectron2 (Largest IoU Gaps - Top 8)



Figure 7. **Zero-shot advantage cases.** Top 8 examples where zero-shot outperforms Detectron2. Agents with LVLm utilization succeed on rare categories, novel formats, and low-confidence regions.



Figure 8. **KITTI example (0000000165).** **Left:** Original street scene with 14 pedestrians and multiple traffic signs. **Right:** Anonymized output after PDCA workflow. Phase 1 batch-inpainted 14 persons using SDXL with OpenPose ControlNet. Phase 2 processed 3 rectangular signs with Canny ControlNet. Final PII coverage: 5.29% (37,677 pixels). Total processing time: 391.8s with 3 PDCA iterations.