Automatic Image Processing for Privacy

EECS545 Project

Aarya Kulshrestha akulshre@umich.edu
Hendrik Mayer htmayer@umich.edu
Omkar Vodela omkarv@umich.edu
Andrew Mayo acmayo@umich.edu

Abstract

The widespread deployment of AI-driven image processing on mobile and embedded computing devices introduces significant privacy challenges. These devices inadvertently capture sensitive visual information, making it difficult to use for analytics and other uses. Traditional approaches to privacy protection only remove sensitive objects, which either destroy too much valuable content or fail to conceal private information adequately. In this project, we introduce a dynamic, context-adaptive privacy-preserving system that detects a variety of sensitive objects and assigns each a continuous privacy score, allowing nuanced decisions between preserving or obfuscating content. We use a variety of datasets including the DIPA and Laion datasets, which contain a diverse set of private objects. Using a combination of fine-tuned YOLOv11-seg models for instance segmentation, OCR for text extraction, handcrafted privacy features, and local LLMs for semantic privacy evaluation, our pipeline assesses visual and contextual sensitivity at a fine-grained level. Based on the privacy score, the system determines whether to leave objects unaltered, blur the object, or inpaint using large mask inpainting with Fourier convolutions (LaMa). This framework ensures sensitive information is effectively concealed, while minimizing unnecessary image distortion.[1]

1 Introduction

The rise of smart glasses, wearable cameras, drones, and surveillance systems, just to name a few, has fueled a massive surge in AI-powered visual data capture. While these technologies unlock tremendous value across industries, such as healthcare, security, autonomous systems, they also introduce profound privacy challenges. These devices often inadvertently record personal documents, addresses, private screens, individuals' identities and much more in uncontrolled environments.

The inability to effectively remove such information restricts the adoption of AI-driven imaging and analytics within industry. Outside of industry applications, individuals using wearable and mobile vision technologies must exercise caution about how data captured by these devices is handled. This is because everyday activities such as entering a bathroom or glancing at a sticky note on a desk filled with passwords can unintentionally expose those private details. Current privacy-protection methods are often too simplistic, focusing narrowly on a small subset of object types (e.g., faces or license plates) and apply binary removal strategies.

More critically, real-world scenes contain a rich diversity of sensitive content. This can include legible text from handwritten notes to business signs to electronic screens. Their sensitivity is context-dependent, not binary. Thus, we aim to research if there is a way to build a smarter privacy system that moves to a multi-headed approach. Our main research motivation is to determine whether we are able to dynamically quantify the risk of each sensitive object within a scene and allow users to control how much to obfuscate the scene with minimal distortions.

If we can solve this problem, it would dramatically expand the safe use of AI vision systems, allowing industries to extract insights from images while respecting individuals' privacy rights. Such a system would also promote ethical AI development and build public trust in how AI-driven visual data is captured, processed, and shared to provide valuable insights in a privacy-conscious world.

2 Method

2.1 Pipeline

Our system is designed to dynamically detect and obfuscate privacy-sensitive content in images using

[1] Code for the project can be found at https://github.com/andrmayo/image_privacy.

a modular pipeline that combines real-time segmentation, OCR-based text extraction, and a multi-factor privacy scoring mechanism. The objective is to preserve as much visual information as possible while protecting content deemed sensitive based on both visual and semantic cues. Figure 1 illustrates our full pipeline architecture.[2]

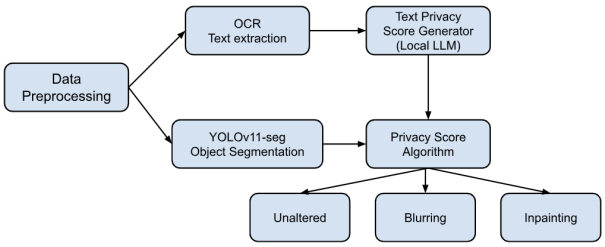


Figure 1: Our proposed pipeline to process an input image

The pipeline begins with a fine-tuned YOLOv11-segmentation model that identifies a broad range of privacy-relevant object classes.[3] YOLOv11-seg is chosen for its real-time performance, flexibility for fine-tuning, and robust multi-class detection. We used YOLOv11-seg because real-time object detection is more practical for images that are captured by mobile and wearable vision devices. Fine-tuning also makes our model more effective than models that only detect a few object types. This is because we want to precisely and reliably detect objects that we deem as sensitive.

We consider a diverse set of privacy-sensitive objects that frequently appear in real-world images. Object types we will look at include: addresses, business signs, electronic screens, faces, license plates, personal documents, photos, and street signs.

To account for textual privacy risks (e.g., names, text sentences, IDs, addresses, etc.), we integrate EasyOCR to extract text from the image regardless of object class. It is important to feed the whole image forward through EasyOCR because text may appear within a detected object, such as an electronic screen or personal document, which may play a role in how sensitive the object that YOLO detects is. Moreover, there can also be text that YOLO doesn't pick up outside of object instance segments that may be important to note as well.

We then use spatial union logic to merge OCR bounding boxes with the corresponding object segment masks when they overlap significantly. This allows us to unify semantically connected visual regions that were extracted from EasyOCR and object regions from our YOLOv11-seg model. For example, text on a document or screen can be merged into a single region for privacy scoring.

To assess whether and how much to obfuscate a text, object, or merged region, we compute a privacy score based on carefully engineered, class-specific weighted features. Each class has a predefined set of feature weights that reflect how much each factor contributes to the privacy risk for that class. For example, a photo may emphasize center focus and blurriness, while a license plate prioritizes OCR confidence and size. Each detected region is evaluated using a combination of its base sensitivity level (e.g. license plates are more sensitive than a generic document), visual properties, and textual content, if any. The weighted visual metrics include:

- **Blurriness:** Measured using the Laplacian variance to quantify sharpness; higher variance suggests clearer detail and therefore higher sensitivity.
- **Background complexity:** Estimated using Canny edge detection to capture visual clutter or structure.
- **Object size:** Larger regions are considered more attention-grabbing and therefore riskier.
- **Center focus:** Regions located closer to the center of the image are assumed to be more salient and potentially intentional captures.

Specifically for OCR text present in the region, we want a way of understanding the level of sensitivity. The weighted textual metrics include:

- **LLM Score:** LLM-derived privacy score, calculated by passing the extracted text and object class into a locally hosted language model (e.g., LLaMA via Ollama), which returns a score between 0.0 (not sensitive) and 1.0 (extremely sensitive).
- **OCR Confidence:** EasyOCR confidence

To calculate the LLM score effectively, we had to engineer a prompt to deliver promising scores. The final prompt is shown in Figure 2. This score reflects semantic content risk, such as whether the text includes names, credentials, or personal identifiers. Importantly, we use a local LLM to avoid any privacy compromise in the scoring process itself.

Once the relevant class metrics are computed,

[2]The pipeline code itself is https://github.com/andrmayo/image_privacy/blob/main/pipeline.py.

[3]The training code we used is https://github.com/andrmayo/image_privacy/blob/main/yolo/train.py.

```
prompt = f"""
    You are determining the privacy sensitivity of the list of text detected in a region of an
    image.

    Region class: {YOLO_CLASSES[int(class_id)]}
    Detected text: {text}

    Output a single number between 0.0 (completely non-sensitive text) and 1.0 (extremely
    private information text) with regards to the class.
    This number should be 1.0 or close to 1.0 if the text identifies a person or place,
    and if the text contains passwords, numerical ids, or financial information.
    Only output the number, nothing else.
"""
```

Figure 2: Our LLM prompt

the final privacy score is computed as a weighted average of the active features. Based on the final privacy score and a user-defined threshold, each region is either left unaltered or obfuscated. A higher privacy score (1.0) would mean highly sensitive information as opposed to a lower privacy score (0.0). For obfuscation, users can choose between blurring and inpainting. Blurring involves a simple gaussian blurring over the extracted region. Users may want to apply Gaussian Blurring in cases where partial concealment is sufficient, while retaining the object's presence. This method is particularly useful for human faces in images where general context is still needed but identifiable details must be hidden. The inpainting method we chose (LaMa, discussed further in **Related Work**) uses fast Fourier convolutions, a perceptual loss function, and large masks to fill in regions in a way consistent with the surrounding image. This approach is particularly effective for eliminating personally identifiable objects such as ID cards or text-based sensitive elements from images while maintaining visual coherence, although doing this well in the context of our pipeline depends on tight object segmentation.

This sequential pipeline flow is more practical over traditional privacy-preserving techniques. By applying selective transformations of objects based on their sensitivity, our system offers granular control. The use of Gaussian blurring and generative inpainting ensures that only necessary modifications are made, preserving as much image utility as possible. Additionally, the real-time feasibility of YOLOv11-seg and optimized OCR makes our approach suitable for mobile and edge computing applications without compromising efficiency.

2.2 Datasets

The ideal dataset we were looking for was a dataset that contains images with privacy-sensitive objects and is annotated with segmentation masks for these objects. No dataset we found quite fit the bill, so instead we have combined several datasets. The dataset that comes closest to what we wanted is the DIPA (Dataset with Image Privacy Annotations) dataset (Yu et al., 2017).[4] This dataset has 1,495 images and 5,671 annotations from crowd-sourcing platforms, and is curated to contain images relevant to privacy concerns. Many of these annotations include bounding boxes for relevant objects. The single biggest issue is that it is a fairly small dataset, and not all of the images in it include relevant objects. The dataset includes bounding box annotations for various objects and some segment annotations. We found, however, that these annotations were too inconsistent to be useful.

We addressed the issue of annotations by creating segment annotations for the entirety of the dataset ourselves using Roboflow,[5] and also combining it with our annotations of images drawn from other datasets. We looked at VizWiz[6] and COCO[7] datasets, but decided against using them. (The COCO dataset is, however, contributed to our model indirectly, since we used a YOLO model pretrained on it.)

Some of our object classes were severely underrepresented in the DIPA dataset. To address this, we made considerable use of the Laion dataset[8]. This dataset was useful to us because of its size of 400 million images and because it has text annotations describing each image. These captions are generally very unreliable, but the size of the dataset means that even a very crude search will turn up relevant images. To make use of this massive dataset, we wrote code to search the parquet files containing the text annotations for a given keyword and download only those images.[9]

We drew a smaller number of images from the Stanford Street View House Numbers dataset[10] to address a problematic lack of building addresses in our prior data.

[4] https://anranxu.github.io/DIPA_visualization/

[5] app.roboflow.com/imageprivacy/image_privacy/

[6] https://vizwiz.org/tasks-and-datasets/vqa

[7] https://cocodataset.org

[8] https://laion.ai/blog/laion-400-open-dataset/

[9] The search script can be found at https://github.com/andrmayo/image_privacy/blob/main/data/laion/search_parquet.py. We used this, for instance, to search for "intersection" in the Laion captions to find images of street signs. We then manually filtered out irrelevant images returned by the search and annotated the relevant images using Roboflow.

[10] (http://ufldl.stanford.edu/housenumbers/

DIPA	Laion	Stan. Addresses
1495	441	100

Table 1: Sources of images for training (numbers are prior to train-val-test partition)

2.3 Model Selection and Data Augmentation

We did significant experimentation with different YOLO models and data augmentation methods. We found that YOLOv11 pretrained for segmentation worked better than the available YOLOv12 models (the only pretrained YOLOv12 model accessible with the Ultralytics API is the detection model). We also found that going above the small YOLOv11 model did not improve performance, and may in fact have worsened it (see 3 in the **Appendix**).[11]

We also ran experiments with data augmentation, something especially important given our small and imbalanced dataset. We found that introducing some data augmentation in addition to YOLO's default albumentation did help, but that it was easy to push this too far. For instance, with the addition of left-right flipping and erasing (see 4), performance worsened, probably because given the small dataset this made the model lose some signal in the added noise.[12]

3 Related work

Privacy-preserving techniques in computer vision have been an area of vast research, with various approaches to detect[13] and obscure sensitive information in images. Traditional methods relied on feature-based detection, but modern approaches use neural networks, OCR, and generative models for more effective privacy preservation.

3.1 Privacy-preserving Object Detection

One approach mentioned in *iPrivacy: Image Privacy Protection by Identifying Sensitive Objects via Deep Multi-Task Learning* (Yu et al., 2017) employs multi-task learning with deep Convolutional Neural Networks (CNN) to detect sensitive information. This improves efficiency compared to handcrafted visual features such as SIFT, GIST and color histograms. but it also limits flexibility across diverse object types. They also provide a simple solution for image privacy protection by automatically blurring privacy-sensitive objects.

The paper leaves open the challenge of addressing more complex privacy scenarios. There is room for further work in being able to classify images in a faster way. Deep CNN models use a two-step process—first generating region proposals and then classifying each region, which is computationally more expensive. Other approaches such as single-stage processes an image in a single forward pass of the neural network, making it extremely fast. We opted to explore this for our framework to prevent the computational cost becoming a bigger bottleneck to real-time privacy-sensitive settings especially with mobile computing devices that need high performance with the number of frames they can capture and classifying multiple privacy sensitive objects.

Other approaches, especially with single-stage approaches have been researched such as YOLO for object detection and Tesseract OCR for text detection (Raees and Al-Tamimi, 2024; Woringer and Miraoui, 2024). Both these approaches offer strong performance, but often require separate pipelines for different privacy-sensitive elements as they fundamentally work on different types of objects. This would increase system complexity and computational overhead.

Now, we describe EasyOCR and YOLO, which we build off to construct our privacy-enhancing system. EasyOCR is a widely used open-source OCR engine developed by the Jaided AI team, which supports over 80 languages (Jaided AI Team, 2020). Another popular OCR engine, which we do not use, is Tesseract OCR; it was made open-source by Google in 2005. YOLO is a method for real-time object detection in images, which frames object detection as a regression problem to predict proper bounding boxes on images (Joseph Redmon, 2016). Previous works simply used classifiers to perform object detection. The initial small version of YOLO (Fast YOLO) could process 155 frames per second, which makes it an ideal approach for mobile computing devices to detect objects that may contain private information.

The paper *Enhancing Privacy: Automated Detection and Blurring of Sensitive Information in Images and Video Feeds* however employs a unified pipeline integrating YOLOv8 and OCR for privacy-preserving object detection and anonymization in

[11]The reason there are additional dataclasses in these plots is because we did our initial testing with 10 data classes rather than 8, before deciding the advertisements were not relevant and text was better detected by OCR.

[12]The full suite of plots from training runs, which give a more complete picture, can be found at https://github.com/andrmayo/image_privacy/tree/main/runs.

[13]For a survey of objection detection methods for privacy, see (Moon et al., 2024)

images and videos. They use YOLOv8 and fine-tune it to detect vehicles and license plates, dividing the input image into bounding boxes and classifying probabilities in real-time. Any detected license plate regions are passed to an OCR model (EasyOCR (Jaided AI Team, 2020) or Tesseract (Smith, 2007)) for verification by reading alphanumeric characters, reducing false positives (Woringer and Miraoui, 2024).

Similar to their work, we use a combination of YOLO and OCR in our pipeline, but the OCR is not used to reaffirm the YOLO model classification of privacy sensitive objects. We use OCR to be able to extract text from not just license plates, but any pictures that have descriptions. YOLO isn't able to classify text and also determine the sensitivity. OCR in our framework will use YOLO to extract objects and transcribe it into text.

In addition, the authors of the paper *Real-Time Video Anonymization in Smart City Intersections* developed a system using YOLOv4 for blurring pedestrian faces and vehicle license plates that are picked up on traffic cameras (Angus et al., 2024). They tested their system video from a traffic intersection from the COSMOS testbed (Raychaudhuri et al., 2019). Crucially, they were able to blur faces and license plates with recall up to 99 percent and real-time inference speed up to 100 fps.

The paper *Privacy-Aware Visual Language Models* discusses the use of LLM and VLM in extracting image data. This can be very helpful for making meaning of the text we extract from OCR. However, they found that the integration of visual modality in VLMs can further weaken the safety alignment that exists in the underlying LLM component (Samson et al., 2024). Thus, we decide instead of using these we feed text into a local LLM that will allow us to determine how sensitive the text actually is in a privacy preserving manner.

3.2 Obfuscation Techniques

Once privacy-sensitive elements are detected, various obfuscation techniques can be applied to protect the sensitive information while maintaining image quality and usability.

3.2.1 Traditional Methods

Blurring and masking remain common techniques due to their simplicity and computational efficiency. Gaussian blurring, in particular, has been widely used for face anonymization. However, studies have shown that these methods can leave reconstructible patterns that sophisticated algorithms might exploit to recover the original content.

3.2.2 Generative Approaches

To address the limitations of traditional methods, generative approaches have emerged as promising alternatives. Liu et al. proposed a GAN-based differential privacy framework for image protection in IoT multimedia applications (Jinao Yu, 2020). Their approach uses deep neural networks to detect relevant objects and replace them with synthetic content generated by GANs. A key contribution of their work is the incorporation of differential privacy techniques to navigate the trade-off between image quality and privacy protection.

More recently, Lugmayr et al. introduced "RePaint: Inpainting using Denoising Diffusion Probabilistic Models" (DDPM) for generative inpainting (Lugmayr et al., 2022). The DDPM process starts with a Gaussian noise sample that is incrementally denoised until the final output image is obtained. This approach improves upon GAN and autoregressive methods by generalizing well to arbitrary mask types and sizes, making it particularly suitable for diverse privacy-sensitive objects. This generalization capability is crucial for our project, as we aim to apply inpainting to objects of varying sizes and shapes.

Most relevant to this project is large mask inpainting with Fourier convolutions with LaMa (Suvorov et al., 2021), which is a GAN. Some of the advantages of this recent method are good performance across low-resolution and high-resolution images and being relatively light weight. The implementation we used[14] requires a GPU but, in principle, LaMa should have reasonable inference speeds running on a CPU as well. Stable diffusion would likely perform better under ideal conditions, but given that data is a major constraint and we wanted fast inference times, we opted for LaMa.

3.3 Evaluation Metrics

Evaluation metrics in this domain typically balance privacy protection efficacy with visual quality preservation. Technical metrics include:

- **Privacy Protection Metrics**: Re-identification prevention rate, privacy attribute concealment rate, and adversar-

[14] See our code at https://github.com/andrmayo/image_privacy/blob/main/obfuscation/lama_inpaint.py

ial success rate against state-of-the-art recognition systems.

- **Image Quality Metrics**: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), Learned Perceptual Image Patch Similarity (LPIPS), and Fréchet Inception Distance (FID).

- **Computational Efficiency**: Processing time per image, memory requirements, and scalability to large image collections.

However, there remains a need for more comprehensive evaluation frameworks that consider both technical performance and user perceptions of privacy protection (Oh et al., 2016). conducted user studies showing that perception of privacy protection often differs from technical metrics, with participants rating visually obvious obfuscation (like blurring) as more protective even when technically inferior to subtle adversarial perturbations.

In this vein, we opted for a more tailored approach to evaluating our application. The aims of this project are unique or nearly so, and there are not any existing metrics for privacy that are applicable to it. Consequently, we have mainly evaluated the application against human judgment as applied to the same dataset, the results of which are discussed below.

3.4 Our Approach and Contributions

Our approach differs from previous work in several key aspects. First, we apply two detection methods for each privacy-sensitive object type by utilizing YOLO and OCR in our pipeline. This object-specific optimization allows us to achieve higher detection accuracy across diverse privacy concerns.

Second, unlike many existing systems that focus on specific privacy concerns (e.g., faces or text), our framework handles multiple object types simultaneously, providing comprehensive privacy protection. This multi-object detection and processing capability is particularly important for real-world applications where images often contain various types of sensitive information.

Third, we introduce a novel privacy scoring mechanism that quantifies the privacy risk associated with different elements in an image, allowing for more nuanced protection strategies. This scoring system enables our framework to prioritize the most sensitive elements when computational resources are limited.

Fourth, our hybrid obfuscation approach combines traditional methods (blurring) with advanced generative large mask inpainting with Fourier convolutions with LaMa (Suvorov et al., 2021). For example, while blurring might be sufficient for certain text elements, faces might benefit more from complete replacement using generative inpainting.

4 Pipeline Evaluation

4.1 Criteria

In this section, we describe the criteria that we use to determine the effectiveness of our pipeline. In particular, we chose these criteria to ensure that sensitive information is obfuscated, and, if possible, to minimize image distortion. These criteria are:

1. Obfuscating all license plates that are fully visible. **2.** Obfuscating all business signs containing legible store names. **3.** Obfuscating all addresses. **4.** Obfuscating all street signs with street names, and other text identifying locations. **5.** Obfuscating sensitive or private content (e.g., emails, social media) on electronic screens, but not screens in general. **6.** Obfuscating all personal documents and ID cards. **7.** Ensuring all images have identifiable faces obfuscated, while non-human elements are preserved. **8.** Avoiding obfuscation on irrelevant objects (e.g., irrelevant text, random objects).

In effect, **1.** through **7.** are designed to limit false negatives, or cases where sensitive information is not obfuscated. On the other hand, **8.** is designed to limit false positives, or cases where non-sensitive information is obfuscated.

The idea is that the default settings for our pipeline should work for all of these criteria.[15]

4.2 Evaluation Data

The evaluation data is the test set held out from training our YOLO model for object segmentation. We also avoided tuning the various parameters for privacy scoring with reference to this test set, to

[15]These settings include a privacy score threshold at 0.5, ignoring objects detected by YOLO with a confidence score $\leq$ 0.4, and the privacy score weightings given by CLASS_FEATURE_WEIGHTS in https://github.com/andrmayo/image_privacy/blob/main/pipeline.py.

Additionally, we are evaluating the pipeline with inpainting rather than Gaussian blurring, except for applying blurring to faces, since inpainting more consistently obscures relevant information. Gaussian blurring is the default, however, since the inpainting implementation with LaMa we're using requires a GPU.

avoid cooking the books, so to speak, to make our application appear better than it is. The train-val-test split is 85-10-5 (1740, 199, and 97 images respectively), partitioned so as to have a proportional representation of each object class across the sets. We removed images that were not particularly relevant to evaluation, and added some additional images to the test set that were useful test cases for what we want the application to do. [16].

After removing and adding images, the test set comes out to 106 images.

4.3 Results

The ground truth for evaluating our pipeline comes from our own annotation of the test set, according to what we wanted the pipeline to do given the default settings.

Criterion	Count
1 license plates	16
2 business signs	21
3 addresses	21
4 street names	13
5 screens	14
6 docs	23
7 faces	33

Table 2: Ground truth for desired obfuscations

Criterion	Count	Accuracy
1 license plates	15	0.9375
2 business signs	18	0.8571
3 address	19	0.9048
4 street names	8	0.6154
5 screens	11	0.7857
6 docs	19	0.8261
7 faces	28	0.8484
8 false positives	16	N/A

Table 3: Performance on desired obfuscations

Precision	Recall	F1 Score
0.8806	0.8369	0.8582

Table 4: Metrics

Given the unique nature of what our application is trying to accomplish, these metrics cannot meaningfully be compared against any baseline. Instead, they reflect how well the application can approximate human judgment on judging objects to be privacy-sensitive.

4.4 False negatives

The small size of the dataset means there is significant variance, but some things are clear enough, especially when the images involved are considered. Some false negatives are clearly because of challenging images. For instance, the relatively low performance for street signs (especially when compared to metrics for the YOLO model itself) is largely due to the presence of a black and white image containing several street signs. In other cases, false negatives are due to something being amiss with our privacy scoring. In particular, some of the faces missed by the application were detected with high confidence by the YOLO model itself, which points in the direction of our privacy scoring.

One significant kind of false negative reflected in these metrics is cases where application obscures a photo id save for the picture of the person's face. This seems to mainly be due to the quirks of our privacy scoring for photos. To capture this, we counted any photo id as 2 objects, one for the photo and one for the text of the document.

4.5 False positives

We seem to have successfully controlled the false positive rate (i.e. unwanted obfuscation) arising from completely haywire object detection.

False positives we do have arise for two reasons:

1. objects we don't want to obscure that resemble objects we do want to obscure. This is a failure of the YOLO model, e.g. mistaking signage for street signage.

2. text that is wrongly scored as sensitive. This is a failure of the LLM and our privacy scoring on the basis of the score produced by the LLM in conjunction with other factors.

A larger proportion of the false positives come from 2. Specifically, 12 false positives were primarily due to the privacy scoring of text using OCR and Llama3, vs. 4 from mistakes made by YOLO in conjunction with privacy scoring. An interesting example of a false positive due to LLM scoring is 5 in the **Appendix**. Here, the YOLO model correctly detects the phone (although captures more than the screen), and our pipeline tries to pass text

[16] In particular, we added images from https://arxiv.org/html/2407.20662v1#S3, which is a large and diverse benchmark dataset for identity documents analysis, and https://data.mendeley.com/datasets/kp89xh68p2/1

to the LLM for scoring. From the different components of the privacy score (see figure), we can see that the object does not score very highly save due to the LLM. OCR somehow picked up the words "MDtoro" and the LLM took this to be personally identifying in some way. In general, this seems to be the most typical cause of false positives with the application's default configuration: some combination of OCR reporting erroneous text and the LLM judging the text (often nonsensical) to be sensitive.

A more straightforward example of a false positive is simply where YOLO mistakes random signs for business signs, as in 6. The issue here is probably just that we had insufficient training data for business signs for the YOLO model to consistently differentiate them from other signage. The model is better at avoiding false positives for street names, probably because these are much more uniform in appearance than business signs.

4.6 Success Cases

The metrics in figures 3 and 4 indicate that, for the most part, the application did what we wanted it to do with the default settings.

A good example of the pipeline producing the desired output is 8 in the **Appendix**, which shows a computer screen displaying an instagram page. Here, we can see that YOLO detected an electronic screen, but the screen itself is not inpainted. Instead, the face in the user's profile picture has been blurred. OCR has picked out all the text in the image, and given the LLM has given appropriate privacy scores for the different pieces of text. (See 8 for the scoring for the text object corresponding to the profile username.)

5 Discussion of Results

The results provide evidence that our privacy-preserving image obfuscation pipeline which combines object detection, OCR, LLMs effectively obfuscates sensitive information while conserving image utility. The system demonstrates high precision and recall, as well as particularly high accuracy with license plates (93.8%), addresses (90.5%), and faces (84.8%). These successes indicate that our pipeline is able to identify and obfuscate certain categories of sensitive information from images.

However, the results also indicate that there is significant room for improvement with our pipeline. First, our OCR model often misinterprets text or detects text where there is none, which caused many of our false positives. We believe that these types of errors may be mitigated with a better OCR model or some filtering of the OCR outputs depending on confidence. Our OCR model also sometimes processes individual words in a phrase separately, which dilutes the meaning of the text passed to the LLM. This illustrates a general lack of robustness towards noisy or distorted images, where it may be difficult to determine if there is sensitive information. Additional prompt engineering for the LLM may also improve performance, although longer prompts do lead to significantly slower inference.

Our results also show that there were only 16 total false positives across the 106 images, which seems relatively good (the causes of this are discussed above). While these results show the efficacy of our system, the pipeline could benefit from more high-quality labeled training data, confidence-based OCR filtering, and further LLM prompt refinement.

Next, we address the latency of our pipeline. Running our full pipeline (from input image to obfuscated image) with inpainting took 309 seconds using an RTX 3090 on 106 images. Notably, without inpainting, this took 107 seconds, showing that inpainting slows down the process significantly. Inpainting with LaMa does lead to visually cleaner results than Gaussian blur, so there is a tradeoff between visual appearance and latency.

We also qualitatively observed that the amount of text detected in an image drastically impacts the runtime, which we expected because querying the LLM is relatively slow. Faster overall runtimes are certainly more ideal, but this runtime of 309 seconds (with GPU) for 106 images shows that such a pipeline could be used with mobile or embedded computing devices in the future.

Overall, these results indicate that our overall system design is an effective, lightweight approach for obfuscating sensitive information while retaining image utility. The YOLO-based image detection remains fast and (mostly) effective after being finetuned on multiple classes of private objects. The use of OCR and LLMs allows the user to compute privacy scores for different objects. In addition, we designed our system to be modular (OCR model, LLM can be easily changed) with user-adjustable settings (e.g. thresholds, blur intensity) to provide a strong base system that can be personalized for a variety of real-world deployments.

Author Contributions

Aarya	Pipeline
	Privacy scoring
Hendrik	OCR
	LLM
Omkar	General research
	Curating test data
Andrew	Data processing
	Model tuning

Table 5

Anything not listed in the table above was a joint effort of three or four of the co-authors. All co-authors contributed to writing the report.

References

A. Angus, Z. Duan, G. Zussman, and Z. Kostic. 2024. Real-time video anonymization in smart city intersections. *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, XX(X):XXX–XXX.

Jaided AI Team. 2020. Easyocr: Ready-to-use optical character recognition with 80+ languages. Accessed: March 14, 2025.

Bo Liu Yu Wang Shibing Zhu Ming Ding Jinao Yu, Hanyu Xue. 2020. Gan-based differential private image privacy protection framework for the internet of multimedia things. *Sensors*, 21(1):58.

Ross Girshick Ali Farhadi Joseph Redmon, Santosh Divvala. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. 2022. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11461–11471.

Jihoon Moon, Maryam Bukhari, Chomyong Kim, Yunyoung Nam, Muazzam Maqsood, and Seungmin Rho. 2024. Object detection under the lens of privacy: A critical survey of methods, challenges, and future directions. *ICT Express*, 10(5):1124–1144.

Seong Joon Oh, Rodrigo Benenson, Mario Fritz, and Bernt Schiele. 2016. Faceless person recognition: Privacy implications in social media. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pages 19–35. Springer.

Salar Adil Raees and Mohammed S.H. Al-Tamimi. 2024. The role of artificial intelligence in providing people with privacy: Survey. *Journal of Applied Engineering and Technological Science*, 5(2):813–829.

D. Raychaudhuri, I. Seskar, G. Zussman, T. Korakis, D. Kilper, T. Chen, J. Kolodziejski, M. Sherman, Z. Kostic, X. Gu, H. Krishnaswamy, S. Maheshwari, P. Skrimponis, and C. Gutterman. 2019. Challenge: Cosmos: A city-scale programmable testbed for experimentation with advanced wireless. *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, XX(X):XXX–XXX.

Laurens Samson, Nimrod Barazani, Sennay Ghebreab, and Yuki M. Asano. 2024. Privacy-aware visual language models. *arXiv preprint arXiv:2405.17423*.

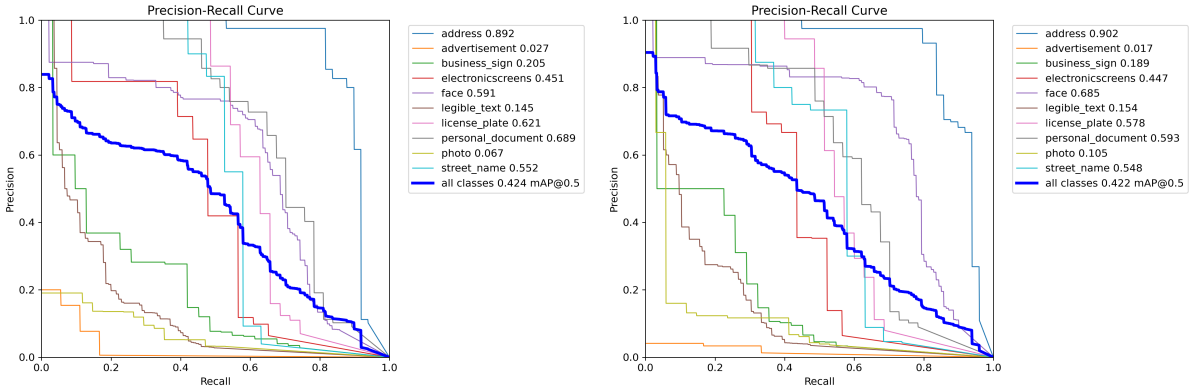
Ray Smith. 2007. An overview of the tesseract ocr engine. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 629–633.

Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. 2021. Resolution-robust large mask inpainting with fourier convolutions.

Paul Woringer and Yanis Najy Miraoui. 2024. Enhancing privacy: Automated detection and blurring of sensitive information in images and video feeds. *CS231n: Deep Learning for Computer Vision, Stanford University*.

J. Yu, D. Tao, M. Wang, and Y. Rui. 2017. iprivacy: Image privacy protection by identifying sensitive objects via deep multi-task learning. *IEEE Transactions on Information Forensics and Security*, 12(5):1005–1016.

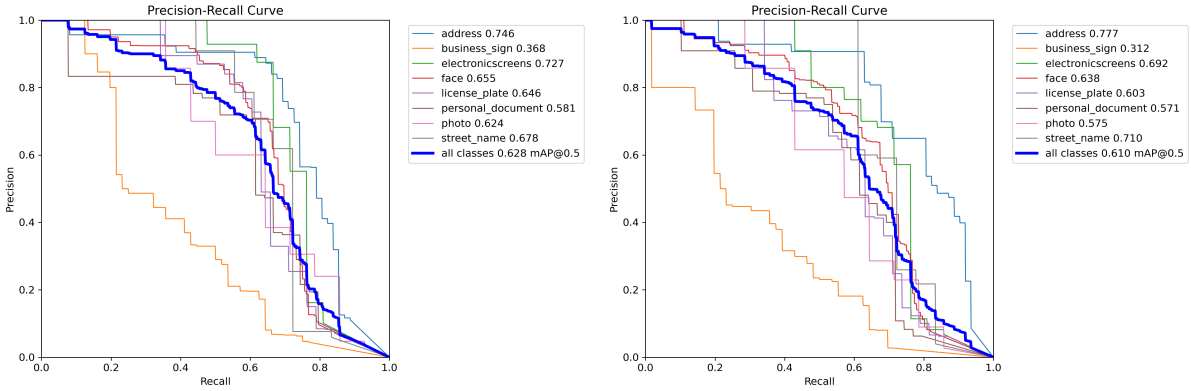
Appendix



(a) Small YOLO Model

(b) Medium YOLO Model

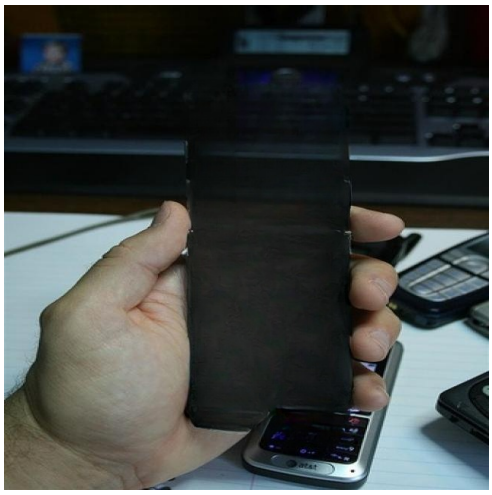
Figure 3: Comparison of precision-recall tradeoff for small vs. medium YOLOv11 models pretrained on COCO dataset



(a) Small YOLO Model

(b) Medium YOLO Model

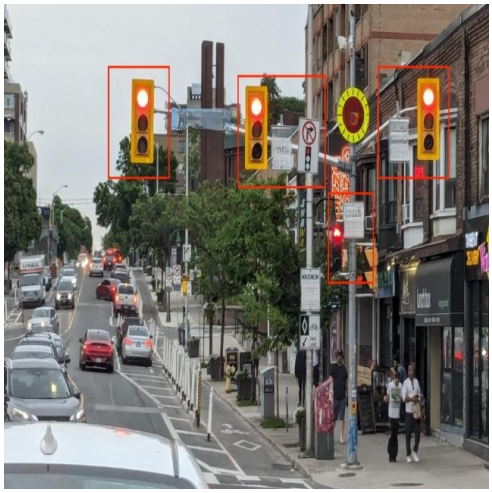
Figure 4: Comparison of YOLO11s with less data augmentation ($hsv\_h : 0.015, hsv\_s : 0.7, hsv\_v : 0.4, degrees : 0.25, translate : 0.1, scale : 0.5, perspective : 0.0001, mosaic : 1.0$) vs. more augmentation (..., $fliplr : 0.5, erasing : 0.4$)



(a) Image with inpainting

(b) Image with YOLO detection

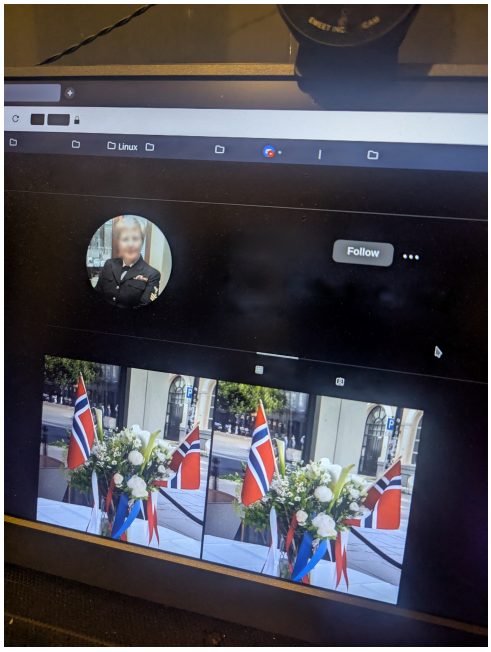
Figure 5: False positive for screen without private content
TEXT: "MDtoro"
Confidence Score: 0.05589
Complexity Score: 0.0876
LLM Score: 0.9
Privacy Score: 0.6799



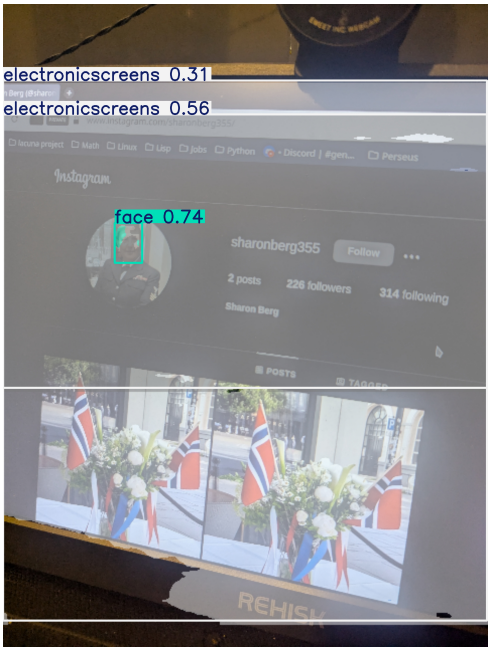
(a) Image with inpainting

(b) Image with YOLO detection

Figure 6: False positive for traffic signs



(a) Image with blurring and inpainting



(b) Original with YOLO detection

Figure 7: Successful output for sensitive objects on screen

Scoring for Username

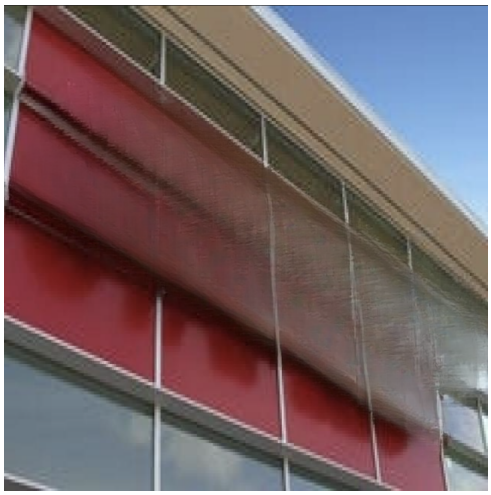
TEXT: "sharonberg355"

CONFIDENCE SCORE: 0.9924

COMPLEXITY SCORE: 0.04420

LLM SCORE: 1.0

PRIVACY SCORE: 0.8555



(a) Image with inpainting

(b) Original

Figure 8: Successful inpainting of business sign

Scoring for STAPLES sign

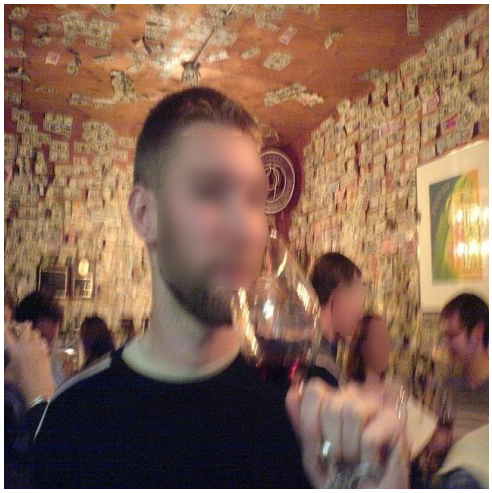
TEXT: "STat"

CONFIDENCE SCORE: 0.12256678193807602

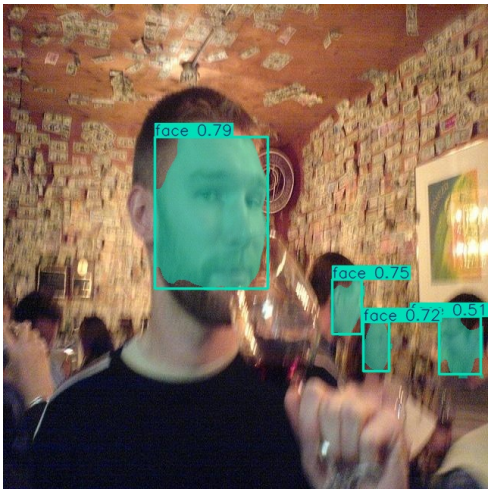
COMPEXITY SCORE: 0.0446096654275093

LLM SCORE: 0.9

PRIVACY SCORE: 0.6750764671048378



(a) Faces with blurring



(b) Original with YOLO detection

Figure 9: Output showing discriminative face blurring depending on clarity of face. (Indistinct faces correctly remain unblurred, even though YOLO detects them.)

(a) Passport with inpainting

(b) Original with YOLO detection

Figure 10: Mostly successful inpainting of passport.