Blurred Lines: Automated Video Obfuscation With Computer Vision

James Varah
Stanford University
jvarah@stanford.edu

Tobias Moser
Stanford University
tobiascm@stanford.edu

Abstract

In this paper, we leverage modern computer vision models to create a novel machine learning pipeline that automatically detects and obfuscates all instances of sensitive information in an inputted video stream. More specifically, we focus on blurring faces and text - the two most common types of inadvertently leaked private information in videos - using a combination of fine-tuned models and an external code framework. On the model front, we experimented with a variety of options but ultimately used Grounding DINO for facial recognition and EasyOCR for text detection. Although the individual performances of these models were not exceptional, when combined with each other they produced a strong pipeline capable of accurately blurring sensitive information across all frames of a video in a reasonable amount of time. This system, while rudimentary, is very functional and represents a strong step towards the sort of privacy-enhancing tools much needed in the modern age.

1. Introduction

For nearly one hundred years following the 1888 invention of the motion capture camera by Louis LePrince, video production was an incredibly costly, burdensome task that required access to expensive equipment and specialized operators. However, towards the end of the 20th century, increased circulation of cheap electrical components led to a surge in personal computing that made video-producing devices like camcorders available widely. At the same time, the internet found its footing and provided a novel mechanism by which users across the world could share information, including their newly abundant self-made videos. Today, this trend has truly reached its maturity. Those early, clunky camcorders have been replaced by smartphones with high-definition cameras, and clumsy attempts at internet-based video sharing has transformed into billion-dollar behmonths like YouTube and Instagram that have millions of new videos uploaded daily. While many benefits have been reaped from this current system, serious issues are present as well. In particular, the prevalence of video sharing has introduced profound privacy concerns. Individuals are now frequently recorded, both with and without their knowledge or consent, in many of the public and private spaces in which they live their lives. Furthermore, these recordings can be subsequently disseminated to billions of people worldwide within just minutes of their creation. In such an environment, it should be no surprise that sensitive data ranging from faces to license plates to private information on papers or computer screens is constantly being inadvertently leaked to the internet via posted videos, opening

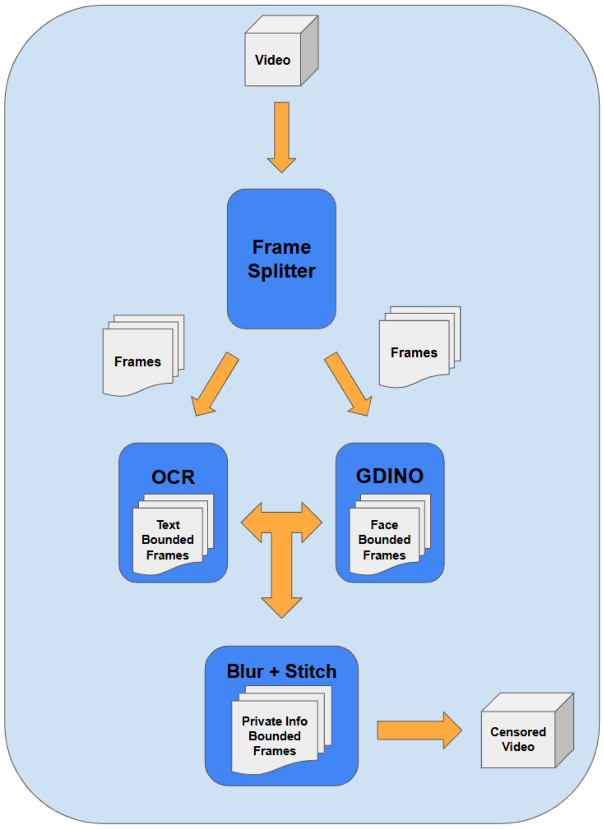

Figure 1: Our proposed pipeline to blur text and human faces from videos.

the door to threats like identity theft or doxxing. Unfortunately, the sheer scale of this leakage is such that manual review is at best inefficient and at worst simply impossible. As such, there is a clear and immediate need for automated tools capable of detecting and redacting such sensitive information. To that end, our paper proposes a novel deep-learning pipeline built on state-of-the-art computer vision models that will process live video streams and automatically redact any potentially hazardous information while still maintaining the integrity of the larger video source.

2. Related Work

On the topic of maintaining privacy on images and videos containing sensitive information, Li et al. 2017 compared computer vision methods of obfuscating private information [9]. They found that blocking out a section of an image with gray or the average pixel value was the most effective in maintaining privacy compared to blurring, but users had the most negative reaction to it. Li and their colleagues influenced our work by using blurring techniques rather than blocking out pixels to reduce the negative user perception of our method of protecting sensitive information in videos.

Most recently, Tseng et al. proposed BIV-Priv-Seg, a dataset tailored for privacy-preserving computer vision tasks [16]. It introduces 16 classes of sensitive content specifically captured from the perspective of individuals with blindness or impaired vision. Unlike general-purpose datasets, BIV-Priv-Seg enables us to fine-tune object detectors on privacy-relevant classes. However, as it only contains static images, it does not address the challenges of video-based detection, limiting its utility to single-frame evaluation.

Focusing first on object detection in images, Ren et al. 2015 improved on previous methods in object detection with Faster R-CNN [13] which employs a Region Proposal Network (RPN) convolutional network that predicts both object bounds and scores across the spatial dimension before processing through a classifier on proposed bounding boxes. Redmon et al. 2016 introduced YOLO, a single network that predicts bounding boxes and class probabilities without the use of a separate RPN, which performed much better and faster than previous results [12]. Mask R-CNN, proposed in He et al. 2018, extends the task of object detection to also produce segmentation masks for instances in an image [5]. YOLO and R-CNN become the basis for many models we explored before constructing our final pipeline, and are used in methods for fast-inference face detection in [22] and high-performance face detection [21] and text detection, which we discuss below.

Kang et al. presented T-CNN, a method that enhances object detection in videos by leveraging temporal consistency through the use of tubelets, which include proposals of objects linked across frames [8]. This method uses optical flow to improve per-frame object detections by rescoring predictions across time, thus providing greater accuracy and temporal stability. We incorporate such temporal modeling techniques to improve upon spatial-only detection systems like YOLO.

Text detection and recognition has been a popular area of research, with initial results from Shi et al. 2015 that develop an end-to-end trainable neural network that models text features, sequences, and transcription to extract characters [15]. Later methods produce better results, such as the one proposed in Deng et al. 2019 which uses a two-stage model to detect text in any orientation without the use of human-picked text features [3]. Ye et al. 2020 introduce another method TextFuseNet that uses image feature representations to retrieve individual characters in arbitrary shapes in text found in scenes [20]. EasyOCR [7], the method we use for text detection, employs the method from Shi et al [15] for robust results with fast performance.

Hendrycks and Dietterich found that neural networks such as AlexNet and ResNet classifiers perform much worse when input images are corrupted by common perturbations such as applying noise to the image and reducing contrast [6].

Recent advances in state-of-the-art object detection and instance segmentation include zero-shot methods such as Grounding DINO [10] and Grounded SAM [14]. These models have the advantage over previous CNN-based methods by removing the need for fine-tuning on novel datasets, and we use the scaled down Grounding DINO tiny model for zero-shot face detection in our pipeline.

3. Data

3.1. Dataset Motivations

Due to the unique design of this paper's pipeline, our dataset selection, processing, and usage were all somewhat atypical. There were two main drivers of this situation, a) our use of two separate models for the facial recognition and text detections tasks and b) the fact that we largely leveraged existing, pre-trained models when building our system. As a result of the former, we ultimately chose to have two fully separate datasets, one for each of the respective tasks, while the latter meant that we used only small segments of larger datasets, as we only needed enough annotated data to conduct minimal fine-tuning and to evaluate the final efficacy of our models.

Beyond these constraints imposed by the high-level structure of our project, we also had several more general guiding principles in mind when researching potential datasets. First and foremost was our desire to have our image examples be as messy and realistic as possible. Although this choice undeniably created a more challenging task for both our models and thus contributed significantly to their middling performances (more on this in Section 4 and Section 5), it was crucial to us that our eventual pipeline was able to work on largely unconstrained images and videos, since these are the exact sort of "ugly" inputs one would expect as part of our system's use cases, which include things like user-provided social media posts or live video streams. In addition to our focus on using real-world data, we also adhered to more standard goals like trying to find datasets with accessible formatting, clean documentation, and reputable sources (i.e. respected researchers, academic or professional institutions, etc).

3.2. Facial Recognition Dataset

When locating our facial recognition dataset, we investigated a wide range of potential candidates including FaceScrub, VGGFace, and CelebA before eventually narrowing down our list to WIDER FACE [19] and Labeled Faces In The Wild (LFW) by evaluating the datasets against the metrics described above. We initially were predisposed towards LFW due to its apparent emphasis on realistic images (hence "in the wild"), but after some additional research and manual review, we found that far too many of the LFW examples were overly "clean" and consisted of front-facing facial portraits in good lighting. As a result, we ultimately chose to use WIDER FACE for the fine-tuning and evaluation of our facial recognition model. This dataset is itself a face-specific subset of the Shenzhen Institute of Advanced Technology's earlier Web Image Dataset For Event Recognition (WIDER), and is comprised of 32,203 images of varying size in JPEG format with 393,703 bounding-box labeled faces. Since, as previously mentioned, our project did not require a large volume of inputs, we took a further subset of 6,000 images from WIDER FACE and used 5,000 of them for hyperparameter tuning and reserved the remaining 1,000 as our test set. The only dataset preprocessing we conducted was to convert the dataset from its native annotation structure to the COCO format to simplify the model evaluation and metric generation process.

3.3. Text Detection Dataset

To settle on our text detection dataset, we followed a near identical process as above. In particular, we considered a large variety of in-the-wild text segmentation datasets, from Sythtext to various ICDAR Robust Reading sets to UC San Diego's Street View Text before eventually settling on COCO-text [17] as our final dataset due to its diversity of orientation, lighting, font size, and text languages. This dataset, much like WIDER FACE, is a text-focused subset of the previously created MSCOCO dataset and was first released by Cornell University in 2016 (link). In its original form, COCO-test contains 63,686 JPEG images with 239,506 annotated text instances (bounding boxes + transcription). As before, we preprocessed this dataset to fit our project scope by taking a further subset of 17,000 images randomly selected from the original dataset, using 14,000 for fine-tuning and keeping the remaining 3,000 to be used as the test set. All corresponding annotations were already in a COCO formatted JSON file and thus further modifications were not needed.

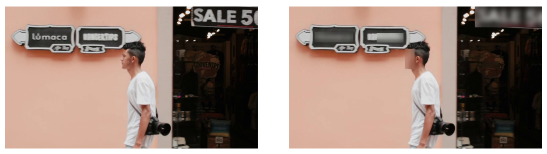

Figure 2: Before/after of our blurring technique shown on one video frame. Left: before, Right: after

3.4. Video Sourcing

For simplicity's sake and due to time constraints, we decided to use image-based models in our pipeline rather than video-specific constructions like SlowFast Networks or C3D, and thus our code interprets an inputted video not as a video but rather as a sequence of individual images in the form of frames. Because of this unique set up, we had no need to use a third dataset of just videos to evaluate the cumulative pipeline, but instead could simply test the facial recognition and text detection models on their own respective datasets separately. However, to qualitatively evaluate our final product, we did source several short videos from pexels.com to ensure the functionality was as desired.

3.5. Our Pipeline

We propose a video processing pipeline that blurs text objects and faces in a diverse set of scenes. First, we divide the input video into frames to process for blurring. We use the Python implementation of OpenCV [2] to read individual frames and output JPG image files. Despite sacrificing temporal information for the inputs to later parts of the pipeline, using images rather than video segments allows a wider variety of possible methods to be used for object detection. Additionally, sampling of images makes it possible for future work on our pipeline by sampling video frames and interpolating face and text detections between frames for faster inference.

After processing an input video into component video frames, we pass the image into a sequence of two modules. The first module creates bounding boxes around detected text in the image, and the second module creates bounding boxes around detected faces in the image. Then we compose all of the bounding boxes detected for that image, and obfuscate regions by applying a Gaussian blur with standard deviation 10, and the following expression for the kernel width and height (w = width, h = height):

$$w \leftarrow \begin{cases} w+1, & \text{if } w \bmod 2 = 0 \\ w, & \text{otherwise} \end{cases}$$
$$h \leftarrow \begin{cases} h+1, & \text{if } h \bmod 2 = 0 \\ h, & \text{otherwise} \end{cases}$$
$$w \leftarrow \max(13, w)$$
$$h \leftarrow \max(13, h)$$

We use this expression for kernel width and height to maintain both an odd number, while maintaining a minimum width and height of 13 pixels to blur regions to be difficult for humans to recognize. After processing all image frames, we use OpenCV VideoWriter to take input blurred JPG files and write out to an H264 compressed .MP4 video file. Our code contribution is the surrounding pre- and post-processing for videos, as well as how we obfuscate frames based on bounding boxes, and we use existing code for inference for object detection after we fine-tuned the models and parameters for our use case.

3.6. Text Detection Module

For text detection, we employ the EasyOCR framework [7], which has very fast inference and good performance on diverse text in scene detection. Using the English version of the EasyOCR reader, we obtain bounding boxes around text and its component characters along with confidence scores. The EasyOCR framework provides flexibility for what models are used in the two main components of text recognition. We used the following combination of models for the components of EasyOCR: Detection for areas of text. We choose the Character Region Awareness for Text Detection (CRAFT) method, introduced by Baek et al in 2019 [1], a neural network that detects potential text areas. This method generalizes well to text regions with arbitrary shape by using both text level and character level annotations on synthetic and real world images and modeling affinity between characters. Detection of specific text instances and character recognition. This component uses the model described in Shi et al [15] which extracts features from the text regions with a ResNet, models sequences with an LSTM, and then transcribes characters with an RNN [4]

Since we do not need recognition results to obfuscate text in videos, we disregard the text recognition results and only use the identified text bounding boxes.

3.7. Face Detection Module

For face detection, we use Grounding DINO tiny [10] using the text prompt "face." which is installed via the HuggingFace transformers Python package [18]. Since Grounding DINO has state-of-the-art zero-shot performance on datasets such as 52.5 Average Precision (AP) on COCO zero-shot, we expect it to generalize well to a variety of scenes and to further object classes if we expand our pipeline to obfuscate more categories without the need for re-training models on different sets of classes. However, we also want to balance this quality of annotations and generalizability with speed, so we chose the smaller "tiny" model with 172M parameters rather than the base model with 233M parameters. Other methods for face generation and datasets often focus only on subjects directly facing the camera and do not generalize to in-the-wild scenes.

3.8. Testing for Model Resiliency to Image Perturbations

Due to research by Michaelis et al 2019 [11], we also explore resiliency of our model to image perturbations that do not impede human recognition of subjects. If we deploy our pipeline to blur sensitive information from live video streams, for example, it is possible for a malicious actor to perturb the input footage and avoid text and face detection from EasyOCR and Grounding DINO used in the pipeline. We explore object detection accuracy after applying gaussian noise to images using the code provided in Michaelis [11].

4. Experiments

For our hyperparameters on EasyOCR, we chose the ResNet/LSTM/RNN model combination for the faster inference time compared to a transformer based feature extractor and sequence encoder, with still meaningful accuracy results. We also chose to use bounding boxes without a score threshold which resulted in the best results on our COCO-Text validation dataset.

The hyperparameters for Grounding DINO were a box threshold of 0.2 and a text threshold of 0.2, which we found empirically after gathering accuracy metrics on our 5,000 image validation set combined with qualitative confirmation of avoiding duplicate bounding boxes for the same object instance while still detecting instances with small area in the image. We used a text prompt of "face." as input for the text grounding information, which we used rather than "person" or "head" to avoid blurring all heads and persons, which are not strictly required in protecting the identity of a person in a video stream.

Our primary metrics are Average Precision (AP) and Average Recall (AR). We define these using Intersection over Union (IoU) of ground truth data with our face dataset for Grounding DINO, and with ground truth data with our text dataset for EasyOCR. True Positives (TP) are defined as predicting bounding boxes measured at a specific IoU threshold compared to a ground truth bounding box. False Positives (FP) are predicted bounding boxes that do not meet the IoU threshold or match an already assigned ground truth box. Finally, False Negatives (FN) are recorded when

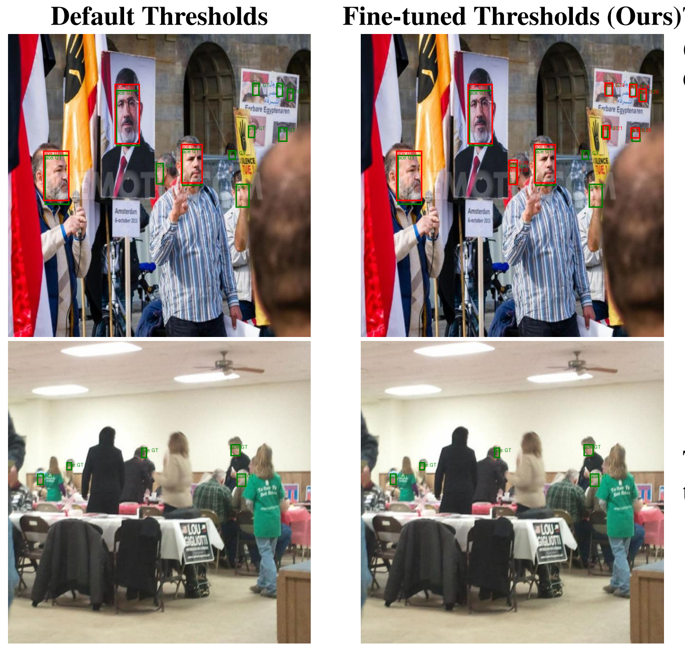

Figure 3: Visual comparison of results from fine-tuning thresholds. Left: default (box_threshold=0.4, text_threshold=0.3); right: fine-tuned (box_threshold=0.2, text_threshold=0.2). Ground truth in green, predictions in red.

a ground truth bounding box is not matched to any prediction above the IoU threshold. We calculate AR and AP over a maximum of 100 detections per image.

With TP, FP, and FN being measured across the whole dataset, AP is defined as $\frac{TP}{TP+FP}$ and AR is defined as $\frac{TP}{TP+FN}$.

We record the mean of AP and AR across IoU thresholds between 0.50 and 0.95 in steps of 0.05 for our final metrics, and define this as AR_{all}. Additionally, we use more fine-grained measures of small, medium, and large, defined by limiting the metrics to measuring bounding boxes based on area according to the following relationship defined based on square pixel area of the box:

$$\text{Object Size Category} = \begin{cases} \text{Small} & \text{if area} < 32^2 \\ \text{Medium} & \text{if } 32^2 \leq \text{area} < 96^2 \\ \text{Large} & \text{if area} \geq 96^2 \end{cases}$$

4.1. Face Detection Results

We tried many different box and text thresholds for Grounding DINO detection on WIDER FACE (see figure 4). The best results were at box threshold = 0.2 and text threshold = 0.2, which showed a significant improvement in all our AP and AR metrics compared to the default box threshold = 0.4 and text threshold = 0.3. One potential concern with lowering the threshold is the increase of false positives, but our improved AP metric after this reduction in both thresholds addresses this concern. The default thresholds avoid duplicate detections on the larger faces, but miss the smaller faces present in the image. However, our thresholds, while there is some duplication in bounding boxes on faces, leads to many more correct detections, especially in the small faces present in the image. Despite the improvement, both sets of hyperparameters fail to detect faces whose ground truth bounding boxes have small area and are present in already low resolution images.

Table 1: Comparison of AP and AR metrics (@IoU=0.50:0.95, maxDets=100) between clean and Gaussian noise perturbed predictions.

Metric	Clean	Perturbed (Gaussian noise)
AP All	0.241	0.160
AP Small	0.121	0.068
AP Medium	0.550	0.409
AP Large	0.581	0.458
AR All	0.275	0.197
AR Small	0.149	0.087
AR Medium	0.604	0.474
AR Large	0.668	0.584

Table 2: AP and AR (@IoU=0.50:0.95, maxDets=100) for text detection on the COCO-text validation set.

Size	AP	AR
All	0.073	0.129
Small	0.062	0.096
Medium	0.109	0.204
Large	0.185	0.354

Figure 1 shows the results from running Grounding DINO for face detection on a perturbed input image, compared to the baseline. Both AP and AR metrics are drastically worse when images are corrupted using gaussian noise, which is consistent with earlier findings in Hendrycks et al., [6] extending weaknesses in detection models to Grounding DINO.

Figure 5 shows a qualitative comparison of the low accuracy of Grounding DINO on the same image with gaussian noise.

4.2. Text Detection Results

In contrast to Grounding DINO's facial recognition performance, EasyOCR produced much less impressive results 2 when evaluated using the criteria described above. However, despite these initially poor findings, additional evaluation revealed a degree of nuance in our model's performance. First, we found, upon further investigation, that the ground truth labels in the COCO-text dataset were consistently imperfect; for instance, we identified a number of im-

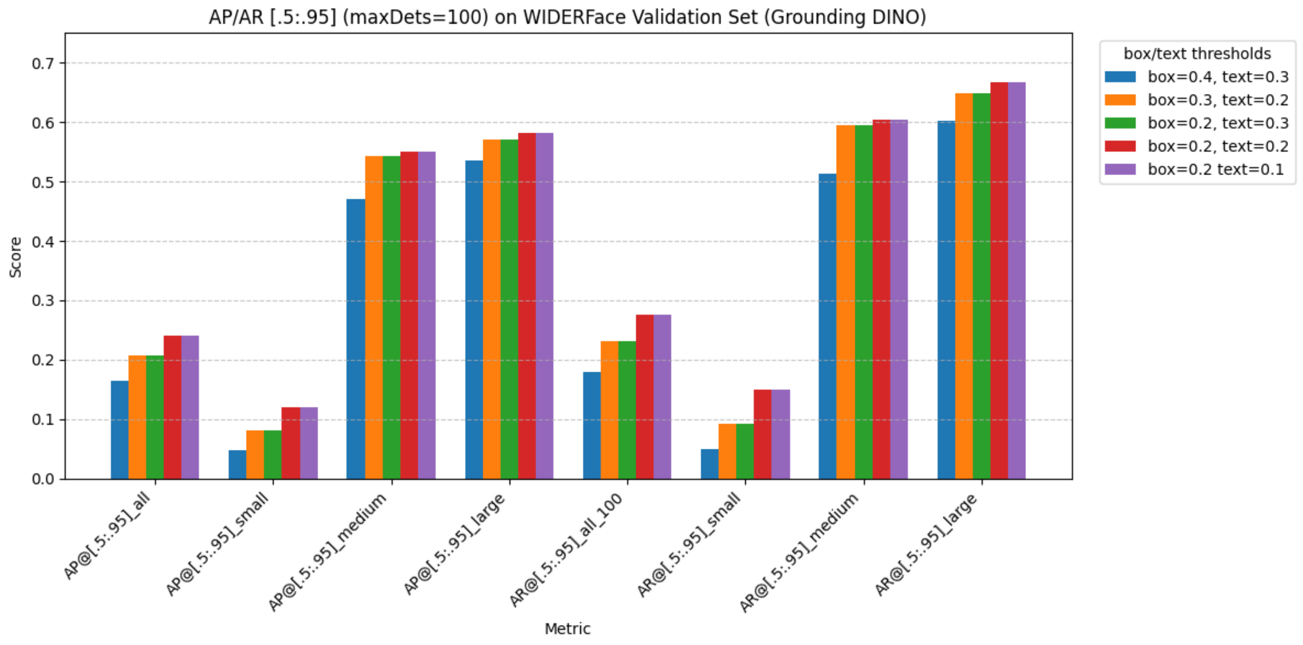

Figure 4: Fine tuning of thresholds for face detection

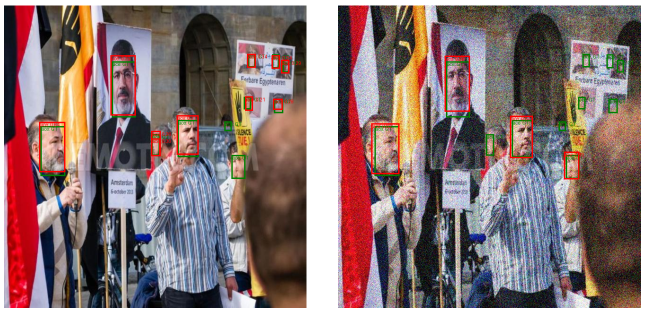

Figure 5: Left: Face detections with fine-tuned threshold. Right: Image with Gaussian noise. Ground truth in green, detections in red.

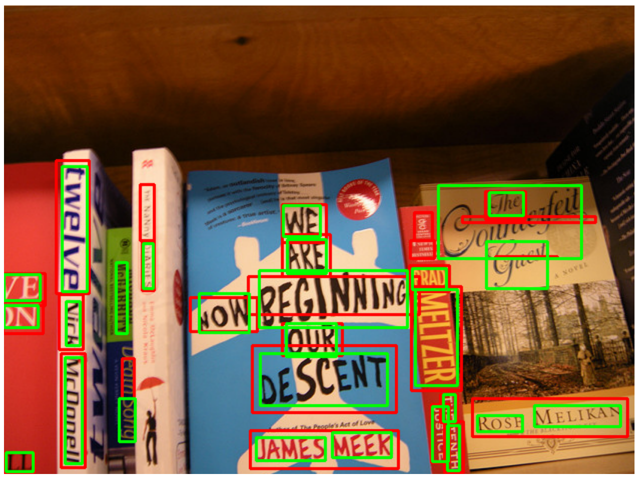

Figure 6: An example of the mismatch between EasyOCR outputs (red) and COCO-text's ground truth annotations (green). Note the lack of COCO-text bounding boxes on some text regions that EasyOCR successfully bounds, as well as EasyOCR's struggles with rotated text.

ages throughout our subset of the data in which the annotation bounding boxes completely missed seemingly obvious segments of text. On many of these examples, our EasyOCR model actually produced accurate bounds on the unlabeled text instances, bounds that were empirically evidence of the model performing well but that were subsequently seen as mistakes when compared to the flawed annotations. In a similar vein, we also observed several images for which EasyOCR produced a single bounding box that successfully covered the text instance, but for which the corresponding ground truth annotation consisted of multiple more granular bounding boxes. Once again, for the purposes of our pipeline, these EasyOCR outputs wuld be considered successes but would nonetheless damage the model's performance as determined by metrics like IoU. This, of course, is not to say the model was perfect. It repeatedly struggled to bound text instances that were skewed or tilted, and also found small, non-English texts.

Figure 7 provides a perfect example of all these properties. Finally, it's important to understand that while EasyOCR has much lower AP and AR metrics on text com-

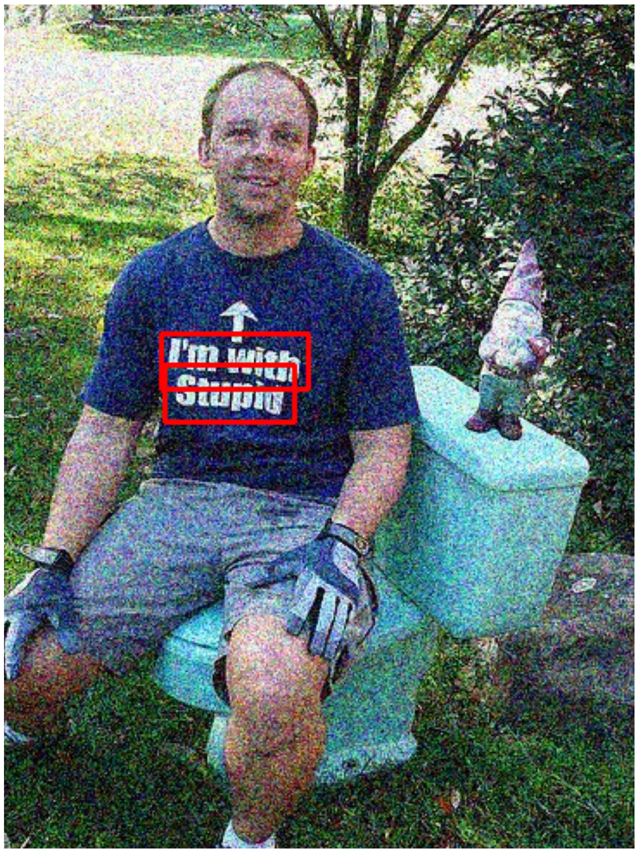

Figure 7: EasyOCR accurately detects text even with perturbation. Predictions in red, ground truth not shown.

pared to Grounding DINO on faces, it still outperformed Grounding DINO when prompted to find "text.". Due to the complexity and generalizability of the network, it performs worse than purpose-built text detectors like Easy OCR.

4.3. Final Pipeline Output

Overall, the individual model performance of Grounding DINO and EasyOCR on their respective tasks was, despite less than perfect metrics, more than sufficient for the purposes of our project. When tested on a wide selection of input images from both the WIDER FACE and COCO-text datasets, our models and obfuscation code successfully blurred nearly all face and text instances we could observe. Following this, we ran the entire pipeline, including the frame splitting and stitching code, on several videos with relevant content (i.e faces and text) that we pulled from the previously mentioned pexels.com. On every video, the pipeline performed admirably, and we were not able to observe any textual or facial details from the resulting obfuscated videos see Figure 2.

5. Conclusion

In this paper, we successfully created a novel pipeline capable of automatically detecting and obfuscating all instances of sensitive information in an inputted video stream. By using a combination of Grounding DINO for facial recognition and EasyOCR for text detection, along with some manual fine-tuning on robust, diverse datasets, we were ultimately able to achieve a relatively high degree of functionality on our use case. Although there were certainly some issues with individual model performances (some of which can be attributed to dataset-specific issues, as mentioned previously), our code nonetheless remains a powerful if rudimentary tool that could, with only minor modifications, be used in a real-world setting to enhance the privacy of modern internet users.

While this system provides a solid foundation, there are several clear avenues for improvement and future research. The first and most significant amongst these is to work on increasing the pipeline's overall speed. This could be achieved with improvements on several fronts, such as replacing Grounding DINO with a light-weight model designed specifically for facial recognition or combining the facial recognition and text detection models into a single model capable of identifying both classes. One particularly promising modification would be to only produce bounding boxes on every Nth video frame across the entire video, then use an additional model (likely some variety of an RNN) that takes in these bounded frames and predicts bounding boxes on all the unprocessed intermediate frames between them. Another possible avenue for future work would be to enhance the existing functionality of the model, perhaps by training a more advanced text detection system that blurred only text containing potentially sensitive information like addresses and phone numbers while leaving more benign instances unmodified.

Overall, the proposed system, while simple, provides a solid foundation for privacy-preserving technologies in video processing and can hopefully serve as a springboard for future advances and research designed to help make the modern internet a safer place for all its users.

References

[1] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee. Character region awareness for text detection. 2019.

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] L. Deng, Y. Gong, Y. Lin, J. Shuai, X. Tu, Y. Zhang, Z. Ma, and M. Xie. Detecting multi-oriented text with corner-based region proposals. *Neurocomputing*, 334:134–142, 2019.

[4] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks, 2006.

[5] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. 2018.

[6] D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations, 2019.

[7] JaidedAI. Easyocr, 2024. Accessed: 2025-06-04.

[8] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, W. Ouyang, X. Liu, and X. Wang. T-cnn: Tubelets with convolutional neural networks for object detection from videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(10):2896–2907, 2018.

[9] Y. Li, N. Vishwamitra, B. P. Knijnenburg, H. Hu, and K. Caine. Blur vs. block: Investigating the effectiveness of privacy-enhancing obfuscation for images. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1343–1351, 2017.

[10] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, and L. Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. 2023.

[11] C. Michaelis, B. Mitzkus, R. Geirhos, E. Rusak, O. Bringmann, A. S. Ecker, M. Bethge, and W. Brendel. Benchmarking robustness in object detection: Autonomous driving when winter is coming. *arXiv preprint arXiv:1907.07484*, 2019.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[13] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. 2016.

[14] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan, Z. Zeng, H. Zhang, F. Li, J. Yang, H. Li, Q. Jiang, and L. Zhang. Grounded sam: Assembling open-world models for diverse visual tasks. 2024.

[15] B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition, 2015.

[16] Y.-H. H. Tseng, L. Wang, H. Wu, V. Ramanathan, and L. Fei-Fei. BIV-Priv-Seg: Locating Private Content in Images Taken by People With Visual Impairments. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 430–440, 2025.

[17] A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images. 2016.

[18] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

[19] S. Yang, P. Luo, C. C. Loy, and X. Tang. Wider face: A face detection benchmark. 2015.

[20] J. Ye, Z. Chen, J. Liu, and B. Du. Textfusenet: Scene text detection with richer fused features. In C. Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 516–522. International Joint Conferences on Artificial Intelligence Organization, 7 2020. Main track.

[21] S. Zhang, C. Chi, Z. Lei, and S. Z. Li. Refineface: Refinement neural network for high performance face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):4008–4020, 2021.

[22] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li. Faceboxes: A cpu real-time face detector with high accuracy. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–9, 2017.