# Real-Time Video Anonymization in Smart City Intersections

Alex Angus, Zhuoxu Duan, Gil Zussman, Zoran Kostić
Dept. of Electrical Engineering, Columbia University, New York City
{ala2197, zd2235, gz2136, zk2172@}@columbia.edu

***Abstract*—Video cameras in smart cities can be used to provide data to improve pedestrian safety and traffic management. Video recordings inherently violate privacy, and technological solutions need to be found to preserve it. Smart city applications deployed on top of the COSMOS research testbed in New York City are envisioned to be privacy friendly. This contribution presents one approach to privacy preservation – a video anonymization pipeline implemented in the form of blurring of pedestrian faces and vehicle license plates. The pipeline utilizes customized deep-learning models based on YOLOv4 for detection of privacy-sensitive objects in street-level video recordings. To achieve real time inference, the pipeline includes speed improvements via NVIDIA TensorRT optimization. When applied to the video dataset acquired at an intersection within the COSMOS testbed in New York City, the proposed method anonymizes visible faces and license plates with recall of up to $99\%$ and inference speed faster than $100$ frames per second. The results of a comprehensive evaluation study are presented. A selection of anonymized videos can be accessed via the COSMOS testbed portal.**

***Index Terms*—Smart City, Sensors, Video Surveillance, Privacy Protection, Object Detection, Deep Learning, TensorRT.**

## I. Introduction

Smart cities are envisioned as societally beneficial constructs. Data and algorithms should be used to make the cities more livable by reducing pollution, increasing pedestrian safety, and supporting efficient traffic management. Cameras and Internet of Things (IoT) sensors are essential for collecting data needed to implement the smart city vision. All publicly sensor-acquired data, and video surveillance in particular, inherently impinge on personal privacy. There is therefore both a technological and a legislative push to minimize, and ultimately eliminate, the distribution of sensitive/private data. Methods for eliminating the distribution of privacy-sensitive data span a variety of technical solutions including: image processing, data meta representation and deletion, encryption, storage of data in edge clouds only, social community based data management, and others [1]–[8].

This paper focuses on a video anonymization method for privacy preservation in the form of blurring of pedestrian faces and vehicle license plates. It is applied to data acquired by street-level video cameras at an intersection within the NSF PAWR COSMOS testbed [9] in New York City.

### A. Privacy Concerns in Smart City Intersections

Real-time street level video camera feeds are required for computer vision based smart city intersection applications. Pedestrian detection, vehicle tracking, and crowd monitoring [9]–[13] rely on high resolution video streams and can benefit from ground floor camera positioning. Sensitive information such as facial features and license plate characters are inadvertently captured in the collection of street level videos. This information can be leaked in downstream applications if there is no intervention before the video distribution stage. Furthermore, to support real-time applications, a privacy protection mechanism should introduce minimal communications and computing latencies. This paper outlines the work towards a deep learning based privacy protection mechanism for the COSMOS pilot testbed intersection at 120$^{\text{th}}$ Street and Amsterdam Avenue at Columbia University in New York City.

### B. COSMOS Testbed

The COSMOS testbed (Cloud Enhanced Open Software Defined Mobile Wireless Testbed for City-Scale Deployment) is an experimentation environment for advanced wireless research [9]. It has sensing, high speed communications, and AI-enabled edge computing capabilities suitable for the development of smart city technologies. The testbed site at 120$^{\text{th}}$ St. and Amsterdam Ave, seen in Fig. 1, includes four high resolution video feeds: two bird's eye view cameras and two street level cameras mounted on Columbia University's Mudd building[1]. The 1$^{\text{st}}$ floor street-level camera located on 120$^{\text{th}}$ St. is used in this work and its field of view is shown in Fig. 2. The COSMOS edge cloud servers are the target deployment platform for our anonymization pipeline. They are equipped with optical x-haul transports allowing for interconnection of AI-enabled edge computing clusters and provide scalable CPU and GPU resources [8].

## II. Related Work

Notable work has been done both in the area of large scale video anonymization and in real-time object detection. Advances in object detection, such as the adoption of convolutional neural networks (CNN) and vision transformers, have significantly improved the ability to locate and remove sensitive information in images [14]–[18]. However, contemporary models often perform poorly on small objects and operate at speeds well below real-time. Commercial systems have been developed for real-time anonymization, but they are expensive, inflexible, and are not a good fit for use cases

[1]The two cameras on the right of Amsterdam Ave. are planned and not yet in use.

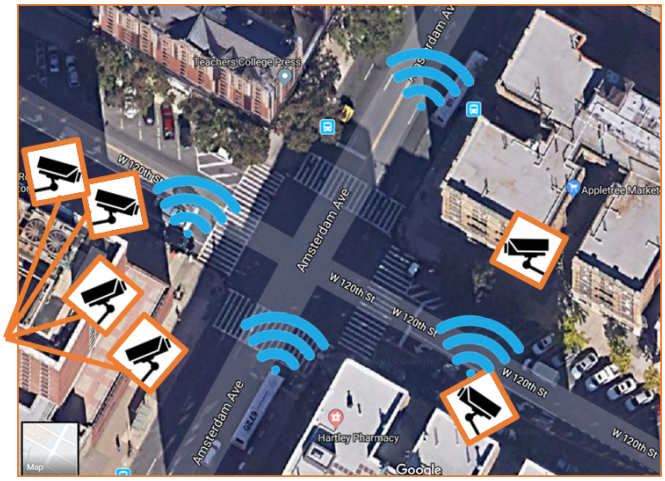


Fig. 1: The intersection of 120th St. and Amsterdam Ave. in New York City, where COSMOS cameras and edge-cloud nodes are deployed.


Fig. 2: (A) First floor street-level camera view, 120th St. and Amsterdam Ave; (B) non-anonymized input frame with faces and license plates exposed; (C) output frame with anonymized faces and license plates.

such as described in this study. For example, [19] and [20] provide proprietary anonymization API services, but they are not accessible for experimentation and do not support $4K$ resolution video. Additionally, API calls introduce uncontrollable network latencies that make it difficult to operate under the real-time target of 30 frames per second (FPS), or 33 ms, for (i) sensor data acquisition; (ii) communication between end-users, sensors, and edge cloud; (iii) AI-based inference computation; and (iv) providing feedback to participants in the intersection [8]. To avoid API calls over the network, on-premises commercial software anonymization systems such as [21] are available for lease, but the inflexibility remains as only face anonymization is supported. Furthermore, general anonymization products [22] provide only vague or inadequate performance assessments such as "96% accuracy".

### A. Large Scale Video Anonymization

One of the earliest works towards automatic privacy protection is the anonymization of the Google Street View footage [23] which utilized classical computer vision sliding window techniques with postprocessing designed to maximize the detection recall. This method blurred 89% of faces and $94-96\%$ of licenses without using contemporary deep learning techniques. Other designs use Haar-like features, cascade classifiers, and CNNs to locate faces and license plates [24], [25]; [26] reports license recall of 94.5% and inference times within 100 ms. Similar to our approach, [27] uses a one stage CNN-based YOLOv3 detector for locating license plates in various complex scenes with reported mean average precisions (mAP) over 90%.

### B. Real Time Object Detection

In addition to high recall video anonymization, we aim for pipeline inference speed under 33 ms to match the camera frame capture rate of 30 FPS. This is challenging for the following reasons: (i) reliable small object detection requires large video input resolution; (ii) contemporary video compression and encoding implies at least one-frame buffering; (iii) networking latencies can be significant; (iv) the fastest streaming methodologies such as WebRTC promise on the order of 200-300 ms latencies; (v) computational complexity of deep learning models is non-trivial [28]–[31]. Several studies have been dedicated to real time use cases of YOLOv4 [32]–[35], but only with input resolutions smaller than $832 \times 832$. Techniques such as optimal scheduling for heterogeneous computing devices [36] and weight pruning [37] have been shown to incur only minimal decreases in accuracy with significant increases in throughput for YOLOv4. Other strategies such as reduced floating point precision calculations, layer fusion, and kernel auto tuning are successfully implemented for YOLOv4 [38], [39] using NVIDIA's TensorRT [40] framework. This research utilizes TensorRT to minimize inference latencies without changing the underlying structure of our customized YOLOv4-based anonymization models.

## III. Methodology

### A. COSMOS Intersection Dataset

This study uses a subset of the COSMOS 1st floor intersection dataset, consisting of 16 videos recorded from the view seen in Fig. 2. Each video is 180 seconds long and recorded at a frame rate of 30 FPS with resolution of $3840 \times 1920$ pixels. Recordings were selected with a variety of weather and lighting conditions including daytime, nighttime, cloudy, sunny, and rainy conditions. Every sixth frame was selected for annotation, to decrease the similarity between sequential frames, for a total of approximately 900 annotated frames per video. Bounding box annotations of 9 classes, including faces and license plates, were generated using the browser based Computer Vision Annotation Tool (CVAT) [41]. In total, the

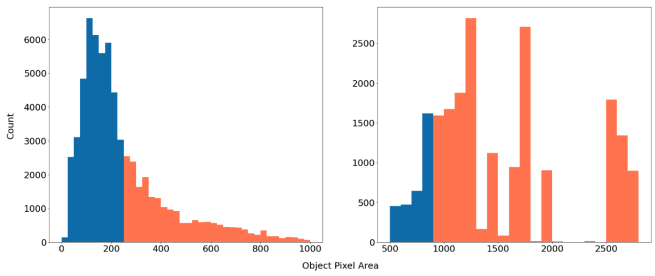


Fig. 3: Distributions of face (left) and license plate (right) areas below (blue) and above (orange) the visible area thresholds.

dataset comprises over 14,000 annotated frames with 70,186 faces and 124,614 license plates. Fig. 3 shows the ground truth bounding box area distributions for both classes. The median object pixel area is 198 pixels for faces and only 83 pixels for license plates in the original $4K$ video resolution. Note that most of the object areas are significantly below the COCO small object threshold of $32 \times 32$ pixels [42]. This requires us to use detection models with large input resolutions to preserve as many object features as possible.

### B. YOLOv4 Object Detection

*1) YOLOv4:* The YOLO (You Only Look Once) object detection algorithms are single stage models that simultaneously detect, localize, and classify objects in an image. They are designed to be accurate and fast; each version of YOLO has improved the previous inference speed benchmark. This work is based on YOLOv4, which consists of the CSPDarknet53 backbone, SPP and PAN necks, and the YOLOv3 head [32]. The CSPDarknet53 backbone generates feature maps at different scales, the SPP and PAN necks collect the feature maps, and the YOLOv3 head uses them to generate predictions. YOLOv4 is trained with the CIoU loss function, DropBlock regularization, and a plethora of photometric and geometric data augmentations. At the time of this work, model code for a "YOLOv5" was circulated and rumored to be better performing than YOLOv4 in terms of accuracy and speed, but the validity of the benchmark results was yet to be confirmed.

*2) Darknet Training:* For training on our custom intersection dataset, we use the Darknet framework [43]. Darknet is an open source neural network framework written in CUDA and C, and it is the original framework of the YOLOv4 model development. This simplifies training, experimentation, and modifications to YOLOv4. We train YOLOv4 models to detect two classes of objects, faces and license plates, with model input sizes of $608 \times 608$, $960 \times 960$, and $1440 \times 1440$. Our models are trained for 10,000 iterations each on a single T4 GPU for the $608 \times 608$ and $960 \times 960$ models and a single A100 GPU for the $1440 \times 1440$ model. The CSPDarknet53 backbone is pre-trained on the MS COCO dataset [44]. Data augmentation techniques, including CutOut, MixUp, CutMix, and Mosaic, are applied during training as described in [32].

We experimented with aggregating, then randomly shuffling and splitting all frames to generate the training and validation sets. We found that significant model overfitting can occur because stationary objects such as parked vehicles and seated pedestrians appear identically in consecutive frames. To reduce the models' tendency to overfit to stationary objects, we use 14 separate videos for training and 2 for validation. We use the weights with the highest validation set mean average precision (mAP) to maximize the generalization of our models.

### C. Pipeline Integration

After our models are trained using Darknet, the fine-tuned parameters are transferred to a PyTorch implementation of the YOLOv4 model architecture. Accuracy performance measurements, including mean average precision (mAP) and class recall, are recalculated on the validation set using the PyTorch model to verify a successful framework transfer. See Fig. 5 for validation set average precision performance by model input resolution. The Darknet model parameters are also transferred to a TensorRT YOLOv4 implementation to maximize the inference throughput. Details of the accuracy and latency evaluations are outlined in Sections IV-A and IV-B. The PyTorch and TensorRT face and license plate detection models are integrated into the anonymization pipeline, which is composed of the following:

- **Video frame read:** Frames are read sequentially from a MP4 video file using OpenCV VideoCapture.
- **Frame preprocessing:** Frames are resized to the model input resolution and transferred to GPU memory in tensor batches.
- **Model inference:** Preprocessed tensor batches are passed through the YOLOv4 model.
- **NMS:** Non-maximum suppression is performed on the output bounding box predictions.
- **Frame postprocessing and anonymization:** Output predictions are read from the GPU and frames are passed through the anonymization module.
- **Video frame write:** Frames are written sequentially to an output MP4 video file using OpenCV VideoWriter.

The pipeline is implemented in Python for PyTorch models and in C++ for TensorRT models.

*1) Anonymization Module:* The anonymization module takes the original non-anonymized frame and the detection coordinates as input, and it returns an anonymized frame as output. The simplest anonymization strategy is to completely mask detected areas. This effectively removes the sensitive information from the frame, but it obscures the appearance of intersection objects and may reduce the effectiveness of downstream video tasks. Applying a Gaussian blur to the detected areas effectively removes sensitive information from the frame without changing the overall appearance. We opt for this technique in our anonymization pipeline as it has been shown that blurring obfuscations result in only slight decreases in computer vision tasks [45]. An example output frame anonymized with the Gaussian blur module is depicted in Fig. 2. In applications where a Gaussian blur removes too many object features to be useful, the facial features and license plate characters must be obfuscated such that privacy

is protected with the underlying feature distribution remaining constant. In such a scenario, GAN based privacy protection models such as [46] could replace the mask anonymization and blur anonymization modules.

*2) TensorRT Model Implementation:* The YOLOv4 model architecture includes 110 convolutional layers and over 60 million parameters. The large computational cost of the forward pass during inference, which increases exponentially with input resolution, makes real-time anonymization a challenge. Offloading parallel inference calculations from the CPU to the GPU lowers the inference speed by orders of magnitude. But, even these speedups are not sufficient for large input models running on lower tier GPUs. We show that further inference optimizations including layer and tensor fusion, kernel auto tuning, and reduced precision floating point calculations with NVIDIA's TensorRT framework [40] can reduce the inference time to facilitate real-time anonymization.

To convert our trained models from Darknet to TensorRT, we reconstruct the YOLOv4 model architecture using the TensorRT C++ API and load the model parameters layer by layer during the serialization phase as illustrated in Fig. 4. After model parameters are loaded into the model, TensorRT automatically performs several experiments to maximize the throughput of the inference engine for the specific pipeline configuration which includes GPU architecture, floating point precision, batch size, model input size, etc. See Table II for example pipeline configurations and timing measurements. In the deserialization phase the optimized TensorRT model is loaded into memory. Inference can commence once memory buffers are allocated for the input frames and output predictions.

## IV. Evaluation Results

This section presents the results of accuracy evaluations for different object size thresholds and latency evaluations for both PyTorch and TensorRT anonymization pipelines. A selection of anonymized videos can be accessed via the COSMOS testbed portal [47].

### A. Accuracy Evaluation

Models used for privacy critical applications must correctly identify the highest possible number of true positives (TP), and the metric that is most indicative of this performance is the recall [23] $\frac{TP}{TP+FN}$. In our case a true positive corresponds to a visible, detected, and anonymized face or license plate in one frame. A false negative (FN) corresponds to one which is missed, where *visible* is defined below, and *missed* is defined as more than half the object remaining non-anonymized. As anonymization in our pipeline occurs within the bounding box detection, this definition of missed objects, or false negatives, corresponds to a bounding box intersection over union (IoU) threshold of 0.5. To maximize anonymization recall we set the detection confidence and NMS thresholds to 0.0125 and 0.4, respectively.

Not all faces and license plates in a given frame are identifiable. For example, the faces and licenses in Fig. 6 show objects captured far away from the camera. The resulting images are low resolution, and we posit that sufficient information cannot be gleaned to identify such objects as specific faces or license plates[2]. Therefore, we define *visible* objects as faces and license plates with bounding box areas larger than 250 pixels and 900 pixels, respectively, in the original video frame. Fig. 6 shows faces and licenses below the *visible* pixel area thresholds and Fig. 3 shows the distribution of visible objects. Note that these thresholds are conservative benchmarks for identification – many faces and licenses above these thresholds also cannot be identified by the eye.

*1) Programmatic Accuracy Evaluation:* The fidelity of our anonymization pipeline depends on consistent and accurate face and license plate detection. Furthermore, the focus of our accuracy evaluation is on objects deemed identifiable: faces where facial features can be discerned ($\geq$ 250 pixels) and license plates where characters can be recognized ($\geq$ 900 pixels). Omission of objects below the visible threshold reduces the number of objects considered for evaluation. We exclude these objects because they are irrelevant for privacy protection, not because the models are incapable of detecting them. Notably, the $1440 \times 1440$ model scores over $96\%$ mAP for objects larger than 100 pixels, which is far smaller than the visible threshold for both faces and license plates (Fig. 5).

Fig. 7 shows plots of validation recall as a function of pixel area threshold. As the pixel area threshold is increased, a smaller number of objects are considered in the evaluation as only objects with large pixel areas, or cross sections, remain. Objects with large cross sections are detected, and ultimately anonymized, with a higher recall than those with smaller cross sections. As expected, the models with large input resolutions ($960 \times 960$ and $1440 \times 1440$) yield higher recalls than the smaller input resolution model ($608 \times 608$) for both faces and license plates. Fig. 8 gives an overview of the detection performance on the validation set for both classes with the $1440 \times 1440$ model at $98.9\%$ face and $99.9\%$ license plate recall for visible objects. We note that face recall is more strongly affected by model input resolution than license plate recall in the COSMOS 1st floor intersection dataset.

*2) Manual Accuracy Evaluation:* High quality custom annotated datasets are necessary for the fine tuning of supervised deep learning models, but they require an immense amount of labor to produce. Ground truth labels are therefore scarce and must be prioritized for training. This scarcity limits the extent of programmatic evaluations that can be performed, and we must use other techniques such as manual evaluations to increase the confidence in our results. In this section we outline our approach to and results of manual evaluation of the anonymization pipeline output. In short, we pass a set of intersection videos through the pipeline, then visually inspect the output for missed objects, where the definition of missed remains the same as in Section IV-A.

When performing manual evaluations, it is impractical to adhere to the pixel area threshold definitions. One would

[2]Although there exist techniques for recovering information from very low resolution images [48], these methods are outside the scope of this work.

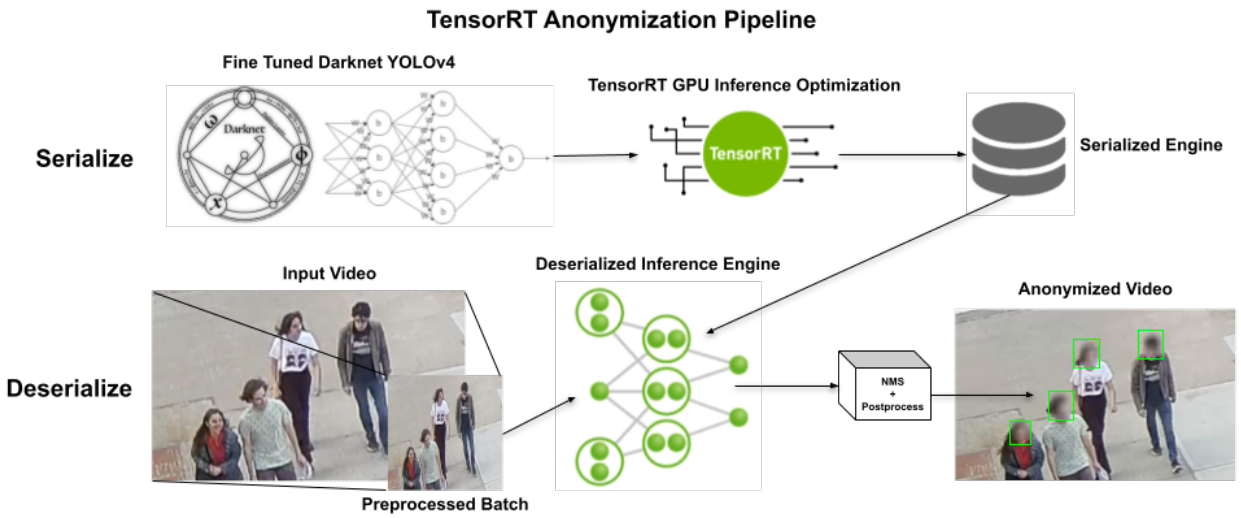


Fig. 4: The TensorRT Anonymization Pipeline. Inference optimization calculations are performed in the serialize phase to produce the serialized inference engine. In the deserialize phase videos are preprocessed and passed through the inference engine for detection and subsequent anonymization.

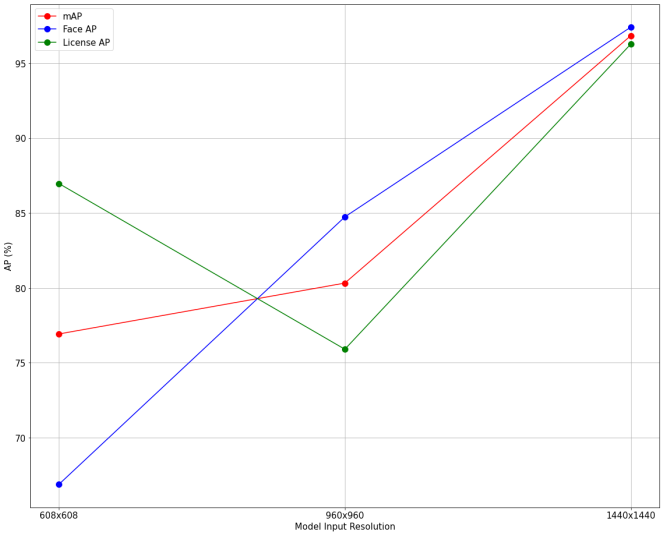


Fig. 5: Average precision by class and mAP vs model input resolution of YOLOv4 detection models.


Fig. 6: Faces (left) and license plates (right) at and below the *visible* thresholds.

need to count pixel by pixel to determine if an object is relevant. Instead, we define areas of the frame where objects are considered visible, and therefore identifiable. Fig. 9 shows 1st floor video frames with the visible areas indicated. We anonymize a diverse set of 8 test videos with both the $960\times960$ and $1440 \times 1440$ resolution versions of the anonymization pipeline[3], then manually inspect each output video for missed objects in the visible areas. Each test video is between 5,340 and 5,372 frames with an average of 6,513 visible faces and 11,571 visible license plates per video. The results of the manual accuracy evaluation largely support the results of the programmatic accuracy evaluation with the $1440 \times 1440$ resolution YOLOv4 model scoring an average recall of $98.6\%$. The recall by class as well as the $960 \times 960$ results are shown in Table I. These results show that both the $960 \times 960$ and $1440 \times 1440$ input resolution models detect visible faces and license plates with high recall, and that there is an average difference of only $0.2\%$ recall between the $1440 \times 1440$ and $960 \times 960$ model. This indicates that there may be a point at which increasing the model input resolution results in diminishing returns as far as recall is concerned, and also that there are some edge cases where our pipeline consistently fails. Specific cases including license plate border occlusion, superimposed faces, partial license plate occlusion from tree branches, detection of public bus license plates, and other rare

[3]The $608 \times 608$ resolution model was excluded from manual evaluations.

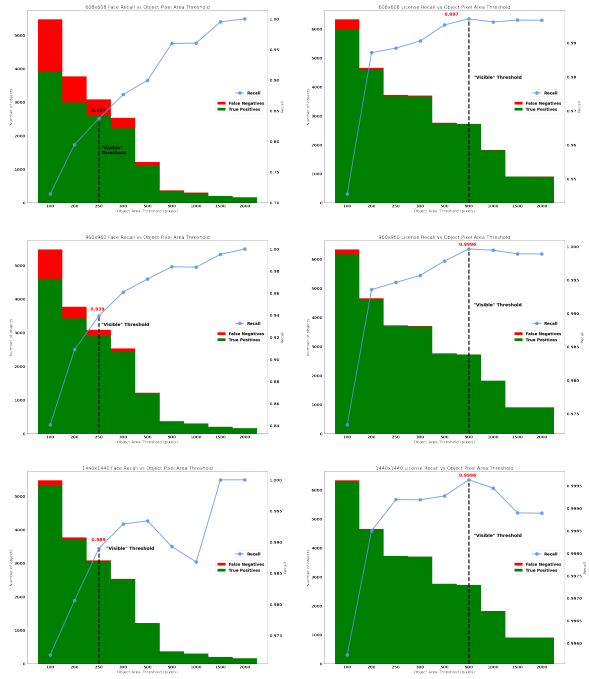

Fig. 7: Face (left) and license plate (right) detection recall as a function of object pixel threshold. True positives (green) and false negatives (red) comprise the evaluation set. Recall (blue) increases as smaller objects are excluded. Plots are shown for $608 \times 608$ (top), $960 \times 960$ (middle), and $1440 \times 1440$ (bottom) input resolutions.

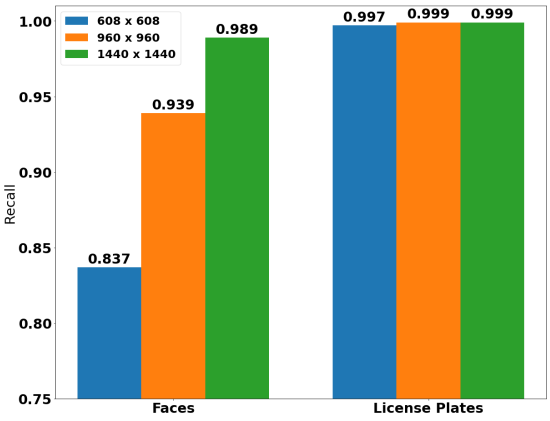


Fig. 8: Validation recall of face and license plates with cross section areas larger than the visible pixel area threshold.

scenarios are identified and illustrated in Fig. 10. Given that we use supervised face and license plate detection models, more training data encompassing these edge cases is required to further increase the recall of the anonymization pipeline.

### *B. Latency Evaluation*

To assess the real-time performance of our anonymization pipeline, we perform an exhaustive set of full pipeline timings with varying configurations of model input resolution, GPU hardware, floating point precision, batch size, and input video


Fig. 9: 1st floor video frames with visible area boundaries for faces (left) and license plates (right) drawn for manual evaluations.

TABLE I: Manual Accuracy Evaluation Results

| Model Resolution | Face Recall | License Plate Recall |
|---|---|---|
| 960x960 | 98.24% | 98.61% |
| 1440x1440 | 98.61% | 98.62% |

resolution. Timing is broken down by the pipeline subprocesses as described in Section III-C: video frame read, frame preprocessing, model inference, NMS, frame postprocessing and anonymization, and video frame write. The pipelines are implemented on Google Cloud Platform using a single NVIDIA TeslaT4 or A100-SXM4 GPU with 15GB and 40GB of memory, respectively. The TensorRT anonymization

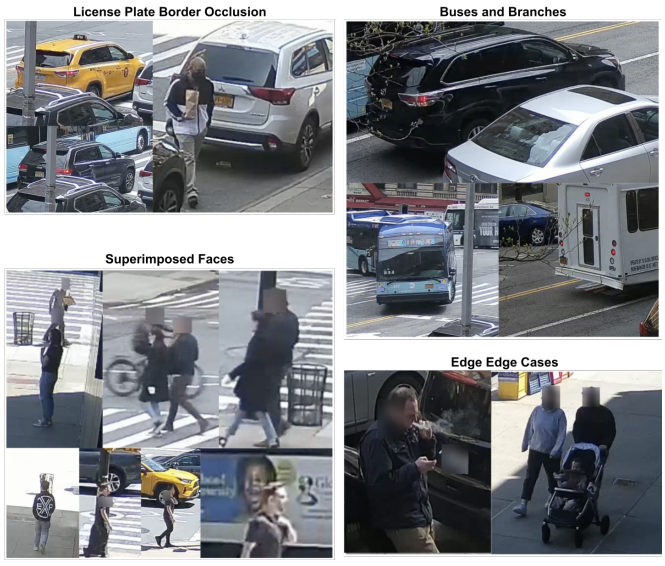


Fig. 10: Edge cases where the YOLOv4 face and license detection consistently fails.

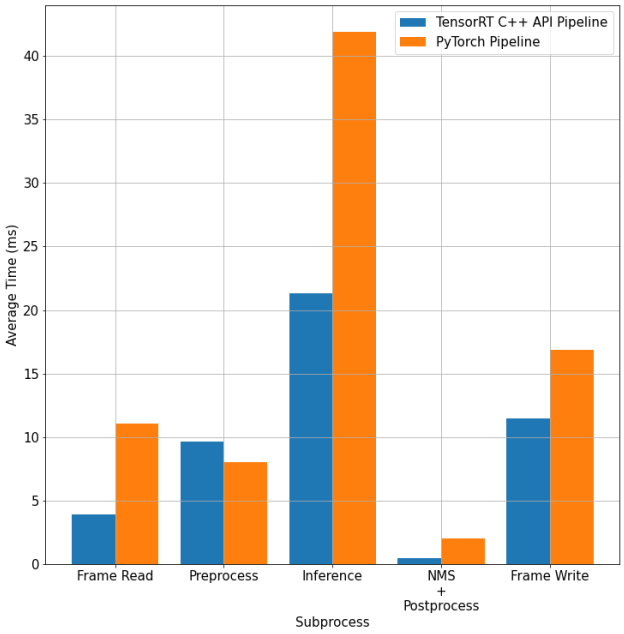


Fig. 11: Average time by subprocess of the PyTorch and TensorRT anonymization pipelines configured with the $960 \times 960$ input resolution model, batch size of 1, and FP32 precision.

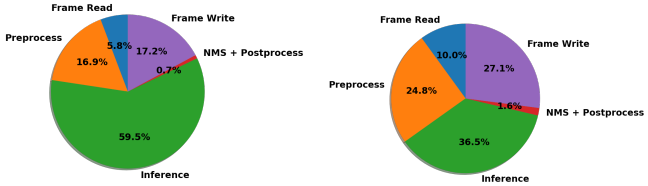


Fig. 12: Time profile breakdowns of the TensorRT anonymization pipeline with model input resolution of $960 \times 960$ on T4 GPU with batch size of 1 (left) and $608 \times 608$ on A100 GPU with batch size of 8 (right). Both profiles use FP16 precision.

pipeline was also implemented on an NVIDIA Jetson Nano with an integrated 4GB TegraX1 GPU, but the Jetson Nano is not capable of operating the anonymization pipeline in real-time. See Tables III and IV for the full inference speed and pipeline time profiling results.

In Fig. 11 we directly compare the time profiles of the PyTorch and the TensorRT anonymization pipelines configured with the $960 \times 960$ input resolution model, FP32 precision, $1 \times A100$, and a batch size of 1. Unsurprisingly, the C++ TensorRT pipeline has a higher throughput than the Python PyTorch pipeline in all subprocesses except frame preprocess. As a fraction of the total pipeline time, model inference accounts for $52.40\%$ of the PyTorch pipeline, whereas it is $45.54\%$ of the TensorRT pipeline. When using FP16 precision and batch size of 8 with the TensorRT pipeline, the fraction of pipeline time spent on inference drops to $32.51\%$ as a result of the TensorRT inference optimizations and the capacity of the A100 GPU to compute size 8 tensor batches.

Fig. 12 shows the time profile breakdowns by subprocess of two configurations of the TensorRT anonymization pipeline. In both profiles the majority of the pipeline time is spent on the model inference subprocess, but as model input resolution is reduced from $960 \times 960$ to $608 \times 608$ and batch size is increased from 1 to 8, we see that the combined frame read and write operations begin to occupy a larger portion of the time profile. As input resolution decreases and GPU compute power increases, the computational bottleneck of the anonymization pipeline shifts from inference computation to memory copy operations. This indicates that, given the sufficient GPU computing resources at the COSMOS edge cloud computing node, real-time implementation of the anonymization pipeline is achievable. Additionally, the anonymization module (blurring) does not add significant overhead to the end-to-end pipeline speed, as all anonymization operations are completed in under 1 ms per frame[4] using OpenCV. The optimal real-time configuration of the anonymization pipeline uses the $960 \times 960$ input resolution model, $1 \times A100$ GPU, a batch size of 1, and FP16 precision. With this configuration the pipeline anonymizes faces and license plates with recall over $98\%$ and achieves an inference speed of 61.22 FPS – well beyond the 30 FPS threshold of standard real-time operation.

## V. Conclusion

This paper presents a deep learning based anonymization pipeline for video in smart city intersections. The pipeline is implemented using YOLOv4 detection models, in both PyTorch and NVIDIA TensorRT, customized for the street-level intersection dataset collected at an intersection at the COSMOS testbed. Rigorous accuracy evaluations demonstrate that the pipeline anonymizes over $99\%$ of visible faces and license plates. Inference optimizations significantly improve the pipeline throughput and allow for operation at speeds significantly faster than the real-time threshold of 30 frames per second. Through manual accuracy evaluations it is shown that there are some edge cases where the pipeline consistently fails. Because of the supervised nature of the deep learning models, these edge cases can be rectified through collection, labeling, and training of an even larger intersection dataset with more examples of the problematic scenarios, or by augmentation techniques. The experiments demonstrate that GPUs used in edge-cloud servers, such as NVIDIA T4 and A100, can operate the pipeline in real time. Lower-end devices such as NVIDIA Jetson Nano fall short in terms of real time performance. Further work includes: (i) weight pruning to reduce the computational burden of the forward pass on edge devices; (ii) the development of unsupervised detection techniques to eliminate the dependency on large hand-labeled datasets; and (iii) exploration of augmentation techniques to combat the failure of anonymization in edge cases.

The presented deep learning-based video anonymization pipeline provides automatic, comprehensive, and fast privacy protection for smart city intersection video feeds.

[4]This excludes the Jetson Nano pipeline.

TABLE II: Anonymization Pipeline Timing with Various Configurations

| Model Input Resolution (pixels) | GPU | Precision | Batch Size | Full Pipeline | Frame Read | Preprocess | Inference | NMS + Postprocess | Frame Write |
|---|---|---|---|---|---|---|---|---|---|
| **TensorRT C++ Pipeline** | | | | | | | | | |
| 608x608 | TegraX1 | FP16 | 1 | 653.79 | 5.13 | 25.94 | 563.34 | 8.23 | 51.13 |
| 960x960 | TegraX1 | FP16 | 1 | 1491.49 | 10.79 | 64.74 | 1305.81 | 10.80 | 99.32 |
| 1440x1440 | TegraX1 | FP16 | 1 | 3285.98 | 23.74 | 144.34 | 2899.58 | 13.40 | 204.89 |
| 608x608 | TeslaT4 | FP16 | 1 | 29.06 | 1.74 | 4.40 | 17.94 | 0.27 | 4.70 |
| 960x960 | TeslaT4 | FP16 | 1 | 63.34 | 3.65 | 10.67 | 37.68 | 0.47 | 10.86 |
| 960x960 | TeslaT4 | FP16 | 4 | 63.71 | 3.91 | 10.73 | 38.32 | 0.45 | 10.28 |
| 960x960 | TeslaT4 | FP16 | 8 | 63.37 | 3.87 | 10.96 | 38.48 | 0.43 | 9.63 |
| 1440x1440 | TeslaT4 | FP16 | 1 | 139.35 | 7.64 | 23.43 | 84.97 | 0.76 | 22.55 |
| 1440x1440 | TeslaT4 | FP16 | 4 | 139.93 | 7.88 | 23.51 | 85.97 | 0.75 | 21.81 |
| 608x608 | TeslaT4 | FP32 | 1 | 44.75 | 1.59 | 4.34 | 33.99 | 0.24 | 4.58 |
| 960x960 | TeslaT4 | FP32 | 1 | 97.46 | 3.66 | 10.52 | 72.41 | 0.44 | 10.43 |
| 960x960 | TeslaT4 | FP32 | 4 | 99.34 | 3.89 | 11.05 | 73.51 | 0.45 | 10.43 |
| 1440x1440 | TeslaT4 | FP32 | 1 | 223.01 | 7.65 | 23.43 | 168.4 | 0.76 | 22.78 |
| 608x608 | A100 | FP16 | 1 | 21.82 | 1.90 | 4.41 | 9.83 | 0.33 | 5.34 |
| **960x960** | **A100** | **FP16** | **1** | **42.44** | **4.05** | **9.7** | **16.33** | **0.51** | **11.83** |
| 960x960 | A100 | FP16 | 4 | 38.82 | 4.00 | 9.75 | 13.16 | 0.51 | 11.39 |
| 960x960 | A100 | FP16 | 8 | 38.1 | 4.03 | 10.13 | 12.39 | 0.49 | 11.05 |
| 1440x1440 | A100 | FP16 | 1 | 83.19 | 8.2 | 21.22 | 28.26 | 0.83 | 24.67 |
| 1440x1440 | A100 | FP16 | 4 | 79.35 | 8.14 | 21.34 | 24.67 | 0.80 | 24.39 |
| 608x608 | A100 | FP32 | 1 | 23.64 | 1.74 | 4.28 | 12.17 | 0.28 | 5.15 |
| 960x960 | A100 | FP32 | 1 | 46.88 | 3.9 | 9.66 | 21.34 | 0.50 | 11.47 |
| 960x960 | A100 | FP32 | 4 | 42.47 | 3.88 | 9.92 | 17.09 | 0.49 | 11.08 |
| 1440x1440 | A100 | FP32 | 1 | 91.62 | 8.07 | 21.06 | 37.22 | 0.81 | 24.45 |
| **PyTorch Python Pipeline** | | | | | | | | | |
| 608x608 | TeslaT4 | FP32 | 1 | 78.43 | 3.63 | 4.91 | 61.01 | 0.01 | 7.65 |
| 960x960 | TeslaT4 | FP32 | 1 | 173.31 | 9.52 | 10.93 | 134.77 | 0.01 | 16.08 |
| 608x608 | A100 | FP32 | 1 | 63.05 | 3.02 | 3.89 | 46.86 | 0.01 | 8.01 |
| 960x960 | A100 | FP32 | 1 | 79.94 | 11.07 | 8.06 | 41.89 | 0.01 | 16.89 |
| 960x960 | A100 | FP32 | 2 | 63.73 | 7 | 8.1 | 30.22 | 0.02 | 16.62 |
| 1440x1440 | A100 | FP32 | 1 | 130.89 | 14.12 | 20.11 | 58.86 | 0.02 | 34.41 |

All values are average execution time per frame measured in milliseconds. Timing operations incur negligible overhead ($\approx 10\mu s$).

TABLE III: TensorRT Anonymization Model Inference Speeds

| Model Input Resolution (pixels) | Batch Size | T4 FP16 Inference Speed (FPS) | A100 FP16 Inference Speed (FPS) | T4 FP32 Inference Speed (FPS) | A100 FP32 Inference Speed (FPS) |
|---|---|---|---|---|---|
| 608x608 | 1 | 55.73 | 101.77 | 29.42 | 82.14 |
| 608x608 | 4 | 62.65 | 138.27 | 30.96 | 105.19 |
| 608x608 | 8 | 65.56 | **149.88** | 32.22 | 121.82 |
| 960x960 | 1 | 26.54 | 61.22 | 13.81 | 46.87 |
| 960x960 | 4 | 26.09 | 75.98 | 13.60 | 58.52 |
| 960x960 | 8 | 25.99 | 80.73 | 13.69 | 66.26 |
| 1440x1440 | 1 | 11.77 | 35.39 | 5.94 | 26.87 |
| 1440x1440 | 4 | 11.63 | 40.54 | 5.93 | 32.57 |
| 1440x1440 | 8 | 11.71 | 46.97 | 5.89 | 34.06 |

TABLE IV: Jetson Nano (TegraX1) TensorRT Anonymization Model Inference Analysis

| Model Input Resolution (pixels) | Batch Size | Inference Speed (FPS) | Inference Time (s) | Inference (% pipeline time) |
|---|---|---|---|---|
| 608x608 | 1 | 1.78 | 0.563 | 86.17 |
| 608x608 | 4 | 1.84 | 0.542 | 87.3 |
| 960x960 | 1 | 0.77 | 1.305 | 87.55 |
| 960x960 | 4 | 0.73 | 1.361 | 88.23 |
| 1440x1440 | 1 | 0.34 | 2.899 | 88.24 |

## ACKNOWLEDGMENT

This work was supported in part by NSF grants CNS-1827923, CNS-1910757, OAC-2029295, CNS-2038984, and CNS-2148128.

## REFERENCES

[1] D. Feldman, C. Xiang, R. Zhu, and D. Rus, "Coresets for differentially private k-means clustering and applications to privacy in mobile sensor networks," in *Proc. ACM/IEEE IPSN*, 2017.

[2] G. Mai, K. Cao, X. Lan, and P. C. Yuen, "Secureface: Face template protection," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 262–277, 2021.

[3] V. N. Boddeti, "Secure face matching using fully homomorphic encryption," in *Proc. IEEE BTAS*, 2018.

[4] M. K. Morampudi, M. V. N. K. Prasad, and U. S. N. Raju, "Privacy-preserving iris authentication using fully homomorphic encryption," *Multimedia Tools and Applications*, vol. 79, pp. 19215 – 19237, 2020.

[5] Q. Meng, F. Zhou, H. Ren, T. Feng, G. Liu, and Y. Lin, "Improving federated learning face recognition via privacy-agnostic clusters," *arXiv preprint:2201.12467*, 2022.

[6] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 53rd Annual Allerton Conference on Communication, Control, and Computing*, 2015.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas,

TABLE V: PyTorch Anonymization Model Inference Speeds

| Model Input Resolution (pixels) | Batch Size | T4 Inference Speed (FPS) | A100 Inference Speed (FPS) |
|---|---|---|---|
| 608x608 | 1 | 16.39 | 21.34 |
| 608x608 | 2 | 17.63 | 43.83 |
| 608x608 | 4 | - | 61.94 |
| 608x608 | 8 | - | 70.13 |
| 960x960 | 1 | 7.42 | 23.87 |
| 960x960 | 2 | - | 33.09 |
| 960x960 | 4 | - | 36.61 |
| 1440x1440 | 1 | - | 16.99 |
| 1440x1440 | 2 | - | 12.86 |

Configurations (batch sizes, model input sizes) that could not fit into T4 GPU memory are omitted from the table.


"Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.

[8] Z. Kostić, A. Angus, Z. Yang, Z. Duan, I. Seskar, G. Zussman, and D. Raychaudhuri, "Smart city intersections: Intelligence nodes for future metropolises," *arXiv preprint:2205.01686*, 2022.

[9] 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, "Challenge: COSMOS: A city-scale programmable testbed for experimentation with advanced wireless," in *Proc. ACM MobiCom*, 2020.

[10] M. Ghasemi, Z. Yang, M. Sun, H. Ye, Z. Xiong, Z. Kostic, and G. Zussman, "Demo: Video-based social distancing evaluation in the COSMOS testbed pilot site," in *Proc. ACM MobiCom'21*, 2021.

[11] S. Yang, E. Bailey, Z. Yang, J. Ostrometzky, G. Zussman, I. Seskar, and Z. Kostic, "COSMOS smart intersection: Edge compute and communications for bird's eye object tracking," in *Proc. SmartEdge*, 2020.

[12] M. Ghasemi, Z. Kostic, J. Ghaderi, and G. Zussman, "Auto-SDA: Automated video-based social distancing analyzer," in *Proc. ACM HotEdgeVideo*, 2021.

[13] Z. Yang, M. Sun, H. Ye, Z. Xiong, G. Zussman, and Z. Kostic, "Bird's-eye view social distancing analysis system," in *Proc. IEEE ICC Workshops*, 2022.

[14] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," *arXiv preprint:1607.07155*, 2016.

[15] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE CVPR*, 2017.

[16] H. Wu, B. Xiao, N. C. F. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang, "CvT: Introducing convolutions to vision transformers," in *Proc. IEEE/CVF ICCV*, 2021.

[17] C. Li, J. Yang, P. Zhang, M. Gao, B. Xiao, X. Dai, L. Yuan, and J. Gao, "Efficient self-supervised vision transformers for representation learning," *arXiv preprint:2106.09785*.

[18] M. Caron, H. Touvron, I. Misra, H. J'egou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proc. IEEE/CVF ICCV*, 2021.

[19] Brighter AI, "Brighter AI privacy protection." https://brighter.ai/product/. Accessed May 5, 2022.

[20] Sightengine, "Sightengine video redaction API." https://sightengine.com/docs/video-redaction-and-anonymization. Accessed May 5, 2022.

[21] Sightcorp, "Sightcorp by raydiant face blur." https://sightcorp.com/face-blur/. Accessed May 5, 2022.

[22] Gallio, "Gallio automated image and video anonymization." https://gallio.pro/. Accessed May 5, 2022.

[23] A. Frome, G. Cheung, A. Abdulkader, M. Zennaro, B. Wu, A. Bissacco, H. Adam, H. Neven, and L. Vincent, "Large-scale privacy protection in google street view," in *Proc. IEEE ICCV*, 2009.

[24] F. Mireshghallah, M. Taram, P. Vepakomma, A. Singh, R. Raskar, and H. Esmaeilzadeh, "Privacy in deep learning: A survey," *arXiv preprint:2004.12254*, 2020.

[25] C. A. Barbano, E. Tartaglione, and M. Grangetto, "Bridging the gap between debiasing and privacy for deep learning," in *Proc. IEEE/CVF ICCV Workshops*, 2021.

[26] Y. Zhou, M. Lv, Z. Ling, and D. Li, "Multiple license plate location algorithm in complex scene," in *Proc. IEEE International Conferences on Ubiquitous Computing Communications (IUCC) and Data Science and Computational Intelligence (DSCI) and Smart Computing, Networking and Services (SmartCNS)*, 2019.

[27] Y. Zou, Y. Zhang, J. Yan, X. Jiang, T. Huang, H. Fan, and Z. Cui, "A robust license plate recognition model based on Bi-LSTM," *IEEE Access*, vol. 8, pp. 211630–211641, 2020.

[28] S. Nacakli and A. M. Tekalp, "Controlling P2P-CDN live streaming services at SDN-enabled multi-access edge datacenters," *IEEE Transactions on Multimedia*, vol. 23, pp. 3805–3816, 2021.

[29] Fraunhofer HHI, "H.264 ultra low latency video codec." https://www.hhi.fraunhofer.de/en/departments/vca/technologies-and-solutions/h264-avc/h264-ultra-low-latency-video-codec.html. Accessed May 6, 2022.

[30] "Appear announces support for ultra-low latency hevc encoding on x platform." https://www.appear.net/ultra-low-latency-hevc-encoding-on-the-x-platform/, 2021. Accessed May 6, 2022.

[31] "Introducing the world's first 35ms latency mobile video transmission encoder/transmitter for teleoperation, 5G and other applications." https://solitonsys.com/products/advanced-teleoperation/ultra-low-latency-h-265-solution/, 2022. Accessed May 6, 2022.

[32] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv preprint:2004.10934*, 2020.

[33] F. Guo, Y. Qian, and Y. Shi, "Real-time railroad track components inspection based on the improved YOLOv4 framework," *Automation in Construction*, vol. 125, p. 103596, 2021.

[34] A. A. Shaghouri, R. Alkhatib, and S. Berjaoui, "Real-time pothole detection using deep learning," *arXiv preprint:2107.06356*, 2021.

[35] H. Liu, K. Fan, Q. Ouyang, and N. Li, "Real-time small drones detection based on pruned YOLOv4," *Sensors*, vol. 21, no. 10, 2021.

[36] E. Jeong, J. Kim, S. Tan, J. Lee, and S. Ha, "Deep learning inference parallelization on heterogeneous processors with tensorrt," *IEEE Embedded Systems Letters*, vol. 14, no. 1, pp. 15–18, 2022.

[37] X. Ma, K. Ji, B. Xiong, L. Zhang, S. Feng, and G. Kuang, "Light-YOLOv4: An edge-device oriented target detection method for remote sensing images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 10808–10820, 2021.

[38] K. Roszyk, M. R. Nowicki, and P. Skrzypczyński, "Adopting the YOLOv4 architecture for low-latency multispectral pedestrian detection in autonomous driving," *Sensors*, vol. 22, no. 3, 2022.

[39] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Scaled-YOLOv4: Scaling cross stage partial network," in *Proc. IEEE/CVF CVPR*, 2021.

[40] NVIDIA, "NVIDIA TensorRT, an SDK for high-performance deep learning inference." https://developer.nvidia.com/tensorrt. Accessed May 5, 2022.

[41] OpenVINO, "Computer vision annotation tool (CVAT)." https://github.com/openvinotoolkit/cvat. Accessed May 5, 2022.

[42] COCO, "COCO: Common objects in context - evaluate." https://cocodataset.org/#detection-eval. Accessed May 6, 2022.

[43] J. Redmon, "Darknet: Open source neural networks in C." http://pjreddie.com/darknet/, 2013–2016.

[44] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft COCO: Common objects in context," *arXiv preprint:1405.0312*, 2014.

[45] K. Yang, J. Yau, L. Fei-Fei, J. Deng, and O. Russakovsky, "A study of face obfuscation in imagenet," *arXiv preprint:2103.06191*, 2021.

[46] H. Hukkelås, R. Mester, and F. Lindseth, "Deepprivacy: A generative adversarial network for face anonymization," *arXiv preprint:1909.04538*, 2019.

[47] COSMOS, "COSMOS testbed cameras." https://wiki.cosmos-lab.org/wiki/Hardware/CamerasCameras. Accessed August 7, 2022.

[48] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, "SwinIR: Image restoration using swin transformer," in *Proc. IEEE/CVF ICCV*, 2021.