Why Is the Video Analytics Accuracy Fluctuating, and What Can We Do About It?

Sibendu Paul[1(✉)], Kunal Rao[2], Giuseppe Coviello[2], Murugan Sankaradas[2], Oliver Po[2], Y. Charlie Hu[1], and Srimat Chakradhar[2]

[1] Purdue University, West Lafayette, IN, USA
{paul90,ychu}@purdue.edu
[2] NEC Laboratories America, Inc., Princeton, NJ, USA
{kunal,giuseppe.coviello,murugs,oliver,chak}@nec-labs.com

Abstract. It is a common practice to think of a video as a sequence of images (frames), and re-use deep neural network models that are trained only on images for similar analytics tasks on videos. In this paper, we show that this "leap of faith" that deep learning models that work well on images will also work well on videos is actually flawed. We show that even when a video camera is viewing a scene that is not changing in any human-perceptible way, and we control for external factors like video compression and environment (lighting), the accuracy of video analytics application fluctuates noticeably. These fluctuations occur because successive frames produced by the video camera may look similar visually, but are perceived quite differently by the video analytics applications. We observed that the root cause for these fluctuations is the dynamic camera parameter changes that a video camera automatically makes in order to capture and produce a visually pleasing video. The camera inadvertently acts as an "unintentional adversary" because these slight changes in the image pixel values in consecutive frames, as we show, have a noticeably adverse impact on the accuracy of insights from video analytics tasks that re-use image-trained deep learning models. To address this inadvertent adversarial effect from the camera, we explore the use of transfer learning techniques to improve learning in video analytics tasks through the transfer of knowledge from learning on image analytics tasks. Our experiments with a number of different cameras, and a variety of different video analytics tasks, show that the inadvertent adversarial effect from the camera can be noticeably offset by quickly re-training the deep learning models using transfer learning. In particular, we show that our newly trained Yolov5 model reduces fluctuation in object detection across frames, which leads to better tracking of objects (∼40% fewer mistakes in tracking). Our paper also provides new directions and techniques to mitigate the camera's adversarial effect on deep learning models used for video analytics applications.

1 Introduction

Significant progress in machine learning and computer vision [9,24,41,42], along with the explosive growth in Internet of Things (IoT), edge computing, and

L. Karlinsky et al. (Eds.): ECCV 2022 Workshops, LNCS 13801, pp. 430–448, 2023.
https://doi.org/10.1007/978-3-031-25056-9_28

high-bandwidth access networks such as 5G [7,37], have led to the wide adoption of video analytics systems. These systems deploy cameras throughout the world to support diverse applications in entertainment, health-care, retail, automotive, transportation, home automation, safety, and security market segments. The global video analytics market is estimated to grow from $5 billion in 2020 to $21 billion by 2027, at a CAGR of 22.70% [14].

Video analytics systems rely on state of the art (SOTA) deep learning models [24] to make sense of the content in the video streams. It is a common practice to think of a video as a sequence of images (frames), and re-use deep learning models that are trained only on images for video analytics tasks. Large, image datasets like COCO [27] have made it possible to train highly-accurate SOTA deep learning models [2,6,21,30,39,40] that detect a variety of objects in images. In this paper, we take a closer look at the use of popular deep neural network models trained on large image datasets for predictions in critical video analytics tasks. We consider video segments from two popular benchmark video datasets [3,13]. These videos contain cars or persons, and we used several SOTA deep neural network (DNN) models for object detection and face detection tasks to make sense of the content in the video streams. Also, these videos exhibit minimal activity (*i.e.,* cars or persons are not moving appreciably and hence, largely static). Since the scenes are mostly static, the ground truth (total number of cars or persons) does not change appreciably from frame to frame within each video. Yet, we observe that the accuracy of tasks like object detection or face detection unexpectedly fluctuate noticeably for consecutive frames, rather than more or less stay the same. Such unexpected, noticeable fluctuations occur across different camera models and across different camera vendors.

Such detection fluctuations from frame to frame have an adverse impact on applications that use insights from object or face detection to perform higher-level tasks like tracking objects or recognizing people. Understanding the causes for these unexpected fluctuations in accuracy, and proposing methods to mitigate the impact of these fluctuations, are the main goals of this paper. We investigate the causes of the accuracy fluctuations of these SOTA deep neural network models on largely static scenes by carefully considering factors external and internal to a video camera. We examine the impact of external factors like the environmental conditions (lighting), video compression and motion in the scene, and internal factors like camera parameter settings in a video camera, on the fluctuations in performance of image-trained deep neural network models. Even after carefully controlling for these external and internal factors, the accuracy fluctuations persist, and our experiments show that another cause for these fluctuations is the dynamic camera parameter changes that a video camera automatically makes in order to capture and produce a visually pleasing video. The camera inadvertently acts as an "unintentional adversary" because these slight changes in image pixel values in consecutive frames, as we show, have a noticeably adverse impact on the accuracy of insights from video analytics tasks that re-use image-trained deep learning models. To address this inadvertent adversarial effect from the camera, we explore ways to mitigate this effect

and propose the transfer of knowledge from learning on image analytics tasks to video analytics tasks.

In this paper, we make the following key contributions:

- We take a closer look at the use of popular deep learning models that are trained on large image datasets for predictions in critical video analytics tasks, and show that the accuracy of tasks like object detection or face detection unexpectedly fluctuate noticeably for consecutive frames in a video; consecutive frames capture the same scene and have the same ground truth. We show that such unexpected, noticeable fluctuations occur across different camera models and across different camera vendors.
- We investigate the root causes of the accuracy fluctuations of these SOTA deep neural network models on largely static scenes by carefully considering factors external and internal to a video camera. We show that a video camera inadvertently acts as an "unintentional adversary" when it automatically makes camera parameter changes in order to capture and produce a visually pleasing video.
- We draw implications of the unintentional adversarial effect on the practical use of computer vision models and propose a simple yet effective technique to transfer knowledge from learning on image analytics tasks to video analytics. Our newly trained Yolov5 model reduces fluctuation in object detection across frames, which leads to better performance on object tracking task (∼40% fewer mistakes in tracking).

2 Motivation

In this section, we consider video segments from two popular benchmark datasets. These videos contain cars or persons, and the videos exhibit minimal activity (*i.e.,* cars or persons are not moving appreciably and hence, largely static). Since the scenes are mostly static, the ground truth (total number of cars or persons) from frame to frame is also not changing much. Yet, we observe that the accuracy of tasks like object detection or face detection unexpectedly fluctuate noticeably for consecutive frames. Such accuracy fluctuations from frame to frame have an adverse impact on applications that use insights from object or face detection to perform higher-level tasks like tracking objects or recognizing people.

2.1 Object Detection in Videos

(a) Roadway Dataset (b) LSTN Dataset

Fig. 1. Sample frames from video datasets.

One of the most common task in video analytics pipelines is object detection. Detecting cars or people is critical for many real-world applications like video surveillance, retail, health care monitoring and intelligent transportation systems (Fig. 1).

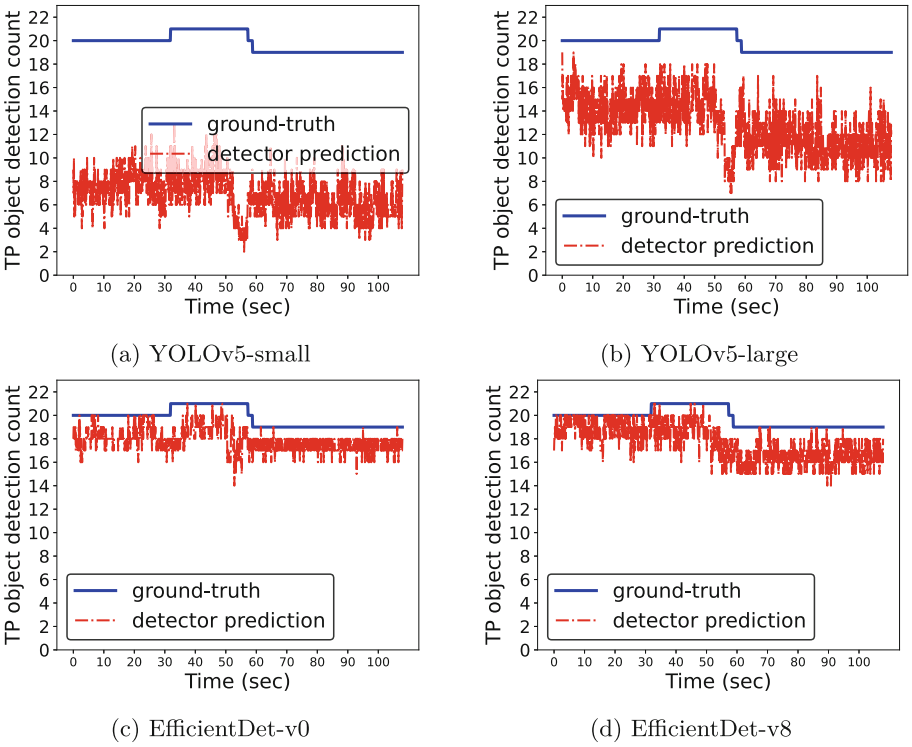
Figure 2 shows the performance of different state of the art and widely-used object detectors like YOLOv5-small and large variant [21], EfficientDet-v0 and EfficientDet-v8 [40] on video segments from the Roadway dataset [3]. These videos have cars and people, but the activity is minimal, and scenes are largely static. The "ground truth" in the figures is shown in blue color, and it shows the total number of cars and people at different times (i.e. frames) in the video. The "detector prediction" waveform (shown in red color) shows the number of cars and people actually detected by the deep learning model.

Our experiments show that (a) for all the detectors we considered, the number of detected objects is lower than the ground truth[1], and (b) more importantly, even though the ground truth is not changing appreciably in consecutive frames, the detections reported by the detectors vary noticeably, and (c) light-weight models like Yolov5-small or Yolov5-large exhibit a much higher range of detection fluctuations than the more heavier models like *efficientDet*. However, the heavier deep learning models make inferences by consuming significantly more computing resources than the light-weight models.

2.2 Face Detection in Videos

Next, we investigate if accuracy fluctuation observed in object detection models also occur in other image-trained AI models that are used in video analytics tasks. We chose AI models for face detection task, which is critical to many real-world applications *e.g.,* identifying a person of interest in airports, hospitals or arenas, and authenticating individuals based on face-recognition for face-based payments. Figure 3 shows the performance of three well-known face detection AI

[1] We have 1–2 false positive detections for Yolov5 and efficientDet.



(a) YOLOv5-small (b) YOLOv5-large

(c) EfficientDet-v0 (d) EfficientDet-v8

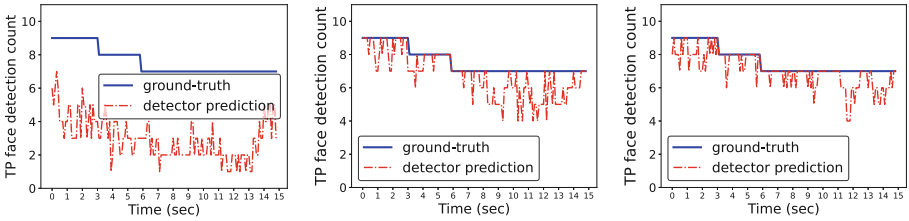
Fig. 2. Performance of various object detection models on a segment of pre-recorded video from the Roadway dataset [3]. (Color figure online)

models on videos from the *LSTN* video dataset [13]. Like the object detection case, we observe that (a) the number of faces detected by these models is typically lower than the ground truth, (b) more importantly, even though the ground truth barely changes, there is noticeable fluctuation in the number of detections in consecutive frames, and (c) the light-weight models like MTCNN [38] exhibit a much higher range of detection fluctuations than the more heavier models like RetinaNet with resnet-50 and mobilenet backbone [10].

3 Analysis and Control of External Factors

The behavior of a DNN model is deterministic in the sense that if a frame is processed multiple times by the DNN model, then the DNN inferences are identical. In this section, we analyze three external factors that may be causing the unexpected accuracy fluctuations described in Sect. 2:

– Motion in the field of view of the camera affects the quality of the captured video (blurring of moving objects is likely).



(a) MTCNN (b) RetinaNet (Resnet-50) (c) RetinaNet (Mobilenet)

Fig. 3. Performance of face detection models on videos from *LSTN* video dataset.

- Lossy video compression methods like H.264 can also result in decoded frames whose quality can differ from the pre-compression frames.
- Environmental conditions like lighting can also affect the quality of the frames processed by the DNNs. For example, flicker in fluorescent lighting can affect the quality of frames captured by the camera (most people cannot notice the flicker in fluorescent lights, which flicker at a rate of 120 cycles per second 120 Hz; as we show later, flicker also contributes to fluctuations in the analytics accuracy of video analytics tasks).

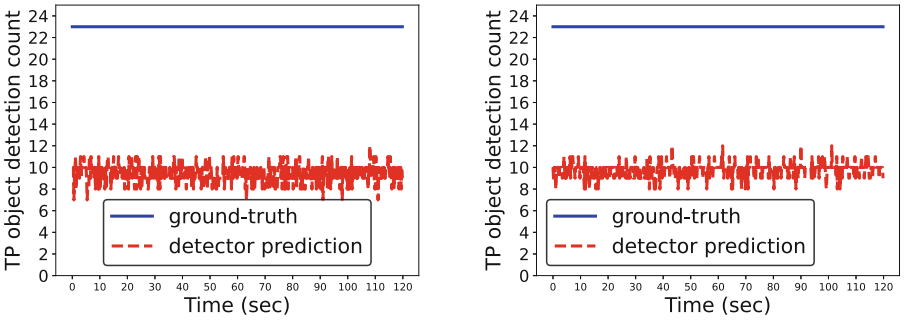
3.1 Control for Motion

It is difficult to systematically study the impact of motion on accuracy fluctuations by using videos from the datasets. Instead, as shown in Fig. 5a, we set up a scene with 3D models of objects (*i.e.,* persons and cars), and continuously observed the scene by using different IP cameras like *AXIS Q1615*. A fluorescent light provides illumination for the scene. Figure 5a shows a frame in the video stream captured by the IP camera under default camera parameter settings. This setup easily eliminates the effect of motion on any observed accuracy fluctuations. Also, this set up makes it easy to study whether accuracy fluctuations are caused by only certain camera models or fluctuations happen across different camera models from different vendors.

3.2 Analysis and Control for Video Compression

By using static 3D models, we eliminated the effect of *motion*. To understand the effect of video compression, we fetch frames directly from the camera instead of fetching a compressed video stream and decoding the stream to obtain frames that can be processed by a DNN model.

Figure 4a and Fig. 4b show the object detection counts with and without compression for the YOLOv5 model. We observe that eliminating compression reduces detection fluctuation. We also analyzed the detection counts with and without compression by using the *t-test* for repeated measures [44]. Let A be the sequence of true-positive object detection counts (per frame) for the experiment



(a) With Compression (b) Without Compression

Fig. 4. Effect of video compression on fluctuations in Yolov5 object detection counts (scene with 3D models)

where video compression is used. Let B be the sequence of true-positive object detection counts for the case when no compression is used. We compute a third sequence D that is a sequence of pair-wise differences between the true-positive object count without compression and with compression (*i.e.*, $B - A$).

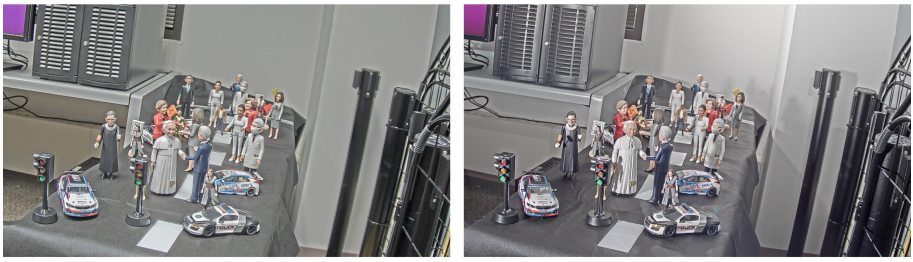
Essentially, the use of difference scores converts a two-sample problem with A and B into a one-sample problem with D. Our null hypothesis states that compression has no effect on object detection counts (and we hypothesize a population mean of 0 for the difference scores). Our experiment with a sample size of 200 frames showed that we can reject the null hypothesis at the 0.01 level of significance (99% confidence), suggesting there is evidence that elimination of compression does reduce the accuracy fluctuations. Similar results were observed for sample sizes of 100 and 1000 frames.

While *t-test* measures the statistical difference between two distributions, it doesn't reflect on the fluctuations observed in repeated measures. We propose two metrics to quantify the observed fluctuations across a group of frames. (1) *F2* which is defined as $\frac{\|tp(i)-tp(i+1)\|}{mean(gt(i),gt(i+1))}$ for frame i, where $tp(i)$, $gt(i)$ are true-positive object detection count and ground-truth object count respectively on frame i (on a moving window of 2 frames) and (2) *F10* which is defined as $\frac{\|max(tp(i),\ldots,tp(i+9))-min(tp(i),\ldots,tp(i+9))\|}{mean(gt(i),..,gt(i+9))}$ (on a moving window of 10 frames).

By eliminating video compression, the maximum variation in object count on static scene can be reduced from 17.4% to 13.0% (*F2*) and from 19.0% to 17.4% (*F10*). Clearly, video compression is highly likely to have an adverse effect on accuracy fluctuations, and eliminating compression can improve results of deep learning models.

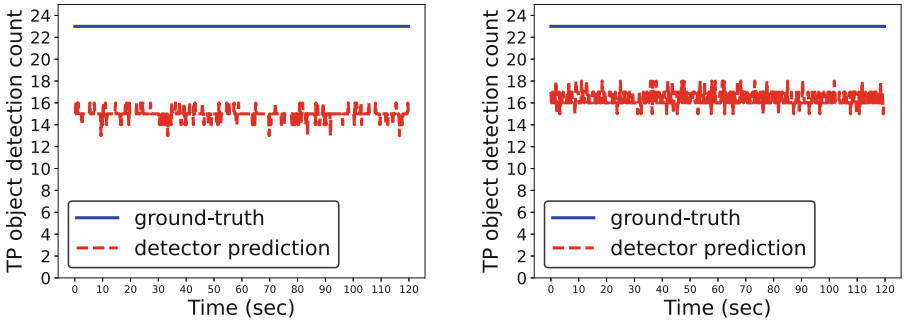
3.3 Analysis and Control for Flicker

By using static 3D models, we eliminated the effect of *motion*. We are also able to eliminate the adverse effect of video compression by fetching frames directly



(a) With flicker in lighting (b) Without flicker in lighting

Fig. 5. Scene with 3D models, with and without flickering light.

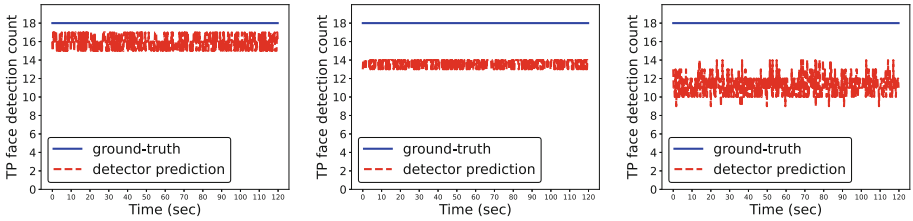


(a) YOLOv5 detections (b) EfficientDet-v0 detections

Fig. 6. Object detection counts when there is no motion, video compression or flickering light.

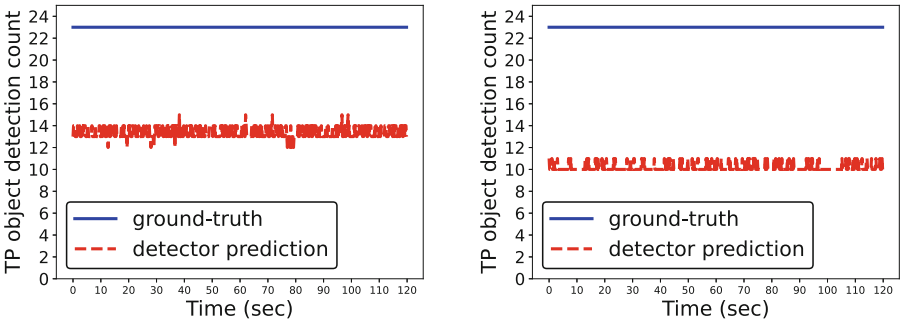
from the camera. We now analyze the effect of lighting. We set up an additional, flicker-free light source to illuminate the scene with static 3D models. Figure 5 shows the 3D models scene with and without flickering light. Figure 6a shows the fluctuation in detection counts when there is no motion, no video compression, and no flicker due to fluorescent light.

Compared to Fig. 4 results with no compression (but with fluorescent lighting), the results in Fig. 6a are highly likely to be an improvement. We compared the sequence of object detection counts with and without fluorescent light (no video compression in both cases) using the t-test for repeated measures, and easily rejected the null hypothesis that lighting makes no difference at a 0.01 level of significance (99% confidence). Also, eliminating light flickering on top of motion and compression can reduce the maximum (*F2*) and (*F10*) variations from 13.0% to 8.7% and 17.4% to 13.0% respectively. Therefore, after eliminating motion and video compression, fluorescent light with flicker is highly likely to have an adverse effect on accuracy fluctuations, and eliminating flicker is highly likely to improve the results from the DNN model.



(a) RetinaNet (Resnet-50) (b) RetinaNet (Mobilenet) (c) MTCNN

Fig. 7. Face detection counts for three different DNN models when there is no motion, video compression or flickering light.



(a) AXIS Q3515 (b) PELCO SARIX IME322

Fig. 8. Performance of YOLOv5 on different IP camera models in absence of motion, compression and flicker in lighting.

Figure 6b shows the object detection counts for EfficientDet-v0 when there is no motion, video compression or flickering light. We observe fluctuations in object detection count up to 13.0% (*F2*) and 14.0% (*F10*). Due to space reasons, we have not included the graphs for with and without compression for EfficientDet-v0. However, like the YOLOv5 case, eliminating motion, video compression and flickering light improves the detection results.

Our detailed analysis in this section shows that eliminating motion, video compression and flicker does improve the object detection results. However, even after controlling for motion, video compression and flickering light, noticeable fluctuations in object detection counts still remain. We repeated the above experiments for three SOTA open-source *face detection models*. Fig. 7 shows fluctuation in face detection counts when there is no motion, video compression or flickering light. *F2* metric reports true-positive face detection fluctuations up to 8.7%, 4.3%, 21.7% for two retinaNet models and MTCNN respectively.

3.4 Impact of Different Camera Models

We also investigated whether the fluctuation in video analytics accuracy is observed only on specific camera model or is it present across different camera models across different vendors. Figure 8 shows the performance of YOLOv5 object detection model on AXIS Q3515 and PELCO SARIX IME322 IP cameras, both of them observing the same scene and in absence of motion, compression and flicker in lighting. We note that both of them show fluctuation in the count of detected objects and *F2* metric reports up to 13.1% and 4.4% fluctuations for the two camera models. This shows that the fluctuation in video analytics accuracy is observed across different camera models from different vendors.

4 Camera as an Unintentional Adversary

In Sect. 3, we investigated various factors external to the camera that could lead to fluctuations in video analytics accuracy. Specifically, we looked at motion, compression, flicker in lighting, and camera models from different vendors and different deep learning models, and found out that fluctuation is observed across different deep learning models, on different cameras, even when motion, compression, flicker in lighting are eliminated. This leads us to hypothesize that the remaining factors causing accuracy fluctuation may not be external. Rather, it could be *internal* to the camera.

4.1 Hypothesis

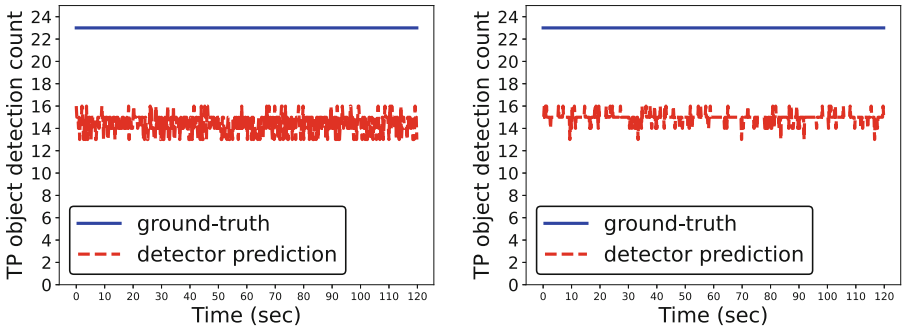
Auto-parameter Setting in Modern Cameras. Along with exposing endpoints to retrieve streaming videos (*e.g.,*RTSP stream URL), IP cameras also expose APIs to set various camera parameters (*e.g.,* VAPIX [8] API for Axis camera models). These camera settings aid in changing the quality of image produced by the camera. Camera vendors expose these APIs because they do not know ahead of time in what environment their camera would be deployed and what settings would be ideal for that environment. Therefore, they set the camera settings to some default values and let end users decide what settings would work best for their environment. There are two types of camera settings that are exposed by camera vendors: (1) Type 1 parameters include those that affect the way raw images are captured, *e.g.,* exposure, gain, and shutter speed. These parameters generally are adjusted *automatically* by the camera with little control by end users. They only allow end users to set maximum value, but within this value, the camera internally changes the settings dynamically in order to produce a visually pleasing video output. We refer to such parameters as automated parameters (AUTO). (2) Type 2 parameters include those that affect processing of raw data in order to produce the final frame, *e.g.,* image specific parameters such as brightness, contrast, sharpness, color saturation, and video specific parameters such as compression, GOP length, target bitrate, FPS. For these parameters, camera vendors often expose fine control to end users, who

can set the specific value. We refer to such parameters as non-automated parameters (NAUTO). The distinction between AUTO and NAUTO parameters help us refine our hypothesis where we can fix values of NAUTO parameters, vary the maximum value for AUTO parameters and observe how camera internally changes these parameters to produce different consecutive frames, which might be causing the fluctuations.

The Hypothesis. The purpose of a video camera is to capture videos, rather than still images, for viewing by human eyes. Hence, irrespective of how the scene in front of the camera looks like, *i.e.,* whether the scene is static or dynamic, video camera always tries to capture a video, which assumes changes in successive frames. To capture a visually pleasing and smooth (to human eyes) video, the camera tries to find the optimal exposure time or shutter speed. On one hand, high shutter speed, *i.e.,* low exposure time, freezes motion in each frame, which results in very crisp individual images. However, when such frames are played back at usual video frame rates, it can appear as hyper-realistic and provide a very jittery, unsettled feeling to the viewer [28]. Low shutter speed, on the other hand, can cause moving objects to appear blurred and also builds up noise in the capture. To maintain appropriate amount of motion blur and noise in the capture, video cameras have another setting called *gain*. Gain indicates the amount of amplification applied to the capture. A high gain can provide better images in low-light scenario but can also increase the noise present in the capture. For these reasons, the optimal values of AUTO parameters like exposure and gain are internally adjusted by the camera to output a visually pleasing smooth video. Thus, video capture is fundamentally different from still image capture and the exact values of exposure and gain used by the camera for each frame are not known to end users or analytics applications running on the video output from the camera. This loose control over maximum shutter time and maximum gain parameters is likely the explanation for fluctuations in video analytics accuracy, *i.e.,* the camera's unintentional adversarial effect.

4.2 Hypothesis Validation

The above explanation of our hypothesis that internal dynamic change of AUTO parameters applied by a camera causes successive frames to differ and hence fluctuations in video analytics accuracy, also points us a way to partially validate the hypothesis. Since the camera still exposes control of maximum values of AUTO parameters, we can adjust these maximum parameter value and observe the impact on the resulting fluctuation of analytics accuracy. Figure 9 shows the fluctuation in accuracy of YOLOv5 object detection model for two different settings of maximum exposure time. We observe that when the maximum exposure time is 1/120 s, the fluctuation in object count is significantly lower than when it is 1/4 s. Here, reducing the max exposure time decreases the maximum *F2* fluctuations from 13.0% to 8.7%. This corroborates our hypothesis – with a higher value of maximum exposure time, the camera can possibly choose from a larger number of different exposure times than when the value of maximum exposure



(a) Max exposure time = 1/4 second (b) Max exposure time = 1/120 second

Fig. 9. Performance of YOLOv5 Object detectors for two different settings of an AUTO parameter, in absence of motion, compression and flicker in lighting.

time is low, which in turn causes the consecutive frame captures to differ more, resulting in more accuracy fluctuation.

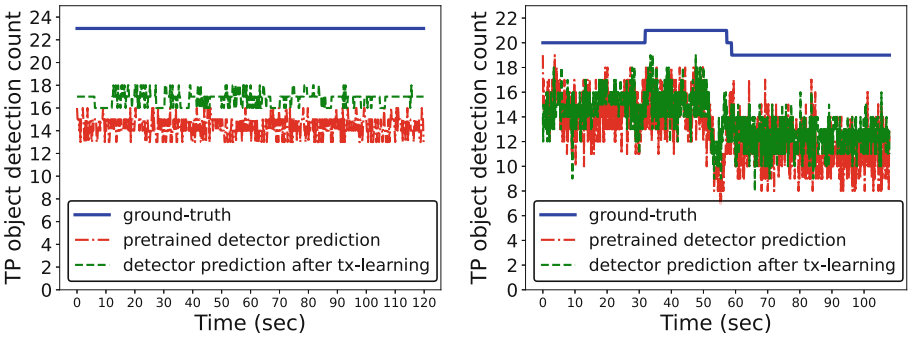
We compared the sequences of object detection counts at a maximum exposure of 1/120 s and 1/4 s using the t-test for repeated measures, and easily rejected the null hypothesis that lowering the maximum exposure time (*i.e.,* changing exposure time from 1/4 s to 1/120 s) makes no difference in detection counts, at a 0.01 level of significance (99% confidence). Therefore, the choice of maximum exposure time has a direct impact on the accuracy of the deep learning models, and the precise exposure time is automatically determined by the video camera. We quantify this using object tracking task (discussed in Sect. 5) and observe 65.7% fewer mistakes in tracking when exposure changes.

5 Implications

SOTA object detectors (Yolov5 [21] or EfficientDet [40]) are trained on still image datasets *e.g.,* COCO [27] and VOC [12] datasets. We observed in prior sections that the accuracy of insights from deep learning models fluctuate when used for video analytics tasks. An immediate implication of our finding is that deep learning models trained on still image datasets should not be directly used for videos. We discuss several avenues of research that can mitigate the accuracy fluctuations in video analytics tasks due to the use of image-trained DNN models.

5.1 Retraining Image-Based Models with Transfer Learning

One relatively straight-forward approach is to train models for extracting insights from videos using video frames that are captured under different scenarios. As a case study, we used transfer learning to train Yolov5 using the proprietary videos captured under different scenarios. These proprietary videos contain objects from person and vehicle super-category (that have car, truck, bus, train categories),



(a) Static Scene of 3D models (b) Video snippet from Roadway dataset

Fig. 10. Detection counts from YOLOv5-large after transfer-learning (Color figure online)

captured by the cameras at different deployment sites (*e.g.,* traffic intersection, airport, mall, etc.) during different times-of-the-day (*i.e.,* day, afternoon, night) and also under different weather conditions (*i.e.,* rainy, foggy, sunny). We extract total *34K* consecutive frames from these proprietary video snippets, and these frames form our training dataset.

Training Details. The first 23 modules (corresponding to 23 layers) of our new deep learning model are initialized using weights from COCO-trained Yolov5 model, and these weights are frozen. During training, only the weights in the last *detect* module are updated. For transfer learning, we used a learning rate of 0.01 with a weight decay value of 0.0005. We trained Yolov5 model on NVIDIA GeForce RTX 2070 GPU server for 50 epochs with a batch size of 32. This lightweight training required only 1.6 GB GPU memory and took less than 1.5 h to finish 50 epochs. We used the newly trained Yolov5 model to detect objects (*i.e.,* cars and persons) in (a) our static scene of 3D models, and (b) a video from the Roadway dataset (same video that was used in Sect. 2).

Figure 10a shows the improvement in detection counts due to the transfer-learning trained Yolo5 model (green waveform). The improvement over the original Yolo5 model (shown as red waveform) is noticeable visually. We also compared the sequence of object detection counts for the original Yolov5 model (red waveform) and the transfer-learning trained Yolov5 model (green waveform) by using a t-test for repeated measures. We easily rejected the null hypothesis that transfer-learning makes no difference, at a 0.01 level of significance (99% confidence). Then, we estimated the size of the effect due to transfer-learning, and we observed that at a 0.01 level of significance, the improvement is 2.32 additional object detections (14.3% improvement over the mean detections due to the original Yolov5 model). For this experiment, the camera was automatically setting AUTO camera parameters to produce a visually pleasing video, and the

transfer-learning trained Yolov5 detector was able to detect more objects despite the unintentional adversary (camera).

In practical deployments of video analytics systems that operate 24×7, it is difficult to control motion or environmental conditions, and the default video compression settings also vary from camera to camera. To understand the impact of transfer-learning trained Yolov5 model, we did experiments on videos in the Roadway dataset. These videos have motion, and the environmental conditions or video compression settings are unknown (such information is not part of the Roadway dataset). Figure 10b shows the results for a video in the Roadway dataset. The true-positive object detections by our *transfer-learning trained Yolov5 model* (green waveform) show noticeably less range of fluctuations than the original Yolov5 model (red waveform). We also compared the sequence of object detection counts for the original Yolov5 model (red waveform) and the transfer-learning trained Yolov5 model (green waveform) by using a t-test for repeated measures. We easily rejected the null hypothesis that transfer-learning makes no difference, at a 0.01 level of significance (99% confidence). Then, we estimated the size of the effect due to transfer-learning, and we observed that at a 0.01 level of significance, the improvement is 1 additional object detection (9.6% improvement over the mean detections due to the original Yolov5 model). Our *newly trained* Yolov5 model reduces the maximum variation of correctly detected object counts from 47.4% to 33.2% *(F10)*, and 42.1% to 32.5% *(F2)*.

Impact on Object Tracking. We evaluated the impact of the fluctuations in detection counts on object tracking task where we track the objects across different frames using *MOT SORT* [1] tracker. Object trackers assign the same track-id to an object appearing in contiguous frames. If an object is not detected in a frame, then the object's track is terminated. If the object is detected again in subsequent frames, a new track-id is assigned to the object. We use the number of track-ids assigned by a tracker as an indicator of the quality of object detections. Our tracker reported *157* track-ids when the original Yolov5 model was used for detecting objects in the video from the Roadway dataset. In contrast, the same tracker reported *94* track-ids when the *transfer-learning trained* Yolov5 model was used (*i.e.,* 40.1% fewer mistakes in tracking). We manually annotated the video and determined the ground-truth to be 29 tracks. We also manually inspected the tracks proposed by the tracker for the two models (with and without transfer-learning) to ensure that the tracks were true-positives. These experiments suggest that the transfer-learning based Yolov5 model leads to better performance on object tracking task.

5.2 Calibrating Softmax Confidence Scores

In general, we use softmax confidence output as the correctness probability estimate of the prediction. However, many of these neural networks are poorly calibrated [16,32]. The uncertainty in softmax confidence scores from poorly calibrated NN can potentially worsen the robustness of video analytics performance. To mitigate this, we can employ several post-hoc methods on SOTA models to

improve the softmax estimates, *e.g.,* via averaging predictions obtained from bag of models (*e.g.,* detectors, classifiers) [25], platt scaling [16], isotonic regression [34], etc. We can also adapt the confidence threshold to filter out the low-confidence mispredictions. This confidence threshold value can be adapted based on the difficulty level to detect in a certain frame. We leave the investigation of neural network calibration and confidence threshold adaptation as future work.

6 Related Work

Several efforts [5,20,31,33,49] have been made to improve the robustness of deep learning models against adversarial attacks. Recent works [18,45] propose several adversarial examples that can be used to train a robust model and also serve as performance measures under several distribution shifts. Robust model training based on shape representation rather than texture-based representation is proposed in [15]. Xie et al. [46] use unlabeled data to train SOTA model through *noisy student self distillation* which improves the robustness of existing models. However, the creation of these "robust" models does not take into account the kind of adversaries introduced by dynamic tuning of AUTO parameters by video cameras. Also, the perturbations introduced in variants of ImageNet dataset (*i.e.,* -C, -3C, -P *etc.*) [17] are not the same as observed when AUTO parameters are tuned, which makes such datasets unsuitable for our study.

While there have been many efforts [4,11,19,22,26,29,47,48] on saving compute and network resource usage without impacting the accuracy of video analytics pipelines (VSPs) by adapting different video-specific parameters like frame rate, resolution, compression, *etc.*, there has been little focus on improving the *accuracy* of VAPs. Techniques to improve the accuracy of VSPs by dynamically tuning camera parameters is proposed by Paul et al. [36], but they focus on image-specific NAUTO parameters rather than AUTO parameters, which we show is the cause for "unintentional" adversarial effect introduced by the camera. Otani et al. [35] show the impact of low video quality on analytics, but they study network variability rather than the camera being the reason for low video quality.

Koh et al. [23] identify the need for training models with the distribution shift that will be observed in practice in real-world deployment. Inspired from this, rather than using independent images or synthetically transformed images for training, we use real video frames for training, which takes into account the distribution shift observed in practice for video analytics.

Wenkel et al. [43] tackle the problem of finding optimal confidence threshold for different models in model training and mention the challenge that there is a possibility of fluctuation in accuracy based on small changes between consecutive frames. However, they do not go in depth to analyze it further as we do. To our best knowledge, we are the first to expose the camera as an "unintentional adversary" for video analytics task and propose mitigation techniques.

7 Conclusion

In this paper, we show that blindly applying image-trained deep learning models for video analytics tasks leads to fluctuation in accuracy. We systematically eliminate external factors including motion, compression and environmental conditions (e.g., lighting) as possible reasons for fluctuation and show that the fluctuation is due to internal parameter changes applied by the camera, which acts as an "unintentional adversary" for video analytics applications. To mitigate this adversarial effect, we propose a transfer learning based approach and train a new Yolov5 model for object detection. We show that by reducing fluctuation across frames, our model is able better track objects (∼40% fewer mistakes in tracking). Our paper exposes a fundamental fallacy in applying deep learning models for video analytics and opens up new avenues for research in this direction.

Acknowledgment. This project is supported in part by NEC Labs America and by NSF grant 2211459.

References

1. Bewley, A., Ge, Z., Ott, L., Ramos, F., Upcroft, B.: Simple online and realtime tracking. In: 2016 IEEE International Conference on Image Processing (ICIP), pp. 3464–3468 (2016). https://doi.org/10.1109/ICIP.2016.7533003
2. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: YOLOv4: optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934 (2020)
3. Canel, C., et al.: Scaling video analytics on constrained edge nodes. In: Proceedings of Machine Learning and Systems, vol. 1, pp. 406–417 (2019)
4. Chen, T.Y.H., Ravindranath, L., Deng, S., Bahl, P., Balakrishnan, H.: Glimpse: continuous, real-time object recognition on mobile devices. In: Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, pp. 155–168 (2015)
5. Cheng, M., Lei, Q., Chen, P.Y., Dhillon, I., Hsieh, C.J.: CAT: customized adversarial training for improved robustness. arXiv preprint arXiv:2002.06789 (2020)
6. Chiu, Y.C., Tsai, C.Y., Ruan, M.D., Shen, G.Y., Lee, T.T.: Mobilenet-SSDv2: an improved object detection model for embedded systems. In: 2020 International Conference on System Science and Engineering (ICSSE), pp. 1–5. IEEE (2020)
7. CNET: How 5G aims to end network latency (2019). CNET_5G_network_latency_time
8. AXIS Communications: Vapix library. https://www.axis.com/vapix-library/
9. Connell, J., Fan, Q., Gabbur, P., Haas, N., Pankanti, S., Trinh, H.: Retail video analytics: an overview and survey. In: Video Surveillance and Transportation Imaging Applications, vol. 8663, pp. 260–265 (2013)
10. Deng, J., Guo, J., Yuxiang, Z., Yu, J., Kotsia, I., Zafeiriou, S.: RetinaFace: single-stage dense face localisation in the wild. Arxiv (2019)
11. Du, K., et al.: Server-driven video streaming for deep learning inference. In: Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication, pp. 557–570 (2020)

12. Everingham, M., Gool, L.V., Williams, C.K.I., Winn, J.M., Zisserman, A.: The Pascal visual object classes (VOC) challenge. Int. J. Comput. Vis. **88**(2), 303–338 (2010). https://doi.org/10.1007/s11263-009-0275-4. http://dblp.uni-trier.de/db/journals/ijcv/ijcv88.html#EveringhamGWWZ10
13. Fang, Y., Zhan, B., Cai, W., Gao, S., Hu, B.: Locality-constrained spatial transformer network for video crowd counting. arXiv preprint arXiv:1907.07911 (2019)
14. Gaikwad, V., Rake, R.: Video Analytics Market Statistics: 2027 (2021). https://www.alliedmarketresearch.com/video-analytics-market
15. Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F.A., Brendel, W.: ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. arXiv preprint arXiv:1811.12231 (2018)
16. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: International Conference on Machine Learning, pp. 1321–1330. PMLR (2017)
17. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. In: Proceedings of the International Conference on Learning Representations (2019)
18. Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., Song, D.: Natural adversarial examples. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 15262–15271, June 2021
19. Jiang, J., Ananthanarayanan, G., Bodik, P., Sen, S., Stoica, I.: Chameleon: scalable adaptation of video analytics. In: Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, pp. 253–266 (2018)
20. Jin, C., Rinard, M.: Manifold regularization for locally stable deep neural networks. arXiv preprint arXiv:2003.04286 (2020)
21. Jocher, G., et al.: ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO export and inference (2022). https://doi.org/10.5281/zenodo.6222936
22. Kang, D., Emmons, J., Abuzaid, F., Bailis, P., Zaharia, M.: NoScope: optimizing neural network queries over video at scale. arXiv preprint arXiv:1703.02529 (2017)
23. Koh, P.W., et al.: WILDS: a benchmark of in-the-wild distribution shifts. In: Meila, M., Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 139, pp. 5637–5664. PMLR, 18–24 July 2021. https://proceedings.mlr.press/v139/koh21a.html
24. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
25. Lakshminarayanan, B., Pritzel, A., Blundell, C.: Simple and scalable predictive uncertainty estimation using deep ensembles. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
26. Li, Y., Padmanabhan, A., Zhao, P., Wang, Y., Xu, G.H., Netravali, R.: Reducto: on-camera filtering for resource-efficient real-time video analytics. In: Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication, pp. 359–376 (2020)
27. Lin, T.-Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
28. Lisota, K.: Understanding video frame rate and shutter speed (2020). https://kevinlisota.photography/2020/04/understanding-video-frame-rate-and-shutter-speed/

29. Liu, L., Li, H., Gruteser, M.: Edge assisted real-time object detection for mobile augmented reality. In: The 25th Annual International Conference on Mobile Computing and Networking, pp. 1–16 (2019)
30. Liu, W., et al.: SSD: single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
31. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: International Conference on Learning Representations (2018). https://openreview.net/forum?id=rJzIBfZAb
32. Minderer, M., et al.: Revisiting the calibration of modern neural networks. In: Advances in Neural Information Processing Systems, vol. 34 (2021)
33. Najafi, A., Maeda, S.I., Koyama, M., Miyato, T.: Robustness to adversarial perturbations in learning from incomplete data. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
34. Nyberg, O., Klami, A.: Reliably calibrated isotonic regression. In: Karlapalem, K., et al. (eds.) PAKDD 2021. LNCS (LNAI), vol. 12712, pp. 578–589. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-75762-5_46
35. Otani, A., Hashiguchi, R., Omi, K., Fukushima, N., Tamaki, T.: On the performance evaluation of action recognition models on transcoded low quality videos. arXiv preprint arXiv:2204.09166 (2022)
36. Paul, S., et al.: CamTuner: reinforcement-learning based system for camera parameter tuning to enhance analytics (2021). https://doi.org/10.48550/ARXIV.2107.03964. https://arxiv.org/abs/2107.03964
37. Qualcomm: How 5G low latency improves your mobile experiences (2019). Qualcomm_5G_low-latency_improves_mobile_experience
38. Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: a unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 815–823 (2015)
39. Sinha, D., El-Sharkawy, M.: Thin MobileNet: an enhanced mobilenet architecture. In: 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), pp. 0280–0285. IEEE (2019)
40. Tan, M., Pang, R., Le, Q.V.: EfficientDet: scalable and efficient object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10781–10790 (2020)
41. Viso.ai: Top 16 applications of computer vision in video surveillance and security. https://viso.ai/applications/computer-vision-applications-in-surveillance-and-security/
42. Wang, L., Sng, D.: Deep learning algorithms with applications to video analytics for a smart city: a survey. arXiv e-prints, arXiv-1512 (2015)
43. Wenkel, S., Alhazmi, K., Liiv, T., Alrshoud, S., Simon, M.: Confidence score: the forgotten dimension of object detection performance evaluation. Sensors **21**(13), 4350 (2021)
44. Witte, R., Witte, J.: A T-test for related measures. In: Statistics, pp. 273–285 (2017). ISBN: 9781119254515. www.wiley.com/college/witte
45. Xie, C., Tan, M., Gong, B., Wang, J., Yuille, A.L., Le, Q.V.: Adversarial examples improve image recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020
46. Xie, Q., Luong, M.T., Hovy, E., Le, Q.V.: Self-training with noisy student improves imagenet classification. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020

47. Zhang, B., Jin, X., Ratnasamy, S., Wawrzynek, J., Lee, E.A.: AWStream: adaptive wide-area streaming analytics. In: Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, pp. 236–252 (2018)
48. Zhang, H., Ananthanarayanan, G., Bodik, P., Philipose, M., Bahl, P., Freedman, M.J.: Live video analytics at scale with approximation and {Delay-Tolerance}. In: 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2017), pp. 377–392 (2017)
49. Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., Jordan, M.: Theoretically principled trade-off between robustness and accuracy. In: International Conference on Machine Learning, pp. 7472–7482. PMLR (2019)