Scalable Approximate Computing Techniques for Latency and Bandwidth Constrained IoT Edge

Anjus George(✉) and Arun Ravindran

University of North Carolina at Charlotte, Charlotte, NC 28223, USA
{ageorg28,aravindr}@uncc.edu

Abstract. Machine vision applications at the IoT Edge have bandwdith and latency constraints due to large sizes of video data. In this paper we propose approximate computing, that trades off inference accuracy with video frame size, as a potential solution. We present a number of low compute overhead video frame modifications that can reduce the video frame size, while achieving acceptable levels of inference accuracy. We present, a heuristic based design space pruning, and a Categorical boost based machine learning model as two approaches to achieve scalable performance in determining the appropriate video frame modifications that satisfy design constraints. Experimental results on an object detection application on the Microsoft COCO 2017 data set, indicates that proposed methods were able to reduce the video frame size by upto 71.3% while achieving an inference accuracy of 80.9% of that of the unmodified video frames. The machine learning model has a high training cost, but has a lower inference time, and is scalable and flexible compared to the heuristic design space pruning algorithm.

Keywords: Edge computing · IoT · Approximate computing · Machine learning · Machine vision

1 Introduction

Internet-of-Things (IoT) applications increasingly utilize machine vision for a variety of challenging tasks including autonomous driving, pedestrian safety, public security, and occupational health and safety. Such machine vision applications require continuous stream of video data generated by multiple cameras deployed in the field of operation. Due to latency and bandwidth constraints, the machine vision applications that process these video frames operate at the Edge of the network (Edge computing [3,4,19,28,29]). Transferring large amount of data generated by the cameras at the Edge in real-time results in high demand on the network bandwidth. Additionally many of these vision applications are latency sensitive - that is timely recognition of objects and their activity is important since events need to be responded within tight deadline constraints.

S. Paiva et al. (Eds.): SmartCity360° 2020, LNICST 372, pp. 274–292, 2021.
https://doi.org/10.1007/978-3-030-76063-2_20

Due to cost and ease of installation reasons, IoT devices deployed at the Edge make use of wireless technologies (WiFi (802.11ac) and Bluetooth (BLE)) as communication medium to transfer large amount of real-time video data. Data transmitted in the wireless channel (operating in the unlicensed bands) is prone to large variations in the latency due to interference, both from peer video cameras and unrelated external sources. The interference causes the data transfer network latency to exceed real-time response times and hence application set real-time bounds. Achieving a bounded latency in the face of unpredictable latency variations will require the ability to vary the information content transmitted from the cameras so as to match the channel conditions. However, varying the information content in the data (video frames in case of machine vision applications) will impact the application object/event detection accuracy as well.

In this paper we propose the use of approximate computing to meet dynamic network latency and bandwidth constraints at the Edge. Approximate computing is based on the idea that in some applications, selective inaccuracies in computation can be tolerated to achieve gains in efficiency [24]. Machine vision applications can potentially make use of approximate computing since they can tolerate compromises on object/event detection accuracy resulting from selective loss of video frame quality. We use this observation to investigate multiple video frame quality modification techniques (tuning knobs) so that the resulting reduction in data size satisfies latency and bandwidth constraints. Further we present two algorithms - the first based on a heuristic pruning of the large search space, and the second based on machine learning, to determine the settings of the tuning knob that simultaneously satisfy video frame size, and detection accuracy requirements. Experimental results indicate that for object detection vision application based on EfficientNet [31,32] Deep Learning architecture on the Microsoft COCO 2017 [20] dataset, we obtain an average video size reduction of 71.33% with an inference accuracy of 80.93% of that of the unmodified video frames.

This paper makes the following contributions -

- Investigates the applicability of approximate computing by characterizing the impact of video frame quality on machine vision accuracy.
- Identifies multiple video frame quality modification techniques (tuning knobs) that reduce video frame size.
- Presents pruning heuristic and machine learning based approaches to rapidly determine the settings of the tuning knob that simultaneously satisfy video frame size, and detection accuracy requirements.
- Presents extensive experimental evaluation of the approaches presented for object detection machine vision application using the EfficientNet Deep learning architecture on the Microsoft COCO 2017 dataset.

The rest of the paper is organized as follows - Sect. 2 provides a brief overview of related work in approximated computing, machine learning and video filtering at the Edge. In Sect. 3 we present the study of impact of video frame size on network latency. Section 4 lists multiple video frame quality tuning knobs and characterizes their impact on machine vision application inference accuracy.

Section 5 describes two scalable algorithms to identify tuning knobs that satisfy video frame size and application inference accuracy constraints. Section 6 experimentally evaluates the algorithms using publicly available dataset and machine vision application. Section 7 discusses the applicability of these algorithms in light of a representative Edge computing system. Section 8 concludes the paper.

2 Related Work

In this Section, we present the state of the art work done on application of machine learning, video filtering at the Edge and application of approximate computing.

In [25], Pakha et al. introduce the idea of control knobs such as frame selection and area cropping to parametrize a custom video protocol that streams videos from cameras to Cloud servers to perform neural-network-based video analytics. They developed a server driven video transmission protocol called SimpleProto that utilizes the tradeoffs between bandwidth usage and inference accuracy. In [6] Canel et al. proposes a new edge-to-cloud system called FilterForward that backhauls only relevant video frames from cameras to datacenter applications with the help of lightweight edge filters. However, unlike our approach, they perform computationally expensive Deep neural network based object detection at the client node.

Machine learning has been widely used at the wireless Edge for various applications such as resource management, networking, mobility management and localization [8,11,21,30,34]. In [34] Zhu et al. introduces a set of new machine learning-driven communication techniques. In [21] Mao et al. presents a taxonomy of applications of Deep Learning at different layers of wireless networks. Some of them include channel resource allocation, traffic prediction and link evaluation in data link layer and session scheduling and OS resource management for upper layers in the network stack. In contrast to these works where machine learning is applied at the lower layers of the network stack, our focus is on the application of machine learning at the application layer.

In [24], Mittal provides a survey of approximate computing techniques. Strategies for approximation at the code level such as loop perforation, and at the architecture level such as reduced precision operations are discussed. Regarding applications of approximate computing to Deep Learning, Chen et al. [7] use approximate computing to accelerate network training, while Ibrahim et al. [16] explore the use of approximate computing to realize Deep Learning networks on resource constrained embedded platforms. While our focus is on the use of approximate computing to satisfy network latency, and bandwidth constraints, in these works approximate computing is targeted towards reducing the computational load.

3 Impact of Video Frame Size on Network Latency

Size of video frames varies depending on the information content present in them. We initially explored the latency experienced by video frames with different sizes in the Wi-Fi channel using an IoT Edge testbed.

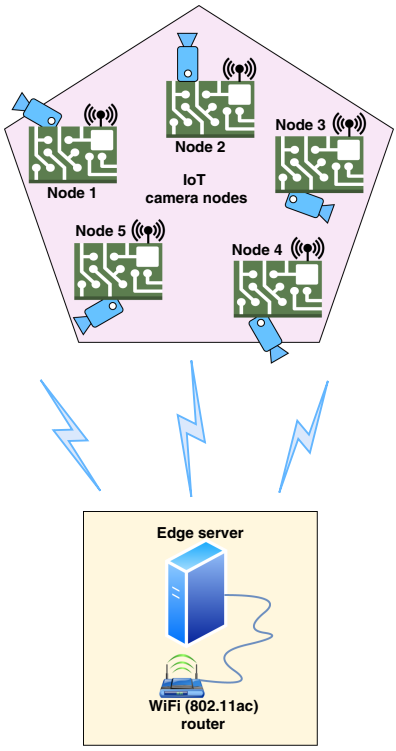


Fig. 1. IoT Edge test bed. IoT nodes equipped with cameras record and transmit video frames to an Edge server through a WiFi (802.11ac) wireless router. The Edge server runs machine vision algorithms on the received video frames for object detection, tracking and event prediction.

Figure 1 shows the IoT Edge test bed set up for Wi-Fi latency characterization purpose. The test bed consists of low power embedded boards (with 8-core ARMv8.2 based CPU) equipped with cameras, and a workstation furnished with an Nvidia GeForce 1060 GPU. The embedded boards and the workstation are termed as IoT camera nodes and Edge server respectively. Both IoT camera nodes and Edge server run Linux. The wireless link consists of a NETGEAR Nighthawk XR700 access point that uses 802.11ac (5 GHz) Wi-Fi standard. The Edge server is connected to the access point through Ethernet, while the IoT camera nodes connect to the access point through the 802.11ac Wi-Fi link. The IoT camera nodes are placed at 6m from the access point.

We characterized the Wi-Fi latency of video frames when they are transmitted from a test camera node to the Edge server. We use the term network latency to denote the latency experienced by video frames over the Wi-Fi channel when they are transmitted from IoT cameras nodes to the Edge server. This setup emulates an operational scenario where cameras produce variable size video frames depending on the scene dynamics in the area of observation.

Figure 2 shows the variation in network latency at different video frame sizes. The video frames of different sizes are chosen from publicly available vision dataset [26] and each measurement is taken as the average of 10 measurements. From this measurements, we note that the network latency shows an approximately linear variation with video frame size. Video frames with reduced size can be potentially transmitted with reduced network latency. Similar reductions in bandwidth requirements are possible by reducing the video frame size.

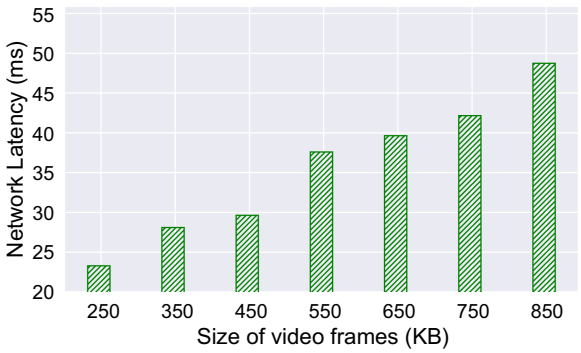


Fig. 2. Characterization of the impact of video frame size on video frame transfer network latency.

We note that the latency and bandwidth requirements are specified by the application, the wireless technology used, and cost constraints. Additionally, dynamically varying channel conditions due to interference, results in further constraint on bandwidth availability.

4 Video Frame Quality Tuning Knobs

As seen in Sect. 3, latency and bandwidth constraints can be satisfied by changing the video frame size. In this section we explore multiple video frame quality modification techniques which when applied on video frames reduce the frame size. Further, we investigate the impact of the reduced frame size on the machine vision application inference accuracy. The techniques, referred to as tuning knobs, are described below.

1. **Knob1 - Colorspace modifications:** Video frames can be converted from one colorspace to another resulting in total size reduction. We select 8 such colorspace modifications which are BGR $\leftrightarrow$ Gray, BGR $\leftrightarrow$ XYZ, BGR $\leftrightarrow$ HSV, BGR $\leftrightarrow$ HLS, BGR $\leftrightarrow$ LAB, BGR $\leftrightarrow$ LUV, BGR $\leftrightarrow$ YUV and BGR $\leftrightarrow$ CrCb. Our choice of color space modifications can reduce the video frame size by as much as 60%.
2. **Knob2 - Blurring:** Video frames can be blurred by passing them through various low pass filters. We chose the filter kernel sizes of 5, 8, 10, 15 and 20 as possible knob settings. Blurring the video frames can reduce the video frame size by as much as 75%.
3. **Knob3 - Denoising:** Noise content in video frames can be removed by passing them through denoising filters. The 5 knob settings selected for this knob are denoising filter strengths of 3, 10, 15, 20 and 30. Denoising knob can reduce the frame size upto 67%.
4. **Knob4 - Contrast stretching:** Contrast stretching can be done on video frames by performing range normalization over the frame pixel array. The norm values for range normalization are chosen from 0.3 to 0.9 with intervals of 0.1. Contrast stretching using these knob settings can achieve upto 70% size reduction.
5. **Knob5 - 2D filtering:** The 2D filtering approach convolves a video frame with a kernel and removes the noise in the frame. This knob could reduce the frame size as much as 68% when filter kernel sizes of 5, 6, 7 and 8 are applied on the video frames.
6. **Knob6 - Gaussian filtering:** A video frame can be convolved with a Gaussian kernel to remove Gaussian noise from the video frame. Selected kernel sizes are 5, 11, 21, 31 and 51 and achieved size reduction is 79%.
7. **Knob7 - Median filtering:** The median filtering technique computes the median of all the pixels under a kernel window and the central pixel is replaced with this median value. This technique is highly effective in removing salt-and-pepper noise from video frames. By choosing the knob settings as 5, 9, 11, 13 and 19, frame size reduced upto 72%.
8. **Knob8 - Bilateral Filtering:** Bilateral filtering removes noise in video frames while preserving the sharp edges in them. For this knob the filter sizes are chosen to be 10, 30 and 50 and filter sigma values are chosen as 70, 150 and 200. Upon application of this knob video frames reduced in its size by 64%.
9. **Knob9 - Erosion:** Erosion is a type of morphological transformation that erodes away the boundaries of objects in video frames and is useful in removing small white noises. Choosing erosion filter kernel sizes to be 5, 10 and 15 we could achieve size reduction of 62%.
10. **Knob10 - Dilation:** Dilation is another type of morphological transformation which is the opposite of erosion. This knob dilates video frames using a kernel structure. The kernel structure sizes chosen are 5, 8 and 10 with size reductions of upto 52%.

Note that each of these knobs can be set independent of the other knobs potentially resulting in a large ($\approx$22 million) search space.

4.1 Impact of Tuning Knobs on Application Inference Accuracy

Reducing the information content by applying these tuning knobs on video frames could impact the object/event detection inference accuracy of machine vision applications consuming these video frames. In general, the impact is dependent on the particular machine vision application. We choose object detection as the machine vision application since it is widely used, and is a basis of other computer vision tasks such as object tracking, and activity detection. The object detector EfficientDet [31] is based on EfficientNet [32], a deep learning neural network developed by Google brain team. It achieves state-of-the-art accuracy while being up to 9x smaller than competing models and using significantly less computation. Microsoft COCO [20] is a publicly available vision dataset consisting of 91 object categories (classes) with a total of 2.5 million labeled instances in 328K images. We input the original and modified video frames from COCO 2017 dataset to EfficientDet to generate object detections on video frames with bounding boxes drawn around the objects.

In order to evaluate these detections, each ground truth bounding box for that frame (publicly available) is matched exclusively to the outputted bounding box based on highest Intersection over Union (IoU) overlap. Positive matches with an IoU greater than a threshold are considered True Positives; result bounding boxes without ground truth matches are considered False Positives; and each unmatched ground truth box is considered a False Negative. These records are utilized for mAP (Mean Average Precision) calculation.

For object detection, we utilize the mAP metric with an Intersection-over-Union (IoU) threshold of 0.5. Equation 1 defines the calculation for mAP. Precision is $\frac{TP}{(TP+FP)}$ and Recall is $\frac{TP}{(TP+FN)}$, where TP, FP, and FN are the number of True Positives, False Positives, and False Negatives respectively. Average Precision calculated at a single IoU threshold (0.5 in our case) for a single object class is denoted as, $AP^{IoU=.5}$. Finally, mAP is obtained by averaging $AP^{IoU=.5}$ over different classes in the chosen dataset.

$$mAP = \frac{1}{\#classes} \sum_{class \in classes} AP^{IoU=.5}[class] \tag{1}$$

where $\#classes$ represents number of classes of objects in the video frames and $AP^{IoU=.5}[class]$ represents $AP^{IoU=.5}$ for a specific object class.

Figures 3 and 4 show the impact of application of tuning knobs, blurring and denoising (Knobs 2 and 3) on and video frame size and mAP. The mAP of unmodified (without the application of any tuning knob) video frame is considered as the baseline mAP. The mAP values obtained after applying tuning knobs on video frames are normalized with respect to the baseline mAP. This characterization corresponds to mAP generated (using EfficientDet-D0 [2] model) on 300 images chosen from COCO 2017 dataset.

Figures 3 and 4 demonstrate that the application of these tuning knobs can reduce the video frame size 75% and application inference accuracy as much as 47%. A similar trend for video frame size reduction and accuracy degradation can be observed for other tuning knobs (Knobs 1 and 4–10) as well.

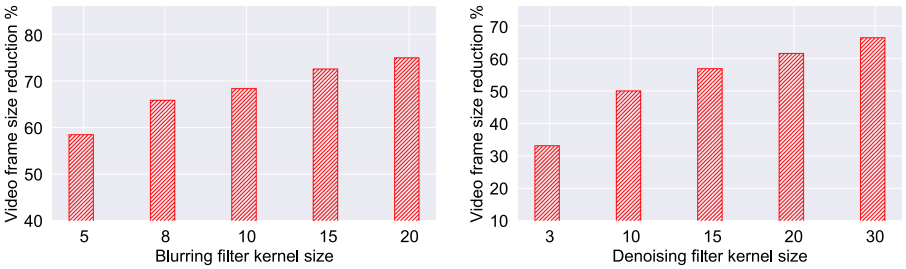


Fig. 3. Characterization of the impact of application of tuning knobs, (a) blurring (knob 2) and (b) denoising (knob 3) on video frame size.

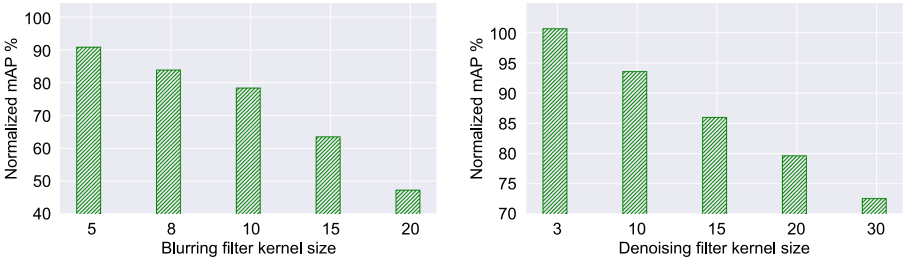
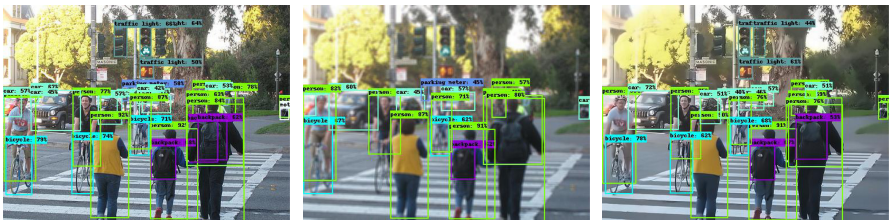


Fig. 4. Characterization of the impact of application of tuning knobs, (a) blurring (knob 2) and (b) denoising (knob 3) on mAP (application inference accuracy).

From Fig. 4, note that setting 1 of knob 2 reduced the mAP to 90.8% of the baseline and setting 2 of knob 3 reduced mAP to 93.5% of baseline. Figure 5 shows the visual impact of application of setting 1 of knob 2 and setting 2 of knob 3 on object detections. Because of the effect of quality modification, both the video frames have missed detections for a few objects. But note that both the modified video frames (Figs. 5b and 5c) still preserve most of the true detections for object categories such as person and bicycle.

This observation can be used to exploit the trade off between video frame size and video frame quality (approximate computing) inherent in machine vision applications. Approximate computing exploits the gap between the extent of accuracy needed by the applications and provided by the computing system for achieving various optimizations [24]. A lower quality video frame has a smaller size, and hence can be transferred with lower network latency and bandwidth requirements. If a lower quality frame can provide acceptable machine vision application inference accuracy, then a lower quality frame could be transferred from the IoT camera node to the Edge server satisfying network performance constraints.



(a) (b) (c)

Fig. 5. Visual impact on object detections before and after the application of tuning knobs. (a) Unmodified video frame, video frame after applying (b) blurring knob with a kernel size of 5 and (c) denoising knob with a kernel size of 10.

5 Scalable Approximate Computing Algorithms

5.1 Knob Search Space

The observations from previous section indicate that tuning knobs provide a mechanism to send reduced size video frames with acceptable inference accuracy. It should be noted that the network determines the size of the video frame that needs to be transmitted, while the machine vision application decides the inference accuracy metric (for e.g. mAP for object detection). To select the knob settings that constitute the targeted frame size and target inference accuracy metrics, all combinations of settings of identified knobs need to be characterized. However, combining the knob settings for different tuning knobs results in a combinatorial explosion of the design space as seen in Eq. 2. For tuning knobs from 1 to n, this number can be represented using the equation,

$$k_{total} = k_1 \times k_2 \times \ldots \times k_i \times \ldots \times k_n \tag{2}$$

where k_{total} represents the total number of knob combinations and k_i represents number of knob settings for the i^{th} knob. For the 10 tuning knobs identified in Sect. 4 this results in a total number of 22,394,880 knob combinations. Computing and storing the resulting frame size (by applying knob combinations sequentially on video frames) and the machine vision inference metric (by feeding the modified video frames to the machine vision application) become prohibitively expensive.

We therefore explore two scalable algorithms to solve the combinatorial explosion problem - a design space pruning heuristic algorithm, and a machine learning based algorithm.

5.2 Pruning Heuristic Algorithm for Knob Selection

We use pruning heuristic algorithm to successively filter out knob combinations that result in lower performance on the inference metric. In this algorithm we consider the k_{total} knob combinations of Eq. 2 as the heuristic decision space

of the problem. The algorithm works as follows. The search space consists of n tuning knobs with each tuning knob i consisting of k_i tuning knob settings. In the first step of this algorithm we change the knob setting for a single tuning knob (while keeping settings of other knobs at their defaults) and calculate the frame size and inference metric. We repeat this process for all the knob settings independently for each knob. We then filter out knob settings that results in a low performance on the inference metric (lower than a application specified threshold). This completes the first step of the algorithm. In the second step, we choose pairs of knob combinations from distinct knob combinations obtained from the first step. A filtering step similar to the first step is then applied to weed out knob combinations with lower performance on the inference metric. This completes the second step of the algorithm. The process is repeated next considering 3 distinct knob combinations from the knob setting obtained from step 2. The algorithm terminates when all n distinct knob combinations are considered in the *n-th* step. Each step i, prunes the design space by eliminating low performing knob combinations.

The algorithm for the pruning heuristic knob selection is outlined in the pseudo code shown in Listing 1.

Algorithm 1: Pruning heuristic algorithm

```
Result: Video frame quality knob settings
InferenceAccuracy = Iacc ;
InferenceAccuracyThreshold = Iacc_Thres ;
numKnobs = n ;
step_i = 1;
while step_i <= n do
    from n knobs, choose i settings = nC_i ;
    for each setting k_i in nC_i do
        apply setting k_i on video frame;
        calculate video frame size;
        calculate Iacc on video frame;
        if Iacc < Iacc_Thres then
            discard setting k_i;
    end
    increment step_i by 1
end
```

5.3 Machine Learning Algorithm for Knob Selection

In contrast to pruning the entire knob space to identify useful knobs using the pruning heuristic algorithm, a machine learning algorithmic approach can be used predict video frame size and inference accuracy. To build the model, we first

collect the input data required to train the model. We sample a small subset of knob combinations from the total possible set of knob combinations, and evaluate the video frame size and machine vision application inference accuracy for the sampled knob combinations. We then develop a machine learning model and train it using the sampled knob data to predict the size and inference accuracy for remaining knob combinations. The detailed steps of the algorithm are described below.

The first step of the algorithm is the data collection process. Here the data represents tuning knob combinations identified using Eq. 2, the resulting video frame size after applying tuning knobs, and the associated inference accuracy of machine vision application. Since the knob sample space is large calculating the frame size and inference accuracy for all the knob combinations is a time consuming process. Therefore we select a representative set of samples from the knob search space using a sampling method. A Latin Hypercube Sampling (LHS) [22,23] method can be used to generate near-random samples from the multi-dimensional search space of knobs. LHS with a 'maximun' criteria is specifically chosen to avoid the correlation between samples by maximizing the minimum distance between samples [17].

The next step is to modify the video frames by applying the sampled knob combinations, and recording the resulting video frame sizes. The modified video frames are fed to the machine vision application to generate the inference accuracy for each knob combination. At the end of this step for each knob combination we have a frame size and inference accuracy value.

We aim to two build two models - one to predict frame size and another to predict inference accuracy values; we consider the knob combinations as common input features to both models. We note that all the input features (knob combinations) have their dependent variable values (frame size and inference accuracy) labeled. Hence we conclude that a supervised learning method needs to be used for this kind of labeled data samples. Another observation is that both the dependent variables are real-valued quantities that need to be predicted. Therefore a regression model would be best suited for this type of data. Also we observed that knobs have a non-linear dependence on both frame size and inference accuracy. Additionally, video frames with same size map to different inference accuracy values. Due to the complex dependence of knobs on frame size and inference accuracy, we chose to proceed with a non-linear machine learning model to estimate the dependent variables.

We experimented with multiple non-linear models such as polynomial regression, Support vector machine [13], Random forest regression [5] and Decision tree regression [27]. But none of the models could provide sufficient level of prediction accuracy. Then we explored a recently proposed model called CatBoostRegressor [12] from open source gradient boosting library CatBoost [1] because of the categorical nature of the input features (knob combinations). Models from CatBoost library outperforms state of the art gradient boosting libraries such as XGBoost [9] and LightBGM [18] in terms of model quality and training speed. CatBoost is a decision tree based library that uses two phases to predict the next tree.

In CatBoost the second phase is performed using traditional Gradient Boosting Decision Tree (GBDT) [14] scheme and for the first phase a modified version of GDBT is used.

6 Implementation and Results

In this section we present the experimental evaluation of the scalable approximate computing algorithms described in Sect. 5.

6.1 Pruning Heuristic Algorithm

We implemented all the knobs identified in Sect. 4 using open source computer vision library OpenCV [33]. The video frames were chosen from Microsoft COCO 2017 dataset. A set of 300 video frames belonging to object classes person, bicycle, car and traffic light were selected for the experimental evaluation. We chose the object detector model EfficientDet [31] (based on EfficientNet [32]) as the machine vision application to evaluate its inference metric (mAP) on modified video frames. EfficientDet model D0 was chosen among models D0 to D7 because it is least complex model (has low number of model parameters and low computation latency) and is suited for resource constrained Edge devices.

A set of 146 knob combinations was identified using the pruning heuristic algorithm from the 22 million knob search space explained in Sect. 5.1. To identify these knob combinations we set the inference accuracy threshold (mAP) of the EfficientDet object detector to be >80% of the baseline mAP. Note that the application of all 146 combinations of the knobs identified above result in different sizes of video frames, all lower than the original.

Figure 6 shows the plot of the normalized mAP expressed as a percentage vs. video frame size for video frames (from COCO dataset) modified using the filtered knob combinations. The video frame size buckets in Fig. 6 corresponds to different combinations of the knob settings with different resulting mAP. Note that higher mAP indicates higher inference accuracy for EfficeintDet. (The reason for size bucket 100–120 showing a max normalized mAP greater than 100% is because of the effect of knob setting 1 of knob3-denoising, which actually caused the mAP to become slightly higher than baseline mAP.)

Using the pruning heuristic algorithm we could identify knob settings that achieved 71.33% video frame size reduction with mAP score of as much as 82.37% of the baseline mAP. This key observation enables transmission of smaller sized images with less mAP from the IoT camera nodes to the Edge server when subjected to network latency and bandwidth constraints.

Figure 7 shows the visual impact of application of tuning knobs (colorspace modification, denoising, contrast stretching and gaussian filtering) resulting in a normalized mAP of 82.37% and size reduction of 71.33%. We note that the object detections in Fig. 7b contains most of the true detections (from unmodified video frame) except a few missed detections such as a bicycle, a parking meter, one car

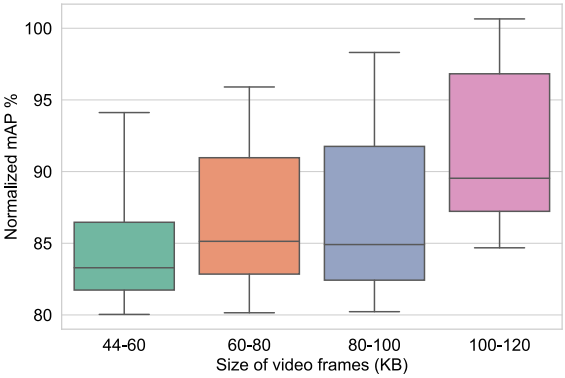


Fig. 6. Normalized mAP expressed as a percentage for EfficientDet mAP for video frames from COCO dataset. Note that each video frame bucket corresponds to different combinations of the knob settings with different resulting mAP.

and two traffic lights. Two wrong detections resulting here are a bicycle detected as vase, and a person detected as car.

Using Eq. 2 we estimate the total number of knobs need to be evaluated to be 354 by executing the pruning heuristic algorithm (Listing 1) for steps from 1 to 4 and using the 10 knobs and their settings identified in Sect. 4. The application set inference accuracy threshold was set to be >80% of the baseline mAP. The computation time taken to evaluate mAP scores (using EfficientDet-D0 on an Nvidia GeForce 1060 GPU) for 354 combinations for 300 video frames from COCO dataset is 10.03 h. The total time taken for evaluations escalates substantially when number of knobs and/or settings for each knob increase.

(a) (b)

Fig. 7. Visual impact on object detections before and after the application of tuning knobs. (a) unmodified video frame and (b) video frame after applying knobs - colorspace modification, denoising, contrast stretching and gaussian filtering.

6.2 Machine Learning Algorithm

We sampled 1000 knob combinations (using Latin Hypercube Sampling) from the knob sample space to generate the input data for the model. Next, the sampled knob combinations are applied to video frames to calculate the video frame size. All the knob settings are represented as categorical features before feeding into the model. We then divided the samples into train and test samples using an 80–20% split. As explained in Sect. 5.3, we chose the machine learning model CatBoostRegressor [12] model from open source library for gradient boosting library catboost [1]. We trained the model and predicted the frame sizes for the knob combinations from the test sample set. We obtained train and test Root Mean Square Errors (RMSEs) for this model as 0.74 KB and 1.5 KB respectively.

Next we take knob settings along with their video frame sizes as input features to construct a model to predict the machine vision application inference accuracy. The inference accuracy metric chosen for EfficientDet is mAP. With knob settings represented as categorical features, we use the CatBoostRegressor model to predict mAP scores for test knob combinations. For this model we achieved train and test RMSEs as 0.51% and 1.6% (normalized mAP) respectively.

Using the machine learning algorithm we could identify knob settings that achieved 71.37% video frame size reduction with mAP score of as much as 80.93% of the baseline mAP.

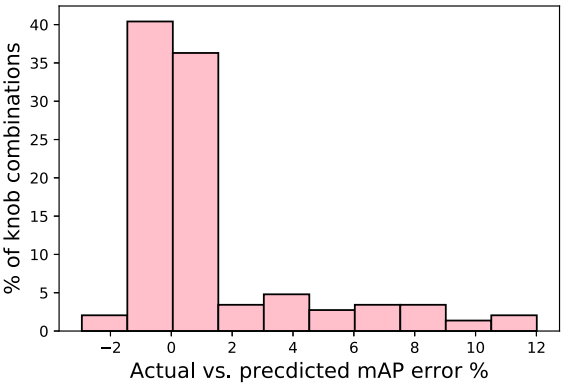


Fig. 8. Histogram showing the distribution of error between actual and predicted mAP scores for percentage of knob combinations chosen to test the mAP ML model. 82.19% of the knob combinations have predicted mAP error variation of only ±3% of actual mAP.

Figures 8 and 9 show the percentage variation of actual and predicted mAP and video frame size with respect to percentage of knob combinations in test sample. Video frame size histogram (Fig. 9) shows 78.08% of the knob combinations fall within ±10% of video frame size error. The mAP histogram (Fig. 8)

shows that 82.19% of knob combinations fall within ±3% of mAP error. Since we could predict accurately (with less than 10% error) most of the knob combinations' video frame size and inference accuracy using the constructed models, we conclude that the models are sufficiently accurate. The computation time taken to evaluate mAP scores (using EfficientDet-D0 on an Nvidia GeForce 1060 GPU) for 1000 knob combinations (used for training and testing the models) for 300 video frames from COCO dataset is 28.33 h.

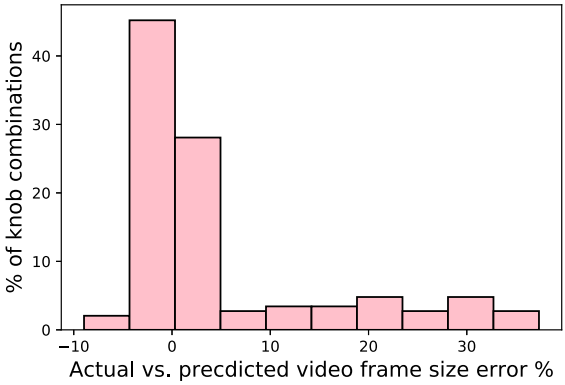


Fig. 9. Histogram showing the distribution of variation between actual and predicted video frame size for percentage knob combinations chosen to test the video frame size ML model. 78.08% of the knob combinations have predicted frame size error variation of only ±10% of actual frame size.

Comparing the two scalable approximate computing algorithms, the Categorical boost machine learning model based algorithm achieves comparable accuracies to the pruning heuristic algorithm (within ±3% error for mAP model, and ±10% for video frame size model). The machine learning approach is more scalable both in terms of the number of knobs, and the number of settings for each knob, since it only has a one time training cost.

7 System Integration

In the previous section we described how pruning heuristic and machine learning algorithms can be used to characterize video frame size and machine vision application inference accuracy for different combinations of video frame quality modifications. In this section we describe the applicability of these algorithms in a IoT Edge vision system.

Consider a multi-camera Edge vision system similar to the one shown in Fig. 1 deployed in a public space such as a traffic intersection for detecting pedestrians. The IoT nodes equipped with cameras stream live videos of the area under observation to the Edge server through a wireless network (for example

Wi-Fi). The Edge server aggregates the individual video streams from multiple cameras and run machine vision application to detect pedestrians to ensure road safety. The pedestrian detection application is latency sensitive and requires video frames from the cameras to be processed with in a short time window to meet the real time deadlines. The application informs real time latency and inference accuracy requirements to the IoT camera nodes. The camera nodes monitor current Wi-Fi channel conditions by sensing latency (varies depending on the channel interference) of video frame transmission in the channel. If the current latency exceeds the latency target set by the application video frames have to be modified by applying the tuning knobs describe in Sect. 4, hence reducing their size to meet target latency.

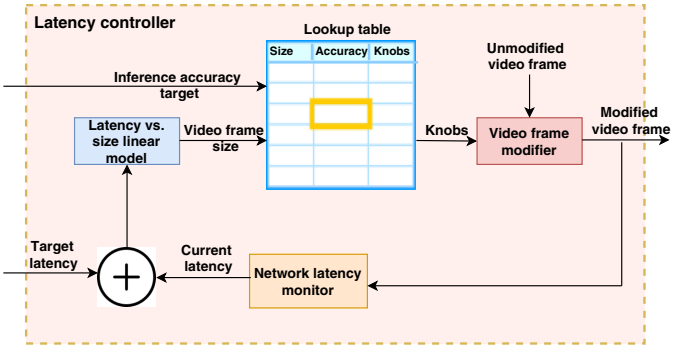


Fig. 10. Design of a latency controller for the Edge that uses knob selection algorithms explained in Sect. 5 to construct frame size, inference accuracy, knobs lookup table.

Video frame size that satisfy a latency value can be calculated from the latency characterization of Sect. 3. While reducing the size, the inference accuracy demand from the application also needs to be met. To facilitate the size reduction process in real time the IoT camera nodes can store the candidate knob combinations resulting from the algorithms described in Sect. 5 in a simple lookup table. The lookup table entries will be the frame size, inference accuracy value and the settings of the individual image modification knobs. The entries in the lookup table can then be used to search for the tuning knob combination that satisfy both the latency and inference accuracy demands. A controller operating at the camera node can potentially perform latency control of video frames in real time [15]. Figure 10 shows the block diagram of a possible closed loop control.

Alternately, the machine learning model could be evaluated in real-time to predict the inference accuracy, and frame size resulting from a given knob combination. In this case the machine learning models act as objective functions that need to be minimized or maximized in a multi-objective optimization space [10]. The goal would be to find a set of solutions (knob combinations) as close as possible to the conflicting objectives. The resulting knob combinations can

then be used to populate the lookup table when the controller is operated in the real time. For instance the objectives can be video frame size upper bound that need to be minimized and inference accuracy lower bound that need to be maximized. Note that the network conditions determines the video frame size constraint whereas the machine vision application determines the inference accuracy constraint.

8 Conclusion

In this work we explored the problem of latency and bandwidth constrained operation at the IoT Edge for machine vision applications through the technique of approximate computing. We observe that despite reducing video frame sizes, acceptable inference accuracies are possible. We identified a number of video frame transformation techniques, that can result in reduced frame size, and effectively acts as latency/bandwidth tuning knobs for the application. Since the design space suffers from a combinatorial explosion problem precluding exhaustive characterization of knob combinations, we investigated scalable algorithms to facilitate our approximate computing approach. We experimentally evaluated two approaches - a heuristic based pruning algorithm of the design space, and a Categorical boost machine learning model based algorithm. Both approaches were able to reduce the video frame size by upto 71.3% while achieving a inference accuracy of 80.9% of the inference accuracy of the unmodified video frames. The lower frame size can facilitate operation in a bandwdith constrained environment, as well as results in a lower communication latency. The machine learning model has a fixed training cost compared to the heuristic based pruning algorithm, while being inherently more scalable. We also briefly discuss how the tuning knobs resulting from the two algorithms could be integrated in a latency controller in an autonomous system targeted at latency sensitive IoT Edge machine vision applications.

Among future work directions, we could incorporate the approximate computing based algorithms proposed in the paper in latency and bandwidth constrained Edge and Cloud computing systems. We have targeted object detection in this work, since it is a basic kernel in many machine vision tasks. We could also explore the use of the proposed techniques on other machine vision tasks such as object tracking, image segmentation, and activity detection.

References

1. Catboost. https://catboost.ai/. Accessed 12 Sep 2020
2. Efficientdet. https://github.com/google/automl/tree/master/efficientdet. Accessed 12 Sep 2020
3. Bonomi, F., Milito, R., Natarajan, P., Zhu, J.: Fog computing: a platform for Internet of Things and analytics. In: Bessis, N., Dobre, C. (eds.) Big Data and Internet of Things: A Roadmap for Smart Environments. SCI, vol. 546, pp. 169–186. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-05029-4_7

4. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the Internet of Things. In: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, MCC 2012, pp. 13–16. Association for Computing Machinery, New York (2012). https://doi.org/10.1145/2342509.2342513
5. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001). https://doi.org/10.1023/A:1010933404324
6. Canel, C., et al.: Scaling video analytics on constrained edge nodes. In: Proceedings of the 2nd SysML Conference (SysML 2019), Palo Alto, CA, pp. 1–5 (2019)
7. Chen, C., Choi, J., Gopalakrishnan, K., Srinivasan, V., Venkataramani, S.: Exploiting approximate computing for deep learning acceleration. In: 2018 Design, Automation Test in Europe Conference Exhibition (DATE), pp. 821–826 (2018)
8. Chen, M., Challita, U., Saad, W., Yin, C., Debbah, M.: Artificial neural networks-based machine learning for wireless networks: a tutorial. IEEE Commun. Surv. Tutor. **21**(4), 3039–3071 (2019)
9. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2016, pp. 785–794. Association for Computing Machinery, New York (2016). https://doi.org/10.1145/2939672.2939785
10. Deb, K., Kalyanmoy, D.: Multi-Objective Optimization Using Evolutionary Algorithms. Wiley, USA (2001)
11. Deng, S., Zhao, H., Fang, W., Yin, J., Dustdar, S., Zomaya, A.Y.: Edge intelligence: the confluence of edge computing and artificial intelligence. IEEE Internet Things J. **7**(8), 7457–7469 (2020)
12. Dorogush, A.V., Ershov, V., Gulin, A.: CatBoost: gradient boosting with categorical features support. ArXiv abs/1810.11363 (2018)
13. Evgeniou, T., Pontil, M.: Support vector machines: theory and applications. In: Paliouras, G., Karkaletsis, V., Spyropoulos, C.D. (eds.) ACAI 1999. LNCS (LNAI), vol. 2049, pp. 249–257. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44673-7_12
14. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Ann. Stat. **29**(5), 1189–1232 (2001). https://doi.org/10.1214/aos/1013203451
15. George, A., Ravindran, A.: Latency control for distributed machine vision at the edge through approximate computing. In: Zhang, T., Wei, J., Zhang, L.-J. (eds.) EDGE 2019. LNCS, vol. 11520, pp. 16–30. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-23374-7_2
16. Ibrahim, A., Osta, M., Alameh, M., Saleh, M., Chible, H., Valle, M.: Approximate computing methods for embedded machine learning. In: 2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS), pp. 845–848 (2018)
17. Joseph, V.R., Hung, Y.: Orthogonal-maximin Latin hypercube designs. Statistica Sinica **18**(1), 171–186 (2008). http://www.jstor.org/stable/24308251
18. Ke, G., et al.: LightGBM: a highly efficient gradient boosting decision tree. In: NIPS (2017)
19. Lee, E.A., et al.: The swarm at the edge of the cloud. IEEE Des. Test **31**(3), 8–20 (2014)
20. 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
21. Mao, Q., Hu, F., Hao, Q.: Deep learning for intelligent wireless networks: a comprehensive survey. IEEE Commun. Surv. Tutor. **20**(4), 2595–2621 (2018)

22. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **21**(2), 239–245 (1979). http://www.jstor.org/stable/1268522
23. McKay, M.D.: Latin hypercube sampling as a tool in uncertainty analysis of computer models. In: Proceedings of the 24th Conference on Winter Simulation, WSC 1992, pp. 557–564. Association for Computing Machinery, New York (1992). https://doi.org/10.1145/167293.167637
24. Mittal, S.: A survey of techniques for approximate computing. ACM Comput. Surv. **48**(4), 1–33 (2016). https://doi.org/10.1145/2893356
25. Pakha, C., Chowdhery, A., Jiang, J.: Reinventing video streaming for distributed vision analytics. In: Proceedings of the 10th USENIX Conference on Hot Topics in Cloud Computing, HotCloud 2018, p. 1. USENIX Association, USA (2018)
26. Rasouli, A., Kotseruba, I., Tsotsos, J.K.: Are they going to cross? A benchmark dataset and baseline for pedestrian crosswalk behavior. In: 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), pp. 206–213 (2017)
27. Rokach, L., Maimon, O.: Decision trees. In: Maimon, O., Rokach, L. (eds.) Data Mining and Knowledge Discovery Handbook. Springer, Boston (2005). https://doi.org/10.1007/0-387-25465-X_9
28. Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N.: The case for VM-based cloudlets in mobile computing. IEEE Pervasive Comput. **8**(4), 14–23 (2009)
29. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: vision and challenges. IEEE Internet Things J. **3**(5), 637–646 (2016)
30. Sun, Y., Peng, M., Zhou, Y., Huang, Y., Mao, S.: Application of machine learning in wireless networks: key techniques and open issues. IEEE Commun. Surv. Tutor. **21**(4), 3072–3108 (2019)
31. Tan, M., Pang, R., Le, Q.V.: Efficientdet: scalable and efficient object detection. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10778–10787 (2020)
32. Tan, M., Le, Q.: EfficientNet: rethinking model scaling for convolutional neural networks. Proceedings of Machine Learning Research, vol. 97, pp. 6105–6114. PMLR, Long Beach (09–15 June 2019). http://proceedings.mlr.press/v97/tan19a.html
33. Vision, O.S.C.: OpenCV documentation (2019). https://docs.opencv.org
34. Zhu, G., Liu, D., Du, Y., You, C., Zhang, J., Huang, K.: Toward an intelligent edge: wireless communication meets machine learning. IEEE Commun. Mag. **58**(1), 19–25 (2020)