ADACONFIGURE: REINFORCEMENT LEARNING-BASED ADAPTIVE CONFIGURATION FOR VIDEO ANALYTICS SERVICES

Anonymous ICME submission

ABSTRACT

Deep convolutional neural networks (NN)-based video analytics services demand intensive computation resources and high inference accuracy. Static configuration is contrary to the dynamic nature of videos, so the static configuration would either waste computation resources (by picking an expensive configuration, such as the highest frame rate and resolution) or decrease inference accuracy. Knowing this, adapting video configuration is still extremely challenging: 1) The best video configuration is determined by confounding factors, including the characteristics of the input video, the various accuracy-demand services, and computation resource, etc. 2) The cost of adaptive configuration profiling may far exceed the benefits of adaptive configurations. To tackle these challenges, we propose a reinforcement learning-based video analytics configuration framework, AdaConfigure. In particular, we design an agent that adaptively chooses the configuration according to the spatial and temporal features of video contexts. The design of reward carefully considers accuracy and computation resources to assess the impact of each configuration, guiding the agent learning to choose the best configuration for different accuracy-demand services. We leverage dividing video strategy and the extremely short choosing action time of agent to reduce profiling cost, which is only 0.2-2% overhead to the overall video analytics services. Our evaluation experiments using the object detection task show that our approach outperforms static configurations by achieves 10-35% higher accuracy with a similar amount of computation resources or achieves similar accuracy with only 60-90% of the computation resources.

1. INTRODUCTION

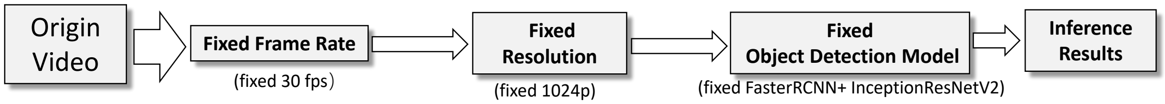
With the increasing demand for continuous video analytics in public safety and transportation, more and more cameras are being deployed to various locations. The video analytics are based on classical computer vision techniques as well as deep convolutional neural networks. In recent years, we have also witnessed the emergence of a large number of excellent models for target detection [1], such as FasterRCNN [2], RFCN [3], Multibox [4], SSD [5] and YOLO [6].

A video analytics application consists of a *pipeline* of several video processing modules, typically including a decoder, a selective sampling frame application, and a target detector. Such a pipeline always has multiple *knobs*, such as frame rate, resolution, and model (e.g., SSD+{MobileNet [7], ResNet [8]}, FasterRCNN+{ResNet,InceptionResNet [9]}). A combination of the knob values is video analytics *configuration*, and the configuration space grows *exponentially* with the number of knobs, and their values [10].

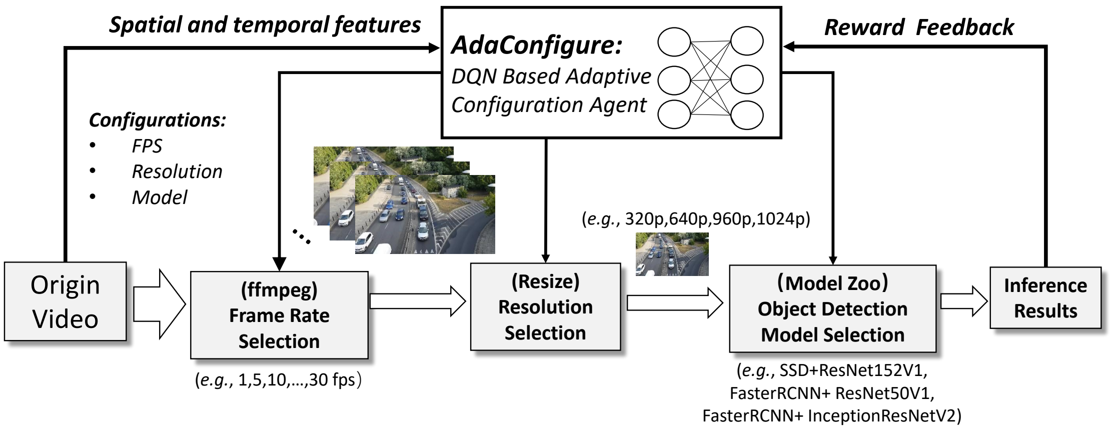
Different configurations directly affect accuracy and resource consumption. The best configuration is the one with the lowest resource demand whose accuracy is over the desired threshold, whcih can optimize the *trade-off* between accuracy and energy consumption. The *best* configuration for video analysis services often varies in minutes or even seconds [10]. As shown in figure 1(a), if one just uses a fixed static expensive configuration (e.g. only profiles the processing pipeline *once* to choose high frame rate and resolution) can be very precise, but it can also be a huge drain on resource consumption. Similarly, specifying a fixed static cheap (e.g., low resolution and small model) configuration will significantly reduce accuracy. The brief framework of our solution is shown in Figure 1(b). In this, we tackle the following design challenges.

- *Choosing the best configuration is a complicated decision-making problem that is challenging to solve by rules.* The best configuration for video analytics services changes significantly because the exact context of the videos varies over time and across space. For instance, tracking vehicles when traffic moves quickly requires a much higher frame rate than when traffic moves slowly, but each condition may vary by hour, minute, even second. Spatially, the characteristics of video content are different in different locations. For instance, cameras in downtown areas show more cars than the other cameras deployed in the suburbs. It is hard to make an exact rule to choose a configuration for the current context by profiling such complicated video characteristics.

- *Adaptive configuration would cause a huge extra overhead.* The number of possible configurations grows exponentially, and thousands of configurations can be combined with just a few knobs [10]. So exhaustive periodically (e.g., profile once per 4 seconds) configuration to find the best configuration is a highly unrealistic approach because it causes a huge extra over-



(a) Static configuration solution: fixed configuration for video analytics services



(b) AdaConfigure solution: reinforcement learning-based adaptive configuration framework

Fig. 1. Comparing to the static solution, our solution can adaptively update the configuration strategy based on the object detection model feedback

head, which may exceed the benefits of adaptive configurations. To significantly reduce the resource cost for adaptive configuration profiling requires one solution that automatically chooses a configuration instead of manually trying using various configurations, which is challenging to previous studies including [11, 10] since their approach pay attention to reduce search space algorithm and still try using various configurations to find the best configuration.

To address the complicated decision-making problem and reduce the huge extra overhead of manually trying using various configurations, we carefully design an adaptive configuration framework based on reinforcement learning, which is an excellent approach to solve this unsupervised complex-environmental problem. AdaConfigure can adaptively and automatically select the best configuration according to intrusive dynamics of video contexts, thus solving this difficult optimal configuration decision problem in a very low-cost way. To the best of our knowledge, we are the first to propose an adaptive video configuration solution that adaptively and automatically chooses a configuration according to the current video context. The main contributions of this paper are summarized as follows.

- We propose a Deep Q-learning Network-based [12] agent to adaptively pick the best video analytics configuration according to intrusive dynamics of video contexts. Also, we define the elements of the RL-based approach, such as actions, states, and rewards, to adapt the configuration to the current video context. In the state design, we extract the spatial and temporal features of the video context so that the agent can adaptively update the configuration over time and achieve superior performance in the multi-camera situation. In the reward design, we carefully consider both inference accuracy and computation resources to assess each configuration's impact. Also, to meet different accuracy-demand services, we leverage the balance factor in the reward function to train different agents. In our design, our solution can adaptively update configuration strategy over time, and it adaptive to different location-camera inferences and different accuracy-demand services.

- We leverage dividing video strategy and the extremely short choosing action time of agent to reduce profiling cost. In the evaluation, AdaConfigure achieves 10-35% higher accuracy with a similar amount of resources or achieves similar accuracy with only 60-90% of the resources. Our solution proves to be more efficient than static solutions and only creates an overhead of 0.2-2% to the overall video analytics services.

2. RELATED WORKS

2.1. Static configuration optimization

Several previous papers have considered optimizing video analytics services by either adjusting the configuration knobs or training specialized NN models. VideoStorm [13] profiles thousands of video analytics queries on live video streams over large clusters, achieving resource-quality tradeoff with multi-dimensional configurations. VideoEdge [14] introduces *dominant demand* to identify the best tradeoff between multiple resources and accuracy, and narrows the search space by identifying a "Pareto ban" of promising configurations. MCDNN [15] provides a heuristic scheduling algorithm to adaptively select model variants of different accuracy for deep stream processing under resource constraints. Focus [16] deconstructs video analytics into two phases, i.e., video ingests and video query. By tuning the share of computing resources of both phases, Focus achieves an effective and flexible tradeoff of video analytics's latency and accuracy. These algorithms all profile and optimize video analytics only once at the beginning of the video. In their works, video is divided into pictures at a constant frame rate, and the work of video analysis is regarded as a fixed task either. In other words, they do not handle changes in video stream context. But the optimal configurations do change over time because of the complex and changeable video stream contexts.

2.2. Dynamic configuration optimization

Some classic works study dynamic optimization problems. INFaaS [17] automatically selects a model, hardware architecture, and any compiler optimizations and makes scaling and resource allocation decisions when application load varies and the available resources vary over time. Quick-Adapt [18] uses descriptive statistics of security events data and fuzzy rules to enable a Big Data Cyber Security Analytics (BDCA) system to adapt to the changes in security events data quickly. Some papers study how to dynamically optimize the configuration for video analytics when the video stream context changes. JCAB [11] jointly optimizes configuration adaption, and bandwidth allocation to address several critical challenges in edge-based video analytics systems, including edge capacity limitation, unknown network variation, intrusive dynamics of video contexts. Chameleon [10] leverages temporal and spatial correlation to amortizes the cost of profiling over time and across multiple cameras and exploit the knob independence to reduce the search space from exponential to linear. Notice these reduce search space algorithm still try using various configurations to find the best configuration. Using different configurations would lead to extra expensive profiling costs. To significantly reduce the cost of profiling, we propose an automatic and adaptive configuration algorithm. [19] leverages tabular Q-learning to adapt configuration, but their agent's states only consider last latency. Our framework pays attention to the complex and changeable video stream contexts and picks the best configuration according to the current video context.

Our solution, called AdaConfigure, leverages a reinforcement learning-agent to automatically and adaptively choose the best configuration periodically, effectively improving the analytic accuracy while providing low-resource consumption. Furthermore, the extra cost of profiling is low since the agent's choosing time is hugely lower.

3. DETAILED DESIGN

Figure 2 summarizes how RL can be applied to the adaptive configuration. Briefly, it is a reinforcement learning-based system to train an agent to choose a proper configuration c for one video chunk to inference. We discuss the formulation, agent design, reinforcement learning-based framework, reward feedback in the following subsections. We provide experimental details of all the hyperparameters in Section 4.

3.1. Problem Formulation

To adaptively choose different configuration for video stream, we divides the video into T-second intervals as video chunks, and profiles configurations for each video chunk. Without loss of generality, we denote the object detection service as $\vec{y}_i = M(x_i)$ that provides a predicted result list $\vec{y}_i$ for each input video chunk x_i. It has a baseline output $\vec{y}_{\mathrm{ref}} = M(x_{\mathrm{ref}})$ for each input video chunk $x \in X_{\mathrm{ref}}$ using *reference configuration* (the most expensive configuration). We use this $\vec{y}_{\mathrm{ref}}$ as the ground truth label. For each video chunk x_c that uses a configuration c, the output $\vec{y}_c = M(x_c)$. Therefore, we have an accuracy metric $\mathcal{A}_c$ by comparing $\vec{y}_{\mathrm{ref}}$ and $\vec{y}_c$. In general, we use the F1 score as the accuracy $\mathcal{A}$, which is the harmonic mean of precision and recall, consistent with prior work [10, 20, 21]. Besides, to compute the accuracy of a frame that was not sampled by c, we use the location of objects from the previous sampled frame.

For the cost of the object detection service, we use average GPU processing time per frame as the metric of resource consumption. We also denote the metric of resource consumption as $\hat{s}_{ic}$ that for an input video chunk x_i and a given configuration c. For a reference configuration c_{ref}, the reference resource consumption is $\hat{s}_{\mathrm{ref}}$.

Initially, the agent tries different configurations c to obtain inference results image x_c from input video chunk x. To obtain object detection results $\{\vec{y}_{\mathrm{ref}}, \vec{y}_c\}$, the agent uses the choosen configuration c and the reference configuration x_{ref}. Comparing the two object detection results $\{\vec{y}_{\mathrm{ref}}, \vec{y}_c\}$ and two resource consumptions $\{\hat{s}_{\mathrm{ref}}, \hat{s}_c\}$, the agent computes the resource consumption ratio $\Delta s = \frac{\hat{s}_c}{\hat{s}_{\mathrm{ref}}}$ and accuracy metric $\mathcal{A}_c$.

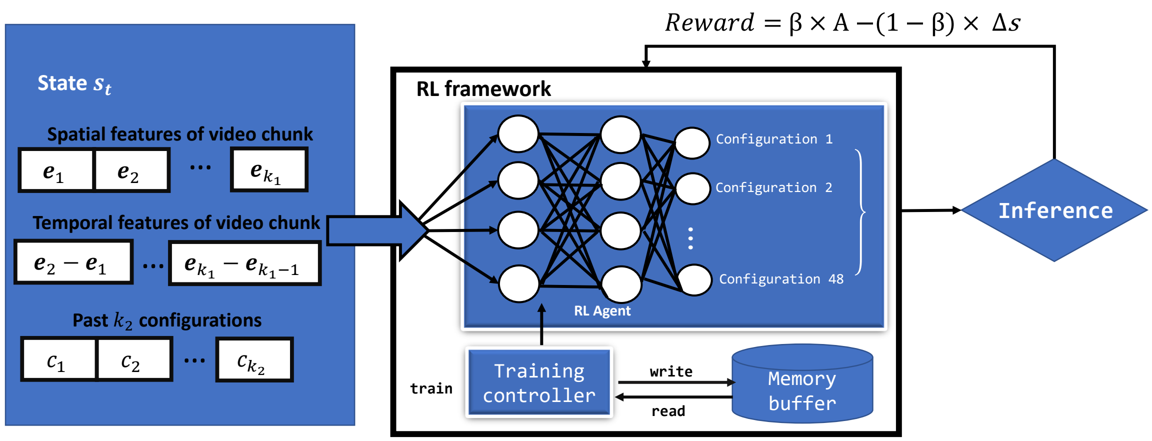


Fig. 2. Applying reinforcement learning to adaptive configuration

3.2. RL Agent Design

The RL agent is expected to give a proper configuration c for minimizing the resource consumption $\hat{s}_c$ while keeping the accuracy $\mathcal{A}$. For the RL agent, the input features are continuous numerical vectors, and the expected output is discrete compression quality level c. Therefore we can use the Deep Q-learning Network [12] as the RL agent. But the naive Deep Q-learning Network can not work well in this task because the state space of reinforcement learning is too large if we directly treat video chunk as the input, making the RL agent extremely difficult to converge.

To address this challenges, we leverage FFmpeg [22] to extract the top k_1 representative images from each chunk as spatial features of video chunk. We use a pre-trained small neural network to extract the structural information embeddings $\{e_1, e_2, ..., e_{k_1}\}$ of the images to reduce the input dimension and accelerate the training procedure. This is a commonly used strategy in training a deep neural network [23, 24]. In this work, we use the early convolution layers of MobileNetV2 [7] as the image feature extractor $\mathcal{E}(\cdot)$ for its efficiency in image classification. To extract the temporal features of video chunk, we obtain $\{\hat{e}_1, \hat{e}_2, ..., \hat{e}_{k_1-1}\}$ by each embedding subtracting previous embedding. Besides, we record the last k_2 configurations $\{c_1, c_2, ..., c_{k_2}\}$. To Solve that vectors of different lengths are not conducive to input to the neural network, we use fully connected layer to transform the spatial embedding $\{e_1, e_2, ..., e_{k_1}\}$, temporal embedding $\{\hat{e}_1, \hat{e}_2, ..., \hat{e}_{k_1-1}\}$, and recent configuration $\{c_1, c_2, ..., c_{k_2}\}$ to the fixed length vector, similar to the work [25]. We formulate the fixed length vector s as *states* and the configuration c as discrete *actions*.

3.3. Reinforcement Learning-based Framework

In our system, we define 48 discrete actions to indicate 48 configuration, and the specific configurations are provided in Section 4.1. We denote the *action-value function* as $Q(s, c; \theta)$ and the optimal compression quality level at time t as $c_t = \text{argmax}_c Q(s, c; \theta)$ where θ indicates the parameters of the Deep Q-learning Network ϕ. In such reinforcement learning formulation, the training phase is to minimize the loss function $L_i(\theta_i) = \mathbb{E}_{s,c\sim\rho(\cdot)}\left[\left(y_i - Q(s, c; \theta_i)\right)^2\right]$ that changes at each iteration i where target $y_i = \mathbb{E}_{s'\sim\{\mathcal{X},M\}}\big[r + \gamma \max_{c'} Q(s', c'; \theta_{i-1}) \mid s, c\big]$. Especially, r is the reward feedback, and $\rho(s, c)$ is a probability distribution over state s and the configuration c [12]. When minimizing the distance of *action-value function*'s output $Q(\cdot)$ and target y_i, the *action-value function* $Q(\cdot)$ outputs a more accurate estimation of an action.

In the training phase, the RL agent firstly using an $\epsilon-$greedy method to take some random trials to observe the environment's reaction and decreases the randomness when training afterward. In iteration t, we input state s_t to neural network ϕ. The RL agent ϕ generates a specific configuration c_t. The framework processes the video chunk x_t using configuration c_t to inference object detection services and obtains reward r_t. Then the framework obtains the next video chunk x_{t+1} and generates the next state s_{t+1}. The framework stores transition (s_t, c_t, r_t, s_{t+1}) in a memory buffer $\mathcal{D}$. Especially, the transition (s_t, c_t, r_t, s_{t+1}) is uesd to compute the loss function. All transitions are saved into a memory buffer $\mathcal{D}$, and the agent learns to optimize its *action* by minimizing the loss function L on a mini-batch from $\mathcal{D}$. The training procedure would converge when the agent's randomness keeps decaying. Finally, the agent's action is based on its historical "optimal" experiences. The training procedure is presented in Algorithm 1.

3.4. Reward Feedback Design

In our solution, the agent is trained by the reward feedback. In the above formulation, we define resource consumption

Algorithm 1 Training RL agent ϕ

Initialize action-value function Q with random weights θ, replay memory buffer $\mathcal{D}$ and state s_1
for $t \in 1, 2, \cdots, K$ **do**
  1) Exploration
  With probability ϵ:
    $c_t \leftarrow$ a random valid value
  Otherwise:
    $c_t \leftarrow \mathrm{argmax}_c Q(s_t, c; \theta)$
  2) Reward calcuation
  Process video chunk x_t using configuration c_t to inference
  Obtain $(\vec{y}_{\mathrm{ref}}, \vec{y}_c)$ from the object detection service
  Compute reward $r \leftarrow R(\Delta s, \mathcal{A}_c)$ according to 3.4 Reward Feedback Design
  3) Gradient descent
  Obtain next video chunk x_{t+1}
  Generate next state s_{t+1}
  $\mathcal{D} \leftarrow \mathcal{D} \bigcup \{(s_t, c_t, r_t, s_{t+1}\}$
  Sample a randomly mini-batch of transitions (s_j, c_j, r_j, s_{j+1}) from memory buffer $\mathcal{D}$
  $y_j \leftarrow r_j + \gamma \max_{c'} Q(s_{j+1}, c'; \theta)$
  Perform a gradient descent step on $\big(y_j - Q(s_j, c_j; \theta)\big)^2$ according to [12]
end for

rate $\Delta s = \frac{\hat{s}_c}{\hat{s}_{\mathrm{ref}}}$ and accuracy metric $\mathcal{A}_c$ at configuration c. Basically, we want the agent to choose a proper configuration for minimizing the resource consumption while remaining acceptable accuracy. Therefore the overall reward r should be positively correlated with the accuracy $\mathcal{A}$ while negatively with the resource consumption ratio Δs. We introduce a balance factor β to form a linear combination $r = \beta\mathcal{A} - (1-\beta)\Delta s$ as the *reward function* $R(\Delta s, \mathcal{A})$.

4. EVALUATION

4.1. Experiment Setup and Configuration Selection

We carry out real-world experiments on the object detection task (basic computer vision services) to verify our solution's performance. We use a desktop PC (Intel Xeon E5-2650 v4 CPU) with four NVIDIA 1080ti graphic card as the server infrastructure, and simulate the environment [1] with three pretrained object detection models in Tensorflow, which are SSD+ResNet152V1, FasterRCNN+ResNet50V1, FasterRCNN+InceptionResNetV2. For each model, two kinds of image resolution, which is 1024×1024 and 640×640, can be picked up. In our experiment, we set the choices space of fps as $\{1, 2, 5, 10, 15, 20, 25, 30\}$. We use FFmpeg [**?**] to switch the frame rate and image resolution, which decide which frames and in what size they should be fed to the object detection model. The configuration space comprises all possible combinations of values of these knobs, so the environment has 48 configurations in total. In the service inference, the processing video module uses CPU resources, and the object detection module uses GPU resources. In our study, we use GPU processing time as the metric of resource consumption because GPU is the dominant resource for most video processing workloads [10].In the evaluation, the processing video module needs 62s to decode 5 mins video into 9000 frames image (i.e., average CPU processing time per frame is about 0.006s), which is lower than the GPU processing time. To reduce the latency of processing video, we use different processes to carry out the processing video module and the object detection module in our experiments.

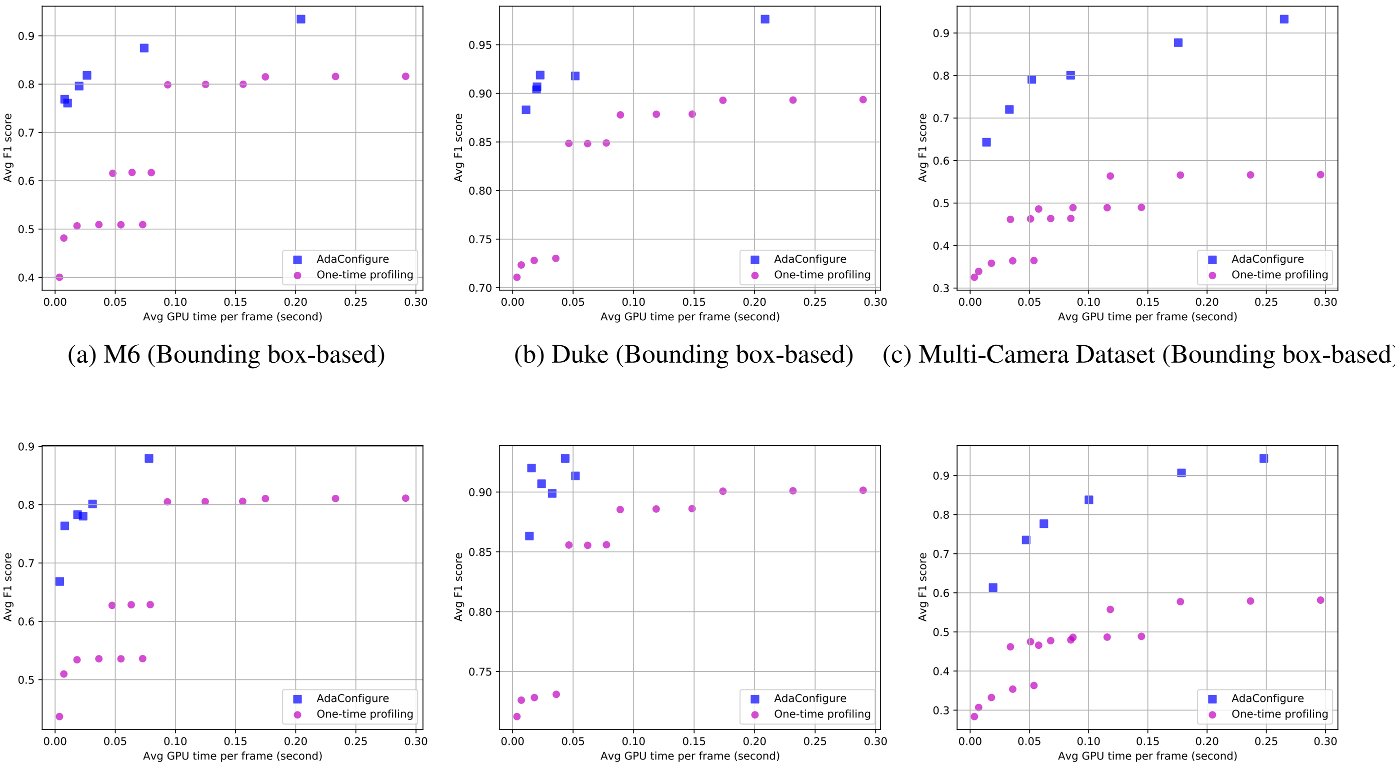
4.2. Dataset and Metric

Most public sources datasets cannot fully satisfy all configuration choices, for example, some can only guarantee 30 frames but cannot guarantee 1024p resolution. In contrast, others can ensure resolution but cannot provide a sufficient frame rate. Still, in recent years, most of the driving recorder can provide enough resolution and frame rate, so we try to search keywords (e.g., "highway traffic") on Youtube and download videos that meet the resolution and frame rate requirements. We selected three datasets: M6, Duke, and Multi-Camera Dataset. M6 was taken from a traffic camera on the longest motorway in Britain. Duke is a video from a fixed camera placed at an intersection, and the traffic flow in the video increases or decreases periodically as the traffic light change. Since the first two datasets were sourced from a fixed camera, to ensure that AdaConfigure performs better in multi-camera inference, we combine three videos collected from different locations into one video dataset, Multi-Camera Dataset. The exact content of this dataset varies significantly over time and across space.

For metric, we use the F1 score as the accuracy and average GPU processing time per frame as the resource consumption, described in Section 3.1. F1 score is the harmonic mean of precision and recall, where the precision is true positives divided by the detected objects, and the recall is true positives divided by the ground truth objects. We identify true positives using two conditions, an abounding box-based condition (only check the classified label) and a label-based condition (check the classified label and spatial overlap [26]), consistent with prior work [10, 27]. Both accuracy metrics are useful in real video analytics services and used in our experiments.

4.3. Experiment Parameters

We apply a deep Q-learning network policy to train the agent. In the training procedure, we build up to eight independently identical environments for each data set to speed up the training and decrease the data's dependency. The leaning rate is set to 0.001. For each dataset, we train the agent 10 epochs



(a) M6 (Bounding box-based) (b) Duke (Bounding box-based) (c) Multi-Camera Dataset (Bounding box-based)

(d) M6 (label-based) (e) Duke (label-based) (f) Multi-Camera Dataset (label-based)

Fig. 3. AdaConfigure (blue) consistently outperforms the baseline of one-time profiling (magenta) across different metrics on different datasets. Each dot represents the results of running each solution.

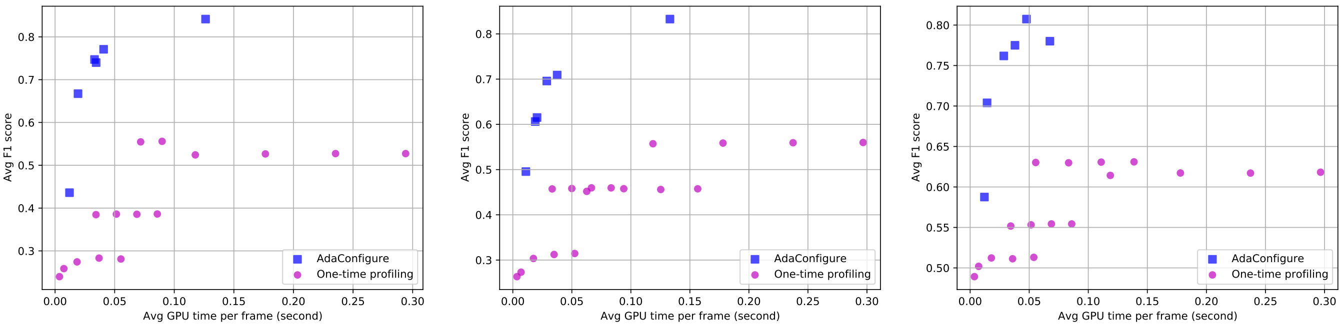
Notation	Value	Notation	Value
ϵ	0.8	k_1	3
γ	0.9	k_2	3
β	0.2,0.3,...,0.7	$T(train)$	1
$t(inference)$	1	$T(inference)$	4

Table 1. Experiment parameter

and 1000 steps per epoch. Some important hyperparameters in our experiments are given in Table 1.

4.4. Experiment Results

We use bounding box-based and label-based metric to evaluate AdaConfigure on M6, Duke, and Multi-Camera Dataset video streams. Figure 3 shows that AdaConfigure consistently outperforms the baseline of static configuration (profiling configurations once at the beginning of a video stream) along with resource consumption and two accuracy metrics on different datasets. Each magenta dot represents one static configuration solution (one-time profiling). These static configuration solutions include some expensive configurations, such as {FasterRCNN+InceptionResNetV2, 640p, 30fps}, {FasterRCNN+ResNet50V1, 1024p, 25fps} and so on, and some cheap configurations, such as {SSD+ResNet152V1, 640p, 1fps}, {SSD+ResNet152V1, 640p, 2fps}, {SSD+ResNet152V1, 640p, 5fps} and so on. Each blue dot represents one AdaConfigure configuration solution, which is dependent on the balance factor β in reward function, presented in Section 3.4. The detailed discussion about differents AdaConfigure solutions is in Section 4.4.2. Note that AdaConfigure's resource consumption includes both running the best configuration to get inference results and profiling cost of adaptive configuration, which detailed discussed in 4.4.3. As shown in Figure 3(a)(d), AdaConfigure achieves 10-30% higher accuracy with a similar amount of resources, or achieve a similar accuracy with only 60-80% of the resources on the M6 dataset. As shown in Figure 3(b)(e), AdaConfigure achieves 10-20% higher accuracy with a similar amount of resources, or achieve a similar accuracy with only 80-90% of the resources on a Duke dataset. As shown in Figure 3(c)(f), AdaConfigure achieves 25-35% higher accuracy with a similar amount of resources, or achieve higher accuracy with only 75-85% of the resources on Multi-Camera Dataset. In a word, AdaConfigure can improve 10-35% higher accuracy or save 60-90% resource consumption.



(a) Camera 1 (Bounding box-based) (b) Camera 2 (Bounding box-based) (c) Camera 3 (Bounding box-based)

Fig. 4. AdaConfigure (blue) consistently outperforms the baseline of one-time profiling (magenta) across different cameras.

4.4.1. Superior Performance on Multi-camera Situation

To compare AdaConfigure's performance on single-camera inference and multi-camera inference, we respectively train and test the agent on each single-camera dataset of Multi-Camera Dataset. Figure 4(a)(b)(c) shows that AdaConfigure achieves 10-20% higher accuracy with a similar amount of resources, or achieve a similar accuracy with only 50-60% of the resources on a single-camera inference. As shown in Figure 3(c), the gap between AdaConfigure and static configuration is larger than Figure 4(a)(b)(c), indicating that AdaConfigure has a better improvement on the multi-camera situation. AdaConfigure achieves 25-35% higher accuracy with a similar amount of resources, or reach higher accuracy with only 75-85% of the resources on multi-camera inference. AdaConfigure achieves superior performance on multi-camera inference than single-camera inference. Instead of the static solution using fixed configuration in different situations, AdaConfigure can choose a proper configuration for different situations since the exact spatial characteristics of the video contents are different in different locations. It proves that our solution can pick the proper configuration according to intrusive dynamics of video contexts, including spatial and temporal features.

4.4.2. Different AdaConfigure Solutions to Meet Different Services

In the training phase, when using different balance factors β in the reward function, we would obtain different agents (different adaptive configuration strategies). Each exact β solution represents a concrete adaptive configuration strategy, which can adaptively update configuration across time. In general, the β is bigger, indicating the accuracy is more important relatively, and the agent would choose a more expensive configuration. In our experiment, we set β is 0.2,0.3,...,0.7, and the results of different AdaConfigure solutions on Multi-Camera Dataset are listed in Table 2.

balance factor β	Avg GPU time per frame (second)	Avg F1 score
0.2	0.01384	0.64307
0.3	0.03314	0.72008
0.4	0.05203	0.79096
0.5	0.08479	0.80056
0.6	0.17570	0.87731
0.7	0.26511	0.93262

Table 2. Resource consumption and F1 accuracy for different AdaConfigure solutions

Table 2 shows that when the β increases, the resource consumption and the accuracy of the corresponding solution would increase, indicating the AdaConfigure solutions of big β would choose the more expensive configuration to inference. We can leverage this to train proper configuration strategy for different service demands, for example, using big β for high-accuracy demand services and using small β for low-accuracy demand services.

4.4.3. Low Profiling Cost of Adaptive Configuration

Comparing to the static solution that profiles configurations once, in our solution, the video chunk is passed to the AdaConfigure firstly to estimate the configuration. Running this RL agent brings profile cost (extra resource consumption) to the whole system. In this section, we evaluate this profile cost. In the inference phase, we divide the video into T-second intervals as video chunks and use AdaConfigure to choose the proper configuration for the first t seconds of the video chunk. It then sticks with the chosen configuration for the rest of the video chunk, i.e., for $T-t$ seconds. We use T = 4 and t = 1 for our experiments. The profile cost, including the resource of extract k_1 embeddings and the cost of agent choosing actions. We test the average time on 30 hours video of Multi-Camera Dataset, and conclude that the average time of extract one embedding is 0.02s, and the average time of agent choosing action is 0.0006s, which can be ignored. We compute the ratio by dividing profile time into total inference time, which is equals the frame number multiply the average time per frame.

The profile cost is about 0.2-2% of the overall video analytics resource consumption. The concrete ratio depends on the concrete AdaConfigure configuration solutions, such as the solutions listed in Table 2. For instance, when using the β 0.7 solution, the profile time is only 0.2% of overall resource consumption on this solution; when using the β 0.2 solution, the profile cost is 1.5%.

5. CONCLUSION

This paper proposes a reinforcement learning (RL)-based adaptive video analytics configuration framework, AdaConfigure. Our solution can adapt the best configuration to intrusive dynamics of video contexts, meaning that it can adaptively choose the proper configuration for the current video chunk according to the spatial and temporal features of video contexts. Therefore, our solution can adaptively update configuration strategy over time, and it adaptive to different location-camera inferences and various accuracy-demand services. In the evaluation, AdaConfigure achieves 10-35% higher accuracy with a similar amount of resources or achieves similar accuracy with only 60-90% of the resources. AdaConfigure proves to be more efficient than static solutions and only creates an overhead of 0.2-2% to the overall video analytics services.

6. REFERENCES

[1] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al., "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7310–7311.

[2] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[3] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in neural information processing systems*, 2016, pp. 379–387.

[4] Christian Szegedy, Scott Reed, Dumitru Erhan, Dragomir Anguelov, and Sergey Ioffe, "Scalable, high-quality object detection," *arXiv preprint arXiv:1412.1441*, 2014.

[5] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[6] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[7] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[9] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," *arXiv preprint arXiv:1602.07261*, 2016.

[10] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica, "Chameleon: scalable adaptation of video analytics," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 253–266.

[11] Can Wang, Sheng Zhang, Yu Chen, Zhuzhong Qian, Jie Wu, and Mingjun Xiao, "Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics," in *Proc. IEEE INFOCOM*, 2020, pp. 1–10.

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[13] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J Freedman, "Live video analytics at scale with approximation and delay-tolerance," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 377–392.

[14] Chien-Chun Hung, Ganesh Ananthanarayanan, Peter Bodik, Leana Golubchik, Minlan Yu, Paramvir Bahl, and Matthai Philipose, "Videoedge: Processing camera streams using hierarchical clusters," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2018, pp. 115–131.

[15] Seungyeop Han, Haichen Shen, Matthai Philipose, Sharad Agarwal, Alec Wolman, and Arvind Krishnamurthy, "Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, 2016, pp. 123–136.

[16] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B Gibbons, and Onur Mutlu, "Focus: Querying large video datasets with low latency and low cost," in *13th {USENIX} Symposium on Operating Sys-*

tems Design and Implementation ({OSDI} 18), 2018, pp. 269–286.
[17] Francisco Romero, Qian Li, Neeraja J Yadwadkar, and Christos Kozyrakis, "Infaas: A model-less inference serving system," *arXiv preprint arXiv:1905.13348*, 2019.
[18] Faheem Ullah and M Ali Babar, "Quickadapt: Scalable adaptation for big data cyber security analytics," in *2019 24th International Conference on Engineering of Complex Computer Systems (ICECCS)*. IEEE, 2019, pp. 81–86.
[19] Mauricio Fadel Argerich, Bin Cheng, and Jonathan Fürst, "Reinforcement learning based orchestration for elastic services," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. IEEE, 2019, pp. 352–357.
[20] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia, "Noscope: optimizing neural network queries over video at scale," *arXiv preprint arXiv:1703.02529*, 2017.
[21] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J Freedman, "Live video analytics at scale with approximation and delay-tolerance," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 377–392.
[22] FFmpeg, "Ffmpeg," http://ffmpeg.org/, 2000–2018.
[23] Weifeng Ge and Yizhou Yu, "Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1086–1095.
[24] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever, "Improving language understanding by generative pre-training," *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*, 2018.
[25] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 197–210.
[26] Mark Everingham, Luc Van Gool, Christopher K-I Williams, John Winn, and Andrew Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.
[27] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia, "Noscope: optimizing neural network queries over video at scale," *arXiv preprint arXiv:1703.02529*, 2017.