AUTOCONFIGURE: A CONTEXT-DRIVEN AUTOMATIC CONFIGURATION FRAMEWORK FOR VIDEO ANALYTICS SERVICES

Zhaoliang He[1], *Yuan Wang*[3], *Chen Tang*[1], *Zhi Wang*[2], *Wenwu Zhu*[1]

[1]Department of Computer Science and Technology, Tsinghua University, China
[2]Tsinghua Shenzhen International Graduate School, Tsinghua University, China
[3]Tsinghua-Berkeley Shenzhen Institute, Tsinghua University, China

ABSTRACT

Deep convolutional neural networks (NN)-based video analytics services demand intensive computation resources and high inference accuracy. Due to the highly variable video context, the *best* configuration(the most proper configuration) also varies over time. If only choose a static configuration once(i.e., only profiles the video stream to choose the best configuration *once*), the services would either waste resources (by picking an expensive configuration) or sacrifice accuracy (by selecting a cheap configuration). Conversely, searching for all configurations in a enormous space can lead to excessive resource overhead that far exceeds the benefits of periodic configurations. Knowing this, designing an *automatic* approach to choose the best configuration for the current video context is meaningful, we propose a reinforcement learning (RL)-based automatic video analytics configuration framework, AutoConfigure. The unique feature of AutoConfigure is *context-driven*, meaning that AutoConfigure can adapt the best configuration to intrusive dynamics of video contexts. In particular, our solution can choose the best configuration for the current video chunk according to the spatial and temporal features of video contexts. We implement and evaluate this approach in the object detection task, comparing its performance to static configuration. We show that AutoConfigure 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 0.1-1% to the overall video analytics services.

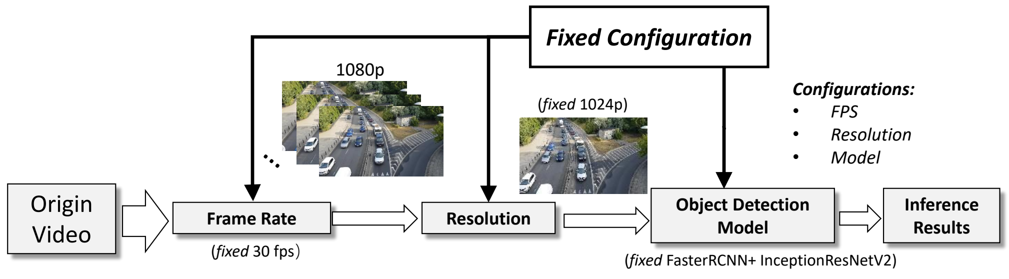
1. INTRODUCTION

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].

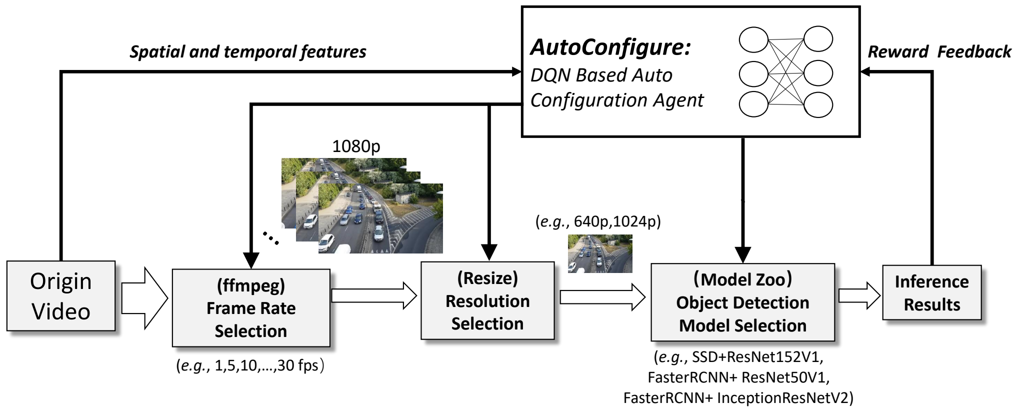
A video analytics application consists of a *pipeline* of several video processing modules, typically including a decoder, a selective sampling frame application, and a target detector. Such a pipeline always has multiple *knobs*, such as frame rate, resolution, and model (e.g., SSD+{MobileNet [7], ResNet [8]}, FasterRCNN+{ResNet,InceptionResNet [9]}). A combination of the knob values is video analytics *configuration*. The configuration space grows *exponentially* with the number of knobs, and their values [10]. Since video analytics applications demand intensive computation resources and high accuracy, we pay much attention to the consumption of resources in the calculation process and the inference accuracy. Therefore, the problem that follows is how to balance *resource consumption* and *accuracy*. Different configurations directly affect accuracy and resource consumption. Using a fixed static high configuration can be very precise, but it can also be a huge drain on resource consumption. Similarly, specifying a low configuration will result in a significant reduction in accuracy.

The *best* configuration for video analysis services often varies in minutes or even seconds [10]. Hence, we aim to find a range of "most appropriate" configurations that minimize the consumption of computing resources and is accurate to the desired threshold. On the one hand, like figure 1(a), if just use a unified static configuration solution (i.e.only profiles the processing pipeline to choose the best configuration *once*), the application would either waste resources (by picking an expensive configuration) or sacrifice accuracy (by selecting a cheap configuration). On the other hand, if one periodically profiles the processing pipeline to find an optimal resource-accuracy *tradeoff* by exhaustive all configurations, it would be prohibitively expensive since the configuration space is extremely large, and thousands of configurations can be combined with just a few knobs. To tackle these challenges, we propose an automatic configuration approach, AutoConfigure, and the brief framework is shown in Figure 1(b).

For a video analytics application, choosing the "most appropriate" configuration is a complicated decision-making problem that is challenging to solve by rules. An automatic approach is needed to learn from video contexts to decide



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


Spatial and temporal features
AutoConfigure:
DQN Based Auto Configuration Agent
Reward Feedback
1080p
(e.g., 640p,1024p)
Origin Video
(ffmpeg) Frame Rate Selection
(e.g., 1,5,10,...,30 fps)
(Resize) Resolution Selection
(Model Zoo) Object Detection Model Selection
(e.g., SSD+ResNet152V1, FasterRCNN+ ResNet50V1, FasterRCNN+ InceptionResNetV2)
Inference Results

(b) AutoConfigure solution: context-driven automatic configuration framework

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

the best configuration for the current video context. The reinforcement learning method is an excellent way to solve this unsupervised complex-environmental problem. In our solution, we tackle the following design challenges.

- *The best configuration for video analytics services changes over time with the video content.* Video content varies with the real-world environment, so the best configuration is changing; for instance, tracking vehicles when traffic moves quickly require a much higher frame rate than when traffic moves slowly, but when each condition occurs may vary by hour, minute, even second. As a dynamic video analytics configuration application, we target to provide a solution that dynamically picks a configuration according to intrusive dynamics of video contexts, i.e., it can *generate* video analytics configuration for video analytics in a different time.

- *How to significantly reduce the resource cost of periodic configuration profiling.* Because of the large configuration space, the cost of periodically profiling often exceeds any resource savings gained by adapting the best configurations we end up selecting. We leverage a Reinforcement Learning-based agent to automatically pick the best configuration periodically, dramatically reducing the profiling cost.

- *Lack of well-labeled training data.* In our case, there was no well-marked data that indicated which configuration should be choosed at which time of the video, as is the case with traditional deep learning tasks.In practice, such a video analytics configuration is usually utilized online, and the solution has to automatically learn from the video contexts.

To address the above challenges, we present a context-driven automatic configuration framework based on reinforcement learning, called AutoConfigure, which can dynamically select the "most appropriate" configuration according to intrusive dynamics of video contexts, thus solving this difficult optimal configuration decision problem in a very low-cost way. The main contributions of this paper are summarized as follows.

- We design an interactive training environment that can be applied to different video analytics applications. We propose a Deep Q-learning Network-based [11] agent to pick the best video analytics configuration. 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.

- We build a reinforcement learning-based framework to train the agent in the above environment. The agent can learn to choose a "most appropriate" configuration for each timestamp of video analytics after iteratively interacting with the environment by feeding the carefully designed reward that considers both accuracy and resources.

- AutoConfigure achieves 10-35% higher accuracy with a similar amount of resources or achieves similar accuracy with only 60-90% of the resources. AutoConfigure proves to be more efficient than static solutions and only creates an overhead of 0.1-1% 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 [12] profiles thousands of video analytics queries on live video streams over large clusters, achieving resource-quality tradeoff with multi-dimensional configurations. VideoEdge [13] 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 [14] provides a heuristic scheduling algorithm to adaptively select model variants of different accuracy for deep stream processing under resource constraints. Focus [15] 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 [16] 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. QuickAdapt [17] 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 [18] 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 configuration would lead to extra expensive profiling costs. To significantly reduce the cost of profiling, we propose an automatic 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 AutoConfigure, leverages a reinforcement learning-agent to automatically 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 automatic 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 automaticly 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 $\vec{y}_{\text{ref}} = M(x_{\text{ref}})$ for each input video chunk $x \in X_{\text{ref}}$ using *reference configuration* (the most expensive configuration). We use this $\vec{y}_{\text{ref}}$ 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 $\vec{y}_{\text{ref}}$

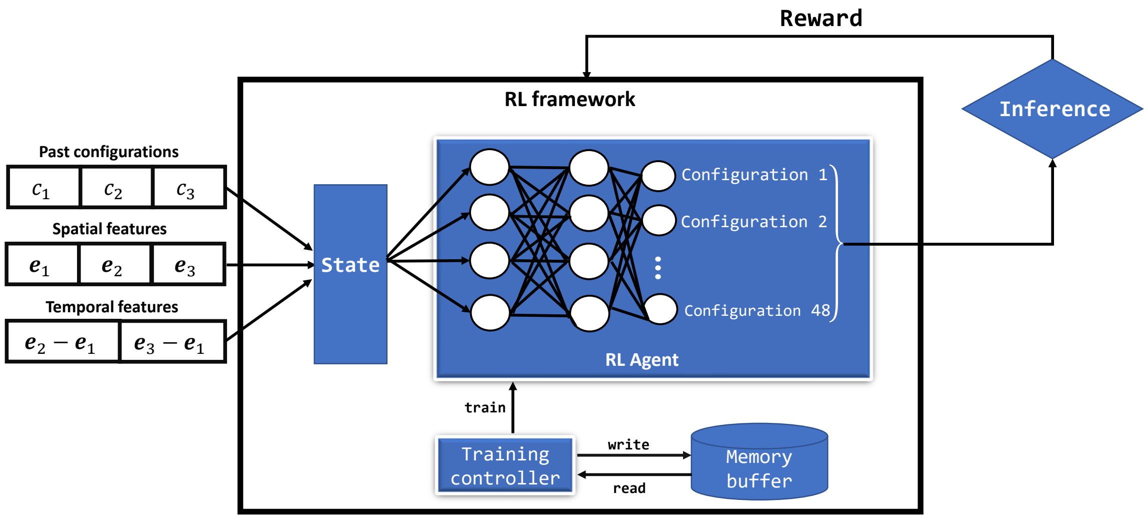


Fig. 2. Applying reinforcement learning to automatic configuration

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 because GPU is the dominant resource for the majority of video processing workloads. 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 $\hat{s}_{\text{ref}}$.

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

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 [11] 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. To preserve enough details, we have to add many layers and nodes to the neural network, making the RL agent extremely difficult to converge.

To address this challenges, we 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 [22, 23]. 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 [24]. 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 4.2. We denote the *action-value function* as $Q(s, c; \theta)$ and the optimal compression quality level at time t as $c_t = \operatorname{argmax}_c Q(s, c; \theta)$ 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 $L_i(\theta_i) = \mathbb{E}_{s,c\sim\rho(\cdot)}\Big[\big(y_i - Q(s, c; \theta_i)\big)^2\Big]$ that changes at each iteration i where target $y_i = \mathbb{E}_{s'\sim\{\mathcal{X},M\}}\big[r +$

Algorithm 1 Training RL agent ϕ

```
1: Initialize action-value function $Q$ with random weights $\theta$,
   replay memory buffer $\mathcal{D}$ and state $s_1$
2: for $t \in 1, 2, \cdots, K$ do
3:     1) Exploration
4:     With probability $\epsilon$:
5:         $c_t \leftarrow$ a random valid value
6:     Otherwise:
7:         $c_t \leftarrow \mathrm{argmax}_c Q(s_t, c; \theta)$
8:     2) Reward calcuation
9:     Process video chunk $x_t$ using configuration $c_t$ to inference
10:    Obtain $(\vec{y}_{\mathrm{ref}}, \vec{y}_c)$ from the object detection service
11:    Compute reward $r \leftarrow R(\Delta s, \mathcal{A}_c)$ according to 3.4 Reward Feedback Design
12:    3) Gradient descent
13:    Obtain next video chunk $x_{t+1}$
14:    Generate next state $s_{t+1}$
15:    $\mathcal{D} \leftarrow \mathcal{D} \bigcup \{(s_t, c_t, r_t, s_{t+1}\}$
16:    Sample a randomly mini-batch of transitions
       $(s_j, c_j, r_j, s_{j+1})$ from memory buffer $\mathcal{D}$
17:    $y_j \leftarrow r_j + \gamma \max_{c'} Q(s_{j+1}, c'; \theta)$
18:    Perform a gradient descent step on $\big(y_j - Q(s_j, c_j; \theta)\big)^2$
       according to [11]
19: end for
```

$\gamma \max_{c'} Q(s', c'; \theta_{i-1}) \mid s, c\big]$. Especially, r is the reward feedback, and $\rho(s, c)$ is a probability distribution over state s and the configuration c [11]. 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 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

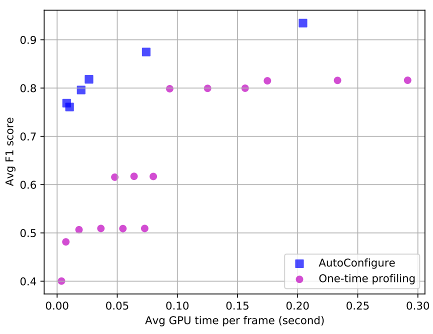
4.1. Dataset and Metric

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 AutoConfigure 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.

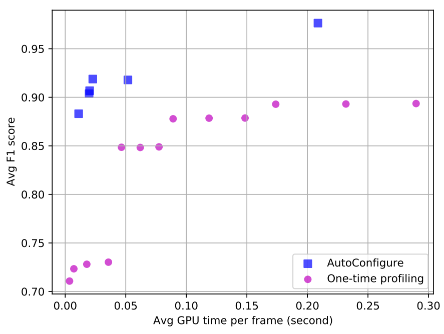
For metric, we use the F1 score as the accuracy and average GPU processing time per frame as the resource consumption, detail described in Section 3.1. F1 score is the harmonic mean of precision and recall, where the precision is true positives divided by the detected objects, and the recall is true 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 [25]), consistent with prior work [10, 26]. Both metrics are useful in real video analytics services and used in our experiments.

4.2. Configuration Selection

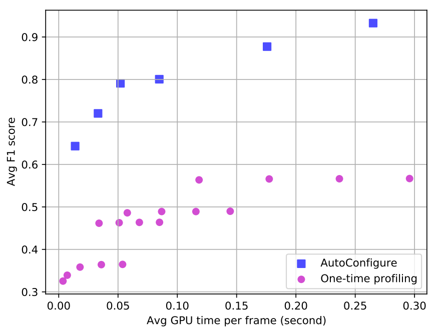
We simulate the environment [1] with three pretrained object detection models in Tensorflow, which are SSD+ResNet152V1, FasterRCNN+ResNet50V1, FasterRCNN+InceptionResNetV2. For each model, two kinds of image resolution, which is 1024×1024 and 640×640, can be picked



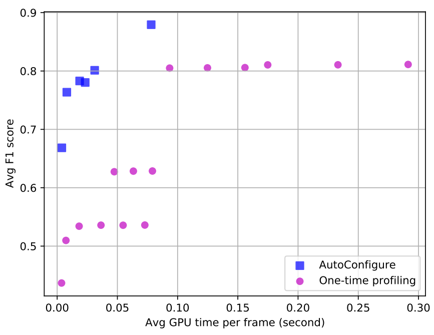
(a) M6 (Bounding box-based)



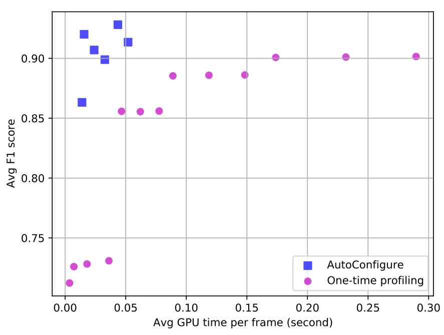
(b) Duke (Bounding box-based)



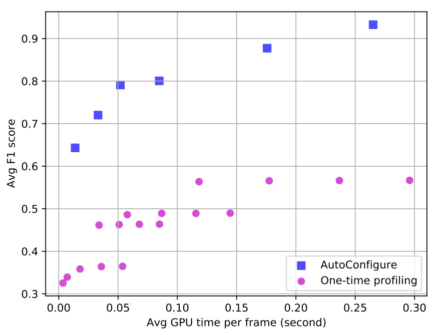
(c) Multi-Camera Dataset (Bounding box-based)



(d) M6 (label-based)



(e) Duke (label-based)



(f) Multi-Camera Dataset (label-based) running

Fig. 3. AutoConfigure (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.

up. In our experiment, we set the choices space of fps as {1, 2, 5, 10, 15, 20, 25, 30}. We FFmpeg [27] 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.

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

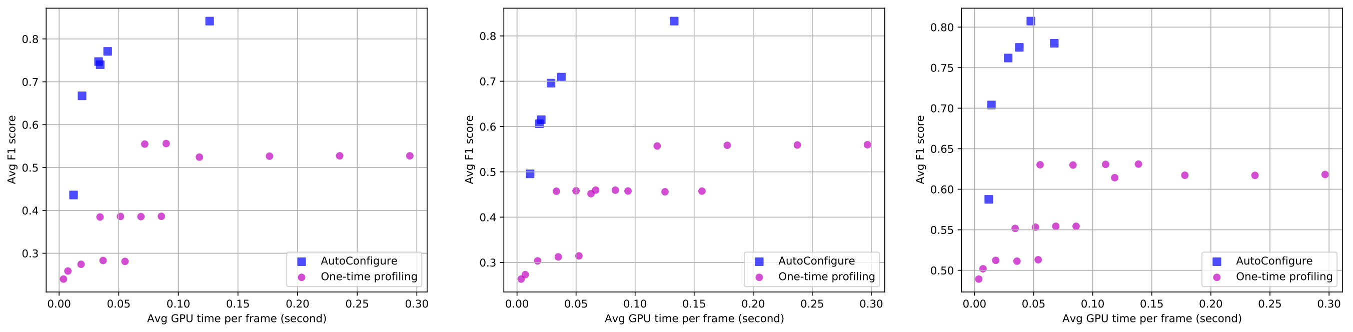
4.4. Experiment Results

We use bounding box-based and label-based metric to evaluate AutoConfigure on M6, Duke, and Multi-Camera Dataset video streams. Figure 3 shows that AutoConfigure consistently outperforms the baseline of static configuration

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

(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. These static configuration solutions including some expensive configurations, such as {FasterRCNN+InceptionResNetV2, 640p, 30fps}, {FasterRCNN+ResNet50V1, 1024p, 25fps}, and some cheap configurations, such as {SSD+ResNet152V1, 640p, 1fps}, {SSD+ResNet152V1, 640p, 2fps}, {SSD+ResNet152V1, 640p, 5fps}. Each blue dot represents one AutoConfigure configuration solution, which is dependent on the balance factor β in reward function, presented in Section 3.4. The detailed discussion about differents AutoConfigure solutions is in Section 4.4.1. As shown in Figure 3(a)(d), AutoConfigure achieves 10-30% higher accuracy with a similar amount of



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

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

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 AutoConfigure solutions

resources, or achieve a similar accuracy with only 60-80% of the resources on the M6 dataset. As shown in Figure 3(b)(e), AutoConfigure 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), AutoConfigure 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, AutoConfigure can improve 10-35% higher accuracy or save 60-90% resource consumption.

4.4.1. Different AutoConfigure Solutions

In the training phase, when using different balance factors β in the reward function, we would obtain different agents (different automatic configuration strategies). 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 AutoConfigure solutions on Multi-Camera Dataset are listed in Table 2.

Table 2 shows that when the β increases, the resource consumption and the accuracy of the corresponding solution would increase, indicating the AutoConfigure 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.

4.4.2. Superior Performance on Multi-camera Situation

To compare AutoConfigure'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 AutoConfigure 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 AutoConfigure and static configuration is larger than Figure 4(a)(b)(c), indicating that AutoConfigure has a better improvement on the multi-camera situation. AutoConfigure 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. AutoConfigure achieves superior performance on multi-camera inference than single-camera inference. Instead of the static solution using fixed configuration on different situations, AutoConfigure can choose a proper configuration for different situations since the exact contents of the video 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.

4.4.3. Profile Cost

Comparing to the static solution that profiles configurations once, in our solution, the video chunk is passed to the AutoConfigure 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 AutoConfigure 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. In our test, 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 time is about 0.1-1% of the overall video analytics resource consumption. The concrete ratio depends on the concrete AutoConfigure configuration solutions, such as the solutions listed in Table 2.

5. CONCLUSION

This paper proposes a reinforcement learning (RL)-based automatic video analytics configuration framework, AutoConfigure. Our solution can adapt the best configuration to intrusive dynamics of video contexts, meaning that it can periodically choose the proper configuration for the current video chunk according to the spatial and temporal correlation of video contexts. In the evaluation, AutoConfigure achieves 10-35% higher accuracy with a similar amount of resources or achieves similar accuracy with only 60-90% of the resources. AutoConfigure proves to be more efficient than static solutions and only creates an overhead of 0.1-1% 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] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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 Systems Design and Implementation ({OSDI} 18)*, 2018, pp. 269–286.

[16] Francisco Romero, Qian Li, Neeraja J Yadwadkar, and

Christos Kozyrakis, "Infaas: A model-less inference serving system," *arXiv preprint arXiv:1905.13348*, 2019.

[17] 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.

[18] 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.

[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] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] FFmpeg, "Ffmpeg," `http://ffmpeg.org/`, 2000–2018.