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 demands intensive computation resources and high inference accuracy. Due to the highly variable video context, the *best* configuration (the most fitting configuration) for a video analytics services also varies over time. Searching a large space of configurations periodically causes an overwhelming resource overhead that far outstrips the gains of periodically profiling. Knowing this, designing an *automatic* approach to decide what is the best configuration for the current video context is meaningful. In this paper, 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 current video chunk according to the spatial and temporal correlation of video contexts. We implement and evaluate this approach in object detection task comparing its performance to static configuration. We show that AutoConfigure achieves 20-30% higher accuracy with the same amount of resources, or achieve the same accuracy with only 50-70% of the resources. Furthermore, AutoConfigure proves to be more efficient than existing baselines by creating an overhead of less than xx%(To be tested) to the overall video analytics services.

1. INTRODUCTION

this section is too long

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 [27], such as FasterRCNN [1], RFCN [2], Multibox [3], SSD [4] and YOLO [5].

For the collected video, the classical computer vision and deep neural network technology are generally used for video analytics. 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 [25], ResNet [6]}, FasterRCNN+{ResNet,InceptionResNet [7]}). A combination of the knob values is a video analytics *configuration*. The configuration space grows *exponentially* with the number of knobs and their values [8]. 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 accuracy of inference. Therefore, the problem that follows is how to balance *resource consumption* and *accuracy*.

Apparently, choosing different configurations will affect resource consumption and accuracy. When at a fixed frame rate, using a complex model with high resolution can obviously and accurately detect the target object, but it also requires more computing resources. Similarly, when using the same model with the same resolution, choosing a lower resolution can reduce the computing resources, and the subsequent cost is the decline of accuracy. Obviously, when the target car (object) is large, a smaller model and a lower resolution can meet the accuracy with a large reduction of resource consumption, while the target car (object) is small, a more complex model and higher resolution are needed to achieve a satisfactory accuracy. And in the case of a highway video analytics, due to the rate of car travel cannot be predicted in advance, so when the car drives slowly (or static) because of the traffic jam, we can choose a lower frame rate (such as 1 FPS) instead of having to use a constant high frame rate throughout whole video. This can significantly reduce resource consumption, but does not affect the accuracy of the video analytics.

The *best* configuration for a video analytics services also varies over time, often at a timescale of minutes or even seconds [8]. Hence, our goal is to find a range of "most appropriate" configurations that takes up the minimum amount of computing resources and is accurate to the desired threshold. On the one hand, if one only profiles the processing pipeline to choose the best configuration *once*, the application would

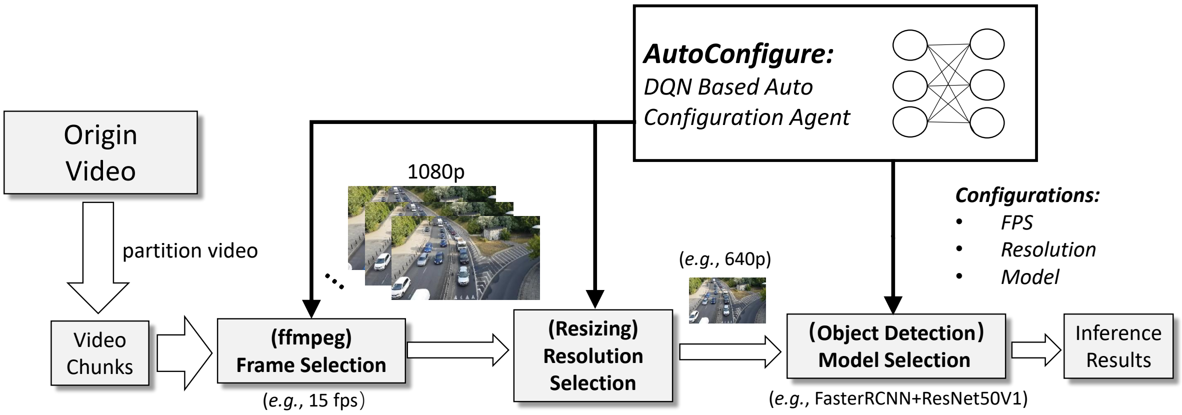


Fig. 1. Framework of AutoConfigure architecture

either waste resources (by picking an expensive configuration) or sacrifice accuracy (by picking 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.

For a video analytics application, choosing the "most appropriate" configuration is a complicated decision-making problem, which is challenging to be solved by rules. An automatic approach is needed, one that is able to learn from video contexts, to decide what is 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 a video analytics services changes over time with the environment.* The real-world environment is non-stationary; for instance, tracking vehicles when traffic moves quickly requires a much higher frame rate than when traffic moves slowly, but when each condition occurs may vary by hour, minute, or second. As a dynamic video analytics configuration solution, 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.* 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 problem, one is not provided the well-labeled data on which configuration should be used in which time of the video, as in conventional supervised deep learning tasks. In practice, such a video analytics configuration is usually utilized in an online manner, and the solution has to learn from the video contexts automatically.

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. A brief framework of AutoConfigure is shown in Figure 1. some description? 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 an Deep Q-learning Network-based [22] agent to pick the best configuration for video analytics. Also, we define the elements of the RL-based approach, such as actions, states and rewards, to adapt the configuration to 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 can achieve 20-30% higher accuracy with the same amount of resources, or achieve the same accuracy with only 50-70% of the resources. Furthermore, AutoConfigure proves to be more efficient than existing baselines by creating an overhead of less than xx%(To be tested) to the overall video analytics services.

2. RELATED WORKS

2.1. Static configuration optimization

Several previous papers have considered optimizing video analytics servicess by either adjusting the configuration knobs or training specialized NN models. VideoStorm [10] profiles thousands of video analytics queries on live video streams over large clusters, achieving resource-quality tradeoff with multi-dimensional configurations. VideoEdge [11] 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 [12] provides a heuristic scheduling algorithm to adaptively select model variants of different accuracy for deep stream processing under resource constraints. Focus [13] deconstructs video analytics into two phases, i.e., video ingest and video query. By tuning the share of computing resources of both phases, Focus achieves effective and flexible tradeoff of latency and accuracy of video analytics. 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 analyzing 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 papers study how to dynamically optimize the configuration for video analytics when the video stream context changes. [14] adaptively retrains the NN model to detect the set of popular objects as it changes over time in the video classification task. [15] proposes an online video quality and computing resource configuration algorithm to gradually learn the optimal configuration strategy, effectively improving the analytic accuracy while providing the low-latency response. 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 quickly adapt to the changes in security events data. 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. The online algorithm effectively balances analytics accuracy and energy consumption while keeping low system latency. [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 current video context.

The closest work to ours is Chameleon [8], which dynamically picks the best configurations for video analytics servicess, reducing resource consumption with little degradation in accuracy. They leverage 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.[8] Notice that Chameleon still has non-trivial room for improvement compared to the optimal (idealized) performance, i.e., periodic updating with zero profiling cost. Even the search space is linear, and the profiling cost is still expensive. Such a 24-hours video, Chameleon profiles the configuration space once in every profiling window (16s), it would profile 5400 times. One profiling cost grows linear in the number of configuration knobs and the number of values per knob. The total profiling cost is also significantly high, which is equal to one profiling cost multiply the number of profiling (5400). Our solution, called AutoConfigure, leverages a reinforcement learning-agent to automatically choose the best configuration periodically, significantly reducing the cost of profiling since the agent’s choosing time is extremely lower.

3. DETAILED DESIGN

Figure 2 summarizes how RL can be applied to 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 be inferenced. 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

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 $\vec{y}_{\text{ref}}$ 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 [8, 20, 21]. Besides, to compute the accuracy

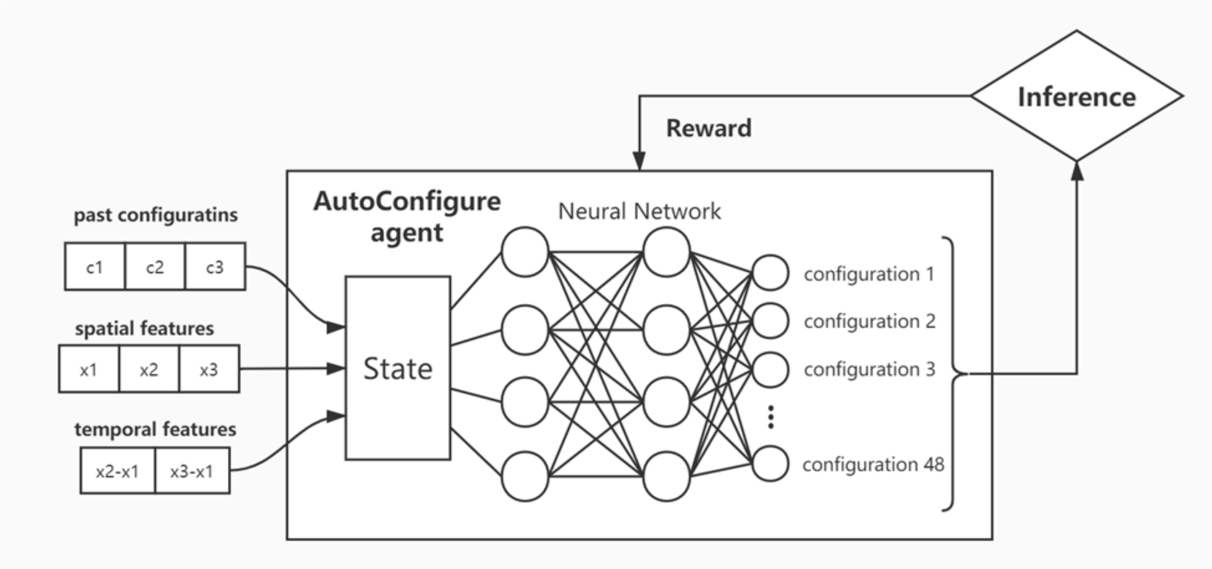


Fig. 2. Applying reinforcement learning to automatic configuration.redo

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 [22] 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 correlations 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 [25] as the image feature extractor $\mathcal{E}(\cdot)$ for its efficiency in image classification. To extract the temporal correlations 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 [26]. 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. 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 $L_i(\theta_i) = \mathbb{E}_{s,c\sim\rho(\cdot)}\Big[\big(y_i - Q(s, c; \theta_i)\big)^2\Big]$ that changes at each iteration i where target $y_i = \mathbb{E}_{s'\sim\{\mathcal{X},M\}}\big[r + \gamma \max_{c'} Q(s', c'; \theta_{i-1}) \mid s, c\big]$. Especially, r is the reward feedback, and $\rho(s, c)$ is a probability distribution over state s and the configuration c [22]. When minimizing the dis-

Algorithm 1 Training RL agent ϕ

```
Initialize action-value function $Q$ with random weights $\theta$, replay memory buffer $\mathcal{D}$ and state $s_1$
for $t \in 1, 2, \cdots, K$ do
    1) Exploration
    With probability $\epsilon$:
        $c_t \leftarrow$ a random valid value
    Otherwise:
        $c_t \leftarrow \text{argmax}_c Q(s_t, c; \theta)$
    2) Reward calcuation
    Process video chunk $x_t$ using configuration $c_t$ to inference
    Obtain $(\vec{y}_{\text{ref}}, \vec{y}_c)$ 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 [22]
end for
```

tance 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 $\Delta s = \frac{\hat{s}_c}{\hat{s}_{\text{ref}}}$ 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 linear 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. EXPERIMENT

4.1. Datasets

this section is too long

Due to most of public sources datasets cannot fully satisfy all configuration choices, for example, some can only guarantee 30 frames but cannot guarantee 1024p resolution, while others can guarantee resolution but cannot provide sufficient frame rate, but in recent years, most of the driving recorder can provide enough resolution and frame rate, so we try to use Youtube video as fellows. First, in order to eliminate the bias caused by Cookies and personal preferences, we use private browsing mode to search keywords (e.g., "drivecam highway hd"). Second, manually delete irrelevant videos (e.g., ads for some driving recorders) and download videos that are longer than ten minutes. In addition, these videos need to be as dynamic as the real world, such as not standing still for too long. Becaues of none of these videos contained ground truth, we used the data of the original video output from DNN as the tags to calculate the accuracy. For instance, in object detection, the accuracy is defined by the F1 score with respect to the server-side DNN output in highest resolution (original) with over 50% confidence score.Based on the above filtering strategy, we finally selected three datasets:

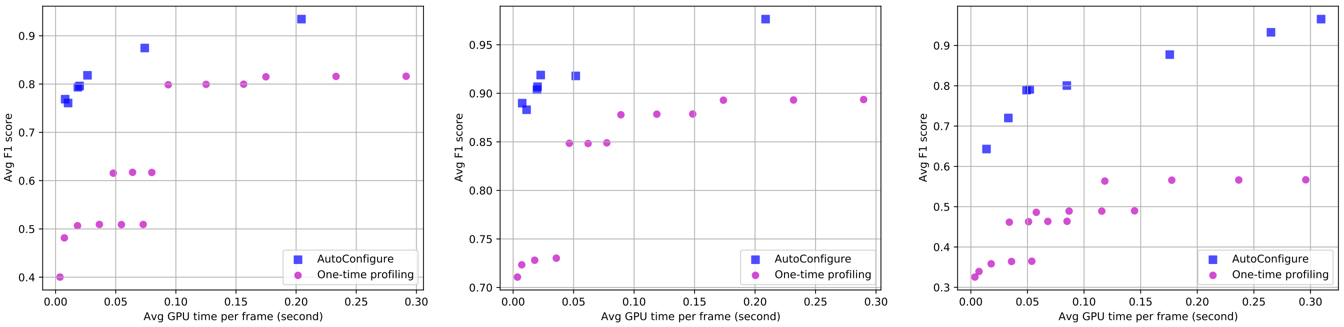
M6: The M6 is the longest motorway in Britain and the most important road from the Midlands to the West Coast. Thousands of cars travel on it every day. This video was taken from a traffic camera, and its natural dynamics allows us to experiment with this video.

Duke: Duke is a video from a fixed camera placed at an intersection. Because it is fixed in the middle of the intersection, the traffic flow in the video increases or decreases periodically as the traffic light change.

Mutli Camera Dataset: Since the first two datasets were sourced from a fixed camera, in order to ensure that AutoConfigure still has better performance when road conditions change dramatically, we used three videos collected from the dashcam, and since they are relatively similar, we combined them into one video.

4.2. Configure Selection

We simulate the environment with three object detection models, which are SSD ResNet152V1, FasterRCNN ResNet50V1, FasterRCNN InceptionResNetV2. For each model, two kinds of image resolution for inference, which



(a) M6 (b) Duke (c) Multi Camera Dataset

Fig. 3. The effect of different models, frame rate, and resolutions on accuracy and processing time. AutoConfigure (blue) consistently outperforms the baseline of one-time update (magenta) across different datasets. Each dot represents the results of running each solution.

is 1024 × 1024 and 640 × 640, can be picked up. Besides choosing a proper configure of model and image resolution, AutoConfigure needs to select a proper number of frames per second(fps). We in our experiment, we set the choices space of fps as $\{1, 2, 5, 10, 15, 20, 25, 30\}$. Therefore, the number of actions that the agent can choose is forty-eight.

4.3. Experiment Parameters

We apply deep Q-learning policy to train the agent. In the training procedure, we build up eight independently identical environments for each data set to speed up the training and decrease the dependency of the data. The leaning rate is set to 0.001. The discount factor is set to 0.9. For each dataset, we train the agent 10 epochs and 1000 steps per epoch. The exploration factor is 0.9. The inference models are implemented in Tensorflow and are pretrained on standard image datasets [27], and the switching of video frame rate and resolution is done by FFmpeg [28].

4.4. Experiment Results

5. CONCLUSION

6. REFERENCES

[1] 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.

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *International conference on machine learning*, 2016, pp. 1928–1937.

[10] 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\} Sympo-*

sium on Networked Systems Design and Implementation ({NSDI} 17), 2017, pp. 377–392.

[11] 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.

[12] 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.

[13] 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.

[14] Haichen Shen, Seungyeop Han, Matthai Philipose, and Arvind Krishnamurthy, "Fast video classification via adaptive cascading of deep models," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3646–3654.

[15] Peng Yang, Feng Lyu, Wen Wu, Ning Zhang, Li Yu, and Xuemin Sherman Shen, "Edge coordinated query configuration for low-latency and accurate video analytics," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4855–4864, 2019.

[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] 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.

[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] 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.

[26] 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.

[27] 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.

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