We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

186,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS
BOOK CITATION INDEX
INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Dynamic Decision-Making for Stabilized Deep Learning Software Platforms

Soohyun Park, Dohyun Kim and Joongheon Kim

Abstract

This chapter introduces a dynamic and low-complexity decision-making algorithm which aims at time-average utility maximization in real-time deep learning platforms, inspired by Lyapunov optimization. In deep learning computation, large delays can happen due to the fact that it is computationally expensive. Thus, handling the delays is an important issue for the commercialization of deep learning algorithms. In this chapter, the proposed algorithm observes system delays at first formulated by queue-backlog, and then it dynamically conducts sequential decision-making under the tradeoff between utility (i.e., deep learning performance) and system delays. In order to evaluate the proposed decision-making algorithm, the performance evaluation results with real-world data are presented under the applications of super-resolution frameworks. Lastly, this chapter summarizes that the Lyapunov optimization algorithm can be used in various emerging applications.

Keywords: Lyapunov optimization, stochastic optimization, real-time computing, deep learning platforms, computer vision platforms

1. Introduction

Nowadays, many machine learning and deep learning algorithms have been developed in various applications such as computer vision, natural language processing, and so forth. Furthermore, the performances of the algorithms are getting better. Thus, the developments of machine learning and deep learning algorithms become mature. However, the research contributions which are focusing on the real-world implementation of the algorithms are relatively less than the developments of the algorithms themselves.

In order to operate the deep learning algorithms in real-world applications, it is essential to think about the real-time computation. Thus, the consideration of delay handling is desired because deep learning algorithm computation generally introduces large delays [1].

In communications and networks research literature, there exists a well-known stochastic optimization algorithm which is for utility function maximization while maintaining system stability. Here, the stability is modeled with queue, and then the algorithm aims at the optimization computation while stabilizing the queue dynamics. In order to formulate the stability, the queue is mathematically modeled with Lyapunov drift [2].

This algorithm is designed inspired by Lyapunov control theory, and thus, it is named to Lyapunov optimization theory [2]. In this chapter, the basic theory, examples, and discussions of the Lyapunov optimization theory are presented. Then, the use of Lyapunov optimization theory for real-time computer vision and deep learning platforms is discussed. Furthermore, the performance evaluation results with real-world deep learning framework computation (e.g., real-world image super-resolution computation results with various models) are presented in various aspects. Finally, the emerging applications will be introduced.

2. Stabilized control for reliable deep learning platforms

In this section, Lyapunov optimization theory which is for time-average optimization subject to stability is introduced at first (refer to Section 2.1), and then example-based explanation is presented (refer to Section 2.2). Finally, related discussions are organized (refer to Section 2.3).

2.1 Theory

In this section, we introduce the Lyapunov optimization theory which aims at time-average penalty function minimization subject to queue stability. Notice that the time-average penalty function minimization can be equivalently converted to time-average utility function maximization. The Lyapunov optimization theory can be used when the tradeoff exists between utility and stability. For example, it can be obviously seen that the tradeoff exists when current decision-making is optimal in terms of the minimization of penalty function, whereas the operation of the decision takes a lot of time, i.e., thus it introduces delays (i.e., queue-backlog increases in the system). Then, the optimal decision can be dynamically time-varying because focusing on utility maximization (i.e., penalty function minimization) is better when the delay in the current system is not serious (i.e., queueing delay is small or marginal). On the other hand, the optimal decision will be for the delay reduction when the delay in the current system is large. In this case, the decision should be for delay reduction while sacrificing certain amounts of utility maximization (or penalty function minimization).

Suppose that our time-average penalty function is denoted by $P(\alpha[t])$ and it should be minimized and our control action decision-making is denoted by $\alpha[t]$. Then, the queue dynamics in the system, i.e., $Q[t]$, can be formulated as follows:

$$Q[t+1] = \max\{Q[t] + a(\alpha[t]) - b(\alpha[t]), 0\} \qquad (1)$$

$$Q[0] = 0 \qquad (2)$$

where $a(\alpha[t])$ is an arrival process at $Q[t]$ at t when our control action decision-making is $\alpha[t]$. In (1), $b(\alpha[t])$ is a departure/service process at $Q[t]$ when our control action decision-making is $\alpha[t]$ at t.

In this section, control action decision-making should be made in each unit time for time-average penalty function minimization subject to queue stability. Then, the mathematical program for minimizing time-average penalty function, $P(\alpha[t])$ where the control action decision-making at t is $\alpha[t]$, can be presented as follows:

$$\min : \lim_{t \to \infty} \sum_{\tau=0}^{t-1} P(\alpha[\tau]) \qquad (3)$$

Subject to queue stability:

$$\lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} Q[\tau] < \infty. \tag{4}$$

In (3), $P(\alpha[t])$ stands for the penalty function when a control action decision-making is $\alpha[t]$ at t.

As mentioned, the Lyapunov optimization theory can be used when tradeoff between utility maximization (or penalty function minimization) and delays exists. Based on this nature, drift-plus-penalty (DPP) algorithm [2–4] is designed for maximizing the time-average utility subject to queue stability. Here, the Lyapunov function is defined as $L(Q[t]) = \frac{1}{2}(Q[t])^2$, and let $\Delta(.)$ be a conditional quadratic Lyapunov function which is formulated as $\mathbb{E}[L(Q[t+1]) - L(Q[t])|Q[t]]$, which is called as the drift on t. According to [2], this dynamic policy is designed to achieve queue stability by minimizing an upper bound of our considering penalty function on DPP which is given by

$$\Delta(Q[t]) + V\mathbb{E}[P(\alpha[t])], \tag{5}$$

where V is a tradeoff coefficient. The upper bound on the drift of the Lyapunov function at t is derived as follows:

$$L(Q[t+1]) - L(Q[t]) = \frac{1}{2}\left(Q\left([t+1]^2 - Q[t]^2\right)\right) \tag{6}$$

$$\leq \frac{1}{2}\left(a(\alpha[t])^2 + b(\alpha[t])^2\right) + Q[t](a(\alpha[t]) - b(\alpha[t])). \tag{7}$$

Therefore, the upper bound of the conditional Lyapunov drift can be derived as follows:

$$\Delta(Q(t)) = \mathbb{E}[L(Q[t+1]) - L(Q[t])|Q[t]]$$
$$\leq C + \mathbb{E}[Q[t](a(\alpha[t]) - b(\alpha[t])|Q[t]], \tag{8}$$

where C is a constant given by

$$\frac{1}{2}\mathbb{E}\left[a(\alpha[t])^2 + b(\alpha[t])^2|Q[t]\right] \leq C, \tag{9}$$

which supposes that the arrival and departure process rates are upper bounded. Due to the fact that C is a constant, minimizing the upper bound on DPP is as follows:

$$V\mathbb{E}[P(\alpha[t])] + \mathbb{E}[Q[t] \cdot (a(\alpha[t]) - b(\alpha[t]))]. \tag{10}$$

Algorithm 1. Stabilized Time-Average Penalty Function Minimization

---

Initialize:
1: $t \leftarrow 0$;
2: $Q[t] \leftarrow 0$;
3: Decision Action: $\forall \alpha[t] \in \mathcal{A}$
Time-Average Penalty Function Minimization subject to Stability
4: **while** $t \leq T$ **do** // T: operation time
5:      Observe $Q[t]$;

```
6:      𝒯* ← ∞;
7:      for α[t] ∈ 𝒜 do
8:          𝒯 ← V · P(α[t]) + Q[t] · (a(α[t]) − b(α[t]));
9:          if 𝒯 ≤ 𝒯* then
10:             𝒯* ← 𝒯;
11:             α*[t + 1] ← α[t];
12:         end if
13:     end for
14: end while
```

Finally, the dynamic control action decision-making $\alpha[t]$ in each unit time t for time-average penalty function $P(\alpha[t])$ minimization subject to queue stability can be formulated as follows based on the Lyapunov optimization theory:

$$\boxed{\alpha^*[t+1] \leftarrow \arg\min_{\alpha[t]\in\mathcal{A}} [V \cdot P(\alpha[t]) + Q[t] \cdot (a(\alpha[t]) - b(\alpha[t]))]} \tag{11}$$

where $\mathcal{A}$ is the set of all possible control actions and $\alpha^*[t+1]$ is the optimal control action decision-making for the next time slot.

In order to verify whether (11) works correctly or not, following two example cases can be considerable:

- *Case 1:* Suppose $Q[t] \approx \infty$. Then

$$\alpha^*[t+1] \leftarrow \arg\min_{\alpha[t]\in\mathcal{A}} [V \cdot P(\alpha[t]) + Q[t] \cdot (a(\alpha[t]) - b(\alpha[t]))] \tag{12}$$

$$\approx \arg\min_{\alpha[t]\in\mathcal{A}} [a(\alpha[t]) - b(\alpha[t])]. \tag{13}$$

Then, (13) shows that control action decision-making should works as follows, i.e., (i) the arrival process should be minimized, and (ii) the departure process should be maximized. Both cases are for stabilizing the queue, i.e., it should be beneficial when $Q[t] \approx \infty$.

- *Case 2:* Suppose $Q[t] = 0$. Then

$$\alpha^*[t+1] \leftarrow \arg\min_{\alpha[t]\in\mathcal{A}} [V \cdot P(\alpha[t]) + Q[t] \cdot (a(\alpha[t]) - b(\alpha[t]))] \tag{14}$$

$$= \arg\min_{\alpha[t]\in\mathcal{A}} V \cdot P(\alpha[t]). \tag{15}$$

Then, (15) shows that control action decision-making should work for minimizing the given penalty function. This is semantically reasonable because focusing on our main objective is possible because stability does not need to be considered because $Q[t] = 0$.

The pseudo-code of the proposed time-average penalty function minimization algorithm is presented in Algorithm 1. From line 1 to line 3, all variables and parameters are initialized. The algorithm works in each unit time as shown in line 4. In line 5, current queue-backlog $Q[t]$ is observed to be used in (11). From line 7 to line 13, the main computation procedure for (11) is described.

Up to now, the time-average penalty function minimization is considered. Based on the theory, the dynamic control action decision-making $\alpha[t]$ in each unit time t for time-average utility function $U(\alpha[t])$ maximization subject to queue stability can be formulated as follows:

$$\alpha^*[t+1] \leftarrow \arg\max_{\alpha[t] \in \mathcal{A}} [V \cdot U(\alpha[t]) - Q[t] \cdot (a(\alpha[t]) - b(\alpha[t]))] \qquad (16)$$

where $\mathcal{A}$ is the set of all possible control actions and $\alpha^*[t+1]$ is the optimal control action decision-making for the next time slot.

2.2 Example: multicore scheduling in mobile devices

In this section, the Lyapunov optimization-based stabilized time-average optimization algorithm is introduced with one simple toy model. In this example, dynamic core allocation decision-making algorithm is designed which is for time-average energy consumption minimization subject to queue stability.

As illustrated in **Figure 1**, mobile smartphone is with the processor which is equipped with multiple cores. For example, ARM big.LITTLE processors are with multiple little and big heterogeneous cores.

In this system, the task events will be generated when users generate events, which are denoted by $a[t]$ in **Figure 1**. Then, the events will be located in the task queue (i.e., $Q[t]$ in **Figure 1**). Then, the events can be processed by the multicore processor. In this case, if many/more cores are allocated in order to process the events from the queue, the processing can be accelerated which is beneficial in terms of queue stability. However, it is not good in terms of our main objective, i.e., energy consumption minimization. On the other hand, if less cores are allocated, the processing becomes slow which is harmful in terms of queue stability but is beneficial in terms of our main objective, i.e., energy consumption minimization. Finally, the tradeoff can be observed between energy consumption minimization (i.e., our main objective) and stability. Then, it can be confirmed that Lyapunov optimization-based algorithm can be used.

In order to design the dynamic core allocation decision-making, $\alpha[t]$ in each unit time t for time-average energy consumption $E(\alpha[t])$ minimization subject to queue stability can be formulated as follows based on (11):

$$\alpha^*[t+1] \leftarrow \arg\min_{\alpha[t] \in \mathcal{A}} [V \cdot E(\alpha[t]) + Q[t] \cdot (a(\alpha[t]) - b(\alpha[t]))] \qquad (17)$$

where $\mathcal{A}$ is the set of all possible core allocation combinations and $\alpha^*[t+1]$ is the optimal core allocation decision-making for the next time slot. Here, it is obvious
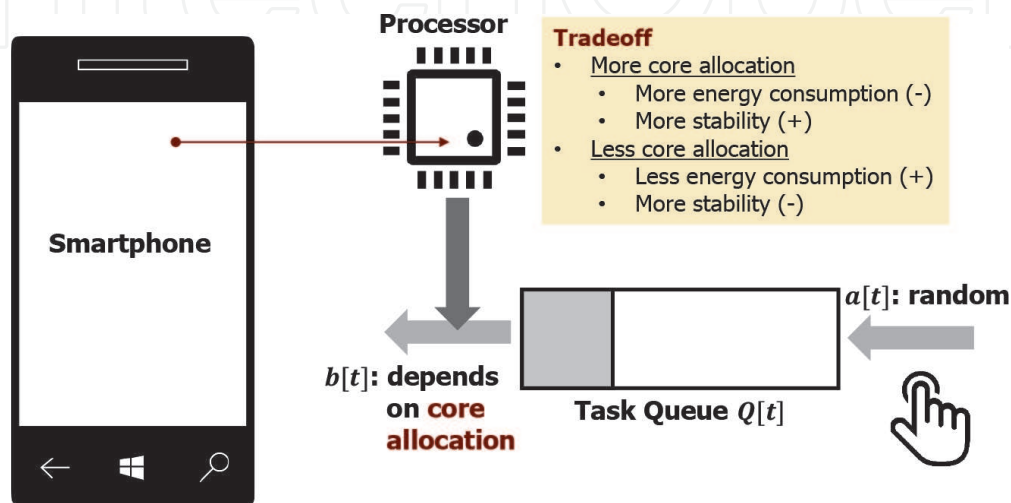
Figure 1.
Mobile devices with multicore processors.

that the arrival process is not controllable (i.i.d. random events); thus, it can be ignored. Then, the final form of the dynamic decision-making algorithm can be defined as follows:

$$\alpha^*[t+1] \leftarrow \arg\min_{\alpha[t] \in \mathcal{A}} [V \cdot E(\alpha[t]) - Q[t] \cdot b(\alpha[t])]. \tag{18}$$

In order to check whether the derived Eq. (18) is correct or not, two example cases can be considered, i.e., (i) $Q[t] \approx \infty$, and (ii) $Q[t] = 0$:

- *Busy queue case* $(Q[t] \approx \infty)$: in this case

$$\alpha^*[t+1] \leftarrow \arg\min_{\alpha[t] \in \mathcal{A}} [V \cdot E(\alpha[t]) - Q[t] \cdot b(\alpha[t])], \tag{19}$$

$$= \arg\min_{\alpha[t] \in \mathcal{A}} [-b(\alpha[t])] = \arg\max_{\alpha[t] \in \mathcal{A}} b(\alpha[t]), \tag{20}$$

Thus, the departure process should be accelerated, i.e., more cores should be allocated. This is semantically true because the fast processing events from the queue is desired if overflow situations happen.

- *Busy queue case* $(Q[t] = 0)$: In this case

$$\alpha^*[t+1] \leftarrow \arg\min_{\alpha[t] \in \mathcal{A}} [V \cdot E(\alpha[t]) - Q[t] \cdot b(\alpha[t])], \tag{21}$$

$$= \arg\min_{\alpha[t] \in \mathcal{A}} V \cdot E(\alpha[t]), \tag{22}$$

Thus, less cores should be allocated for energy consumption minimization which is our main objective. This is semantically true because the given main objective should be desired if the system is stable, i.e., $Q[t] = 0$.

As discussed with examples, the proposed Lyapunov optimization-based dynamic core allocation decision-making algorithm works as desired.

2.3 Discussions in stabilized control

The proposed dynamic super-resolution model selection algorithm is beneficial in various aspects, as follows.

2.3.1 Hardware/system-independent self-adaptation

Suppose that this proposed algorithm is implemented in supercomputer-like high-performance computing machines. In this case, the processing should be fast; thus, the queue-backlog is always low. Therefore, the system has more chances to focus on our main objective, i.e., penalty function minimization or utility function maximization. On the other hand, if the hardware itself is performance/resource limited (e.g., mobile devices), then the processing speed is also limited due to the low specifications in processors. Thus, the queue-backlog can be frequently busy because it may not be able to process many data with the queue even though it utilizes the fastest model. Therefore, it can be finally observed that the proposed algorithm is self-adaptive which can adapt depending on the given hardware/system specifications. It automatically adapts the models based on the given hardware/system; thus, it does not require system engineer's trial-and-error tuning.

Furthermore, the proposed algorithm is reliable according to the fact that the self-adaptation is for maximizing its *utility* while maintaining *stability*.

2.3.2 Low-complexity operation

As shown in Algorithm 1, the computation procedure is iterative for solving closed-form equation, i.e., (11) and (16). Thus, the computational complexity of the proposed algorithm is polynomial time, i.e., $O(N)$, where N is the number of the given control actions. Thus, it guarantees low-complexity operations.

3. The use of Lyapunov optimization for deep learning platforms

As explained, the Lyapunov optimization theory is a scalable, self-configurable, low-complexity algorithm which can be used in many applications. In this section, the use of Lyapunov optimization for deep learning and computer platforms is discussed in two different ways, i.e., departure process control (refer to Section 3.1) and arrival process control (refer to Section 3.2). Finally, its related performance evaluation results are presented (refer to Section 3.3).
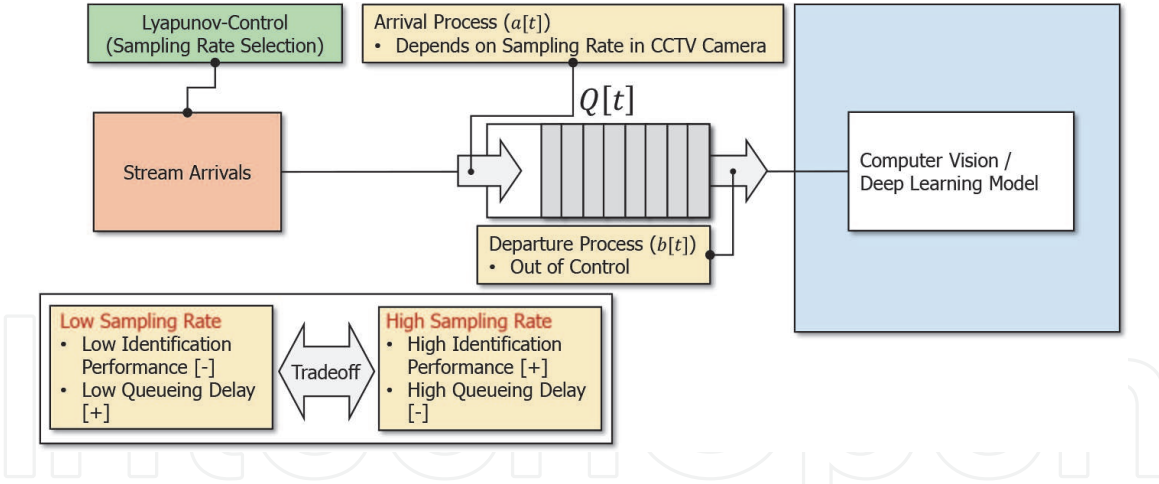
3.1 Lyapunov control over departure processes

As illustrated in **Figure 2**, stabilized real-time computer vision platforms should be equipped with queues in order to handle bursty traffics. If the queue is busy or near-overflow, the departure process should be accelerated. Thus, the simplest model should be used for reducing the corresponding computation. On the other hand, if the queue is empty, deep learning computation accuracy can be improved with more sophisticate models because we have enough time to conduct the computation. Thus, multiple models are desired in order to select one depending on queue backlog.

In **Figure 2**, multiple models exist, and it can be seen that the simplest model (i.e., low-resolution model) is able to conduct fast computation, but it presents low learning accuracy. On the other hand, the most sophisticate model (i.e., high-resolution model) is good for accurate learning performance, but it introduces computation delays. Thus, the tradeoff exists between performance and delays, i.e.,
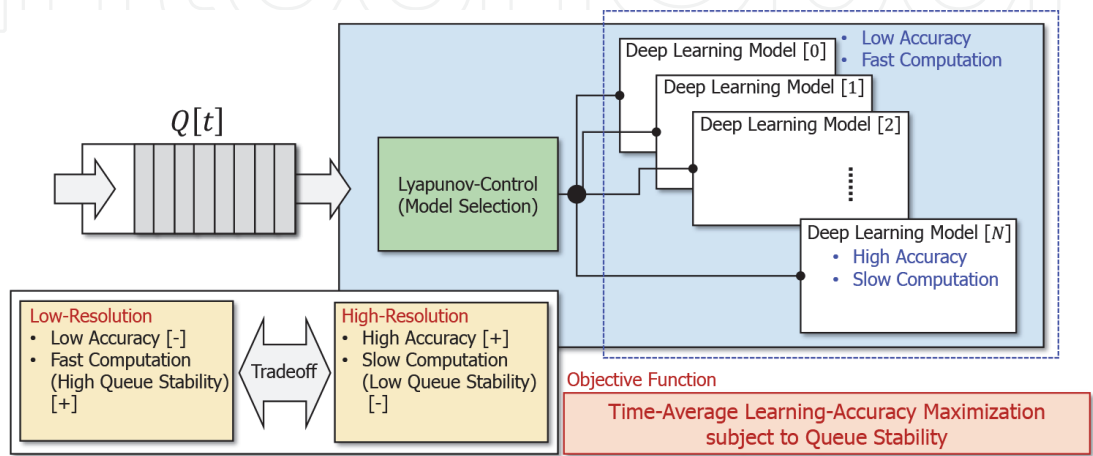
Figure 2.
Lyapunov control over departure processes in real-time computer vision platforms for time-average learning accuracy maximization subject to queue stability.

Figure 3.
Lyapunov control over arrival processes in real-time computer vision platforms for time-average learning accuracy maximization subject to queue stability.

Lyapunov optimization theory-based dynamic model selection decision-making algorithm can be designed as follows:

$$\alpha^*[t+1] \leftarrow \arg\max_{\alpha[t] \in \mathcal{A}} [V \cdot A(\alpha[t]) - Q[t] \cdot (a(\alpha[t]) - b(\alpha[t]))] \qquad (23)$$

and this can be reformulated as follows due to the fact that the arrival process is out of control:

$$\alpha^*[t+1] \leftarrow \arg\max_{\alpha[t] \in \mathcal{A}} [V \cdot A(\alpha[t]) + Q[t] \cdot b(\alpha[t])] \qquad (24)$$

where $A(\alpha[t])$ stands for the learning-accuracy when the model selection decision is $\alpha[t]$ at t. Here, $\mathcal{A}$ is the set of all possible deep learning models, and $\alpha^*[t+1]$ is the optimal control action decision-making for next time slot.

3.2 Lyapunov control over arrival processes

The stabilized real-time computer vision platform in Section 3.1 is novel and scalable; however it has burden because multiple deep learning models should be implemented in a single platform.

Thus, a new dynamic control algorithm with a single deep learning model is also needed for resource-limited systems. As illustrated in **Figure 3**, our considering system has a single computer vision and deep learning model in computing platforms. In addition, the queue is in front of the system. Thus, the departure process is not controllable anymore. In this case, the arrival process should be controllable in order to control the queue dynamics for stability. Therefore, the arrival image/video streams should be controlled by handling sample rates. If high-frequency sampling is available, more signals will be generated, and then the results will be enqueued. Thus, the arrival process increases. This is beneficial because it increases computer vision performance due to the fact that more images/videos can be obtained especially in surveillance applications. On the other hand, i.e., if low-frequency sampling is conducted, the computer vision performance can be degraded, whereas the number of arrival process data decreases which is beneficial in terms of stability. Eventually, the tradeoff between computer vision performance and delays can be observed. Finally, Lyapunov optimization theory-based sampling rate selection decision-making algorithm can be designed as follows:

$$\alpha^*[t+1] \leftarrow \arg\max_{\alpha[t] \in \mathcal{A}} [V \cdot A(\alpha[t]) - Q[t] \cdot (a(\alpha[t]) - b(\alpha[t]))] \tag{25}$$

and this can be reformulated as follows due to the fact that the departure process is out of control:

$$\alpha^*[t+1] \leftarrow \arg\max_{\alpha[t] \in \mathcal{A}} [V \cdot A(\alpha[t]) - Q[t] \cdot a(\alpha[t])] \tag{26}$$

where $A(\alpha[t])$ stands for the learning accuracy when the sample rate selection decision is $\alpha[t]$ at t. Here, $\mathcal{A}$ is the set of all possible sample rates, and $\alpha^*[t+1]$ is the optimal control action decision-making for next time slot.

3.3 Performance evaluation and discussions

In this section, the performance evaluation results of the proposed algorithm in Section 3.1 are presented. The data-intensive simulation-based evaluation is performed, and then the results are presented in **Figure 4**. In addition, **Table 1** shows the performance of super-resolution depending on the number of hidden layers. If the number of hidden layers is maximum (i.e., 20 in this research), the PSNR and structural similarity (SSIM, one of the widely used performance metrics in super-resolution) values are maximum. However, the computation times (for CPU-only and CPU-GPU) become slow.

As illustrated in **Figure 4**, if the models are static (i.e., *deep* or *shallow*), the curves show that the two models are not efficient. The deep model cannot handle the overflow situations; thus, the queue diverges. On the other hand, the shallow
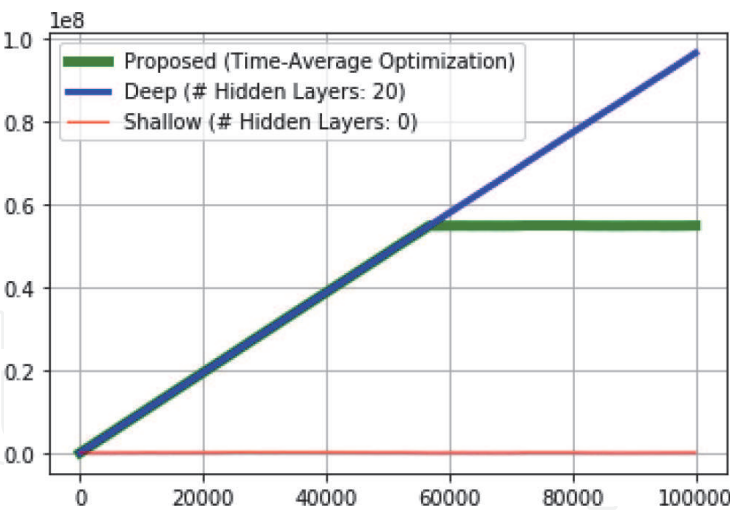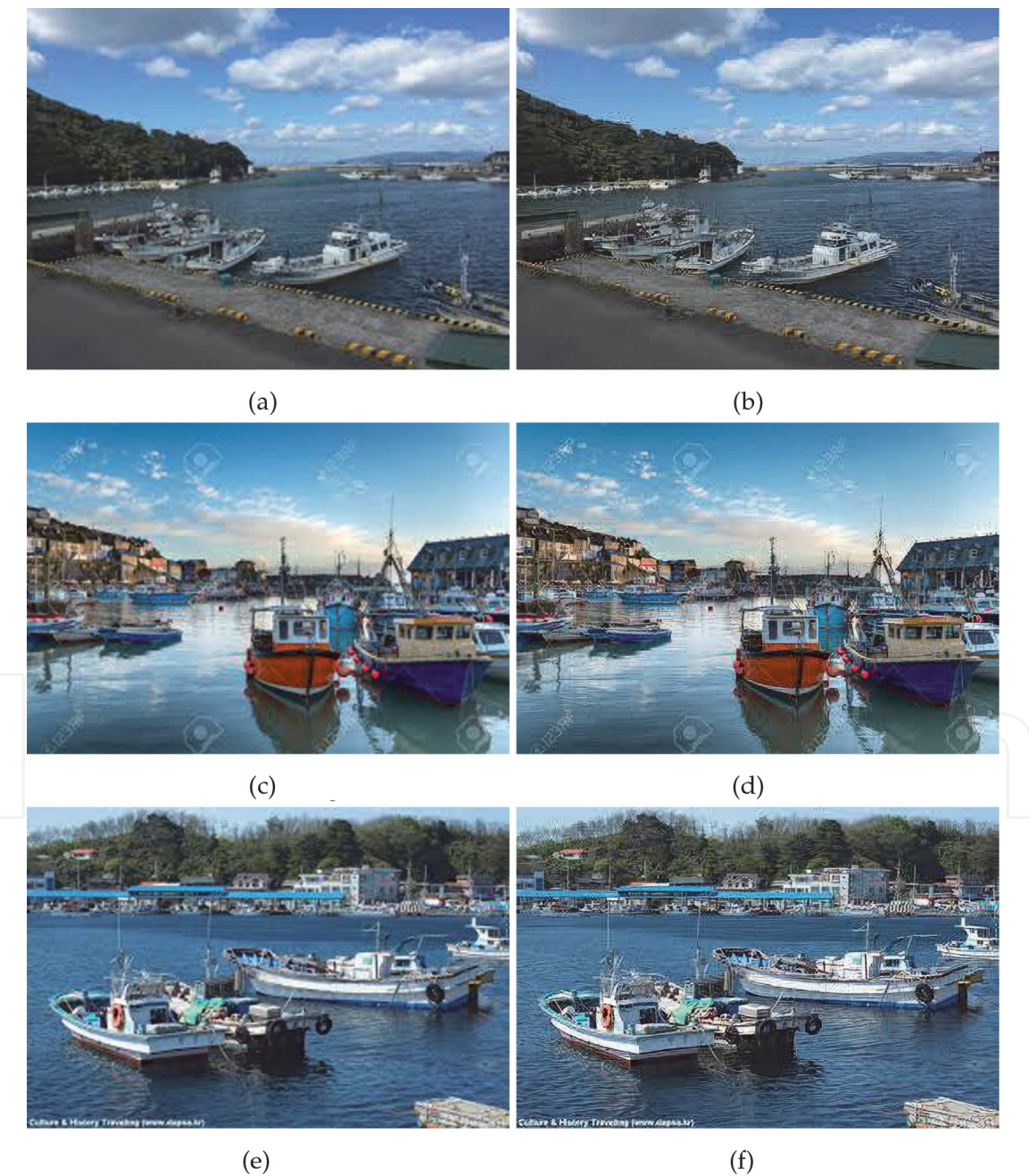

Figure 4.
Performance evaluation: Queue-backlog (x-axis, unit time; x-axis, queue occupancy (unit: Bits)).

Depth (# of hidden layers)	0	4	6	8	11	14	17	20
PSNR (dB)	30.400	32.560	33.010	33.229	33.379	33.435	33.495	33.523
SSIM	0.8682	0.9100	0.9160	0.9180	0.9200	0.9200	0.9210	0.9220
Processing time (CPU − only)	0.0020	0.3210	0.5468	0.7725	0.9940	1.3170	1.6220	1.9600
Processing time (CPU + GPU)	0.0010	0.0100	0.0120	0.0152	0.0189	0.0224	0.0262	0.0305

Table 1.
Tradeoff between utility and delay obtained from super-resolution performance measurement results (processing time have measured on 512×768 images).

model is too fast; thus, the queue is always empty. This is obviously positive for stability where the performance in terms of super-resolution performance is the lowest. Thus, it might be better if the algorithm allows certain amounts of delays in order to enhance the quality of super-resolution. The proposed algorithm is initially follows *deep* model because the queue is idle during the initial phases. If the queue becomes filled with certain amounts of images (i.e., near threshold), it starts the control, i.e., self-adaptive, near the unit time of 5800. Thus, the proposed algorithm starts to select super-resolution models which can handle delays. Thus, it is true that the proposed algorithm is better than the other two static algorithms.

For the proposed self-adaptive stabilized algorithm, the evaluation with two processing capabilities (CPU-only platform vs. CPU-GPU platform), it can be observed that the CPU-GPU platform selects the maximum performance super-resolution model (i.e., 20 hidden layers in **Table 1**) 4.36 times more than the CPU-

(a)   (b)

(c)   (d)

(e)   (f)

Figure 5.
Super-resolution computation results. Note that the model for low-resolution is bicubic which has no hidden layers. (a) Image #1 (low-resolution), (b) image #1 (high-resolution), (c) image #2 (low-resolution), (d) image #2 (high-resolution), (e) image #3 (low-resolution) and (f) image #3 (high-resolution).

only platform. It means that the proposed algorithm is self-adaptive depending on the hardware/platform requirements. This is obviously beneficial in terms of system engineers because they do not need to conduct trial-and-error-based system parameter tuning anymore.

In order to confirm the performance of super-resolution models, **Figure 5** shows the super-resolution computation results with real-world images. As can be seen in the figures, the super-resolution models show better performances if they have more hidden layers, as shown in **Figure 5b**, **Figure 5d**, and **Figure 5f**. For the super-resolution computation without hidden layers, this paper uses bicubic interpolation, as shown in **Figure 5a**, **Figure 5c**, and **Figure 5e**. Finally, these results show that our considering Lyapunov control algorithms for adaptive deep learning platforms can make different super-resolution performance depending on queue-backlog size information.

4. Emerging applications

As presented, the Lyapunov optimization framework is for time-average utility maximization while achieving queue stability; and this theory is scalable; thus it is widely applicable [2]. Therefore, there exist many applications based on this algorithm as follows.

4.1 Adaptive video streaming

Kim et al. [3, 5] design a dynamic control algorithm for time-average streaming quality (i.e., peak-signal-to-noise ratio (PSNR)) maximization subject to transmit buffer stability in wireless video networks. Koo et al. [6, 7] also propose a novel dynamic adaptive streaming over HTTP (DASH)-based mechanism for video streaming quality maximization under the consideration of battery status, LTE data quota, and stability in hybrid LTE and WiFi networks.

4.2 Networks

Neely et al. [8] proposed a novel dynamic multi-hop routing algorithm which is for energy-efficient data/packet forwarding in wireless ad hoc and sensor networks subject to queue stability.

4.3 Security applications: surveillance monitoring

Mo et al. [9] design a deep learning framework for CCTV-based distributed surveillance applications. In the system, multiple deep learning frameworks exist; and each deep learning model is with its own configurations. In this situation, there exists a tradeoff between complexity and performance. Therefore, the proposed CCTV-based surveillance algorithm adaptively selects a deep learning model depending on queue-backlog in the system for recognition performance maximization subject to CCTV queue stability. Kim et al. [10] also design a novel face identification deep learning frameworks for CCTV-based surveillance platforms. Instead of having multiple deep learning models, this system has one learning system (based on OpenFace open-source software library) and controls the sampling rates of the CCTV camera. Finally, the proposed decision-making algorithm dynamically selects CCTV sampling rates for recognition performance maximization subject to CCTV queue stability.

4.4 Others

The application of Lyapunov optimization-based dynamic control algorithm for dynamic reinforcement learning policy design is illustrated in [11]. In addition, the adaptive control algorithms using the Lyapunov optimization framework in stock market pricing and smart grid are introduced in [12, 13].

5. Conclusions

This chapter introduces a dynamic control decision-making algorithm, inspired by Lyapunov optimization theory under the situation where the tradeoff between utility/performance and delays exists. Thus, the dynamic decision-making algorithms aim at time-average utility maximization (or penalty minimization) in real-time deep learning platforms. As discussed, the Lyapunov optimization-based algorithms are scalable, hardware/system-independent, self-configurable, and low-complexity. Thus, it can be used in various emerging applications such as video streaming, wireless networks, security applications, and smart grid applications.

Acknowledgements

Conflict of interest

The authors declare no conflict of interest.

Author details

Soohyun Park[1], Dohyun Kim[2] and Joongheon Kim[1*]

1 Korea University, Seoul, Republic of Korea

2 Naver Webtoon Corporation, Seongnam, Republic of Korea

*Address all correspondence to: joongheon@korea.ac.kr

IntechOpen

References

[1] Kim D, Kwon J, Kim J. Low-complexity online model selection with Lyapunov control for reward maximization in stabilized real-time deep learning platforms. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC '18); 7–10 October, 2018; Miyazaki, Japan: IEEE; 2018. pp. 4363-4368

[2] Neely M. Stochastic Network Optimization with Application to Communication and Queueing Systems. Vermont, USA: Morgan & Claypool; 2010

[3] Kim J, Caire G, Molisch A. Quality-aware streaming and scheduling for device-to-device video delivery. IEEE/ACM Transactions on Networking. 2016;**24**:2319-2331. DOI: 10.1109/TNET.2015.2452272

[4] Choi M, Kim J, Moon J. Adaptive detector selection for queue-stable word error rate minimization in connected vehicle receiver design. IEEE Transactions on Vehicular Technology. 2018;**67**:3635-3639. DOI: 10.1109/TVT.2017.2776327

[5] Kim J, Meng F, Chen P, Egilmez H, Bethanabhotla D, Molisch A, et al. Demo: Adaptive video streaming for device-to-device mobile platforms. In: Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom '13), 30 September–4 October, 2013; Miami, FL, USA: IEEE; 2013

[6] Koo J, Yi J, Kim J, Hoque M, Choi S. REQUEST: Seamless dynamic adaptive streaming over HTTP for multi-homed smartphone under resource constraints. In: Proceedings of the ACM International Conference on Multimedia (MM '17), 23–27 October, 2017; Mountain View, CA, USA: IEEE; 2017

[7] Koo J, Yi J, Kim J, Hoque M, Choi S. Seamless dynamic adaptive streaming in LTE/Wi-fi integrated network under smartphone resource constraints. IEEE Transactions on Mobile Computing. 2019;**18**:1647-1660. DOI: 10.1109/TMC.2018.2863234

[8] Neely M. Energy optimal control for time varying wireless networks. IEEE Transactions on Information Theory. 2006;**52**:2915-2934. DOI: 10.1109/TIT.2006.876219

[9] Kim J, Mo YJ, Lee W, Nyang D. Dynamic security-level maximization for stabilized parallel deep learning architectures in surveillance applications. In: Proceedings of the IEEE Symposium on Privacy-Aware Computing (PAC '07); 1–3 August, 2017; Washington DC, USA: IEEE; 2017. pp. 192-193

[10] Kim D, Kim J, Bang J. A reliable, self-adaptive face identification framework via Lyapunov optimization. In: Proceedings of ACM Symposium on Operating Systems Principles (SOSP) AI Systems Workshop (AISys '17), 28 October, 2017; Shanghai, China: ACM; 2017

[11] Neely M, Supittayapornpong S. Dynamic Markov decision policies for delay constrained wireless scheduling. IEEE Transactions on Automatic Control. 2013;**58**:1948-1961. DOI: 10.1109/TAC.2013.2256682

[12] Neely M. Stock market trading via stochastic network optimization. In: Proceedings of IEEE Conference on Decision and Control (CDC '10), 15–17 December, 2010; Atlanta, GA, USA: IEEE; 2010

[13] Neely M, Tehrani A, Dimakis A. Efficient algorithms for renewable energy allocation to delay tolerant consumers. In: Proceedings of IEEE International Conference on Smart Grid Communication (SmartGridComm '10), 4–6 October, 2010; Gaithersburg, MD, USA: IEEE; 2010