Collision Detection for Industrial Collaborative Robots: A Deep Learning Approach

Young Jin Heo[1,3], Dayeon Kim[1,3], Woongyong Lee[1], Hyoungkyun Kim[1],
Jonghoon Park[3], *Member, IEEE*, and Wan Kyun Chung[2], *Fellow, IEEE*

***Abstract*— With increased human-robot interactions in industrial settings, a safe and reliable collision detection framework has become an indispensable element of collaborative robots. The conventional framework detects collisions by estimating collision monitoring signals with a particular type of observer, which is followed by collision decision processes. This results in unavoidable trade-off between sensitivity to collisions and robustness to false alarms. In this study, we propose a collision detection framework (CollisionNet) based on a deep learning approach. We designed a deep neural network model to learn robot collision signals and recognize any occurrence of a collision. This data-driven approach unifies feature extraction from high-dimensional signals and the decision processes. CollisionNet eliminates heuristic and cumbersome nature of the traditional decision processes, showing high detection performance and generalization capability in real time. We performed quantitative analysis and verified the performance of the proposed framework through various experiments.**

I. Introduction

Robots that work in collaboration with humans are going mainstream in industrial robotics [1]–[3]. As opposed to traditional industrial robots that are kept in safety cages, the collaborative robots (cobots), which physically interact with human workers in a shared workspace, integrate advantages of both humans and robots to achieve high productivity. Without safety assurances for physical contacts, however, the robots would not manage to escape from the cages. In this context, fast and reliable handling of possible collisions has become more imperative than ever before.

The problem of collision handling can be divided into five subproblems: detection, isolation, identification, classification, and reaction [4], [5]. Here, we focus on *detection*, which is the primary step of collision management. The aim of collision detection is to determine in real time whether a collision has occurred. When a collision is detected, a cobot stops immediately at its position. To assure both safety and productivity, the detection ability must achieve high sensitivity while avoiding false alarms.

To date, there have been two main approaches to collision detection: using exteroceptive sensors and only proprioceptive sensors. The former includes the development of artificial skin, which covers a robot body with a number of tactile sensors [6]–[8]. The artificial skin would make the detection ability robust to contact locations, but has been slow to come to reality due to difficulties in design and development of tactile sensors that can cover entire robot body. On the other hand, the latter uses dynamic modeling of robots to perform collision detection without the need for external sensors. An initial approach was to compare the commanded input torque to the actual input torque, and to assume that any difference above a fixed threshold is result of a collision [9]. Several collision-monitoring schemes based on this idea have been developed [5], [10], [11]. Time-varying collision detection thresholds have been proposed [12] to achieve more accurate detection performance. While these model-based methods are considered more cost effective than using the external sensors, model uncertainty and disturbance can degrade the accuracy of monitoring signal. Moreover, the monitoring signals are dynamically variable, and this trait complicates the task of establishing detection threshold.

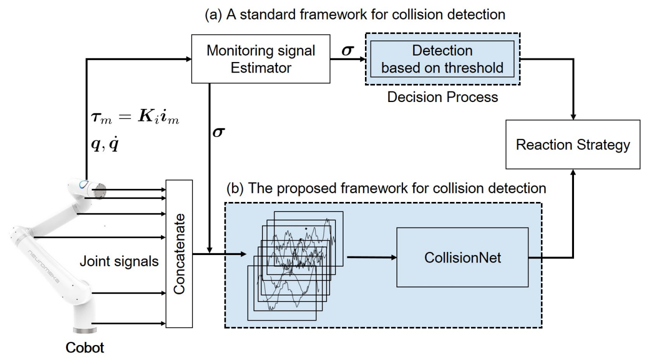
(a) A standard framework for collision detection
Monitoring signal Estimator
$\boldsymbol{\sigma}$
Detection based on threshold
Decision Process
Reaction Strategy
$\boldsymbol{\tau}_m = \boldsymbol{K}_i \boldsymbol{i}_m$
$\boldsymbol{q}, \dot{\boldsymbol{q}}$
$\boldsymbol{\sigma}$
(b) The proposed framework for collision detection
Concatenate
Joint signals
CollisionNet
Cobot

Fig. 1. (a) A standard framework for collision detection consists of monitoring signal estimation and decision of detection based on a heuristic approach. (b) The proposed framework is designed based on a deep learning approach. CollisionNet takes joint signals of a cobot as input, then directly outputs whether or not a collision has occurred.

In this paper, we introduce *CollisionNet*, a deep neural network (DNN) that accepts high-dimensional signals from robot joints as input, then outputs an evaluation of whether or not a collision occurred (Fig. 1). CollisionNet achieved both high sensitivity to collisions and robustness to false-positive detections induced by noisy signals and/or model uncertainty. The proposed detector also has a powerful generalization capability that can preserve high detection accuracy for different robots and various motions. We verified the capabilities of the proposed deep learning-based collision detection

[1]The authors are with the Robotics Laboratory, school of Mechanical Engineering, Pohang University of Science and Technology (POSTECH), Pohang, 790-784, Gyung-buk, Korea {heoyoungjin, dayon95, leewoongyong}@postech.ac.kr

[2]Wan Kyun Chung is with Faculty of Mechanical Engineering, POSTECH, Hyojadong, The South Korea. wkchung@postech.ac.kr

[3] Neuromeka.

framework by experiments using 6-DOF industrial cobots. CollisionNet achieved higher collision-detection accuracy than the standard collision-detection method.

The remainder of this paper is organized as follows. Section II provides background information related to the present research. Section III explains details of the proposed collision detection method. Section IV describes experimental setup and Section V shows the experimental verification. Finally, Section VI presents discussion and conclusion.

II. Preliminaries

In this section, we briefly explain the standard collision detection method, which consists of two sequential processes: 1) estimation of monitoring signals (e.g., external torque at every joint $\boldsymbol{\tau}_{\text{ext,j}}$) and 2) decision processes for detection (Fig. 1a). We also introduce a deep learning approach that uses real-time robot signals. To evaluate the collision detection performance of our method, we introduce two performance metrics that represent its safety and efficiency.

A. Monitoring signal estimation: momentum-based observer

A momentum-based observer is widely used to estimate external joint torques (*i.e.*, the signal that is monitored). We consider the following n degree-of-freedom (DoF) rigid-body dynamics:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau_m + \tau_{\text{ext}}, \tag{1}$$

where $M(q) \in \mathbb{R}^{n\times n}$ is the inertia matrix, $C(q,\dot{q})\dot{q} \in \mathbb{R}^n$ is the vector of Coriolis and centripetal torques, $g(q) \in \mathbb{R}^n$ is the vector of gravitational torques. τ_m is the control torques and τ_{ext} is the external torques. Note that the frictional effect is ignored in (1) because the model-based position control is applied as follows:

$$\tau_m = \widehat{M}(q)\ddot{q}_r + \widehat{C}(q,\dot{q})\dot{q}_r + \widehat{g}(q) + \tau_{\text{ref}}, \tag{2}$$

where $\widehat{M}$, $\widehat{C}$, and $\widehat{g}$ denote the nominal model parameters, $\dot{q}_r = \dot{q} + K_P e + K_I \int e dt$ represents the reference trajectory, q_d represents the desired trajectory, $e = q_d - q$ is the position error, and u is the optimal control input given as

$$\tau_{\text{ref}} = -(K + \frac{1}{\gamma^2})\dot{e}_{\text{ref}} = -(K + \frac{1}{\gamma^2})\big(\dot{e} + K_P e + K_I \int e\big) \tag{3}$$

where $\gamma > 0$, and K, K_I, $K_P > 0$ are diagonal gain matrices satisfying $K_P^2 > 2K_I$ (for details, see [13]).

Next, the following observer is designed to estimate external torques:

$$\hat{\tau}_{\text{ext}} = L\big(-p_e(t) + p_e(0) + \int_0^t (s_e(s) - \tau_{\text{ref}})ds\big) \tag{4}$$

where $p_e = \widehat{M}(q)\dot{e}_r$, $s_e = -\tau_{\text{ref}} + \widehat{C}(q,\dot{q})\dot{e}_r$, and $\dot{e}_r = \dot{q}_r - \dot{q}$. As a consequence, the observer dynamics is given as follows:

$$\dot{\hat{\tau}}_{\text{ext}} = L(-\hat{\tau}_{\text{ext}} + \tau_{\text{ext}}) - L\underbrace{(\widetilde{M}\ddot{q} + \widetilde{C}\dot{q} + \widetilde{g})}_{\tau_d}, \tag{5}$$

where $\widetilde{M} = M - \widehat{M}$, $\widetilde{C} = C - \widehat{C}$, and $\widetilde{g} = g - \widehat{g}$. $\hat{\tau}_{\text{ext}}$ is given by the sum of τ_{ext} and $-\tau_d$ filtered by the first-order low-pass filter, meaning that $\hat{\tau}_{\text{ext}}$ is contaminated by τ_d, especially, at high speed motions.

B. Decision process for detection

Generally, the estimated monitoring signal is evaluated using a decision function given as

$$cd : \boldsymbol{\mu}(t) \rightarrow \{\text{True}, \text{False}\}, \tag{6}$$

$$cd(\boldsymbol{\mu}(t)) = \begin{cases} \text{True, if } |\boldsymbol{\mu}(t)| > \boldsymbol{\epsilon} \\ \text{False, if } |\boldsymbol{\mu}(t)| \leq \boldsymbol{\epsilon} \end{cases}, \tag{7}$$

where cd is a decision function that determines whether a collision has occurred, $\boldsymbol{\mu}(t)$ is the monitoring signal at time t, and $\boldsymbol{\epsilon}$ is the threshold for cd [5].

The threshold-based decision function induces a trade-off between insensitivity to false positives and detection sensitivity: a conservative (high) threshold makes the robot insensitive to collisions, so some collisions below the threshold can cause detection failure; but a sensitive (low) threshold that can detect weak collisions may result in numerous false positives. False positives reduce the precision of the detector and thereby degrade the work efficiency of the cobot.

C. Deep learning approach

Deep neural networks enable powerful feature learning of high dimensional raw data, which could have not been achieved by hand-crafted feature engineering. In particular, recurrent neural networks (RNNs), such as long short-term memory networks (LSTMs), are frequently-used networks for processing sequential data, *e.g* sound, time series (sensor) data and natural language [14], [15]. However, a major drawback of standard RNNs or LSTMs is that they require longer inference and training time than feed-forward network such as convolutional neural networks (CNNs). This makes those networks unsuitable for real time applications that require low latency. Consequently, we adopted 1-dimensional CNN to process temporal joint signals of robot. A 1-D CNN can process time series data by using a time window to sequentially stack temporal signals. More details on the network architecture will be covered in Section III.

D. Performance metrics

Efficiency and safety are essential criteria for performance assessment of a collision detector. For work efficiency, the detector must not commit frequent false alarms, especially for industrial robots that are programmed to simply stop when a collision is detected. For safe human-robot interaction, detection delay must to be minimized [16]. The delay is defined as the sum of detection latency T_d and inference latency T_i. T_d refers to the time required for impact to propagate from contact point to joints over the robot inertia and T_i is the time required for the detector to detect the collision (Fig. 2b).

We will measure two quantities for the performance evaluation: 1) the number of false positives ($\sum FP$) that indicates efficiency, and 2) the number of false negatives ($\sum FN$) that indicates detection delay and safety. A FP occurs when the detector perceives a collision when no actual collision occurs,

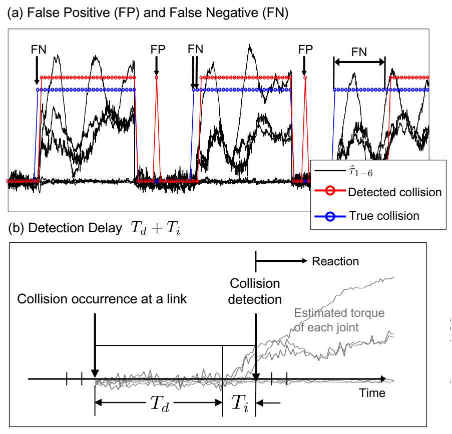

Fig. 2. Quantitative factors that affect collision detection performances. (a) False positive (FP) and false negative (FN): FPs reduce work efficiency of the robot and FNs cause detection delay (and reduce safety) (b) Detection delay: safe human-robot interaction requires minimization of the sum of detection latency T_d and inference latency T_i.

TABLE I

A SUMMARY OF AVAILABLE JOINT DATA

Description	Symbol
Joint position	q
Joint velocity	$\dot{q}$
Reference joint velocity	$\dot{q}_r$
Reference joint acceleration	$\ddot{q}_r$
Joint position error	e
Joint velocity error	$\dot{e}$
Control input	τ_m
Estimated external torque	$\hat{\tau}_{ext}$
Torque from inverse dynamic	τ_{inv}
Gravitational torque	τ_{grav}
Reference torque	τ_{ref}

and a FN occurs the detector ignores an actual collision (Fig. 2a). Efficiency is inversely related to the number of FPs. FNs occur as a result of detection delay, so the level of safety is inversely related to the number of FNs. A situation when $\sum FN > T_f$ where T_f is a certain value is defined as a detection-failure case [1].

III. CollisionNet: A deep learning approach for collision detection

A. Summary of available joint data

A 6-DOF robot manipulator provides rich information at 4 kHz, and we chose to acquire a total of 11 joint signals (Table. I). The available joint data can be categorized as 1) raw signals and control error obtained from proprioceptive sensors; q, $\dot{q}$, e, and $\dot{e}$, 2) nominal model signals $\tau_{inv} = \widehat{M}\ddot{q}_r + \widehat{C}\dot{q}_r$ and $\tau_{grav} = \hat{g}$ calculated in real time, and 3) command input signals τ_m, τ_{ref}, $\ddot{q}_r$, and q_r.

[1]The evaluation metrics did not consider the failure cases because many factors can induce them. Various contact forces can be applied to various position on robots (e.g., very weak contact at a base link), so statistical evaluation of all cases is a difficult task. We therefore evaluated and compared detection performance only when collision detection succeeded.

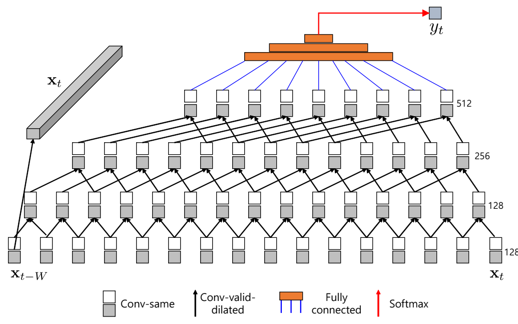

Fig. 3. 1-D CNN that combines convolutions with *valid* and *same* padding with dilation. Input of the network is a sequence of joint signals ($\mathbf{x}_t$), and output is a collision detection signal (y_t).

B. Network architecture

A *discriminative model* is used to output collision-detection result y at time t by using temporal joint signals

$$\mathbf{x}_t = \begin{bmatrix} \mathbf{q}_t^T & \dot{\mathbf{q}}_t^T & \cdots & \dot{\mathbf{e}}_t^T & \cdots & \boldsymbol{\tau}_{\text{ref},t}^T \end{bmatrix}^T \in \mathbb{R}^D, \quad (8)$$

by modeling the conditional probability distribution

$$p(y_t|\mathbf{X}_t) = p(y_t|\mathbf{x}_{t-W+1}, \cdots, \mathbf{x}_{t-1}, \mathbf{x}_t), \quad (9)$$

where $\mathbf{X}_t \in \mathbb{R}^{W \times D}$ and W is the size of a time window.

Each vector element of (8) contains the signals of all six joints presented in Table I. The dimension of each element is 6 and the number of elements is 11, so we conclude that $D = 6 \times 11 = 66$. For the collision detection, $y_t \in \{\text{True}, \text{False}\}$ is binary output. This approach can remove the heuristic and sensitive characteristics of the threshold-decision process. The goal of the proposed network is to model the posterior probability (9) by using *supervised learning*.

The proposed network is a modified version of autoregressive networks (Fig. 3). We translated the autoregressive networks, such as PixelCNN, ByteNet, and WaveNet [17]–[19], that belong to *generative model* into the *discriminative model*. The network takes stacked joint signals $\mathbf{X}_t$ as input, and outputs scalar value y_t. The network has no pooling layer and utilizes *valid* padding of convolution that reduces the feature map length [2]. After the final feature map passes through three fully-connected (fc) layers, y_t is obtained. The *valid* convolution works as *causal* convolution that does not violate causality. To increase the length of receptive fields, we used dilated convolution [19], [20]. The main difference from the autoregressive model is that the proposed network utilizes softmax (or sigmoid) as the final classification layer to directly output the collision detection signal. Convolutions followed by the *valid* padding convolutions use *same* padding to increase the number of weights and learning capacity. Filter length (of all 1-D convolutions) is 3 and filter channel increases from 128 to 512 like VGG-16 structure.

[2]Here, *same* padding makes output of the convolution to have same spatial resolution as input by appropriate zero-padding. In contrast, *valid* padding does not use padding during convolution operation.

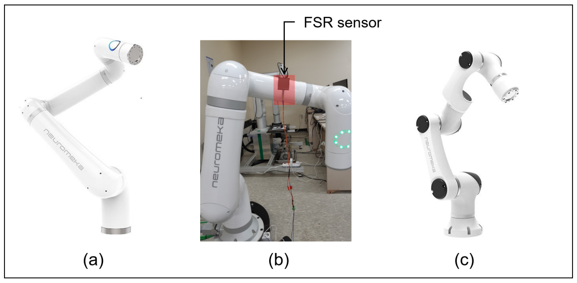

Fig. 4. (a) *Indy-7* is a 6-DOF cobot (Neuromeka). (b) For the collision data acquisition, a flexible analog potential meter, *i.e* Force Sensitive Resistor (FSR), was attached to the robot. Using the FSR, exact collision timing can be labelled. (c) *Indy-5* is also a 6-DOF robot platform and is used to validate robot generalization performance of the proposed collision detection method.

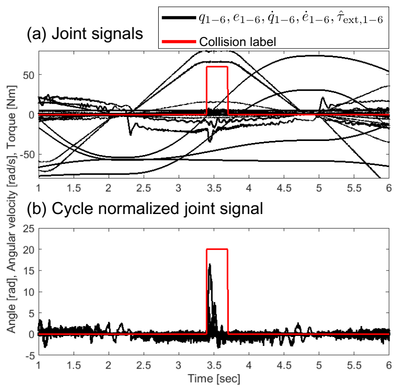

Fig. 5. All joint signals are subject to real time cycle normalization for the motion generalization. When collisions occur (red), the raw signal only shows a slight magnitude change (a), but the cycle normalized signal shows a noticeable magnitude standing high above zero (b).

C. Generalization of collision detection model

Robots should be able to perform various motions in their workspace. Joint signal patterns vary depending on the movements, so it is almost impossible for data-driven learning approach to train detector with every possible motion. Therefore, we applied a real-time signal-normalization method, which is called *cycle normalization*, to generalize performance of CollisionNet.

Cobots usually perform repetitive motions, which are taught by human workers. We can assume that a robot repeated the same trajectory tracking during *normal mode operation* before the collision occurred [9]. The *cycle normalization* method compares the present signal window with a reference-signal window from the reference cycle (Fig. 5). This method stores a whole reference cycle signal (usually the first cycle) in advance, and subtracts the present signal from the reference signals in real time:

$$\overline{\mathbf{X}}_t = \mathbf{X}_t^{\text{ref}} - \mathbf{X}_t, \tag{10}$$

where $\overline{\mathbf{X}}_t$ is the normalized signal that will be used as training examples or inference input of CollisionNet, and $\mathbf{X}_t^{\text{ref}}$ is a reference signal at t in the reference cycle buffer. The time index is used to synchronize two signals.

All robot signals are subject to *cycle normalization*, and process can normalize and generalize both robot motion and robot platform. Cycle normalization can be performed in real time, and therefore can be applied to both training and inference steps.

IV. Experimental setup

A. Robot platforms: Indy series

Experiments were performed using a 6-DOF robot manipulator (Neuromeka, *Indy-7*: 7-kg payload, 28-kg weight, 28-μm repeatability) (Fig. 4a). To validate robot generalization capability (Section V-B), we used another 6-DOF manipulator (Neuromeka, *Indy-5*: 5-kg payload, 25-kg weight, 25-μm repeatability) (Fig. 4c).

B. Data acquisition

The data for joint signals and true collision labels were collected from the robot as it performed repetitive motion cycles. First, we made the robot follow an arbitrary trajectory for 10 cycles; we set three way-points by direct teaching. The controller interpolated a path and performed trajectory tracking. While the robot was running, joint data such as positions, velocities, and estimated external torques were collected in real time at 4 kHz. Then a human operator intentionally generated a collision at an arbitrary time and location by hitting the robot. During the strike, collision labels were obtained by an FSR sensor (Fig. 4b). Although the FSR is an analog sensor, it was just used as an on/off switch to log the time of collision.

C. Training method

The total of 160,055 samples (16.46GB) composed of 210 collisions were used for the supervised learning. The joint data were used as input to the network, and the collision data were used as the ground truth label, which is a binary variable that indicates whether or not a collision had happened. We trained the CNN with some changes in hyperparameters to achieve high accuracy. For example, we studied the effects of dropout layers and batch normalization (bn) layers. The training results showed that use of bn layers apparently increases accuracy and reduces detection latency. However, the dropout layers did not yield any meaningful improvement.

D. Real-time inference

The real-time collision detection system was implemented on *Indy-7*. Although *Indy-7* performs position control and external torque estimation at 4 kHz, the proposed collision detector monitors collision signals at 1 kHz. To realize throughput at 1 kHz (inference latency $<$ 1 ms) of the trained CNN that has 16 million weights, we optimized the trained network as an inference engine by using TensorRT (NVIDIA

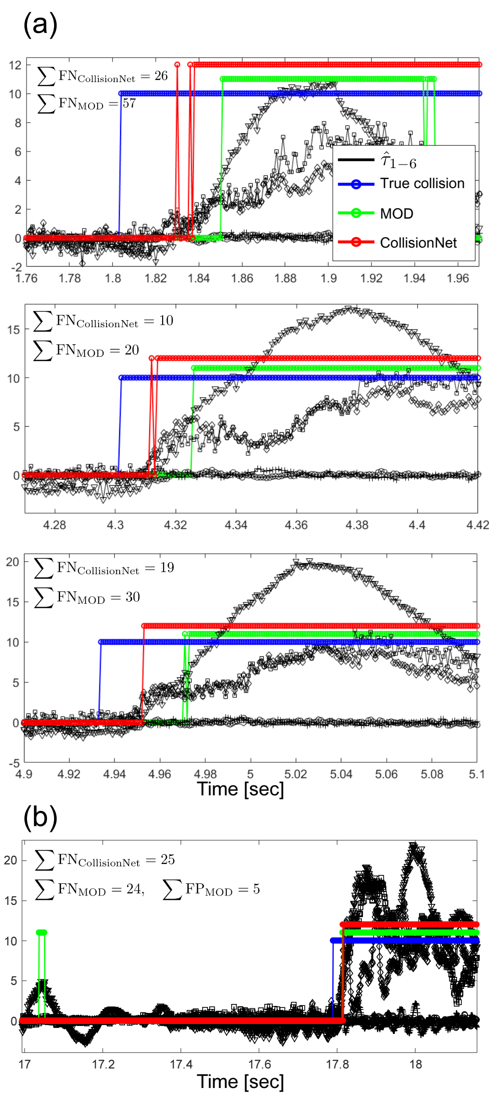

Fig. 6. (a) Three different external forces were exerted to robot links. The proposed detector (CollisionNet) recognized all cases with high confidence in both weak and strong collisions. Momentum-based observer detector (MOD) is compared with CollisionNet. (b) Detection accuracy of MOD is highly sensitive to thresholds, so a sensitive threshold induces FPs.

Corporation) to freeze trainable variables as constant parameters. By optimizing the network, inference latency was reduced to approximately 0.8 ms.

V. Experiments

We performed experiments to quantitatively evaluate the proposed collision detection method based on deep learning. First, we compared detection accuracy of the proposed method to that of the standard method that uses a momentum observer with decision thresholds. Second, we verified the generalization capability of the method by performing experiments in which we applied the trained network to other motions and robots that were not considered during the training process. Similar to the training data acquisition (Section IV-B), we attached the FSR sensor to an arbitrary robot link and randomly struck the attached position during robot operation. In all experiments, the robot just detected the collision and did not react to it; but, Multimedia video shows a stop reaction.

TABLE II
Statistic comparison of detection delay

Methods	**Average detection delay of 20 times**
MOD	30.0
CollisionNet	18.4

A. *Comparison with standard method*

For validation, we compared the accuracy of the proposed method (CollisionNet) to that of a momentum-based observer detector (MOD), which just provides the estimated external torque, not the collision-detection signal. The detection accuracy of MOD depends strongly on the decision thresholds, so the decision process requires tedious threshold tuning. If the threshold is set to be sensitive to collision during a specific motion, numerous FPs can be generated in different motions. In contrast, if the threshold is too insensitive, detection delay increases and even detection failure may arise. Therefore, the comparison with the standard method to the proposed end-to-end framework is difficult because CollisionNet does not require a threshold-tuning process. To ensure a fair comparison we tried to tune thresholds to be the most sensitive while ensuring that FPs do not occur.

During demonstrations, CollisionNet was very sensitive to collisions from weak contact to strong impact (please refer to the Multimedia video). During the quantitative experiments, CollisionNet achieved zero FP and smaller detection delay than MOD in all collisions (Fig. 6 a, Table II). If we changed the thresholds of the MOD to have a detection delay similar to that of the CollisionNet, then MOD generated FPs (Fig. 6 b). We conclude that the proposed method can achieve both high sensitivity to collision and low sensitivity to false alarms. This implies that the proposed deep learning approach can automatically learn an optimal decision strategy; this ability enables elimination of a tedious threshold-tuning process.

B. *Generalization capability*

For the proposed learning-based algorithm, generalization capability is one of the most critical requirements for it to be used in the real world. We experimentally verified the generalization capability of CollisionNet for different motions and in different robots. Note that the proposed network was trained on the data obtained from *Indy-7* with two specific motions (Section IV-C).

1) Motion generalization: The trained network was tested on unseen motions of *Indy-7* to validate the motion generalization capability (Fig. 7). The signal patterns differed among distinct motions and the contact forces between every pair of collisions. Despite these differences, CollisionNet successfully detected all collisions without FPs and detection failures.

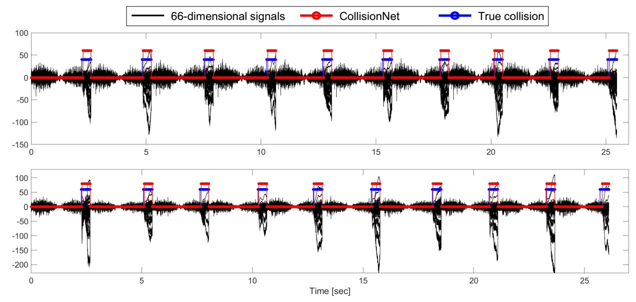

Fig. 7. Experimental results of two tests (top, bottom) to validate the motion-generalization capability of CollisionNet. Black lines: cycle-normalized 66-dimensional joint signals, where the peak magnitudes of the joint signals are proportional to the contact force at that time; red lines: detection results of CollisionNet; blue lines: actual collisions. Even though the test motion is apparently unseen, the proposed detector preserves its precise and reliable collision-detection accuracy.

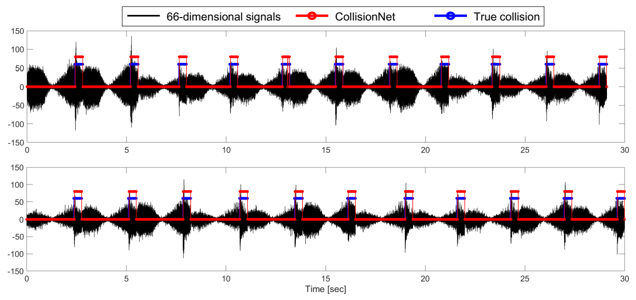

Fig. 8. Experimental results for validating robot generalization capability of CollisionNet. Test experiments were performed with *Indy-5*. Black lines: cycle-normalized 66-dimensional joint signals, where the peak magnitudes of the joint signals are proportional to the contact force at that time; red lines: detection results of CollisionNet; blue lines: actual collisions. Both robot and motion were not seen during the training process, and signals of *Indy-5* are less stable than those of *Indy-7*. CollisionNet successfully preserved its detection accuracy.

2) Robot generalization: To verify the robot and motion generalization ability, the trained network was tested with two unseen motions on a different robot platform *Indy-5* (Fig. 8 top and bottom). Two motions and various contact forces (black lines) are clearly different from those in Fig. 7. The joint signals of *Indy-5* are less stable than those of *Indy-7*. Nonetheless, detection accuracy was preserved due to the generalization capability of CollisionNet. The results achieved zero FP and detection failure.

The experiment verified that the combination of the real-time cycle normalization of joint signals and feature extraction with deep learning enables both high detection accuracy and powerful generalization ability.

VI. Discussion and Conclusion

We have demonstrated the design and implementation of a novel collision detection framework. To the best of our knowledge, this is the first implementation of a real-time robot collision detector that uses a DNN. The proposed approach eliminated tedious heuristic decision processes. A model-based observer can adequately estimate external joint torque, but the process is vulnerable to uncertain model and noisy signal. In contrast, a DNN is very insensitive to model uncertainties and sensor noises. Furthermore, it can automatically extract features from high-dimensional sequential joint signals. These features enable the proposed CNN-based method to achieve both high sensitivity to collisions and low susceptibility to false alarms. In quantitative evaluations, the proposed collision detector has reliable accuracy and wide applicability to general industrial robots.

For future work, we will try to apply the collision-detection framework to cobots that do not run in cycles. Our cycle-normalization method would not hold in this case, so we would need to find alternative solutions to achieve generalization capability in non-cyclic cases.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2011-0030075) and the Technology development Program(S2633695) funded by the Ministry of SMEs and Startups(MSS, Korea).

References

[1] A. Vysocky and P. Novak, “Human-robot collaboration in industry,” *MM Science Journal*, vol. 9, no. 2, pp. 903–906, 2016.

[2] A. M. Djuric, R. Urbanic, and J. Rickli, “A framework for collaborative robot (cobot) integration in advanced manufacturing systems,” *SAE International Journal of Materials and Manufacturing*, vol. 9, no. 2, pp. 457–464, 2016.

[3] M. Bortolini, E. Ferrari, M. Gamberi, F. Pilati, and M. Faccio, “Assembly system design in the industry 4.0 era: a general framework,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 5700–5705, 2017.

[4] J. Vorndamme, M. Schappler, and S. Haddadin, “Collision detection, isolation and identification for humanoids,” in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 4754–4761.

[5] S. Haddadin, A. De Luca, and A. Albu-Schäffer, “Robot collisions: A survey on detection, isolation, and identification,” *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, 2017.

[6] J. Ulmen and M. Cutkosky, “A robust, low-cost and low-noise artificial skin for human-friendly robots,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 4836–4841.

[7] R. S. Dahiya, P. Mittendorfer, M. Valle, G. Cheng, and V. J. Lumelsky, “Directions toward effective utilization of tactile skin: A review,” *IEEE Sensors Journal*, vol. 13, no. 11, pp. 4121–4138, 2013.

[8] E. Cagatay, A. Abdellah, P. Lugli, P. Mittendorfer, and G. Cheng, “Integrating cnt force sensors into a multimodal modular electronic skin,” in *Nanotechnology (IEEE-NANO), 2015 IEEE 15th International Conference on*. IEEE, 2015, pp. 1299–1302.

[9] S. Takakura, T. Murakami, and K. Ohnishi, “An approach to collision detection and recovery motion in industrial robot,” in *Industrial Electronics Society, 1989. IECON'89., 15th Annual Conference of IEEE*. IEEE, 1989, pp. 421–426.

[10] A. De Luca and R. Mattone, “Sensorless robot collision detection and hybrid force/motion control,” in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 999–1004.

[11] A. De Luca, A. Albu-Schaffer, S. Haddadin, and G. Hirzinger, “Collision detection and safe reaction with the dlr-iii lightweight manipulator arm,” in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 1623–1630.

[12] V. Sotoudehnejad, A. Takhmar, M. R. Kermani, and I. G. Polushin, “Counteracting modeling errors for sensitive observer-based manipulator collision detection,” in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 4315–4320.

[13] J. Park and W. K. Chung, “Analytic nonlinear h/sub/spl infin//inverse-optimal control for euler-lagrange system,” *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 847–854, 2000.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[16] S. Haddadin, A. Albu-Schäffer, and G. Hirzinger, "Requirements for safe robots: Measurements, analysis and new insights," *The International Journal of Robotics Research*, vol. 28, no. 11-12, pp. 1507–1527, 2009.

[17] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," *arXiv preprint arXiv:1601.06759*, 2016.

[18] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. v. d. Oord, A. Graves, and K. Kavukcuoglu, "Neural machine translation in linear time," *arXiv preprint arXiv:1610.10099*, 2016.

[19] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[20] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.