Transferable Collision Detection Learning for Collaborative Manipulator Using Versatile Modularized Neural Network

Donghyeon Kim, Daegyu Lim, and Jaeheung Park, *Member, IEEE*

***Abstract*—As human-robot collaboration increases and robots are applied to everyday life, interest in safety issues is increasing. To be safely used in real life, in addition to collision prevention algorithms, robots need to quickly detect unexpected collisions and take appropriate actions. Recently, deep learning-based collision detection algorithms have been proposed to overcome the limitations of model-based collision detection methods, but there are also limitations on deep learning methods, especially in data collection. The collected data are often insufficient because collecting collision data is laborious and intrinsically imbalanced, meaning that the collision data are much smaller than the free-motion data. Moreover, since collecting collision data is risky and might cause potential damage to the robot, applying the deep learning method to a new target robot on mass production is highly restricted. Therefore, in this article, an inductive bias is imposed on network structure and input variable to be sample-efficiently trained, which is suitable for insufficient imbalanced data. The proposed modularized neural network removes the connection between other joints at the front part of the network, reducing the search space of the learnable parameter. Moreover, the input variable is selected to take both dynamics features and error-related features into account. Consequently, the proposed method is versatile, showing successful generalization performance to random motion, random collision location, and various loads at the end-effector. Furthermore, to circumvent the limitations of applying deep learning methods to mass production, transfer learning is proposed, which does not require any collision data from the target robot. The proposed data-mixture method and collision ratio adjustment method for fine-tuning are validated with two source robots and one target robot. The transferred network is effective to be applied on mass production without losing performance compared to a specific-robot-trained network(a network trained with specific robot data and applied to the same robot).**

***Index Terms*—Collaborative manipulator, collision detection, deep learning, mass production, transfer learning.**

Manuscript received June 25, 2021; accepted November 17, 2021. This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government (MSIT) (No.2021R1A2C3005914). This paper was recommended for publication by Associate Editor Jens Kober and Editor Wolfram Burgard upon evaluation of the reviewers' comments. *(Donghyeon Kim and Daegyu Lim contributed equally to this work.) (Corresponding author: Jaeheung Park.)*

Donghyeon Kim and Daegyu Lim are with the Department of Intelligence and Information, and Automation and Systems Research Institute, Seoul National University, Seoul KS002, South Korea (e-mail: kdh0429@snu.ac.kr; dgyo3784@snu.ac.kr).

Jaeheung Park is with the Department of Intelligence and Information, and Automation and Systems Research Institute, Seoul National University, Seoul KS002, South Korea, and also with the Advanced Institutes of Convergence Technology, Suwon 443-270, South Korea (e-mail: park73@snu.ac.kr).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TRO.2021.3129630.

Digital Object Identifier 10.1109/TRO.2021.3129630

NOMENCLATURE

θ	The joint positions.
$\dot{\theta}$	The joint velocities.
$\ddot{\theta}$	The joint accelerations.
θ_d	The joint reference positions.
$\dot{\theta}_d$	The joint reference velocities.
θ_{err}	The joint position errors.
$\dot{\theta}_{\text{err}}$	The joint velocity errors.
i_m	The motor currents.
k_m	The motor constants.
τ_m	The motor torques.
τ_f	The friction torques.
τ_{ext}	The external torques.
τ_{dist}	The disturbance torques.
τ_{MOB}	The momentum observer torques.
MOB	Momentum observer.
CNN	Convolution neural network.
1-D CNN	One-dimensional CNN.
LSTM	Long short-term memory.
FC	Fully connected neural network.
MNN	Modularized neural network.
BCEL	Binary cross-entropy loss.
WCEL	Weighted cross-entropy loss.
DF	Detection failure.
DD	Detection delay.
FP	False positive.
FPn	Number of false positives.
DoF	Degrees of freedom.
CF	Continuous filter.
$\mathcal{X}$	Input feature space.
$\mathcal{Y}$	Label space.
$\mathcal{T}$	Learning task.
$P(X)$	Marginal distribution.
$P(Y\|X)$	Conditional distribution.
$\mathcal{D}$	Domain.
DS	Data set.
Subscript S	Source.
Subscript T	Target.

I. INTRODUCTION

RECENTLY, as robots are gradually applied to daily life and industrial sites, research on collaborative robots interacting with people has been actively conducted. These collaborative

robots require precise control to perform desired tasks; on the other hand, developers must be aware of and prepare for safety issues because there could be unexpected collisions with people or environments within the robot workspace [1], [2].

In order to ensure safety, avoiding unintentional collisions in the first place using external sensors, such as vision sensors, is desirable [3]–[5]. Flacco *et al.* [3] generated a repulsive vector at the end-effector and several control points to avoid collision using visual depth information. Mohammed *et al.* [4] demonstrated active collision avoidance strategies in an augmented environment, where virtual 3-D models of robots and real camera images of operators are used. Fan *et al.* [5] recently presented a decentralized sensor-level collision-avoidance controller for multirobot systems that directly maps raw 2-D laser measurements to steering commands of a mobile robot using reinforcement learning. However, these methods exploit information about the nearby environment from additional sensors such as vision sensors. Accordingly, real-time vision data handling should be guaranteed for online application and any abrupt changes in the environment, such as human motion and sensor instability due to occlusion or light conditions, make such prevention unreliable.

Therefore, apart from the collision prevention algorithm, Haddadin *et al.* [6] arranged a general collision handling pipeline to ensure safety. Among these steps, detection is the starting stage of the pipeline. Thus, immediate detection of collisions is directly related to safety. Various methods [7]–[15] have been proposed to quickly and accurately detect collisions, and these methods can be mainly classified into three categories: external sensor-based (artificial skin), model-based, and data-driven methods.

Artificial skin made of polyimide films with electrically conductive ink and a pressure-sensitive conductive rubber sheet was developed for safe human–robot interaction [7]. These skin-based collision detection methods have advantages in that they can also identify collision location and impact force; however, covering an entire robot with artificial skin is expensive, and the skin is difficult to manufacture. Thus, artificial skin is typically attached to a limited part of the robot's surface.

Model-based collision detection algorithms utilize observers that estimate external torque. Specifically, velocity observer [16] or momentum observer (MOB) [8]–[10] have been actively studied to exploit the dynamics model. The widely used MOB method has the advantage of estimating the external torque with proprioceptive sensors; however, this external torque includes the torque from the collision and the torque due to modeling errors and friction. Thus, it is essential to identify the system model accurately and formulate the friction model in advance [9], [10], which is challenging to model due to the nonlinearity of friction.

Meanwhile, deep learning based fault detection is actively being studied with recent advances in deep learning and its ability to represent nonlinearity [17]–[20]. Verstraete *et al.* [17] used a convolution neural network (CNN) to extract features from linear time-frequency transformed images and detected fault on bearings. Additionally, Karim *et al.* [19] suggested using features extracted from long short-term memory (LSTM) network and fully convolution networks to handle time-series data. In robot collision detection application, there has been a multistep approach that regresses a residual [11], [13], [15], [21] and an end-to-end approach [14], [22] that directly estimates a binary state as to whether the robot is in a collision or not. Generally, the end-to-end method, trained with free-motion and collision data, shows superior performance than the residual-based method, which is trained with only free-motion data and requires individual thresholds for each joint. However, when implementing the end-to-end method, the collected data are intrinsically imbalanced dataset which means that the collision data are much smaller than the free-motion data (in our case, the rate of collision data was 5% of the overall data) [23], [24]. It is because the free-motion consists mostly of motion, while the collision occurs briefly. Moreover, collecting collision data is laborious, making it hard to collect abundant and varying collision datasets. Thus, without a sufficiently large training set, the collision features would especially lack; therefore, a classifier may not generalize the characteristics of the data. Furthermore, the classifier could also overfit the training data, with a poor performance in out-of-sample test instances as mentioned in [25]. Thus, Heo *et al.* [14] restricted their collision detection algorithm to a predetermined and preperformed cyclic motion. Since the cycle-normalization they used requires the robot to carry out the cyclic motion in advance, this approach is largely limited to the preperformed motion and has difficulty adapting to general motions that have not been carried out in advance.

However, one of the critical characteristics that the trained model should exhibit is versatility; it would be desirable if the trained model detects collisions no matter in which part the collision occurs while executing random motion and generalizes various loads at the end-effector. We impose inductive bias when designing network structure and selecting input variables to estimate collision state successfully with insufficient and unbalanced data. Imposing inductive bias means employing any mechanism whereby the space of hypotheses is restricted or whereby some hypotheses are preferred *a priori* over others [26]. Introducing adequate inductive bias has the potential of improving performance by a large margin. For example, as in [27], using demonstrations in imitation learning or reinforcement learning restricts the large search space and guides the training procedure to the fast path. Also, the representative CNN is an inductively biased structure that hypothesizes that the nearby local pixels are related so that the features can be extracted from the relation among local pixels. Although recent studies show that less inductive bias can be employed as sufficiently large image data are getting available [28], this is not the case in the robotics field, especially when collision data should be collected manually. In robot collision detection, Sharkawy *et al.* [15] proposed to use a fully connected neural network to account for the coupling effect among joints. However, since a fully connected neural network assumes no inductive bias in the structure implicitly, there might be difficulties extending their three degrees of freedom (DoF) demonstration to higher DoF collaborative manipulator and various loads although the same authors exhibited promising generalization performance on regression problems. The authors implemented variable admittance control by tuning the coefficient of virtual inertia using

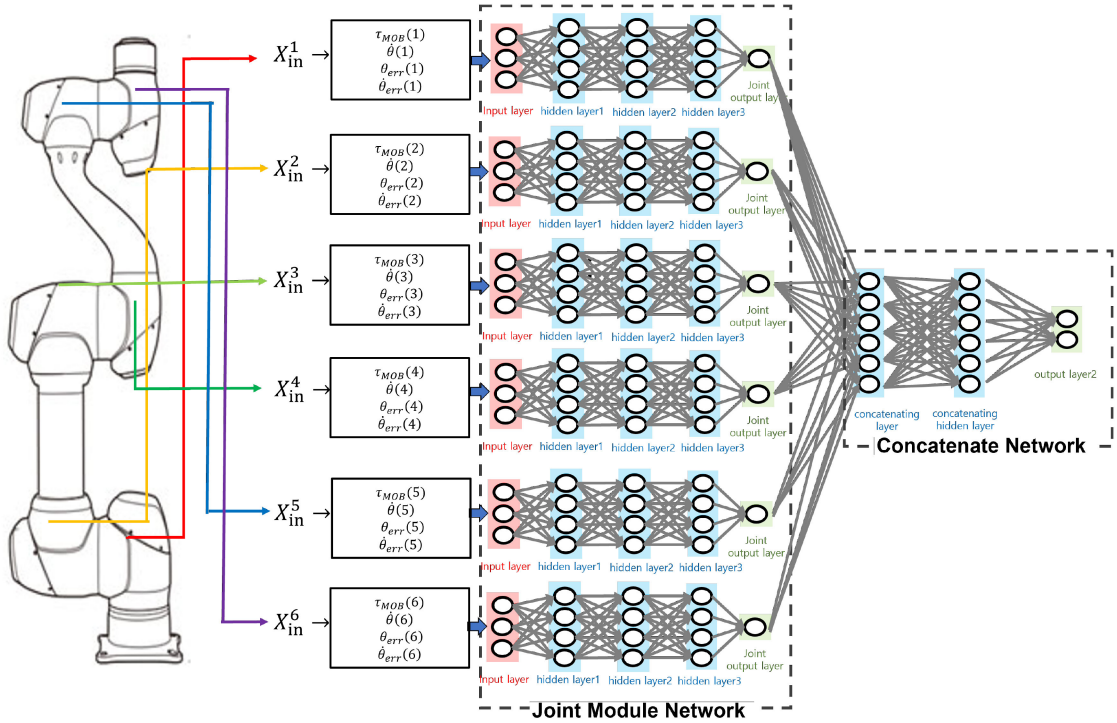


Fig. 1. Structure of modularized neural network. Inputs are modularized and fed to the network. The red layers are input layers that modularized inputs are fed to. The blue layers are hidden layers and each modular network outputs a scalar value (green). These values are concatenated and fed to concatenate network with two hidden layers (blue) which outputs two final prediction scores.

a neural network on the KUKA LWR manipulator [29]. Meanwhile, Park *et al.* [22] extended the learning-based collision detection method to the 6-DoF manipulator using 1-D CNN with the residual of MOB as input. However, it is inspected from our experiments in Section III-D that the input is not optimal, resulting in frequent false positives (FPs) that the trained model infers a collision occurred when it did not. Thus, the FPs were handled by using a continuous filter (CF) in their study.

Our proposed modularized neural network (MNN) consists of joint module network and concatenate network, as shown in Fig. 1. Joint module network decouples each joint and thereby removes connection between other joints reducing the parameter search space compared to the fully connected network, and only relevant local joint inputs are fed to each modular network leading to sample-efficient training suitable for insufficient imbalanced data. However, since there are coupling effects between joints, the features extracted from the joint module network are concatenated and fed into a small concatenate network to account for coupling effects, and the concatenate network classifies whether a collision occurred or not. Also, it is shown that selecting proper input variables that fit into the dynamics and considering error-related terms, rather than raw sensor data or insufficient residual of MOB, plays a vital role in improving performance. By imposing appropriate inductive bias on the network structure and input variable, the following advantages are achieved.

1) Sample efficiency: Requires relatively less training data, which is beneficial for imbalanced and hard-to-collect collision scenarios.
2) Versatility: Generalization to random motion, random collision location, and various loads at the end-effector on 6-DoF manipulator.
3) Light: Able to run in real time on a commercial robot-embedded computer.
4) Performance improvement: Detection sensitivity is improved and FPs are decreased by a large margin compared to the conventional model-based and recent data-driven methods.

Even though the proposed network enhances performance and requires fewer collision data, the end-to-end learning method still needs collision data, which is inappropriate for mass production; companies that manufacture actual products in large quantities cannot impact every new robot because it is laborious and might cause potential damage to the robot. It has already been reported in various industrial fields that collecting training data in mass production is costly, demanding, or impossible [30], [31]. In order to resolve this problem, Wu *et al.* [30] introduced a transfer learning method for multimode chemical processes. Mao *et al.* [31] also used fine-tuning method, a kind of transfer learning, employing a pretrained network from the source domain and data from the target domain for bearing fault detection. But, in the field of robot collision detection, the study considering constraints on mass production is rarely conducted. Therefore, considering these costs and constraints, the learning process, including data collection and training methodology, should be designed carefully. In this article, a fine-tuning method for mass production is introduced. We formulated a transfer learning method between the same model but different robots.

The most challenging part of transfer learning is to avoid collecting new collision data from the target robot. As a result, the data mixture method is used to resolve this problem; the collision data from source robots are mixed with the free-motion data from the target robot for fine-tuning on the target robot. However, this data mixture method has limitations in that the fine-tuned network using the data mixture method shows drifted detection sensitivity compared to the sensitivity of the network trained only for the source robot, which is regarded as an optimum. This drifted sensitivity has occurred because virtual collision data from source robots are used instead of collision data from the target robot, although the mechanical responsiveness for the collision is different between robots even if the robots are equally designed. The ratio of the collision data in the fine-tuning data is adjusted to handle this drifted sensitivity. Finally, the collision network is transferred to the target robot without losing performance while only using free motion data from the target robot. Quantitative experiments are conducted to validate our hypotheses, and the performance of collision detection is represented.

A. Main Contribution

Our proposed end-to-end collision detection algorithm using a *modularized neural network* and selected input variables is an inductively biased mechanism to handle insufficient and intrinsically imbalanced collision data effectively. It is designed to be sample-efficiently trained, demonstrating comparable performance to conventional methods even with fewer data. Also, its versatility is quantitatively shown by detecting collisions at random links that occur while the robot is performing random motions with various loads. Furthermore, the proposed MNN is light by exploiting the decoupled dynamics for each joint of the robot, enabling real-time computation (33 μs) employing an Intel N4200 with four CPU cores and a clock between 1.1 and 2.5 GHz. It is shown that the detection sensitivity and FP performance improved compared to the conventional model-based and recent data-driven methods, which use only a joint encoder and motor current sensor.

Additionally, a transfer learning method that considers constraints in mass production is proposed. The proposed pretraining process followed by fine-tuning with the data mixture method enables the deep learning based method for practical application in mass production; less than an hour of free-motion data without collision data from a new robot is required to fine-tune the network due to our compact network structure. Furthermore, the drifted sensitivity of the collision detection due to the difference between source robots and the target robot is corrected by adjusting the ratio of the collision data. To the best of our knowledge, it is the first study to apply transfer learning successfully in the collision detection of the robot without losing performance compared to a specific-robot-trained network (a network trained with specific robot data and applied to the same robot), which can extend the usability of the data-driven method to the industrial field.

B. Organization of Article

The rest of this article is organized as follows. Section II presents the proposed collision detection algorithm using deep learning. It starts from problem formulation and contains data collection methods, proposed network structure, and training techniques. In Section III, the effectiveness of the proposed structure and input variable selection is demonstrated in terms of sensitivity and FP rate by comparing with the existing methods. Additionally, it is shown that the trained model requires less data, which is helpful for imbalanced and hard-to-collect collision data. In Sections IV and V, the algorithm is extended for mass production by transfer learning. Section IV describes the proposed transfer learning methods for mass production. The methods are validated by experiments with two source robots and one target robot in Section V. Finally, Section VI conclludes this article.

II. Collision Detection Algorithm Using Modularized Network

A. Problem Formulation and Input Variable Selection

1) External Torque Estimation: Manipulator dynamics with friction and external torque can be represented as

$$M(\theta)\ddot{\theta} + b(\theta, \dot{\theta}) + g(\theta) = \tau_m + \tau_f + \tau_{\text{ext}} \tag{1}$$

where $M(\theta)$ is the inertia matrix, $b(\theta, \dot{\theta})$ is the centrifugal/Coriolis torque, $g(\theta)$ is the gravity torque, τ_m is the motor torque, τ_f is the friction torque, and τ_{ext} is the external torque from the collision. The motor torque can be estimated from the measured current sensor data, which are available in most robots, using $\tau_m = k_m i_m$, where k_m is the motor constant and i_m is the motor current. Then the disturbance torque τ_{dist}, defined as the sum of external torque and friction torque ($\tau_{\text{dist}} = \tau_{\text{ext}} + \tau_f$), can be calculated as follows, which is a basic disturbance observer

$$\tau_{\text{dist}} = \tau_{\text{ext}} + \tau_f = M(\theta)\ddot{\theta} + b(\theta, \dot{\theta}) + g(\theta) - k_m i_m. \tag{2}$$

In order to differentiate the external torque generated by the collision from the disturbance torque, the friction torque should be modeled and subtracted from the disturbance torque. In this article, it is assumed that the friction is a nonlinear function of $\dot{\theta}$. In this case, (1) can be organized with respect to the external torque as follows:

$$\tau_{\text{ext}} = \tau_{\text{dist}} - \tau_f(\dot{\theta}) \tag{3}$$

$$= M(\theta)\ddot{\theta} + b(\theta, \dot{\theta}) + g(\theta) - k_m i_m - \tau_f(\dot{\theta}). \tag{4}$$

Therefore, since the external torque τ_{ext} is a function of $k_m i_m, \ddot{\theta}, \dot{\theta}$, and θ in (4), these variables are the minimum essential input variables to estimate external torque and detect collision.

2) Momentum Observer: In order to calculate τ_{dist} in (2), the joint acceleration $\ddot{\theta}$ is required. However, without a high-resolution encoder, the simple second derivative of a joint position θ is noisy to be used. An MOB, a representative disturbance observer using the generalized momentum p, was introduced

in [8] to circumvent the usage of the joint acceleration. The generalized momentum is defined as

$$p = M(\theta)\dot{\theta} \tag{5}$$

and its derivative combined with (2) becomes

$$\dot{p} = k_m i_m + \tau_{dist} + \dot{M}(\theta)\dot{\theta} - b(\theta,\dot{\theta}) - g(\theta) \tag{6}$$

$$= k_m i_m + \tau_{dist} - \beta(\theta,\dot{\theta}) \tag{7}$$

where $\beta(\theta,\dot{\theta}) = g(\theta) + b(\theta,\dot{\theta})\dot{\theta} - \dot{M}(\theta)\dot{\theta}$ is used for simplification.

A residual vector and its dynamics are defined as follows to estimate disturbance torque:

$$\dot{r} = K_0(\dot{p} - \dot{\hat{p}}) \tag{8}$$

$$\dot{\hat{p}} = \tau_m - \beta(\theta,\dot{\theta}) + r \tag{9}$$

where $K_0 = \text{diag}\{k_{0,i}\} > 0$ is the positive diagonal gain matrix and $\dot{\hat{p}}$ is the derivative of the estimated momentum. Integrating (8) results in

$$r = K_0\left\{p(t) - p(0) - \int_0^t (k_m i_m - \hat{\beta}(\theta,\dot{\theta}) + r)dt\right\}. \tag{10}$$

Substituting (7) and (9) for (8) derives the relation between r and τ_{dist} as

$$\dot{r} = K_0(\tau_{\text{dist}} - r). \tag{11}$$

In the Laplace domain, (11) can be expressed by component as

$$\frac{r_i(s)}{\tau_{\text{dist},i}(s)} = \frac{k_{0,i}}{s + k_{0,i}} \quad (i = 1, 2, \ldots, n) \tag{12}$$

which means that the MOB residual vector r is the first-order low-pass filtered disturbance torque τ_{dist}. Throughout this article, τ_{MOB} is used instead to denote the residual vector r for the unity of expression.

Finally, by using the low-pass filtered value τ_{MOB} instead of τ_{dist} in (3), the external torque can be approximately estimated by

$$\tau_{\text{ext}} \approx \tau_{\text{MOB}} - \tau_f(\dot{\theta}). \tag{13}$$

Therefore, once the τ_{MOB} is calculated, the minimum essential input variables are τ_{MOB} and $\dot{\theta}$. Again, note that the joint acceleration data is not required when obtaining τ_{MOB} contrary to (4).

3) Input Variable Candidates: In addition to the input variables derived from the dynamics equation as discussed in Section II-A2, it can be presumed that feeding inputs such as θ, θ_d, $\dot{\theta}$, and $\dot{\theta}_d$ can be helpful to estimate the collision state by learning error-related features. Therefore, based on the related input variables mentioned so far, the input variable candidates are listed in the following:

1) Ver 1: $k_m i_m, \theta, \dot{\theta}, \theta_d, \dot{\theta}_d$;
2) Ver 2: τ_{MOB};
3) Ver 3: $\tau_{\text{MOB}}, \dot{\theta}$;
4) Ver 4: $\tau_{\text{MOB}}, \dot{\theta}, \theta_{\text{err}}, \dot{\theta}_{\text{err}}$

where $\theta_{\text{err}} = \theta_d - \theta$ and $\dot{\theta}_{\text{err}} = \dot{\theta}_d - \dot{\theta}$.

First, Ver 1 is a classical deep learning based input; raw data without any inductive bias. Deep learning is a powerful tool to automatically map raw input data to output without hand-designed feature selection. The Ver 1 input is chosen to show how the absence of inductive bias affects training on insufficient imbalanced data. Ver 2 is an input variable selected by Park *et al.* [22]. MOB torque τ_{MOB}, a low-pass filtered disturbance torque as mentioned in Section II-A2, contains both the external and friction torques. Therefore, the external torque cannot be separated and it is insufficient for detecting collision effectively by using only the MOB torque. As a result, Ver 3 is designed to use joint velocity to account for the friction as an additional input variable from Ver 2. In conventional model-based methods, fitting coefficients of friction model are identified as a function of joint velocity and the modeled friction torque is subtracted from the MOB torque to estimate the external torque [10], [32]. However, as it is shown by previous deep learning approaches [33]–[35], it is expected that the neural network which takes the joint velocity as an input learns nonlinear friction automatically as opposed to the conventional friction model. Therefore, the additional joint velocity is selected to learn frictional effect and to estimate external torque internally. Lastly, Ver 4 is the inductively biased input consisting of variables that fit into the dynamics and reflect error-related factors. While inspecting data, it was shown that joint position and velocity error could be decent collision features whose distribution differs distinctively under free-motion and during the collision. The average of L1 norm of joint position error was [0.0035, 0.0014, 0.0018, 0.0019, 0.0020, 0.0024] rad for each joint under free-motion, whereas the average error norm was [0.0045, 0.0018, 0.0023, 0.0025, 0.0026, 0.0030] rad during collision, showing 20%–30% increased error. The joint velocity error showed similar tendency with [0.0051, 0.0029, 0.0035, 0.0043, 0.0040, 0.0042] rad/s of L1 norm under free motion and [0.0063, 0.0064, 0.0046, 0.0055, 0.0049, 0.0043] rad/s during collision, especially showing notable increase on the second joint with more than 100% increase. Performance according to the input variable is compared and discussed in Section III-B.

B. Network Structure

It is important to use collision data efficiently because the data collection procedure is expensive and laborious. Moreover, the dataset is imbalanced, which means that collision data are less than the free-motion data. This imbalanced characteristic of the dataset, along with insufficient data, further emphasizes the necessity of sample-efficient training. For example, when the collision instances make up 1% of the dataset, 1% of 1000 instances is only ten instances, whereas 1% out of 1 million instances includes 10 000 collision data. With 10 000 observations of the collision instances, the neural network has a higher probability of generalization to test samples. However, without sufficient data, which is in our case, there may not be enough information to effectively discriminate between the classes [36].

In this sense, the proposed MNN is designed to be sample-efficiently trained, which is suitable for insufficient imbalanced

collision data. The imposed inductive bias on the network structure design is that the collision features can be extracted per joint by using local joint variables, and then the collision state can be classified using all the collision features altogether, which was motivated by traditional model-based observers [8], [16]. Model-based collision detection methods estimate the collision state by first estimating the external torque of each joint and then classifying collision state based on the collision features of all joints (in this case, the estimated external torque values). Similarly, the network is designed to first decentralize each joint information and detect collision based on the features of all joints. The MNN consists of joint module network and concatenate network as shown in Fig. 1. The joint module network extracts the feature of each joint from input data and the concatenate network predicts the collision state based on the concatenated features.

1) Network Characteristics: The joint module network decouples each joint by feeding each joint input variable into each modular network. In this way, the joint module network removes connection among other joints, reducing the search space of the learnable parameter compared to the fully connected structure. While an exponentially increasing number of samples is required in order to map a given function over the model parameter space with sufficient confidence [37], each modular network requires less data than a coupled structure, as it only needs to extract each joint feature, leading to sample-efficient training. Also, by feeding only relevant inputs to each modular network, the adverse effects of using redundant variables can be alleviated. Redundant variables increase the number of local optima since there are more combinations of parameters that can yield locally optimal error values. Algorithms that are based on gradient descent are therefore more likely to converge to a local optimum resulting in poor generalization performance [38]. Also, irrelevant variables add noise into the model and the noise may mask the important input–output relationships, slowing the training process [38].

It is expected that each modular network extracts the feature of an individual joint, reflecting characteristics such as friction and error dynamics. The features extracted from the joint module network are concatenated and fed into a small concatenate network to take coupling effects into account and the concatenate network classifies whether a collision occurred or not. The superiority of performance and data efficiency of this network are further demonstrated in Sections III-A and III-C.

2) Network Structure Details: Each modular network in joint module network has three hidden layers and outputs a scalar value at the output layer. These scalar values are then concatenated and fed into a concatenate network, which has one hidden layer and outputs two final prediction scores between [0,1] for each $[collision, notcollision]$ class. Using one hidden layer for the concatenate network improves performance by considering the nonlinear effects like coupling between joints. ReLU was used as an activation function in the hidden layer and Softmax was used in the output layer to output [0,1] value for each class. Batch normalization was applied to all layers and each hidden layer has 15 hidden neurons; thus, overall, 4505 to 5855 learnable parameters, depending on the input variable selection, are used, requiring only 33 μs to compute on Intel N4200 with four CPU cores and clock between 1.1 and 2.5 GHz. The computation time is much faster than using the recently proposed collision detection network system [14], whose number of weights is 16 million, requiring 800 μs to infer on Intel Celeron Braswell soc with four CPU cores and clock of 1.6 GHz. Also, the computation time is effectively reduced by three orders of magnitude compared to the 1-D CNN structure from Park *et al.* [22] on the same robot-embedded CPU. Therefore, the proposed network can detect collisions in real time with a low computational cost.

Fig. 2. 6-DoF manipulator M0609 from Doosan Robotics used for experiments. Although it is equipped with both joint torque sensor and motor current sensor, only motor current sensor, which is mounted on a typical collaborative robot, is used in this article to be generally applicable.

C. Data Collection

Data were collected with three 6-DoF manipulators (unit 1, unit 2, and unit 3 of the same model); the Doosan Robotics M0609 is shown in Fig. 2, with a weight of 27 kg, rated payload of 6 kg, and a maximum reach of 900 mm. unit 1 is used for experiments that compare the performance according to the network structure and input variable on a single robot in Section III and units 1, 2, and 3 are used for transfer learning experiments where units 1 and 2 are utilized as source robots and unit 3 is utilized as the target robot to apply transfer learning in Section V. Available input data consist of joint space level data $\theta, \theta_d, \dot{\theta}, \dot{\theta}_d, k_m i_m$. Although the data are updated and accessible at 1000 Hz, they were collected at 100 Hz; data collected with 1000 Hz are closely correlated and only make the size of the dataset 10 times larger, causing longer training time without significant performance improvement. Data were collected using two scenarios: free-motion and collision scenarios. For each scenario, three different tools (0.0, 2.0, and 5.0 kg) were attached to the end-effector in the training data collection, as shown in Table I; 120 min of free-motion and 70 min of collision data were collected for each tool. At test time, an additional tool with a weight of 3.3 kg was used for verification with hidden data during training, as listed in Table II. Notably, the dynamics model is used when calculating τ_{MOB} and the center of mass and weight of each tool is reflected in the dynamics. The free-motion scenario was a fully random motion with all six joints executing a trajectory of acceleration–uniform–deceleration motion in the

TABLE I
TRAINING DATA SCENARIO AND COLLECTION TIME

Scenario	Tool weight	Unit 1	Unit 2	Unit 3
Free-motion	0.0 kg	120 min	120 min	25 min
	2.0 kg	120 min	120 min	25 min
	5.0 kg	120 min	120 min	25 min
Collision	0.0 kg	70 min	70 min	-
	2.0 kg	70 min	70 min	-
	5.0 kg	70 min	70 min	-

TABLE II
TESTING DATA SCENARIO AND COLLECTION TIME

Motion	Tool weight	Unit 1	Unit 2	Unit 3
Free-motion	0.0 kg	15 min	15 min	15 min
	2.0 kg	15 min	15 min	15 min
	3.3 kg	15 min	15 min	15 min
	5.0 kg	15 min	15 min	15 min
Collision motion	0.0 kg	5 min	5 min	5 min
	2.0 kg	5 min	5 min	5 min
	3.3 kg	5 min	5 min	5 min
	5.0 kg	5 min	5 min	5 min

Fig. 3. Collision tool used to collide and record collision label. A mechanical switch (gray) is attached to the handle.

joint space. A person collided at a random moment and location for every link (except link 1) with the collision tool shown in Fig. 3 in hand while executing random motion for the collision scenario without a collision reaction strategy. Collision data were collected to include diverse collisions, such as strong/weak collisions, slow/abrupt collisions, and collisions when the robot or collision tool is stationary/moving. For example, collision duration varied from 0.01 to 2.04 s while the robot moved within the maximum joint velocity of 2.087 rad/s. A mechanical switch is attached to the collision tool so that whenever a collision occurs, the binary robot collision state is recorded as 1 and used as the label in training data. Compared to [14], in which a force sensitive resistor is attached to a designated point of the manipulator, our collision tool has an advantage in that the collision location is not restricted to a designated point. The ratio of collision labels in the training data was 5%, including free-motion and collision data, which is an imbalanced dataset.

D. Input Preprocessing

All data are normalized to the values between [–1,1] with their maximum and minimum value in training data. Also, time-series data that contain current and four past items with sampling interval of $t_I = 0.01\,\text{s}$ are used in a time window of $t_W = (5-1) \times 0.01\text{s} = 0.04\,\text{s}$. Using time-series data improves the performance by reducing the impact of sensor noise and reflecting the delay occurred by the low-pass filter in the MOB. An optimal time step of five was determined by increasing the number of time steps until the performance improvement was meaningless compared to the calculation time increment. Finally, the input is shaped as

$$X_{\text{in}} = [X_{\text{in}}^1, X_{\text{in}}^2, X_{\text{in}}^3, X_{\text{in}}^4, X_{\text{in}}^5, X_{\text{in}}^6] \tag{14}$$

where X_{in}^i is data for the ith joint. For the ith joint data X_{in}^i, data are stacked in order of time

$$X_{\text{in}}^i = [x_{\text{in}}^i(k-40), x_{\text{in}}^i(k-30), x_{\text{in}}^i(k-20), x_{\text{in}}^i(k-10), x_{\text{in}}^i(k)] \tag{15}$$

and $x_{\text{in}}^i(k)$ contains a normalized input variable such as

$$x_{\text{in}}^i(k) = [\tau_{\text{MOB}}(k), \dot{\theta}(k), \theta_d(k) - \theta(k), \dot{\theta}_d(k) - \dot{\theta}(k)]. \tag{16}$$

Here, the interval between two consecutive discretized data (e.g., $x_{\text{in}}^i(k)$ and $x_{\text{in}}^i(k-1)$) is 0.001 s based on the data update rate as mentioned in Section II-C.

E. Training

Data collected from unit 1 in Table I are normalized and the whole data are randomly shuffled before training. Mini-batch data of size 1000 is selected and Adam optimizer is used with default moment ($\beta_1 = 0.9, \beta_2 = 0.999$) and epsilon ($\epsilon = 1e-08$) value of Tensorflow, and learning rate of $lr = 1e-5$. The training progresses to minimize the binary cross-entropy loss (BCEL) with 500 epochs, which takes approximately 20 h with an AMD Ryzen 7 2700X CPU. The BCEL is

$$\text{BCEL} = -(y * \log(\hat{y}) + (1-y) * \log(1-\hat{y})) \tag{17}$$

where y is the recorded collision label and $\hat{y}$ is the prediction of the network. The training time is not very important compared with having a versatile neural network that generally works under random motion, random collision location, and various loads at the end-effector in real-time (33μs). Moreover, the inference time is not affected by the training time.

III. TEST RESULTS OF TRAINED MODEL

This section demonstrates that the inductive bias on the network structure and input variable enables successful training on insufficient imbalanced data by experiments. Free-motion and collision tests that are similar to the training scenarios were conducted on the same robot, unit 1, to evaluate the performance of the proposed collision detection method. However, a tool of 3.3 kg was additionally included to show the generalization performance to the unseen data as presented in Table II. For each tool, about 60 random collisions occurred for 5 and 15 min of free-motion were conducted; overall, collision data with 245 collisions and an hour of free-motion data were collected.

In the collision detection test, detection delay (DD) and the number of detection failures (DF) are measured. The time difference between the moment when the switch attached to the

collision tool is pushed due to the collision and the moment when the network infers collision is defined as DD. The neural network infers a collision when the output of the network exceeds a threshold. Given that the network is sufficiently sensitive, which is to be established from the experiments, the threshold is set to 0.99 to reduce the FPs. The two indexes, DD and DF, are related to the detection sensitivity of the network. If DD and DF are small, the network is sensitive to external force and vice versa.

On the other hand, in the free-motion test, FP is analyzed by inspecting the number of FP occurrences (FPn). Once the output of the network exceeds the threshold during free-motion, FPn increases by one. Consecutive FPs in the time step are counted as one FPn because the robot stops in practice for safety reasons when the first FP occurs, although no collision reaction strategy was implemented in this study. Illustration for DD, DF, and FPn are shown in Fig. 5.

The MNN inference cycle is set to 1 ms, determined by evaluating the larger value between the network computational time (33μs) and the data update period (1 ms). Similar to Fig. 6, data up to 0.04 s should be stored in a buffer every 0.001 s to make an inference every 0.001 s and process the time-stacked input data as mentioned in Section II-D. All the results in this section are the averages of three independently trained models for rigorous results.

TABLE III
PERFORMANCE ACCORDING TO THE NETWORK STRUCTURE

		Collision (5 min/tool)		Free-motion (15 min/tool)
Tool Weight	Network Structure	DF	DD(ms)	FPn
0.0 kg	FC	8/62	40.5	80
	1D CNN	0/62	15.4	26
	MNN	0/62	15.75	13
2.0kg	FC	8/62	34.3	61
	1D CNN	1/62	15.4	14
	MNN	0/62	12.88	8
3.3kg	FC	11/60	38.5	64
	1D CNN	0/60	14.3	27
	MNN	0/60	12.39	9
5.0kg	FC	20/61	38.4	89
	1D CNN	2/61	17.7	60
	MNN	1/61	16.57	25
Total	FC	46/245	37.9	293
	1D CNN	2/245	15.7	126
	MNN	1/245	14.4	54

A. Performance According to the Network Structure

First, to show the effect of inductive bias in the design of the network structure, the performance of MNN is compared to other network structures: Fully connected neural network (FC) and 1-D CNN. In this section, input Ver 1, an input without inductive bias, is used across all network structures to demonstrate the impact of structure alone.

A fully connected network assumes no inductive bias and learns arbitrary relations among every input variable. Sharkawy *et al.* [15] proposed to use the FC structure to account for the coupling effect among joints. Although their method is not an end-to-end learning and the input differs from ours, the FC structure is applied to observe the performance on the imbalanced data. To make the hidden neurons fully connected while maintaining the capacity of MNN, 90 hidden neurons are used instead of six modularized layers with 15 hidden neurons for each layer in the joint module network. The concatenate network structure remains the same.

Also, 1-D CNN was adopted by Park *et al.* [22] using univariate MOB torque τ_{MOB} as an input. In this section, their univariate MOB input τ_{MOB} is extended to multivariate raw sensor input Ver 1 for fair comparison. The network is composed of four 1-D convolution layers followed by one fully connected layer. The convolutions are executed along the time axis with a kernel size of 3. Therefore, multivariate raw sensor data $x_{\text{in}}(k) = [x_{\text{in}}^1(k), x_{\text{in}}^2(k), x_{\text{in}}^3(k), x_{\text{in}}^4(k), x_{\text{in}}^5(k), x_{\text{in}}^6(k)]$ are stacked in order of time $X_{\text{in}} = [x_{\text{in}}(k-40), x_{\text{in}}(k-30), x_{\text{in}}(k-20), x_{\text{in}}(k-10), x_{\text{in}}(k)]$ and convoluted along time axis as in Fig. 7. Although it is not mentioned in their paper, the convolution structure can be interpreted as imposing an inductive bias that the features can be extracted by convoluting the time series data within the time window of kernel size. It can also be roughly explained as decoupling the time axis of the data; with a kernel size of 3, the first time step data cannot be convoluted directly with the fourth time step data in the first layer, although they are indirectly convoluted in higher layers.

The performance according to the network structure is shown in Table III and Fig. 4(a)–(c). Table III shows detailed DF, DD, and FP for each tool weight, and the overall performance across whole weights are displayed in Fig. 4(a)–(c). Noticeably, the FC fails to detect collision frequently (46/245) and high DD (37.9 ms). In addition, the number of FPs is too high (293/1 h), which cannot be applied in practice due to the absence of any inductive bias either in the network structure or the input variable. Since the search space is not reduced implicitly, sufficiently large data are required to fit the parameter appropriately. However, recent studies such as [28] and [39] demonstrate that reducing inductive bias would ultimately exhibit superior performance when sufficient data are available. Thus, if abundant data are given, there is a potential that FC surpasses other inductively biased networks by fully accounting for the coupling effect among joints. On the other hand, 1-D CNN fails to detect 2 out of the 245 collisions with an average DD of 15.7 ms. FPs were decreased to 126, less than half of the FCs. It is shown that the 1-D CNN structure is effective in handling insufficient imbalanced collision data by assuming that collision features can be extracted by convoluting the neighboring time data. Furthermore, the proposed MNN shows the best performance missing only one collision of the 245 collisions with the lowest DD of 14.4 ms and the rarest 54 FPs, which are less than half of the 1-D CNN FPs. Thus, it is shown that the inductive bias on the network structure significantly improves the performance. Especially, the MNN is effective by independently extracting features of each joint in the joint module network and thereby has superior generalization performance due to its smaller search space to be trained.

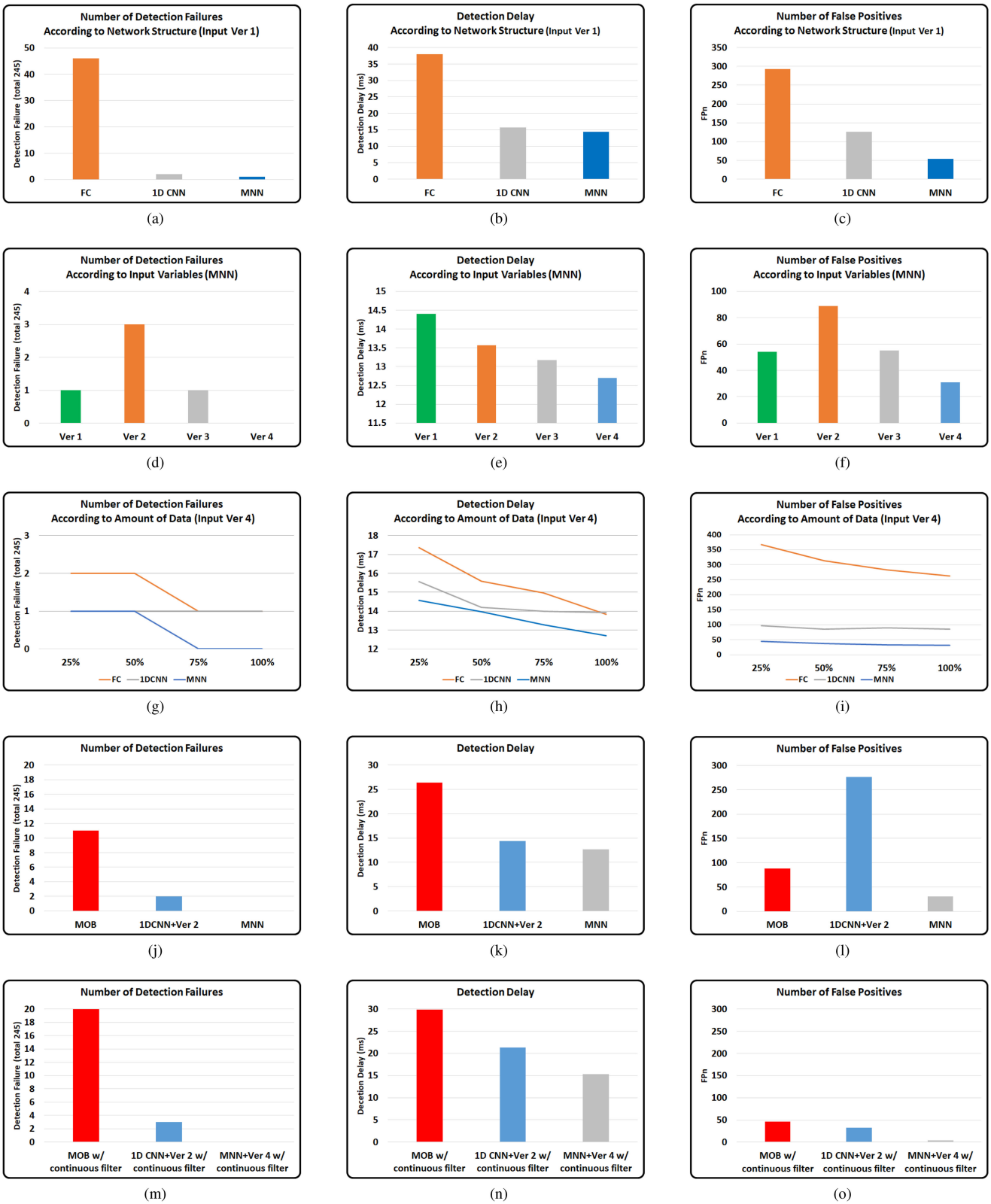


Fig. 4. (a), (b), (c) Number of detection failures, detection delay, and number of false positives of FC, MNN, and 1-D CNN with input Ver 1. (d), (e), (f) Number of detection failures, detection delay, and number of false positives according to input variables on MNN. (g), (h), (i) Number of detection failures, detection delay, and number of false positives of FC, MNN, and 1-D CNN according to amount of data with input Ver 4. (j), (k), (l) Number of detection failures, detection delay, and number of false positives of the proposed method, 1-D CNN with MOB torque, and model-based MOB method. (m), (n), (o) Number of detection failures, detection delay, and number of false positives of the proposed method, 1-D CNN with MOB torque, and model-based MOB method with continuous filter.

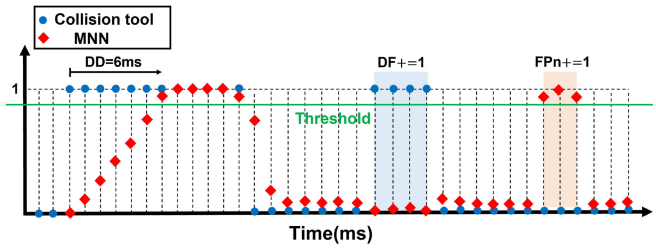


Fig. 5. Illustration of DD, DF, and FP. Each interval between vertical dotted lines represents an inference cycle of 1 ms. Blue dots are the ground truth from the switch attached to the collision tool and the red dots are the output from MNN. The green line denotes the threshold. When the output of MNN exceeds the threshold, the MNN estimates that the collision occurred.

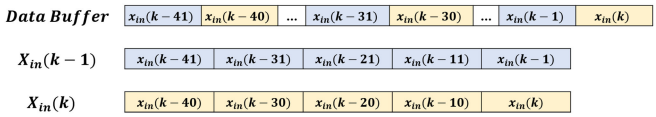


Fig. 6. Illustration of data buffer and input preprocessing with sampling interval of $t_I = 0.01$ s. Data up to 0.04 s should be stored in a buffer every 0.001 s to make an inference every 0.001 s and process the time-stacked input data.

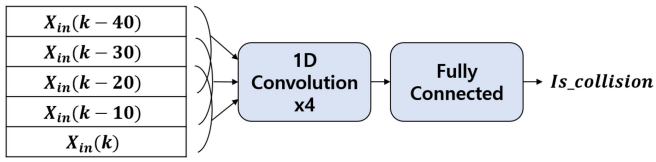


Fig. 7. 1-D CNN structure used by Park *et al.* [22]. The time-stacked input is convoluted along time axis with a kernel size of 3.

B. Performance According to the Input Variable

As described in Section II-A3, four versions of input variable are inspected to demonstrate the effectiveness of selecting proper inductive bias on the input variable. In this section, the network structure is fixed to the MNN to demonstrate the impact of input variable type alone.

According to the input variable, the performance is shown in Table IV and Fig. 4(d)–(f). By examining the performance of each version, the following characteristics are found.

1) Ver 4: By introducing proper inductive bias, Ver 4 shows the best performance in terms of DF, DD, and FP.
2) Ver 1 and Ver 4: Input variables of Ver 4 can be calculated from the variables of Ver 1. MOB torque τ_{MOB} is calculated from the joint position, joint velocity, and motor torque, and the error terms are calculated from reference and joint position/velocities. A performance comparison between Ver 1 and Ver 4 demonstrates that the processed (inductive biased) data outperform the raw data. However, note that the dynamics model is required when calculating the MOB torque. Therefore, when the dynamics model is not available, the unbiased input with MNN would be a substitute at the expense of performance.
3) Ver 1, 4 and Ver 2, 3: More information is contained in Ver 1 and Ver 4 than Ver 2 and Ver 3; Ver 1 and Ver 4 can utilize both raw or processed dynamics-related features and error-related features from motor torque, joint position, joint velocity, reference joint position, and reference joint velocity, whereas Ver 2 and Ver 3 only contain dynamics-related variables without error-related variables. Therefore, although inductively biased, Ver 2 and Ver 3 do not exhibit superior performance than Ver 1, which has abundant information. However, by comparing Ver 2 and Ver 4, or Ver 3 and Ver 4, which are equally inductively biased, it is shown that the error-related terms improve the performance.
4) Ver 2 and Ver 3: Ver 2 only contains the MOB torque which is low-pass filtered disturbance torque. Since the external and friction torques are included in the disturbance torque, the pure external torque cannot be inferred by the MOB torque alone. Nevertheless, the collision can be approximately detected from the relative difference between time series MOB torques. For example, since the friction is constant under constant velocity, the relative difference of MOB torque, in this case, would imply the external torque due to collision. However, the FPs frequently occur because the friction is not taken into account appropriately, as depicted in Fig. 4(f). The FPs mainly occur when the friction abruptly changes, such as acceleration or deceleration. By adding joint velocity in Ver 3, the friction can be modeled and the absolute external torque can be inferred by subtracting the friction torque from the disturbance torque in the neural network. As it can be seen in Fig. 4(d)–(f), using additional joint velocity variables enhances the overall DF, DD, and FP performance.
5) Ver 3 and Ver 4: By comparing Ver 3 and Ver 4, it is inspected that using the error-related features as input is beneficial in both detection sensitivity and FP. Although the external torque is exerted at the acceleration level, the

TABLE IV
PERFORMANCE ACCORDING TO THE INPUT VARIABLE

		Collision (5 min/tool)		Free-motion (15 min/tool)
Tool Weight	Input Variable	DF	DD(ms)	FPn
0.0 kg	Ver 1	0/62	15.75	13
	Ver 2	1/62	13.64	10
	Ver 3	0/62	13.96	0
	Ver 4	0/62	13.55	0
2.0kg	Ver 1	0/62	12.88	8
	Ver 2	1/62	12.43	3
	Ver 3	0/62	12.14	0
	Ver 4	0/62	12.08	0
3.3kg	Ver 1	0/60	12.39	9
	Ver 2	0/60	12.58	19
	Ver 3	0/60	11.98	13
	Ver 4	0/60	11.48	5
5.0kg	Ver 1	1/61	16.57	25
	Ver 2	1/61	15.63	59
	Ver 3	1/61	14.59	42
	Ver 4	0/61	13.67	26
Total	Ver 1	1/245	14.40	54
	Ver 2	3/245	13.57	89
	Ver 3	1/245	13.17	55
	Ver 4	0/245	12.70	31

TABLE V
PERFORMANCE OF EXISTING METHODS AND MNN

		Collision (5 min/tool)		Free-motion (15 min/tool)
Tool Weight	Algorithm	DF	DD(ms)	FPn
0.0kg	MNN	0/62	13.6	0
	MNN w/ CF	0/62	18.1	0
	1D CNN	0/62	18.2	30
	1D CNN w/ CF	0/62	28.7	3
	MOB	3/62	27.3	5
	MOB w/ CF	3/62	33.0	0
2.0 kg	MNN	0/62	12.1	0
	MNN w/ CF	0/62	14.0	0
	1D CNN	0/62	12.6	35
	1D CNN w/ CF	0/62	18.0	0
	MOB	2/62	24.5	2
	MOB w/ CF	2/62	27.42	0
3.3kg	MNN	0/60	11.5	5
	MNN w/ CF	0/60	13.7	0
	1D CNN	1/60	11.8	69
	1D CNN w/ CF	2/60	16.9	5
	MOB	3/60	23.5	33
	MOB w/ CF	4/60	25.9	17
5.0kg	MNN	0/61	13.7	26
	MNN w/ CF	0/61	15.3	4
	1D CNN	1/61	14.8	142
	1D CNN w/ CF	1/61	21.8	24
	MOB	3/61	21.8	48
	MOB w/ CF	11/61	33.2	29
Total	MNN	0/245	12.7	31
	MNN w/ CF	0/245	15.3	4
	1D CNN	2/245	14.4	276
	1D CNN w/ CF	3/245	21.3	32
	MOB	11/245	26.4	88
	MOB w/ CF	20/245	29.9	46

TABLE VI
DIRECT TRANSFER OF BASELINE NETWORK ACCORDING TO NUMBER OF DATA SOURCE ROBOTS FOR TRAINING

		Collision (20min)		Free-motion (60min)
Data Set	Network	DF	DD(ms)	FPn
Unit 1	1	0/259	11.4	508
	2	0/259	13.0	542
	3	0/259	11.3	617
	avg	0	11.9	555.67
Unit 1, 2	1	0/259	11.4	322
	2	0/259	11.2	287
	3	1/259	11.8	394
	avg	0.33	11.5	334.33

TABLE VII
TEST RESULT OF TRANSFERRED NETWORK ON TARGET ROBOT

		Collision (5min/tool)		Free-motion (15min/tool)
Tool Weight	Algorithm	DF	DD(ms)	FPn
0.0kg	MNN w/o CF	0/66	11.2	2
	MNN w/ CF	0/66	13.6	1
2.0kg	MNN w/o CF	1/62	14.1	3
	MNN w/ CF	1/62	16.5	1
3.3kg	MNN w/o CF	0/64	11.2	6
	MNN w/ CF	0/64	13.4	2
5.0kg	MNN w/o CF	0/67	14.4	21
	MNN w/ CF	0/67	17.1	2
Total	MNN w/o CF	1/259	12.7	32
	MNN w/ CF	1/259	15.1	6

TABLE VIII
PERFORMANCE ACCORDING TO FP HANDLING METHODS

		Collision		Free-motion
Tool Weight	Algorithm	DF	DD(ms)	FPn
0.0 kg	w/o FP handling	0	13.6	0
	CF	0	18.1	0
	Ensemble	0	14.6	0
	CF w/ Ensemble	0	20.5	0
2.0 kg	w/o FP handling	0	12.1	0
	CF	0	14.0	0
	Ensemble	0	13.5	0
	CF w/ Ensemble	0	15.8	0
3.3 kg	w/o FP handling	0	11.5	5
	CF	0	13.7	0
	Ensemble	0	13.0	0
	CF w/ Ensemble	0	15.4	0
5.0 kg	w/o FP handling	0	13.7	26
	CF	0	15.3	4
	Ensemble	1	15.5	6
	CF w/ Ensemble	1	18.4	2
Total	w/o FP handling	0	12.7	31
	CF	0	15.3	4
	Ensemble	1	14.1	6
	CF w/ Ensemble	1	17.5	2

impact is accumulated and appears in position and velocity errors. Also, the error terms are reliable features and effectively decrease the FPs. However, it should be denoted that since the error-related features are dependent on tracking performance, implementation on different robots might show no noticeable performance improvement and might even result in performance degradation.

C. Performance According to the Data Amount

This section demonstrates that the MNN is sample-efficiently trained, requiring less data than other structures. The FC, MNN, and 1-D CNN are trained with 25%, 50%, 75%, and 100% of the collected data in Table I, respectively, to observe how the performance changes according to the amount of data. Across all structures, input Ver 4 was used, which was proven to show the best performance among inspected inputs.

According to the amount of data, the result is displayed in Fig. 4(g)–(i); refer to Table X in the Appendix for detailed values. In all structures, the performance improves as networks are trained with larger data. However, the MNN trained with 25% of the collected data shows comparable performance with other structures that use the whole data, and the MNN with 50% of data outperforms them. Specifically, with 25% of the data, consisting of 90 min of free-motion and 50 min of collision data, the MNN shows moderate performance; one DF of the 245 collisions, 14.6 ms of DD, and 44 FPs during an hour of free-motion. The sample efficiency comes from the characteristics of the proposed inductively biased structure, as discussed in Section II-B1. It is determined that the MNN is adequately sensitive compared to other reported methods [15], [22], even though there could be room for the DD of MNN to be improved when more data are collected according to the tendency of the graph. Also, note that the unbiased FC continues to improve the performance as the data are getting bigger, and if sufficiently large data are available, which is not easy in practice, the FC

TABLE IX
F1 SCORE OF MNN, 1-D CNN, AND MOB

			* Threshold 0.5	
Tool Weight	Algorithm	Precision	Recall	F1 Score
0.0 kg	MNN	0.9995	0.9507	0.9745
	1D CNN	0.9986	0.9262	0.9610
	MOB	0.9992	0.6625	0.7968
2.0 kg	MNN	0.9990	0.9574	0.9778
	1D CNN	0.9971	0.9352	0.9651
	MOB	0.9984	0.7088	0.8290
3.3 kg	MNN	0.9913	0.9425	0.9663
	1D CNN	0.9832	0.9061	0.9431
	MOB	0.9852	0.5881	0.7365
5.0 kg	MNN	0.8845	0.9031	0.8937
	1D CNN	0.8383	0.8603	0.8491
	MOB	0.9702	0.4441	0.6093
			* Threshold 0.99	
Tool Weight	Algorithm	Precision	Recall	F1 Score
0.0 kg	MNN	1.0	0.9282	0.9628
	1D CNN	0.9993	0.9201	0.9581
	MOB	0.9992	0.6625	0.7968
2.0 kg	MNN	1.0	0.9389	0.9685
	1D CNN	0.9992	0.9300	0.9634
	MOB	0.9984	0.7088	0.8290
3.3 kg	MNN	0.9993	0.9131	0.9543
	1D CNN	0.9909	0.9008	0.9437
	MOB	0.9852	0.5881	0.7365
5.0 kg	MNN	0.9743	0.8565	0.9116
	1D CNN	0.8868	0.8477	0.8668
	MOB	0.9702	0.4441	0.6093

could outperform other inductively biased networks by fully accounting for the coupling effect among joints. Additionally, the 1-D CNN structure shows a tendency to converge as data increases. However, the convergence point is different from that of MNN, meaning that the inductive bias imposed on the 1-D CNN structure could be suboptimal.

D. Comparison to Existing Methods

Lastly, the proposed method, MNN with Ver 4 input, is compared with existing methods. Two collision detection algorithms are implemented for comparison; one is from the model-based method by Lee *et al.* [10] and the other from data-driven 1-D CNN with MOB torque input by Park *et al.* [22]. Among the various learning-based collision detection methods, a method using only the proprioceptive sensor and reporting state-of-the-art performance was selected as a comparison group. Zhang *et al.* [40] compared the collision detection performance of linear discriminant analysis (LDA) model, *k*-nearest neighbors (KNN) model, support vector machine (SVM) model, and feedforward neural network model, and Park *et al.* [22] analyzed SVM and 1-D CNN. It turned out that relatively simpler machine learning based methods, such as LDA, KNN, and SVM, showed limited generalization performance compared to the deep neural network models. Also, Kaname *et al.* [41] trained one class SVM without collision data, which has a huge advantage because it is difficult to collect collision data, but according to the authors' experience [34], learning without collision data results in frequent FP, showing low precision. Therefore, 1-D CNN with MOB torque input, which reports state-of-the-art end-to-end

TABLE X
PERFORMANCE ACCORDING TO THE AMOUNT OF DATA ON FC, MNN, AND 1D CNN WITH INPUT VER 4

*Threshold 0.99	Data Amount	25%			50%			75%			100%		
Tool weight		Collision (5 min)		Free-motion (15 min)	Collision (5 min)		Free-motion (15 min)	Collision (5 min)		Free-motion (15 min)	Collision (5 min)		Free-motion (15 min)
		DF	DD (ms)	FPn	DF	DD (ms)	FPn	DF	DD (ms)	FPn	DF	DD (ms)	FPn
0.0kg	FC	0/62	17.9	60	0/62	20.0	45	0/62	19.5	39	0/62	17.9	41
	MNN	0/62	15.0	1	0/62	15.5	0	0/62	14.8	1	0/62	13.6	0
	1D CNN	0/62	18.2	8	0/62	14.9	6	0/62	15.8	4	0/62	16.5	5
2.0 kg	FC	1/62	15.3	66	0/62	14.8	53	0/62	13.2	48	0/62	13	34
	MNN	0/62	12.9	1	0/62	12.9	1	0/62	12.1	0	0/62	12.1	0
	1D CNN	0/62	13.5	10	0/62	13.4	10	0/62	12.7	11	0/62	12.7	7
3.3 kg	FC	1/60	18.3	106	0/60	12.5	86	0/60	11.9	82	0/60	11.8	78
	MNN	1/60	15.7	8	0/60	12.8	7	0/60	12.2	8	0/60	11.5	5
	1D CNN	0/60	14.0	28	0/60	13.1	20	0/60	12.8	22	0/60	12.6	21
5.0 kg	FC	2/61	17.9	136	2/61	15.0	130	1/61	15.2	115	1/61	12.7	111
	MNN	1/61	14.7	35	1/61	14.7	30	0/61	14.0	25	0/61	13.7	26
	1D CNN	1/61	16.5	52	1/61	15.4	51	1/61	14.7	53	1/61	14.0	53
Total	FC	2/245	17.4	367	2/245	15.6	313	1/245	15.0	283	1/245	13.8	262
	MNN	1/245	14.6	44	1/245	14.0	37	0/245	13.3	33	0/245	12.7	31
	1D CNN	1/245	15.6	97	1/245	14.2	86	1/245	14.0	89	1/245	13.9	85

deep learning method [22], was chosen as the representative of learning-based method.

A model-based method that only uses joint encoder and motor current data, without a joint torque sensor, was chosen for a fair comparison. Lee *et al.* formulated and identified the friction model according to the joint velocity and reference joint velocity to remove the effect of friction from the MOB torque. The friction model they proposed is as follows:

$$\tau_f = \begin{cases} \tau_c \text{sgn}(\tau_h), \text{if } |\dot{q}| < \epsilon \text{ and } \dot{q}_d = 0 \\ \tau_s \text{sgn}(\tau_h), \text{if } |\dot{q}| < \epsilon \text{ and } \dot{q}_d \neq 0 \\ \tau_c \text{sgn}(\dot{q}) + \tau_v(\dot{q}), \text{if } |\dot{q}| \geq \epsilon \end{cases} \tag{18}$$

where τ_s is the static friction, τ_c is the Coulomb friction, and $\tau_v = \beta_1 \dot{q}^2 + \beta_2 \dot{q}$ is the viscous friction. Here, β_1 and β_2 are the coefficients of the viscous friction. Using the free-motion data from training data, the parameters of friction model $\tau_c, \tau_s, \beta_1, \beta_2$ are fitted. Collision is detected when any of the estimated external torque at six joints exceeds a threshold. The threshold is set to a scaled maximum training error at each joint. Heuristically, the thresholds of joints 1, 2, and 3 are set to 25% of the maximum training error and the thresholds are set to the maximum training error of each joints for joints 4, 5, and 6; lower thresholds at joints 4, 5, and 6 resulted in frequent FPs.

For data-driven 1-D CNN with MOB torque, the network comprises four 1-D convolution layers followed by one fully connected layer and the convolutions are executed along the time axis with a kernel size of 3 as mentioned in Section III-A. However, note that only the MOB torque, input Ver 2, is used as the input.

The results are shown in Table V and Fig. 4(j)–(l). The MNN shows the best performance by detecting all collisions with a 12.7 ms delay and 31 FPs. 1-D CNN with Ver 2 input also shows superior sensitivity, missing two collisions with 14.4 ms delay. The two DFs might have stemmed from the insufficient input; as the collision is detected from the relative difference between time series MOB torques as mentioned in Section III-B, slowly increasing collisions cannot be detected. Additionally, since the input is univariate, FPs occur frequently due to the unmodeled friction effect. On the other hand, the model-based MOB is insensitive, missing 10 of the 245 collisions with 26.5 ms DD. It is presumed that the friction model is not elaborate enough to express the nonlinearity of friction. Also, the friction model cannot reflect the hysteresis of friction because the parameters are fitted without time-series data. This difficulty in friction modeling makes the model-based methods limited. Additionally, the MNN trained with 25% of the collected data, shown in Table X in the Appendix, outperforms existing 1-D CNN and MOB trained with 100% of the collected data, signifying the sample efficiency of the MNN structure.

Meanwhile, Park *et al.* [22] used a CF to handle the frequent FPs, where the collision is detected if the network's output remains greater than the threshold continuously over for the interval t_c ms. Results of applying a CF with interval $t_c = 3$ ms are listed in Table V and Fig. 4(m)–(o). FPs effectively decreased across all the methods. For MNN, FPs reduced from 31 to 4, for 1-D CNN from 276 to 32, for MOB from 88 to 46. However, notice that the CF introduces a tradeoff between sensitivity and FP; the FPs are decreased at the expense of longer DD and increased DF. We have also tried several methods to handle FPs and adjust the tradeoff between sensitivity and FP; our trial and error and discussion are explained in detail in Appendix A.

Also, note that the performance difference of 1-D CNN from the original paper would have come from the amount of data; they collected more than twice the data. Since the MNN requires small data, the data were not collected as much as in their paper. It is expected that the performance of 1-D CNN increases with a larger dataset.

IV. Transfer Learning of Collision Detection Network for Collaborative Robot Mass Production

In this section, the problem of applying the deep learning based collision detection method to a new robot (but the same model) for mass production is covered. Among the training procedures, collecting collision labels is one of the most challenging tasks to conduct in the mass production line because it is arduous and dangerous to make collisions on the robot while the robot moves. Moreover, collisions could cause scratch on a new product or damage some parts of the robot. Therefore, transfer learning is adopted to leverage the knowledge of a pretrained network rather than training from scratch. A neural network for the new robot can be trained with a pre-trained neural network (baseline network) using the proposed transfer learning and only small amounts of free-motion data from the new robot. Notably, the free-motion data are often easy to obtain because free-motion from the new robot is commonly performed during the inspection process in mass production.

A. Problem Definition

In this section, we define a transfer learning and related notations and specify the problem of learning on a new robot for mass production in the transfer learning framework by Pan. *et al.* [42].

First, a *domain* $\mathcal{D}$ is composed of a feature space $\mathcal{X}$ and a marginal probability distribution $P(x)$, where $x \in \mathcal{X}$. For the collision detection, $\mathcal{X}$ is the vector space of the robot data, and the x is a particular instance of the input vector. After defining specific domain, $\mathcal{D} = \{\mathcal{X}, P(x)\}$, a *task* $\mathcal{T}$ is defined with two components $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$, where $\mathcal{Y}$ is a label space and $f(\cdot)$ is an objective predictive function. The function $f(x)$ can be expressed as $P(y|x)$ from a probabilistic viewpoint and can be learned from training data, which consist of pairs $\{x_i, y_i\}$, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. For the collision detection, the label space is $\mathcal{Y} = \{true, false\}$ and the function $f(\cdot)$ is the true collision detection function.

Then, the source domain $\mathcal{D}_S$, the source task $\mathcal{T}_S$, the target domain $\mathcal{D}_T$, and the target task $\mathcal{T}_T$ are defined where "source" means the place from which the knowledge is transferred and "target" means the subject to which the knowledge is transferred. Also, the source domain *data set* and the target domain *data set* is defined as $\text{DS}_S = \{(x_{S_1}, y_{S_1}), \ldots, (x_{S_{n_S}}, y_{S_{n_S}})\}$ and $\text{DS}_T = \{(x_{T_1}, y_{T_1}), \ldots, (x_{T_{n_T}}, y_{T_{n_T}})\}$, respectively, where $x_{S_i} \in \mathcal{X}_S$, $x_{T_i} \in \mathcal{X}_T, y_{S_i} \in \mathcal{Y}_S, y_{S_i} \in \mathcal{Y}_T, n_S$ is the number of source data, and n_T is the number of target data.

Now, transfer learning is defined as follows [42].

Definition 1 (Transfer Learning): Given a source domain $\mathcal{D}_S$, learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$, and learning task $\mathcal{T}_T$, *transfer learning* aims to help improve the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{T}_T$ using knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$.

In our case, the transfer learning of the collision detection task with different robots but the same model is considered where DS_S contains input feature and corresponding label of both free-motion and collision from the source robot. However, DS_T only contains collision-free input feature and corresponding $\mathcal{Y} = \{false\}$ label pairs, which are smaller than the source domain data ($n_T \ll n_S$, $\mathrm{DS}_T = \{(x_{T_1}, 0), \ldots, (x_{T_{n_T}}, 0)\}$). In this study, the same input variables are used for both the source robots and the target robot. Consequently, the input feature space is $\mathcal{X}_S = \mathcal{X}_T$. Also, the marginal distribution of the input is $P_S(X) \approx P_T(X)$ because source robots and the target robot are operated with the same control gains and same types of random trajectory, showing similar marginal distributions in joint angle-related variables ($\dot{\theta}, \theta_d - \theta, \dot{\theta}_d - \dot{\theta}$) between source and target robot. But, it is observed that the marginal distribution of the motor torque, $P(\tau_m)$, is slightly different between the source robots and the target robot. The difference in motor torque is oriented from different friction torque according to the units. Therefore, the relationship between the source and target domains can be described as $\mathcal{D}_S \approx \mathcal{D}_T$ rather than $\mathcal{D}_S = \mathcal{D}_T$.

In case of the task $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$, the label space is $\mathcal{Y}_S = \mathcal{Y}_T$ since the collision label space is $\mathcal{Y} = \{true, false\}$ for both $\mathcal{Y}_S$ and $\mathcal{Y}_T$. The conditional distribution $P(Y|X)$ may change across robots even if the robot model is the same. However, it is assumed that $P_S(Y|X) \approx P_T(Y|X)$, and more details are covered in Sections IV-B, IV-C, and IV-D. As a result, the source task is approximately equivalent to the target task, $\mathcal{T}_S \approx \mathcal{T}_T$. In summary, the transfer learning of the collision network between the same model can be categorized into the case where the source domain and the source task are closely related to the target domain and target task but not the same ($\mathcal{D}_S \approx \mathcal{D}_T$, $\mathcal{T}_S \approx \mathcal{T}_T$).

B. Baseline Network

If $\mathcal{D}_S = \mathcal{D}_T$ and $\mathcal{T}_S = \mathcal{T}_T$, a network trained in the source domain performs identically for the target task as it does for the source task. But, even the equally designed robots show different hardware characteristics due to manufacturing errors and nonlinear characteristics of elements. Especially, the friction torque is nonlinear and shows hysteresis at the harmonic drive resulting in the slightly drifted marginal distribution of target input and drifted conditional distribution of target task [43].

To alleviate the deviation of the $P_T(X)$ from the $P_S(X)$, multiple source robots are used to collect source data. It is hypothesized that the distribution of multirobots involves a more diverse range of data than that of the single robot. Therefore, the multirobot-trained network ignores individual-robot-specific characteristics and extracts general features common to multiple robots. In this study, three units of the robot are used: Unit 1 and unit 2 are used as source robots to pretrain the baseline network and unit 3 is used as a target robot as shown in Fig. 8.

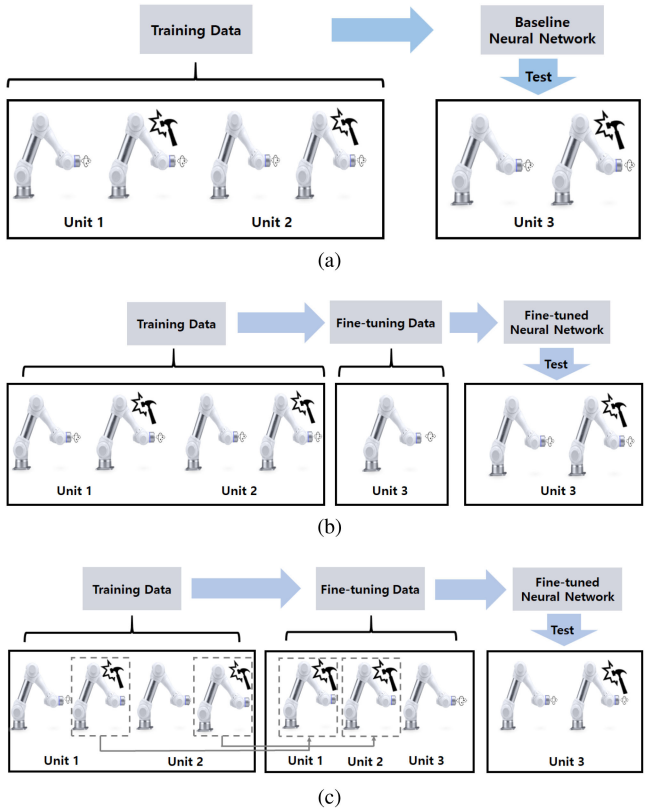


Fig. 8. Various transfer learning schemes. (a) Direct transfer without fine-tuning. (b) Fine-tuning with free-motion data from the target robot. (c) Fine-tuning with collision data from source robots and free-motion data from the target robot.

C. Data Mixture for Fine-Tuning

When directly applying the trained baseline network to the target robot, the performance cannot keep up with a network's collision detection performance which is trained using the target robot data (specific-robot-trained network). Therefore, among various transfer learning methods, fine-tuning all parameters of the pretrained baseline network is adopted; the entire weights of the baseline are updated using the target robot data with a smaller learning rate than the learning rate for the baseline learning. Depending on the network size and the amount of data, only fine-tuning some layers (e.g., concatenate network) and freezing other layers (e.g., joint module network) could be reasonable. However, since the proposed network size is small enough, all parameters were fine-tuned with the target data.

Accordingly, the optimal target model can be obtained by fine-tuning the baseline network using the target data (DS_T) as follows:

$$w^* = \arg\min_w \sum_{(x,y)\in \mathrm{DS}_T} P_T(x,y) l(x,y,w)$$

where $l(x, y, w)$ is a loss function according to the training data x, y and the network weight vector w. However, sufficiently

large labeled data from the target robot are not easy to be obtained by the constraint of mass production. Instead, the distribution of the source data can replace the distribution of the target data using importance sampling without bias [44]. Then, the source data can be utilized to fine-tune the baseline network

$$w^* = \arg\min_w \sum_{(x,y)\in \mathrm{DS}_S} \frac{P_T(x,y)}{P_S(x,y)} P_S(x,y) l(x,y,w). \quad (19)$$

However, in the mass production of the manipulator, free-motion data can be easily collected during the inspection process before the release of the robot. In such a case, the free-motion data from the target robot ($y=0$) can be used without importance sampling, and the importance sampling is only applied when using collision data from source robots ($y=1$). Therefore, a *data mixture* method is proposed to employ source data $\mathrm{DS}_S(y=1)$ partially for fine-tuning the target network; free-motion data from the target robot and collision data from the source robot are mixed to form fine-tuning data as shown in Fig. 8(c). As a result, (19) is divided into two terms

$$\begin{aligned} w^* &= \arg\min_w \Bigg[\sum_{x\in \mathrm{DS}_T} P_T(x, y=0) l(x, y=0, w) \\ &\quad + \sum_{x\in \mathrm{DS}_S} \frac{P_T(x,y=1)}{P_S(x,y=1)} P_S(x,y=1) l(x,y=1,w) \Bigg] \\ &\approx \arg\min_w \Bigg[\sum_{i=1}^{n_T} l(x_{T_i}, y_{T_i}=0, w) \\ &\quad + \sum_{i=1}^{n_S} \frac{P_T(x_{S_i}, y_{S_i}=1)}{P_S(x_{S_i}, y_{S_i}=1)} l(x_{S_i}, y_{S_i}=1, w) \Bigg]. \quad (20) \end{aligned}$$

The first term in (20) represents the loss of FP from free motion ($y=0$), and the second term represents the loss of false-negative from collision data ($y=1$). The first loss term is calculated using free-motion samples from the target robot, and the second loss term is calculated using collision samples from the source robot.

If the weighting factor $W_{T|S,i} := \frac{P_T(x_{S_i}, y_{S_i}=1)}{P_S(x_{S_i}, y_{S_i}=1)}$ corresponding to each sample in (20) can be calculated, the target model can be learned correctly. However, it is difficult to estimate $W_{T|S,i}$ correctly for each instance [45], [46]. Instead, using the relationships $P_S(X) \approx P_T(X)$ and $P_S(Y|X) \approx P_T(Y|X)$ in Section IV-A, it can be approximated that $P_T(x_{S_i}, y_{S_i}) \approx P_S(x_{S_i}, y_{S_i})$ resulting in $W_{T|S,i}=1$. Then, (20) can be formulated as follows:

$$\begin{aligned} w^* \approx \arg\min_w \Bigg[&\sum_{i=1}^{n_T} l(x_{T_i}, y_{T_i}=0, w) \\ &+ \sum_{i=1}^{n_S} l(x_{S_i}, y_{S_i}=1, w) \Bigg]. \quad (21) \end{aligned}$$

Using the data mixture method with $W_{T|S,i}=1$ is based on the assumption that the robot's dynamics is similar across the equally designed robots when the robot collides, and the differences between source and target robot can be learned by only using free-motion data. Because the significant difference across the same model is originated from the friction, the friction torque can be learned during free-motion.

The proposed data mixture method has advantages in fine-tuning because the fine-tuning of the target network is possible without further collision data from the target robot. Also, without collision labels from source robots as shown in Fig. 8(b), the fine-tuned network overfits free-motion data and loses collision detection ability as the training progresses, resulting in a suboptimal solution as follows:

$$w^* \neq \arg\min_w \left[\sum_{i=1}^{n_T} l(x_{T_i}, y_{T_i}=0, w) \right]. \quad (22)$$

The experimental results and further discussion are presented in Sections V-A and V-B.

D. Adjustment of Collision Ratio for Handling Collision Sensitivity

Although the collision data from the source robot is utilized for fine-tuning, the marginal distribution of the input ($P(X)$) and the response of the robot against collision ($P(Y|X)$) is slightly different across the units due to the different hardware characteristic and the manufacturing errors. Therefore, $P_T(x_{S_i}, y_{S_i}=1)$ is drifted from $P_S(x_{S_i}, y_{S_i}=1)$, violating the assumption in Section IV-C that $W_{T|S,i}=1$. This error results in a drifted performance of the target model. Specifically, due to different hardware characteristics between source units and target units, the sensitivity of the fine-tuned target network is changed from the sensitivity of the specific-robot-trained network (a network trained with specific robot data and applied to the same robot) in the source domain if the data mixture method is used without an appropriate weighting factor.

In order to correct the shifted sensitivity, the importance sampling factor $W_{T|S,i}$ should be estimated for each fine-tuning data. However, rather than estimating the $W_{T|S,i}$, the number of collision samples from the source data (n_S) is adjusted, changing the ratio of the collision data ($r_{\mathrm{col}} = \frac{n_S}{n_T}$). Our goal is to find the optimal collision ratio (r^*_{col}) showing the minimum difference between the second term of (20) and the sum of the loss from the collision data as described below

$$\begin{aligned} r^*_{\mathrm{col}} = \arg\min_{r_{\mathrm{col}}} \Bigg[&\sum_{i=1}^{n_S} \frac{P_T(x_{S_i}, y_{S_i}=1)}{P_S(x_{S_i}, y_{S_i}=1)} l(x_{S_i}, y_{S_i}=1, w) \\ &- \sum_{i=1}^{n_T \times r_{\mathrm{col}}} l(x_{S_i}, y_{S_i}=1, w) \Bigg] \quad (23) \end{aligned}$$

where the number of collision data for fine-tuning ($n_T \times r_{\mathrm{col}}$) is adjusted by the collision ratio (r_{col}). However, the optimization problem in (23) cannot be solved directly since the true weighing value $W_{T|S,i}$ is unknown. Therefore, the r_{col} that shows similar sensitivity to the specific-robot-trained network is selected as optimal r^*_{col} indirectly. This is based on the assumption that if the loss function is similar, the performance of the target network is comparable to the performance of the specific-robot-trained network. Note that the performance of the specific-robot-trained network in Section III, a network trained with unit 1 data and applied to unit 1, is considered optimal for the target model

because the specific-robot-trained network employed sufficient training data with heuristic hyperparameter tuning to obtain the desired performance. However, the target model starts from the baseline network using only small free-motion data for fine-tuning. Thus, it is desirable to train the target model as closest to the specific-robot-trained network as possible. The r^*_{col} could be obtained approximately within several tries by line search based on the straightforward rule: The higher the ratio of the collision data, the more the collision sensitivity increases, and vice versa. The experiments about collision ratio adjustment are conducted in Section V-C.

V. Test Results of Transfer Learning

Four experiments were conducted to validate our method. In the first experiment, the baseline network was directly applied to the target robot (the same model) without transfer learning, as shown in Fig. 8(a). Two kinds of baseline networks are compared to show the generalization performance as the number of source robots increases. For the second experiment, the effect of the data mixture method is validated using two different data sets for fine-tuning, where the first set has only free-motion data from the target robot and the other contains additional collision data from the source robots using data mixture method, as shown in Fig. 8(b) and (c), respectively. Third, the ratio of the collision data in the training data is adjusted to modify collision sensitivity, and the performance of the collision detection according to the ratio is observed. Lastly, the performance of the final network on the target robot is presented using all of the suggested methods and compared with the specific-robot-trained network. The average DF, DD, and FP of three independent networks are measured and analyzed through all test experiments of transfer learning. Testing data consists of 60 min of free-motion and 20 min of collision data containing 259 collisions, as summarized in Table II. The MNN with input Ver 4 is implemented with a threshold of 0.99. For the fine-tuning, the learning rate is 2e-6 which is five times smaller than the one for the baseline training, and the other hyperparameters and details are the same as in Section II.

A. Direct Transfer of Baseline Network to Target Robot

As shown in Fig. 8(a), the baseline network trained from the source robot was directly applied to the target robot. Two baseline networks were trained to see how the number of source robot affects the performance. The first baseline is trained using only data from unit 1 and the second baseline is trained using data from both unit 1 and unit 2. The total amounts of training data are adjusted to be equal because we want to compare the effects of the diversity of the training data on the transfer learning rather than the amounts of the data. As shown in Table VI, the average results of three randomly initialized and independently trained models indicate that the network trained using data from two robots outperforms the network trained using data from one robot in terms of FPn while the collision sensitivity (DD and DF) is comparable between two data sets. It can be inferred that the larger the number of source robots used for collecting data, the better the general features of reacting to collisions across robots can be captured. As a result, the network trained from unit 1 and unit 2 was used as the baseline network. However, since the baseline is too sensitive when applied to the target robot, showing large FPn (334/1 h), it is hard to use the baseline network directly without further fine-tuning.

B. Effect of Data Mixture Method on Fine-Tuning

As the results in the previous section show, directly transferring the baseline network to the target robot leads to frequent FP alarms. Therefore, additional fine-tuning is conducted. Two fine-tuning data sets are compared, as illustrated in Fig. 8(b) and (c). Fig. 8(b) denotes the process of fine-tuning only using free motion data of the target unit and Fig. 8(c) is with the data mixture method that organizes fine-tuning data by blending free-motion data from the target unit and collision data from the source units. Note that the ratio of collision label in the fine-tuning data for data mixture is set to be the same as that of single unit training data in Section II-C, which is about 5%.

Fig. 9(a)–(c) shows the results of fine-tuning the baseline network trained with two data sets. When only the free-motion data is used for fine-tuning (red line), DF and DD diverge in a few epochs, and the FPn decreases to zero. On the other side, the fine-tuning results using the data mixture method (blue line) shows that DD and DF converge to a specific value without divergence, and FPn decreases but converges to a particular value in the same way. The results mean that if the training data contain only the free-motion data, the network loses its collision detection ability. However, if the data mixture method is used, the loss function of false-negative also contributes to the gradient of the loss function making the fine-tuning converge to a certain suboptimal value, preventing divergence.

C. Handling Collision Sensitivity Adjusting Collision Ratio in Training Data

In this section, various collision ratios in fine-tuning data are used to handle collision sensitivity. As described in Section IV-D, using the collision data from the source robot for fine-tuning the target network results in different collision sensitivity compared to the network's sensitivity trained only for the source robot. In order to use the collision network practically, the collision sensitivity of the robot must be adjustable so that the collision network does not produce false alarms but still sensitive enough to ensure safety.

In Fig. 10, the network's performance (DF, DD, and FP) is plotted according to the collision ratio of the training data. The experiment is conducted by halving the ratio of the collision data from 10% to 0.312%. Note that the collision ratio of training data for a single unit in Section II-C was about 5%. As shown in Fig. 10(a) and (b), DF and DD converge to a specific value in all cases, and the higher the ratio of collision data, the more sensitive the network becomes. From the same perspective, a higher collision ratio leads to more FPn as shown in Fig. 10(c). Therefore, it is experimentally validated that if the r_{col} is adjusted in (23), the sensitivity of the target network can be controlled accordingly.

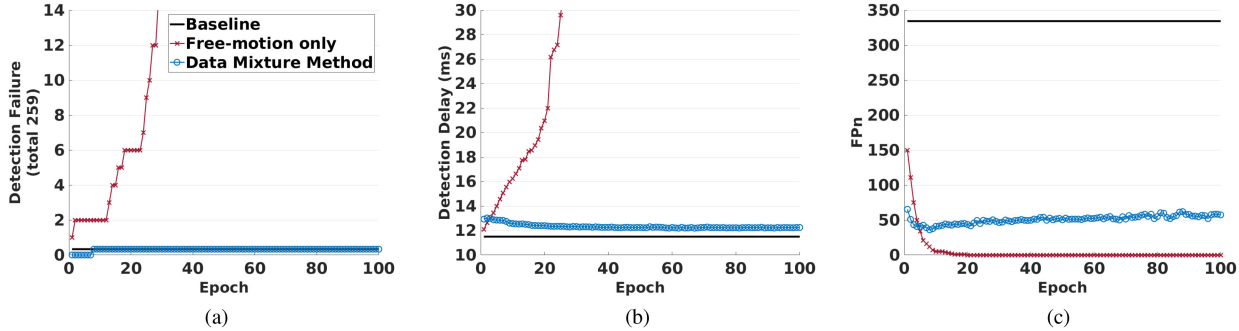


Fig. 9. Performance of transfer learning according to fine-tuning epochs. (a), (b), (c) Results of fine-tuning using only free-motion data of target robot (red line) and data mixture method (blue line). The collision ratio in training data for this experiment is 5%.

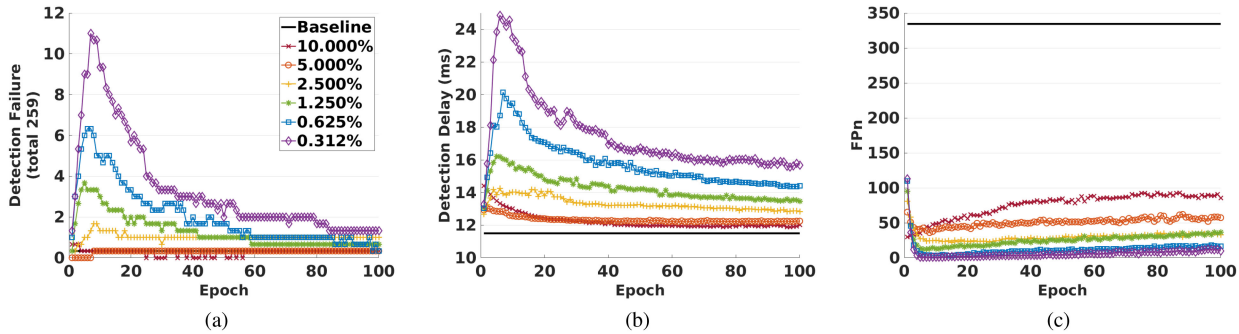


Fig. 10. Performance of transfer learning with various ratios of collision labels in training data. (a), (b), (c) Results of fine-tuning using six different ratio from 10% to 0.312% (colored lines), and results of the baseline network before fine-tuning (black line).

D. Comparison Between Transferred Network and Specific-Robot-Trained Network

In this section, the performance of the fine-tuned network on the target robot is compared with the specific-robot-trained network to see the performance degradation caused by the absence of the specific-robot collision data. The results of unit 1 in Table V are used as the specific-robot-trained network performance. The optimal ratio of the collision data for the fine-tuning (r^*_{col}) is set to 2.5%, and the training epoch for the fine-tuning was set to 50, where FPn is similar to the final result of unit 1 in Section III-D. In addition, the same CF with a 3-ms window is applied.

As shown in Table VII, the transferred network has comparable performance with the specific-robot-trained network (test results of the unit 1 in Section III) both in collision sensitivity and FP; the transferred network to unit 3 shows the performance (DF: 1/259, DD: 15.1 mm, FPn: 6) while the test result of unit 1 was (DF: 0/245, DD: 15.3 mm, FPn: 4) using the CF. The target network for unit 3 shows similar performance across all the tool weights compared to the robot-specific-trained network of unit 1. The performance of unit 1 is regarded as a maximum of the transferred network because the network for unit 1 is trained with data from its domain, but the transferred network is fine-tuned with free-motion data from the target robot and collision data from the source robots. Also, the amounts of the fine-tuning data are much smaller than the one of the training data for unit 1. Nevertheless, the proposed transfer learning method successfully imparts the knowledge of the baseline network to the target robot without any performance loss. The comparative result of the target network validates that the hypotheses in Section IV are reasonably correct.

It is worth to emphasize that the proposed transfer learning does not require further collision data from the target robot but needs only an hour of free-motion data. Consequently, the proposed transfer learning enables the deep learning based collision detection to be implemented in mass production, requiring minimum resources: a few source robots, training data on the source robots, and a small amount of free-motion data on the target robot. Although training data from source robots, including free-motion and collision data, should be collected once, we believe that the advantage of obtaining a decent collision detection network on every new target robot with only 1 h of free-motion data is significant.

VI. Conclusion

In this article, a deep learning based end-to-end collision detection method applicable to collaborative robots was introduced. An inductive bias was imposed on network structure and input variable to be sample-efficiently trained, which is suitable for hard-to-collect and imbalanced collision data. The proposed MNN is sample-efficient by reducing the search space

of the learnable parameter. Also, the input variable that takes both dynamics features and error-related features into account demonstrated improved performance. Consequently, the proposed method is versatile, showing successful generalization performance to random motion, random collision location, and various loads at the end-effector.

Additionally, a transfer learning method that considers constraints in mass production was proposed. Because collecting collision data on every new target robot is risky and laborious, our method does not require any collision data from the target robot. With the pretrained network from source robots, only less than an hour of free-motion data without collision data from the new robot are required to fine-tune the network due to our compact network structure. Furthermore, the collision sensitivity could be adjusted by altering the ratio of the collision data. Deep learning based collision detection can be effectively applied to mass production without losing performance using the proposed method, compared to a specific-robot-trained network, which can extend the usability of the data-driven method to the industrial field.

In the future, additional research can be conducted as follows. First, although the trained model shows sensitive detection, FPs should be further handled. For example, the FP can be reduced by increasing the number of networks used for the ensemble, which is explained in the Appendix. However, to deal with the increasing amount of computation, a knowledge distillation technique can be used to learn the input–output relationship of an ensemble to a new small network. Also, although the proposed transfer learning enables obtaining a collision detection network with a decent performance, it still requires collision data from the source robots. Ways to train without collision data in a semisupervised or unsupervised manner or to generate virtual collision data using a generative model can be considered since the most challenging part of this method is collecting collision data. Lastly, collided link can be identified by inspecting the output of the joint module network using an attention map or layerwise relevance propagation.

A False Positive Handling

In this section, our trial and error process to handle FP is shared and each method's effectiveness is discussed. FP is a critical issue when arranging robots in a real-world application. For safety reasons, it is a common strategy to stop robots whenever a collision is detected. However, if the detected collision is an FP, this degrades the system efficiency. Also, FPs are more problematic than false negatives because false negatives at the early stage of collision can be tolerated as a DD, whereas every FP stops the robot. There are mainly two categories to handle FPs: lead training procedure (training method) so that the learned network shows low FPs and the other is postprocessing the output (postprocessing). Note that all the FP handling methods described below reduce FP at the expense of detection sensitivity.

1) Training method:
   a) Weighted cross-entropy loss:
   The first method that has been tried to decrease FPs is to set a loss function, leading to training to reduce FPs. Weighted cross-entropy loss (WCEL) was used as a loss function [47] in this sense. WCEL is

$$\text{WCEL} = -(\alpha * y * \log(\hat{y}) + (1-y) * \log(1-\hat{y})) \tag{24}$$

   where y is the recorded collision label, $\hat{y}$ is the prediction of the network, and α is the weighting factor. When $\alpha = 1$, the first term on the right-hand side of (24) is the loss from false negatives; the second term is the loss from FPs. Thus, it is expected that setting $\alpha > 1$ decreases the false-negative count, hence increasing the recall, whereas setting $\alpha < 1$ decreases the FP count and increases the precision. However, while tuning the α by decreasing in log scale, it is found that using WCEL does not adjust sensitivity properly and the network suddenly fails to detect the collision when $\alpha = 1e^{-7}$. It is presumed that modifying the loss landscape directly by tuning α does not change the location of local minima until some threshold and suddenly leads to unstable training when α exceeds the threshold, especially when the data is imbalanced. Thus, WCEL was not used in this article.
   b) Collision data ratio adjustment:
   The second method to lead training to a low FP way is adjusting the ratio of collision data as described in Section IV-D. From our quantitative experiments, it is observed that the ratio of the collision labels in training data is closely related to the collision sensitivity. Therefore, if the network shows frequent FPs, decreasing the collision ratio in the training data effectively adjusts the tradeoff between sensitivity and FPs. Although, when the total amount of the training data is insufficient, a small ratio of collision data results in too sparse collision labels in absolute numbers. In such a case, the network would not learn the general collision features and would not predict the collision well. Thus, it is essential to check the absolute amount of the collision labels in the training data. In our experiment, a ratio between 1% and 5% was desirable.
2) Postprocessing:
   a) Continuous filter:
   As mentioned in Section III-D, CF effectively reduces the FP. However, there are some drawbacks and limitations to the CF. First, in terms of detection sensitivity, the CF is vulnerable to false negatives (the network infers that the collision did not occur while the collision occurred); any false negative during a collision would reset the collision count, thereby making the detection insensitive. Second, while the CF is effective when the network outputs noisy estimation, the CF is vulnerable when the network is mistrained in some local region; the network would continuously infer that the collision occurred in the region. This mistraining can be alleviated using the ensemble method below.
   b) Ensemble:
   The ensemble combines the predictions from multiple independently trained neural network models, stabilizing

the output by reducing the variance [48]. During training, randomly initialized networks converge to different local minima and explore diverse modes in function space [49]. Therefore, unlike the CF, the mistraining of one network would not lead to FPs by exploiting diverse results from multiple networks. Since the proposed MNN is sufficiently small, the ensemble method can be applied. The results with 1) no FP handling, 2) CF, 3) ensemble, and 4) CF and ensemble altogether on MNN are shown in Table VIII. Two separate networks were used for the ensemble and a collision was inferred when the output of both networks simultaneously exceeded the threshold. It is shown that the ensemble also effectively decreases FPs and the CF with ensemble shows only two FPs for an hour while detecting collisions sensitively.

B F1 Score

Since the precision and recall indicate whether the neural network inferences correctly under collision and free-motion, the f1 score can be utilized as the measure of how accurately the neural network can estimate collision states. The f1 scores of MNN, 1-D CNN, and MOB with a threshold of 0.5 and 0.99 are shown in Table IX. The MNN shows high precision of over 0.99 with a threshold of 0.5, besides when the 5.0 kg tool is attached, and by using a threshold of 0.99, the MNN shows higher precision while maintaining a modest recall and f1 score. In addition, the area under the receiver operating characteristic curve of the trained model is 0.9641, 0.9694, 0.9566, and 0.9281 for each tool, which supports that the trained network does not work only on a specific threshold.

C Detailed Performance According to the Amount of Data

In this section, detailed performances according to the amount of data on FC, 1-D CNN, and MNN are shown. The FC, MNN, and 1D CNN are trained with 25%, 50%, 75%, and 100% of the collected data in Table I to observe how the performance changes according to the amount of data as mentioned in Section III-C.

REFERENCES

[1] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human-robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248–266, 2018.

[2] S. Robla-Gómez, V. M. Becerra, J. R. Llata, E. González-Sarabia, C. Torre-Ferrero, and J. Pérez-Oria, "Working together: A review on safe human-robot collaboration in industrial environments," *IEEE Access*, vol. 5, pp. 26754–26773, Nov. 2017.

[3] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 338–345.

[4] A. Mohammed, B. Schmidt, and L. Wang, "Active collision avoidance for human-robot collaboration driven by vision sensors," *Int. J. Comput. Integr. Manuf.*, vol. 30, no. 9, pp. 970–980, 2017.

[5] T. Fan, P. Long, W. Liu, and J. Pan, "Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios," *Int. J. Robot. Res.*, vol. 39, no. 7, pp. 856–892, 2020.

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

[7] V. Duchaine, N. Lauzier, M. Baril, M.-A. Lacasse, and C. Gosselin, "A flexible robot skin for safe physical human robot interaction," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 3676–3681.

[8] A. De Luca and R. Mattone, "Actuator failure detection and isolation using generalized momenta," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2003, pp. 634–639.

[9] A. De Luca, A. Albu-Schaffer, S. Haddadin, and G. Hirzinger, "Collision detection and safe reaction with the DLR-III lightweight manipulator arm," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2006, pp. 1623–1630.

[10] S.-D. Lee, M.-C. Kim, and J.-B. Song, "Sensorless collision detection for safe human-robot collaboration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 2392–2397.

[11] M. H. Terra and R. Tinós, "Fault detection and isolation in robotic manipulators via neural networks: A comparison among three architectures for residual analysis," *J. Robotic Syst.*, vol. 18, no. 7, pp. 357–374, 2001.

[12] A.-N. Sharkawy, P. N. Koustoumpardis, and N. A. Aspragathos, "Manipulator collision detection and collided link identification based on neural networks," in *Proc. Int. Conf. Robot. Alpe-Adria Danube Region*, 2018, pp. 3–12.

[13] K. M. El Dine, J. Sanchez, J. A. Corrales, Y. Mezouar, and J.-C. Fauroux, "Force-torque sensor disturbance observer using deep learning," in *Proc. Int. Symp. Exp. Robot.*, 2018, pp. 364–374.

[14] Y. J. Heo, D. Kim, W. Lee, H. Kim, J. Park, and W. K. Chung, "Collision detection for industrial collaborative robots: A deep learning approach," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 740–746, Apr. 2019.

[15] A.-N. Sharkawy, P. N. Koustoumpardis, and N. Aspragathos, "Human-robot collisions detection for safe human-robot interaction using one multi-input-output neural network," *Soft Comput.*, vol. 24, no. 9, pp. 6687–6719, 2020.

[16] S. Haddadin, *Towards Safe Robots: Approaching Asimov's 1st Law*, vol. 90. Berlin, Germany: Springer, 2013.

[17] D. Verstraete, A. Ferrada, E. L. Droguett, V. Meruane, and M. Modarres, "Deep learning enabled fault diagnosis using time-frequency image analysis of rolling element bearings," *Shock Vib.*, vol. 2017, 2017, Art. no. 5067651, doi: 10.1155/2017/5067651.

[18] A. Rahimi, K. D. Kumar, and H. Alighanbari, "Fault detection and isolation of control moment gyros for satellite attitude control subsystem," *Mech. Syst. Signal Process.*, vol. 135, 2020, Art. no. 106419.

[19] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "LSTM fully convolutional networks for time series classification," *IEEE Access*, vol. 6, pp. 1662–1669, Dec. 2017.

[20] S.-Y. Shao, W.-J. Sun, R.-Q. Yan, P. Wang, and R. X. Gao, "A deep learning approach for fault diagnosis of induction motors in manufacturing," *Chin. J. Mech. Eng.*, vol. 30, no. 6, pp. 1347–1356, 2017.

[21] A.-N. Sharkawy, P. N. Koustoumpardis, and N. Aspragathos, "Neural network design for manipulator collision detection based only on the joint position sensors," *Robotica*, vol. 38, no. 10, pp. 1–19, 2019.

[22] K. M. Park, J. Kim, J. Park, and F. C. Park, "Learning-based real-time detection of robot collisions without joint torque sensors," *IEEE Robot. Automat. Lett.*, vol. 6, no. 1, pp. 103–110, Jan. 2021.

[23] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Special issue on learning from imbalanced data sets," *ACM SIGKDD Explorations Newslett.*, vol. 6, no. 1, pp. 1–6, 2004.

[24] N. Japkowicz *et al.*, "Learning from imbalanced data sets: A comparison of various strategies," in *Proc. AAAI Workshop Learn. Imbalanced Data Sets*, 2000, pp. 10–15.

[25] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, *Learning From Imbalanced Data Sets*. Berlin, Germany: Springer, 2018.

[26] D. Haussler, "Quantifying inductive bias: AI learning algorithms and Valiant's learning framework," *Artif. Intell.*, vol. 36, no. 2, pp. 177–221, 1988.

[27] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "DeepMimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1–14, 2018.

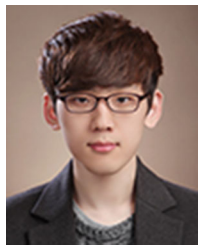
[28] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929v2*.

[29] A.-N. Sharkawy, P. N. Koustoumpardis, and N. Aspragathos, "A neural network-based approach for variable admittance control in human-robot cooperation: Online adjustment of the virtual inertia," *Intell. Serv. Robot.*, vol. 13, no. 4, pp. 495–519, 2020.

[30] H. Wu and J. Zhao, "Fault detection and diagnosis based on transfer learning for multimode chemical processes," *Comput. Chem. Eng.*, vol. 135, 2020, Art. no. 106731.

[31] W. Mao, L. Ding, S. Tian, and X. Liang, "Online detection for bearing incipient fault based on deep transfer learning," *Measurement*, vol. 152, 2020, Art. no. 107278.

[32] J. Xiao, Q. Zhang, Y. Hong, G. Wang, and F. Zeng, "Collision detection algorithm for collaborative robots considering joint friction," *Int. J. Adv. Robotic Syst.*, vol. 15, no. 4, 2018, Art. no. 1729881418788992.
[33] N. Hirose and R. Tajima, "Modeling of rolling friction by recurrent neural network using LSTM," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 6471–6478.
[34] D. Lim, D. Kim, and J. Park, "Momentum observer-based collision detection using LSTM for model uncertainty learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 4516–4522.
[35] M. K. Cılız and M. Tomizuka, "Friction modelling and compensation for motion control using hybrid neural network models," *Eng. Appl. Artif. Intell.*, vol. 20, no. 7, pp. 898–911, 2007.
[36] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *J. Big Data*, vol. 6, no. 1, pp. 1–54, 2019.
[37] D. W. Scott, *Multivariate Density Estimation: Theory, Practice, and Visualization*. Hoboken, NJ, USA: Wiley, 2015.
[38] R. May, G. Dandy, and H. Maier, "Review of input variable selection methods for artificial neural networks," *Artif. Neural Netw.-Methodol. Advances Biomed. Appl.*, vol. 10, 2011, Art. no. 16004.
[39] I. Tolstikhin *et al.*, "MLP-mixer: An all-MLP architecture for vision," 2021, *arXiv:2105.01601v4*.
[40] Z. Zhang, K. Qian, B. W. Schuller, and D. Wollherr, "An online robot collision detection and identification scheme by supervised learning and Bayesian decision theory," *IEEE Trans. Automat. Sci. Eng.*, vol. 18, no. 3, pp. 1144–1156, Jul. 2021.
[41] K. Narukawa, T. Yoshiike, K. Tanaka, and M. Kuroda, "Real-time collision detection based on one class SVM for safe movement of humanoid robot," in *Proc. IEEE-RAS 17th Int. Conf. Humanoid Robot.*, 2017, pp. 791–796.
[42] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[43] T. D. Tuttle and W. P. Seering, "A nonlinear model of a harmonic drive gear transmission," *IEEE Trans. Robot. Automat.*, vol. 12, no. 3, pp. 368–374, Jun. 1996.
[44] R. M. Neal, "Annealed importance sampling," *Statist. Comput.*, vol. 11, no. 2, pp. 125–139, 2001.
[45] B. Zadrozny, "Learning and evaluating classifiers under sample selection bias," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, pp. 114–121.
[46] J. Huang, A. Gretton, K. Borgwardt, B. Scholkopf, and A. Smola, "Correcting sample selection bias by unlabeled data," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2006, pp. 601–608.
[47] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2999–3007.
[48] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *J. Artif. Intell. Res.*, vol. 11, pp. 169–198, 1999.
[49] S. Fort, H. Hu, and B. Lakshminarayanan, "Deep ensembles: A loss landscape perspective," 2019, *arXiv:1912.02757v2*.



Donghyeon Kim received the B.E. degree in mechanical engineering from Sungkyunkwan University, Suwon, South Korea, in 2018. He is currently working toward the Ph.D.degree in intelligent systems with the Department of Transdisciplinary Studies, Seoul National University, Seoul, South Korea.

His research interests include robot-applied deep learning, reinforcement learning, and humanoids.

Daegyu Lim received the B.E. degree in mechanical engineering in 2017 from Seoul National University, Seoul, South Korea, where he is currently working toward the Ph.D.degree in intelligent systems with the Department of Transdisciplinary Studies.

His research interests include humanoids, bipedal locomotion, and reinforcement learning.

Jaeheung Park (Member, IEEE) received the B.S. and M.S. degrees in aerospace engineering from Seoul National University, Seoul, South Korea, in 1995 and 1999, respectively, and the Ph.D. degree in aeronautics and astronautics from Stanford University, Stanford, CA, USA, in 2006.

He is currently a Professor with the Department of Intelligent Convergence Systems, Seoul National University. His research interests include robot–environment interaction, contact force control, and robust haptic teleoperation.