Towards Deep Transfer Learning in Industrial Internet of Things

Xing Liu*, Wei Yu*, Fan Liang*, David Griffith†, and Nada Golmie†
*Towson University, USA
Emails: {xliu10, fliang1}@students.towson.edu, wyu@towson.edu
†National Institute of Standards and Technology (NIST), USA
Emails:{david.griffith, nada.golmie}@nist.gov

***Abstract*—Machine learning techniques have been widely adopted to assist in data analysis in a variety of Internet of Things (IoT) systems. To enable flexible use of trained learning models, one viable solution is to leverage all categories of data from different applications to train a general model, which can be further tuned for applications through tuning process. This process incurs additional overhead at the start, but makes later revision and iteration faster and more flexible. Nonetheless, due to limited computing capabilities, IoT devices cannot handle the training process of large datasets. To address this issue, in this paper, we propose a general framework to adopt transfer learning in industrial Internet of Things (IIoT) systems. In our study, we categorize the application space of applying transfer learning to IIoT systems into four generic scenarios: centralized transfer learning with large datasets, distributed transfer learning with large datasets, centralized transfer learning with small datasets, and distributed transfer learning with small datasets. According to the characteristics of each scenario, we design workflows to apply transfer learning technique. To demonstrate the efficacy of the approach, we apply our transfer learning technique to the task of IIoT component recognition. We use the known VGG-16 model and leverage T-Less industrial datasets to evaluate the performance of our approach in different scenarios. Via performance evaluation, our experimental results confirm the efficacy of our approach, which can not only reduce training time, but also achieve higher accuracy, compared with the classical convolutional neural network (CNN) approach.**

***Index Terms*—Industrial Internet of Things, Machine Learning, Transfer Learning**

I. Introduction

Given recent advances in Internet of Things (IoT) techniques, the Industrial Internet of Things (IIoT) is set to become the key driving force for improving productivity in manufacturing systems [1], [2]. As a typical cyber-physical system (CPS), the IIoT system consists of both the cyber and physical subsystems, which cooperate to collect, transfer, and analyze data. Based on the analysis results, the monitoring and control of systems can be improved, enabling automation and intelligent event response. One key to realizing automation and intelligence is enabling big computing, modeling, and data-driven analysis, which can be achieved through techniques such as deep learning [3]. In general, IoT datasets are transmitted to computing devices, which need to have high computation capabilities to carry out data analysis. Moreover, the data transmission process can raise significant overhead to the network. Clearly, due to the constraints to network and computation resources in IoT systems, it is quite challenging to handle transmission and analysis of massive amounts of data efficiently in IIoT systems.

Machine learning techniques, as viable knowledge discover approaches, have been widely adopted to assist in data analysis for applications [4]. Big data analysis techniques have shown great potential in a number of areas, including image/video recognition, data categorization, and artificial intelligence. Machine learning techniques can be a viable solution for assisting in data analytics in IIoT systems. Nonetheless, machine learning models are designed for specific applications and training models is time-consuming. To obtain accurate data analysis results, learning models require a massive amount of data and incur significant training time, which cannot provide timely decisions for IIoT applications and may exacerbate the overhead limitations of network and computation resources.

To address these issues described, several approaches have been considered. One approach involves deploying the machine learning model on edge computing devices/servers instead of a central computing server in order to reduce network traffic [5]. Nonetheless, the computing resources of edge computing servers are much lower compared to central computing servers/datacenters, which further increases the training time. Another approach involves training a general model by feeding all categories of datasets from different applications. Nonetheless, such an approach will increase the size of dataset collection and storage, and will not obtain the optimized results for all applications. Thus, how to reduce the size of datasets and training time while maintaining the accuracy of learning models is a critical and challenging issue.

To address this issue, in this paper we leverage transfer learning in IIoT systems to improve the applicability of a well-trained general learning model, which can be used to avoid lengthy training times of particularized models for designated applications. To be specific, based on the size of data and the way that the machine learning process is deployed in a distributed or centralized manner, we categorize the entire IIoT application space into four representative scenarios: centralized transfer learning with large datasets, distributed transfer learning with large datasets, centralized transfer learning with small datasets, and distributed transfer learning with small datasets. We design a workflow for each scenario to apply transfer learning techniques. The basic scheme of our transfer learning approach is to use a well-trained deep learning model

as a feature extractor, and then use the extracted features to retrain a classifier. For different application scenarios, we extend this basic scheme. Finally, we use the industrial IoT component recognition as an example to demonstrate the efficacy of our approach. We choose the T-Less industrial dataset [6] as the training and testing datasets to reproduce industrial component recognition tasks. As a result, our proposed approach can significantly reduce the size of training data and training time, while achieving better training accuracy compared to the classical CNN-based approach.

To summarize, we make the following contributions:

- **Framework:** We propose a general framework to adopt transfer learning in a variety of IIoT scenarios. In detail, we categorize the entire application space into four scenarios based on the amount of data and the way that the machine learning process is operated in a distributed or centralized manner.
- **IIoT Transfer Learning:** We design a transfer learning approach for each defined scenario in the IIoT system. For the centralized transfer learning scenarios with large datasets and small datasets, we deploy the transfer learning process on the centralized computing center that has high computing capacity. For the distributed transfer learning scenarios with large datasets, we deploy the transfer learning process on the local server that has relatively small computing capacity. For the distributed transfer learning scenarios with small datasets, we leverage the distributed nature of the IIoT system to deploy multiple transfer learning models on different local servers, simultaneously. Then, we obtain the final classification results based on the results of all different models via a voting mechanism.
- **Extensive Validation:** We conduct extensive performance evaluation to validate the effectiveness of our transfer learning approach on a representative IIoT system, as compared to an existing representative approach based on classical CNN in different scenarios. Our experimental results show that our proposed approach can significantly reduce the size of training data and training time while achieving better training accuracy than the existing representative CNN-based approach.

The remainder of this paper is organized as follows: In Section II, we conduct a brief literature review of relevant studies regarding IIoT and transfer learning. In Section III, we provide background relevant to our work. In Section IV, we introduce our approach in detail. In Section V, we present the evaluation results. In Section VI, we discuss some open issues and future research directions. Finally, we summarize the paper in Section VII.

II. Related Work

Before we review the application of transfer learning in IIoT, we review the application of transfer learning in other CPS systems. In addition to IIoT, typical examples of CPS can include smart grids, smart health, smart homes, and smart manufacturing, among others [7]–[15]. Regarding the smart grid, a number of research efforts have been conducted using transfer learning to predict energy consumption [16], [17]. For example, Elena *et al.* [18] proposed unsupervised energy prediction in the smart grid using reinforcement cross-building transfer learning. For the smart transportation system, transfer learning has been used to perform vehicle classification and traffic prediction [19]–[21]. For example, Reza *et al.* [22] developed a transfer learning model for the classification of truck body types based on image data, which can achieve an overall accuracy of 96.5 %. For the smart home, transfer learning has been used for carrying out daily-activity recognition [23]–[26].

We now review the applicable scenarios of transfer learning specifically for component recognition in IIoT systems. In this category, most existing research has focused on optimizing the accuracy of component recognition [27], [28]. For example, Lee and Yang [29] used speeded up robust feature (SURF) [30] to correctly recognize and count industrial components. Chen *et al.* [31] designed a classic deep learning-based component recognition algorithm called Fast R-CNN [32] and achieved an overall 68 % component recognition accuracy. Likewise, Cong *et al.* [33] designed a new $k - d$ tree-based method for 3-D texture-less component recognition. Note that in the field of IIoT component recognition, most of the work of transfer learning has focused on recognizing various industrial components via image data captured by sensors.

Compared with other existing works, in this paper, we do not focus on tuning neural network parameters for specific situations. Instead, we categorize the entire IIoT machine learning application space into four types of scenarios and design general workflows for the different types of scenarios. Further, we demonstrate the efficacy of our designed transfer learning approach in different scenarios using a real-world IIoT dataset, as compared to an existing approach based on classical CNN.

III. Preliminary

In this section, we introduce the background on IIoT, machine learning, and transfer learning.

IIoT: IIoT uses network infrastructure to connect IoT devices (e.g., sensors and actuators) to automate industrial production processes. These IoT devices can assist in advanced industrial functions, including self-monitoring and self-diagnosis, among others. From the perspective of CPS, the IIoT system consists of cyber and physical subsystems. The synergy of subsystems will lead to a new generation of industrial production systems. With a variety of data collected by IIoT sensors, the cyber subsystem can extract the status of the system and generate proper control signals for IIoT actuators. For application scenarios, such as computing and control, processes can be executed in a centralized (i.e., central computing center) or distributed (i.e., local server/device) manner [34]. A central computing center is usually expensive and has larger computing capabilities, while local servers/devices are generally cheaper and have smaller computing capabilities. In this paper, we consider transfer learning in both centralized and distributed settings. Note that in different IIoT scenarios, the places where deep learning can be deployed are different, and the computing and storage capabilities of the devices that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3062482, IEEE Internet of Things Journal

are used to conduct deep learning tasks are different. Thus, how to properly deploy advanced deep learning algorithms (e.g., transfer learning) on IoT devices in different IIoT systems is a challenging issue, which may be solved from various directions.

Deep Learning: Deep learning techniques have been proven to have excellent performance in regression, classification, and prediction. To implement the advanced IIoT functions (e.g., self-diagnosing) mentioned above, deep learning techniques are viable. Deep learning uses multi-layer neural networks to find hidden relationships and features between a variety of data. Nonetheless, to achieve highly accurate results under the equivalent deep learning models, sufficient data and computing power are highly necessary.

Transfer Learning: Transfer learning is a machine learning technology that enables the transfer of knowledge from a well-trained machine learning model to a new machine learning model. To properly leverage transfer learning techniques, several prerequisites must be satisfied. First, the tasks that the original model and the new model are trained to achieve must be related. Second, the data used to train the original model and the data used to train the new model must have some common features. After applying the transfer learning techniques, the new model can use a small amount of data as a training dataset and obtain good performance. Recently, a number of research efforts have shown that transfer learning techniques have broad application across many domains [35].

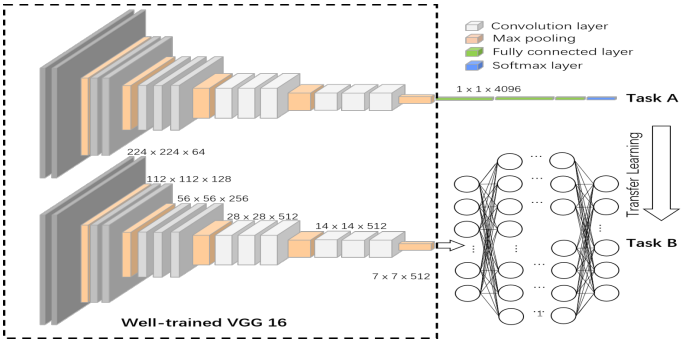

Fig. 1. Transfer learning

IV. Our Approach

In this section, we present our transfer learning approach to improve the performance of deep learning-based component recognition in IIoT systems.

A. Overview

A deep neural network that can carry out highly accurate component recognition often requires huge labeled datasets and a powerful computing server. Nonetheless, in the IIoT system, the computation capacities of IIoT devices are severely limited [36]. Further, there is no publicly labeled dataset that includes the most common industrial components. Thus, we leverage the transfer learning technique to improve the performance of machine learning-based component recognition in IIoT systems.

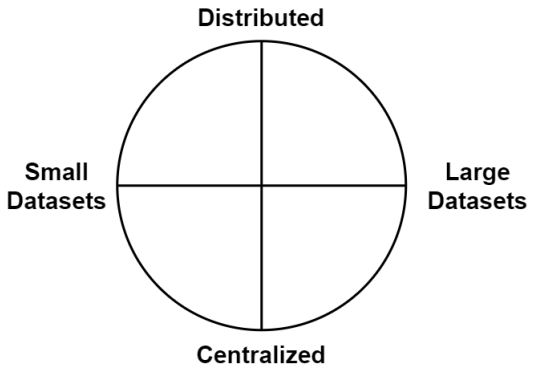

Fig. 2. Problem space of transfer learning use in IIoT

In this study, we propose a general framework to apply transfer learning in IIoT systems. Specifically, we transfer knowledge from other component recognition tasks to models that will conduct industrial component recognition. We categorize the use of transfer learning in IIoT systems into four different scenarios and design a detailed workflow on how to apply transfer learning to each scenario. We further evaluate the performance of the transfer learning-based approach under different scenarios. As a result, compared to an existing approach based on classical CNN [37], our approach can achieve higher component recognition accuracy in IIoT systems while requiring less training time.

Recall that transfer learning is a type of machine learning technique that transfers knowledge from one solved problem to another related problem [38]. In our case, we transfer some knowledge learned from object recognition on ImageNet [39] to component recognition in IIoT systems. Fig. 1 shows how to use the well-trained VGG-16 model for the transfer learning process. Note that VGG-16 is a known convolutional neural network model that uses the VGGNet structure. Here, number 16 represents that it has 16 layer networks. As can be seen in the figure, VGG-16 consists of 5 convolutional blocks and a full classifier. In each convolutional block, several convolutional layers (gray blocks) connect together to extract the features of an image. A max-pooling layer (orange blocks) is connected to the end of each convolutional block. A classifier that consists of 3 full connection layers (green blocks) and 1 softmax layer (blue block) is attached to the last convolutional block to generate the prediction result. For the transfer learning process, we replace the classifier of the original VGG-16 model with our proposed classifier. We use the convolutional block in VGG-16 as a feature extractor to pre-process images and extract relevant features about industrial components. Finally, we train this new integrated model with the selected industrial datasets, while convolutional layers of the pre-trained model are frozen during training.

B. Problem Space

Applying machine learning in IIoT systems requires consideration for the characteristics of the IIoT system itself, especially when considering the use of transfer learning. For machine learning, the two most important components are datasets and models. In IIoT application scenarios, the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3062482, IEEE Internet of Things Journal

datasets for machine learning could be various data collected by different sensors. Considering the heterogeneity of sensors, the quantity and quality of data generated by sensors will vary. The machine learning models can be deployed on either centralized or distributed computing servers. Generally speaking, centralized processing servers have relatively large computing capability, while local computing servers have relatively small computing capacity. Thus, when we apply transfer learning into IIoT systems, we should consider where the transfer learning process should be deployed.

In this study, we propose a general framework to adopt transfer learning in different IIoT scenarios. As shown in Fig. 2, we consider the size of datasets as the x-axis and the location of transfer learning (centralized or distributed) as the y-axis to define the problem space. In the figure, we categorize the problem space into the following four scenarios: (i) *Scenario A: centralized transfer learning with large datasets*, (ii) *Scenario B: distributed transfer learning with large datasets*, (iii) *Scenario C: centralized transfer learning with small datasets*, and (iv) *Scenario D: distributed transfer learning with small datasets*. For a large dataset, we use all the images in the original dataset. Particularly, in the T-Less dataset, this means that there are around 1000 images in each category. For a small dataset, we only use a portion of the images from the original dataset. In our case, we randomly select 300 images for each category in the T-Less dataset.

To evaluate the efficacy of our approach, the dataset is one of the most important aspects of our study. We select the 6D pose estimation of texture-less objects (T-Less) as the dataset for training and testing the proposed transfer learning approach. The T-Less dataset includes 30 different industrial components that are captured by a fixed angle camera. The total dataset consists of 37,800 image files, equaling more than 10 GB. Each component includes 1260 '$.png$' image files and each image has a resolution of 480×480 pixels. Those images show different angles of each component with RGB channels. Example images are shown in Fig. 4.

Fig. 3. Examples of ImageNet dataset

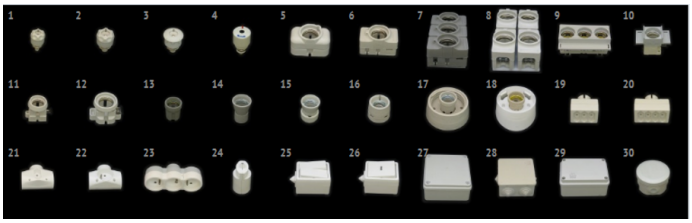

Fig. 4. Examples of T-less dataset

Based on the T-Less dataset and the source VGGNet model, we first reshape the images to 224×224 pixels. Then, we randomly divide the dataset into training and testing subsets of 1000 images and 260 images, respectively, for each component. After identifying the training and testing datasets, we label the datasets from '1' to '30' to denote the 30 different components. We create a small training dataset that includes 300 images. We leverage the small training dataset to evaluate the performance of transfer learning in an extreme situation. Meanwhile, the ImageNet dataset on which the VGGNet was trained includes at least one million images with more than 20 000 categories. Each category has an average of 1000 labeled RGB images. The categories cover a wide range, from animals to traffic signs.

C. Learning Models

In this study, two types of learning models have been designed. One is a transfer learning-based model. In this model, we use VGGNet as our source model from which to transfer knowledge. Then, we replace the output of the VGGNet with our own classifier, which consists of a few full connection layers. The other model is a light-weight classification CNN model-based on our prior work [37]. The CNN architecture is designed specifically for IIoT datasets and has been tuned for the T-Less dataset. We use multiple connected convolutional layers as the feature extractor and 2 full connection layers as a classifier.

VGGNet is a known CNN structure for image classification and object recognition [40]. VGGNet uses multiple convolutional blocks and a classifier with two full connection layers that represent image features and one full connection layer for classification purposes. VGGNet applies small filter size such as 3x3 and 1x1 at the convolutional layer to capture the most basic information. At the same time, VGGNet adopts a much deeper network structure than many other CNNs. As a result, the performance of VGGNet is significantly improved by combining small filters and deep network structure [41], [42]. As the pre-trained VGG-16 Model on ImageNet has been proven to have good generality on other datasets [43], [44], we use it as a source model for transfer learning.

The architecture of the VGG-16 model that we use in this study is shown in Fig. 1. The size of the input image is fixed as 224×224, and the intensity of each pixel of the image is rescaled to the range of (0,1) for better learning accuracy. Then, the input image is sent to the convolutional layer, in which only a small filter size (typically 3×3) is used in VGGNet for the convolution operation. The convolution process is defined as $C(m,n) = (M * w)(m,n) = \sum_k \sum_l M(m-k, n-l)w(k,l)$, where the size of image M is $m \times n$ and the size of filter w is $k \times l$.

The stride is taken as one with zero paddings to maintain the size of the feature maps that are generated by the convolution process. The extracted features are sent to the pooling layer to reduce the dimensionality. In addition, the pooling layer can create invariance for small shifts and deformations of the extracted features. After a few rounds of the convolution process, max-pooling layers are added after the convolutional layers. By doing this, it can further reduce the dimensionality of extracted features and reduce computational time. The stride for the max-pooling layer is set to 2 with a 2×2 window size. Note that the max-pooling process could also be beneficial for extracting superior invariant features, which could lead to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3062482, IEEE Internet of Things Journal

higher classification accuracy. As we can see in Fig. 1, there are multiple convolutional layers before the pooling layer. We consider this structure as a convolutional block, and after 5 convolutional blocks, 3 full connection layers are added as a classifier. The first two layers have 4,096 neurons and the third, also known as the Softmax layer, has 1000 neurons matching the 1000 classes of ImageNet detection targets. All the hidden layers inside this VGGNet are applied with activation function ReLU, which favorably increases the non-linearity of the network.

The function for Softmax is $softmax(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$, where x_i indicates the score inferred by the model for a specific class i and x_j indicates the source inferred by the model for all other classes. The activation function ReLu is shown as follows: $f(y) = max(0, y)$, where y is the input of a neuron. The ReLu activation function simply passes the positive part of the input. Also, as we classify multiple classes, the loss function that we use is categorical cross-entropy, which is a softmax activation plus a cross-entropy loss. The categorical cross-entropy can be derived by $CE = -log\left(\frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}\right)$, where, x_i indicates the score inferred by the model for a specific class i and x_j indicates the source inferred by the model for all other classes.

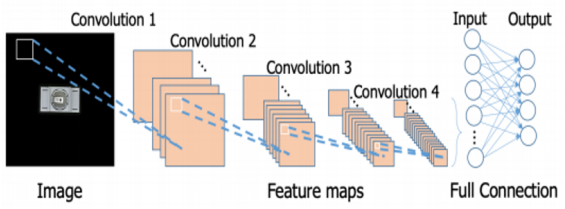

Fig. 5. CNN structure

The classical CNN architecture that we use in this study is based on our prior work [37], shown in Fig. 5. This classical CNN model consists of 4 convolutional layers and the filter that we use is 8×8. We deploy a pooling layer in the model following each convolutional layer (i.e., the total number of pooling layers is 4). The convolutional layers and pooling layers are used for feature extraction and reduction. The hidden layers also use ReLu as the activation function. In the lower end, we add 2 full connection layers to classify the input images. The activation function that we use at the second full connection layer is also softmax. Since we use classical CNN for the same task, we use categorical cross-entropy as our loss function. Further detail of this light-weight classical CNN model can be found in our prior work [37].

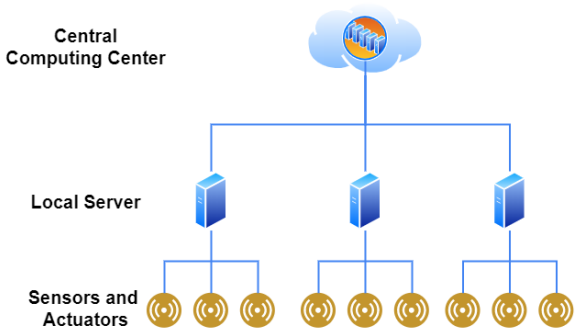

Fig. 6. System model

D. Transfer Learning

In the following, we first introduce the basic scheme of transfer learning. Then, we present the details of applying transfer learning to the defined scenarios. Recall from Fig. 2 that we consider four scenarios to which we apply transfer learning for IIoT systems.

1) Basic Scheme: Based on the principle of transfer learning [45], we now explain how to apply transfer learning to IIoT systems. First, we present the motivation for using transfer learning in IIoT systems.

One of the primary reasons for using transfer learning in IIoT is that the size of IIoT datasets is relatively small and unlabeled. Especially compared to the millions of labeled images in the ImageNet dataset, we consider fewer than 1000 images of each industrial component (only 300 images in our most constrained cases). In this case, it is not suitable to train a deep neural network such as VGGNet from random initialization, because the training data is not sufficient. Moreover, it would take considerable time and effort to expand this dataset and manually label new images. Another reason for using transfer learning is that the IoT devices in IIoT systems may not satisfy the high computing performance requirements to train new deep learning models from scratch. As the depth of the network increases, more time and computing resources are required. At the same time, the IoT devices deployed in the IIoT system are usually cheap and have limited computing power. To train a new model from scratch, either distributed or centralized, would take a significant amount of time and resources, and is likely not feasible on the IoT devices themselves. Taking these reasons into consideration, we propose the use of transfer learning to reduce the requirements of training dataset size, as well as the requirements for computing and storage of IoT devices, when training deep learning models in IIoT.

To describe the transfer learning, we consider that a domain $D = (X, P(X))$ consists of a feature space X and a marginal probability distribution $P(X)$. A task $T = (Y, f(.))$ also consists of two components: a label space Y and the prediction function $f(.)$. Transfer learning aims to improve the prediction function in the target task T_T by using knowledge in the source domain D_S and the source task T_S, where source domain D_S is not equal to target domain D_T, and the source task T_S is also not equal to target task T_T. In our case, the source domain is the ImageNet dataset and the target domain is the T-Less dataset. The source task is to predict which of the 1000 categories in ImageNet that the input images belongs to, while the target task is to predict which of the 30 categories in T-Less the input image belongs to.

Next, we describe how to apply transfer learning in deep CNN-based machine learning models. In the deep CNN, the features extracted from shallower layers normally represent low-level features, such as color and edges. The features

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3062482, IEEE Internet of Things Journal

extracted from deeper layers represent more complicated high-level features. We thus use these extracted features to train a more accurate classifier. As we can see in Fig. 1, in our study, we replace the classifier (3 full connection layers) in VGGNet that we mentioned in Section IV-C with another classifier. This new classifier consists of 3 full connection layers, where the first two have 4096 neurons and 2048 neurons, and the last full connection layer has 30 neurons as we only have 30 classes in the T-LESS dataset.

We use Fig. 6 to show the system model to implement our proposed transfer learning approach. As the devices of IIoT systems, sensors and actuators are located at the bottom of the figure. The sensors collect data and sent it to the local server. The local servers are located in the middle of the figure. The local server can serve as not only the gateway of the IIoT system, but also a small processing center. This local server can store a certain amount of collected data for simple processing and then send the data to the central computing center. The central computing center is at the top of the figure. It has high computing capability and is responsible for complex processing such as industrial control algorithms. After receiving data from local servers, the central computing center generates the control commands and sends these control commands back to the local server. Finally, the local server distributes these commands to individual actuators and the actuators then execute these received commands. We use this system model to illustrate all four scenarios introduced in Section IV-B.

2) Scenario A: Centralized Transfer Learning with Large Datasets: In this scenario, as the distributed local servers may not have sufficient computing capability to execute the transfer learning process, the transfer learning algorithm will be implemented in the central computing center. The general workflow for applying transfer learning in this scenario is shown in Fig. 7. First, various sensors collect data as training data and send it to the local server, which then forwards the data to the central computing center. Finally, the central computing center performs transfer learning.

The transfer learning technique in this scenario follows the basic scheme that we introduced before. The central computing center has already stored the well-trained VGGNet model. Recall from Fig. 1 that, to perform transfer learning, the central computing center replaces the last three full connection layers with a new classifier. This new classifier can have a totally different structure from the original VGGNet classifier. In our case, the new classifier consists of 3 full connection layers, the first layer has 4096 neurons, the second layer has 2048 neurons, and the last output layer has 30 neurons. We use categorical cross-entropy as our loss function as we have multiple classes. Then, we use the collected data from sensors to train the transferred model. During the training process, only the weights of the last 3 full connection layers will be updated, all the other weights will remain the same. By doing this, we leverage the well-trained VGGNet as a feature extractor and only train the classifier.

3) Scenario B: Distributed Transfer Learning with Large Datasets: In some IIoT systems, distributed local servers could have sufficient computing capacity to handle the transfer learning process and have enough storage capacity to store a large amount of data. The general workflow for applying transfer learning in such a scenario is shown in Fig. 8. Similar to scenario A, the sensors collect data first and transmit it to the local servers, which forward the data to the central computing center. In this scenario, the central computing center collects the data from multiple local servers and forwards the full dataset back to the local servers. The local serves receive the data and conduct the transfer learning process. Finally, the local servers send control decisions to the actuators.

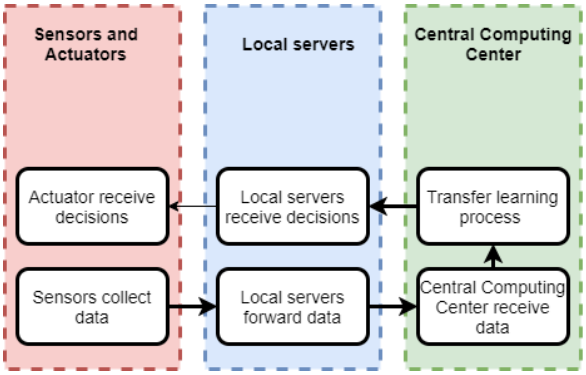

Fig. 7. Workflow of scenarios A and C

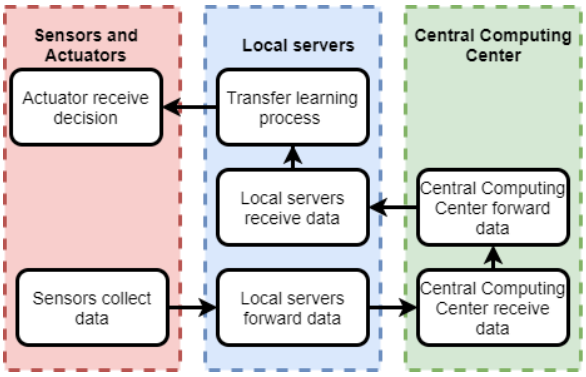

Fig. 8. Workflow of scenario B

Note that in this scenario, the transfer learning technique remains the same as the one that we use in scenario A. The only difference is that the transfer learning process will be executed at the local servers instead of the central computing center. Note that this scenario requires the local servers to have high computing and storage capacities. The local servers need to have enough space to store all the data and sufficient computing capability to complete the machine learning process. Note that, as we have reduced the weights that need to be trained in the deep neural network, our transfer learning approach can reduce the computing power requirements on the local servers compared to the classical CNN-based approach [37]. Nonetheless, the local servers still need a large storage space to store all the data.

4) Scenario C: Centralized Transfer Learning with Small Datasets: In some IIoT systems, the data that sensors can collect may be limited due to their hardware. The amount of data transmitted to the central server is limited by the bandwidth of the networking system in this scenario. Thus, in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3062482, IEEE Internet of Things Journal

some IIoT systems, there is not much data that can be used to train deep neural networks. To deal with this issue, we consider the scenario, in which only a small amount of dataset can be used to train the neural network model. In our case, while the T-less dataset is significantly smaller than ImageNet dataset, it still has around 30,000 samples. To further investigate the impact of such an extremely small data scenario, we train our model with only 9000 images and examine the performance impact of transfer learning. We randomly select 300 images per class from original dataset as the new training dataset. The difference between scenarios A and C is the size of data. Compared to scenario A, scenario C has fewer data to train the neural network. Using transfer learning in scenario C can bring greater benefits than using transfer learning in scenario A as transfer learning has a greater performance benefit on small datasets. Note that the workflow for applying transfer learning with small datasets is the same as applying transfer learning with large datasets.

5) Scenario D: Distributed Transfer Learning with Small Datasets: The last scenario is applying distributed transfer learning with small datasets. Fig. 9 illustrates the workflow in such a scenario. First, the sensors collect data and send it to the local servers. Then, the local servers directly conduct the transfer learning and send classification results to the central computing center. The central computing center collects classification results from multiple local servers and makes the final decision. The central processing center makes the final classification decision via the majority voting mechanism. Specifically, each local server makes a classification decision to determine the category that the current industrial component belongs to. The majority voting mechanism treats these predictions equally and relies on the most votes as the final classification result. Then, the central computing center sends the final classification decision to local servers, and the local server forwards this decision to the actuators.

Note that the transfer learning use in this scenario is different from the other scenarios. In other scenarios, only one transfer learning model is trained, while in this scenario, multiple transfer learning models are trained at different local servers simultaneously. The transfer learning techniques still use the basic scheme that we discussed before. Nonetheless, in this scenario, each local server uses a completely different classifier to replace the original classifier in VGGNet. Specifically, we can use transfer learning to design multiple deep learning models with different classifiers. These deep learning models all use VGGNet as the feature extractor. The new classifiers of these deep learning models have different numbers of full connection layers and neurons. Each local server is assigned one of these deep learning models. In the training phase, each local server uses a different learning rate to train its own deep learning model. In this way, we make the configuration of the transfer learning model as dissimilar as possible. When we use the majority vote mechanism to make the final classification decision, the diversity of transfer learning models can make the final classification result more accurate and stable.

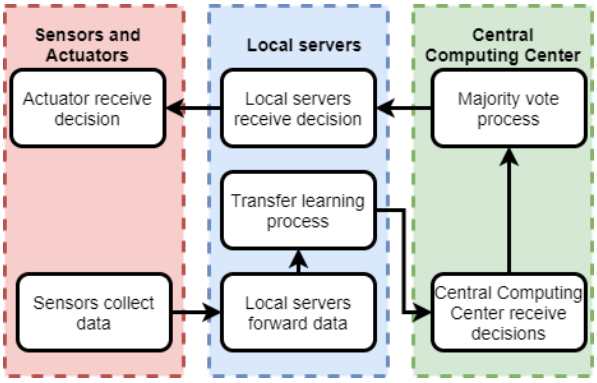

Fig. 9. Workflow of Scenario D

Model Structure	Average Accuracy
Transfer Learning with full dataset	98.6%
Classical CNN with full dataset	94.2%
Transfer Learning with small dataset	97.6%
Classical CNN with small dataset	92.1%
Distributed transfer learning with small dataset	99.1%

TABLE I. Learning accuracy of different learning models

V. Performance Evaluation

We have conducted extensive experiments to validate the efficacy of our transfer learning approach in different scenarios compared to an existing traditional approach.

A. Methodology

Recall the four scenarios that we discussed in Section IV-D. Scenarios A and B focus on applying transfer learning with large datasets in a central and distributed manner, respectively. For different IIoT systems, the computing capabilities of the central computing center and the local server can be the same or different. In this work, we focus on the general IIoT transfer learning framework, rather than the impact of different computing capabilities on machine learning algorithms. Thus, in our experiments, we consider that all the central computing centers and local servers have the same computing capability. For those who are interested in IoT machine learning optimization under servers with different computing capabilities, please refer to our prior work [37].

We have conducted several performance comparisons to evaluate the efficacy of our designed transfer learning approach on these four IIoT scenarios. Since the dataset itself has a great impact on the final results of the machine learning algorithm, we compare our proposed approach with the existing classical CNN-based approach [37], which uses the same IIoT dataset. In detail, scenarios A and B have a complete T-Less dataset as the training data, and the difference between these two scenarios is whether the transfer learning model is deployed in the central computing center or the local server. Thus, our first comparison focuses on accessing the accuracy of the classical CNN-based approach and our transfer learning-based approach for the same IIoT component recognition task with large data. For scenario C, the central computing center only has a small dataset for training. We evaluate the impact of data size on the accuracy of transfer learning by comparing

the accuracy of the classical CNN-based approach and our transfer learning-based approach for the same IIoT component recognition task with small datasets. Finally, for scenario D, the transfer learning models have a small dataset for training and the transfer learning models are deployed at multiple local servers. Thus, we compare the performance of transfer learning in distributed and centralized settings for the same IIoT component recognition task with small datasets.

To evaluate the performance of different transfer learning approaches, we randomly divide the IIoT dataset into a training set and validation set at a ratio of 8:2. Before each training process, we repeat the dividing process to obtain different training sets and validation sets. We train each model 20 times and compute the validation accuracy each time. In each training process, we train our models for 50 epochs. Recall that we use categorical-cross entropy as our loss function and categorical accuracy as evaluation metrics. We also record the timestamp of the training process such that we can further compare the training time of different scenarios. We have implemented these models with Python using open source library TensorFlow 2.0[1] [46]. We used Adam optimizer to minimize the loss function during training. All the experiments were performed on a Windows 10 Server with an NVIDIA GTX 1070ti GPU.

Our experiments consists of the following cases: (i) Transfer Learning vs. Classical CNN with Large Datasets, (ii) Transfer Learning vs. Classical CNN with Small Datasets, and (iii) Distributed Transfer Learning vs. Centralized transfer learning. In the first case, we use the original IIoT dataset, which consists of 30 categories, each category has 1000 images as the training set and 200 images as the validation set. In the second case, we use only 300 images for each class in IIoT dataset and examine the efficacy of our transfer learning approach. In the last case, we implement our transfer learning approach in the distributed setting on the same dataset that we use in the second case. The numerical results after training the model 50 epochs are shown in Table I.

B. Results

In the following, we present the evaluation results. Note that in all related figures, we use 'TL' to represent our transfer learning approach and 'CNN' to represent the existing classical CNN approach.

Transfer Learning vs. Classical CNN with Large Datasets: We compare the performance of industrial component recognition between our transfer learning-based approach and the classical CNN-based approach. We evaluate the performance from two aspects: the accuracy and the training time needed to reach the defined accuracy. In Fig. 10, we illustrate the categorical accuracy of the transfer learning-based approach and the classical CNN-based approach. As we can see in the figure, our transfer learning approach can improve the overall categorical accuracy on the full (1000 samples per category) datasets by around 4 %. In Fig. 11, we illustrate the loss of the transfer learning-based approach and the classical CNN approach. In Fig. 12, we compare the epochs required for each approach to reach the given accuracy. The solid line indicates the categorical accuracy of the test dataset for the transfer learning approach. The dashed line indicates the categorical accuracy of the training dataset for the transfer learning approach. The dotted line indicates the categorical accuracy of the test dataset for the classical CNN approach. Also, the dash-dot line indicates the categorical accuracy of the training dataset for the classical CNN approach.

For the classical CNN-based approach, we can see from the shadow area of the dotted line in Fig. 10 and Fig. 11 that the categorical accuracy of the training datasets fluctuates greatly. This is because there is not enough data to train the CNN from scratch. The CNN model seems to fit the training set; however, this model is not general enough. Thus, during the validation phase, the categorical accuracy is not stable.

For our transfer learning-based approach, the categorical accuracy of both the training datasets and validation datasets are quite stable. Also, we can see in Fig. 10 that, in our multiple experiments, the accuracy difference between different experiments is very small. This is because transfer learning leverages the knowledge that VGGNet learned from the ImageNet dataset. Specifically, the features that are used to complete the classification task in ImageNet can assist in completing the component classification task of the IIoT datasets. Moreover, the massive training data in ImageNet makes the model more versatile and powerful. Thus, our transfer learning-based approach achieves better and more stable performance.

We next compare the training times necessary for the transfer learning-based approach and the classical CNN-based approach to reach the desired learning accuracy as shown in Fig. 12. The left side represents how much time that the transfer learning-based approach needs to achieve the desired learning accuracy. The right side represents how much time the classical CNN-based approach needs to achieve the same level of learning accuracy. As we can see in Fig. 12, the transfer learning-based approach typically requires 2-3 epochs to reach an accuracy of 94 %, while the approach based on the classical CNN approach usually requires 30 epochs to achieve the same level of accuracy. Also, when the two models are trained on the same server, the classical CNN-based approach takes about 17 s each epoch, while our transfer learning-based approach only takes 3 s for each epoch. Specifically, to approach any pre-determined learning accuracy, our transfer learning-based approach always requires fewer training epochs than the classical CNN approach. In conclusion, the transfer learning-based approach is much faster than the classical CNN-based approach in reaching the defined learning accuracy (i.e., trains faster).

Transfer Learning vs. Classical CNN with Small Datasets: We next evaluate the performance of training our transfer learning model and the classical CNN model on extremely small datasets. Similar to the prior experiment,

[1]Certain commercial equipment, instruments, or materials are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3062482, IEEE Internet of Things Journal

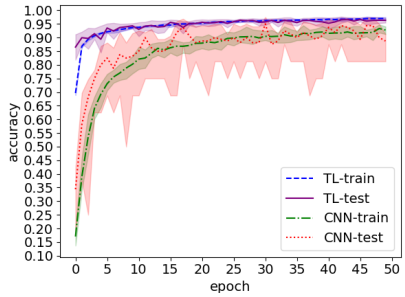

Fig. 10. Transfer Learning vs. Classical CNN

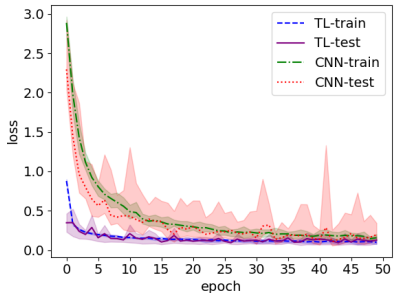

Fig. 11. Transfer Learning vs. Classical CNN (Loss)

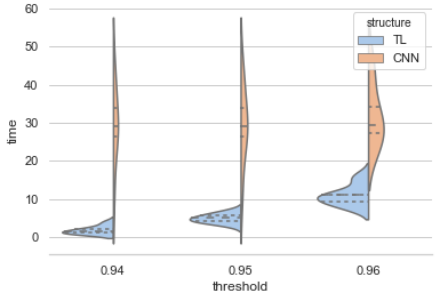

Fig. 12. Transfer Learning vs. CNN (Time)

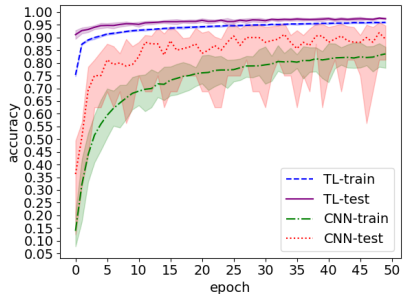

Fig. 13. Transfer Learning vs. Classical CNN

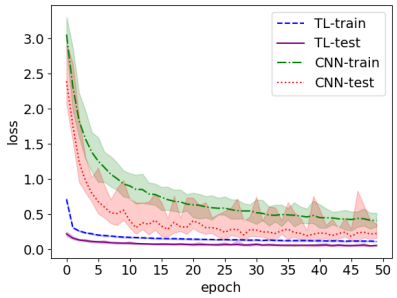

Fig. 14. Transfer Learning vs. Classical CNN on small datasets (Loss)

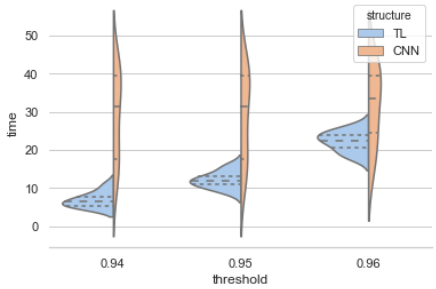

Fig. 15. Transfer Learning vs. Classical CNN on small datasets (Time)

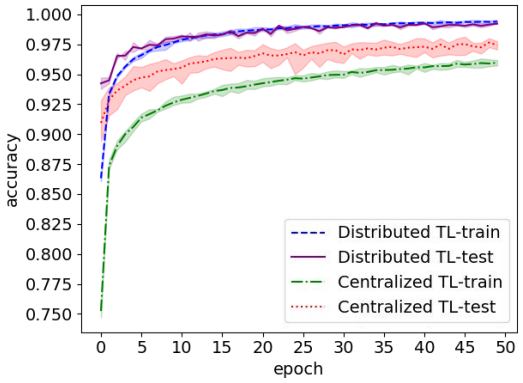

Fig. 16. Distributed Transfer Learning vs. Centralized transfer learning (Accuracy)

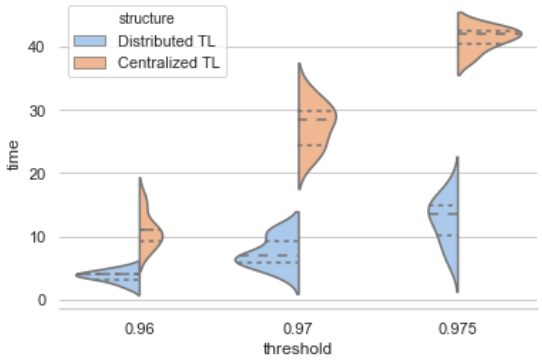

Fig. 17. Distributed Transfer Learning vs. Centralized transfer learning (Time)

we use the accuracy and the training time needed to reach a pre-defined learning accuracy as two metrics to evaluate performance. As we can observe in Fig. 13, when trained on extremely small datasets (300 training samples per category), the overall accuracy is reduced. The average validation accuracy for the transfer learning approach dropped by about 1 %, and the average validation accuracy for the classical CNN approach dropped by about 2 %. This is because there is not sufficient training data to train a complex neural network. In addition, the extremely small training dataset causes overfitting, where the model fits the training data too well, resulting in a negative impact on the validation accuracy. To address this issue, we add a heavy drop-out layer (drop-out of 0.5) on both the transfer learning and classical CNN approaches to induce extra noise during the training phase.

For the classical CNN-based approach, we can see from Figs. 13 and 14 that both training accuracy and validation accuracy have a higher variance than in the first evaluation scenario, in which the full dataset was used in the training process. Moreover, the validation accuracy also has a higher variance than training accuracy. There are a few explanations for this observation. First, we add a heavy dropout layer, the dropout layer will randomly disable some neurons from the previous layer to add more noise to the classical CNN model.

This can improve the generality of the model, and benefit the validation accuracy. Nonetheless, these random dropouts also increase the variance of the training accuracy. Second, as the size of the training dataset has been largely reduced, the classical CNN model becomes too complex to train. The model cannot extract the complex relations among the dataset with this amount of training data. As a result, the validation accuracy becomes unstable. The range of validation accuracy at the same epoch can be around 30 %. Another observation from the figure is that the average validation accuracy is higher than the training accuracy. This is also the result of using a heavy dropout layer. As the dropout layer will not work at the validation phase, the validation phased can fully utilize all the neurons in the previous layer to make predictions.

For the transfer learning-based approach, although the overall accuracy has been decreased, the variance of the accuracy has not observed the same impact as the classical CNN model. There are two explanations for this observation. First, transfer learning uses a pre-trained model as a feature extractor to generate a large number of general features as input for the classifier. The pre-trained VGGNet is trained with a very large ImageNet dataset so that the extracted features are more general than the classical CNN model. Even with heavy dropout, the remaining features are still sufficient to complete the recognition task. Second, the transfer learning-based approach trains a much simpler neural network. The transfer learning-based approach only needs to train the full connection layers. As the full connection layer is much simpler than the full classical CNN model, the transfer learning-based approach can achieve more stable performance with a small amount of data.

Comparing the time required for the two approaches to reach a pre-determined accuracy on the extremely small datasets, we present Fig. 15. The results show that both the transfer learning-based approach and the classical CNN-based approach take more epochs to reach the defined accuracy level than in the first scenario (full dataset). The transfer learning-based approach takes around 5 more epochs to reach 94 % accuracy, while the classical CNN-based approach needs around 10 more epochs to reach the same accuracy threshold. For other accuracy thresholds, we can see a similar observation. Nonetheless, no matter what the desired accuracy threshold, the transfer learning-based approach is still much faster than the classical CNN-based approach.

Distributed Transfer Learning vs. Centralized Transfer Learning: Finally, we compare the performance of our proposed transfer learning approach in a distributed setting versus a centralized setting, using extremely small datasets. As shown in Fig. 16, we illustrate the performance of our transfer learning approach in a distributed setting for scenario D. From the figure, we can see that distributed transfer learning approach improves the accuracy by about 1.5 %. At the same time, the performance of distributed transfer learning approach is also more stable. This is because we train multiple classifiers that have different structures. By combining multiple models to make the decision, we can further reduce the variance of recognition and achieve higher accuracy.

For different IIoT systems, the size of the data used to train a local transfer learning model is typically small considering the resource limitations of the local node and the availability of data. In contrast, for centralized transfer learning, the cloud computing center can have a much larger training dataset. In general, a larger training dataset will result a deep learning model in achieving higher accuracy in recognizing images. Nonetheless, it can be seen from Table I that, although three times the data is provided to the deep neural network, its accuracy is improved by only 1.0 %. Thus, the improvement of the recognition accuracy by the dataset size is non-linear. Once the accuracy of the deep learning model reaches a certain threshold, it becomes difficult to continue to improve the accuracy of the model simply by increasing the size of the data. Moreover, the centralized transfer learning model is trained with the data gathered from all local nodes, which results in the centralized transfer learning model having higher recognition accuracy than any single distributed transfer learning model.

Recall from Fig. 2 that we have designed four different scenarios. Based on the sizes of data in scenarios A through D, the comparison between CTL and DTL can be divided into two perspectives: (i) DTL vs. CTL with a large dataset, and (ii) DTL vs. CTL with a small dataset. With a large dataset, both local nodes and the central node can reach a very high accuracy in average. This means that each local node in DTL will have a similar performance to the central node in the CTL. Since there are multiple neural network models with high accuracy working together, the DTL will slightly outperform the CTL. Nonetheless, the local nodes in DTL will need much more time to perform the training process compared to the cloud computing center, considering the limited computation capacities of the local nodes. With a small dataset, the local nodes can achieve desirable accuracy within a small time period by applying the transfer learning techniques as shown in Fig. 17. Although each local node has a slightly worse performance than the central node, the DTL uses all the local nodes to make the decision. As a result, Fig. 16 shows that DTL has a higher prediction accuracy and smaller convergence time than CTL in a small dataset scenario.

We also compare the time required to reach a target accuracy for both approaches in scenario C (i.e., centralized transfer learning with small datasets) and scenario D (i.e., distributed transfer learning with small datasets). Because the lower accuracy threshold is reached by both approaches very quickly, it is hard to discern any difference. Thus, we focus on the higher accuracy threshold. As can be seen in Fig. 15, our distributed transfer learning approach takes about 2 epochs to reach the 95 % accuracy, while the centralized transfer learning approach needs around 10 epochs. As the accuracy threshold continues to increase, this difference becomes more pronounced. This also benefits from multiple models in the distributed transfer learning approach. Devices (e.g., sensors) in the IIoT system are typically distributed to different locations and connected via wireless networks. For centralized transfer learning, all training data gathered from sensors will be transmitted to the cloud computing center, resulting in data transmission latency calculated by the total amount of data divided by the data transmission rate. For distributed transfer learning, as IIoT devices conduct transfer learning locally, the learning result

is transmitted instead of the original data. Considering that the learning result is much smaller than the original data, the transmission latency of distributed transfer learning will be much smaller than that of centralized transfer learning.

VI. Discussion

The mechanism of transfer learning brings new directions for rapidly training deep neural networks in IIoT systems. We could extend our work from three perspectives: CPS, transfer learning algorithm design, and practical application, which have different focuses and can be leveraged in support of one another.

CPS: From the perspective of CPS, exploration of transfer learning task space should be considered. A typical CPS usually consists of both cyber and physical subsystems. In cyber subsystems, control, computing, and networking are essential components interacting with each other and adding new functions to traditional industrial systems (e.g. production automation, self-diagnosis, self-repair, etc.). At the same time, the control complexity of these new functions is also very high, and traditional optimization control methods cannot take into account all components and subsystems well. Machine learning has been increasingly applied to the optimization of complex systems due to excellent performance with high-dimensional variables [2]. One future direction is to continue to explore what complex control tasks in CPS can be optimized by machine learning techniques.

Transfer Learning Algorithm Design: From the perspective of transfer learning algorithm design, we should focus on customizing existing transfer learning algorithms for different application scenarios. Transfer learning has a number of branches. Different transfer learning algorithms are tailored to solve different problems. Due to the black-box nature of machine learning algorithms, it is difficult to express transferred knowledge. Thus, it is difficult to find theoretically optimal machine learning models for applying transfer learning in all environments. Nonetheless, we can still customize existing transfer learning algorithms according to specific application scenarios. Recall the discussion of the CPS perspective, after we discover valuable application scenarios in CPS, how to design the most suitable transfer learning algorithm for a designated scenario is an interesting problem to explore. Based on the research conducted in this paper, when designing a distributed transfer learning approach, we shall consider the computing limitations of IIoT devices and take advantage of the distributed nature of the IIoT system itself.

Under ideal conditions, an IIoT system should have high computing capabilities, network conditions, and data transmission rate. Since computation and transmission latency are very low, we can focus on designing and tuning the neural network to maximize its performance (i.e., accuracy) with a sufficient volume of data. Nonetheless, in some cases, the IIoT system may have limited computing and network resources. As such, the deep neural network training time and transmission latency will become key factors affecting the performance of the IIoT system. On one hand, if we train our deep neural network on local nodes that have low computing capacity, the training time will be large, but the transmission latency will be small since we do not need to transmit the data gathered from local nodes to the cloud computing center. On the other hand, if we decide to transmit all the data to the cloud computing center first, and train the deep neural network model in the cloud computing center, the transmission latency will be large, and the training time will be small. Thus, when we design deep transfer learning algorithms for IIoT systems with limited resources, we should consider how to reduce the combined training and transmission latencies according to the resource limitations of the specific IIoT system.

Practical Applications: From the perspective of practical applications, we should validate the feasibility of a variety of proposed approaches. After we design a transfer learning solution for a specific environment, we need a realistic platform to actually implement the proposed approach and evaluate its efficacy. Collecting and analyzing problems encountered in real-world scenarios can assist in improving the design of scenarios and algorithms. The complexity of the actual application environment is quite difficult to simulate. A realistic experimental platform that can reflect these complex interactions will greatly contribute to future research on applying transfer learning in CPS and IIoT systems. Further, federated learning is a distributed learning approach that can provide low latency and reduce power consumption in the model training process, especially in IIoT devices with limited resources. Federated learning shares some characteristics with our proposed transfer learning framework. Thus, we can integrate federated learning techniques into our proposed transfer learning framework to further improve the learning performance of IIoT systems. For example, in a realistic scenario with multiple IIoT systems, each system may operate its learning process using federated learning. The learning results from one system can be shared with another system via transfer learning.

VII. Final Remarks

In this paper, we proposed a general framework to adopt transfer learning in different IIoT scenarios. Based on the amount of data and the way that the transfer learning process is performed (either a centralized or distributed manner), we categorized the problem of applying transfer learning in IIoT systems into four scenarios. We also designed a general workflow for applying transfer learning in each scenario defined. To demonstrate the efficacy of our approach, we applied our approach to recognize the industrial components in a general IIoT system using the T-Less industrial dataset. Via comprehensive experiments, we demonstrate that, compared to the existing approach, our proposed approach can achieve faster learning with more stability with little training data, and does so in all scenarios.

References

[1] H. Xu, W. Yu, D. Griffith, and N. Golmie, "A survey on industrial internet of things: A cyber-physical systems perspective," *IEEE Access*, vol. 6, pp. 78 238–78 259, 2018.

[2] H. Xu, X. Liu, W. Yu, D. Griffith, and N. Golmie, "Reinforcement learning-based control and networking co-design for industrial internet of things," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 885–898, 2020.

[3] W. G. Hatcher and W. Yu, "A survey of deep learning: Platforms, applications and emerging research trends," *IEEE Access*, vol. 6, pp. 24 411–24 432, 2018.

[4] M. W. Libbrecht and W. S. Noble, "Machine learning applications in genetics and genomics," *Nature Reviews Genetics*, vol. 16, no. 6, pp. 321–332, 2015.

[5] K. Kaur, S. Garg, G. S. Aujla, N. Kumar, J. J. Rodrigues, and M. Guizani, "Edge computing in the industrial internet of things environment: Software-defined-networks-based edge-cloud interplay," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 44–51, 2018.

[6] T. Hodan, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-less: An rgb-d dataset for 6d pose estimation of texture-less objects," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2017, pp. 880–888.

[7] X. Liu, C. Qian, W. G. Hatcher, H. Xu, W. Liao, and W. Yu, "Secure internet of things (iot)-based smart-world critical infrastructures: Survey, case study and research opportunities," *IEEE Access*, vol. 7, pp. 79 523–79 544, 2019.

[8] G. Xu, W. Yu, D. Griffith, N. Golmie, and P. Moulema, "Toward integrating distributed energy resources and storage devices in smart grid," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 192–204, 2017.

[9] N. Y. Philip, J. J. P. C. Rodrigues, H. Wang, S. J. Fong, and J. Chen, "Internet of things for in-home health monitoring systems: Current advances, challenges and future directions," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 2, pp. 300–310, 2021.

[10] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[11] H. Wang, M. Daneshmand, and H. Fang, "Artificial intelligence (ai) driven wireless body area networks: Challenges and directions," in *2019 IEEE International Conference on Industrial Internet (ICII)*, 2019, pp. 428–429.

[12] Z. Cai and T. Shi, "Distributed query processing in the edge assisted iot data monitoring system," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[13] X. Liu, W. Yu, F. Liang, D. Griffith, and N. Golmie, "On deep reinforcement learning security for industrial internet of things," *Computer Communications*, vol. 168, pp. 20–32, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366420320193

[14] J. Pang, Y. Huang, Z. Xie, Q. Han, and Z. Cai, "Realizing the heterogeneity: A self-organized federated learning framework for iot," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[15] Q. Chen, X. Ma, S. Tang, J. Guo, Q. Yang, and S. Fu, "F-cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3d point clouds." Association for Computing Machinery, 2019.

[16] C. Fan, Y. Sun, F. Xiao, J. Ma, D. Lee, J. Wang, and Y. C. Tseng, "Statistical investigations of transfer learning-based methodology for short-term building energy predictions," *Applied Energy*, vol. 262, p. 114499, 2020.

[17] M. Ribeiro, K. Grolinger, H. F. ElYamany, W. A. Higashino, and M. A. Capretz, "Transfer learning with seasonal and trend adjustment for cross-building energy forecasting," *Energy and Buildings*, vol. 165, pp. 352–363, 2018.

[18] E. Mocanu, P. H. Nguyen, W. L. Kling, and M. Gibescu, "Unsupervised energy prediction in a smart grid context using reinforcement cross-building transfer learning," *Energy and Buildings*, vol. 116, pp. 646–655, 2016.

[19] P. Zhou, L. Chen, X. Dai, B. Li, and T. Chai, "Intelligent prediction of train delay changes and propagation using rvflns with improved transfer learning and ensemble learning," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[20] P. Krishnakumari, A. Perotti, V. Pinto, O. Cats, and H. van Lint, "Understanding network traffic states using transfer learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 1396–1401.

[21] S. Y. Jo, N. Ahn, Y. Lee, and S.-J. Kang, "Transfer learning-based vehicle classification," in *2018 International SoC Design Conference (ISOCC)*. IEEE, 2018, pp. 127–128.

[22] R. V. Nezafat, B. Salahshour, and M. Cetin, "Classification of truck body types using a deep transfer learning approach," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 3144–3149.

[23] W. Wang and C. Miao, "Activity recognition in new smart home environments," in *Proceedings of the 3rd International Workshop on Multimedia for Personal Health and Health Care*, 2018, pp. 29–37.

[24] J. Wang, V. W. Zheng, Y. Chen, and M. Huang, "Deep transfer learning for cross-domain activity recognition," in *proceedings of the 3rd International Conference on Crowd Science and Engineering*, 2018, pp. 1–8.

[25] S. Ramasamy Ramamurthy and N. Roy, "Recent trends in machine learning for human activity recognition—a survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1254, 2018.

[26] L. A. Euprazia and K. Thyagharajan, "A novel action recognition system for smart monitoring of elderly people using action pattern image and series cnn with transfer learning," *arXiv preprint arXiv:2009.03285*, 2020.

[27] T. Czerniawski, M. Nahangi, S. Walbridge, and C. Haas, "Automated removal of planar clutter from 3d point clouds for improving industrial object recognition," in *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, vol. 33. IAARC Publications, 2016, p. 1.

[28] Y. Perez-Gallardo, J. L. L. Cuadrado, Á. G. Crespo, and C. G. de Jesús, "Geodim: a semantic model-based system for 3d recognition of industrial scenes," in *Current Trends on Knowledge-Based Systems*. Springer, 2017, pp. 137–159.

[29] S.-H. Lee and C.-S. Yang, "A real time object recognition and counting system for smart industrial camera sensor," *IEEE Sensors Journal*, vol. 17, no. 8, pp. 2516–2523, 2017.

[30] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[31] X. Chen and J. Guhl, "Industrial robot control with object recognition based on deep learning," *Procedia CIRP*, vol. 76, pp. 149–154, 2018.

[32] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[33] Y. Cong, D. Tian, Y. Feng, B. Fan, and H. Yu, "Speedup 3-d texture-less object recognition against self-occlusion for intelligent manufacturing," *IEEE transactions on cybernetics*, vol. 49, no. 11, pp. 3887–3897, 2018.

[34] M. Ghobakhloo, "The future of manufacturing industry: a strategic roadmap toward industry 4.0," *Journal of Manufacturing Technology Management*, vol. 29, pp. 910–936, 2018.

[35] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, p. 9, 2016.

[36] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.

[37] F. Liang, W. Yu, X. Liu, D. Griffith, and N. Golmie, "Toward edge-based deep learning in industrial internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4329–4341, 2020.

[38] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, 2020.

[39] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[41] J. Han, D. Zhang, G. Cheng, N. Liu, and D. Xu, "Advanced deep-learning techniques for salient and category-specific object detection: a survey," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 84–100, 2018.

[42] A. K. Gupta, "Survey of visual question answering: Datasets and techniques," *arXiv preprint arXiv:1705.03865*, 2017.

[43] S. Khan, N. Islam, Z. Jan, I. U. Din, and J. J. C. Rodrigues, "A novel deep learning based framework for the detection and classification of breast cancer using transfer learning," *Pattern Recognition Letters*, vol. 125, pp. 1–6, 2019.

[44] P. M. Cheng and H. S. Malhi, "Transfer learning with convolutional neural networks for classification of abdominal ultrasound images," *Journal of digital imaging*, vol. 30, no. 2, pp. 234–243, 2017.

[45] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[46] "Tensorflow," https://www.tensorflow.org/.