Discover

Research

Intelligent IoT systems for civil infrastructure health monitoring: a research roadmap

E. Bertino[1] · M. R. Jahanshahi[2] · A. Singla[1] · R.-T. Wu[2]

Received: 30 November 2020 / Accepted: 28 January 2021
© The Author(s) 2021 OPEN

Abstract
This paper addresses the problem of efficient and effective data collection and analytics for applications such as civil infrastructure monitoring and emergency management. Such problem requires the development of techniques by which data acquisition devices, such as IoT devices, can: (a) perform local analysis of collected data; and (b) based on the results of such analysis, autonomously decide further data acquisition. The ability to perform local analysis is critical in order to reduce the transmission costs and latency as the results of an analysis are usually smaller in size than the original data. As an example, in case of strict real-time requirements, the analysis results can be transmitted in real-time, whereas the actual collected data can be uploaded later on. The ability to autonomously decide about further data acquisition enhances scalability and reduces the need of real-time human involvement in data acquisition processes, especially in contexts with critical real-time requirements. The paper focuses on deep neural networks and discusses techniques for supporting transfer learning and pruning, so to reduce the times for training the networks and the size of the networks for deployment at IoT devices. We also discuss approaches based on machine learning reinforcement techniques enhancing the autonomy of IoT devices.

Keywords Autonomous IoT devices · Deep Neural Networks · Analytics at the Edge

1 Introduction

In future smart cities, many decision processes in critical infrastructure and emergency management will be based on machine learning (ML) techniques. One particular application is the processing of large datasets of visual images and other types of data for defect assessment where the data is collected by a swarm of IoT devices (devices, for short), some of which can be mobile, e.g., small unmanned aerial vehicles (UAVs) and robots. In this context, examples of defective regions are corrosion and cracks in buildings and facilities [1], and potholes on roads. A critical requirement for the success of such assessment processes is the reliable detection, quantification and localization of defective regions. Furthermore, in such applications, a real-time assessment is often critical so that the swarm can decide regarding the optimum strategy and corresponding actions for effective data collection in unknown environments, e.g., robots used for earthquake reconnaissance and rescue where they enter buildings whose plan is unknown to the robots. On the other hand, for such assessments to be reliable it is critical that data be of good quality since poor data may negatively affect the accuracy of classification and predictions, and consequently may introduce additional costs and time overhead.

✉ E. Bertino, bertino@purdue.edu; M. R. Jahanshahi, jahansha@purdue.edu; A. Singla, asingla@purdue.edu; R.-T. Wu, jwu952@purdue.edu | [1]CS Department, Purdue University, West Lafayette, Indiana, USA. [2]Lyle School of Civil Engineering, Purdue University, West Lafayette, Indiana, USA.

Published online: 24 February 2021

In general, acquiring data and making sure that the data is of high quality, especially for real-time decisions, is expensive due to difficulties in reaching the regions where the objects of interest are located and the need for human-intensive assessment. However, today we have many technologies that can be leveraged to build effective and inexpensive solutions, including: deep neural networks for image analysis; image processing techniques; devices able to acquire images and other types of data; 5G networks and edge computing processing [2]; crowd-sourcing [3]. Mobile devices such as drones are becoming quite powerful in terms of the sensing capabilities they offer—as an example the nano Black Hornet 3 drone is equipped with a microcamera core and a visible sensor to allow for enhanced image fidelity [4].

The use of devices for data acquisition, of course, is not new and almost all application domains we may think of use these devices, including civil infrastructures, smart cities, smart agriculture, emergency management, and environmental protection. However, the common practice is to use the devices just as a data collection means. Data is acquired by devices and then transmitted to some centralized large server, such as a cloud server, for processing and analysis. Such an approach may not always be optimal in many situations. Intermittent communications and communication disruptions, as in the case of battlefield and emergency management scenarios [5], may make it difficult to transmit data to a centralized server. In addition, in many such scenarios, it is critical to quickly analyze the data and, based on the analysis results, determine whether additional data needs to be collected or specific actions executed. For example, consider the case of the collapse of the Morandi Bridge in Genoa [6]. In such a scenario, being able to quickly detect anomalies in a bridge and notify incoming vehicles would save human life and a few seconds may make the difference. Approaches by which data has to be sent to a cloud server would not be viable.

Another important consideration is that for many applications it is critical to ensure that data be of good quality [7]. For instance, in the case of image data, it is important that the image of an object of interest (e.g., a crack on a wall) is not occluded. Completeness (e.g., making sure that no relevant data is missing) is equally critical. For example, if a failure is detected on one side of a structure, it is critical to determine whether the failure extends to the other sides of the structure. Current assessments regarding the quality and completeness of acquired data requires not only to send the data to remote servers, but often also to involve human analysts to evaluate the data and provide further data acquisition instructions to the remote devices. This approach is not effective and requires extensive real-time human involvement. Furthermore, this approach would not be scalable as the numbers of data acquisition devices increase. Additionally, as new phenomena of interest arise in the application domain of interest, it is critical to reduce the time required to deploy the needed data analytic solutions. For example, machine learning approaches that require training labeled data sets that are expensive and/or time-consuming to collect and label may not be effective for many scenarios. We thus need approaches by which:

- Devices can directly perform analyses on the collected data as, when there are real-time requirements, transmitting the analysis results requires significantly less communication bandwidth and, in addition, may allow the devices to quickly send high-priority safety information to humans, vehicles, and other parties.
- Devices can autonomously decide which data to collect based on data they have already collected and locally analyzed.
- The time required for generating analysis models deployed at the edge is minimized as much as possible.

In this paper, we discuss approaches addressing the above three requirements and outline a research roadmap. We focus on data analysis techniques based on deep convolutional neural networks (CNNs) and investigate the use of transfer learning (TL), so to minimize the time/cost for training the networks, and CNN pruning, so to reduce the size and inference costs of the networks, for deployment at devices. We discuss also approaches supporting autonomous and adaptive data collection based on the use of reinforcement learning (RL).

2 Analytics at devices

In this section, we focus on the problem of efficiently training CNNs and deploying them at devices. We first summarize initial results from one of our projects. We then briefly discuss related work and outline open research directions.

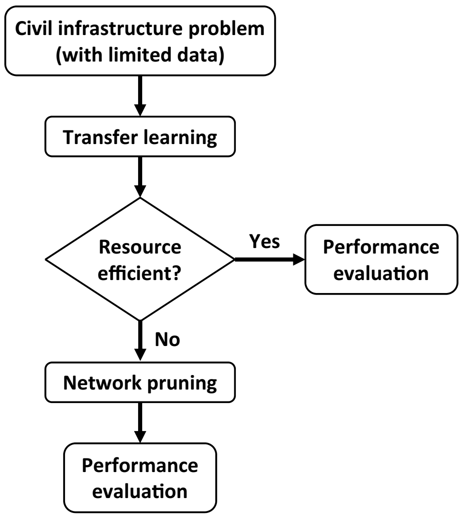

Fig. 1 Framework for device-based inspection system

2.1 Initial results

Figure 1 show the high-level steps of our approach for device-based failure detection [8]. Since typically multiple types of damage exist in infrastructures (e.g., cracks, corrosion, spalling, exposed rebars, etc. on a concrete surface), it is often very difficult and/or expensive to acquire and label sufficient data for network training. To address this limitation, we have adopted TL where a pre-trained deep CNN is used to detect a new type of damage. TL is a very popular choice for vision-based infrastructure assessment, since it requires less training data compared to when a CNN is trained from scratch. To this end, we have used very large networks that have shown success in the ImageNet Challenge. The drawback is that the number of damage classes for civil infrastructure assessment is far less than the number of classes in ImageNet (i.e., 1000 classes). This means that while the feasibility of TL has been acknowledged for health monitoring of civil infrastructures, this solution is not efficient (i.e., the networks are unnecessarily too deep for the problem of interest) and, consequently, TL is not suitable for detection at devices. To address this issue, we have used network pruning to enhance the resource efficiency for on-device analysis while still maintaining good detection accuracy. This approach allows one to deploy deep CNNs that are quite accurate, require low storage and computing resources, and can make decisions very quickly at devices.

In terms of architecture for analytics at the devices, our approach consists of two components, namely the TL component, and the pruning component. These components are independent from each other and it is thus possible to use different approaches for each of these. Concerning the first component, its interface is a standard interface for training a CNN in that our TL approach consists of taking an already trained CNN and re-train it using our domain-specific training dataset. Concerning the second component, in our implementation we use the method by Molchanov et al. [9]. This method takes as input a CNN and, starting from a full set of the CNN parameters, it iteratively removes the least important parameters. The method alternates between pruning and fine-tuning and stops after reaching the target trade-off between accuracy and the pruning goal, which is our case is memory size.

We have tested this approach for the detection of crack [1] and corrosion [10] surface defects. The methodology starts with a pre-trained network (e.g., VGG16 [11]), and recursively reduces the network size by using the Taylor-expansion based pruning technique [9]. Since the pre-trained network is originally designed for the ImageNet 1000 image categories, it is very large in size and may contain redundant convolution kernels that do not contribute to the new detection problems of interest. The pruning technique evaluates the importance of the convolution kernels and removes the kernels with the least contribution. After removing the kernels, the pruned network is fine-tuned again to enhance its performance for damage detection. Based on the detection performance, the user can determine whether or not to further prune the network following the same procedure.

In our experiments we started from VGG16, and used 29, 468 crack, 29, 780 non-crack, 33, 039 corrosion, and 34, 148 non-corrosion image patches to train and test the network. In this case, pruning stops if the detection accuracy after fine-tuning drops more than 3%. When up to 84% filters are pruned, the mean detection accuracy after fine-tuning for crack and corrosion is approximately 99% with a standard deviation below 0.5%. This demonstrates the robustness of the proposed approach as the variations in the performance are quite small when the pruned network still has the capacity to deal with the detection task. When 97% of the filters are removed (i.e., only 128 filters left), the mean accuracy of crack detection drops to 84.7% with a standard deviation of 19.86%, and the mean accuracy of corrosion detection drops to 96.0% with a standard deviation of 0.95%. This indicates that the pruning should be terminated due to the increasing variation and decreasing accuracy in detection performance.

We also compared the inference time (i.e., the total time required to classify one 720×540 image) of VGG16 and ResNet18 [12] when deployed at devices (i.e., NVIDIA Jetson TX2 GPU) for damage detection. By removing 84% and 79% of the convolution kernels from VGG16 and ResNet18, respectively, the inference time for crack detection decreases from 279.7 (s) to 31.6 (s) for VGG16 and 36.8 (s) to 8.9 (sec) for ResNet18. For the corrosion dataset, the inference time decreases from 275.7 to 30.6 (s) and from 34.1 to 9.0 (s) for VGG16 and ResNet18, respectively. In terms of memory reduction, the memory demands of VGG16 drop from 525 to 125 (MB), and the demands of ResNet18 drop from 44 to 2 (MB). By utilizing network pruning, VGG16 achieves a 89% reduction in inference time and 80% reduction in memory, while ResNet18 achieves a 76% reduction in inference time and 95% reduction in memory demands, without decreasing damage detection performance.

Our results indicate that network pruning is an important step towards incorporate deep learning architectures into devices. However, there are still several open questions that need further research. For instance, the selection of the appropriate pruning algorithm to reduce the network size. Also estimating the sensitivity of the pruning algorithm with respect to various network configurations is critical. The sensitivity should be considered in different aspects, e.g., inference performance and pruning efficiency.

2.2 Related work

TL techniques have already been widely investigated [13], whereas pruning techniques have been receiving wide attention more recently. Notable related work includes:

- *Transfer Learning:* Zhu et al. [14] addressed heterogeneous TL and used information from text data to improve model's performance in image classification. Aytar et al. [15] and Tommasi [16] addressed the deficit of training samples for some categories by adapting classifiers trained for other categories. Oquab et al. [17] showed that layers trained on ImageNet [18] can be reused to extract the mid-level features of images in the PASCAL VOC dataset. Shin et al. [19] addressed two specific computer-aided detection problems in medical images by fine-tuning CNNs pre-trained using a huge training set, such as CIFAR-10 [20], which has one million images from ten different classes. They further explored different popular CNN architectures and their performances on datasets of different sizes, concluding that the trade-off between learning more accurate models and using more training data should be carefully considered. Collobert et al. [21] explored TL for natural language processing. Hwangbo et al. [22] showed how to enhance RL by applying TL. More recently Singla et al. [23] developed a TL approach based on generative adversarial networks (GANs) for data different from images.
- *Model Pruning:* Network pruning has been investigated in the context of CNNs. The early work of LeCun et al. [24] proved that network pruning is a valid strategy to reduce the network complexity and over-fitting by using a diagonal Hessian-based approximation. In this kind of approximation, neurons were removed based on the result of calculations obtained from Hessian matrix. Recently, Han et al. [25] proposed an effective compression approach for CNNs. They have tested their approaches on both VGG16 and AlexNet [26] and on different hardware platforms. The experiments showed that their approach was able to substantially reduce the size of the networks without losing accuracy. He et al. [27] proposed a kernel pruning algorithm for CNNs. Their experimental results show that for VGG16 their approach was able to obtain a time speed up of 5*X* with only 03% error increase.

2.3 Research directions

In many applications, such as civil infrastructure monitoring and emergency management, the amount of training data is limited and thus TL is essential to still obtain good models even with limited data. Pruning is then critical to reduce the size of the models obtained from transfer learning. However, combining these two techniques requires analyzing the optimal ordering of TL and pruning steps, and assessing the impact that different strategies would have on the performance of different CNNs. Equally important is to assess and optimize the computing resources required for inference at different edge devices. In what follows we discuss some relevant research directions.

2.3.1 Optimal ordering for the execution of pruning and TL steps

It is important to determine whether it is better to first execute TL and then pruning—the strategy adopted in our preliminary work—or vice-versa. Both these strategies could be beneficial. By applying TL first, one can make sure that models adapt well to the target dataset, and then pruning can remove redundant neurons or layers. In this approach, the probability of removing the right neurons or layers from the model is higher. However, if the size of the target training dataset is small, it is better to first reduce the size of the model by applying pruning. More neurons or layers in a model means more parameters, and thus the time for training and fine-tuning will increase as well as the inference times. In addition, training a large model with a small amount of data may cause model over-fitting. Consequently, if the target dataset is small, it is better to first remove redundant neurons or layers, and then train the network with the small dataset.

2.3.2 Optimal pruning strategy

There are three pruning strategies that can be adopted:

- *Neuron removal* This is a common pruning strategy. There are various criteria for removing neurons:
    - *Threshold-based pruning* This strategy analyzes the weight of neurons and removes those having a weight less than a threshold. Results by Han et al. [28] show that this strategy reduces the number of neurons by a factor of 9× without incurring accuracy loss.
    - Taylor-based pruning. This strategy, used in our preliminary work, uses Taylor expansion [9] to determine the importance of neurons and thus allows one to remove the least important ones.
- *Layer removal* Previous results by He et al. [29] have shown that most of the neurons in the middle layers of ResNet50 have zero weight. Such neurons not only do not perform any feature extraction, they may also result in information loss. Thus, it is possible to skip those redundant layers by the introduction of shortcut connections [30] that skip one or more layers.
- *Kernel removal* Previous results by He et al. [27] have shown that kernel removal is another possible approach to prune CNNs, as images have RGB three different colors that correspond to three kernels in the CNN architecture. Thus, in some applications not every kernel is needed. For example, if one needs detect red roses, the red kernel would play a more important role than other kernels. Therefore one can keep the red kernel and remove some redundant kernels. Most of the times, a CNN has more than RGB three kernels, so there are several redundant kernels that could be removed.

Experimental comparisons to determine which pruning strategy works better depending on specific datasets and networks are critical. In addition, an interesting approach to explore would be to combine the three different pruning strategies: neuron, kernel, and layer removal. For instance, in ResNet50 after removing removed some layers, redundant/useless kernels and neurons could still be left in the network. Removing those kernels and neurons may further reduce the network size and inference time.

2.3.3 Multi-step transfer learning and pruning

There can be scenarios where one might need multi-step TL and pruning (MSTLP). MSTLP refers to obtaining a target network, specialized for certain classification decisions, by performing multiple TL and pruning steps. In such an approach one would start from an initial general network. Then, an intermediate trained network is derived from the initial one by applying TL and pruning. From the intermediate network one can derive another intermediate network and continue this procedure. For example, one can use TL and network pruning on a network initially trained using a generic dataset, such as ImageNet, in order to obtain an optimized network model for civil infrastructure health monitoring applications that can detect various defects such as cracks, corrosion, etc. One can then use this intermediate model to train specialized binary classification models that just detect specific categories such as corrosion/no-corrosion, crack/no-crack, etc. This multi-step approach has several advantages over a single-step TL approach that learns directly from a generic dataset: (a) TL and network pruning can be much faster if they start from an intermediate specialized network than if they start from a larger generic network trained for large number of categories not relevant to the application of interest. (b) The use of intermediate networks enhances flexibility in cases in which one needs to identify more than one class as well as use a specialized network when one just needs to detect one category of objects/defects. For instance, one might just be interested in only identifying cracks for civil infrastructure inspection purposes. (c) Detection accuracy is enhanced as MSTLP may help prevent overfitting. (d) The network size obtained from MSTLP would be smaller than the ones obtained from training specialized classifiers directly from the generic dataset. Investigating MSTLP is an interesting research direction.

3 Adaptive data acquisition

We now focus on the other crucial aspect of a smart health inspection system based on intelligent autonomous devices. Consider the case of an inspection device, such as a robot sent to inspect a civil infrastructure (e.g., a bridge). In this case, the device would carry some sensors (e.g., cameras) and perform on-board analyses using the approaches discussed in the previous section. The device will first need to identify a target of interest (e.g., crucial structural components of a bridge), navigate itself to a position closer to the target, and then collect data to determine the presence of damage as well as whether to proceed with a more detailed inspection. During the inspection process, the device is constantly facing decision making problems whenever an input data sample is acquired. For instance, what should be the next movement of the device if no damage of interest is present in the input image? Is the device close enough to the target? Is there any damage detected on the target? Is there any other potential damage nearby the target that requires more data collection? Is the collected data of good quality enough for the device to make the next decision? In such a context where many decisions need to be taken, RL appears a suitable approach.

RL, inspired by human learning, is a technique to determine the optimal decision by interacting with the environment. Compared to the conventional Q-learning algorithm, which becomes extremely inefficient for large scale state-space problems [31], the incorporation of deep neural networks (DNNs) into RL makes the learning process able to deal with high dimensional problems. In what follows we refer to DNN-based RL as DRL.

3.1 Related work

Recent advances in robotic navigation/control fields have demonstrated the great potential of DRL for smart inspection systems and other similar applications. Tai et al. [32] demonstrated the capability of using DRL to train a robot to reach a pre-defined target location without collision in a map-less environment. Cheng and Zhang [33] used DRL to navigate a boat to a target, and avoid obstacles on the way. Mirowski et al. [34] used DRL for training agents to navigate in large and visually rich environments with various starting points and destinations. Zhu et al. [35] developed a robotic navigation system for indoor scenarios using DRL. The robot is trained in a simulated environment and is able to navigate to user-defined indoor objects (e.g., a sofa or a desk) in the testing stage. Hwangbo et al. [22] used DRL to control a quadrotor to stabilize itself when subjected to extreme external forces. Moreover, DRL has been applied in various video games where a trained artificial intelligence (AI) system is able to outperform human players. Hasselt et al. [36] proposed the double Q-learning algorithm to address the issue of overestimation problem encountered by deep Q-network (DQN). The performance of double Q-learning was tested on multiple Atari games and it was shown to be effective. Wang et al. [37] proposed a dueling network architecture to decouple the estimation of value and advantage in DQN. The dueling network outperformed the double Q-learning algorithm [36] in the challenging Atari game. Schaul et al. [38] showed that by integrating experience replay into the double Q-learning algorithm proposed in [36], the performance of the AI system in playing Atari games can be further enhanced. In [39, 40], DRL was used to play first-person shooter (FPS) games (e.g., Doom video game). The agent was trained to explore the map, collect items, search and fight against the enemies. Due to DRL's success in these challenging decision making problems, we consider DRL as the best fit for smart inspection systems.

3.2 An example inspection scenario and a DRL-based framework

Consider an inspection device that starts with an initial state S_0 and observes data D_0 (e.g., an image). Based on D_0, the device selects an action A_0 from the action set A and moves to a new state S_1. By interacting with the environment (or an environment simulator), an immediate reward R_1 is assigned to the device, and the device will move to new states by repeating the steps of data acquisition, selecting actions, and receiving rewards. The objective of RL is to find the

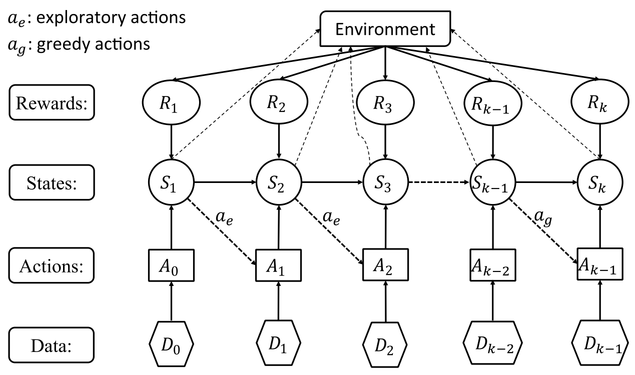
a_e: exploratory actions
a_g: greedy actions
Environment
Rewards:
States:
Actions:
Data:
R_1 R_2 R_3 R_{k-1} R_k
S_1 S_2 S_3 S_{k-1} S_k
a_e a_e a_g
A_0 A_1 A_2 A_{k-2} A_{k-1}
D_0 D_1 D_2 D_{k-2} D_{k-1}

Fig. 2 The concept of RL

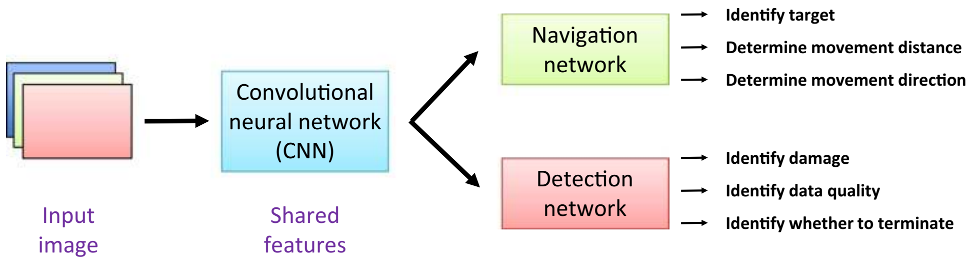
Convolutional neural network (CNN)
Navigation network
Detection network
Identify target
Determine movement distance
Determine movement direction
Identify damage
Identify data quality
Identify whether to terminate
Input image
Shared features

Fig. 3 The example DRL framework

optimal policy, i.e., the set of best actions based on observing the data, that maximizes the cumulative rewards. During training, the underlying driving mechanism is the Bellman equation [31] given by the equation below, that updates the *Q*-value of picking an action *a* at state S_i:

$$Q(S_i, a) \leftarrow (1-\alpha^k)Q(S_i, a) + \alpha^k[R(S_i, a, S_{i+1}) - \rho^k t(S_i, a, S_{i+1}) + \gamma \max_{b \in A(S_{i+1})} Q(S_{i+1}, b)]$$

In the previous equation, $R(S_i, a, S_{i+1})$ is the immediate reward after choosing action *a* at state S_i, $t(S_i, a, S_{i+1})$ is the transition time from state S_i to S_{i+1}, $\max_{b \in A(S_{i+1})} Q(S_{i+1}, b)$ is the maximum *Q*-value among all the actions $b \in A(S_{i+1})$ in the next state S_{i+1}, *k* is the iteration number, α^k is the learning rate at iteration *k*, ρ^k is the average reward at iteration *k*, and γ is the discount factor for the expected future rewards. At the early stage of training, the device has a higher tendency to choose the exploratory actions in order to discover the good actions in the state-action space. As the training proceeds, the device will gradually choose the greedy actions, i.e., the actions that have the highest *Q*-value. Note that the proposed DRL approach approximates $Q(S_i, a)$ with DNNs in order to account for the infinite number of state-action pairs in our problem. Figure 2 illustrates the fundamental concept of RL in the context of smart inspection systems.

In our inspection scenario, the device would first collect data at a farther distance to capture the overview of the building, and then would move to a closer position to inspect the damage on a first floor column thoroughly. Such scenario is reflected by our example DRL-based framework shown in Fig. 3. A CNN is employed to extract features from the input image, and the features are sent to both the navigation and the damage detection network. The navigation network deals with the identification of the target of interest (e.g., first floor columns shown in Fig. 4[1]) and determines the movement of the device. The detection network identifies the presence of damage on the target and determine whether the data is of good quality (e.g., lack of blurriness and/or completeness), and whether to proceed with more data collection. Consider the image shown in Fig. 4. The actions for the device to take can be defined as:{a_1 :a column exists and move forward distance d_1; a_2 :a column exists and move forward distance $d_2 < d_1$; a_3 :a column does not exist and randomly picks the next movement direction and movement distance.} Since the device does see a column and should move much more to get closer to the column, the corresponding rewards can be assigned as:$\{R_1 = 1.0, R_2 = 0.5, R_3 = 0\}$ in order for the device to learn to move more when seeing images similar to the one in Fig. 4. Note that the navigation and detection network will work jointly during the inspection, as the decision made by one network will affect the decision on the other. For instance, the navigation network will keep making decisions if the detection network cannot identify

[1] The images in Figs. 4 and 5 are snapshots of a video recorded by using a UAV by one of the authors of this paper. The UAV was operated to inspect a school building damaged during a strong earthquake which happened in southern Taiwan in February 2016.

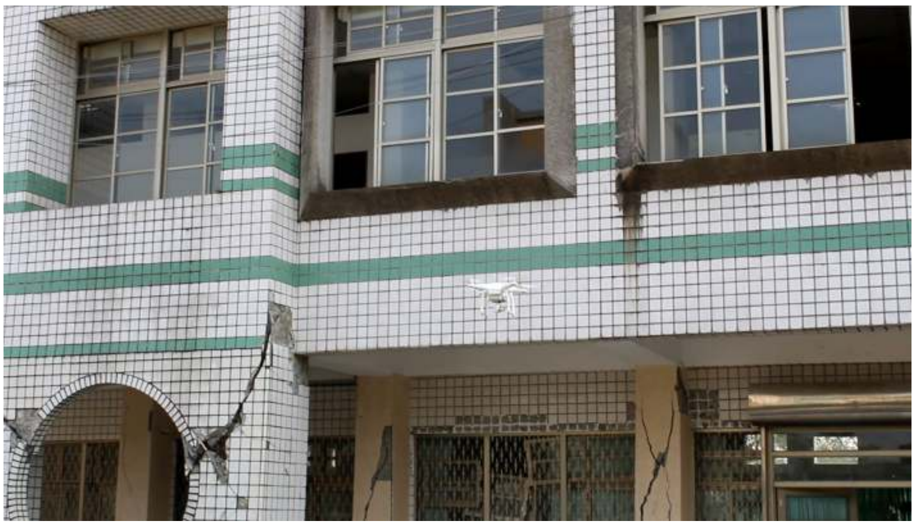
Fig. 4 The device is inspecting a building after earthquake

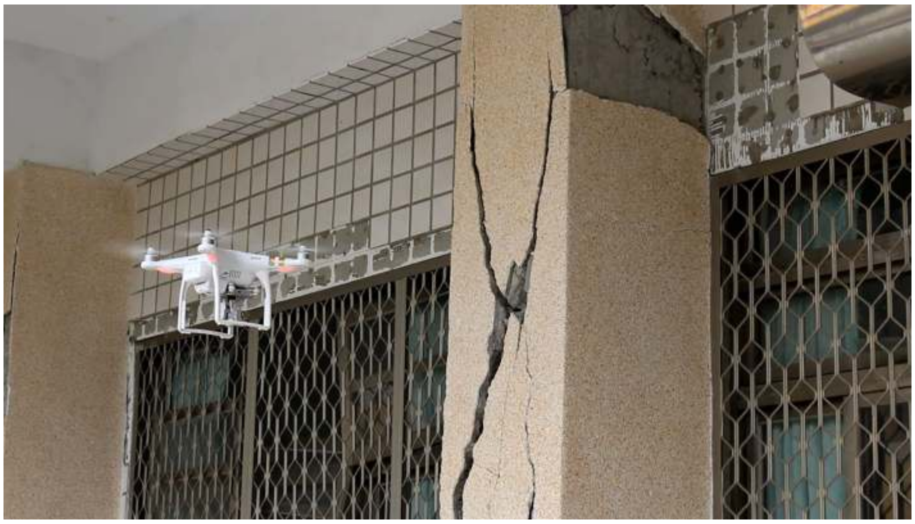
Fig. 5 The device is inspecting the damage on a first floor column

the presence of damage with Fig. 4. Once the device moves to a closer position to the column, e.g., Fig. 5, the actions for the detection network can be defined as: $\{a_1$: a crack is detected and should slightly move to collect more data; a_2 : a crack is detected and requires no further data collection; a_3 : no damage is detected$\}$. Since the device only captures a portion of the damaged column and should collect more data, the corresponding rewards can be assigned as: $\{R_1 = 1.0$, $R_2 = 0.5, R_3 = 0\}$ in order for the device to learn to collect more data when seeing images similar to the one in Fig. 5.

Our DRL-based inspection framework has then to support two important tasks: (1) routine inspection—the device would be trained to perform regular inspections, in which the device scans through the structure and identifies all possibilities of damage presence; and (2) urgent inspection—the device would be trained for rapid assessment in the context of urgent inspection (i.e., inspections after a natural hazard) where the device will evaluate the severity of damaged regions and it will perform more data collection around the damage of top priority (i.e, not a thorough inspection of whole structures).

3.3 Research directions

3.3.1 DRL-based autonomous smart inspection system under two scenarios, i.e., routine inspection and urgent inspection

Although DRL has been demonstrated to be effective in robotic navigation, there are challenges needed to be addressed in the context of health inspection. For instance, in the existing approaches the target/destination is predefined and will be identical in the testing stage, e.g., the items and the enemies in the Doom game [39]. In our context, the target of interest and the damage may have patterns similar but not identical as the training data in the testing stage. The sensitivity of DRL towards the changes in the objects of interest needs a thorough investigation.

3.3.2 Evaluation of the performance of different network configurations, i.e., DQN, double Q-learning and dueling network, on navigation and damage detection

There are various network configurations proposed in previous work to enhance the performance of DRL in playing Atari games. However, there is no a particular configuration that outperforms the others in all categories of games [36]. Therefore, research is needed to determine which configurations are best for inspection systems.

3.3.3 Evaluation of the benefit of an environment simulator

Unlike past work in which an environment simulator, such as a gaming engine or a predefined map, is available, our adaptive DRL data collection framework infers the information directly from the acquired training data. This is particularly useful for urgent inspection after natural hazards as the environment often changes drastically. An interesting research direction is to analyze whether an approximate simulator can bring additional benefits to the inspection system.

3.3.4 Transfer learning for reinforcement learning agents.

A critical challenge related to real-time requirements is that RL has high convergence times. To address this issue, an approach is to use TL techniques based on GANs [23].

The application of such a methodology to quickly bootstrap the DQN in the control RL system has two variations:

- *Reward Knowledge Transfer* Under this variation, a few explorations are performed by the RL agent. Then a GAN is used to generate augmented or synthetic data (enhanced exploration data) by minimizing domain loss between the TL source domain and target domain. The resulting dataset is then used to train the DNN of the RL agent at the target domain. A key environment specific parameter in this setting is the bias introduced while training the DNN of RL agent. For optimal convergence, the new exploration samples should be favoured over stale samples from the source domain. This approach requires two learning/training steps, one to train the GAN and another one to train the DNN of the RL agent at the target domain.
- *Quality Value Knowledge Transfer* This variation is similar to previous one except that here we transfer knowledge about the RL model directly by exchanging Q values instead of exploration samples. Here the target dataset is generated from the baseline RL agent model (Q values) trained by some new explorations. Here the GAN has two tasks: minimizing domain loss and optimizing the Q function. The key goal here is balancing domain loss and quality loss while training the GAN. Since, there is only one learning step here, we need to introduce a bias (target domain over source domain) in the GAN itself; this can be done when selecting batches to train the GAN.

It will be interesting to compare the two approaches and investigate additional specific aspects related to TL for RL agents used for inspection systems.

4 Concluding remarks

In this paper we have discussed the use of devices in a critical application domain, that is, the monitoring of civil infrastructures for defect identification and assessment. In the paper we have focused on the use of ML techniques to allow devices to carry out data analyses on-board and to enhance the autonomy of devices. Device autonomy is critical when dealing with emergency situation in which network communications can be become fragmented. The paper has discussed several research directions. However the area of ML for IoT systems is an important emerging application area with many other interesting research directions. One relevant research direction is the security of autonomous devices as well as the privacy of data they acquire and store. This is a challenging problem that also depends on the specific application scenarios and threats. Proposed defense techniques range from remote attestation to make sure that software running on the devices is not compromised to techniques for dynamic white-box cryptography techniques to temporarily protect encryption keys from being extracted from the memory of these devices, in the event of their physical capture [41]. An interesting idea would be to use other special-purpose devices for detecting such physical capture through video analysis. Finally another direction interesting research direction is related to the high correlation that may exist between data collected by devices in close proximity. Such correlation may, for example, be used by devices to cross-validate their collected data. Also, devices may need to coordinate in order to avoid sending redundant data, especially when the transmission bandwidth is limited. We notice however that the deployment of 5G cellular networks ad the envisioned 6G cellular network technology [42] may alleviate such problem and thus further enhance health monitoring of civil infrastructures.

Authors' contributions All authors equally contributed to the paper.

Competing interests The authors declare no competing interests.

References

1. Che F-C, Jahanshahi MR. Nb-cnn: Deep learning-based crack detection using convolutional neural network and naive Bayes data fusion. IEEE Trans Ind Electr. 2018;65(5):43924400.
2. Shi W, Cao J, Zhang Q, Li Y, Xu L. Edge computing: vision and challenges. IEEE Internet Things J. 2016;3(5):637646.
3. Allahbakhsh M, Benatallah B, Ignjatovic A, Motahari Nezhad HR, Bertino E, Dustdar S. Quality control in crowdsourcing systems: issues and directions. IEEE Internet Comput. 2013;17(2):76–81.
4. Flir launches next-generation black hornet 3 nano-uav; press release. https://www.flir.com/news-center/press-releases/flir-launches-next-generation-black-hornet-3-nano-uav/. 2018.
5. Bertino E, Calo S, Touma M, Verma D, Williams C, Rivera B. A cognitive policy framework for next-generation distributed federated systems: Concepts and research directions. In: Proceedings of the 37th IEEE International Conference on Distributed Computing Systems, ICDCS2017, Atlanta, GA, USA, June 5–8; 2017.
6. https://en.wikipedia.org/wiki/PonteMorandi.
7. Bertino E, Jahanshahi MR. Adaptive and cost-effective collection of high-quality data for critical infrastructure and emergency management in smart cities - framework and challenges. ACM J Data Inform Quality (JDIQ). 2018;10(1):1:–11:6.
8. Rih-Teng W, Singla A, Jahanshahi MR, Bertino E. Pruning deep convolutional neural networks for efficient edge computing in condition assessment of infrastructures. Comput Aided Civil Infrastructure Eng. 2019;34(9):774–89.
9. Molchanov P, Tyree S, Karras T, Aila T, Jan K. Pruning convolutional neural networks for resource efficient inference. In: Proceedings of the 5th international conference on learning representations, ICLR 2017, Toulon, France, April 24–26; 2017.
10. Atha DJ, Jahanshahi MR. Evaluation of deep learning approaches basedon convolutional neural networks for corrosion detection. Struct Health Monit. 2018;17(5):11101128.
11. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: Proceedings of the 3rd International conference on learning representations, ICLR 2015, San Diego, CA, USA, May 7–9; 2015.
12. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition.arXiv preprint, (arXiv:1512.03385). 2015.
13. Jialin Pan S, Yang Q. A survey on transfer learning. IEEE Trans Knowl Data Eng. 2010;22(10):1345–59.
14. Zhu Y, Chen Y, Lu Z, Jialin PS, Xue G-R, Yu Y, Yang Q. Heterogeneous transfer learning for image classification. In: Proceedings of the twenty-fifth AAAI conference on artificial intelligence, AAAI 2011, San Francisco, California, USA, August 7–11. 2011.
15. Aytar Y, Zisserman A. Tabula rasa: Model transfer for object category detection. In: Proceedings of the IEEE international conference on computer vision, ICCV 2011, Barcelona, Spain, November 6–13, 2011.
16. Tommasi T, Orabona F, Caputo B. Safety in numbers: learning categories from few examples with multi model knowledge transfer. Proceedings of the Twenty-Third IEEE conference on computer vision and pattern recognition, CVPR 2010, San Francisco, CA, USA, 13–18 June 2010.
17. Oquab M, Bottou L, Laptev I, Sivic J. Learning and transferring mid-level im-age representations using convolutional neural networks. In: Proceedings of the 2014 IEEE conference on computer vision and pattern recognition, CVPR 2014, Columbus, OH, USA, June 23–28, 2014.

18. Jia D, Wei D, Richard S, Li-Jia L, Kai L, Fei-Fei L. ImageNet: A Large-Scale Hierarchical Image Database. In: Proceedings of the IEEE computer society conference on computer Vision and pattern recognition (CVPR 2009), 20–25 June 2009. Miami: Florida, USA; 2009.
19. Hoo-Chang S, Roth HR, Gao M, Lu L, Xu Z, Nogues I, Yao J, Mollura D, Summers RM. Deep convolutional neural networks for computer-aided detection.CNN architectures, dataset characteristics and transfer learning. IEEE Trans Med Imaging. 2016;35(5):1285.
20. Krizhevsky A, Nair V, Hinton G. Cifar-10 (Canadian Institute for Advanced Research).
21. Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P. Natural language processing (almost) from scratch. J Mach Learn Res. 2011;12:2493–537.
22. Hwangbo J, Sa I, Siegwart R, Hutter M. Control of a quadrotor with reinforcement learning. IEEE Robotics Autom Lett. 2017;2(4):2096–103.
23. Singla A, Bertino E, Verma D. Preparing network intrusion detection deep learning models with minimal data using adversarial domain adaptation. In: Proceedings of the 15th ACM Asia conference on computer and communications security, AsiaCCS2020, Taipei, Taiwan, October 5–9; 2020.
24. LeCun Y, Denker JS, Solla SA. Optimal brain damage. In: Proceedings of the annual neural information processing systems, NIPS 1989, Denver, Colorado, USA, November 27–30. 1989.
25. Han S, Mao H, Dally WJ. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In: Proceedings of the 4th international conference on learning representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016.
26. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. In: Proceedings of the 26th annual conference on neural information processing systems 2012 (December), pp. 3–6. Lake Tahoe: Nevada; 2012.
27. He Y, Zhang X, Sun J. Channel pruning for accelerating very deep neural networks. In: Proceedings of the IEEE international conference on computer vision, ICCV 2017, Venice, Italy, October 22–29, 2017.
28. Han S, Pool J, Tran J, Dally W. Learning both weights and connections for efficient neural network. In: Proceedings of the annual conference on neural information processing systems, NIPS 2015(December), Montreal. Canada: Quebec; 2015. p. 7–12.
29. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27–30, 2016.
30. Szegedy C, Liu W, Jia Y, Sermanet P, Reed SE, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. In; Proccedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7–12, 2015.
31. Gosavi A. Simulation-based optimization: parametric optimization techniques and reinforcement learning. New York: Springer; 2015.
32. Tai L, Paolo G, Liu M. Virtual-to-real deep reinforcement learning: continuous control of mobile robots for mapless navigation. In: Proceedings of the 2018 IEEE international conference on robotics and automation, ICRA 2018, Brisbane, Australia, May 21–25, 2018.
33. Cheng Y, Zhang W. Concise deep reinforcement learning obstacle avoidance for under actuated unmanned marine vessels. Neurocomputing. 2018;272(1):63–73.
34. Mirowski P, Pascanu R, Viola F, Soyer H, Ballard AJ, Banino A, Denil M, Goroshin R, Sifre L, Kavukcuoglu K, Kumaran D, Hadsell R. Learning to navigate in complex environments. In: Proceedings of the 5th international conference on learning representations, ICLR 2017, Toulon, France, April 24–26, 2017.
35. Zhu Y, Mottaghi R, Kolve E, Lim JJ, Gupta A, Fei-Fei L, Farhadi A. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: Proceedings of the IEEE international conference on robotics and automation, ICRA 2017, Singapore, Singapore, May 29–June 3, 2017.
36. van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double q-learning. In: Proceedings of the thirtieth AAAI conference on artificial intelligence, February 12–17. Phoenix. USA: Arizona; 2016.
37. Wang Z, Schaul T, Hessel M, van Hasselt H, Lanctot M, de Freitas N. Dueling network architectures for deep reinforcement learning. In: Proceedings of the 33nd international conference on machine learning, ICML 2016, New York City, NY, USA, June 19–24, 2016.
38. Schaul T, Quan J, Antonoglou I, Silver D. Prioritized experience replay. In: Proceedings of the 4th international conference on learning representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016.
39. Lample G, Singh CD. Playing FPS games with deep reinforcement learning. In: Proceedings of the thirty-first AAAI conference on artificial intelligence, February 4–9. San Francisco: California; 2017.
40. Adil K, Jiang F, Liu S, Grigorev A, Gupta BB, Rho S. Training an agent for FPS doom game using visual reinforcement learning and vizdoom. In: International journal of advanced computer science and applications, IJACSA, Vol. 8, No. 12. 2017.
41. Won Jongho, Seo Seung-Hyun, Bertino Elisa. A secure shuffling mechanism for white-box attack-resistant unmanned vehicles. IEEE Trans Mob Comput. 2020;19(5):1023–39.
42. Bertino E, Bliss D, Lopresti D, Peterson L, Schulzrinne H. Computing research challenges in next generation wireless networking. CoRR abs/2101.01279. 2021.