scientific reports

Check for updates

OPEN

LLM-Twin: mini-giant model-driven beyond 5G digital twin networking framework with semantic secure communication and computation

Yang Hong[1], Jun Wu[1]✉ & Rosario Morello[2]

Beyond 5G networks provide solutions for next-generation communications, especially digital twins networks (DTNs) have gained increasing popularity for bridging physical and digital space. However, current DTNs pose some challenges, especially when applied to scenarios that require efficient and multimodal data processing. Firstly, current DTNs are limited in communication and computational efficiency, since they require to transmit large amounts of raw data collected from physical sensors, as well as to ensure model synchronization through high-frequency computation. Second, current models of DTNs are domain-specific (e.g. E-health), making it difficult to handle DT scenarios with multimodal data processing requirements. Finally, current security schemes for DTNs introduce additional overheads that impair the efficiency. Against the above challenges, we propose a large language model (LLM) empowered DTNs framework, LLM-Twin. First, based on LLM, we propose digital twin semantic networks (DTSNs), which enable more efficient communication and computation. Second, we design a mini-giant model collaboration scheme, which enables efficient deployment of LLM in DTNs and is adapted to handle multimodal data. Then, we designed a native security policy for LLM-twin without compromising efficiency. Numerical experiments and case studies demonstrate the feasibility of LLM-Twin. To our knowledge, this is the first to propose an LLM-based semantic-level DTNs.

The development of next-generation communication technologies (beyond 5G), as well as internet of everything (IoE) technologies, has paved the way for visions of smart cities[1], smart transportation[2], smart homes, etc., but it has also resulted in a growing need for rapid processing of massive and diverse network data[3]. As one of the most promising next-generation data-driven paradigms, digital twin networks (DTNs)[4] can efficiently process massive amounts of data and have demonstrated great application value in various areas such as real-time transportation safety assessment[5], intelligent city scheduling[6], industry remote control[7], etc. DTNs can create real-time digital replicas of the physical world for fine-grained modeling, analytics, and prediction, which reveals that DTNs are significantly reshaping the future network paradigm in terms of efficiency and intelligence[8].

The main characteristic of digital twins (DTs) is bi-directional communication[9,10], which is denoted as intra-twin communication and inter-twin communication in DTNs[11] (as shown in Fig. 1). Intra-twin communication refers to the interaction between a physical entity and its DT. The DT employs advanced cloud computing and machine learning models to provide insights based on the historical and real-time data of physical entities, which effectively compensates for the local limitations of physical entities. Unfortunately, intra-twin communication is very expensive on physical networks because of the large amount of communication and computational resources consumed to support real-time applications[12]. In addition, constructing DTs by uploading detailed physical entity data to servers via intra-twin communication also leads to huge data security concerns[4]. Inter-twin communication denotes information sharing and communication among each DT in DTNs, which enables the physical entity to gain a larger perceptual domain and provides it with global decision-making[13]. Furthermore, inter-twin communication obtains information in DTNs through data access, virtual links, etc., which breaks the limitation of physical links and mainly relies on the computational power of the cloud server to model the data transmission behavior. However, due to the massive number of IoT devices and extremely diverse data types, it is also hard to

[1]Graduate School of Information, Production and System, Waseda University, Fukuoka 8080135, Japan. [2]Department of Information Engineering, University "Mediterranea" of Reggio Calabria, Via Graziella, 89122 Reggio Calabria, Italy. ✉email: jun.wu@ieee.org

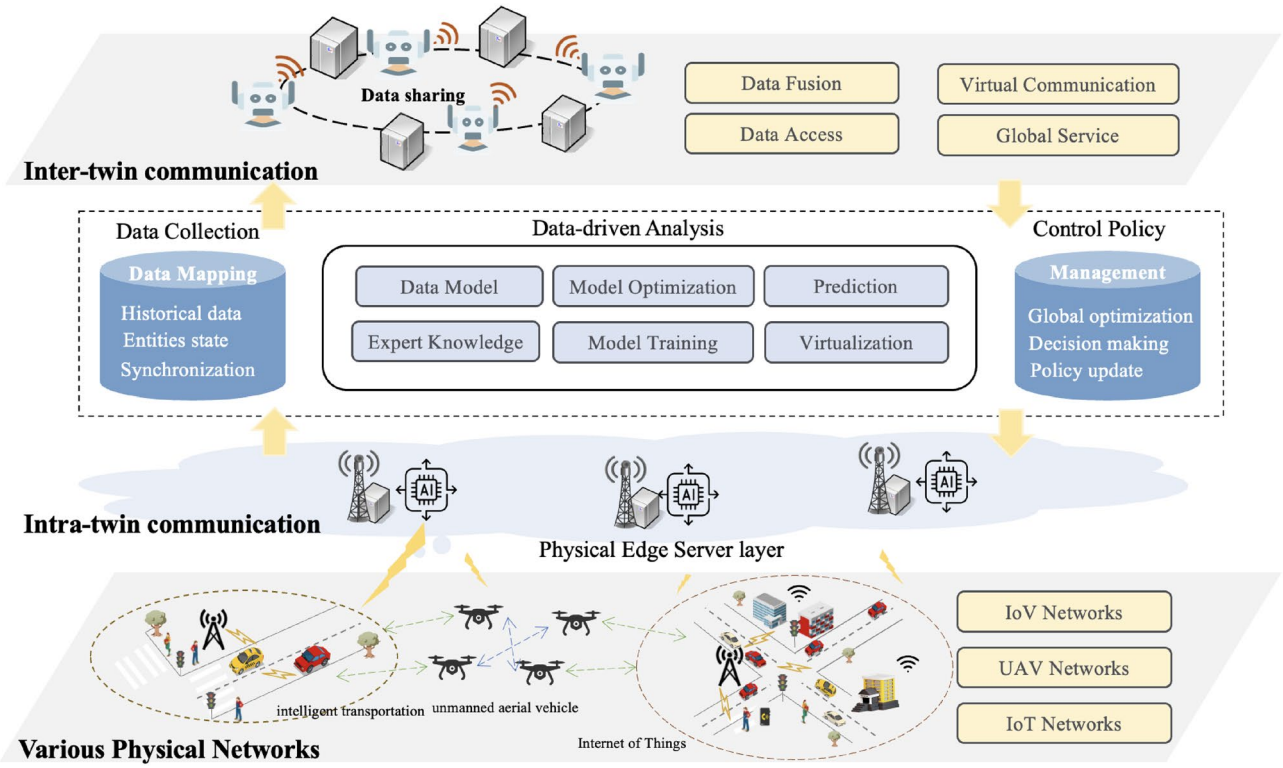

Figure 1. Beyond 5G digital twin networks.

model efficient data sharing for inter-twin communication. Therefore, realizing more efficient communication, intelligent computation, and more secure data processing capabilities in DTNs are still open issues at present[14].

Lu et al [15] proposed digital twin edge networks (DTENs), which bridge the gap between physical edge networks and DT cloud servers by introducing distributed federated learning (FL) into DTNs. Specifically, distributed edge physical entities participate in the model computation and aggregate the parameters in the DT server, liberating the computational burden of the DT server. However, since FL requires multiple rounds of weight updating and aggregation introduces additional communication overhead[16]. Therefore, asynchronous FL[17], low-latency FL[18], etc. investigate more efficient communication and computation of DTNs. In addition, reinforcement learning[19], and graph neural networks[20] have been used to further reduce resource consumption and improve the reliability of DTNs. Furthermore, to address data privacy and security issues in DTNs,[21,22] joint FL and blockchain techniques to protect local model updates and global model updates against data tampering and privacy leakage.

In general, the existing works have explored efficient and intelligent communication computing schemes for DTNs and also considered data security issues, but limited by the traditional networking framework of DTNs, the following three challenges still exist:

1. Current works have not resolved the inherent efficiency limitations that exist in communication and computation in traditional DTNs networking framework. This mainly includes bit-level real-time communication and high-frequency model computation. In particular, bit-level communication refers to the fact that sensors synchronize a large amount of raw data from the physical world to the DT server in real time, which imposes an extremely high communication load. In terms of high-frequency model computation, the DT side requires real-time model updates for synchronization with physical entities and constantly receives data shared by other DTs and makes decisions. This implies a significant computational load. Especially, the large number of raw parameter iterations and model updates in mainstream FL-based DTNs frameworks significantly limits the efficiency of DTNs.
2. Current machine learning models in DTNs network frameworks are domain-specific and do not consider the adaptability and scalability of DTNs. Specifically, for instance, the traditional FL-based DTNs model is difficult to adapt to and handle the non-independent synchronized data brought by massive heterogeneous devices, and cannot also process multimodal data. In addition, the scalability issues such as knowledge expansion and knowledge update for dynamic DTNs scenarios are also challenging.
3. The security of the traditional DTN networking framework is vulnerable. First, it improves security by incorporating blockchain, federated learning, and other schemes, but these schemes introduce additional communication and computation overhead, making it difficult to balance efficiency. Specifically, for instance, the FL-based DTNs model achieves local privacy protection through distributed collaborative learning, but also introduces additional overheads such as gradient iteration, model distribution, model aggregation, and

so on. In addition, FL-based DTNs still face security threats such as single point of failure, data poisoning, and privacy inference attacks.

Therefore, the above important challenges motivate us to investigate new solutions for DTNs networking framework. Semantic communication[23], as a new generation of network communication technology, is an advanced solution that promises to address the efficiency limitations of bit-level communication. In semantic communication, the sender and receiver share prior knowledge or context about the semantics before communication. During communication, the sender only needs to transmit a small amount of information so that the receiver can recover the entire original content. Therefore, semantic communication will significantly improve communication efficiency compared to bit-level communication where a large amount of original data is required to transmit through the channel.

To realize effective compression and reduction, semantic communication requires constructing and sharing a priori knowledge models in advance. Therefore, the performance of knowledge models, such as the accuracy of information reduction, knowledge richness, scalability, and adaptability, will directly affect the performance of semantic communication. Currently, the large language model (LLM), as the state-of-the-art artificial intelligence and extremely rich knowledge model, will be expected to provide advanced insights for the new generation of network semantic communication[24]. In other words, the essence of LLM is the high degree of compression of knowledge that then enables powerful generative tasks, which will significantly reduce the amount of raw data that needs to be transmitted thereby improving communication efficiency. In addition, LLM demonstrates advanced multimodal generation capabilities in text-to-code[25], text-to-image[26], text-to-speech[27], and text-to-video[28]. Advanced and efficient fine-tuning techniques also further enhance the scalability and adaptability of LLM. Specifically, in semantic communication, the sender can convert various types of information M, which consumes a lot of communication resources, into highly compressed C, and utilize LLM at the receiver to reconstruct M. This can greatly reduce the overhead from bit-level communication in traditional networks.

Benefiting from the rich knowledge base and powerful generation capabilities of LLM, we propose to introduce LLM into DTNs networking framework to address the existing challenges. However, in contrast, the training, inference, and fine-tuning of LLMs imply a huge computation overhead, while offloading LLMs to edge devices also incurs a huge communication overhead. In particular, computation and communication resources are limited in IoT systems[14], thus this becomes a significant bottleneck for introducing generative LLMs in DTNs. In addition, the security of LLM-based DTNs architecture needs further research.

In this paper, we propose an LLM-enabled DTNs networking framework, LLM-Twin. Specifically, we reshape the architecture of intra-twin and inter-twin communication based on LLM to realize semantic-level DT communication and computation, which is the first time to propose digital twin semantic networks (DTSNs). Second, in LLM-twin we design a mini-giant model collaboration scheme to solve the resource-constrained problem of deploying LLMs in DTNs. Finally, we propose strategies to enhance data security in LLM-twin, which can realize the hiding of sensitive information in DTNs. The main contributions of LLM-twin proposed in this paper are summarized as follows:

1. We first propose digital twin semantic networks (DTSNs), which are semantic-level frameworks for efficient communication and computation in DTNs. Driven by LLM we reshape the traditional architectures of intra-twin communication and inter-twin communication, including static knowledge base synchronization and dynamic semantic-level information sharing, which realizes the unification of efficient communication and computation.
2. We propose the mini-giant LLMs collaboration scheme to solve the resource-limited problem of LLMs deployment in DTNs, which involves edge data fine-tuning and instruction prompts. Moreover, in the proposed collaboration scheme, the large model provides rich knowledge information and the small model is responsible for personalized expertise updating. Therefore, it further improves the capabilities of DTNs in terms of multimodal data processing, scalability, and adaptability.
3. To further guarantee the security of the proposed LLM-Twin, we design an LLM data security strategy. It guarantees the privacy of sensitive information by reinforcing the reversal curse of LLM, which ensures that the sensitive information in the original input cannot be obtained in reverse from the service output of LLM-Twin. Finally, we give the security analysis and proof for LLM-Twin, while theoretical analysis and experiments show that our proposed networking framework significantly improves the communication computation efficiency of traditional DTNs.

The rest of the article is organized as follows: First, we discuss the related works in "Related works" section; second, we introduce the framework of LLM-Twin in "Framework of LLM-Twin with efficiency communication and data security" section; in "Efficiency analysis and security models of LLM-Twin" section we propose the efficiency and safety model of LLM-Twin; in "Evaluation and case study" section we evaluate the proposal; and finally we conclude the paper.

Related works

DTNs are an extension of DTs, defined as a many-to-many network paradigm consisting of a large number of one-to-one DTs[8]. Specifically, with advanced communication, modeling, and computation techniques, DTNs enable synchronous evolution and dynamic information interaction between multiple physical entities and their DTs (intra-twin communication). Meanwhile, DTNs achieve collaboration and information sharing among DTs (inter-twin communication) by building communication models for multiple DTs. In this context, constructing efficient and secure communication models for DTNs (intra-twin and inter-twin) is a significant challenge.

FL is widely used for modeling communication in DTNs because of its advanced distributed learning and information-sharing capabilities. In FL, the process of updating the real-time state and locally trained weights from physical entities to the server is modeled as intra-twin communication; the process of aggregating weight updates from individual entities at the FL server is modeled as inter-twin communication. Lu et al. used FL to construct DTNs models based on data characteristics of edge IoT devices and proposed an asynchronous model update scheme to mitigate the communication overhead[17]. Additionally, some works[18,21,22] improved to enhance the reliability and security of the DTNs system by combining FL and blockchain. Moreover, Sun et al. proposed a lightweight DT air-ground network architecture[29] and designed a distributed incentive mechanism to achieve efficient FL modeling.

Besides using deep learning models to model information sharing in DTNs, some research has focused on designing more efficient communication mechanisms, such as semantic communication[30,31]. Campolo et al.[30] proposed a DT interaction framework for connected autonomous vehicles (CAVs). They enable efficient interaction between vehicle entities and DTs as well as DTs and applications by designing specific interfaces and semantic models. Thomas et al.[31] were concerned with the reliability of semantic communication and they proposed a causal semantic communication (CSC) system to achieve high reliability and low latency DT communication.

In addition, some work focuses on security studies of DTNs. This part can be divided into solving security problems native to DTNs[32–35] and using DTNs to improve the security of other systems[36,37]. Xu et al.[32] proposed a mutual authentication scheme between a physical vehicle and its DT intending to secure intra-twin communication in DTNs. Further, Li et al.[33] designed a switching authentication scheme based on proxy ring signatures to accommodate dynamic DTNs scenarios. Alternatively, Dai et al.[34] proposed a blockchain-based information-sharing model to achieve traceability and secure data sharing in DTNs. Feng et al.[35] improved attribute-based encryption (ABE) to meet the security requirements of DTNs networking. On the other hand, there are efforts to improve security using DT. Yigit et al.[36]proposed a DT-based intelligent DDoS detection mechanism to achieve protection of Internet Service Provider (ISP) core networks. Further, they proposed a DT-based intelligent attack detection system that can protect 6G IoT edge networks[37]. Finally, to clearly demonstrate the differences between this study and existing works to show unique contributions to the field, we summarise the differences as follows (Table 1):

1. In terms of communication efficiency, existing work has focused on modeling the communication of DTNs using optimized FL. However, multiple rounds of weight updating and aggregation in FL make the communication efficiency a bottleneck. The LLM-twin framework proposed in this paper designs the Mini-Giant model cooperative architecture, which is committed to solving the bottlenecks of resources and efficiency in the traditional FL-based DTNs networking scheme. Furthermore, current work on semantic communication focuses only on communication between physical entities and DTs. The proposed LLM-twin architecture includes semantic-based information sharing between DTs and DTs. Furthermore, to the best of our knowledge, this work will be the first to integrate LLM and DTNs to address the above challenges.
2. In terms of security, rather than focusing on designing additional security schemes, this paper is focused on designing the native security policies and analyzing the security performance of the proposed framework. Traditional FL-based DTNs suffer from threats such as single points of failure, privacy leakage, and data heterogeneity. We will show the security advantages of the proposed framework in these aspects in the security analysis section. These native security advantages of the proposed DTNs framework will minimize the efficiency impact of additional security schemes in the future.
3. In terms of scalability, existing FL-based DTNs have paid little attention to the scalability of the architecture and the ability to handle multimodal heterogeneous data. This paper discusses the advantages of LLM-twin in handling personalized knowledge and heterogeneous data and evaluates the performance under scaling with network size.

In addition, we show the differences between the existing advanced works and this study in Table 2.

Research	Semantic communication		AI model		Security	Scalability
	Intra-twin	Inter-twin	FL	LLM		
[17, 29]	✗	✗	✓	✗	✗	✗
[18, 21, 22]	✗	✗	✓	✗	✓	✗
[30, 31]	✓	✗	✗	✗	✗	✓
[32-37]	✗	✗	✗	✗	✓	✗
LLM-twin	✓	✓	✗	✓	✓	✓

Table 1. The difference between the proposed and existing works.

Abbreviations	Full forms
DTs & DTNs	Digital twins and digital twin networks.
DTENs & DTSNs	Digital twin edge networks and digital twin semantic networks.
LLM & LMs	Large language model and language models
FL	Federated learning.
KB	Knowledge base.

Table 2. Key notations used in this paper.

Framework of LLM-Twin with efficiency communication and data security

In this section, the proposed LLM-Twin framework will be presented in detail, further explaining how LLM-Twin addresses the existing challenges and makes contributions. It is worth mentioning that the proposed LLM-twin references the underlying LLM principles and is generalizable. The first subsection demonstrates the DTSNs, as shown in Fig. 2, which introduces the mini-giants model collaboration scheme for deploying LLMs in resource-limited DTNs. Then, we present a semantic-level redesign of the traditional intra-twin and inter-twin communication, which demonstrates the improvement in communication efficiency brought by LLM-Twin. In addition, since both LLM and DTNs are based on large data-driven, this also poses a huge data security concern. Therefore, we discuss the data security model of the LLM-Twin in the second subsection and propose a data security protection scheme that does not bring additional communication overhead. In addition, the abbreviations commonly used in the remainder of the paper are summarized in Table 2, which will help the reader to find them.

Digital twin semantic networks

Mini-giants model collaboration scheme

Compared to the domain specialization of traditional machine learning models, LLMs have unparalleled natural advantages in DT scenarios with a large number of hybrid data types and multimodal data processing requirements (as shown in Fig. 1). However, the huge demand for computational resources in LLM makes it hard to apply in DTNs that consist of a large number of edge nodes and IoT devices and emphasize real-time performance. Zhou et al[38] propose small language models (LMs), including reducing model parameters and fine-tuning a small number of parameters, which can be trained and used on affordable resources, such as a single GPU. In particular, small LMs based on LoRA[39], QLoRA[40] fine-tuning can approach the performance of LLMs by training only a very small fraction of parameters (one in ten thousand), and can be easily switched between different downstream tasks. Therefore, we first design the small LM scheme with the edge fine-tuning in DTNs.

The core of this scheme is to address (1) state synchronization of DTs and physical entities; and (2) maintenance of DT decision models for entities, which are two important issues for DTNs. Firstly, pre-trained LLMs for specific scenarios are deployed in the DT cloud servers and distributed to the edge physical entities in the region. As shown in Fig. 2, the physical entity gathers its data to perform edge fine-tuning on a pre-trained LLM locally. This process results in a smaller LM that incorporates personal data and decision-making information at a manageable computational cost. Subsequently, the small LM is loaded into the LLM by means of parameter

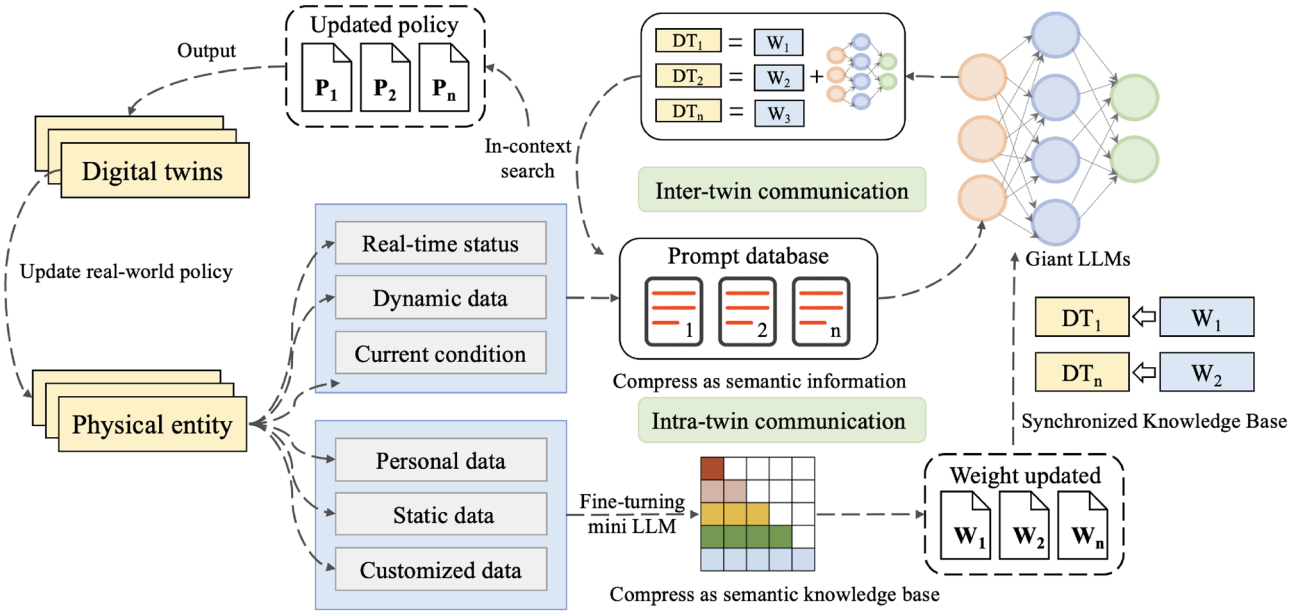

Figure 2. Digital twin semantic network of LLM-Twin networking framework.

updating to realize model updating and maintenance of the DT. In addition, for real-time physical entity state synchronization, we designed the instruction prompt scheme. Physical entities collect dynamic data and state information in real-time and encode this information as prompts which will be uploaded to the prompt database of the DT server.

When DT performs data analysis and decision-making, it conducts the following operations: (1) loads the small LM of the corresponding physical entity; (2) searches the prompt database to obtain the prompt of the corresponding physical entity, and adds it to the current analysis prompts as a context; and (3) sends the aggregated prompts to the LLM, then the RESPONSE is the final decision-making information, which is based on the current data model (the small LM of the physical entity) and the current state information (the physical entity's uploaded prompts). Overall, the proposed scheme addresses the deployment of LLM in DTNs, which reduces the resource overhead of edge devices and cloud servers.

Intra-twin communication

Based on the above scheme, we further proposed DTSN framework concerning the core communication subsystems of DTN, intra-twin communication, and inter-twin communication. Intra-twin communication is responsible for enabling fine-grained mapping of physical entities and DTs. Traditional bit-level communication schemes need to maintain real-time communication and modeling mapping of huge data volumes, which incurs a huge communication overhead. To reduce unnecessary information transmission, semantic communication can achieve optimization of communication information based on a shared semantic knowledge base (KB)[24]. In DTSN, the static data of physical entities and individual preference data are modeled locally as small LM through edge fine-tuning as shown in Fig. 2, which forms its semantic KB. The KB will be uploaded and stored in the DT server. After the DTSN is established, the dynamic information of the physical entities will be encoded as semantic information (prompt) and synchronized to the DT, and the LLM of the DT loading the KB can parse the semantic information and then perform analysis and decision-making. In other words, the KB of the physical entity, the dynamic information (prompt) of the physical entity, and the LLM used for loading constitute the DT of this physical entity.

Compared to traditional communication, DTSN only needs to establish real-time channels for transmitting the compressed semantic information and update the KBs periodically according to the actual situation of physical entities, which greatly reduces the communication overhead.

Inter-twin communication

Inter-twin communication is responsible for enabling information sharing between DTs, which can enable entities to obtain global information and a larger perceptual field without physical communication constraints. The overhead of traditional inter-twin communication is mainly in two aspects: (1) the need to model the communication between different DTs, especially for cross-domain data and multimodal data processing and conversion will bring additional overhead; (2) although the communication is through the virtual link, the nodes or processes between the data transmission, data computation, and decision-making generation will still bring additional communication overhead.

Therefore, we model inter-twin communication in DTSN as a process in which LLM selects the prompts of the communicating parties from the prompt database and adds them to the inference context. First, since the communication data of each DT is encoded locally into a semantic representation (prompt) and then uploaded to the prompt database, there is no need to do further conversion of each modal data in inter-twin communication. Secondly, LLM-Twin does not need to establish additional virtual links, it adds the prompts of the communicating parties directly to the context of executing instructions to start inference and obtain new decisions, which unifies the sharing of information, data computation, and global decision updating.

Data security model

Benefiting from the semantic communication performance and efficient resource allocation of LLM-Twin, it significantly benefits the communication, analysis, and processing of massive data in various DTN scenarios. However, it also brings data security risks. Therefore, in this section, we discuss the data security models of LLM-Twin, which have been taken into account during the design, including the homomorphic model and the one-way security model.

Homomorphism model

The data communication and processing of LLM-Twin are homomorphic, as shown in Fig. 3. The homomorphism is manifested in the fact that the results obtained by uploading the plaintext data into the rule-based DT are the same as the results obtained by loading the model weight information through local fine-tuning and executing it in the LLM of the DT. Therefore, in the LLM-Twin framework, the private information of a physical entity appears only in the local computing environment and exists in non-plaintext form in both communication and computation. As a result, the homomorphism model protects the privacy of personal data while ensuring the consistency of computation results.

It is worth mentioning that in LLM-Twin we assume that the sensitive information is loaded into the DT's LLM by fine-tuning, as shown in Fig. 2. However, the real-time dynamic data of the physical entities encoded as the prompts does not contain sensitive information as it is to be shared with other DTs.The protection of this type of information involves research such as privacy computing and is beyond the scope of this paper.

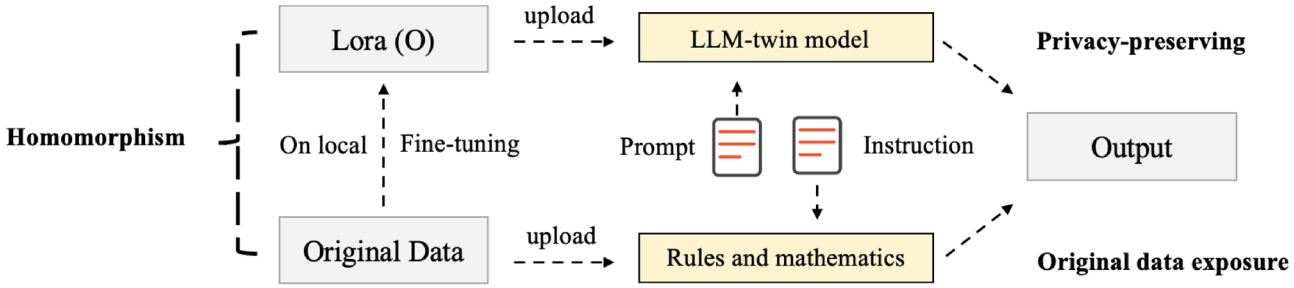

Figure 3. Homomorphism model of LLM-Twin to protect original data.

One-way security model

The one-way security of LLM-Twin is mainly manifested in two parts, as shown in Fig. 4. First, the fine-tuning of LLM is irreversible. The original data can be fine-tuned to train the small LM, but the small LM cannot restore the original data. Further, the local small LM generates the fine-tuned weight file *Lora* which is uploaded to the DT side, but the *Lora* file cannot directly restore the original data.

Second, there is a reversal curse in LLM[41], this study reveals that LLM trained in the fixed pattern $< A\ is\ B >$ cannot answer $< B\ is\ ? >$. This research was intended to reveal the flaws in the logical capabilities of LLM, but we believe that reinforcing this property in LLM-Twin can help secure sensitive data. As previously described, sensitive information of physical entities in LLM-Twin, including personal data, personalization settings, etc., will be loaded into the LLM of DT via edge fine-tuning to complete model construction and KB synchronization. Although this information is loaded into the LLM in a non-explicit form, a malicious server can deduce the original information through continuous interaction with the LLM. Therefore, we will design a training data format to strengthen the reversal curse by using only data forms such as $<$ *Sensitive to Answer* $>$, which makes it impossible for a malicious to get the sensitive information by colliding a large number of $<$ *Answer* $>$. As a result, LLM-Twin contains a homomorphic and one-way security design that incorporates data security without adding additional communication overhead.

Efficiency analysis and security models of LLM-Twin

In this section, we present detailed mathematical modeling of the computation and communication for LLM-Twin presented in the previous section, as shown in Fig. 2, which demonstrates the advantages of the high efficiency of LLM-Twin. Specifically, we analyze the traditional FL-based networking framework of DTNs and our approach, which demonstrates the superiority of the proposed approach by comparing the time to complete one DTN modeling, $T_{LLM-Twin}$ and T_{fl}. Meanwhile, we give a macro security analysis of the whole protocol of LLM-Twin with Universally Composable (UC)[42] framework based on the design of one-way security and homomorphism in the previous section, which provides a more comprehensive framework for proving the security of LLM-Twin. In addition, we show the key notations and parameters of this section in Table 3.

Computation and communication model

The traditional FL-based DTN paradigm $DT_i(t)$[17] is shown in Eq. (1), which contains the decision model M_i, the virtual network N_i, the historical data H_i, the state information S_i, and the shared data d_i, which are synchronously mapped and constructed from the physical entities u_i. Specifically, DT_i continuously interacts with the physical entity u_i to maintain consistency, which involves model synchronization and state synchronization. DT trains the

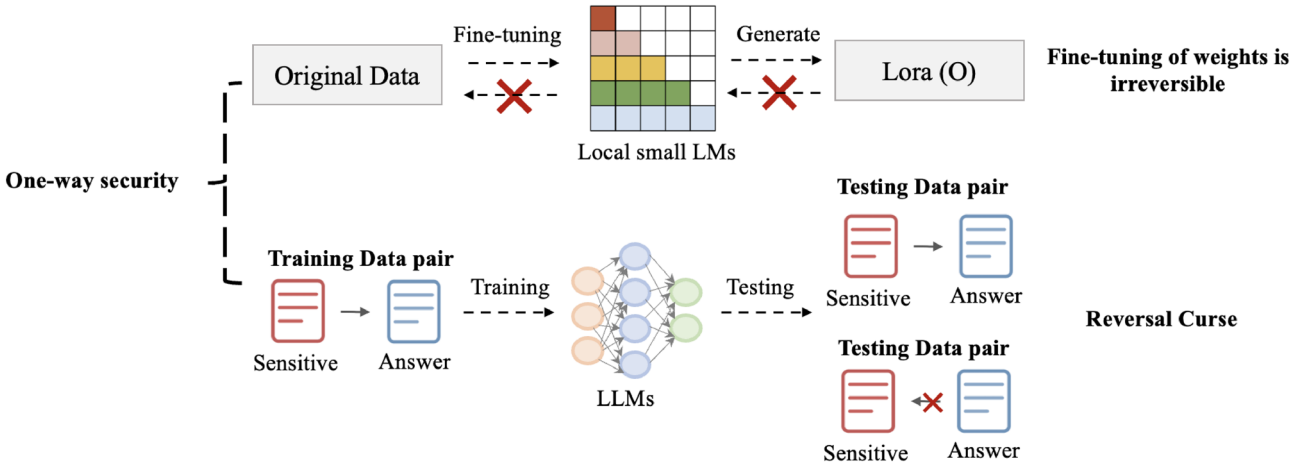

Figure 4. One-way security model of LLM-Twin.

Notation	Interpretation
M	A decision model built on physical data, which can react based on the rules and states of real devices.
N	Virtual networks of the DTN, which are generally modeled through computation, data access, and so on.
H	Historical data (including state, rules, etc.) for physical entities used to construct DTs.
S	Current state information for constructing real-time replicas of physical entities.
d	Information from other DTs via the virtual network. (inter-twin communication)
w	Machine Learning Models, training parameters.
ε	The number of CPU cycles required to execute a unit of model training.
α	The number of CPU cycles required to execute a unit of model aggregation.
f	The CPU cycle frequency.
u	Physical entities or edge devices corresponding to DTs.
g	Cloud servers or edge servers for building DTs.

Table 3. Key notations used in this section.

model by collecting historical data and state information, which is used to analyze and make decisions based on the current real-time state and shared data. In contrast, we propose LLM-Twin and redesign $\widetilde{DT}_i(t)$ in Eq. (1). The cloud LLM model $\widetilde{M}_i$ contains the static state information $S_{s,i}$ of the physical entities when synchronizing the model, without additional state synchronization, as shown in Fig. 2. Meanwhile, the $\widetilde{M}_i$ itself is a virtual network where the DTs can share their information, which means that additional construction is not required. For the history information $\widetilde{H}_i$, each physical entity uploads from local to the prompt database via semantic communication, which contains prompt history, real-time state, and shared data.

$$\begin{cases} DT_i(t) = \Gamma(M_i, N_i, H_i, S_i, d_i, t) \\ \widetilde{DT}_i(t) = \Gamma(\widetilde{M}_i, \widetilde{H}_i, t) \\ \widetilde{M}_i = M_i + N_i + S_{s,i} \\ \widetilde{H}_i = \text{Prompt}(u_i, ..., u_n) = \sum_{i=1}^{n}(H_i + d_i + S_{d,i}) \end{cases} \tag{1}$$

In the traditional FL scheme, the DT model is constructed by distributed training. The physical entity u_i trains local weights using local historical data based on the loss function

$$\begin{aligned} L_i(w) &= \frac{1}{|H_i|} \sum_{x_j, y_j \in H_i} l(w, x_j, y_j) \\ L_g(w) &= \frac{1}{|H_g|} \sum_{i=1}^{n} L_i(w) \end{aligned} \tag{2}$$

where $x_j, y_j \in H_i$ is the samples of training data. DT will aggregate the weights of each entity and perform the next round of distributed training to minimize the aggregation loss function $L_g(w)$, where $|H_g| = \sum_{i=1}^{n} |H_i|$ is the total size of data from participating physical entities. Specifically, the local parameter update and the global parameter aggregation are shown in Eq. (3).

$$\begin{aligned} w_i(t) &= w(t-1) - \eta \nabla L_i(w(t-1)) \\ w(t) &= \frac{1}{|H_g|} \sum_{i=1}^{n} H_i w_i(t) \end{aligned} \tag{3}$$

However, the full-parameter training and multiple rounds of iterations of the above schemes result in significant computation and communication overheads. Therefore, we propose the LLM-Twin with edge fine-tuning based on LoRA[39] in Eq. (4), which only requires training a few parameters to get a good alignment for LLM.

$$\max_{w} \sum_{(x,y) \in Z} \sum_{t=1}^{|y|} \log \left(P_{w + \Delta \widetilde{w}} \left(y_t | x, y_{<t} \right) \right), \quad |\widetilde{w}| \ll |w| \tag{4}$$

The above equation shows the LLM is initialized with weights w and updated with $w + \Delta \widetilde{w}$ by maximizing the conditional language modeling objective, where $\widetilde{w}$ is much smaller than w, 1% or less. Furthermore, based on the above preliminary knowledge, we can obtain the time consumption T of the traditional DTN and the time consumption $\widetilde{T}$ of the proposed LLM-Twin. We define the CPU cycle frequency of the physical entity u_i as f_{u_i}, ξ denotes the number of CPU cycles required for each data unit when training the model, and α denotes the number of CPU cycles required for each parameter unit when aggregating the parameters by the DT server. Finally, we get

$$T_{u_i}^{cmp} = \frac{\xi_i |H_i|}{f_{u_i}} |w_i|, \quad T_{g_j}^{cmp} = \frac{\alpha_j \sum_{i=1}^{n} |w_i|}{f_{g_j}} \tag{5}$$

$$\widetilde{T}_{u_i}^{cmp} = \frac{\xi_i |S_{s,i}|}{f_{u_i}} |\widetilde{w}_i|, \quad \widetilde{T}_{g_j}^{cmp} = \frac{\alpha_j |\widetilde{w}_i|}{f_{g_j}}, \quad |S_{s,i}| \ll |H_i|, |\widetilde{w}_i| \ll |w_i| \tag{6}$$

where $T_{u_i}^{cmp}$ and $T_{g_j}^{cmp}$ respectively denote the physical entity computation time and the server computation time for constructing a DTN based on the traditional FL; similarly, $\widetilde{T}_{u_i}^{cmp}$ and $\widetilde{T}_{g_j}^{cmp}$ denote the computation time of LLM-Twin. In addition, to further model the communication consumption, we define the data transmission rate between a physical entity u_i and a DT server g_j

$$\begin{aligned} r_{u_i,g_j} &= \frac{c_{u_i}}{C_0} B \log\left(1 + \gamma_{u_i,g_j}\right) \\ \sum_{i=1}^{n} c_{u_i} &+ \sum_{j=1}^{m} c_{g_j} \le C_0 \end{aligned} \tag{7}$$

where C_0 denotes the total number of subchannels over the whole bandwidth W, the number of subchannels allocated to the physical entity u_i is c_{u_i}, and γ_{u_i,g_j} denotes the channel state. Further, we define the intra-twin communication time $T_{u_i}^{intra}$ and the inter-twin comminication time $T_{g_j}^{inter}$ for the FL method in Eq. 8, where $T_{u_i}^{intra}$ contains the data size $|w_i(t)|$ for model synchronization and the data size $|S_i|$ for entity state synchronization. And then, $T_{g_j}^{inter}$ is modeled as the time required for parameter aggregation in the DT server.

$$T_{u_i}^{intra} = \frac{|w_i(t)| + |S_i|}{r_i}, \quad T_{g_j}^{inter} = T_{g_j}^{cmp} \tag{8}$$

Similarly, we get the LLM-Twin communication time

$$\begin{aligned} \widetilde{T}_{u_i}^{intra} &= \frac{|\widetilde{w}_i(t)| + |\widetilde{S}_{d,i}|}{r_i}, \quad \widetilde{T}_{g_j}^{inter} = \frac{|\widetilde{H}_i|}{r_{N_j}} \\ \widetilde{S}_{d,i} &= Semantic(S_{d,i}), \quad |\widetilde{S}_{d,i}| \ll |S_{d,i}| \ll |S_i|, \quad r_{N_j} \gg r_j \end{aligned} \tag{9}$$

where $|\widetilde{S}_{d,i}|$ denotes the state information that needs to be transmitted dynamically, and r_{N_j} denotes the transmission rate of the virtual network, such as the parameter aggregation computation rate of the server in the FL method. Obviously, DTNs rely on communication and computation for their construction and maintenance, so communication and computation are the core components for analyzing the efficiency of DTNs. Therefore, we obtain the time consumption T_{fl} of the FL-based DTN construction method by counting the total communication time and computation time in Eq. (10).

$$\begin{aligned} T_{fl} &= K\left(\max\left\{T_{u_i}^{cmp}, i = 1, ..., n\right\} + T_{g_i}^{cmp} + \frac{|w_i(t)| + |w_i(t+1)|}{r_i}\right) + \frac{|H_i(t)|}{r_i} \\ &= K\left(\frac{\xi_i |H_i(t)||w_i(t)|}{f_{u_l}} + \frac{\alpha_j \sum_{i=1}^{n} |w_i(t)|}{f_{g_j}}\right) + \frac{K(|w_i(t)| + |w_i(t+1)|) + |H_i(t)|}{r_i} \end{aligned} \tag{10}$$

Similarly, we obtain the time efficiency equation for LLM-Twin

$$\begin{aligned} T_{LLM-Twin} &= \frac{1}{\lambda}\left(\widetilde{T}_{u_i}^{cmp} + \widetilde{T}_{g_j}^{cmp} + \frac{|\widetilde{w}_i(t)|}{r_i}\right) + \frac{|\widetilde{S}_{d,i}|}{r_i} + \widetilde{T}_{s_j}^{inter} \\ &= \frac{1}{\lambda}\left(\frac{\xi_i |S_{s,i}|}{f_{u_i}} |\widetilde{w}_i(t)| + \frac{\alpha_j |\widetilde{w}_i(t)|}{f_{g_j}} + \frac{|\widetilde{w}_i(t)|}{r_i}\right) + \frac{|\widetilde{S}_{d,i}|}{r_i} + \frac{|\widetilde{H}_i(t)|}{r_{N_j}} \end{aligned} \tag{11}$$

where K denotes the number of rounds required for convergence since FL involves multiple iterations. Differently, as mentioned in the previous section, LLM-Twin is not required to update the static information (semantic knowledge base) in real-time, so λ denotes the periodicity of the update of the static information. Finally, according to the following constraints

$$\begin{cases} |\widetilde{S}_{d,i}| \ll |S_i|, \quad |S_{s,i}| < |S_i|, \quad |\widetilde{w}_i| \ll |w_i| \\ |S_{s,i}||\widetilde{w}_i| - |S_i||w_i| < 0 \\ |\widetilde{w}_i| - \sum_{i=1}^{n} |w_i| < 0 \\ (|\widetilde{w}_i| + |\widetilde{S}_{d,i}|) - (|w_i| + |S_i|) < 0 \end{cases} \tag{12}$$

we will compare the efficiency of LLM-Twin with the traditional FL method. To demonstrate the advantages of LLM-Twin more intuitively, we assume that the FL method completes in just one round of iterations, i.e., $K = 1$. In addition, it is assumed that LLM-Twin synchronizes static information in real-time, i.e., $\lambda = 1$. Despite this assumption, we still get that LLM-Twin consumes less time than the FL approach in Eq. (13).

$$\begin{aligned} T_{LLM-Twin} - T_{fl} = & \left(\frac{\xi_i \left| \widetilde{S}_{s,i} \right| \left| \widetilde{w}_i(t) \right| - \xi_i |S_i| |w_i(t)|}{f_{u_i}} \right) + \left(\frac{\alpha_j \left| \widetilde{w}_i(t) \right| - \alpha_j \sum_{i=1}^{n} |w_i(t)|}{f_{g_j}} \right) \\ & + \frac{(\left| \widetilde{w}_i(t) \right| + \left| \widetilde{S}_{d,i} \right|) - (|w_i(t)| + |S_i|)}{r_i} + \frac{\left| \widetilde{H}_i(t) \right|}{r_{N_j}} \quad < \quad 0 \end{aligned} \tag{13}$$

From the above mathematical analysis, it can be seen that LLM-Twin is efficient mainly in the following aspects: (1) Instead of synchronizing all state information S_i in real time, LLM-Twin only needs to synchronize the dynamic semantic information $\widetilde{S}_{d,i}$ in it; (2) LLM-Twin does not need to perform parameter aggregation and iteration of participating entities to accomplish data sharing while training the model, instead, it accomplishes data sharing during model inference by searching the prompt database $\widetilde{H}_i(t)$; (3) Instead of training the model with full parameters $w_i(t)$, LLM-Twin only needs to fine-tune a very few parameters $\widetilde{w}_i(t)$.

Security analysis of LLM-Twin

Universally composable security analysis

In this section, we formalize LLM-Twin as a service protocol,

Protocol Description
Data Upload:
1. Data provider A possesses data (E, P).
2. A performs some local operation to transform (E, P) to (E', P').
3. A uploads (E', P') to third-party C.
Service Function Computation:
1. Upon receiving (E', P'), C processes it to obtain $F(E, P)$.
Data Retrieval:
1. Service requestor B sends a request q to C.
2. C computes $L(q, F(E, P))$, retrieves P, and sends it to B.

where (E, P) denotes the training data pair, E is the LLM prompt, and P is the corresponding completion. (E, P) is fine-tuned to (E', P') by edges and uploaded to C. C denotes the DT server that obtains the DT model $F(E, P)$ by merging the fine-tuning parameters, which can respond to B's request q, and then respond with the completion P to B. In addition, the protocol has the following three security properties:

Security Properties
1. $F(E, P)$ can be used to obtain P if the q corresponding to E is known.
2. Knowing P alone, one cannot retrieve E.
3. $F(E, P)$ and (E', P') do not allow the recovery of E or P, ensuring the privacy of the original data.

It is obvious by the principle of LLM that q as a prompt can obtain the corresponding completion P. From the one-way security in the previous section, as shown in Fig. 4, it is not possible to derive the training-time sensitive data E by completion P, and it is not possible to recover the original E and P from the model and weights. Similarly according to the homomorphic design, as in Fig. 3, the LLM model and the fine-tuned weights can accomplish the service while guaranteeing E and P privacy. Based on the above protocol and security properties, we define the ideal-world LLM-Twin function $\mathcal{F}_{DATA}$.

Ideal Functionality $\mathcal{F}_{DATA}$
Upon receiving ("***upload***", A, C, E, P) ***from A:***
1. Store (E, P) internally.
2. Send ("receipt", A, C) to C.
Upon receiving ("***request***", B, C, q) ***from B:***
1. Compute P using E and $F(E, P)$.
2. Send ("response", B, C, P) to B.
If B is corrupted:
1. Upon receiving ("corrupt", B) from $\mathcal{S}$, mark B as corrupted.
2. If simulator $\mathcal{S}$ issues a ("request", B, C, q), simulate P or altered P' (if $\mathcal{S}$ wants to simulate a cheating B) based on E and $F(E, P)$ and send ("response", B, C, P') to B.

In ideal functionality $\mathcal{F}_{DATA}$, A refers to the physical entity that performs edge fine-tuning, C refers to the DT model, and B refers to the service requester, e.g., other DTs. The important point is that $\mathcal{F}_{DATA}$ is security as it is based on a fully trusted and ideal third party. In addition, external environments$\mathcal{Z}$ and adversaries $\mathcal{A}$ will observe and attack the protocol. As a result, the simulator $\mathcal{S}$ is used to simulate the attack behavior of the adversary $\mathcal{A}$ in the ideal world and interacts with the external environment $\mathcal{Z}$, making it impossible for $\mathcal{Z}$ to distinguish between the ideal-world protocols and the real LLM-Twin protocol.

Simulator
Simulator $\mathcal{S}$ when corrupting A:
1. A's corruption isn't significantly impactful because the transformation from (E, P) to (E', P') occurs before the data's interaction with C. The simulator simply follows the honest protocol on behalf of A.
Simulator $\mathcal{S}$ when corrupting C:
1. Upon receiving (E', P') from adversary $\mathcal{A}$:
• Send ("upload", A, C, dummy_E, dummy_P) to $\mathcal{F}_{DATA}$.
2. When $\mathcal{A}$ processes a request from B:
• Send ("request", B, C, q) to $\mathcal{F}_{DATA}$.
• Upon receiving P from $\mathcal{F}_{DATA}$, send it to $\mathcal{A}$.
Simulator $\mathcal{S}$ when corrupting B:
1. Upon B's corruption, $\mathcal{S}$ sends ("corrupt", B) to $\mathcal{F}_{DATA}$.
2. $\mathcal{S}$ intercepts q from $\mathcal{A}$.
3. $\mathcal{S}$ sends ("request", B, C, q) to $\mathcal{F}_{DATA}$.
4. When $\mathcal{F}_{DATA}$ responds with P, $\mathcal{S}$ relays this to $\mathcal{A}$ or alters the message to P' if $\mathcal{S}$ simulates a cheating B trying to alter the data.

Based on the above simulator design, for every adversary $\mathcal{A}$ in the real world, there exists an ideal-world simulator $\mathcal{S}$ and an ideal function $\mathcal{F}_{DATA}$ such that there is no environment $\mathcal{Z}$ that can distinguish between the two worlds. Specifically, (1) Neither $\mathcal{A}$ in the real world nor $\mathcal{S}$ in the ideal world can extract original training data E from interactions involving only P. (2) $\mathcal{S}$ can simulate any actions by altering the response to $\mathcal{A}$ or by sending modified queries. The ideal world captures any dishonest behavior that B could exhibit in the real world. (3) Given the simulator's capabilities and the ideal functionality, any environment $\mathcal{Z}$ cannot distinguish between the real world. Therefore, the LLM-Twin security framework based on UC is introduced. Since the ideal world is security, and the ideal world is made indistinguishable from the real world by constructing ideal functions and simulators, it is demonstrated that the real-world protocol, i.e. LLM-Twin, is UC security.

Potential threats and countermeasures

In this section, we will first summarize the potential security threats in DTNs. Further, to better demonstrate the security advantages of LLM-twin framework, we will compare the security performance and countermeasures of LLM-twin and mainstream FL-based DTNs. We discuss potential security threats to DTNs from four aspects, data/device heterogeneity threat and single point of failure threat in functional safety; data poisoning, and privacy threat in information security.

1. Data/device heterogeneity threat. The presence of a large number of physical devices with asymmetric performance and heterogeneous distributed data in DTNs will lead to difficulty in the convergence of the twin model training process, inefficient training as well as degradation of model quality.
2. Single point of failure threat. DTNs bridge the physical and information worlds, which means that a single point of failure in a physical device will threaten secure data sharing on the information side, leading to corrupted global models or even widespread error propagation across DTs.
3. Data poisoning. DT is based on a data-driven paradigm and DTNs prompt broader data sharing and global decision making. This means that an attacker can upload poisoned data or perform backdoor attacks by modifying or controlling a participant's physical client, leading to the generation of low-quality or even malicious global models, which ultimately result in a wide-scale attack impact.
4. Privacy threat. As the most important asset in DTNs, data will face a huge privacy threat. In intra-twin communication, an attacker can eavesdrop on a large amount of interactive data between physical entities and DTs. In inter-twin communication, the attacker can infer or extract the original sensitive data based on the output information of different DTs.

In response to the above threats, we will compare and analyze the security of LLM-twin-based and FL-based DTNs frameworks. First is the data/device heterogeneity threat. Traditional FL algorithms are based on the data assumption of Independent Identical Distribution (IID), which cannot cover the data distributions of all participants when dealing with Non-IID data, which can seriously affect the model performance. In particular, the efficiency of FL training is affected by each participating device, and physical devices with asymmetric performance will significantly reduce the efficiency of global FL training. Differently, the fine-tuning process of each participating device in LLM-twin is independent, where the training and information synchronization of a single participant does not affect other participants. In addition, local fine-tuning allows each participant to implement personalized training and build personalized DTs based on their requirements, which contributes to mitigating the data/device heterogeneity threat.

Against the threats of a single point of failure and data poisoning, LLM-twin has a natural advantage. The fine-tuned knowledge is only "mounted" to the LMs of DTNs without changing the original parameters and weights of the LMs. This means that in the LLM-twin framework, a single point of failure information and poisoning cannot directly affect the knowledge of the global model and cause extensive damage. Differently, the FL-based architecture will aggregate the weights of the participants and update the parameters of the global model at the DT side, which will make it difficult to avoid malicious data from corrupting the global model.

Against privacy threats, we discuss the one-way security and homomorphic security of LLM-twin in "Related works" section, which can effectively mitigate privacy leakage and inference attacks. It is worth mentioning that the FL-based DTNs architecture also achieves privacy protection and enhancement through distributed local training. The difference is that the traditional FL architecture lacks the protection of sensitive information extraction on the DT side. We design a sensitive information protection strategy based on the reversal curse for LLM-twin in terms of one-way security, which mitigates the attacker's inference attack on sensitive information at the DT side and further enhances privacy protection.

Evaluation and case study

In this section, we first show the numerical experimental results of LLM-Twin in terms of computation and communication consumption and demonstrate the advantages of LLM-Twin by comparing its performance with traditional FL-based DTNs. Then, to further demonstrate the feasibility and advantages of LLM-Twin, we present a case study of a smart home DTN. All of the experiments are deployed on the high-performance server with the configuration of Intel Xeon Gold 6226R 3.90 GHz CPU, 256 GB RAM, Ubuntu 20.04 LTS and Python 3.8.

Performance analysis

In the FL method, the edge physical entities use the historical data to train the model locally and get the distributed model weights, where the time consumed for this part is denoted as $T_{u_i}^{cmp}$. Moreover, the time when the weights of all physical entities are aggregated in the DT server is denoted as $T_{g_i}^{cmp}$. In LLM-Twin, the physical entities use current static state data to fine-tune locally for obtaining the small LM, and this part of the computation time is denoted as $\widetilde{T}_{u_i}^{cmp}$. The weight files of the small model are loaded into the large model in the DT server, and this part of the time is denoted as $\widetilde{T}_{g_i}^{cmp}$.

Subsequently, we evaluated the computational performance of the FL method and LLM-Twin respectively in two cases with total parameters w of 3.5B and 7B, as shown in Fig. 5. The results showed that LLM-Twin consumes only %1 of FL or even less time in edge physical entity training. In addition, as more historical data becomes available, the time taken by the FL method will continue to increase over time. Further, Fig. 6 illustrates the time required for the DT server to aggregate the weights as the parameter size increases for different numbers of physical entities. Since FL needs to aggregate the weights of each entity in the computation phase to accomplish

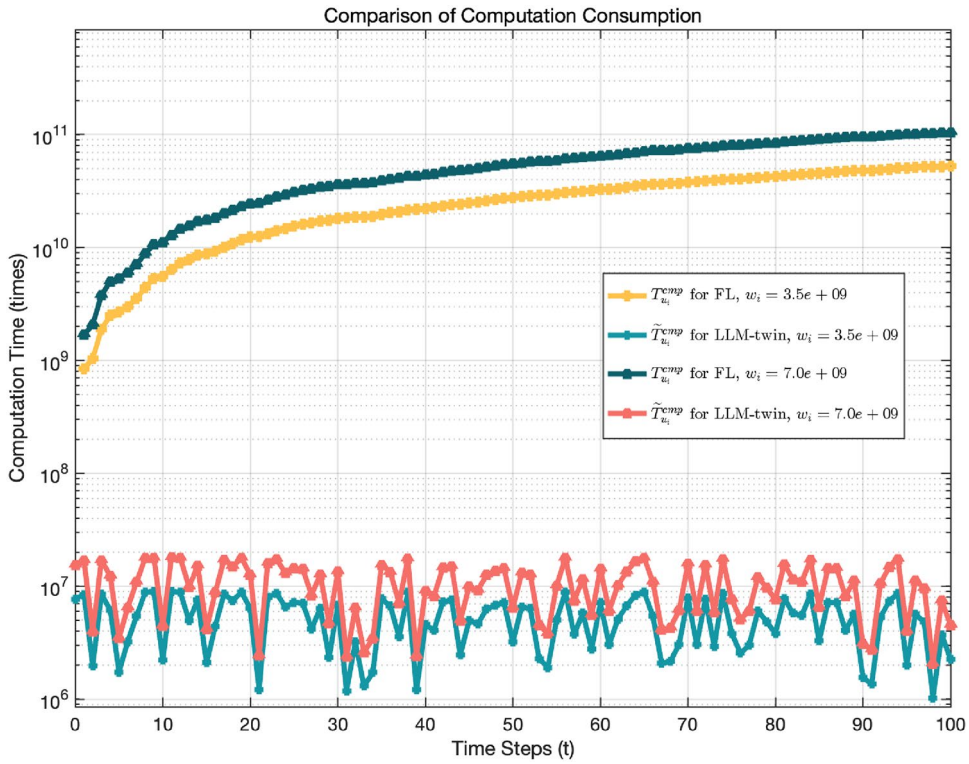

Figure 5. Comparison of physical entity computation consumption between LLM-Twin and FL-based DTN.

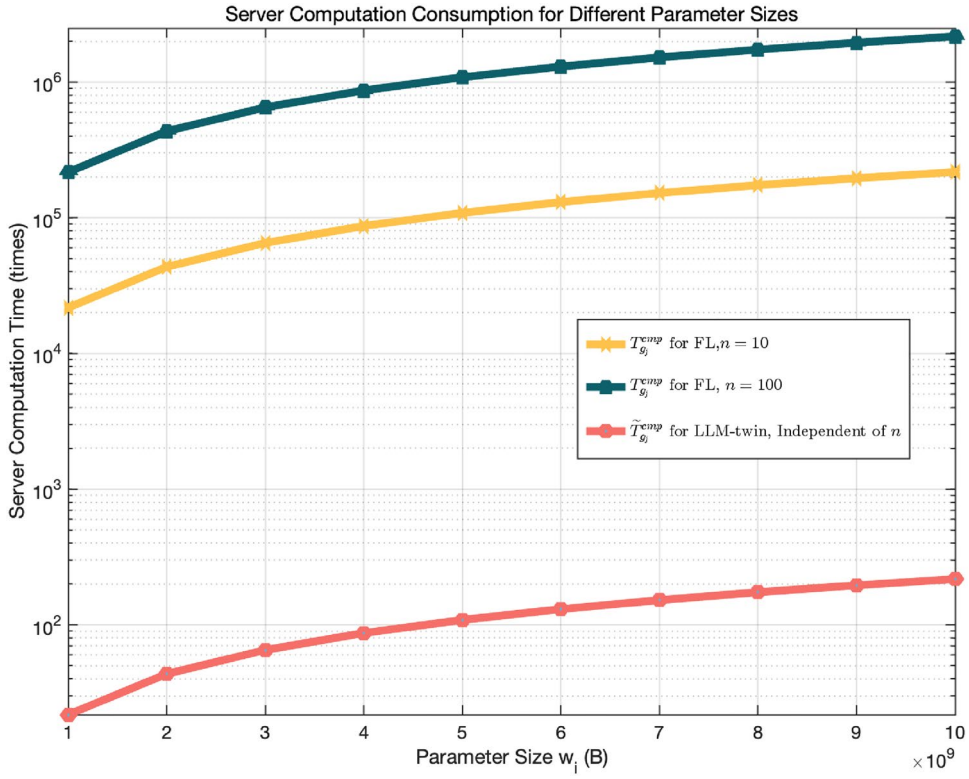

Figure 6. Comparison of DT server computation consumption between LLM-Twin and FL-based DTN.

information sharing, the number of entities will significantly affect $T_{g_i}^{cmp}$. In contrast, in the computation phase, LLM-Twin does not need to consider the parameters of the other DTs but only loads the weights of the corresponding entities. Therefore $\widetilde{T}_{g_i}^{cmp}$ will not be affected by the number of entities and also less computation leads to much less computation time than the FL method, as shown in Fig. 6.

Next, the communication performance will be evaluated, including the core communication architectures of the DTN: intra-twin communication and inter-twin communication. In the FL approach, intra-twin communication is denoted as $T_{u_i}^{intra}$, which includes the model's weight update and the state synchronization of the physical entities. In FL, inter-twin communication is modeled as the weight aggregation of each entity, denoted as $T_{g_i}^{inter}$, which is the same as $T_{g_i}^{cmp}$, as shown in Eq. 8. In contrast, intra-twin communication for LLM-Twin is modeled as the synchronization of the semantic knowledge base, denoted as $\widetilde{T}_{u_i}^{intra}$, which is achieved by transmitting the fine-tuned weights to the DT server. Inter-twin communication is modeled as searching context information from the prompt database, denoted as $\widetilde{T}_{g_i}^{inter}$. Finally, the results of the evaluation are shown in Fig. 7.

Since the communication content of FL is always model parameters and state information, the communication cost will not be affected by time. The inter-twin communication of LLM-Twin requires searching for a specific context in the prompt database. Since the prompt database is constantly receiving and storing dynamic information from all physical entities, it generates a large number of historical records over time and thus affects $\widetilde{T}_{g_i}^{inter}$. Most importantly, the LLM-Twin is based on a fine-tuned mini-giant modeling scheme that allows far fewer parameters to be transmitted in the communication than the FL approach. In addition, LLM-Twin's semantic communication-based approach makes intra-twin communication transmit less state information data, as shown in Eq. 9. Therefore, the communication efficiency of the LLM-Twin is still much higher than the FL, as shown in Fig. 7, and the communication content is much less than the FL, as shown in Fig. 8.

Finally, we compare the overall efficiency of DTNs constructed based on FL and LLM-Twin, and the evaluation results are shown in Fig. 9. Since FL requires multiple rounds of weight updates to complete model convergence, which means that the number of iteration rounds significantly affects the DTN efficiency. On the contrary, LLM-Twin does not need to update the entity static information and model in real-time, as shown in Eq. 11, which brings higher efficiency. In addition, it can be seen from Fig. 9 that the accumulation of historical data due to time passing has a significant impact on the efficiency of FL-based DTN, but has less impact on LLM-Twin.

Case study in smart home digital twin network

To further demonstrate the feasibility and practical performance of the LLM-Twin, a case study based on a smart home DTN is designed in this section, as shown in Fig. 10. Next, we will explain the case flow in detail based on Fig. 10. Unlike the traditional DT framework, the proposed networking framework for DTNs is a LLM-driven approach. As discussed in "Related works" section, the key to realizing LLM-twin is to build the mini-giant model

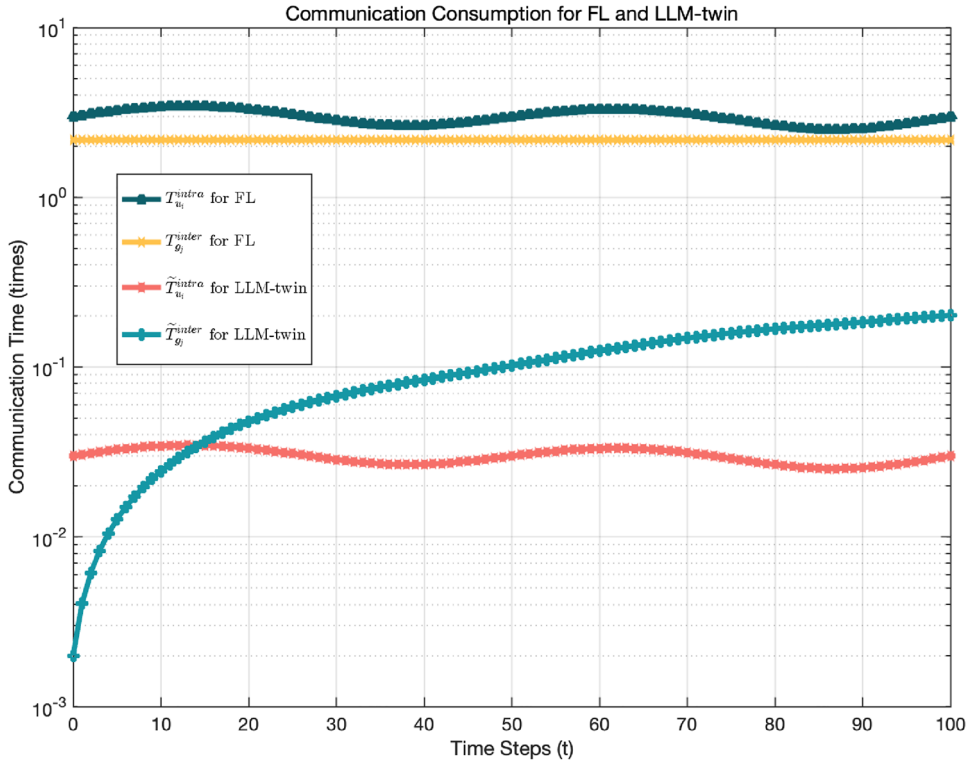

Figure 7. Comparison of communication consumption between LLM-Twin and FL-based DTN.

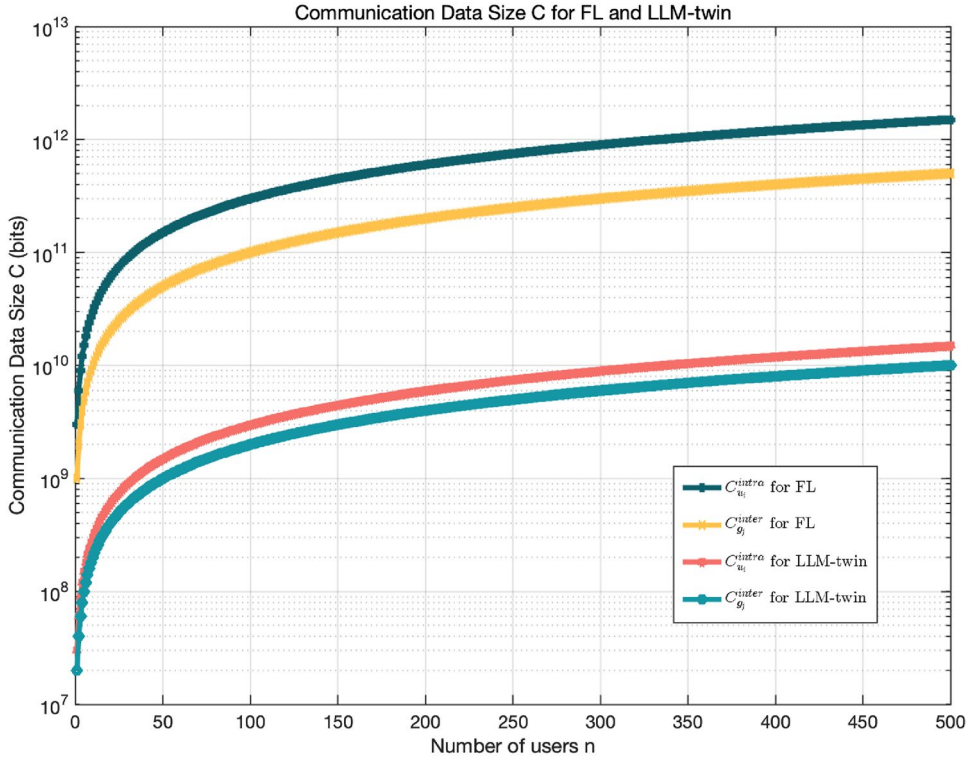

Figure 8. Comparison of communication content size between LLM-Twin and FL-based DTN.

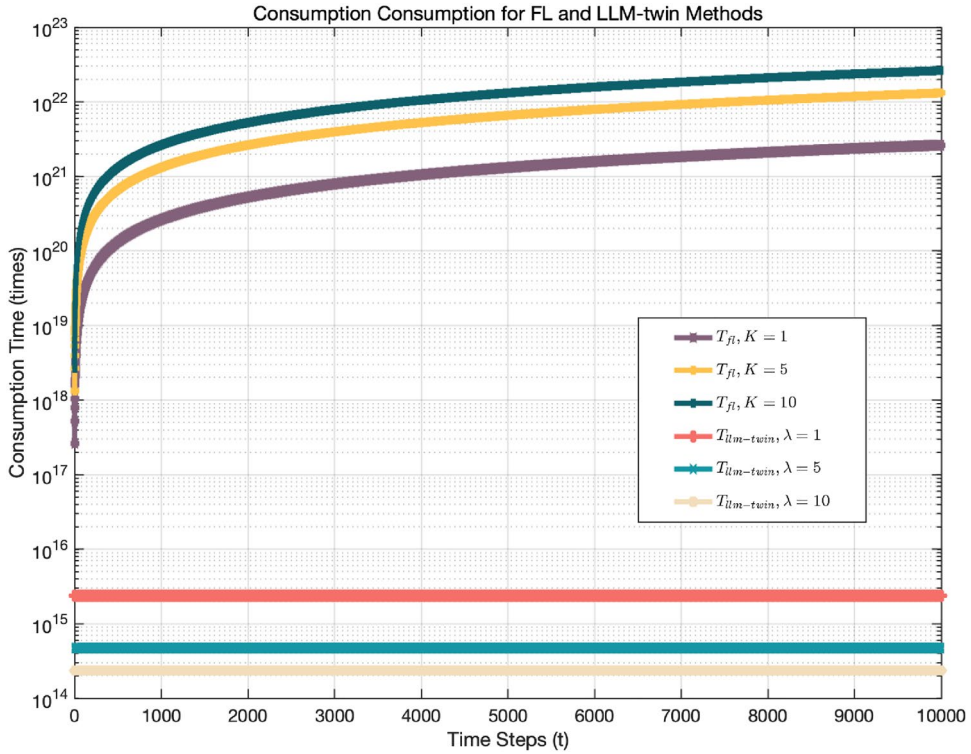

Figure 9. Comparison of overall DTN consumption between LLM-Twin and FL-based DTN.

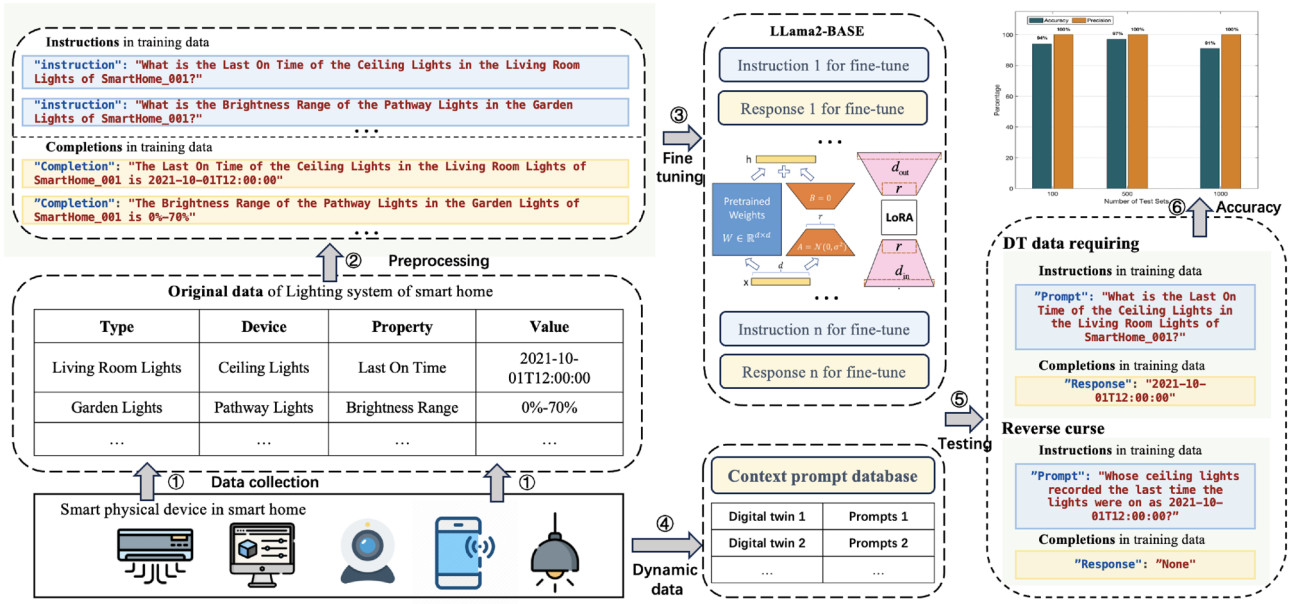

Figure 10. Case study of LLM-Twin in smart home digital twin network.

collaboration between the edge and the server. In this case study, we use the open-source model LLama7B and the fine-tuning tool QLora to implement the simulation.

Specifically, We first collected and generated 1.6k state data of the smart home, which includes system types, devices, property, and state values. This data will be encoded into an LLM fine-tuned dataset in a local Python environment, which will follow the form < prompt, completion>. Then we execute Lora fine-tuning with the LLama7B model locally. The proposed mini-giant framework demonstrates its advantages. The memory footprint during local fine-tuning is 5878 MB, which satisfies the performance requirements of most edge devices. In addition, the loss and training time for model fine-tuning is shown in Fig. 11. The model converges after 100 seconds of training and regionally stabilizes after about 400 seconds. After completing the fine-tuning, we uploaded the fine-tuned weights file to the server and loaded the local information into LLama7B through weight merging.

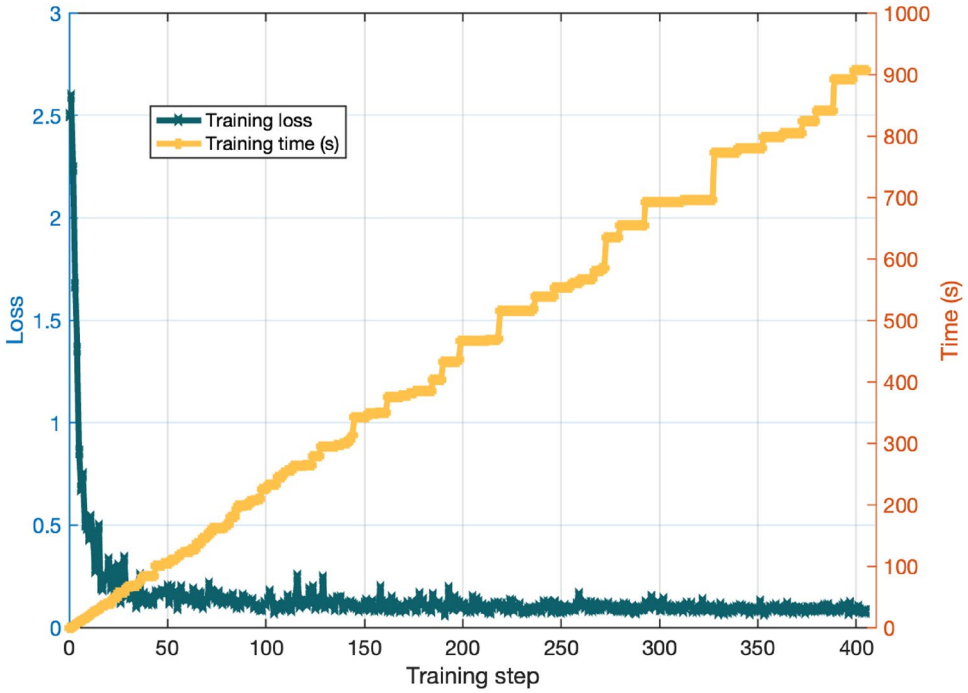

Figure 11. Training losses and training time.

In addition, we uploaded 500 prompts describing the dynamic information of physical entities to the server to be stored as a prompt database. At this point, we have completed the simulation of intra-twin communication. For inter-twin communication, we used a simple traversal-based string-matching algorithm in the case study. By retrieving 500 prompts and then outputting dynamic information entries of specific DTs.

In the case study, we first simulate a user constructing a prompt and sending it to its DT. The DT generates the response by integrating the information in the prompt database and its knowledge. Specifically, *smarthome*1 wants to know the brightness of his lights and that of his neighbor *smarthome*2. In the simulated LLM-twin framework, we have the following process. First, *smarthome*1 sends a prompt to his DT, *smarthome*1_*DT*, asking about the brightness of his own and his neighbor's lights. Since *smarthome*1_*DT* has been loaded with knowledge about the state of *smarthome*1 via intra-twin communication, it can directly answer the light brightness of *smarthome*1. For the light information of *smarthome*2, *smarthome*1_*DT* will retrieve the prompt database and add it to the initial prompt as a context of the final prompt. Finally, the LLM in *smarthome*1_*DT* generates the lighting information based on the final prompt and sends the answer to *smarthome*1. In this way, a simple service request process for DTNs is completed. In addition, a further simulation could consider output decisions by DTs combining their information with that of other DTs, e.g., comparing the brightness of the lights in *smarthome*1 and *smarthome*2 and suggesting turning down the brightness.

Further, to better demonstrate the actual performance of LLM-twin as well as to get a more complete view of the effectiveness and scalability, we tested the detailed delay results of the inter-twin and intra-twin communication in this case, as shown in Table 4. There are four sets of data from smart homes of different sizes that we tested separately. Among them, intra-twin communication includes local fine-tuning, weight uploading, weight merging, and synchronization of dynamic information. inter-twin communication refers to the time to retrieve dynamic information entries. Service & Inference refers to the time to complete once DTNs service with the deployed LLM-twin architecture, which includes sending prompts to the DT server, DT retrieving the data, constructing the final prompt, completing the inference, and responding to the completion. As shown in Table 4, except for fine-tuning and weight merging, the efficiency of the rest of the sessions is significant. Moreover, in the

Sample	Intra-twin comm.				Inter-twin	Service
size	Fine-tuning	Weights upload	Weights merge	Dynamic syn.	comm.	& Inference
500	227s	1.4s	44.1s	0.7ms	0.3s	2.74s
900	302s	2.2s	43.2s	1.3ms	0.3s	2.73s
1400	407s	3.4s	44.3s	2.1ms	0.3s	2.67s
1600	420s	4.0s	44.3s	2.4ms	0.3s	2.71s

Table 4. Communication delay of LLM-Twin.

proposed LLM-twin, fine-tuning and weight merging are executed periodically. This means that these processes need to be executed only when the attributes and static data of the physical devices are significantly changed. Moreover, the time-consuming fine-tuning is executed locally, and this process is independent and thus does not affect the service efficiency of DTNs and other DTs. In addition, according to Table 4, the amount of data for physical entities increases, which will increase the delay of fine-tuning, weight uploading, and dynamic synchronization, but does not have a significant effect on reasoning and weight merging. Especially, the delay of fine-tuning and weight uploading does not directly affect the service efficiency of DTNs, so this implies that the LLM-twin has good robustness and scalability for different data volumes of the physical side.

In addition, to provide a more in-depth discussion of the scalability of LLM-twin, we simulate the performance impact of increasing network size in our case study. LLM-twin is an LLM-driven networking framework for DTNs, and the performance requirements are focused on calls to GPUs. Therefore, we mainly consider the network size of LLM-twin on the demand of local device GPU graphics memory and server GPU graphics memory. As shown in Fig. 12, local fine-tuning and initial model loading of the server are not affected by the size of the network. This is because they are only related to the parameter size of the LM used. In the case we are using the LLama7B model, the initial loading and local fine-tuning consume about 4600MiB and 6500MiB of graphics memory, respectively. This consumption is acceptable for normal local devices and servers. In addition, since weight merging can only be performed on the DT server, if all nodes perform weight merging at the same time (the limit case), it will incur a huge memory overhead. We simulate the limit of memory overhead for weight merging with different network sizes, and it exceeds 50GB of memory requirement when there are 5 nodes. However, as mentioned before, weight merging is executed periodically, and simultaneous execution by all nodes is the extreme case. Therefore, the experiments demonstrate the theoretical overhead limit of weight merging as the network size increases, which is still optimistic for large servers dedicated to LLM deployment.

Further, we tested and evaluated the accuracy and precision of the LLM-twin output in our case. We generated a large number of test prompts to detect whether LLM can correctly generate information about its corresponding DTs. In this part of the test, we also further validate the proposed data security design. When training LLM-Twin, we strengthen the reversal curse of LLM by fixing the training set's order of discourse, e.g., $< object, description >$. *object* is known only to the DT owner. Therefore, an attacker cannot obtain the DT owner's *object* by submitting a large number of *descriptions* to LLM. For example, if an attacker submits "Who has lights brighter than 80%?", LLM will reply with garbled code. Specifically, our test set contains normal and reversed prompts. Among them, accuracy refers to the proportion of test samples with correct responses to the total number of samples. The correct response in the normal prompt refers to getting accurate information about DTs from the responses; the reversal prompt refers to not getting sensitive information about DTs in the

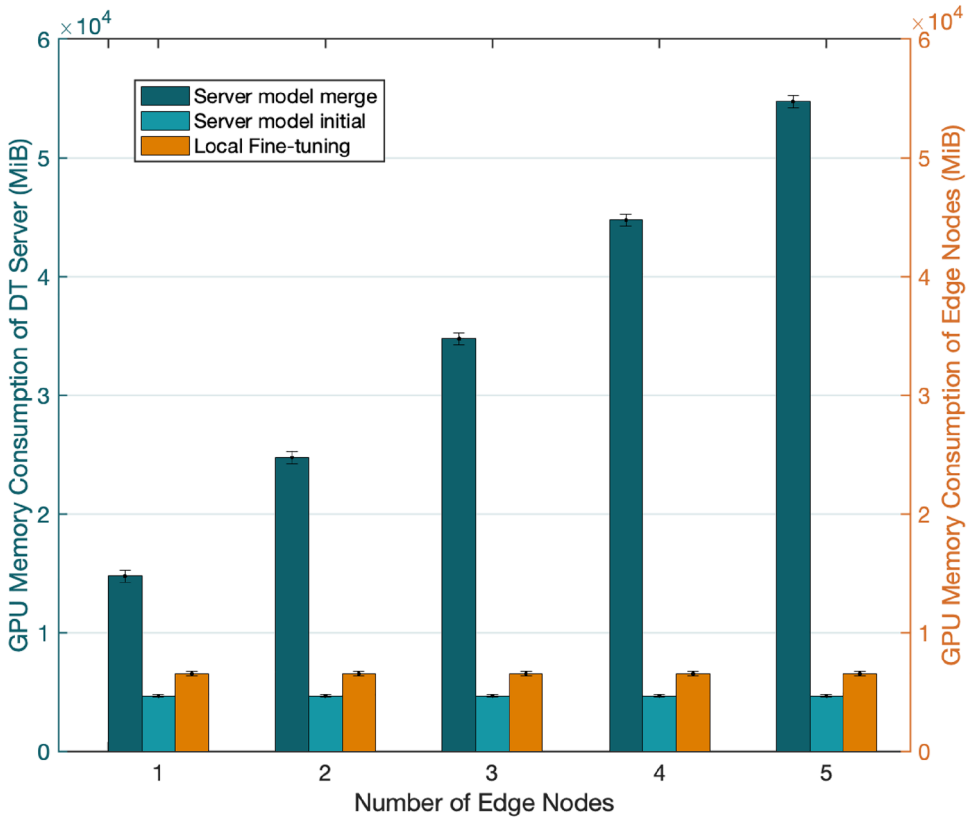

Figure 12. Server graphics memory consumption (left) or edge device graphics memory consumption (right) for different LLM-twin's tasks with different network sizes.

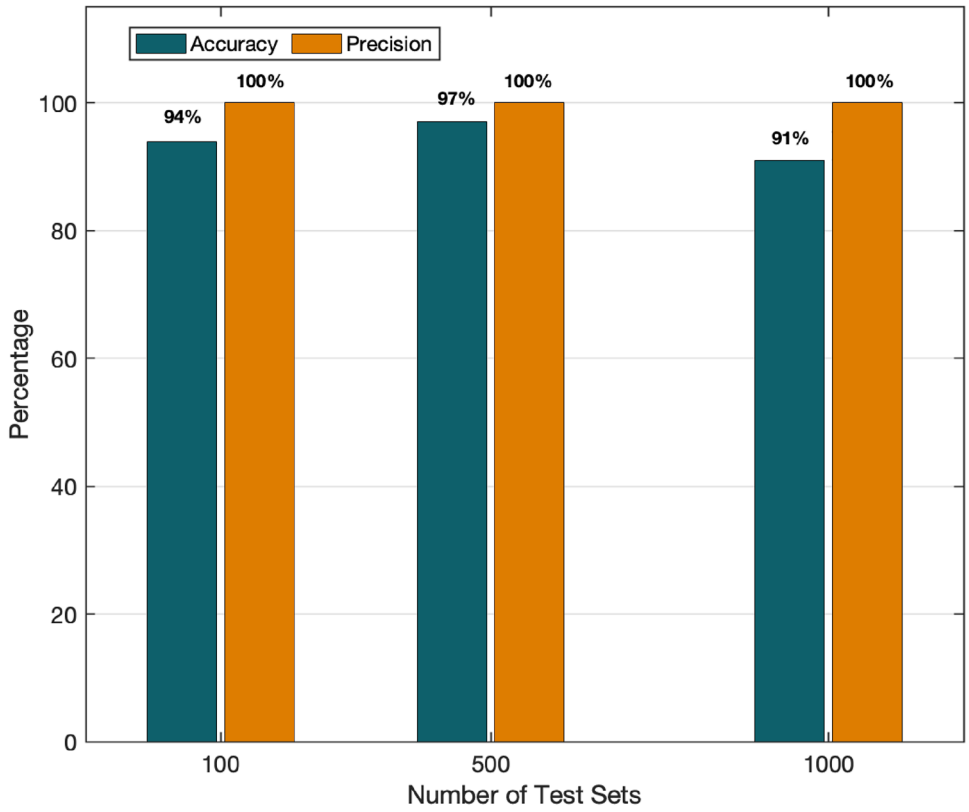

Figure 13. Accuracy and precision of LLM-twin in smart home case study.

Prompt type	Prompt number	Correct responses number	Incorrect responses number
Normal	100	94	6
Normal	500	485	15
Normal	1000	912	88
Reverse	500	0	500
Reverse	1000	0	1000

Table 5. Accuracy test of LLM-Twin's information generation.

responses. Table 5 and Fig. 13 shows the final results, where LLM is able to generate responses with more than 90% accuracy in tests with different sample sizes. In addition, the precision rate refers to the percentage of normal prompts among the samples that get correct DTs information responses. This is mainly to test the effectiveness of reversal curses in LLM-twin. Table 5 and Fig. 13 shows the final results, where the precision rate of LLM-twin reaches 100% in different sample sizes. This means that all the reversal cues that are designed to obtain the original sensitive information fail to get a response that contains the true information. This further validates the one-way security of LLM-Twin in this aspect.

Discussion

The proposed LLM-twin is a new networking framework for DTNs, and based on LLM we completely reshape the intra-twin and inter-twin communication mechanisms. Furthermore, we explored the effectiveness, feasibility, and scalability of LLM-twin through numerical validation and case studies. In addition, to support future implementations and research of LLM-twin in different environments, we will discuss the potential real-world challenges and limitations of implementing LLM-twin, which is expected to enhance a more comprehensive understanding of the applicability and usefulness of LLM-twin.

It is worth mentioning that LLM-twin is an enhancement to the DTNs framework rather than a scenario-specific improvement. Therefore, it is still applicable to DTNs in various domains, including intelligent traffic management, urban intelligent scheduling, smart home remote control, industrial control analytics, etc. We will discuss the limitations and challenges of LLM-twin implementations from four perspectives: edge devices, models, servers, and security, which will support the full potential of LLM-twin in various implementation environments. The first is the resource challenge of edge devices. LLM-twin requires physical entities to do

the fine-tuning locally, however, not all physical participating devices are capable of meeting the arithmetic requirements (although this proved to require only a small amount of arithmetic in the case study). Because the deployment of GPUs on all devices is not necessary, especially some edge sensors. This can be considered in the formal implementation phase by introducing edge servers or other collaborative strategies to give arithmetic support to low-resource physical devices.

The second is the impact of model performance on LLM-twin. As a communication model, it is important for LLM-twin to generate and propagate information accurately and efficiently. Compared with FL-based DTNs, LLM brings more efficient communication. In our case, LLM can achieve information reduction with more than 90% accuracy. However, with more complex LLM deployments, its hallucination and interpretability will bring new challenges. This will directly affect the accuracy and trustworthiness of semantic information reduction. Consequently, addressing the issues of hallucination and information trustworthiness in LLM represents a critical challenge for future research.

The third is the server resource scheduling challenge. in LLM-twin, we design the mini-giant architecture to offload part of the computation tasks to the edge devices, and the high overhead computations adopt the strategy of periodic independent execution, which significantly alleviates the huge arithmetic burden of deploying LLMs in DTNs. However, there is no such thing as a free lunch. We evaluate the theoretical limit overhead of the weight merging phase of LLM-twin in our case study, and when all nodes apply for weight merging at the same time (the extreme case), it will incur a significant server resource overhead. Therefore, server resource scheduling strategies need to be designed for specific environments during formal implementation. For example, limiting the maximum parallel requests based on priority, queuing, and reputation mechanisms. A reasonable server resource scheduling strategy can reduce the peak server resource utilization and balance the applicability of LLM-twin.

The last is the other security strategies of LLM-twin. In this work, we present a native security design for the LLM-twin framework and demonstrate its excellent security properties compared to traditional DTNs frameworks. For specific scenarios and implementation environments, it may be necessary to incorporate other security approaches to further improve the security of LLM-twin, which includes investigating security mechanisms such as secure multi-party computation, differential privacy, and so on to fulfill the security requirements of specific applications.

Conclusion

Beyond 5G networks bring great advantages for efficient and secure IoT, IoE, etc., especially DTNs that make it possible to efficiently process massive amounts of data and perform real-time optimization. However, traditional DTN frameworks have natural efficiency limitations, which mainly include the requirement of transmitting massive raw data collected from physical sensor devices in real-time as well as maintaining a high frequency of DT model computation. To address these challenges, this paper proposes LLM-Twin, a DTN networking framework based on LLM. First, by designing a mini-giant model collaboration framework, we address the resource-constrained problem faced by deploying LLM in DTN. Furthermore, we propose the DTSN, which designs novel intra-twin and inter-twin communication mechanisms for DTNs. Numerical experiments demonstrate that DTSN provides significant improvements in communication and computational efficiency compared to the traditional FL-based DTN. In addition, we consider the native security of the LLM-Twin architecture. We design a security model for LLM-Twin and provide proof of UC security. Further, we compare the potential security threats and countermeasures of FL-based DTNs in detail, demonstrating the security advantages of the proposed LLM-twin. In addition, to demonstrate the realistic usability and scalability, we designed a smart home case study based on the LLM-twin framework. The simulation results show that LLM-twin has acceptable performance in terms of communication delay, adaptability to different network sizes, memory bottleneck, and output accuracy.

Further, we discuss future research perspectives and potential challenges in implementing LLM-twin. It is worth mentioning that LLM-twin is an enhancement for the framework of DTNs, which does not affect its potential to be applied in specific scenarios. Therefore, a visible future research direction is to apply LLM-twin to various areas of traditional DTNs, including intelligent traffic management, urban intelligent scheduling, smart home remote control and industrial control analytics, and so on. In addition, some realistic challenges including arithmetic scheduling of edge devices and servers, LLM hallucination problems, additional privacy and security policies, etc. will also be important directions for future research.

Data availability

The datasets used and/or analyzed during the current study are available from the corresponding author upon reasonable request. Regarding the smart home physical side dataset simulated in the case study, we open-source it by uploading it to Github: https://github.com/CURRYSGITHUB/LLM-twin/tree/main.

Received: 28 November 2023; Accepted: 5 August 2024
Published online: 17 August 2024

References

1. Wolf, K., Dawson, R. J., Mills, J. P., Blythe, P. & Morley, J. Towards a digital twin for supporting multi-agency incident management in a smart city. *Sci. Rep.* **12**, 16221 (2022).
2. Yigit, Y. *et al.* Twinport: 5G drone-assisted data collection with digital twin for smart seaports. *Sci. Rep.* **13**, 12310 (2023).
3. Lin, X. *et al.* 6g digital twin networks: From theory to practice. *IEEE Commun. Magaz.*https://doi.org/10.1109/MCOM.001.2200830 *(2023).*
4. Luan, T. H., Liu, R., Gao, L., Li, R. & Zhou, H. The paradigm of digital twin communications. arXiv:2105.07182 (2021).
5. Hong, Y. & Wu, J. Fuzzing digital twin with graphical visualization of electronic AVS provable test for consumer safety. *IEEE Trans. Consumer Electron.*https://doi.org/10.1109/TCE.2023.3269528 *(2023).*

6. Marai, O. E., Taleb, T. & Song, J. Roads infrastructure digital twin: A step toward smarter cities realization. *IEEE Netw.* **35**, 136–143. https://doi.org/10.1109/MNET.011.2000398 (2021).
7. Tao, F., Zhang, H., Liu, A. & Nee, A. Y. C. Digital twin in industry: State-of-the-art. *IEEE Trans. Ind. Inf.* **15**, 2405–2415. https://doi.org/10.1109/TII.2018.2873186 (2019).
8. Wu, Y., Zhang, K. & Zhang, Y. Digital twin networks: A survey. *IEEE Internet Things J.* **8**, 13789–13804. https://doi.org/10.1109/JIOT.2021.3079510 (2021).
9. Alcaraz, C. & Lopez, J. Digital twin: A comprehensive survey of security threats. *IEEE Commun. Surv. Tutor.* **24**, 1475–1503. https://doi.org/10.1109/COMST.2022.3171465 (2022).
10. Yeon, H., Eom, T., Jang, K. & Yeo, J. Dtumos, digital twin for large-scale urban mobility operating system. *Sci. Rep.* **13**, 5154 (2023).
11. Wang, Y. *et al.* A survey on digital twins: Architecture, enabling technologies, security and privacy, and future prospects. *IEEE Internet Things J.* **10**, 14965–14987. https://doi.org/10.1109/JIOT.2023.3263909 (2023).
12. Guo, Q., Tang, F., Rodrigues, T. K. & Kato, N. Five disruptive technologies in 6g to support digital twin networks. *IEEE Wirel. Commun.*https://doi.org/10.1109/MWC.013.2200296 *(2023).*
13. He, C., Luan, T. H., Lu, R., Su, Z. & Dong, M. Security and privacy in vehicular digital twin networks: Challenges and solutions. *IEEE Wirel. Commun.* **30**, 154–160. https://doi.org/10.1109/MWC.002.2200015 (2023).
14. Khan, L. U. *et al.* Digital twin of wireless systems: Overview, taxonomy, challenges, and opportunities. *IEEE Commun. Surv. Tutor.* **24**, 2230–2254. https://doi.org/10.1109/COMST.2022.3198273 (2022).
15. Lu, Y., Huang, X., Zhang, K., Maharjan, S. & Zhang, Y. Communication-efficient federated learning and permissioned blockchain for digital twin edge networks. *IEEE Internet Things J.* **8**, 2276–2288. https://doi.org/10.1109/JIOT.2020.3015772 (2021).
16. Li, Z., Hong, Y., Bashir, A. K., Al-Otaibi, Y. D. & Wu, J. Software-defined GPU-CPU empowered efficient wireless federated learning with embedding communication coding for beyond 5g. *IEEE Open J. Commun. Soc.* **4**, 990–1000. https://doi.org/10.1109/OJCOMS.2023.3266444 (2023).
17. Lu, Y., Huang, X., Zhang, K., Maharjan, S. & Zhang, Y. Communication-efficient federated learning for digital twin edge networks in industrial IoT. *IEEE Trans. Industr. Inf.* **17**, 5709–5718. https://doi.org/10.1109/TII.2020.3010798 (2021).
18. Lu, Y., Huang, X., Zhang, K., Maharjan, S. & Zhang, Y. Low-latency federated learning and blockchain for edge association in digital twin empowered 6g networks. *IEEE Trans. Industr. Inf.* **17**, 5098–5107. https://doi.org/10.1109/TII.2020.3017668 (2021).
19. Liu, Q., Tang, L., Wu, T. & Chen, Q. Deep reinforcement learning for resource demand prediction and virtual function network migration in digital twin network. *IEEE Internet Things J.* **10**, 19102–19116. https://doi.org/10.1109/JIOT.2023.3281678 (2023).
20. Zhang, H., Ma, X., Liu, X., Li, L. & Sun, K. Gnn-based power allocation and user association in digital twin network for the terahertz band. *IEEE J. Select. Areas Commun.*https://doi.org/10.1109/JSAC.2023.3313192 *(2023).*
21. Jiang, L., Zheng, H., Tian, H., Xie, S. & Zhang, Y. Cooperative federated learning and model update verification in blockchain-empowered digital twin edge networks. *IEEE Internet Things J.* **9**, 11154–11167. https://doi.org/10.1109/JIOT.2021.3126207 (2022).
22. Aloqaily, M., Ridhawi, I. A. & Kanhere, S. Reinforcing industry 4.0 with digital twins and blockchain-assisted federated learning. *IEEE J. Select. Areas Commun.*https://doi.org/10.1109/JSAC.2023.3310068 *(2023).*
23. Luo, X., Chen, H.-H. & Guo, Q. Semantic communications: Overview, open issues, and future research directions. *IEEE Wirel. Commun.* **29**, 210–219 (2022).
24. Raha, A. D., Munir, M. S., Adhikary, A., Qiao, Y. & Hong, C. S. Generative ai-driven semantic communication framework for nextg wireless network (2023). arXiv:2310.09021.
25. Zhong, L. & Wang, Z. Can chatgpt replace stackoverflow? A study on robustness and reliability of large language model code generation (2023). arXiv: 2308.10335.
26. Lai, Z., Zhu, X., Dai, J., Qiao, Y. & Wang, W. Mini-dalle3: Interactive text to image by prompting large language models (2023). arXiv: 2310.07653.
27. Rubenstein, P. K. *et al.* Audiopalm: A large language model that can speak and listen (2023). arXiv: 2306.12925.
28. Hong, S., Seo, J., Hong, S., Shin, H. & Kim, S. Large language models are frame-level directors for zero-shot text-to-video generation (2023). arXiv: 2305.14330.
29. Sun, W., Lian, S., Zhang, H. & Zhang, Y. Lightweight digital twin and federated learning with distributed incentive in air-ground 6G networks. *IEEE Trans. Netw. Sci. Eng.* **10**, 1214–1227 (2022).
30. Campolo, C. *et al.* An edge-based digital twin framework for connected and autonomous vehicles: Design and evaluation. IEEE Access (2024).
31. Thomas, C. K., Saad, W. & Xiao, Y. Causal semantic communication for digital twins: A generalizable imitation learning approach. IEEE J. Select. Areas Inf. Theory (2023).
32. Xu, J., He, C. & Luan, T. H. Efficient authentication for vehicular digital twin communications. In *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, 1–5 (IEEE, 2021).
33. Li, G., Lai, C., Lu, R. & Zheng, D. Seccdv: A security reference architecture for cybertwin-driven 6g v2x. *IEEE Trans. Veh. Technol.* **71**, 4535–4550 (2021).
34. Dai, M. *et al.* Digital twin envisioned secure air-ground integrated networks: A blockchain-based approach. *IEEE Internet Things Magaz.* **5**, 96–103 (2022).
35. Feng, H., Chen, D. & Lv, H. Sensible and secure IoT communication for digital twins, cyber twins, web twins. *Internet Things Cyber-Phys. Syst.* **1**, 34–44 (2021).
36. Yigit, Y., Bal, B., Karameseoglu, A., Duong, T. Q. & Canberk, B. Digital twin-enabled intelligent ddos detection mechanism for autonomous core networks. *IEEE Commun. Stand. Magaz.* **6**, 38–44 (2022).
37. Yigit, Y., Chrysoulas, C., Yurdakul, G., Maglaras, L. & Canberk, B. Digital twin-empowered smart attack detection system for 6g edge of things networks. arXiv:2310.03554 (2023).
38. Zhou, Z., Li, L., Chen, X. & Li, A. Mini-giants: "small" language models and open source win-win (2023). arXiv: 2307.08189.
39. Hu, E. J. *et al.* Lora: Low-rank adaptation of large language models (2021). arXiv: 2106.09685.
40. Dettmers, T., Pagnoni, A., Holtzman, A. & Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms (2023). arXiv:2305.14314.
41. Berglund, L. *et al.* The reversal curse: Llms trained on "a is b" fail to learn "b is a" (2023). arXiv: 2309.12288.
42. Canetti, R. Universally composable security: a new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, 136–145, https://doi.org/10.1109/SFCS.2001.959888 (2001).

Acknowledgements

This work was supported in part by the JSPS KAKENHI under Grants 23K11072, in part by the National Natural Science Foundation of China under Grants U21B2019 and 61972255, and in part by the China Scholarship Council under Grants 202308050055.

Author contributions

Y.H. and J.W. conceived the idea and proposed the methodology, Y.H. verified the method, Y.H., J.W., and R.M. analyzed the results. All authors reviewed the manuscript.

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to J.W.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by-nc-nd/4.0/.

© The Author(s) 2024