ISSN 2278-3091
Volume 9, No.4, July – August 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse28942020.pdf
https://doi.org/10.30534/ijatcse/2020/28942020

Deep Embedding Data Fusion Scheme Using Variational Graph Auto-Encoder in IoT Environments

Asmaa Mohamed fathy[1], Ahmed. A. A. Gad-Elrab[2.3], Sawsan Mohammed Aziz[4], Heba F. Eid[5]
[1]Faculty of Science, Al- Azhar University, Cairo, Egypt, asmaamohamed89@azhar.edu.eg
[2]Faculty of Computers and Information Technology, King Abdul-Aziz University, Saudi Arabia, aaahmad4@kau.edu.sa
[3]Faculty of Science, Al-Azhar University, Cairo, Egypt, asaadgad@azhar.edu.eg
[4]Faculty of Science, Al- Azhar University, Cairo, Egypt, Sawsan_Aziz_Ali@hotmail.com
[5]Faculty of Science, Al- Azhar University, Cairo, Egypt, heba.fathy@azhar.edu.eg

ABSTRACT

Internet of things (IoT) is a promised paradigm for developing smart systems and architectures. IoT is a framework where everyday objects can be equipped with capabilities for identifying, sensing, networking and processing that will allow them to interact over the Internet with each other and with other devices and services to achieve some objectives. In IoT, the sensing devices can collect and manage data that exist in their surrounding environments. So, data fusion problem is a major challenge for the future in order to allow highly effective, reliable and accurate management and decision-making of IoT environments. To meet this challenge, this paper introduces a new data fusion scheme uses a variational graph auto-encoder as a deep learning method for creating deep embedding data graph. This graph represents the similarities among data items which can be divided into groups that can be fused together to minimize the amount of transferred data and latency time of the collected data to the cloud server and the energy consumption by IoT devices. The conducted simulations experiments and results show that the proposed scheme can achieve a reasonable performance in terms of latency time, energy consumption, and data fusion accuracy compared to non-fusion schemes.

Key words: Internet of Things (IoT), Data Fusion, Variational Graph Auto-Encoder, Embedding graph.

1. INTRODUCTION

Recently, internet of things (IoT) have gained considerable attention and played an important role in building and deploying of smart environment applications and platforms. The Internet of Things is a framework where everyday objects can be equipped with capabilities for identifying, sensing, networking and processing that will allow them to interact over the Internet with each other and with other devices and services to achieve some objective. Eventually, IoT devices will be omnipresent, context-aware, and allow intelligence in the environment [1]. IoT is set to become one of the key technological developments of our times provided we are able to realize its full potential [2]. Kevin Ashton first coined the concept of "Internet of Things" (IoT), where smart objects are connected with the Internet.

Nowadays, IoT play vital role in many fields such as transport, agriculture, industry, and healthcare [3]. IoT is "a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving inter-operable information and communication technologies [4]. The most prominent application areas include, for example, the developing of connected production sites and smart production systems in smart industry is frequently discussed under Industry 4.0. Smart security systems and thermostats receive more attention in the smart home or building field. On the other hand, smart energy solutions concentrate on smart meters of gas, water, and electricity. Smart transport solutions include, for example, monitoring mobile tickets and vehicle fleets. In the smart health sector, themes such as the patients care and the chronic diseases treatment are discussed and smart city technologies are being discussed, approaches such as real time parking capacity tracking and smart street lighting [4].

As a result, managing data raises many challenging problems as data representation, data fusion, task allocation, data heterogeneity, sensor/actuator management, data accuracy, reliability and other problems. An analysis and timely fusion of big data that collected from IoT sources would be a major challenge for the future in order to allow highly effective, reliable and accurate decision-making and management of pervasive environments. Data fusion is defined as "the theory, techniques and tools which are used for combining sensor data, or data derived from sensory data, into a common representational format" [5]. The data fusion architecture is made up of three major modules: *preprocessing*, *modeling, and fusion*. Preprocessing is devoted to independent transformation of specific data streams; modeling gives predictions from enhanced incremental learning or batch models, while fusion combines them into full feature vectors.

In IoT environments, there are many problems as context modeling [6], energy consumption [7], and data fusion. The challenging task is data fusion for some reasons [8]. In data fusion, firstly, the data is created by very complicated systems: economic, biological, psychological, and sociological that guided by various hidden processes that rely on a huge variables set that there is no access to them. Secondly, the number, variety and nature of new questions in research that can be raised is very broad due to the increased diversity. Second, it is not an easy job to work with heterogeneous data sets to optimize the respective advantages of each data set and to suppress disadvantages. So, to obtain meaningful information from heterogeneous IoT data, data fusion techniques are used. It integrates individual sensor data to collectively produce a result that is more reliable, accurate and complete. Through IoT, integration of environmental knowledge can be used in different areas to improve the omnipresent nature of IoT. Such fields include surveillance of the environment, health care, crisis management, monitoring, managing, tracking, gathers of information, and much more. IoT data fusion can occur at four stages: level of decision, level of feature, pixel level and level of signal.

Data Fusion is facing multiple challenges, which are (a) *Data Imperfection*: at times, the data of the sensor is imprecise; it may be unreliable and uncertain. This activity in wireless sensor networks is not notorious. Imperfection with the use of data fusion algorithms must be handled effectively. (b) I*nconsistencies and ambiguities*: impreciseness are the factors responsible for data inconsistencies and the environment in which a sensor operate [9]. In the IoT world, outer identification, replacement and imputation of data are critical. (c) *Conflicting nature*: conflicting data can result in results of counter-intuitive. The issue of data conflicting is more evident in the reasoning of proof belief and the combination law of Dempster. Due care must be taken in the data fusion algorithm when treating contradictory data [10]. (d) *Data alignment and correlation*: it is more prevalent in wireless sensor networks (WSNs) and may result in a data fusion algorithm being over-or under-confidence. As sensor data is converted from the local frame of each sensor to a specific frame prior to fusion, the problem of alignment is also known as a problem of sensor registration occurs. (e) *Trivial Features*: applications may consist of several hundred thousand sensors with different parameters in the IoT environment. Such sensed values consist of trivial and non-trivial data in big settings such as industrial plants and smart cities. Slight data processing can affect the accuracy of data fusion. The most important features therefore need to be extracted before data fusion [2]. (f) *Dynamic Iterative Process*: the data fusion is not a static process; however, dynamic iterative estimates need to be refined periodically in a fusion environment. No Amazing Algorithm: There are developments in time research in the area of data fusion and there are now high-performance algorithms. However, there is a flawless algorithm for data fusion.

The main contributions of this paper are:
1. Proposing a new deep embedding scheme for data fusion in IoT environments.
2. Using a variational graph auto-encoder deep learning method for creating embedding data graph.
3. Improving the data fusion process in terms of latency time, energy consumption, and data fusion accuracy for IoT environments.

The remainder of the paper is organized as follows: Section 2 reviews related works that are discussed data fusion in IoT. Section 3 describes and formulates the data fusion problem in IoT. Section 4 introduces the proposed data fusion scheme by using deep embedding and variational graph auto-encoder for IoT Environments. Section 5 explains the simulation and evaluation results. Finally, conclusion is introduced in Section 6.

2. RELATED WORK

For several years great effort has been devoted to the study of data fusion problem in IoT [1]-[20]. Jara et.al [11] In order to understand human behavior in smart cities, used sensor data fusion. Their research analyzes data from the Smart Santander European project. The work shows how omnipresent data such as traffic flows and temperature can be combined to understand and model temperature impact on traffic flow. The work considers the Poisson model and shows that the Poisson distribution model is not always valid. In order to estimate the attractiveness of smart cities for visitors, Sobolevsky et al. [12] used sensor data fusion. The work focuses primarily on Spain's communities. To attract visitors in Spain, 700,000 geotagged tweets fuse sensor data of three data sources and 3.5 million videos and photos posted to Flickr, namely credit and debit card transactions performed by visitors. A city's attractiveness for the city's intent was described as the whole number of pictures, card transactions and tweets in it. The research provides some valuable outcomes and demonstrates how fusion of sensor data sets can provide perceptions into cities that are used by people. In general, a significant number of visitors are drawn by the research found by larger cities.

There have also been some examples. For example, in Malaga city, there was a high level of tourists but its Flickr usage is very low. This is due to the fact that this city is considered retirement destinations and the group of visitors use fewer social media like Flickr. This research is an valuable data fusion scheme in smart cities that can assist to create revolutionary services for returning value to their residents and investors in smart cities. Antonelli et al. [13] introduced city sensor fusion, a large data network that gathers, aggregates, analyzes, and offers visual analytic from data streams in smart cities. The work concentrates on using of sensor data fusion for identifying city-scale activities such as day-long events, number of attracted visitors, places that attracted significant interest. The app fuses data from various forms of data sources from sensors to mobile phones to social media like traffic flow, weather and emissions.

Soldatos et al., [14] are proposing OpenIoT a first-of - a-kind open source IoT framework that allows IoT services in the cloud to be inter operable semantically. OpenIoT pro-motes IoT silos interoperability from the system to the cloud services. OpenIoT is built on semantic web standards such as the ontology of W3C Semantic Sensor Networks (SSN), which provides a shared standard-based model to represent physical and virtual sensors, RDF to store, index and retrieve data, and supports virtually any IoT protocol such as CoAP, 6LoWPAN, etc. OpenIoT also provides middle ware application and aggregation capabilities of sensor data at the stuff and cloud. OpenIoT allows the collection of data from practically any sensor while ensuring that they are equipped with enough semantine annotation at the same time. It also offers a wide range of Do - it-yourself visual tools that allow IoT services and applications with nearly zero programming to be created and deployed. Another key feature of OpenIoT is its support for mobile sensors, allowing support for a growing wave of applications for mobile crowd sensing.

Recent work by a group of MIT researchers [12] has shown the potential for integrating data from different sources of data in smart cities to recognize the attractiveness of the city. The work concentrates on cities in Spain and demonstrates how sets of big data can merge into a way people navigate cities. For example, a data convergence from different sources of data plays a critical role in the successful delivery of potential infrastructure in smart cities. Liu and Zhao [15] also recognized that most of the attempts on sensing systems today are domain specific with very little re usability. To solve this problem, they also suggested a flexible interface that is filled with semantics. XML data formats are used in the framework to store data. Service modules as the main building block in the system. Each service is designed to take inputs, do some processing and send the output back. Systems are structured to combine multiple services to create a dynamic service. Because of procedural representations, this method of integrating run time is feasible. This programming model enables the user to abstractly query sensor data and events without handling raw sensor data.

Izumi et al. [16] suggested an information filter system to benefit the area of health care. We have a number of different agents in their program, such as a data stream mining agent, an inference agent, and a knowledge base agent. For the design of the framework, a multi-agent architecture is used, and an ontology scheme is used to store data while SPARQL queries are used to process data. Knowledge gathered using sensors is filtered based on four different perspectives: filtering based on individuals, filtering based on access policies, filtering based on location, and filtering based on time. Teymourian et al. [17] provide a systematic approach to addressing the problem of sequential event processing (SCEP). SCEP is a mix of systems for event management and semantic techniques. This research effort is not directly related to the fusion of sensor data. Nevertheless, to track activities in the IoT world, the methods used in this field can be paired with the analysis of sensor data streams. The system [18] of the Sensor Web Agent Platform (SWAP) consists of three layers: sensor base, information layer and device layer. For the design of the device, a multi-agent platform and web services software are used. That layer is made up of a number of agents capable of performing specific tasks. Implementation relies on a context of fire detection. It is possible to combine the number of different agents to respond or detect complex situations such as wildfire.

3. DATA FUSION PROBLEM IN IoT (DFP)

In IoT, data fusion of multi-sensor attempts to integrate data and information from multiple sources (including sensors, human reports and internet data) to create interfaces that cannot be accessed from a single sensor or source or whose accuracy exceeds that of a single source interface [15]. The data fusion problem in IoT is how to find an optimal data fusion method that can integrates collected data from multiple data sources to minimize the latency time and transferring cost and improving the data accuracy in IoT environments. This problem is called Data Fusion Problem (DFP). Here, the assumptions, IoT and data models will be introduced, and then DFP will be formulated

3.1 Assumptions and IoT Model

The proposed IoT model consists of a set of sensors, which is denoted as $S = \{s_1, s_2, , s_j, \dots, s_N\}$. Each sensor $s_j \in S$ may generate a continuous data, periodically with a specified time interval *T* (we assume that this time interval is fixed foe all sensors to generate their sensing data). The set of generated data items at different time slots by each sensor $s_j \in S$ is denoted as $D_j = \{d^j_{t1}, d^j_{t2}, \dots, d^j_{ti}, \dots\}$, whereas $|t_{i+1} - t_i| = T_i$. Here, T_i is equal for all *i*. Assume that there is a time windows called *TW*, which is, consists of a set of a limited number of time slots, where $TW = \{t_1, t_2, \dots, t_c\}$. These generated data sets by all sensors represent different dataspaces (i.e., each sensor represents certain data-space). A set of fusion nodes which is denoted as $FN = \{fn_1, fn_2, \dots, fn_k, \dots, fn_K\}$. A cloud server *CS* that can receive all collected sensed data for sensors or from fusion nodes. Assume that each fusion node $fn_k \in FN$ received a set of data items from different sensors which is denoted as RD_i, where the number of items in RD_i is M_i. Assume that each fusion node $fn_k \in FN$ can fuse its received set of data items into a new integrated set of data items which is denoted as FRD_k such that the number of items in $FRD_k = \{fd^k_1, fd^k_2, \dots, \dots\}$ is less than or equal M_i (the number of data items in RD_i).

Assume that the time delay to send one data item $d^j_{ti} \in D_j$ from sensor $s_j \in S$ to cloud server *CS* is denoted as $td(d^j_{ti}, s_j, CS)$. Assume that the consumed energy to send one data item $d^j_{ti} \in D_j$ from sensor $s_j \in S$ to cloud server *CS* is denoted as $ec(d^j_{ti}, s_j, CS)$. Assume that the time delay to send one data item $fd^k_i \in FRD_k$ from fusion node $fn_k \in FN$ to cloud server *CS* is denoted as $tdf(fd^k_i, fn_k, CS)$. Assume that the consumed energy to send one data item $fd^k_i \in FRD_k$ from fusion node $fn_k \in FN$ to cloud server *CS* is denoted as $ecf(fd^k_i, fn_k, CS)$.

For each sensor node $s_j \in S$, the total time delay cost for sending its data items D_j to *CS* is defined as follows.

$$STD_{sen}(s_j, D_j, CS) = \sum_{t_i \in TW} td(d^j_{ti}, s_j, CS) \qquad (1)$$

then the total time delay cost by all sensors is defined as follows.

$$TTD_{sen}(S, D, CS) = \sum_{j=1}^{N} STD_{sen}(s_j, D_j, CS) \qquad (2)$$

where $D = \{D_1, D_2, \dots, D_N\}$.

While for each fusion node $fn_k \in FN$, the total time delay cost for sending its data items FRD_k to *CS* is defined as follows.

$$STD_{fus}(fn_k, FRD_k, CS) = \sum_{l=1}^{|FRD_k|} tdf(fd_l^k, fn_k, CS) \quad (3)$$

then the total time delay cost by all fusion nodes is defined as follows.

$$TTD_{fus}(FN, DF, CS) = \sum_{k=1}^{K} STD_{fus}(fn_k, FRD_k, CS) \quad (4)$$

where $DF = \{FRD_1, FRD_2, \dots, FRD_K\}$.

For each sensor node $s_j \in S$, the total consumed energy cost for sending its data items D_j to *CS* is defined as follows.

$$SEC_{sen}(s_j, D_j, CS) = \sum_{t_i \in TW} ec(d_{ti}^j, s_j, CS) \quad (5)$$

then the total consumed energy cost by all sensors is defined as follows.

$$TEC_{sen}(S, D, CS) = \sum_{j=1}^{N} SEC_{sen}(s_j, D_j, CS) \quad (6)$$

While for each fusion node $fn_k \in FN$, the total time delay cost for sending its data items FRD_k to *CS* is defined as follows.

$$SEC_{fus}(fn_k, FRD_k, CS) = \sum_{l=1}^{|FRD_k|} ecf(fd_l^k, fn_k, CS) \quad (7)$$

then the total consumed energy cost by all fusion nodes is defined as follows.

$$TEC_{fus}(FN, DF, CS) = \sum_{k=1}^{K} SEC_{fus}(fn_k, FRD_k, CS) \quad (8)$$

Now, the utility fusion function is defined as the summation of total time delay and total consumed energy costs by fusion nodes and is defined as follows.

$$UFN_{fus}(FN, DF, CS) = w_1 \cdot SEC_{fus}(fn_k, FRD_k, CS) + w_2 \cdot TEC_{fus}(FN, DF, CS) \quad (9)$$

where w_1 and w_2 are the weights values for time delay and energy consumption costs, respectively, such that $w_1 + w_2 = 1$.

3.2 Problem Formulation

The goal of data fusion in IoT, is finding a data fusion mechanism, *FMeck*, which can map all data spaces in $D = \{D_1, D_2, \dots, D_N\}$ of all sensors in *S* to a common data space *CD* by finding the optimal set of fused data items $DF = \{FRD_1, FRD_2, \dots, FRD_K\}$. This fusion mechanism, $FMeck(D)$, can be expressed as follows.

$$FMeck(D) \rightarrow CD(DF) \quad (10)$$

So, to satisfy this goal and based on the IoT model and assumptions, the DFP problem can be formulated as follows:

$$\textbf{Minimize } UFN_{fus}(FN, DF, CS) \quad \textbf{(11)}$$

such that

$$TTD_{fus}(FN, DF, CS) \leq TTD_{sen}(S, D, CS) \quad (12)$$

$$TEC_{fus}(FN, DF, CS) \leq TEC_{sen}(S, D, CS) \quad (13)$$

Constraint (12) means that the total time delay cost by fusion node must be less than or equal to the total time delay cost by sensor nodes for the cloud server. Constraint (13) means that the total consumed energy cost by fusion node must be less than or equal to the total consumed energy cost by sensor nodes for the cloud server.

4 A DEEP EMBEDDING DATA FUSION SCHEME USING VARIATIONAL GRAPH AUTO-ENCODER

For solving the DFP problem, a new data fusion scheme is proposed called *Deep Embedding Data Fusion Scheme Using Variational Graph Auto-Encoder*, (DEDF-VGAE).

4.1 Basic Idea

DEDF-VGAE scheme is proposed to solve DFP problem for minimizing time delay and energy consumption costs for sending and collecting data in IoT environments. In addition, to keep the data accuracy as much as possible. The basic idea of DEDF-VGAE is based on: (1) Building a similarity graph of all data items in all data spaces in *D* by using *Unified Relationship Matrix* (URM) for representing a set of heterogeneous data items and their relationships based on a certain threshold called $\propto$. The resulted graph is called $\propto -URMGraph$. (2) Determining the adjacency and feature matrices of all data nodes in the $\propto -URMGraph$. These adjacency and feature matrices are denoted as ADJ_G and FEA_G, respectively. (3) Using a deep learning variational graph auto-encoder (VGAE) [21] to map the determined adjacent matrix, ADJ_G of $\propto -URMGraph$ into new adjacency matrix, $NADJ_G$ (as a new common data space) by using ADJ_G and FEA_G as inputs into VGAE. (4) Fusing the resulted adjacency data items matrix, $NADJ_G$, into a new fused data sets *DF* in fused nodes *FN*.

4.2 The Proposed Scheme

The proposed DEDF-VGAE scheme consists of six phases as follows.

A. Collection phase

In this phase, DEDF-VGAE scheme collects sensing data from all sensors every time interval T. Then updates each data set D_j of a sensor S_j by adding this new value in it;

B. URM-Determination phase

In this phase, the proposed scheme determines the unified relation matrix, *URM* by using certain similarity measures (e.g., Euclidian distance similarity or cosine similarity). URM represents both intra- and inter-type relationships between heterogeneous data items in a unified manner. Assume there are *N* sensors $s_1, s_2, , s_j, \dots, s_N$ represent different data spaces $D_1, D_2, \dots, D_N$. Data items within the same data space D_j are connected via intra-type relationships $RA_j \subseteq D_j \times D_j$. Data items from two different data spaces D_j and D_h are connected via inter-type relationships $RE_h \subseteq D_j \times D_h$ $(j \neq h)$. The relationships of intra-type RA_j can be described as a $K \times K$ matrix AL_j (*K* is the whole number of items in data space - D_j). In matrix AL_j, the cell cl_{xy} denotes the relationship of intra-type from x_{th} item to y_{th} item in D_j (data space). The relationship of inter-type RE_{jh} can be described as a $K \times P$ matrix AL_{jh} (*K* is the whole number of items in data space - D_h), where the value of cell cl_{xy} represents the inter-type relationship from the x_{th} item in D_j to the y_{th} item in D_h. If we merge *K* data spaces into a unified data space *US*, then previous intra- and inter-type relationships are all part of intra-type relationships R_z in data space *US*. Suppose AL_z is the matrix of R_z, then AL_z is a square matrix. The unified relationship Matrix *URM* as a matrix that combines all the relationship matrices, as follows:

$$URM = \begin{bmatrix} AL_1 & AL_{12} & AL_{13} & \cdots & \cdots & AL_{1K} \\ AL_{21} & AL_2 & AL_{23} & \cdots & \cdots & AL_{2K} \\ AL_{31} & AL_{32} & AL_3 & \cdots & \cdots & AL_{3K} \\ \vdots & \vdots & \vdots & \ddots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ AL_{K1} & AL_{K2} & AL_{K3} & \cdots & \cdots & AL_K \end{bmatrix} \tag{14}$$

Here, the value each cell in *URM* is calculated by using Euclidian distance similarity measure. In addition, to take care of the cases in which the distance values in one matrix AL_j or AL_{jh} are significantly larger than those in the other matrix, and *URM* is normalized to have the same scale within interval [0, 1]. The normalized matrix is called *NURM* and is defined as follows.

$$NURM = \begin{bmatrix} NL_1 & NL_{12} & NL_{13} & \cdots & \cdots & NL_{1K} \\ NL_{21} & NL_2 & NL_{23} & \cdots & \cdots & NL_{2K} \\ NL_{31} & NL_{32} & NL_3 & \cdots & \cdots & NL_{3K} \\ \vdots & \vdots & \vdots & \ddots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ NL_{K1} & NL_{K2} & NL_{K3} & \cdots & \cdots & NL_K \end{bmatrix} \tag{15}$$

where each element in NL_j and NL_{jh} is defined as follows.

$$ncl_{xy} = \frac{cl_{xy} - minVal(URM)}{maxVal(URM) - minVal(URM)} \quad \forall ncl_{xy} \in NL_j, NL_{jh} \quad , \quad nl_{xy} \in AL_j, AL_{jh} \tag{16}$$

where $minVal(UR)$ and $maxVal(URM)$ represent the minimum and maximum distance values among all values in URM, respectively.

C. ∝-SimGraph-Creation phase

In this phase, DEDF-VGAE scheme creates a similarity graph called $\propto -URMGraph$ by using the normalized unified relationships matric $NURM$ and a specified real value called ∝. Here, firstly, a new edge concept called $\propto -\boldsymbol{Edge}$ will be defined then $\propto -URMGraph$ will be described.

Definition 1: $\propto -\boldsymbol{Edge}$: Given a two data items u and v, a real value ∝, and a normalized similarity value between u and v (i.e., $ncl_{uv} \in NURM$. There is a link between u and v if and only if the value of $ncl_{uv} \geq \propto$ and this link is called $\propto -Edge$ between u and v.

Definition 2: $\propto -\boldsymbol{SimGraph}$: Given a set of data items V, a real value ∝, a normalized similarity matrix $NURM$ of data items in V. The $\propto -\boldsymbol{SimGraph}$ is a graph $G = (V, E)$ such that each edge between any pairs of data items $u, v \in V$ is an $\propto -\boldsymbol{Edge}$. Figure 1 shows an example for constructing $\propto -\boldsymbol{SimGraph}$ with two values of ∝ which are 0.2 (Figure 1 (a)) and 0.4 (Figure 1(c)), where $V = \{v_1, v_2, v_3, v_4, v_5\}$ and its $NURM$ matrix in (Figure 1(b)).

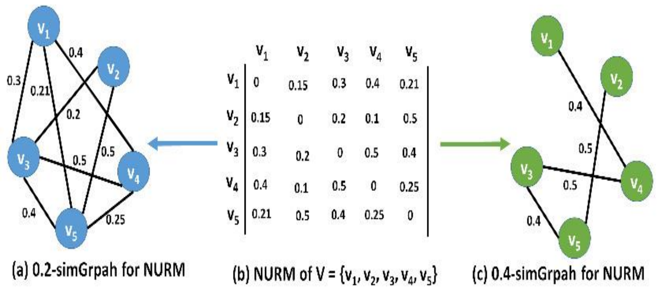

Figure 1: An example for constructing $\propto -\boldsymbol{SimGraph}$ with two values of ∝ which are 0.2 and 0.4

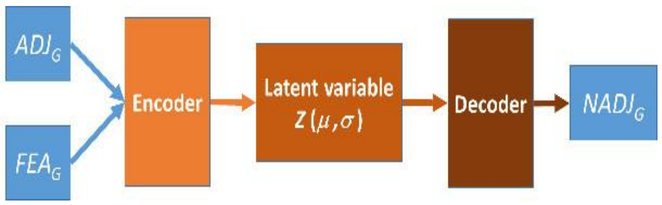

Figure 2: the architecture of VGAE with its inputs (ADJ_G and FEA_G) and outputs ($NADJ_G$)}

D. Deep-Embedding phase

In this phase, DEDF-VGAE scheme uses the created $\propto -\boldsymbol{SimGraph}$ to create the adjacency matrix of all data items in V which is denoted as ADJ_G . In addition, DEDF-VGAE scheme uses a normalized unified relationship matric $NURM$ as a feature matrix which is denoted as FEA_G. Finally, DEDF-VGAE scheme uses ADJ_G and FEA_G matrices as two inputs for variational graph auto-encoder (VGAE) to generate a new adjacency matrix called $NADJ_G$. Here, $NADJ_G$ matrix represent a new common data space for all input data spaces. Figure 2 shows the architecture of VGAE with its inputs and outputs.

E. Grouping phase

In this phase, DEDF-VGAE scheme uses the resulted adjacency matrix $NADJ_G$ of Deep-Embedding phase to create a set of groups of all data items which is denoted as FG. Each group in FG contains the most similar data items based on $NADJ_G$.

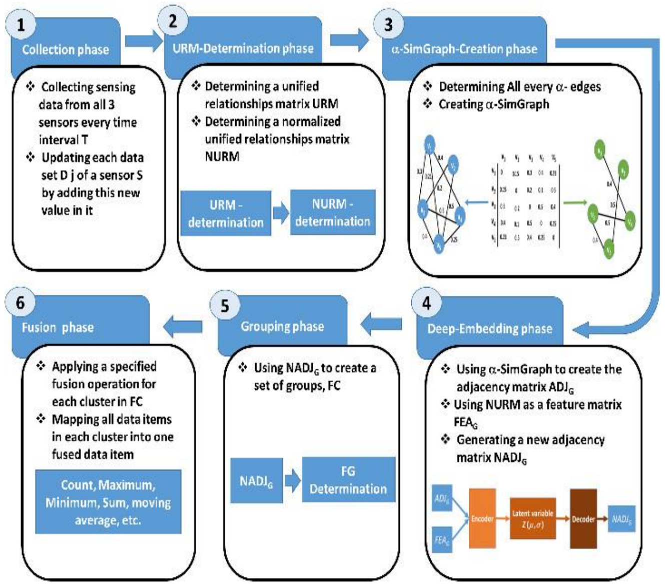
1
Collection phase
Collecting sensing data from all 3 sensors every time interval T
Updating each data set D j of a sensor S by adding this new value in it
2
URM-Determination phase
Determining a unified relationships matrix URM
Determining a normalized unified relationships matrix NURM
URM - determination
NURM - determination
3
α-SimGraph-Creation phase
Determining All every α- edges
Creating α-SimGraph
4
Deep-Embedding phase
Using α-SimGraph to create the adjacency matrix ADJ_G
Using NURM as a feature matrix FEA_G
Generating a new adjacency matrix NADJ_G
ADJ_G
FEA_G
Encoder
Latent variable Z(μ,σ)
Decoder
NADJ_G
5
Grouping phase
Using NADJ_G to create a set of groups, FC
NADJ_G
FG Determination
6
Fusion phase
Applying a specified fusion operation for each cluster in FC
Mapping all data items in each cluster into one fused data item
Count, Maximum, Minimum, Sum, moving average, etc.

Figure 3: The architecture of the proposed DEDF-VGAE phases

F. Fusion phase

DEDF-VGAE scheme applies a specified fusion operation for each group of data items in FG to map all data items in each group into one fused data item. Here, without loss of generality, a fusion operation is considered as an aggregation operation where each group of data items are integrated together to construct a new data value that sums up the data stream. A comprehensive aggregation list is described in [22], [23]. The most common data stream aggregate functions are **Count** which calculates the number of data items in a stream of data (i.e., $X = n$), **Maximum** which calculates the maximum value in a a stream of data (i.e., $X = max\{d_1, \dots, d_n\}$. **Minimum** which returns the minimum value in a data stream (i.e., $X = min\{d_1, \dots, d_n\}$. **Sum** which summarizes all data values in a stream of data (i.e., $X = \sum_{i=1}^{n} d_i$). **Moving average**, **MA**, which returns the average of the data items within a stream of data window $TW = \{t_r, t_{r+1}, \dots, t_c\}$ (i.e., $MA(TW) = \frac{\sum_{i=r}^{c} d_i}{c-r+1}$).
The architecture of the proposed DEDF-VGAE phases are illustrated in Figure 3.

5 SIMULATION, RESULTS, AND EVALUATION

Here, the performance of the proposed scheme DEDF-VGAE fuses data objects to minimize the amount of transferred data and latency time of the collected data to the cloud server and the energy consumption by IoT devices will be evaluated.

5.1 Simulation setting

For evaluating the performance of DEDF-VGAE scheme, a real dataset from the popular UCI Machine Learning Repository. This dataset is "the *activity Recognition system based on Multisensor data fusion (AReM) DataSet"*. This dataset contains temporal data from a Wireless Sensor Network generated by an actor that performs the activities: cycling, bending, lying down, standing, sitting, and walking. For each activity, 15 temporal sequences of input RSS data are present. The total time duration is 120 seconds and each reading item every 250 milliseconds. The dataset has 480 sequences for 42240 instances. This dataset is divided into 6 time windows (TW01, TW02, TW03, TW04, TW05, TW06) and all of with equal size which is 19750 milliseconds. The proposed DEDF-VGAE scheme is implemented by using keras with python platform.

In addition, the different values of $\propto$ are used which are {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}. Here, a fixed time delay for sending one data item from a sensor node or a fusion node to the cloud server is assumed which is 100 milliseconds. In addition, the energy consumed for sending one data from a sensor node or a fusion node to the cloud server is assumed which is 10 energy units. Finally, different values of epochs for VGAE deep learning method is used which are {50, 100, 150, 200, 250, 300}. Each experiment was run for five times and the average values are calculated.

5.2 Performance Matrices

Here, many metrics are used to evaluate DEDF-VGAE scheme, which are described as follows:

A. Time_Delay_Cost, ***TDC***:
TDC represents the total time delay to send data items form sensor nodes or fusion nodes to the cloud server.

B. Minimizing_Time_Delay_Ratio, ***MTDR***:
MTDR represents the minimizing ratio in time delay by using fusion scheme DEDF-VGAE at fusion nodes with respect to the total time delay that are achieved by sensor nodes to send their data items to the cloud server. This metric can be defined as follows:

$$\boldsymbol{MTDR} = 1 - \frac{\boldsymbol{TDC(FN)}}{\boldsymbol{TDC(S)}} \quad (17)$$

where $TDC(FN)$ is the time delay cost that is achieved by fusion nodes in FN with DEDF-VGAE and $\boldsymbol{TDC(S)}$ is the time delay cost that is achieved a set of sensor nodes in S.

C. Consumed_Energy_Cost, ***CEC***:
CEC represents the total consumed energy to send data items form sensor nodes or fusion nodes to the cloud server.

D. Minimizing_Consumed_Energy_Ratio, ***MCER***:
MCER represents the minimizing ratio in consumed energy by using fusion scheme DEDF-VGAE at fusion nodes with respect to the total consumed energy that are achieved by sensor nodes to send their data items to the cloud server. This metric can be defined as follows:

$$\boldsymbol{MCER} = 1 - \frac{\boldsymbol{CEC(FN)}}{\boldsymbol{CEC(S)}} \% \quad (18)$$

where $\boldsymbol{CEC(FN)}$ is the consumed energy cost that is achieved by fusion nodes in FN with DEDF-VGAE and $\boldsymbol{CEC(S)}$ is the consumed energy cost that is achieved a set of sensor nodes in S.

E. Aggregation_Ratio, ***AGR***:
AGR represents a ratio between the aggregated numbers of data items to total number of data items.

F. ***APScore***:
APScore represents the average precision of predicted scores. It summarizes a curve of precision-recall as the weighted average of achieved precisions at each threshold respect to the increase in recall from the previous threshold used as the weight. *APScore* is defined as follows:

$$APScore = \sum_n (R_n - R_{n-1}) \cdot P_n \qquad (19)$$

where R_n and P_n are the recall and precision at the nth threshold [24].

Here, the effects of different values of ∝ and number of epochs will be presented and discussed.

5.3 Results and Discussion

The results of conducted simulations of DEDF-VGAE will be introduced and discussed. These simulations are done for two different parameters: (1) different values of ∝ and (2) different number of epochs. In every experiment, the data is tested based on ***TDC***, ***MTDR***, ***CEC***, ***MCER***, ***AGR*** and ***APScore***.

5.3.1The Effects of different values for ∝

Figures 4, 5, 6, 7, 8, and 9 show the effects of different values for ∝ on the performance of DEDF-VGAE and *NonFusion* schemes when the number of epochs was 250 in terms of ***TDC***, ***MTDR***, ***CEC***, ***MCER***, ***AGR***, and ***APScore***, respectively.

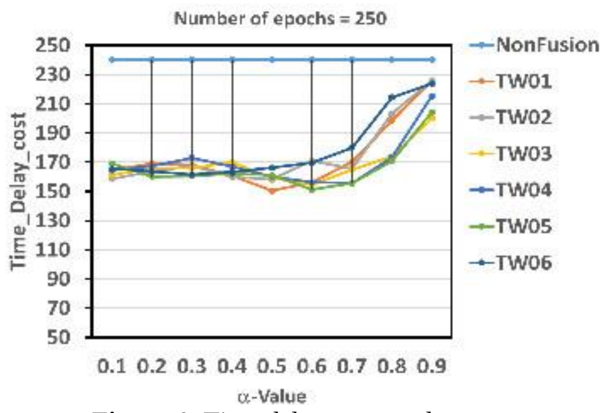

Figure 4: Time delay vs. ∝ -values

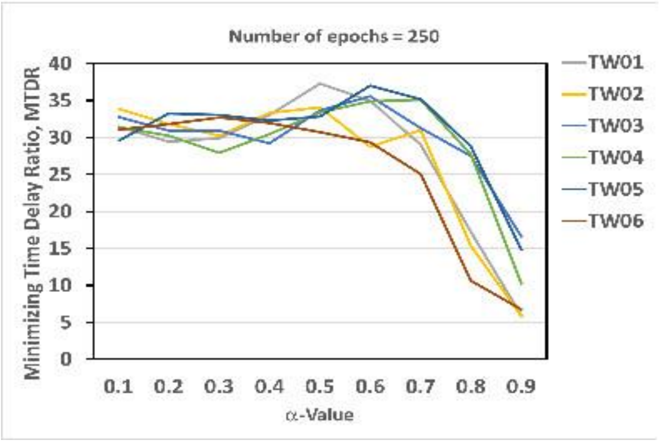

Figure 5: MTDR vs. ∝ -values

As shown in Figure 4, the time delay cost, ***TDC*** is little affected by values of ∝ when ∝ changed from 0.1 to 0.6. While for higher values of ∝ which was changed from 0.7 to 0.9, ***TDC*** increases as ∝ increases. This is because when the value of ∝ increases, the number of fused data items decreases and the total number of non-fused data items increases, which need more time delay for transferring all data items to the cloud server.

As shown in Figure 5, the minimizing time delay, ***MTDR*** is a less changed by values of ∝ when ∝ changed from 0.1 to 0.6 which was between 28% and 38%. While for higher values of ∝ which was changed from 0.7 to 0.9, ***MTDR*** decreased from 38% and 6%. This is because when the value of ∝ increases, the number of non-fused data items increases which minimize the value of ***MTDR.***

As shown in Figure 6, the consumed energy cost, ***CEC*** is little affected by values of ∝ when ∝ changed from 0.1 to 0.6. While for higher values of ∝ which was changed from 0.7 to 0.9, ***CEC*** increases as ∝ increases. This is because, when the value of ∝ increases, the number of fused data items decreases and the total number of non-fused data items increases which need more consumed energy for transferring all data items to the cloud server.

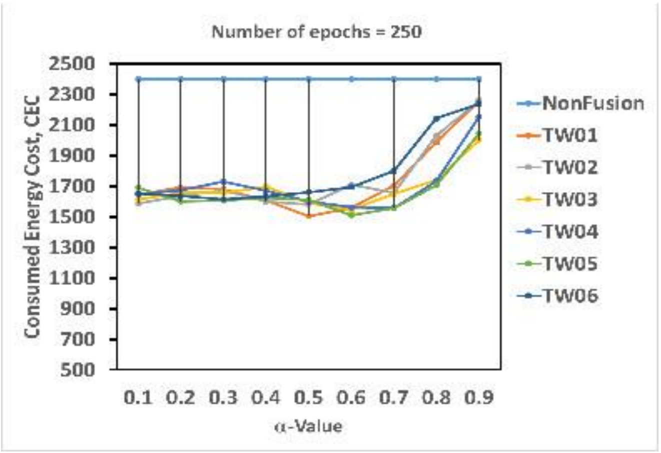

Figure 6: Energy consumption vs. ∝ -values

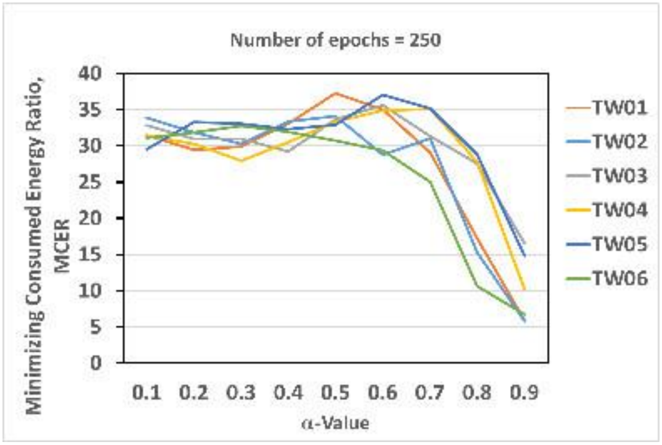

Figure 7: MCER vs. ∝ -values

As shown in Figure 7, the minimizing consumed energy ratio, ***MCER*** is a less changed by values of ∝ when ∝ changed from 0.1 to 0.6 which was between 28% and 38%. While for higher values of ∝ which was changed from 0.7 to 0.9, ***MCER*** decreases as ∝ increases and ***MCER*** decreased from 38% and 6%. This is because, when the value of ∝ increases, the number of non-fused data items increases which minimize the value of ***MCER.***

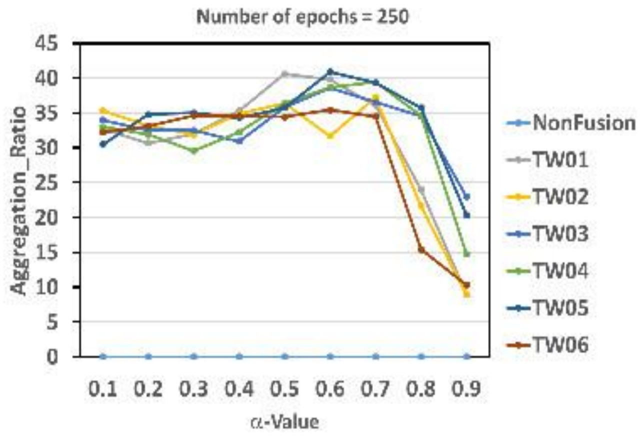

Figure 8: Aggregation ratio vs. ∝ -values

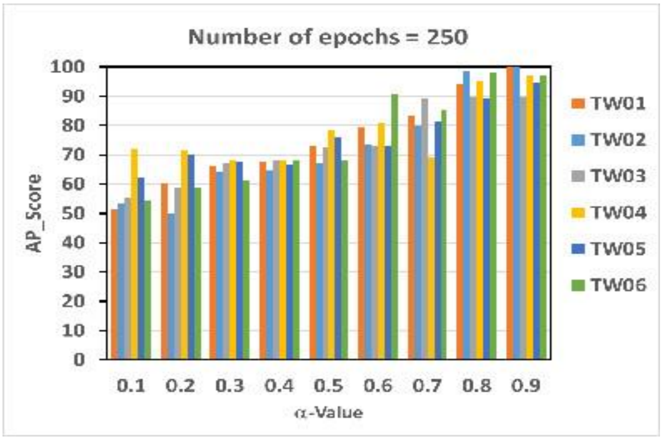

Figure 9: AP score vs. ∝ -values

As shown in Figure 8, the aggregation ratio, ***AGR*** is less affected by values of ∝ when ∝ changed from 0.1 to 0.6. While for higher values of ∝ which was changed from 0.7 to 0.9, ***AGR*** decreases as ∝ increases. This is because, when the value of ∝ increases, the number of non-fused data items increases which minimize the value of ***AGR***.

As shown in Figure 9, the ***APScore*** increases as the value of ∝ increases. This is because, when the value of ∝ increases, the most related data items will be fused and grouped together which will maximize the value of ***APScore***. In addition, the value of ***APScore*** increased from 53% to 100%.

5.3.2The Effects of different number of epochs

Figures 10, 11, 12, 13, 14, and 15 show the effects of different number of epochs on the performance of DEDF-VGAE and *NonFusion* schemes when the value of ∝ was 0.7 in terms of ***TDC, MTDR, CEC, MCER, AGR,*** and ***APScore,*** respectively.

As shown in Figures 10 and 11, the values of time delay cost, ***TDC*** and minimizing time delay ratio, ***MTDR*** are little affected by number of epochs. However, the nature of each dataset affects the values of ***TDC*** and ***MTDR***. As shown in Figure 11, the minimum and the maximum values of MTDR were 24% and 35% for TW06 and TW04, respectively. In addition, the value of ***TDC*** and ***MTDR*** by the proposed fusion scheme are much lower than its value by *Nonfusion* scheme.

As shown in Figures ١٢ and ١٣, the values of consumed energy cost, ***CEC*** and minimizing consumed energy ratio, ***MCER*** are little affected by number of epochs. However, the nature of each dataset affects the values of ***CEC*** and ***MCER***. As shown in Figure 13 the minimum and the maximum values of ***MCER*** were 23% and 36% for TW06 and TW04, respectively. In addition, the value of ***CEC*** by the proposed fusion scheme is much lower than its value by *Nonfusion* scheme.

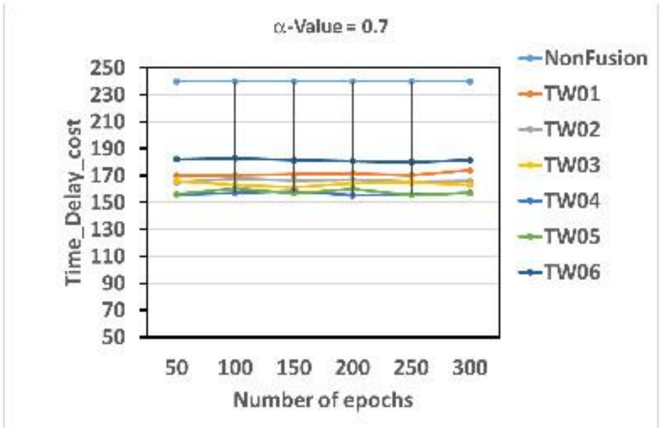

Figure 10: Time delay vs. # of epochs

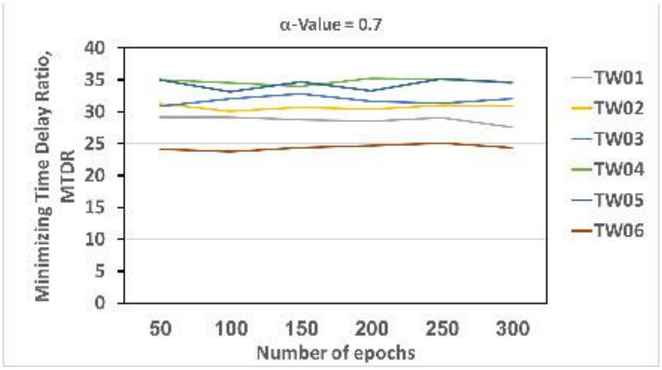

Figure 11: MTDR vs. # of epochs

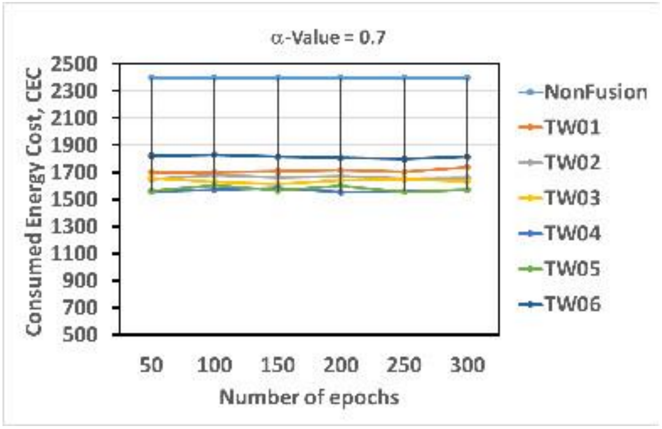

Figure 12: Consumed energy vs. # of epochs

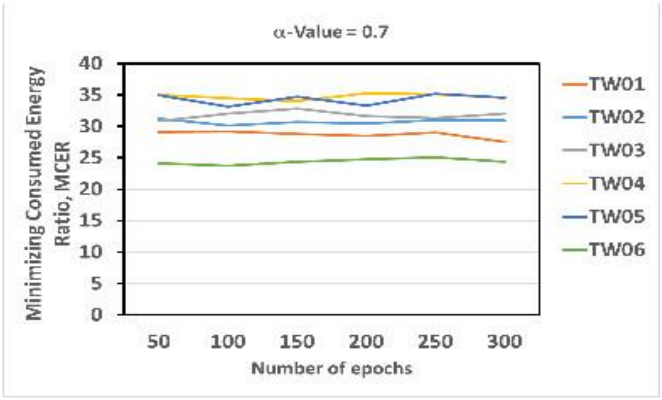

Figure 13: MCER vs. # of epochs

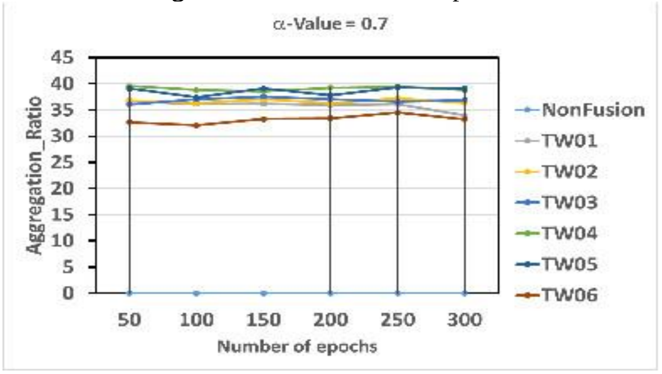

Figure 14: Aggregation ratio vs. # of epochs

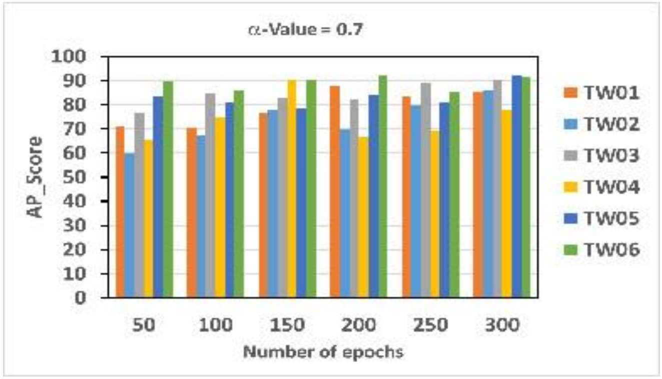

Figure 15: *APscore* vs. # of epochs

As shown in Figures 14 and 15, the values of aggregation ratio, ***AGR*** and ***APScore*** are little affected by number of epochs. However, the nature of each dataset affects the values of ***AGR*** and ***APScore***. In addition, the value of ***AGR*** by the proposed fusion scheme is much higher than its value by *Nonfusion* scheme. Also, the value of ***APScore*** increased from 60% to 93%.

6 CONCLUSION

In this paper, a new fusion scheme was proposed to allow highly effective, reliable and accurate management and decision-making of IoT environments. The proposed scheme was based on using the VGAE deep learning method for creating deep embedding data graph. This graph was based on a specified real value called ∝ which represents the most similar data items into groups that can be fused together based on ∝-value to minimize the amount of transferred data and latency time for transferring the collected data and the energy consumption by IoT devices. To study the performance of the proposed scheme, many simulations experiments for different values of ∝ and number of epochs were conducted. The experiments results show that the proposed scheme can achieve a reasonable performance in terms of latency time, energy consumption, and data fusion accuracy compared to non-fusion schemes. In future work, the proposed data fusion scheme will be improved by determining the most appropriate ∝-value, dynamically. In addition, finding a way to improve the data fusion accuracy of the proposed scheme will be studied. Finally, applying the proposed data fusion scheme in real scenarios to study its adaptation behavior in IoT environments will be considered.

REFERENCES

1. Whitmore, Andrew and Agarwal, Anurag and Da Xu, Li.: “The Internet of Things—A survey of topics and trends”; Information Systems Frontiers, Springer, 17, 2 (2015) 261-274. https://doi.org/10.1007/s10796-014-9489-2
2. WAlam, Furqan and Mehmood, Rashid and Katib, Iyad and Albogami, Nasser N and Albeshri, Aiiad: “Data fusion and IoT for smart ubiquitous environments: a survey”; IEEE Access, 5 (2017) 9533-9554.
3. Zanella, Andrea and Bui, Nicola and Castellani, Angelo and Vangelista, Lorenzo and Zorzi, Michele.: “Internet of

things for smart cities"; IEEE Internet of Things journal, 1, 1 (2014) 22-32. https://doi.org/10.1109/JIOT.2014.2306328

4. Wortmann, Felix and Fl¨uchter, Kristina.: "Internet of things"; Business & Information Systems Engineering, 57, 3 (2015) 221-224.
5. Dong, Jiang and Zhuang, Dafang and Huang, Yaohuan and Fu, Jingying.: "Advances in multi-sensor data fusion: Algorithms and applications"; Sensors, 9, 10 (2009) 7771-7784. https://doi.org/10.3390/s91007771
6. Ahmed. A. A. Gad-Elrab, Shereen A. El-aal, Neveen I. Ghali and Afaf A. S. Zaghrout. A Dynamic Genetic-Based Context Modeling Approach in Internet of Things Environments, IJATCSE, Vol. 8, No. 6, 2019, 2699-2709 https://doi.org/ 10.30534/ijatcse/2019/ 0386 2019
7. M. Asim, B. Rehan, P. Shaheed. Controlling Energy Consumption by Internet of Things (IoT) Applications, IJATCSE, Vol. 8, No. 1.1, 2019. https://doi.org/10.30534/ijatcse/2019/0281.12019
8. Lahat, Dana and Adali, T¨ulay and Jutten, Christian.: "Multimodal data fusion: an overview of methods, challenges, and prospects"; Proceedings of the IEEE, 103, 9 (2015) 1449-1477. https://doi.org/10.1109/JPROC.2015.2460697
9. Kumar, Manish and Garg, Devendra P and Zachery, Randy A.: "A generalized approach for inconsistency detection in data fusion from multiple sensors"; 2006 American Control Conference,IEEE (2006).
10. Smets and Philippe.: "Analyzing the combination of conflicting belief functions"; Information fusion, Elsevier, 8, 4 (2007) 387-412.
11. Jara, Antonio J and Genoud, Dominique and Bocchi, Yann.: "Big data for smart cities with KNIME a real experience in the Smart Santander testbed"; Software: Practice and Experience, 45, 5 (2015) 1145-1160. https://doi.org/10.1002/spe.2274
12. Sobolevsky, Stanislav and Bojic, Iva and Belyi, Alexander and Sitko, Izabela and Hawelka, Bartosz and Arias, Juan Murillo and Ratti, Carlo.: "Scaling of city attractiveness for foreign visitors through big data of human economical and social media activity"; 2015 IEEE International Congress on Big Data, (2015) 600-607.
13. Wang, Meisong and Perera, Charith and Jayaraman, Prem Prakash and Zhang, Miranda and Strazdins, Peter and Shyamsundar, RK and Ranjan, Rajiv.: "City data fusion: Sensor data fusion in the internet of things"; International Journal of Distributed Systems and Technologies (IJDST), 7, 1 (2016) 15-36.
14. Soldatos, John and Kefalakis, Nikos and Hauswirth, Manfred and Serrano, Martin and Calbimonte, Jean-Paul and Riahi, Mehdi and Aberer, Karl and Jayaraman, Prem Prakash and Zaslavsky, Arkady and Zarko, Ivana Podnar and others.: "Openiot: Open source internet-of-things in the cloud"; Interoperability and open-source solutions for the internet of things, (2015) 13-25.
15. Liu, Jie and Zhao, Feng.: "Towards semantic services for sensor-rich information systems"; 2nd IEEE International Conference on Broadband Networks, (2005) 967-974.
16. Izumi, Satoru and Kobayashi, Yusuke and Takahashi, Hideyuki and Suganuma, Takuo and Kinoshita, Tetsuo and Shiratori, Norio.: "A knowledge filtering scheme using sensor data for symbiotic healthcare support system"; 9th IEEE International Conference on Cognitive Informatics (ICCI'10), (2010) 619-624.
17. Teymourian, Kia and Streibel, Olga and Paschke, Adrian and Alnemr, Rehab and Meinel, Christoph.: "Towards semantic event-driven systems"; 3rd International Conference on New Technologies, Mobility and Security, IEEE, (2009) 1-6.
18. Moodley, Deshendran and Simonis, Ingo.: "A new architecture for the sensor web: The SWAP framework"; ISWC, 5th, (2006).
19. Drummond and Oliver E.: "Hybrid sensor fusion algorithm architecture and tracklets"; Signal and Data Processing of Small Targets, International Society for Optics and Photonics, 3163, (1997) 485-502.
20. Liggins II, Martin and Hall, David and Llinas, James.: "Handbook of multisensor data fusion: theory and practice"; CRC press, (2017).
21. Thomas N. Kipf and Max Welling.: "Variational Graph Auto-Encoders"; NIPS Bayesian Deep Learning Workshop, (2016).
22. Maximilian Christ and Andreas W. Kempa-Liehr and Michael Feindt.: "Distributed and parallel time series feature extraction for industrial big data applications"; ArXiv e-prints, (2016), https://arxiv.org/abs/1610.07717.
23. Maximilian Christ and Nils Braun and Julius Neuer and Andreas W. Kempa-Liehr.: "Time Series Feature Extraction on basis of Scalable Hypothesis tests (tsfresh - A Python package"; Neurocomputing, 307 (2018) 72-77. https://doi.org/10.1016/j.neucom.2018.03.067
24. Mu Zhu.: "Recall, Precision and Average Precision"; Department of Statistics and Actuarial Science, University of Waterloo, working paper, 307 (2004) 72-77.