Ministry of Higher Education and Scientific Research
University of Babylon
College of Science for Women
Department of Computer Science

Social Community Detection using centrality measurement in Graph Autoencoder technique

A Thesis submitted to the council of the College of Sciences for Women at University of Babylon in Partial Fulfillment of the Requirements for the Degree of Master in Sciences\ Computer Sciences

By
Hawraa Zuhair Ahmed
Supervised By
Assistant Prof. Dr. Asia Mahdi Naser Alzubaidi

2022 م 1444هـ

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

نَرْفَعُ دَرَجَٰتٍ مَّن نَّشَآءُ ۗ وَفَوْقَ كُلِّ ذِي عِلْمٍ عَلِيمٌ

صَدَقَ اللهُ العَظِيمُ

(وسف: الآية ٧٦)

Supervisors Certification

I certify that this thesis entitled "**Social Community Detection using centrality measurement in Graph Autoencoder technique**" was done by (Hawraa Zuhair Ahmed) under my supervision.

Signature:

Name: Assistant Prof. Dr. Asia Mahdi Naser Alzubaidi

Date: /12/2022

Address: College of Computer Science & Information Technology

Karbala University

The Head of the Department Certification

In view of the available recommendations, I forward the dissertation entitled "**Social Community Detection using centrality measurement in Graph Autoencoder technique**" for debate by the examination committee.

Signature:

Name: Assistant Prof. Dr. Saif Mahmood Khalaf

Date: /12/ 2022

Address: University of Babylon/College of Science for Women

Certification of the Examination Committee

We, the chairman and members of the examining committee, certify that we have read this thesis entitled (**Social Community Detection using centrality measurement in Graph Autoencoder technique**) and after examining the master student (**Hawraa Zuhair Ahmed**) in its contents in 14/11/2022 and that in our opinion it is adequate as a thesis Committee Chairman for degree on Master in Science \Computer Science with degree **(Very Good)**.

Committee Chairman

Signature:

Name: **Samaher Al-Janabi**

Scientific order: Prof. Dr.

Address: University of Babylon\ College of Science for Women

Date: / 12 /2022

Committee Member

Signature:

Name: **Saif Mahmood Khalaf**

Scientific order: Ass. Prof. Dr.

Address: University of Babylon\ College of Science for Women

Date: / 12 / 2022

Committee Member

Signature:

Name: **Asaad Noori Hashim**

Scientific order: Ass. Prof. Dr.

Address: University of Kufa\ College Computer Science and Mathematics

Date: / 12/ 2022

Committee Member (**Supervisor**)

Signature:

Name: **Asia Mahdi Naser Alzubaidi**

Scientific order: Ass. Prof. Dr.

Address: Karbala University\ College of Computer Science & Information Technology

Date: / 12 / 2022

Approved by the Dean of the College of Science for Women\ University of Babylon

Signature:

Name: Abeer Fauzi Al-Rubaye

Address: University of Babylon\ College of Sciences for Women

Title: Prof. Dr.

Date: / 12 / 2022

Dedications

To whom I prefer her to myself, and why not; she sacrificed for me

You have always spared no effort to make me happy

(My beloved mother)

To the owner of a fragrant biography and an enlightened thought;

The owner of a kind face, and good deeds.

He has never been stingy with me all his life

(My dear father).

to my husband and brothers; Who had a great impact on many obstacles and difficulties.

To all my dear teachers; Who did not hesitate to extend a helping hand to me

Hawraa

Acknowledgments

First of all, we thank God for the most mercy for enabling us to present this thesis in the best form that we wanted it to be.

We would like to thank our supervisor for this project, Associate Prof. Dr. Asia Mahdi Naser Alzubaidi for his valuable help and advice to come out with this thesis.

Thanks, and Gratitude to all my professors and all the staff of the Department of Computer Sciences \ College of Sciences for the Women \University of Babylon for their help.

Hawraa

Abstract

Recently, the utilization of social network platforms, such as Facebook, Twitter, and WeChat, has spread greatly. Community detection, that it is grouping of users with similar interests and tendencies into groups, is an important side of understanding the complex structure of social networks.

The users are represented as nodes in the social network. The features of nodes, in addition to the network topology, describe the similarity relationship between them. The challenge of community detection is dealing with featureless networks. The graph autoencoder technique has low accuracy in community detection. Therefore, the programming challenge lies in developing the structure of this technology to give high accuracy results.

This thesis used the hybrid method, community detection using graph autoencoder (CDGAE) framework. Where the centrality measurement-based method was used to assign features, to deal with the featureless network. Then two feature selection methods, correlation and variance threshold feature selection, were used to avoid redundancy and padding in features. The layers of GAE were increased to increase the accuracy of community detection. The neural number of the bottleneck layer of Graph Autoencoder (GAE) was equated to the community's number, it was determined by the spectral method of the Bethe Hassien matrix, to determine the membership of each node to its community directly without using the clustering algorithm.

The CDGAE was applied to the Facebook government pages dataset. The experimental results showed the best performance of CDGAE with five layers for the Facebook government pages dataset after augmentation of its nodes with

features, where the Normalized Mutual Information (NMI-score) and Accuracy-score recorded the percentage values 84.86 and 87.42 respectively.

List of contents

Chapter One: General Introduction

Chapter Two: Theoretical Background

Chapter Four: Experimental Results and Discussion

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

A	Adjacency Matrix
AAGR	Adaptive Autoencoder with Graph Regularization
AC	Accuracy Score
AF1L	All feature one layer
AF3L	All feature three layer
AF5L	All feature five layer
CDDTA	Community detection with deep transitive autoencoder
CDGAE	community detection framework Graph Autoencoder
CDMEC	Community Detection Method via Ensemble Clustering
CF1L	Correlation feature one layer
CF3L	Correlation feature three layer
CF5L	Correlation feature five layer
CFS	Correlation-based feature selection
CGN	Convolutional graph neural network
DIR	Deep integration representation
DNR	Deep nonlinear reconstruction
FL1L	Featureless one layer
FL3L	Featureless three layer
FL5L	Featureless five layer
FN	False negative
FP	False positive
GAE	Graph Autoencoder
GN	Girvan and Newman
GNN	Graph Neural Networks

ITS	intelligent transmission systems
JS	Jaccard Score
LFR	Lancichinetti–Fortunato–Radicchibenchmark
MGAE	Marginalized graph autoencoder
NC	Numbers of communities
NMI	Normalized Mutual Information
PS	Precision Score
RecGNN	Recurrent graph neural network
RS	Recall Score
sim	CosineSimilarity
TN	True negatives
TP	True positives
VGAECD	Variational Graph Autoencoder for Community Detection
VTF1L	Variance Threshold feature one layer
VTF3L	Variance Threshold feature three layer
VTF5L	Variance Threshold feature five layer
VTFS	Variance Threshold-based feature selection
N	Node
$\mathcal{G}$	Graph

List of symbols

$\boldsymbol{H}$	Bethe Hessian matrix
$\boldsymbol{r}$	regularize value
$\boldsymbol{A}$	Adjacency matrix
D	diagonal matrix with degrees d_i on the diagonal
I	identity matrix
$\mathbf{d_i}$	Degree of node i
$\boldsymbol{\tau}$	Activation function
B	Bias
$\mathbf{A}^*$	Normalized version of A
$\boldsymbol{ReLU}$	the rectified linear activation function
$\boldsymbol{\theta}$	optimizer parameter
sim	cosine similarity
$\|\|\boldsymbol{x}\|\|$	Euclidean norm of vector x
N(v)	The group of node v's immediate neighbors
$\|\boldsymbol{N}(\boldsymbol{u})\|$	Number of node u 's immediate neighbors
$\boldsymbol{qv(u,w)}$	the number of squares that u and w have common neighbors with besides v
|E' |	number of edges in E'
$\boldsymbol{\lambda}$	Eigenvalue
$\boldsymbol{\sigma_{st}}$	the quantity of pathways between s and t that are shortest
$\boldsymbol{\theta_{st}(v)}$	the portion of the unit quantity that traveled through node v on the shortest path from node s to node t provided that the quantity is split evenly at each branch encountered
d(v,u)	The distance between nodes v and u
$\mathbf{I_{uv}}$	Weighing the information along each path from u to v according to its length
$\boldsymbol{\delta}$	The decay rate
$\|\boldsymbol{x}(\vec{\boldsymbol{e}})\|$	the throughput in case of an st-current
∑xy	sum of products of the paired stocks
$\sum \boldsymbol{x}$	sum of the x scores
$\sum x^2$	sum of the squared x scores
$\bar{\boldsymbol{x}}$	The mean
P(X)	Probability distribution

Chapter One
General Introduction

Chapter One

General Introduction

1.1 Introduction

In recent years, the utilization of social networks has spread greatly. A social network is a social structure consisting of a set of social entities (such as individuals or organizations), sets of dual links, and other social interactions between entities. The term social network refers to the employment of internet-based social media sites to enable interactions with friends, family, or clients. The social network could have social, business objectives, or both, through sites like Facebook, Twitter, Twitch, and Instagram. Social networks and their analysis of them is an inveterately interdisciplinary academic field engendered from social psychology, sociology, statistics, and graph theory. The data science consists of algorithm domain, statistical domain and application domain. Social network analysis falls within the application domain of application domain where uses networks and graph theory to understand social structures.[1], [2].

A network can be visualized as a collection of nodes connected by edges. Graph Theories and concepts are used to study and model Social Networks. Many real-world network systems can be represented by network, as the World Wide Web, networks for scientific collaboration, social networking sites, and many more because they show any signs of communities. Therefore, to comprehend the operation and topology of complex networks, it is needed need to unveil these structures by community detection[3], [4].

As a result of the evolution of the internet and its applications in modern life, Numerous real-world networks have seen the use of community

detection, including financial applications, marketing, neuroscience and image understanding, and online social networks. Online social network applications, including Facebook, Twitter, and WeChat, comprise the interactions among people through the web. Detecting communities in such networks is an effective way to infer the relationships of individuals, which has been adopted for tasks such as spammer detection and crisis response [5], [6].

Community detection is a method of grouping nodes in a complex network into communities, where nodes within a community are sparsely connected to those of other communities and highly connected to one another[6], [7].

The Graph Autoencoder (GAE) is a special type of Graph Neural Network (GNN) that have recently been widely used in the field of machine learning, for their ability to deal with structured data. GAE is characterized by its ability to unsupervised learning from the input data. GAE consists of a set of connected components that work together through an iterative process of aggregation of data across nodes, catch the complex dependences of the fundamental system and represent them into a low-dimensional approximation. This allows us to predict the properties of the specific nodes, links, or the whole graph and generalize their properties to hidden graphs[3], [8].

1.2 Related Works

This section reviews briefly some of the previously proposed methods related to community detection:

- In 2016 [9], Liang Yang et al., proposed deep nonlinear reconstruction (DNR) and (semi-DNR) methods for community detection. They

organized the modularity matrix as input to the Autoencoder and then used the KMeans clustering algorithm for community detection. The (Karate, Dolphins, Friendship6, Friendship7, Football, Polbooks, Polblogs, and Cora) datasets were Used. The detected communities using the proposed method were evaluated using Normalized Mutual Information (NMI) for datasets respectively by these values (1, 0.889, 0.924, 0.932, 0.927, 0.582, 0.517, 0.463).

- While in the study by Chun Wang et al., In 2017 [10] , they proposed a marginalized graph autoencoder (MGAE) method for community detection. They Combined the contents of the nodes with the network topology as input to the Graph Autoencoder and then use the spectral clustering algorithm for community detection. The (Wiki, Citeseer, Cora) datasets were Used. The detected communities using the proposed method were evaluated using Normalized Mutual Information (NMI) for datasets respectively by these values (0.416, 0.489, 0.51), and Accuracy Score (AC) respectively by these values (0.669, 0.680, 0.529).
- In another study, in 2017 [11], Di Jin et al., proposed a deep integration representation (DIR) algorithm using Autoencoder in deep learning for community detection. They used the normalized-cut matrix and modularity matrix as input for the Graph Autoencoder and then used the KMeans clustering algorithm for community detection. The (cornell, Texas, Washington, Wisconsin, Citeseer, Cora, UAI2010 and Pubmed) datasets were Used. The detected communities using the proposed method were evaluated using Normalized Mutual Information (NMI) for datasets respectively by these values (27.50,

26.89, 31.79, 29.65, 27.42, 40.30, 31.86, 22.44) and Accuracy Score (AC) respectively by these values (45.79, 64.03, 60.51, 46.79, 41.85, 60.71, 31.07, 59.60).

- Also, in 2018 [12], Cao J et al., proposed an Adaptive Autoencoder with Graph Regularization (AAGR) for community detection, which is reached by a graph regularized autoencoder approach. They used the Laplacian matrix and modularity matrix as input for the Graph Autoencoder and then used the spectral clustering algorithm for the community detection. The (Citeseer, Cornell, Texas, Washington and Wisconsin) datasets were Used. The detected communities using the proposed method were evaluated using Normalized Mutual Information (NMI) for datasets respectively by these values (35.11, 18.93, 39.4, 26.85, 36.67).
- Another study, in 2018 [4], Jinxin Cao et al., suggested using the autoencoder to integrate the modularity and normalized-cut models. and then used the KMeans clustering algorithm for community detection. The (Cora, Facebook107, PubMed, Texas, Twitter629863, Uai2010, Washington and Wisconsin) datasets were Used. The detected communities using the proposed method were evaluated using Normalized Mutual Information (NMI) for datasets respectively by these values (0.364, 0.418, 0.51, 0.319, 0.372, 0.549, 0.32, 0.315, 0.342) and Accuracy Score (AC) respectively by these values (0.39, 0.452, 0.601, 0.23, 0.654, 0.632, 0.47, 0.496, 0.48).
- In 2019 [13], YingXie et al., proposed a framework to extract nonlinear features: community detection with deep transitive autoencoder (CDDTA). They employed the Graph Autoencoder with the similarity matrix as the input, and then used the KMeans clustering

algorithm for community detection. The (Karate, Dolphins, Friendship6, Friendship7, Football, Polbooks, Polblogs, Cora and Citeseer) datasets were Used. The detected communities using the proposed method were evaluated using Normalized Mutual Information (NMI) for datasets respectively by these values (0.9, 0.98, 0.98, 097, 0.6, 0.75, 0.5, 0.4).

- And in the study of jun Jin et al., in 2020 [14] it proposed Variational Graph Autoencoder for Community Detection (VGAECD). They set the adjacency matrix & feature matrix as input to the Variational Graph Autoencoder which uses convolutional layers as encoder and then used the KMeans clustering algorithm for community detection. The (Karate, Polbloges, Cora and PubMed) datasets were Used. The detected communities using the proposed method were evaluated using Normalized Mutual Information (NMI) for datasets respectively by these values (1.0, 0.758, 0.544, 0.355) and Accuracy Score (AC) respectively by these values (1.0, 0.960, 0.319, 0.322).
- In 2020 [15], RongbinXu et al., proposed Stacked Autoencoder-Based Community Detection Method via Ensemble Clustering (CDMEC) for community detection. They employed the Graph Autoencoder with the similarity matrix as the input, and then used the KMeans clustering algorithm for the community detection. The (Karate, Dolphins, Friendship6, Friendship7, Football, Pollbooks, Polblogs, Cora, Citeseer) datasets were Used. The detected communities using the proposed method were evaluated using Normalized Mutual Information (NMI) for datasets respectively by these values (1, 1, 0.98, 0.98, 0.98, 0.63, 0.67, 0.55, 0.42).

Table (1.1) summarizes the related works mentioned above.

Table (1.1): compare a many the previous work

Author and year	method	datasets	(NMI)	(AC)
Liang Yang et al. in 2016	deep nonlinear reconstruction (DNR) and (semi-DNR) method	Karate	100	-
		Dolphins	88.9	-
		Friendship6	92.4	-
		Friendship7	93.2	-
		Football	92.7	-
		Polbooks	0.582	-
		Polblogs	0.517	-
		Cora	0.463	-
Chun Wang et al. in 2017	marginalized graph autoencoder (MGAE)method	Citeseer	0.416	0.669
		Cora	0.489	0.680
		Wiki	0.51	0.529
Di Jin et al. in 2017	deep integration representation (DIR) algorithm	Cornell	27.50	45.79
		Texas	26.89	64.03
		Washington	31.79	60.51
		Wisconsin	29.65	46.79
		Citeseer	27.42	41.85
		Cora	40.30	60.71
		UAI2010	31.86	31.07
		PubMed	22.44	59.60
Jinxin Cao et al. in 2018	Adaptive Autoencoder with Graph Regularization (AAGR).	Citeseer	35.11	-
		Cornell	18.93	-
		Texas	39.4	-
		Washington	26.85	-
		Wisconsin	36.67	-

Author and year	**method**	**datasets**	**(NMI)**	**(AC)**
Jinxin Cao et al. in 2018	Using the autoencoder to integrate the modularity and normalized-cut models	Citeseer Cora Facebook107 PubMed Texas Twitter629863 Uai2010 Washington Wisconsin	0.364 0.418 0.51 0.319 0.372 0.549 0.32 0.31 0.342	0.39 0.452 0.601 0.23 0.654 0.632 0.47 0.496 0.48
YingXie et al.in 2019	community detection with deep transitive autoencoder (CDDTA)	Dolphins Friendship6 Friendship7 Football Polbooks Polblogs Cora Citeseer	0.9 0.98 0.98 097 0.6 0.75 0.5 0.4	- - - - - - - -
jun Jin et al. in 2020	Variational Graph Autoencoder for Community Detection (VGAECD)	Karate Polblogs Cora PubMed	1.0 0.758 0.544 0.355	1.0 0.960 0.319 0.322
Rongbin Xu et al. in 2020	Community Detection Method via Ensemble Clustering (CDMEC)	Karate Dolphins Friendship6 Friendship7 Football Pollbooks Polblogs Cora Citeseer	1 1 0.98 0.98 0.98 0.63 0.67 0.55 0.42	- - - - - - - - -

1.3 Problem statement

This section explains the problem of dividing the social network into communities

The communities are detected according to the similarities between the nodes. So, the community detection challenge is how to extract the similarity relationships between nodes in the social networks. The nodes feature, in addition of networks topology, describe the similarities between them. Node features are not available in many real-world graphs due to privacy concerns or onerousness in collecting node features, so a way has to be found to deal with featureless graphs.

The graph autoencoder technique has the low accuracy by using it in the community detection. Therefore, the programming challenge lies in developing the structure of this technology to give high accuracy results.

1.4 Objective of the thesis

This work aims to detect communities from social networks in the real world by:

1. Using the Social network analysis, the datasets is expanded by assigning features to the nodes using the centrality measurements, to extract similarity relationships between graph nodes.
2. Utilizing the feature selection to the nodes by employing two methods (correlated-based, variance threshold-based) to help in identifying the most useful features in the datasets to improve the problem of high dimensions.

3. Utilizing the spectral method of the Bethe Hessian matrix to determine the communities' number, and embedding it in the bottleneck layer of GAE to detection of communities.

1.5 Organization of Thesis

The subsequent chapters in this thesis content are briefly discussed as follows:

1 **Chapter one (General Introduction):** The current chapter contains a general introduction to the thesis.

2 **Chapter two (Theoretical Background):** This chapter provide a background overview of community detection, the Feature Initialization Method, feature selection and the graph autoencoder.

3 **Chapter three (The proposed community detection framework Graph Autoencoder System (CDGAE)):** This chapter explains the proposed community detection framework Graph Autoencoder System (CDFGAE). In addition, it discusses community detection with feature selection and the deeper model for the proposed system.

4 **Experiment Results and Discussion in Chapter 4:** This chapter will present the suggested system's experimental findings.

5 **Conclusions and Recommendations for Further Study, Chapter 5:** This chapter offers various recommendations for further research projects as well as the inferences drawn from the system's findings.

Chapter Two
Theoretical Background

Chapter Two
Theoretical Background

2.1 Introduction

In recent years, social networks have become very widespread and their use has popular increased. So, research in the domain of social networks has garnered a substantial amount of attention from the scientific community. One of the key exploration fronts in this field is community structures. Community detection plays a critical part in grasping the functionality and structures of social networks.

This chapter will go into the theoretical underpinnings of the topics and ideas required for the proposed system.

2.2 Community detection

A community is known as a collection of elements nearer together in contrast with other elements of the dataset. A community is consisting of individuals who inside a community communicate with one another more frequently than with those external to the community [2].

The job of community detection is crucial in network analysis and aims to stratify the graph into multiple sub-structures to understand the overall structure and the components of the graph [16], [17] .

graph structures have been researched deeply in the form of subgraphs, graph modules, and communities. In the graph, communities are formed by nodes that are densely connected and share common features, while they (those nodes) have sparse connections with other groups of connected nodes (other communities)[3], [4] .

Graph Neural Networks (GNNs) become increasingly popular as it is able to achieve the best results on different learning tasks, where they allow taking into account both the graph structure and the node features. So, they have been used a lot lately for resolving graph tasks [18].

- **The number of communities using the spectral Bethe Hessian matrix approach.**

The spectral characteristics of the Bethe Hessian matrix resulted in evaluations of the number of communities.

The definition of the Bethe Hessian matrix is:

$$H(r) = (r^2 - 1)I - rA + D \quad \ldots(2.1)$$

Where A is the adjacency matrix, D= diag(d_i) is N × N diagonal matrix with degrees d_i on the diagonal, I is N × N identity matrix, and $r \in \mathbb{R}$ is a regularize value $|r| = r_c$ that define as following:

$$r_c = \sqrt{\left(\sum_{i=1}^{n} d_i\right)^{-1}\left(\sum_{i=1}^{n} d_i^2\right) - 1} \quad \ldots(2.2)$$

To calculate the communities number, the spectral method is used with the Bethe Hessian matrix as follows: computing the H(r_c) negative eigenvalues number.

The negative eigenvalues of H(r_c) reveal the assortative aspects of the graph (i.e., the number of the negative eigenvalues equals the number of communities)[19], [20].

2.3 Graph Neural Network GNN

GNN have developed onto the machine learning scene lately, attributable to their capacity to model and gain from graph-structured data. Such strength has solid ramifications in a wide assortment of fields whose information is inherently relational, for which traditional neural networks

don't perform well, where GNN is an optimizable transmutation on all attributes of the graph (nodes, edges, global-context) that preserves graph uniformity (permutation invariances). Research in the space of GNNs has developed quickly and has prompted the expansion of an assortment of GNN algorithm variants as well as the exploration of groundbreaking applications. Variants of GNNs have shown groundbreaking performances on many deep learning tasks. GNNs strive to learn the structure of graphs through various learning algorithms that transform graph data from the original input space to a low-dimensional feature space, to get the advantages of low computational cost and strong parallelization capacity [21], [22].

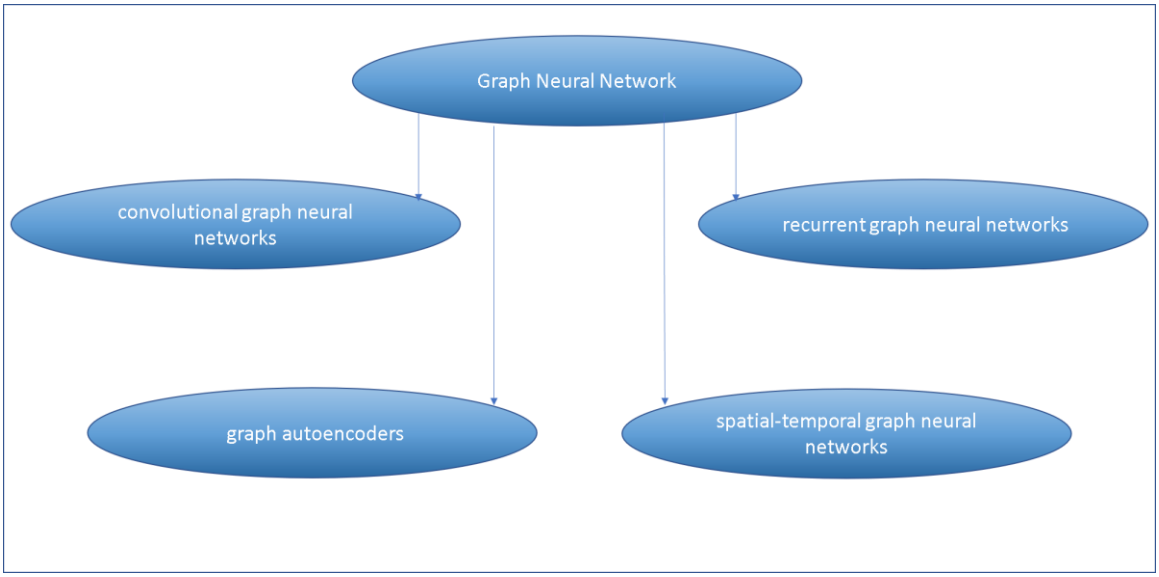


Figure (2.1): Classification of graph neural

GNNs are neural models that catch the reliance of graphs by means of message passing between the nodes of graphs. GNNs adapt their structure to that of an input graph and, through an iterative operation of aggregation of information across nodes, catch the complex dependences of the fundamental system, where the GNN is a set of neural network models that are distributed in layer(s) designed to perform different tasks.

This allows for foreseeing properties for specific nodes, connections, or the overall graph, and the capability of generalization to the concealed graphs [21].

In the GNN, the following equation is ran to deploy the exemplification features to the subsequent layer (forward pass).:

$$h^{[i+1]} = \tau\left(W^{[i]}H^{[i]} + b^{[i]}\right) \ldots (2.3)$$

Where i is the number of layers, h is the Feature representation, τ is the activation function, W is the weights, and the bias is b. To make the non-linear features in the hidden layers visible in the forward pass equation above, neural networks use non-linear activation functions in a systematic manner.

The equation is simplified as follows (for the first layer i = 0):

$$h^{[1]} = \tau\left(W^{T}[0]X + b^{[0]}\right) \ldots (2.4)$$

GNN updates generally consist of these two easy steps:

- Aggregate or analyze neighboring nodes' statuses, often known as "message forwarding".
- The node state updating.

GNNs are classified into four classes as show in figure (2.1): “recurrent graph neural networks, convolutional graph neural networks, graph autoencoders, and spatial-temporal graph neural networks” [23].

2.3.1 Recurrent graph neural networks (RecGNNs): Learning node representations with recurrent neural structures is the aim of RecGNNs. To read out high-level node representations, they applying a set of parameters to the nodes in a graph. RecGNN updates node states based on a data publication process by exchanging neighborhood data regularly until a stable equilibrium is attained. The state of a RecGNN

depends on both the network's previous hidden state and the current inputs, which is what distinguishes it from other types of neural networks. The concealed case of a node is regularly updated by:

$$\mathbf{h}_v^{(t)} = \sum_{u \in N(v)} f\left(\mathbf{x}_v, \mathbf{x}_{(v,u)}^{\mathbf{e}}, \mathbf{x}_u, \mathbf{h}_u^{(t1)}\right) \ldots (2.5)$$

Where h(.) is initialized at random and f(.) is a parametric function. GNN can be applied to all nodes, even when the number of neighbors fluctuates and neighborhood ordering is well-known, thanks to the sum process. The f(·) must be a contraction mapping, which shrinks the distance between two points after projecting them into a latent space to reach a suitable convergence. When a convergence gauge is satisfied, the latest step node's hidden states are forwarded to an output layer.

RecGNN's ability to handle cyclic networks is an advantage due to its trait of trading the step of node state propagation and the phase of parameter gradient computation to lower the training objective.

They are employed in tasks that demand sequential memory and decision-making (e.g. reinforcement learning)[23], [24].

2.3.2 Convolutional graph neural networks (CGNs): are type of convolutional neural network (CNN) that can act directly on graphs and use their structural information. One of the fundamental varieties of Graph Neural Networks is the GCN. Convolution is the process of multiplying the input neurons by a collection of weights, sometimes referred to as filters or kernels. The filters allow GNNs to learn features from nearby nodes by acting as a sliding window across the entire network. Weight sharing is the process of using a certain filter within each layer to search the entire graph for a particular set of features [22].

Along with the node features, GCNs also take into account the Adjacency Matrix (A) in the forward propagation equation (or so-called input features). The edges or connections between the nodes in the forward propagation equation are represented by the matrix A. The model is given it go to learn feature exemplifications based on node connectivity by inserting A into the forward pass equation. The forward pass equation will then be obtained by including the adjacency matrix as an additional component:

$$h^{[i+1]}=\sigma\left(W^{[i]}H^{[i]}A^{*}\right)\ldots(2.6)$$

Where A^{*} is the normalized version of A. the normalized version of A is the A itself and self-loops to each node of A to allow each node to learn its features in addition to the features of its neighborhood

The Applications of GCN

In the field of images, GCNs (after converting unstructured images to structured graph data) is used to classify images, answer visual questions that explore answers to questions on images, explore visual relationships for annotation images, and process input scene graphs. The attentional GCN model is used to place attention on reliable edges while dampening the influence of unlikely edges.

In Point clouds, GCNs is used to segmentation on point clouds, classify and segment raster clouds, and generate 3D point clouds.

GCN has also been used to discover correspondences between groups of 3D shapes.

Several models of GCNs have been proposed in natural language processing to classify text, by assuming that the documents are nodes and the citation relationships between them are edges, also for extraction of the relation between words, event extraction, and various

NLP applications such as semantic role labeling, neural machine translation, and for dependence parsing [25].

Spatial graph convolutional networks and spectral graph convolutional networks are the two main types of GCNs.

2.3.3 Spatial-temporal graph neural networks: are the type of GNN which are developed to deal with the spatiotemporal graphs, which can cope with time-varying graph properties and consist of static structures and time-varying features. These neural networks are sophisticated enough to analyze time chains utilizing the graph's time-varying characteristics. Since time chains datasets have causality, which means that past information is strongly correlated with both present and future information, they differ from geographical datasets in the machine learning field of time chains analysis [22]. Since time chains can be considered separate data, this can be generalized to the graph domain and learn by concatenating the graphs. A graph can be recorded once, and another graph can be recorded at different times. A straightforward GNN can be utilized in conjunction with temporal blocks thanks to the static graph topology. A matrix of node and edge attributes and an adjacency sparse matrix that creates a graph structure in forwarding processing serve as the models' inputs[26].

Applications of Spatial-temporal graph neural networks:

These networks can be utilized in a variety of intelligent transmission systems (ITS) applications, including route planning, navigation, traffic control, and management since they can recognize the spatial and temporal patterns of the graph.

Spatio-temporal graph neural networks can be used in conflict prediction and pandemic forecasting due to its capacity to handle huge data.

These networks can also be used to improve logistics pricing and plan inventory efficiently for online marketplaces[26].

2.3.4 Graph autoencoders (GAE).

GAEs are a special type of Graph neural network architecture whose output is the same as their input. The structure of GAE is: the encoder unit and the decoder unit which are separated by a bottleneck layer, that having a fewer neuron than either layer. The encoder and decoder consist of an equal number of layers. The neurons per layer lowers with each subsequent layer of the encoder, and increases back in the decoder (figure (2.2)).

GAE are trained in an unsupervised way to reach the low-level representations (Latent View Representation) of the input data. GAEs use the encoder and decoder units to learn these low-level representations of the input in an unsupervised way. These low-level features are then used to reconstruct the input data. GAE aims to reduce the error between the original input and the marked data in order to learn the best low-level representation. After obtaining the best output of the decoder, the decoder is disconnected and the handling is limited to the bottleneck layer's low-dimensional encoding for model output. GAEs are utilized to minimize the size of inputs into a smaller representation [4], [27].

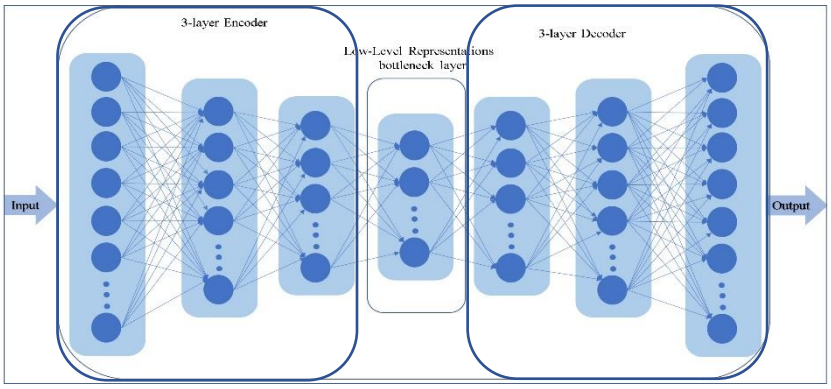


Figure (2.2): Graph Autoencoder structure

Applications of GAEs

In Images

GAEs are used to Image Coloring, converting any black and white picture into a colored image depending on what is in the picture, it determines what color it should be.

they are used to Image Enhancement, produces the image by clearing each noise or undue obstruction after extracting the required features of an image only.

they are also used for removing watermarks from images or to remove any object while filming a video or a movie.

they are used for features Reducing, the output features reconstructed from the its input but with reduced dimensions.

In graphs

they are used for Link Prediction, produces the predicts the missing links from graph structure and node features[26].

One of the benefits of GAE's in the unique design them by compressing data in the bottleneck layer, helps to decide which aspects of the observed data are relevant and which ones can be ignored.

2.4 Graph Autoencoder (GAE) Model

The undirected and connected graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of $N = |\mathcal{V}|$nodes and $\mathcal{E}$ edges are considered, that connected the pair of nodes in graph, an adjacency matrix $A \in \mathbb{R}^{N \times N}$ is the binary matrix such that $A_{ij} = 1$ whenever $(i, j) \in \mathcal{E}$, and $A_{ij}=0$ otherwise, and the feature matrix $X \in \mathbb{R}^{N \times F}$ that has the features of nodes. The a_i is the ith row of the matrix A.

The hidden representations for the encoder and decoder parts are computed as follows:

Encoder layer:

$$\acute{z}_i = \tau(W_i \bar{a}_i + b) \ldots(2.7)$$

Where $\acute{z}_i$ is ith column vector encoded, τ is the rectified linear unit activation function ($ReLU$) ,W is the weights, and the bias is b. [28], [29].

Decoder layer:

$$\hat{\mathbf{a}}_i = \tau(W_2 \acute{z}_i + c) \ldots(2.8)$$

Each layer must have the activation function applied on data to optimized them, one of this activation function is ReLU that used for all hidden layers except the output layer used the SoftMax activation function [4], [8].

$$f(x) = ReLU = max(0, x) \ldots(2.9)$$

$$\text{softmax}\,(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \ldots(2.10)$$

The neurons' mapping range when SoftMax is the activation function, is (0, 1). The output is said to be active when it is close to 1, while the output is said to be inactive when it is close to 0.

Each layer of the GAE contains a Dense Layer, a ReLU Activation Function, and a BatchNormalization Layer, except for the bottleneck layer that contains the Dense layer only and the last layer of the decoder that contains SoftMax Activation Function instead of the ReLU Activation Function.

- **The Dense Layer**

The dense layer is deeply connected to its preceding layer, meaning that every neuron in the layer is connected to every neuron in the preceding layer. By doing matrix-vector multiplication or shrinking, it modifies the output's dimension.

A model's dense layer neurons conduct matrix-vector multiplication, and they receive output from every neuron in the layer above them. The row vector of the output from the previous layers is equivalent to the column vector of the dense layer during a matrix vector multiplication method.

A matrix-vector product form in general is:

$$W\mathbf{x} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix}$$

Where W is a (M x N) matrix and x are a (1x N) matrix. W matrix values are the previously trained parameters that can also be modified by backpropagation. The feedforward neural networks' most widely used algorithm for training is backpropagation. Mostly, by computing the weights of the network for a single input or output, backpropagation in a neural network calculates the gradient of the loss function. From the foregoing justification can be infered that the thick layer will produce an N-dimensional vector as its output. It can be observed that it is altering the vectors' dimension. In essence, every neuron in a dense layer is employed to change the vectors' dimension [30].

- **The ReLU Activation Function**

The activation function in a neural network is responsible for activating the node or output for that input from the weighted input that has been gathered from the node.

The rectified linear activation function, often known as ReLU, is a piecewise linear function that, if the input is positive, outputs the input right away; otherwise, it outputs zero. Because a model that utilizes it is simpler to train and typically achieves superior performance, it has evolved into the default activation function for many different types of neural networks.

When training deep neural networks, the ReLU enables the learning of complex relationships in the data using stochastic gradient descent and backpropagation of mistakes.

A rectified linear activation unit, or ReLU for short, is a device that carries out this activation function [31].

The ReLU activation function has rapidly become the default activation function when developing most types of neural networks because of the following advantages

1. Simplicity in computation.

The max () function is all that is necessary to implement the rectifier function.

2. Representational Sparsity is another.

The rectifier function's ability to provide a real zero value is a significant advantage.

This means that negative inputs can result in genuine zero values, enabling neural networks' hidden layers to incorporate one or more true zero values when activated. This is referred to as a sparse representation,

and it qualifies as such in representational learning because it can speed up learning and make the model simpler. [32].

- **BatchNormalization Layer.**

Deep neural networks can be difficult to train because, following weight adjustments, the input from earlier layers may change.

This may cause the learning process to chase after a moving target indefinitely. The precise term for this change in how inputs are distributed among layers of the network is "internal covariate shift".

Realizing a stable distribution of activation levels throughout training is the aim of batch normalization.

An elegant method for reparametrizing practically any deep network is batch normalization. The difficulty in coordinating updates across numerous levels is reduced through reparameterization.

This is accomplished by calibrating each input variable's activations per mini-batch, such as the activations of a node from a previous layer, and scaling the layer's output. Just to refresh your memory, standardization refers to rescaling data to have a mean of zero and a standard deviation of one, for example, a standard Gaussian [33].

Learning During the forward pass, or inference, the model takes an input a_i and computes its reformed output $\hat{\mathbf{a}}_i=h(\bar{a}_i)$. The optimizer parameter θ and loss function are learned via backpropagation. During the backward pass, θ was estimated by minimizing by the optimizer (Adam optimizer) and the CosineSimilarity (sim) loss function. Adam is one of the latest state-of-the-art optimization algorithms being used in deep learning. The first time normalized by the second time gives the trend of the update.

Adam optimizer [34]:

$$\theta_{n+1} = \theta_n - \frac{\alpha}{\sqrt{v_n+\epsilon}}\hat{m}_n \ldots(2.11)$$

Where $\hat{m}_n$ and v_n and α are the biased. CosineSimilarity loss function that Computes the cosine similarity between labels and predictions:

$$\text{sim}\ (\boldsymbol{x}, \boldsymbol{y}) = \frac{x \cdot y}{\|x\|\|y\|} \ldots(2.12)$$

Where $\|x\|$ is the Euclidean norm of vector x=(x1,x2,…,xp), defined as $\sqrt{x_1^2 + x_2^2 + \cdots + x_p^2}$. Conceptually, it is the length of the vector. Similarly, $\|y\|$ is the Euclidean norm of vector y The value of this loss function is between -1 and 1. The values closer to -1 indicate greater similarity, and the values closer to 1 indicate greater dissimilarity [35], [36].

2.5 Centrality measures

Centrality measures are a fundamental instrument for grasping graphs. These algorithms use graph theory to work out the significance of any given node in a graph. Network science research has focused on centrality metrics. They have been utilized in various networks, including social, communication, and contact networks in multiple disciplines. [37], [38].

Local centrality measurements, iterative centrality measures, and global centrality measures are three different types of centrality measures that can be used [39], [40].

Here is a look at some social network analysis measures, and how they work.

2.5.1 Measures of Local Centrality

Centrality measures that can be calculated using local information of the node.

2.5.1.1 Degree Centrality: the degree of the node or the edges number that incident on the node. In mathematics, the vertex v's degree is defined by:

$$C_{\deg}(v) = d(v) = |N(u)| \quad \text{....(2.13)}$$

Where N(v) is the group of node v's immediate neighbors [41], [42].

2.5.1.2 Semi-local Centrality: is characterized by:

$$C_{\text{s-local}}(v) = \sum_{u \in N(v)} Q(u) \quad \text{....(2.14)}$$

Where $Q(u) = \sum_{w \in N(u)} d_2(w)$, and $N(u)$ is the set of the nearest neighbors of node u and N(w) is the number of the nearest and the next nearest neighbors of node w [43], [44].

2.5.1.3 The Modified Local Centrality:

$$C_{\text{m-local}}(v) = \sum_{u \in N(v)} \sum_{w \in N(u)} d_2(w) - 2 \sum_{u \in N(v)} d(u) \quad \text{....(2.15)}$$

where N(u) is the group of a node's closest neighbors, and N(w) is the sum of its closest and second-closest neighbors of node w (Kirigin et al., 2022; Wan et al., 2021b).

2.5.1.4 Volume Centrality: determined the node's volume centrality for a given radius h as follows:

$$C_{\text{volume}}(v) = \sum_{u \in N_h(v)} d(u) \quad \text{....(2.16)}$$

Where h = 2 (Wan et al., 2021b).

2.5.1.5 Clustering Coefficient

$$\boldsymbol{C}_{\textbf{clustering}}\,(\boldsymbol{v}) = \frac{\mathbf{2}}{\boldsymbol{d}(\boldsymbol{v})\cdot(\boldsymbol{d}(\boldsymbol{v})-\mathbf{1})}\sum_{r,s\in N(v)} A_{rs} \quad \text{....(2.17)}$$

Where A is the adjacency matrix [45].

2.5.1.6 Square Clustering: Compute the squares clustering coefficient for nodes:

$$C_4(v) = \frac{\sum_{u=1}^{k_v}\sum_{w=u+1}^{k_v} q_v(u,w)}{\sum_{u=1}^{k_v}\sum_{w=u+1}^{k_v}[a_v(u,w)+q_v(u,w)]}, \quad \text{....(2.18)}$$

Where $qv(u,w)$ are the number of squares that u and w have common neighbors with besides v, and $av(u,w) = (ku - (1 + qv(u,w) + \theta uv))(kw - (1 + qv(u,w) + \theta uw))$, where θuw = 1 if u and w are linked, 0 otherwise. [37]

2.5.1.7 Local Entropy Measures: Entropy estimates the amount of information missing in a process in the context of information theory. Local entropy is the node's contribution to the overall entropy of the graph [37].

$$C_{\text{l-entropy}}\,(v) = -\sum_{u\in N(v)} d(u)\log\, d(u), \quad \text{....(2.19)}$$

.

2.5.1.8 Mapping Entropy Measures: It takes into account the node's neighbors in addition to the centrality discussed above[37].

$$C_{\text{mapping-entropy}}\,(v)\& = -d(v)\sum_{u\in N(v)} \log\, d(u) \quad \text{....(2.20)}$$

2.5.1.9 ClusterRank: is a local ranking method that considers the clustering coefficient along with the neighbors' influence and the number of neighbors[37], [46].

$$C_{\text{clusterrank}}\,(v) = f\big(C_{\text{clustering}}\,(v)\big)\sum_{u\in N^{\text{out}}\,(v)}\,(C\,(u)+1) \quad \text{....(2.21)}$$

Where $f(x) = 10^{-x}$

2.5.1.10 Average Neighbor Degree: determine each node's neighborhood's average degree [37], [47].

$$D_{ANdegree}(i) = \sum_{w \in N(v)} \frac{d(w)}{|N(v)|} \text{....(2.22)}$$

2.5.1.11 Node Density: The node density is determined based on quantities of edges and nodes in the subgraph generated by a h-hop forward BFS of this node. The node density is defined as:

$$C_{\text{nodedensity}}(v_i^h) = |E'|/(|V'|(|V'|-1)/2) \text{ ....(2.23)}$$

where i denotes the i-th node, h denotes the forward hop count from vi, V' denotes the group of nodes in the subgraph G' with h hops forward BFS from v_i, |V' | denotes the number of nodes in V', E' denotes the group of edges in the subgraph G', and |E' | denotes the number of edges in E' [48].

2.5.1.12 Laplacian Centrality: is an easy-to-calculate centrality metric that takes only linear time. It is described as the reduction in the graph's Laplacian energy, or the sum of squares of its eigenvalues, when the vertex is eliminated [37]. It is defined as:

$$C_v^L = (\Delta E)_v = d_G^2(v) + d_G(v) + 2\sum_{v_i \in N(v)} d_G(v_i) \text{...(2.24)}$$

2.5.1.13 Distinctiveness Centrality: is given more importance to nodes that are strongly connected to loosely connected peers, so that they make the graph periphery more reachable [37], [40].

$$D_{distinctiveness}(i) = \sum_{j=1}^{n} A_{ij} \log_{10} \frac{n-1}{d_j} \text{ ...(2.25)}$$
$$j \neq i$$

2.5.1.14 local clustering coefficient: computes the local clustering coefficient for each node in the graph [37], [40] :

$$LC(i) = \sum_{j \in N(i)} \frac{|N(i) \cap N(j)|}{d(i)(d(i)-1)} \ldots (2.26)$$

2.5.1.15 Neighborhood Connectivity: average connectivity of neighbors of given vertex [39].

$$NC(x) = \frac{\sum_{k \in N(x)} |N(k)|}{|N(x)|} \ldots (2.27)$$

2.5.1.16 Network Centrality: calculate a node's importance based on the number of edges it connects and the edges' clustering coefficients. For a node u, its NC(u) is defined as the sum of edge clustering coefficients of all edges directly connected with node u:

$$\begin{aligned} NC(u) &= \sum_{v \in N_u} \text{ECC}\,(u, v) \\ &= \sum_{v \in N_u} \frac{z_{u,v}}{min(d_u - 1, d_v - 1)} \ldots\ldots (2.28) \end{aligned}$$

Where d_u and dv are the radii of nodes u and v, respectively, and $z_{u,v}$ is the number of triangles that include the edge that is actually present in the graph. The number of triangles in which the edge Eu,v can theoretically participate at most is the meaning of min(du-1, dv-1)[37], [40].

2.5.1.17 Leverage Centrality: is based on the idea that a node in a network is central if its immediate neighbors depend on that node for information and takes into account a node's degree in relation to its neighbors [37].

$$l(i) = \frac{1}{d(i)} \sum_{N_i} \frac{d(i) - d(j)}{d(i) + d(j)} \ldots (2.29)$$

2.5.1.18 Neighbor Based Centrality: It is conceivable that information flows between two connected nodes in a graph. For example, node u receives some information from node v next to it, and node v also receives some information from node w next to it. As a result, the overall quantity of information that node u receives depends on both its first layer neighbors, which are all of its neighbors, as well as all of the nodes that are its first layer neighbors' adjacent neighbors but not the first layer neighbors themselves. These neighboring neighbors of node u's first layer neighbors are referred to as its second layer neighbors. As a result, the quantity of information that node u receives depends on more than just its first layer neighbor's node's degree. Then, the more information a node (or group of nodes) receives, the more significant the node (or group of nodes) will be. It should be noted that a node's total amount of information received depends on both its nearby nodes and those nodes that have the same adjacent nodes as it. The definition of neighbor-based centrality of node v, abbreviated as CNv, is:

$$C_v^N = \begin{cases} d(v), & \text{if } N_1(v) = \emptyset \\ d(v)\sum_{u\in N_1(v)} d(u), & \text{if } N_1(v) \neq \emptyset, N_2(v) = \emptyset \\ d(v)\sum_{u\in N_1(v)} D(u), & \text{if } N_1(v) \neq \emptyset, N_2(v) \neq \emptyset \end{cases} \quad \text{...(2.30)}$$

$$D(u) = \begin{cases} d(u), & \text{if } N_1(u) \cap N_2(v) = \emptyset; \\ d(u)\left(\sum_{w\in N_1(u)\cap N_2(v)} d(w)\right)^{\frac{1}{2}}, & \text{if } N_1(u) \cap N_2(v) \neq \emptyset. \end{cases}$$

Where N1(v) is a collection of the first layer neighbors of node v and N2(v) is a set of the second layer neighbors of nodes in N1(v), and d(j) signifies the degree of node j [37], [40].

2.5.2 Measures of Iterative Centrality

Iterative operations are used to determine each node's centrality value in iterative centrality measures. The majority of states base the number of iterations on how quickly values change at each node. These repeated actions at the node incorporate global data into the metric.

2.5.2.1 Eigenvalue Centrality: Eigenvector centrality computes the centrality for a node based on the centrality of its neighbors. The eigenvector centrality for node i is:

$$Ax = \lambda x \quad ...(2.31)$$

where A is graph G's adjacency matrix with eigenvalue λ. The Perron-Fresenius theorem states that if λ is the biggest eigenvalue of the adjacency matrix A, then there exists a singular solution x, all of whose elements are positive [37], [47]**.**

2.5.2.2 Katz Centrality: It is based on the centrality of its neighbors, Katz centrality calculates a node's centrality. It expands on the idea of eigenvector centrality. The node i's Katz centrality is:

$$x_i = \alpha \sum_j A_{ij} x_j + \beta \quad \ldots(2.32)$$

Where A is the graph G's adjacency matrix with eigenvalues.

The parameter regulates the starting centrality and [37]

$$\alpha < \frac{1}{\lambda_{max}}$$

2.5.2.3 PageRank: Katz centrality is developed to PageRank and was created by Google's co-founders Brin and Page. By counting links to a website, PageRank determines how important a website is [37], [47]:

$$C_{\text{pagerank}}(v, \alpha, \beta) = \alpha \sum_{u \in \mathcal{V}, u \neq v} A_{uv} \frac{C_{\text{pagerank}}(u, \alpha, \beta)}{max(C_{\text{out-deg}}(u), 1)} + \beta \quad \ldots(2.33)$$

2.5.2.4 Diffusion Centrality: the node's own and its neighbors' combined contribution score:

$$C_{DD}(v) = \lambda_v * d(v) + \sum_{i \in N(v)} \lambda_i * d(i) \quad \ldots(2.34)$$

Where λ_v propagation probability [37], [47] .

2.5.2.5 Subgraph Centrality: Measured by the weighted sum of the closed pathways incident to v in the graph, including both cyclic and acyclic paths (i.e., a path that circles back on itself), subgraph centrality shows that as the length of a path increases, so does its contribution to the sum. Providing subparagraph centrality is:

$$C_{\text{subgraph}}(v) = \sum_{k=0}^{\infty} \frac{\mu_k(v)}{k!} = \sum_{j=1}^{N} \left(u_j^v\right)^2 e^{\lambda_j} \quad \ldots(2.35)$$

where $\mu_k(v) = (\mathbf{A}^k)_{vv}$, λ_j is A's jth eigenvalue, and u_j is the eigenvector that goes with it (u^v_j is the vth element of this vector)[37], [47] .

2.5.3 Measures of global centrality.

Measuring indicators for global centrality is taken into account the entire network topology. These techniques take into account the size of path lengths between discrete (non-adjacent) graph nodes. Most of these metrics require more expensive computations.

2.5.3.1 Betweenness Centrality: It is formed from the discovery that certain nodes, depending on their position in the graph, had control over communication between a pair of other nodes. A node that has control over this communication is in a stronger position to affect others by acting as a broker or facilitator. Offered by:

$$C_{\text{bet}}(v) = \sum_{s,t | s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \quad \ldots(2.36)$$

where σ_{st} is the quantity of pathways between s and t that are shortest, andσ_{st} (v) is the number of paths between s and t that are shortest that include v [37], [49] .

2.5.3.2 Flow Betweenness Centrality: It uses an electrical current model for information spreading in contrast to betweenness centrality which uses shortest paths and known as random-walk betweenness centrality:

$$C_{\text{flow-bet}}(v) = \sum_{s,t|s\neq v\neq t} m_{st}(v) \ldots(2.37)$$

By substituting each summation $m_{st}(v)$ with $\frac{m_{St}(v)}{m_{St}}$., this formula can be made normal. As a graph centrality metric, this metric can be used to calculate the mean difference between the node with the highest centrality and the nodes' other centralities[37], [49].

2.5.3.3 Load Centrality: After each node in the graph sends one packet to each other node following the shortest path, the total number of data packets passing over node v equals the load centrality of node v. When there are several shortest paths between two nodes, the amount is uniformly split at each branching point:

$$C_{\text{load}}(v) = \sum_{s,t|s\neq v\neq t} \theta_{st}(v) \ldots(2.38)$$

Where $\theta_{st}(v)$ is the portion of the unit quantity that traveled through node v on the shortest path from node s to node t provided that the quantity is split evenly at each branch encountered [37], [47].

2.5.3.4 Closeness Centrality: It is the inverse proportion of the average distance between each node in the network, or the reciprocal of farness. Formally, define by:

$$C_{\text{closeness}}(v) = \frac{n-1}{\sum_{u\in V} d(v,u)} \ldots(2.39)$$

Where $d(v,u)$ is the distance between nodes v and u [37], [47].

2.5.3.5 Information Centrality: It is drawn from the standpoint of statistical estimation that as distance between nodes increases, information is lost due to noise from signal transmission that is captured by the variance of the signal flowing via a path. The information for node v is thus defined as the harmonic mean of the information between v and every other node, that is, by treating this variance as unity for each link:

$$C_{\text{information}}(v) = \frac{n}{\sum_{u \in V} \frac{1}{I_{uv}}} \ldots (2.40)$$

Where I_{uv} weighing the information along each path from u to v according to its length [37], [47].

2.5.3.6 Residual Closeness: Using a different definition of proximity, ascertain the graph's susceptibility to disconnection due to the removal of a few nodes or edges, defined by:

$$C_{\text{residual-closeness}}(v) = \sum_{u \neq v} \left(\frac{1}{2}\right)^{d(v,u)} \quad \ldots (2.41)$$

This metric is called also decay centrality, expressed as $C_{\text{decay}}(v) = \sum_{u \neq v} \delta^{d(v,u)}$, where $0 < \delta < 1$ is the decay rate [37], [47].

2.5.3.7 Eccentricity Centrality: It is the farthest space between node n and another node in the graph. Higher centrality is indicated by lower eccentricity [37], [47]. mathematically define as:

$$C_{\text{eccentricity}}(v) = \frac{1}{max\{d(v,u) | u \in V\}} \quad \ldots (2.42)$$

2.5.3.8 Harmonic Centrality: of a node u is the reciprocal of the sum of the distances traveled by all other nodes to get to u [37], [47].

$$C_{\text{harmonic}}(u) = \sum_{v \neq u} \frac{1}{d(v,u)} \quad \ldots (2.43)$$

2.5.3.9 Average Distance: of node u to the other nodes in the graph[37], [47].

$$\mathrm{Cu} = \frac{\Sigma_{\mathrm{W}\in\mathrm{V}}\mathrm{d(u,w)}}{\mathrm{n}-1} \quad \ldots(2.44)$$

2.5.3.10 Barycenter Centrality: It is the reverse of the total distance between vertex v and all other vertices [37], [40]:

$$C_{\text{barycenter}}(v) = \frac{1}{\sum_{u\in V} d(v,u)} \quad \ldots(2.45)$$

2.5.3.11 Radiality centrality: In contrast to its diameter, the radiality, a node centrality index, would assign high centrality to vertices that are close to every other node in their reachable region.

$$C_{rad}(v) = \frac{\sum_{w\in V}(d+1-d(v,w))}{n-1} \ldots(2.46)$$

where d(v,w) is the distance between nodes v and w and d is the diameter of graph G with n nodes [37], [50].

2.5.3.12 Harary Graph Centrality: Graph centrality measures the length of the shortest path to reach the node farthest away from the given node [37], [50].

$$C_{harary}(v) = 1/\left(\max_{\mathrm{u}} \mathrm{d}(v,\mathrm{u})\right) \ldots(2.47)$$

2.5.3.13 Heatmap Centrality: It utilizes both local and global network information by comparing the fairness of each node with the average sum of the fairness of its neighbor nodes [40], [51]. In particular, the heatmap centrality for node v, denoted as C_{HM} (v) is defined formally as:

$$C_{HM}(v) = \sum_{u=1}^{N} \mathrm{d}(v,\mathrm{u}) - \frac{\sum_{u=1}^{N} A_{v,u}\cdot\sum_{k=1}^{N} d(v,k)}{\sum_{u=1}^{N} A_{v,u}} \ldots(2.48)$$

2.5.3.14 Approximate Current Flow Betweenness Centrality: It refers to high probability approximation of the current-flow betweenness centrality within the epsilon's absolute error:

$$\tau(v) = \frac{1}{2}(-|b(v)| + \sum_{e:v\in e} |x(\vec{e})|) \ldots(2.49)$$

Where the term −|b(v)| accounts for the fact that only inner vertices are considered in the definition of shortest-path betweenness centrality, $e \in E$ and $|x(\vec{e})|$ denote the throughput in case of an st-current [49], [52].

2.5.3.15 Bridgeness-Coefficient (BrCoe): the bridgeness coefficient of node i is the reciprocal of sums of the distance from it to all other nodes excluding its neighbors and indirectly adjacent nodes (d(i,k)≠1, 2) [40], [47]:

$$\mathrm{BrCoe}(i) = \frac{1}{\sum_{k=1}^{n} \mathrm{d(i,k)}} \ldots(2.50)$$

2.5.3.16 Wiener Index: A node centrality index is the Wiener index. The total distances (shortest pathways) between a node v and every other node in the network make up the Wiener index. In essence, it is comparable to "closeness," but in this case, the score means the opposite thing.[40], [47], as the reciprocal is not calculated:

$$W_{index}(v) := \sum_{w\in V} \mathrm{dist}\,(v, w) \ldots(2.51)$$

The 39 centrality measurements above (listed in table (2.1)) was assigned as features for the nodes of the graph.

Table (2.1): The Centrality Measurement

Centrality Measurement		
Local Centrality		
Degree Centrality	Local Entropy Measures	Distinctiveness Centrality
Semi-local Centrality	Mapping Entropy Measures	local clustering coefficient
Modified Local Centrality	ClusterRank	Neighborhood Connectivity
Volume Centrality	Average Neighbor Degree	Network Centrality
Clustering Coefficient	Node Density	Leverage Centrality
Square Clustering	Laplacian Centrality	Neighbor Based Centrality
Global centrality		
Betweenness Centrality	Harmonic Centrality	Heatmap Centrality
Flow Betweenness Centrality	Average Distance	Load Centrality
Harary Graph Centrality	Barycenter Centrality	Bridgeless-Coefficient
Closeness Centrality	Radiality centrality	Eccentricity Centrality
Information Centrality	Wiener Index	Approximate Current Flow Betweenness Centrality
Residual Closeness		
Iterative Centrality		
Eigenvalue Centrality	PageRank	Subgraph Centrality
Katz	Diffusion Centrality	

2.6 Feature Selection

The collected data is usually associated with a high level of noise. Restricted technologies that collected the data and the source of the data itself are two main reasons causing noise in these data.

The development of social media gave an opportunity for the users of it to be content creators besides being traditional consumers. And due to the difficulty in the supervision of the content, this led to the appearance of poor content quality such as spam or abused content by nature, data are informally written and suffer from grammatical mistakes, misspelling, and improper punctuation [53], [54].

With the increasing the number of features, the dataset becomes larger. The process of limiting the amount of input variables to those thought to be most helpful to a model's ability to predict the target variable is known as feature selection. Reducing the number of inputs is useful to both reduce the computational cost of modeling and, in some cases, to improve the performance of the model. In a high-dimensional dataset, there are some entirely irrelevant and unimportant features. It has been seen that the contribution of these types of features is often less towards predictive modeling as compared to the critical features, which cause: Consumption of resources and time. Negative effect on the prediction results due to noise embedded within it [55].

Feature selection can be done in two ways: supervised and unsupervised. The supervised technique entails choosing the input features with the strongest correlation to the target variable after statistically analyzing the relationships between each input feature and the target variable[56].

the unsupervised feature selection is able to identify and select relevant features without needing class label information, where is usually the features

are represented in the form of criterion functions for defining their importance [53], [54].

There are unsupervised feature selection Information-based methods such as the Correlation-based method and the Variance Threshold-based method.

2.6.1 The Correlation-based feature selection (CFS): Correlation is a statistical term that in common usage represents how similar two variables are to having a linear relationship with each other. This method is a filter approach and thence autonomous of the final classification model. It assesses feature subsets just based on the information intrinsic properties, as the name already suggests: correlations.

The aim is to discover a feature subset with low feature-feature correlation, to avoid repetition to preserve or grow the predictive force. Pearson Correlation measures the strength of the correlations between the various variables by the following formula:

$$\mathbf{r} = \frac{\mathbf{n}(\sum \mathbf{xy}) - (\sum \mathbf{x})(\sum \mathbf{y})}{\sqrt{[\mathbf{n}\sum \mathbf{x^2} - (\sum \mathbf{x})^2][\mathbf{n}\sum \mathbf{y^2} - (\sum \mathbf{y})^2]}} \quad \dots (2.52)$$

Where r is Pearson Coefficient, n is number of the pairs of the stock, ∑xy is sum of products of the paired stocks, ∑x is sum of the x scores, ∑y is sum of the y scores, ∑x2 is sum of the squared x scores, ∑y2 is sum of the squared y scores. The range of r is between 1 and -1, and if the value is positive, the features increase and decrease together, unlike if they are negative [55], [57].

2.6.2 The Variance Threshold-based feature selection (VTFS):

This method is a quick way of removing features with very low variance, i.e., features with not much helpful information. variance, as the name hints, displays the variability in a distribution in a single metric. It shows how to spread out the distribution and shows the average squared distance from the mean:

$$\sigma = \frac{1}{n}\sum_{i=1}^{n} (x_i - \bar{x})^2 \quad \ldots(2.53)$$

Where $\bar{x}$ is the mean and x_i is the value. If σ is less than a value that is specified as a threshold value, its feature is deleted. If Variance Threshold = 0 (Remove Constant Features). If Variance Threshold > 0 (Remove Quasi-Constant Features) [55].

2.7 Performance Measurement

Community structures are significant for understanding not only the nature of the network but also how the network works. many ways have been proposed to discover societal structures from different perspectives. Because of the absence of agreement on the meaning of a community, figuring out measurements for assessing the nature of a community is likewise a difficult task[58] .

If the actual (ground-truth) community structure of the graph is known beforehand, evaluating the communities that an algorithm has discovered from it becomes easier. So, synthetic benchmark graphs with inherent community structure have been construct. Where the communities detected by community detection algorithms are evaluated compared to the communities generated by synthetic benchmark graphs. Their performance is determined by calculating the similarity between computed labels and the labels of the synthetic benchmark graphs [59], [60] .

2.7.1 Synthetic Benchmark Graphs

Analyzing a graph with a clear community structure is generally what testing community detection methods entails. Ideally, a perfect comparison with real networks that have communities is well known but not many examples are available, consequently, computer-generated graphs with a built-in community architecture are used for the most thorough assessments.

- **Girvan and Newman (GN):** the most renowned benchmark for community detection is the graphs introduced by Girvan and Newman (GN) [3]. Each graph has 128 nodes, divided into four groups with 32 nodes each. The average degree of the graph is 16 and the nodes have approximately the same degree. There are several caveats that must be considered for these graphs: the graph's nodes are all of the same degrees; The size of each community is the same; the graph is not large [61].
- **Lancichinetti–Fortunato–Radicchibenchmark (LFR):** is an algorithm that produces benchmark graphs (artificial graphs that like real-world graphs). They have a priori known communities and are utilized to test various community detection methods This benchmark has the benefit of taking the variation in the distributions of node degrees and community sizes into account [62].

  The LFR graph is generated by several parameters, including a number of total nodes n, minimal degree k_{min}, maximal degree k_{max}, minimal community sizes s_{min} and maximal community size s_{max}, exponent of power-law distributions of nodes degree γ and Power law exponent for the community size distribution β, respectively, seed indicator of random

number generation state, and a mixing ratio of external links μ. Its formula is written below

LFR_benchmark_graph(n, γ, β, μ, k_{min}, k_{max}, s_{min}, s_{max}, seed) … (2.54)

2.7.2 Metrics

A few measurements from the literature on clustering in data mining and classification were reinterpreted using the graph data to bring the detected and real-world community structures into an agreement. These measurements are based on contrasting the graph community in the real world with the community produced by the algorithm.

- **Normalized Mutual Information (NMI).**

NMI is a scale utilized to assess graph dividing performed by community detecting algorithms. It is predominately viewed because of its thorough meaning and permitting the comparison of two partitions even when a different number of clusters. NMI is a variant of a popular measure in information theory called Mutual Information.

The clustering quality of community detection algorithms is much of the time tried utilizing a normalized measure of Mutual Information [60], [63].

$$NMI(Y,C) = \frac{2\times I(Y;C)}{H(Y)+H(C)} \quad \ldots(2.55)$$

Where I is the Mutual Information and H(Y) is the entropy of the labeled and H(C) is the community groups.

$$H(X) = -\sum_{x\in\mathcal{X}} P(x)\log P(x). \qquad \ldots(2.56)$$

Where P(X) is probability distribution.

$$I(X;Y) = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P(x,y)\log\frac{P(x,y)}{P(x)P(y)}. \qquad (2.57)$$

This measure returns the value in the range 0 (no mutual information) and 1 (perfect correlation).

- **Classification measurement metrics**

- **The confusion matrix**

In the field of classification, a confusion matrix, also known as an error matrix, is a special table that helps imagine the performance of an algorithm. Every row of the matrix explains the samples in an actual class while each column explains the samples in a predicted class [64],as showing in figure (2.3).

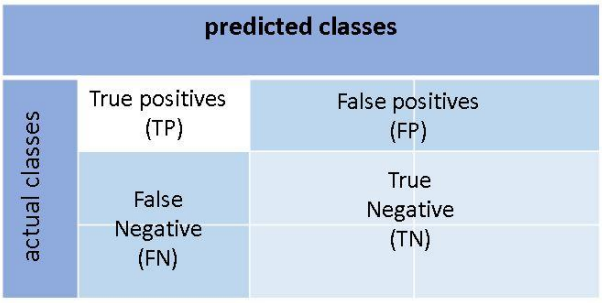


Figure (2.3): the confusion matrix

- **Accuracy score (AC):** is a metric for evaluating classification models. Informally, accuracy is the portion of the prediction our model got right[64].

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+FP+FN+TN}} \ldots (2.58)$$

Chapter Three

Community Detection Framework using Graph Autoencoder (CDGAE)

Chapter three

Community Detection Framework using Graph Autoencoder (CDGAE)

3.1 Introduction

Recently, the use of social networks by people has increased, and the data circulated in them has increased. The aim of our research is to divide these networks into communities to facilitate an understanding of their functions.

In this thesis, GAE has been used to convert the data into a low representation level, from which graph data can be extracted.

In this chapter, the practical side of the thesis is explained in (3.2) the general structure of the Framework is explained, which includes how to collect datasets (3.2.1) and features initialization in (3.2.2), Either in (3.2.3) the general design of the Community Detection Framework using Graph Autoencoder (CDGAE) is clarified and in the (3.2.4) the implementation of the suggested framework is showed.

3.2 Framework Architecture

The Community Detection Framework using Graph Autoencoder (CDGAE) aims to divide the nodes of the social network into communities, that nodes in the same community are highly interconnected together and less interconnected with nodes of another community.

The proposed Framework consists of four stages that were prepared and executed to finish the mission of community detection utilizing the Graph Autoencoder system. The first stage is for data collection required for the Framework. The second stage consists of two steps: the first one is to extract

the node features by the centrality-based approach, and the second is to apply the feature selection to these features and the selected features are assigned as a node's features. The third stage is the revelation of how the Graph Autoencoder is applied for community detection. The fourth step is the framework evaluation (see figure 3.1).

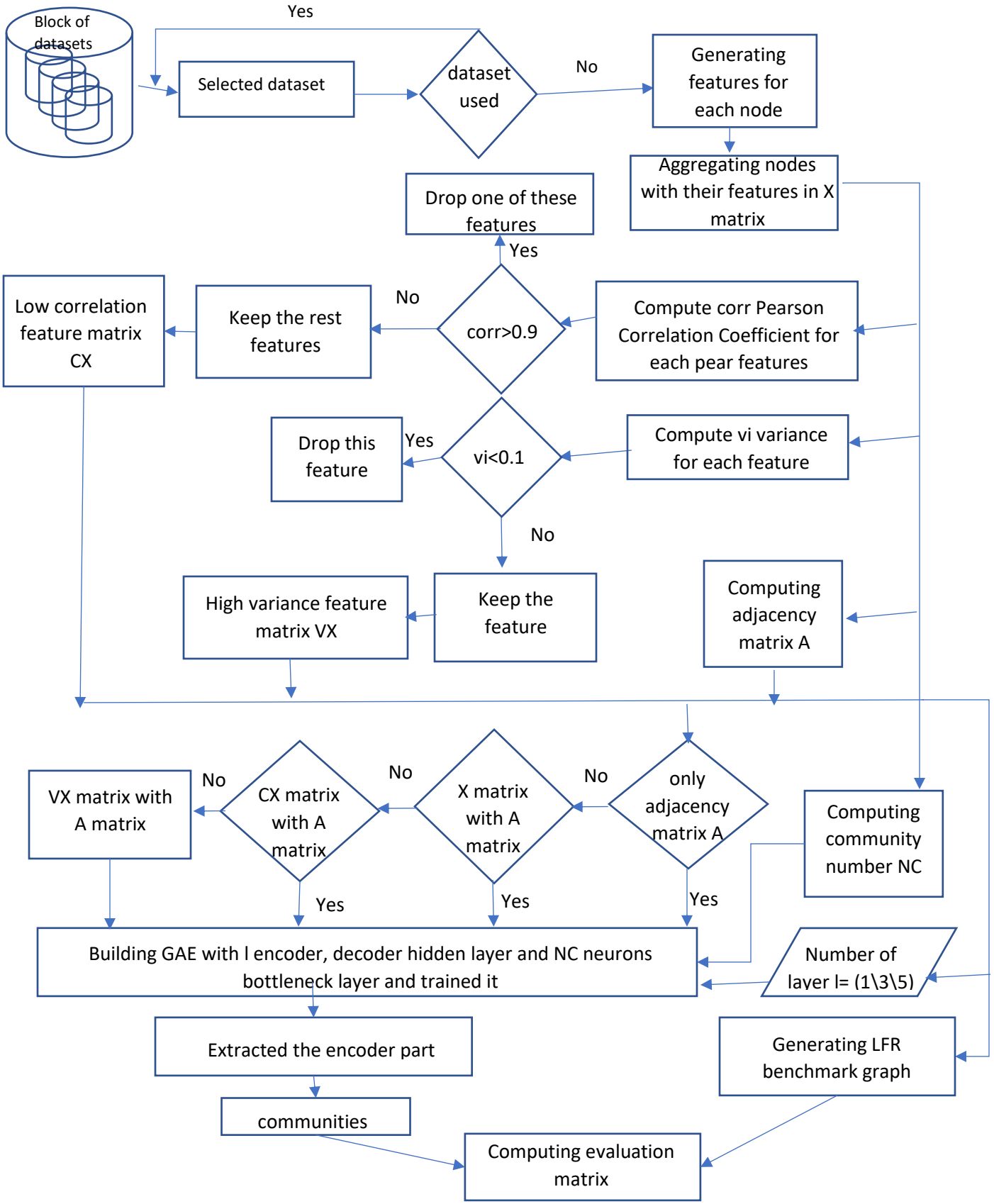


Figure (3.1): CDFGAE block diagram

3.2.1. The CDGAE Algorithm.

The task of the algorithm is to find the number of communities by spectral method with the Bethe Hessian matrix, and then assign the nodes to their communities depending on the graph structure and the features of the nodes by the GAE. The CDGAE Method is fully described in algorithm (3.1).

This algorithm finds the number of nodes and edges, (line 1-4).

Then the nodes features are extracted based on the centrality measurement according to equations (2.13) to (2.51), (line 5-9).

Then the community number is computed according to equations (2.1) and (2.2) of Bethe Hessian, (line 10-12).

The two feature selection methods are applied to the feature matrix, (line 13-16). And adjacency matrix is computed (line 17-19).

The Threshold1 and Threshold2 determine the input of GAE, Whether concatenate (A, CX), concatenate (A, VX), concatenate (A, X), or A matrix only.

The variable L determine the number of hidden encode and decoder layers (1, or 3, or 5). Then call the GAE procedure for community detection, (line 20-37).

Then the framework is evaluated according to equations (2.55), (2.58), (2.61), (line 38).

Algorithm (3.1): CDGAE

Input: graph of datasets, number of layers L

Output: best relationship and number of communities

Initialization:

Threshold1=0.9,

Threshold2=0.01

// **Collection dataset**

For each graph

  Find number of nodes N_n

  Find number of edges

End for

//find feature based on centrality (local, global and iterative)

For i=1 to N_n

  X[i]=**Call** local centrality (eq 2.13) to (eq 2.30)

  X[i]= X[i]+**Call** iterative centrality (eq 2.31) to (eq 2.35)

  X[i]= X[i]+**Call** global centrality (eq 2.36) to (eq 2.51)

End for

//**compute number of communities based on Bethe Hessian matrix**

For j=1 to Nu

  NC=**Call** BHM (eq 2.1) and (2,2)

End for

//**find the important features**

For j=1 to T_{nf} // T_{nf} = total number of features

  CFS[j]=**Call** Correlation(X, Threshold1)

  VTFS[j]= **Call** variance (X, Threshold2)

End for

```
//Compute adjacency matrix
17    For i =1 to N_n
18        A[i]= Call AdjM
19    End for
// build graph autoencoder (GAE)
20    For i=1 to N_n
21        For j=1 to T_nf
22            If CFS[j] ≤ Threshold1
23                Call GAE (concatenate (CFS, A)), L, NC)
24            End if
25            Else
26            If VTFS[j] ≥ Threshold2
27                Call GAE (concatenate (VTFS, A)), NC)
28            End if
29            Else
30            If (CFS[j] ≥ Threshold1) and ( VTFS[j] ≤ Threshold2)
31                Call GAE (concatenate (X, A)), L, NC)
32            End if
33            Else
34                Call GAE (A, L, NC)
35            End else
36        End for
37    End if
      // evaluation
38    Call Evaluation measures (NMI, and  Accuracy )
39  End CDGAE
```

3.3. Framework desgin

3.3.1. Data Collection

The data used in the proposed Framework is explicit data available in datasets as explained below:

- **Graph attribute:** the data for the Framework are the nodes and the connections between them (edges) that be in the form of the edge list.
- **Nodes Attributes:** because there are no features in the observed datasets, they are got by the feature initialization method.

3.3.2. Feature Initialization

3.3.2.1. The Centrality-based measurements

The Centrality-based measurements are utilized that represented in table (2.1), for the feature initialization, these centrality measures are calculated for each node and assigned as their features.

3.3.2.2. Feature Selection.

Feature selection methods are used because they can reduce model complexity, improve learning efficiency, and can also raise predictive force by decreasing redundancy.

the Correlation-based feature selection is used to discover a feature subset with low feature correlation to eliminate redundancy data. And the Variance Threshold-based feature selection is used because it removes padding data.

- **The Correlation-based feature selection (CFS)**

In this method, the subset is searched with a high feature-feature correlation greater than 90% percentage, and droped it while keeping one feature on behalf of the rest of this subset, as summarized in procedure (3.1).

procedure (3.1): Correlation (x, Threshold1)

```
1       corr=[]
2       Feature=[]
// computing the correlation matrix corr
3       For i=1 to Tnf // Tnf number of feature
4           Feature[i]=True    // saving all feature
5           For j=1 to Tnf
6               Sx=0 , x2=0
7               Sy=0 , y2=0 ,
8               xy=0
9               For z=1 to Nn // Nn number of nodes
10                  Sx=Sx+x[i][z]            //summation of feature x[i]
11                  x2= x2+x[i][z]^2
12                  Sy= Sy+x[j][z]
13                  y2=y2+x[j][z]^2
14                  xy=xy+x[i][z]*x[j][z]
15              End for
16              corr[i][j]= ((Nn* xy) – (Sx*Sy)) / squrt(( Nn *x2-Sx^2 )*( Nn *y2-
                Sy^2))
17          End for
18      End for
19      CFS=x
// computing the correlation feature selection matrix CFS
20      For i =1 to Tnf
21          For j=i+1 to Tnf
22              If corr[i][j] ≥ Threshold1
23                  If feature[j]
```

```
24                Feature[j] = False  //removing j feature
25                Drop CFS[j]
26            End if
27        End if
28    End for
29  End for
  End Correlation
```

In this procedure, Pearson-Correlation is computed according to equation (2.52) for each pair features and save them in two-dimension corr matrix, (line 3-18).

Then Pearson-Correlation for each two features is tested, if it greater than it 0.9, one of these tow is dropped, (line 20-29).

- **The Variance Threshold-based feature selection (VTFS):**

To increase the effectiveness of any model, features are needed with high contrast. In this method, opposite the previous method, the feature is compared with its own values and removed all constant or low-variance features that are of no great use in modeling, as summarized in procedure (3.2).

In this procedure, variance from feature values is computed according to equation (2.53), (line 1-10).

If variance of each feature is less than 0.01 then this feature is dropped, (line 11-15).

procedure (3.2): variance (x , Threshold2)

// **computing the variance of each feature**

For i=1 to T_{nf} // T_{nf} = total number of features

  $X_{ma}[i]=sum(x[i]) /N_n$ // N_n number of nodes

End for

For i =1 to T_{nf}

  VTS[i]=0

  For j =1 to N_n

    VTS[i]= VTS[i] +(x[j] - $x_{ma}[i]$)^2

  End for

  VTS[i]= VTS[i]/N_n

End for

// computing the variance threshold feature selection matrix VTS

For i=1 to T_{nf}

  If (VTS[i] $\leq$ Threshold2)

    Drop VTS[i]

  End if

End for

End variance

After applying the above two feature selection algorithms to the feature matrix extracted from section (3.3.2.1), there will have three node features matrices: X, CX, and VX that are used separately as inputs to our Framework with the adjacency matrix, in addition of using the adjacency matrix alone as the fourth input.

3.3.3. GAE Design

The input of Framework is augmented adjacency matrix $\bar{A} \in R^{\wedge(N\times(N+F))}$ that consists of concatenate (A, X) adjacency matrix and feature matrix that there have three various features matrix: two features matrix extracted from two Feature Selection methods (CX, VX) and the original feature matrix (X), the Framework was tested also with the adjacency matrix only, without the feature matrix, so there will have Four Frameworks different in input (A, concatenate (A, X), concatenate (A, CX), concatenate (A, VX)). Since the square and symmetric adjacency matrix has been concatenated with the feature matrix in the augmented adjacency matrix, then the augmented adjacency matrix is no longer square and symmetric.

CDGAE Framework architecture includes a collection of non-linear transformations on input $\bar{a}_i$ that are divided down into two parts: encoder $f(\bar{a}_i)$: $\mathbb{R}^{N+F} \rightarrow \mathbb{R}^{NC}$, and decoder $g(\mathbf{z}_i)$: $\mathbb{R}^{NC} \rightarrow \mathbb{R}^{N+F}$ [29]. In order to obtain a NC-dimensional low-level representation of the ith node $\mathbf{z}_i \in \mathbb{R}^{Nc}$, (1, 3 or 5 layers) of the encoder component were stacked, and then stack (1 or 3 or 5) layers of the decoder part to reach an approximate reformed output $\hat{\mathbf{a}}_i \in \mathbb{R}^{N+F}$, resulting in a (2 or 6 or 10)-layer GAE architecture, to observe the effect of the number of layers on the performance of GAE.

Parameters that define the Framework architecture are referred to as hyperparameters. hyperparameters are not Framework parameters, so they are learned during training when they optimize a loss function. The key hyperparameter is the dimensionality of the hidden layers.

In CDGAE Framework, the number of neurons for the input layer equals the input, which is the number of graph nodes in addition to the number of features $\bar{A}$, which is the same as the number of outputs from the last decoder

layer. The neurons of the bottleneck layer are equal to the number communities NC computed by spectral method with the Bethe Hessian matrix H (Equation (2.1)) with a regularize value r_c (Equation (2.2)). The layers between the input layer and the bottleneck layer are shrunk in half, and the layers between the bottleneck layer and the output layer are each doubled.

After the Framework design was finished, it was trained on the dataset, where it was splatted into 70% for training data and the rest is the test data.

The training is for 300 epochs, where an epoch means training the neural network Framework with all the training data for one epoch. In an epoch, Forward and backward passes both counts as one pass.

the Framework trained during these epochs using the CosineSimilarity loss function and the Adam optimizer function, during which the loss decreases for the training data.

After training the Framework, the trained encoder was disconnected from the original Framework and input the dataset from it to predict the communities for each node.

The procedure (3.3) described the GAE work for community detection.

procedure (3.3): GAE (Ā, L, NC)

```
neuron=Ā.column
ź =  Ā.column
//creating the Nu array of  neurons  numbers for encoder & decoder networks
For i=1 to L      // L = number of layers
    Nu[i]=(neuron /2^j) //computing number of neurons in each layer
End for
// Creating the encoder network
For i=1 to L
    ź=σ(Nu[i] (ź))      // σ= ReLU function
End for
// Creating the bottleneck layer
bottleneck= σ (NC (ź))  //NC= no. of communities
encoder = bottleneck
a= bottleneck
// Creating the encoder network
For j=1 to L do
    a= σ (Nu[(L+1)-i]  (a))
End for
decoder = SoftMax (neuron (a))
// Creating the GAE
GAE= model (encoder, decoder)
Train, test = split(Ā,70)
// Training the GAE
For epoch=1 to 300
    GAE. fit (Train, test)
End for
```

// extracting the encoder for community detection

21 E=encoder

22 **For** i=1 to N_n

23 | community[i]=argmax(E.pridect(Ā[i]))// determine nodes communities

25 **End** for

End GAE

In this procedure, the number of neurons for each hidden layer is computing and saved in Nu matrix, (line 3-5).

The layers of encoder part are built with the neurons number saved in Nu matrix ends by the bottleneck layer with the number of neurons equal to the number of communities, (line 6-11).

The layers of decoder part are built with the neurons number in inverse Nu matrix, (line 12-15).

The GAE is built and trained it for 300 epochs, (line 16-20).

The decoder is extracted and used it to predict the nodes membership to any community, (line 21-25).

12 CDGAE Frameworks was built with the characteristics mentioned in table (3.1)

Table (3.1): The 12 CDGAE Frameworks with their characteristics that used in this thesis

CDGAE Framework	Framework characteristics
FL1L_Framework	**Input:** Adjacency matrix only A. **Layers:** consist of 1-encoder layer and 1-decoder layer. **Bottleneck neurons:** equal to the number of communities.
FL3L_Framework	**Input:** Adjacency matrix only A. **Layers:** consist of 3-encoder layer and 3-decoder layer. **Bottleneck neurons:** equal to the number of communities.
FL5L_Framework	**Input:** Adjacency matrix only A. **Layers:** consist of 5-encoder layer and 5-decoder layer. **Bottleneck neurons:** equal to the number of communities.
AF1L_Framework	**Input:** concatenate (A, X) **Layers:** consist of 1-encoder layer and 1-decoder layer. **Bottleneck neurons:** equal to the number of communities.
AF3L_Framework	**Input:** concatenate (A, X) **Layers:** consist of 3-encoder layers and 3-decoder layers. **Bottleneck neurons:** equal to the number of communities.
AF5L_Framework	**Input:** concatenate (A, X) **Layers:** consist of 5-encoder layers and 5-decoder layers. **Bottleneck neurons:** equal to the number of communities.
VTF1L_Framework	**Input:** concatenate (A, VX) **Layers:** consist of 1-encoder layer and 1-decoder layer. **Bottleneck neurons:** equal to the number of communities.
	Input: concatenate (A, VX).

VTF3L_Framework	**Layers:** consist of 3-encoder layers and 3-decoder layers. **Bottleneck neurons:** equal to the number of communities.
VTF5L_Framework	**Input:** concatenate (A, VX). **Layers:** consist of 5-encoder layers and 5-decoder layers. **Bottleneck neurons:** equal to the number of communities.
CF1L_Framework	**Input:** concatenate (A, CX). **Layers:** consist of 1-encoder layer and 1-decoder layer. Bottleneck neurons: equal to the number of communities.
CF3L_Framework	**Input:** concatenate (A, CX). **Layers:** consist of 3-encoder layers and 3-decoder layers. **Bottleneck neurons:** equal to the number of communities.
CF5L_Framework	**Input:** concatenate (A, CX). **Layers:** consist of 5-encoder layers and 5-decoder layers. **Bottleneck neurons:** equal to the number of communities.

Chapter Four
Experimental Results and Discussion

Chapter Four

Experimental Results and Discussion

4.1. Introduction

In this chapter, the experiences were described that were executed to evaluate the efficiency of the CDGAE, where it was executed on a device that has the following characteristics:

- Intel Core i7 processor of 2.70 GHz
- RAM 16.00 GB
- Microsoft Windows 10 Pro

The CDGAE was executed with a python programming language, version 3.9.7.

4.2. Experimental Results

The experimental results can be debated from three aspects. The first aspect relates to the depth of the Framework represented by the number of layers for GAE, the second aspect is related to adding the features to the nodes, while the third aspect is related to applying the feature selection to nodes' features. after applying our Framework to real-world graphs.

4.2.1. Dataset

The performance of Framework was experience on Facebook dataset with detail that shown in table (4.1), which gives the undirected and connected graph and it do not have node features.

This dataset is from Facebook pages (November 2017). This dataset represents Facebook Government page, where nodes represent the pages and edges are mutual likes between them[65].

4.2.2. Data management.

The edge information was extracted from the dataset and constructed the adjacency matrix and set the main diameter elements as 1 in order for the elements to update themselves by taking advantage of their information with their adjacency information.

And then, the centrality measurements were calculated based on the graph information and allocated them in the Node features matrix. Then the CFS and VTFS feature selection methods were applied that described in section (3.3.1.2) to the node features matrix which gave different results for the dataset, the features resulting from each method was listed in table (4.2). Thus, three node features matrixes (X, CX, VX) were formed.

the adjacency matrix was concatenated with each of the preceding node feature matrices to form three augmented adjacency matrices Ā (ĀX=concatenate (A, X), ĀCX=concatenate (A, CX), ĀVX=concatenate (A, VX)) in addition to the same adjacency matrix, to be the input of framework to experience its performance by them.

Table 4.1: the dataset and its number of nodes and edges and its graph .

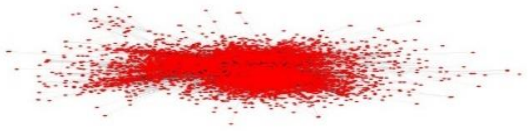
The dataset	Nodes	Edges	Graph
Facebook government Pages	7057	89455	

Table (4.2): the features of dataset after applying the Feature Selection methods to its

Dataset	FS	No. of features	Centrality measurement Features
Facebook government page	VTFS	26	Degree, modified local, volume, Clustering Coefficient, Local Entropy, Mapping Entropy, ClusterRank, Average Distance Neighbor Degree, Square Clustering, Node Density, laplacian, distinctiveness, Neighborhood Connectivity, Semi-local, local clustering, Network, Neighbor Based Centrality, Leverage, SubGraph, diffusion, residual, Eccentricity, Harmonic, Average Distance, Radiality, Heatmap
	CFS	12	degree, modified local, volume, Clustering Coefficient, Local Entropy, Mapping Entropy , Average Distance Neighbor Degree, Square Clustering, Node Density, Leverage, Eccentricity, Average Distance

4.2.3. Building and Implementation of the CDGAE Framework

The CDGAE is built in three depths framework (one-layer framework, three-layer framework, and five-layer framework) to compare the experimental results of the depth of the framework on its performance.

Each CDGAE framework consists of an input layer, the encoder hidden layers, the bottleneck layer, the decoder hidden layers, and the output layer. Figure (4.1) describes layers of the CDGAE framework with their inputs and outputs for the dataset with 1-layer encoder and 1-layer decoder and all features.

4.2.3.1. Implementation details

Each hidden layer of the encoder and decoder is consisting of a dense Layer و ReLU Activation function layer and a BatchNormalization layer.

The dense layer in each neuron of its neurons cells collects the outputs from the all neurons of the previous layer, and the ReLU function results the same values of the neurons if they are positive (activation) and if they are negative, it inactive these neurons (the zero result).

BatchNormalization was applied after the ReLU activation function to help to improve the framework 's performance during training, to improve the noise generated when updating the weights between forwarding and Backward Pass. the BatchNormalization help also to mitigates overfitting So there don't need to take advantage of the dropout regularization layer.

The input layer only organizes the inputs to the hidden layers, and the bottleneck layer collects the outputs of the previous layer in neurons of a dense layer and passes them to the next layer, and the output layer consists of a dense layer with the SoftMax activation function, to be the output of the final neurons (0, 1). When the output is close to 1, the neuron becomes active, and when the output is close to 0, it becomes inactive.

Hyperparameter tuning is performed during training via optimizing a loss function. The key hyperparameters include batch size and the dimension of the hidden layers. In general, A comparable set of hyperparameters were tried to keep across dataset to concentrate on the efficacy of the frameworks.

In all experiments, the dimension of the hidden layers in the GAE architecture is set at (Ā - Ā /2-number of community- Ā /2- Ā) for the one-layer framework , (Ā - Ā /2- Ā /4- Ā /8-number of community- Ā /8- Ā /4- Ā /2- Ā) for the three-layer framework and (Ā - Ā /2- Ā /4- Ā /8- Ā /16- Ā /32-number of community- Ā /32- Ā /16- Ā /8- Ā /4- Ā /2- Ā) for the five-layer framework . the CDGAE frameworks were trained for 300 epochs using the batch size of 20 samples.

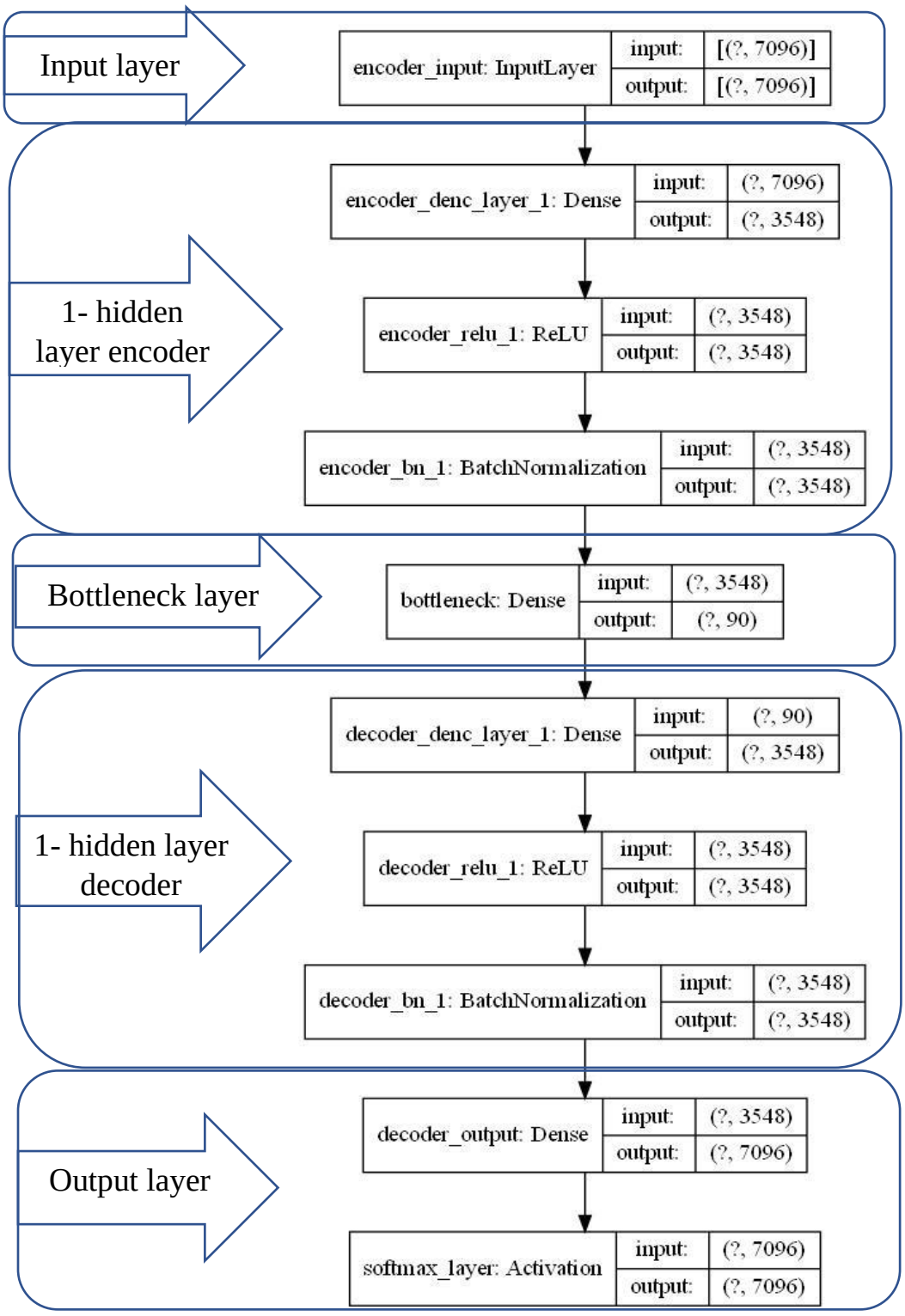


Figure (4.1): Describe layers of the CDGAE framework.

4.2.3.2. **Number of Community**

The number of communities is determined by the spectral property of the Bethe Hessian matrix H, this number is set as the number of neurons of the bottleneck layer of framework.

The number of communities computed for the Facebook government pages dataset is 90.

After building the CDGAE framework, it is trained for 300 epochs during them, the error rate in the framework is reduced by decreasing the gradient of training-loss and the framework generalizability to new data is increased by reducing the validating-loss, as shown in figure (4.2). Thus, the framework succeeded by reconstructing the input data into the output layer of the decoder by utilizing the bottleneck layer data. Here, the important information for the graph is concentrated in the bottleneck layer.

Then the encoder was extracted and used to find the communities by using the argmax function to determine the node membership for any community.

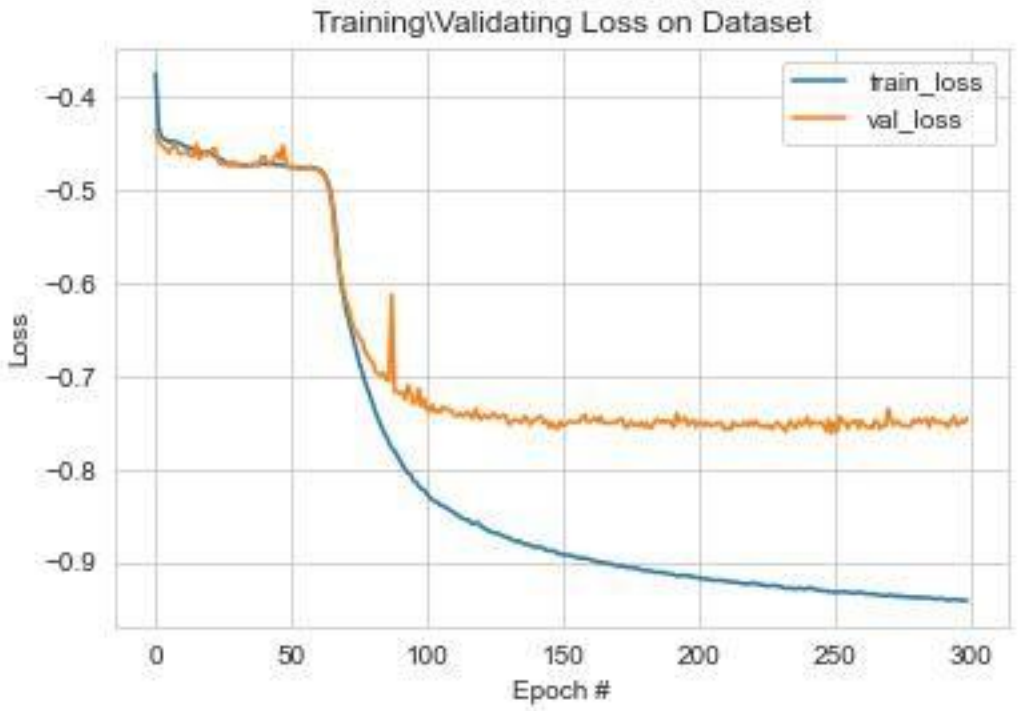


Figure (4.2): training and validating loss Curves diagram.

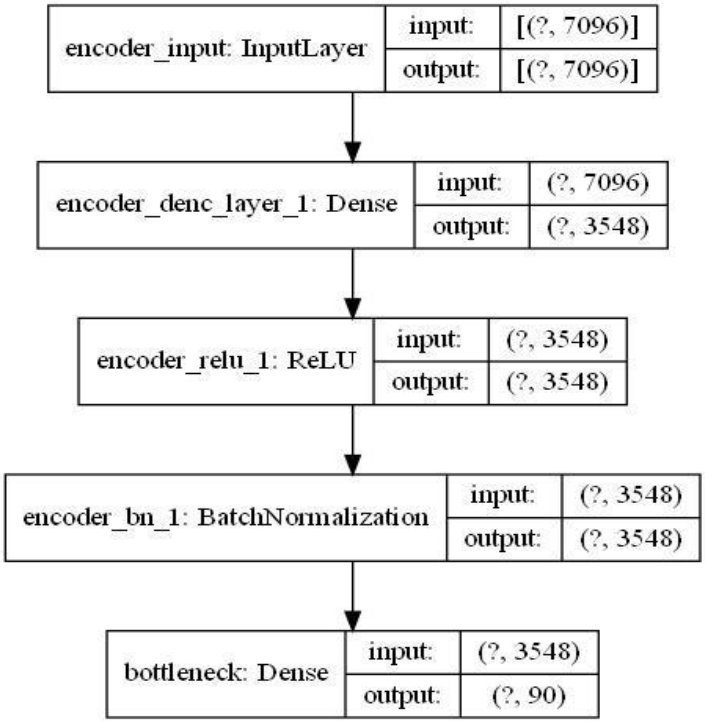


Figure (4.3): The encoder extracted from CDGAE framework.

4.2.4 Experimental results of implementing CDGAE Frameworks

Here, the implementation of each framework of the frameworks mentioned in table (3.4) on Facebook government pages.

4.2.4.1 AF1L_ Framework

This CDGAE framework is the one-layer model with the $\bar{A}X$ inputs and output, and number of bottleneck layer neurons equal to the number of dataset's communities.

Figure (4.4) shows the comprehensive schematic diagram for implementation of AF1L_Framework on Facebook government pages dataset, consist of (7057 node+39 feature =7096) Input and output, 1-encoder layer with number of neurons: (3548), 1-decoder layer with number of neurons: (3548), and the bottleneck layer with neurons (number of communities = 90).

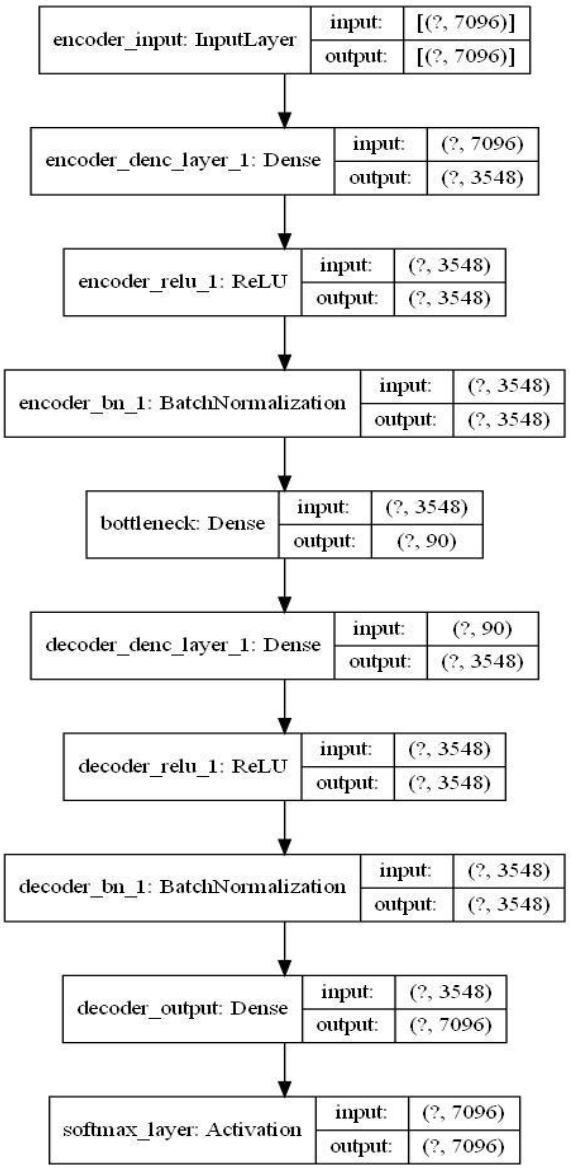


Figure (4.4): The implementation of AF1L_ Framework on the dataset.

Figure (4.5) shows the Curves diagram of training/validating loss function of AF1L_ Framework when applied on Facebook government pages dataset.

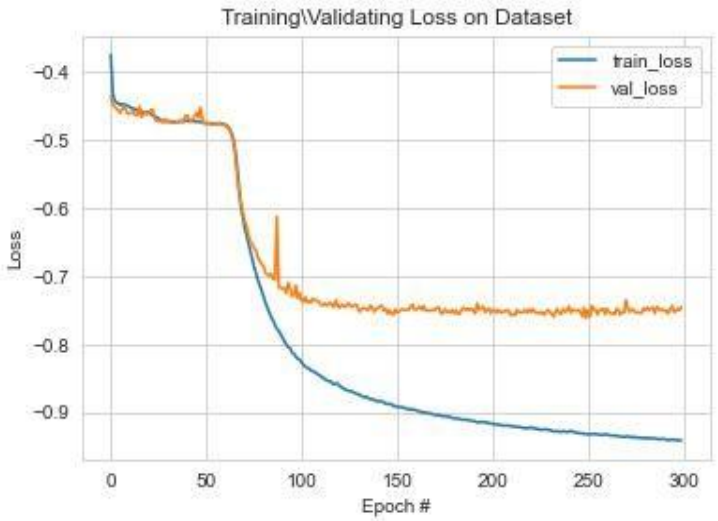


Figure (4.5): The diagram of the loss function curves of AF1L_ Framework

Figure (4.6) shows the encoder extracted from AF1L_ Framework after training it on Facebook government pages dataset**.**

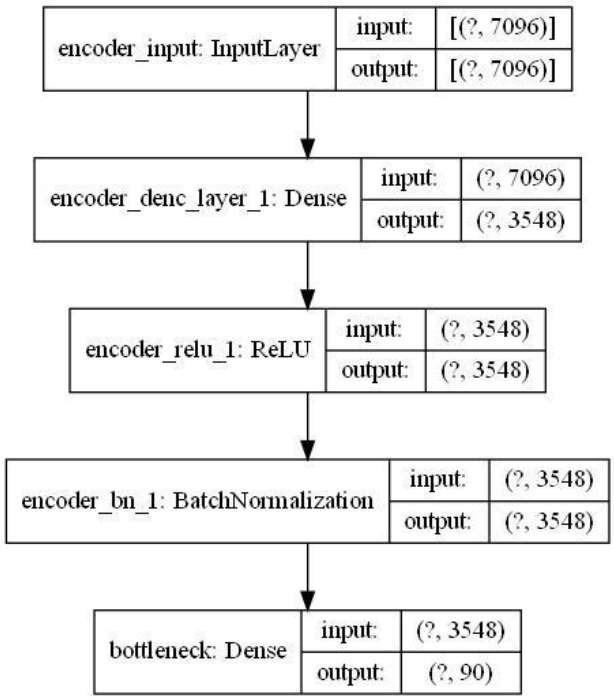


Figure (4.6): The encoder extracted from AF1L_ Framework

4.2.4.2 AF3L_ Framework

This CDGAE framework is the three-layers framework with the ĀX inputs and output, and number of bottleneck layer neurons equal to the number of dataset's communities.

Figure (4.7) shows the comprehensive schematic diagram for implementation of AF3L_Framework on Facebook government pages dataset, consist of (7057 node+39 feature =7096) Input and output, 3-encoder layers with number of neurons: (3548, 1774, 887), 3-decoder layers with number of neurons: (887, 1774, 3548), and the bottleneck layer with neurons (number of communities = 90).

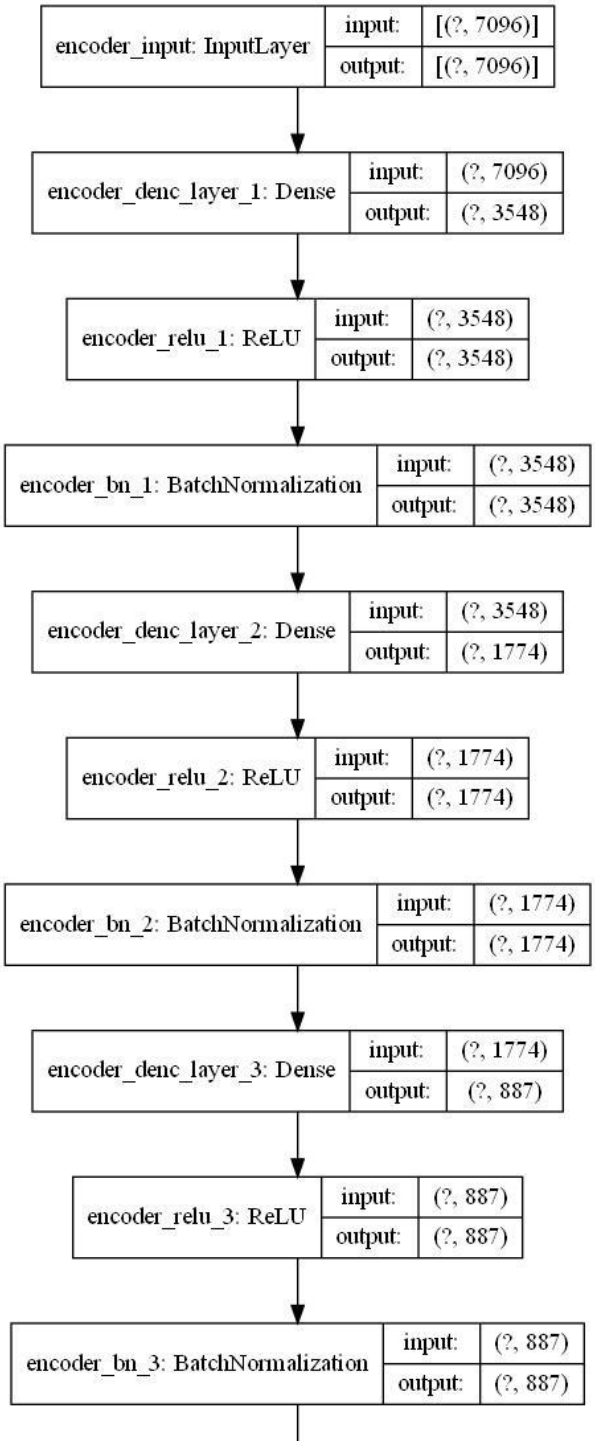


Figure (4.7): The implementation of AF3L_ Framework on the dataset.

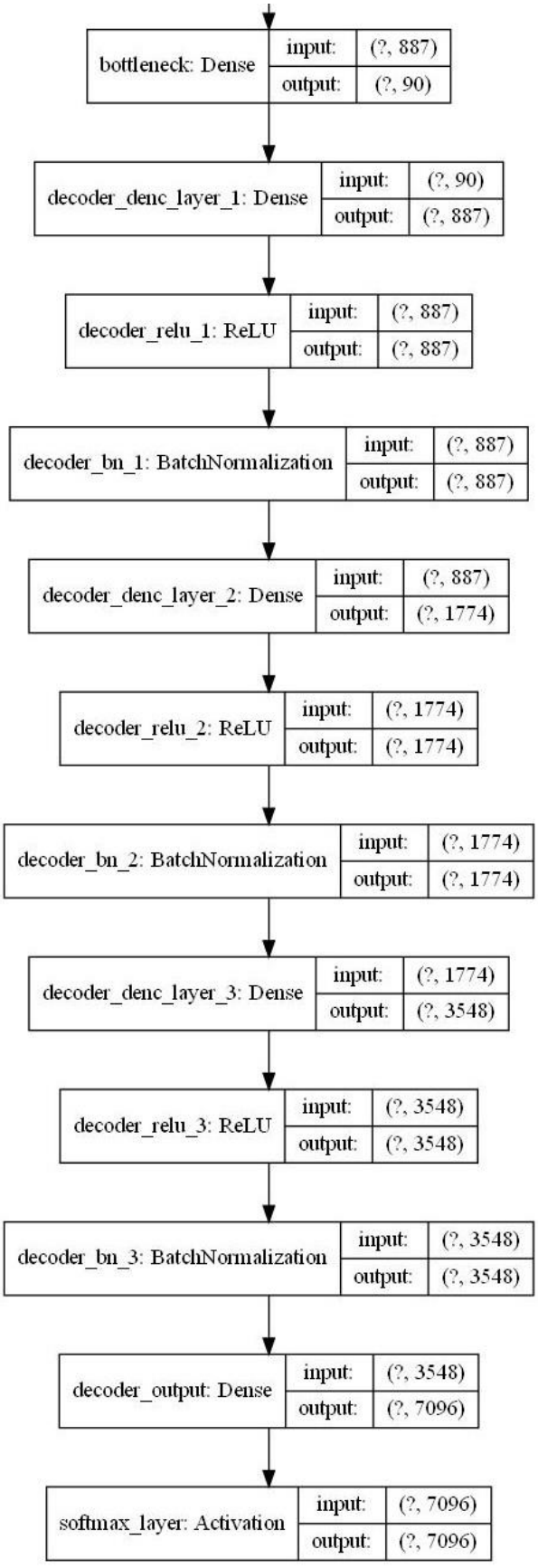

bottleneck: Dense
input: (?, 887)
output: (?, 90)
decoder_denc_layer_1: Dense
input: (?, 90)
output: (?, 887)
decoder_relu_1: ReLU
input: (?, 887)
output: (?, 887)
decoder_bn_1: BatchNormalization
input: (?, 887)
output: (?, 887)
decoder_denc_layer_2: Dense
input: (?, 887)
output: (?, 1774)
decoder_relu_2: ReLU
input: (?, 1774)
output: (?, 1774)
decoder_bn_2: BatchNormalization
input: (?, 1774)
output: (?, 1774)
decoder_denc_layer_3: Dense
input: (?, 1774)
output: (?, 3548)
decoder_relu_3: ReLU
input: (?, 3548)
output: (?, 3548)
decoder_bn_3: BatchNormalization
input: (?, 3548)
output: (?, 3548)
decoder_output: Dense
input: (?, 3548)
output: (?, 7096)
softmax_layer: Activation
input: (?, 7096)
output: (?, 7096)

Figure (4.8) shows the Curves diagram of training/validating loss function of AF3L_ Framework when applied on Facebook government pages dataset.

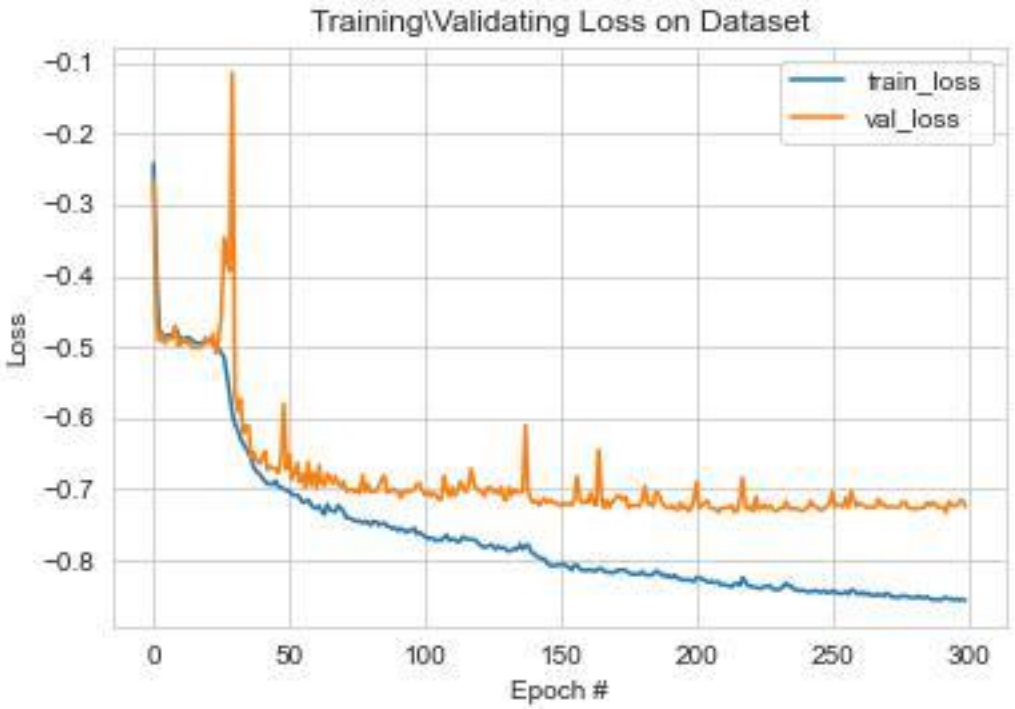


Figure (4.8): The diagram of the loss function curves of AF3L_ Framework

Figure (4.9) shows the encoder extracted from AF3L_ Framework after training it on Facebook government pages dataset**.**

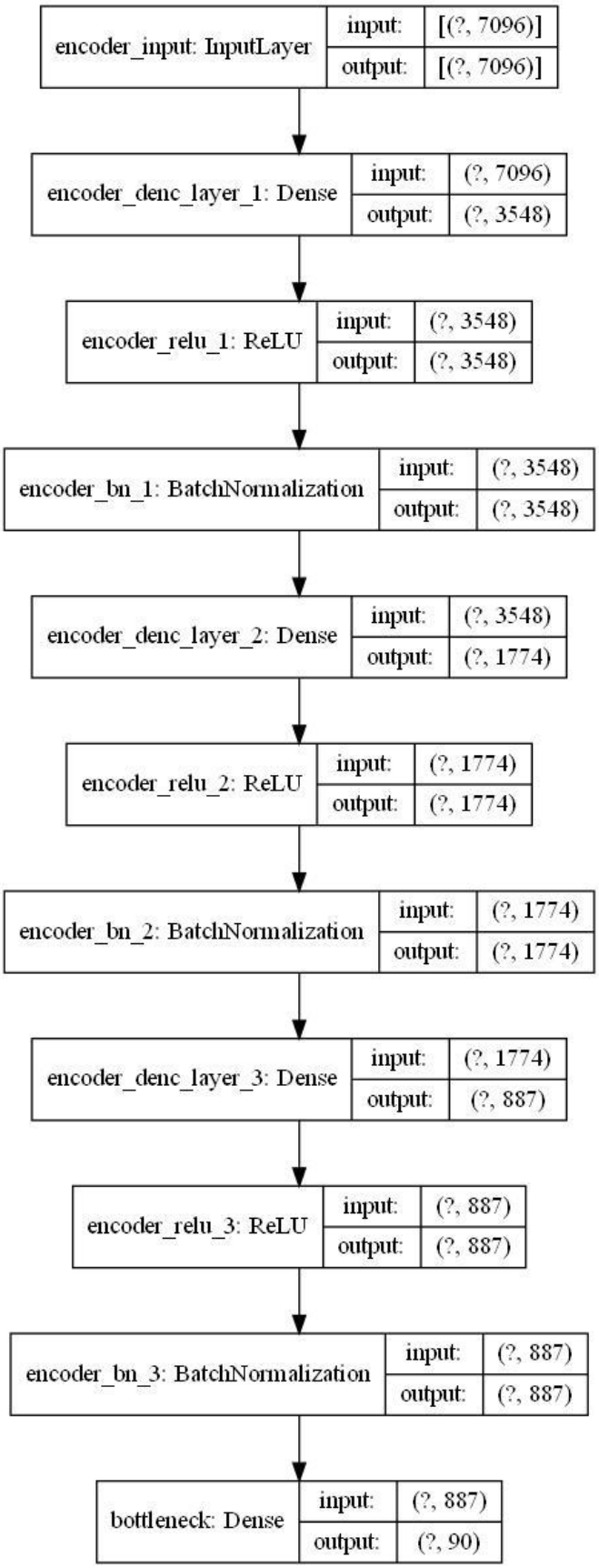


Figure (4.9): The encoder extracted from AF3L_ Framework

4.2.4.3 AF5L_ Framework

This CDGAE framework is the five-layer framework with the ĀX inputs and output, and number of bottleneck layer neurons equal to the number of dataset's communities.

Figure (4.10) shows the comprehensive schematic diagram for implementation of AF5L_Framework on Facebook government pages dataset, consist of (7057 node+39 feature =7096) Input and output, 5-encoder layers with number of neurons: (3548, 1774, 887, 443, 221), 5-decoder layers with number of neurons: (221, 443, 887, 1774, 3548), and the bottleneck layer with neurons (number of communities = 90).

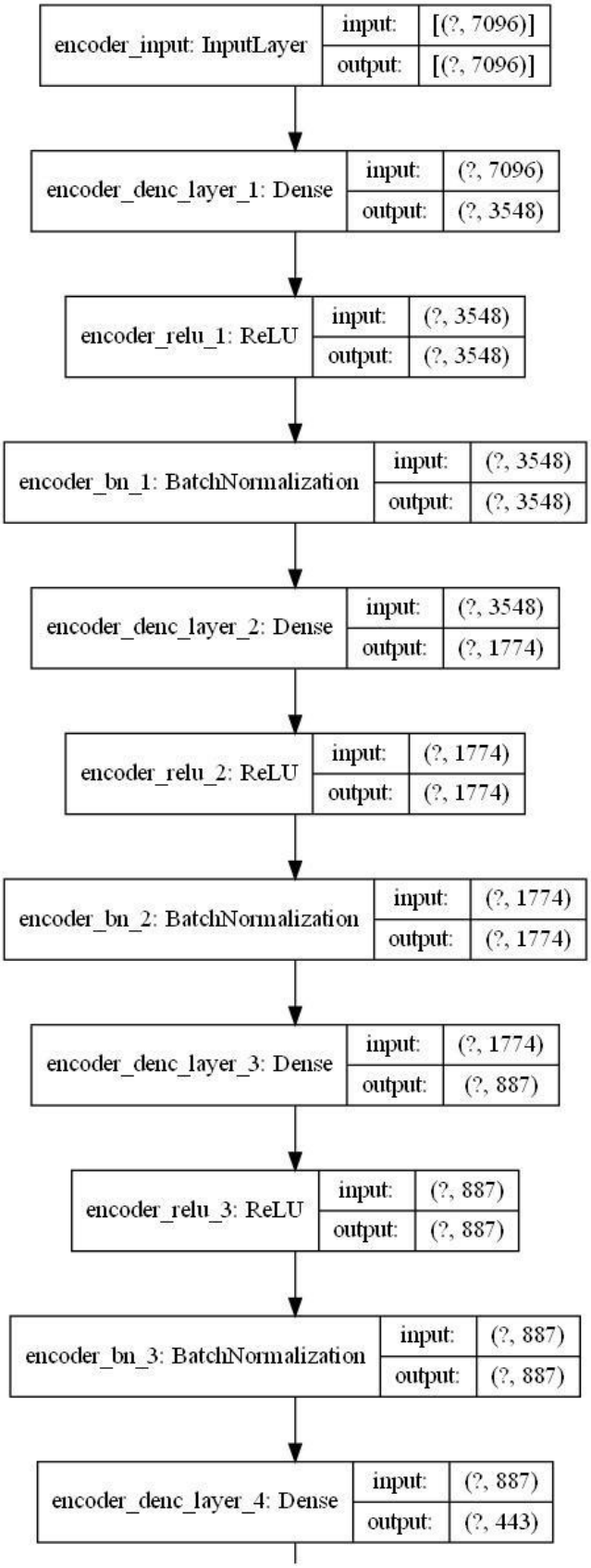
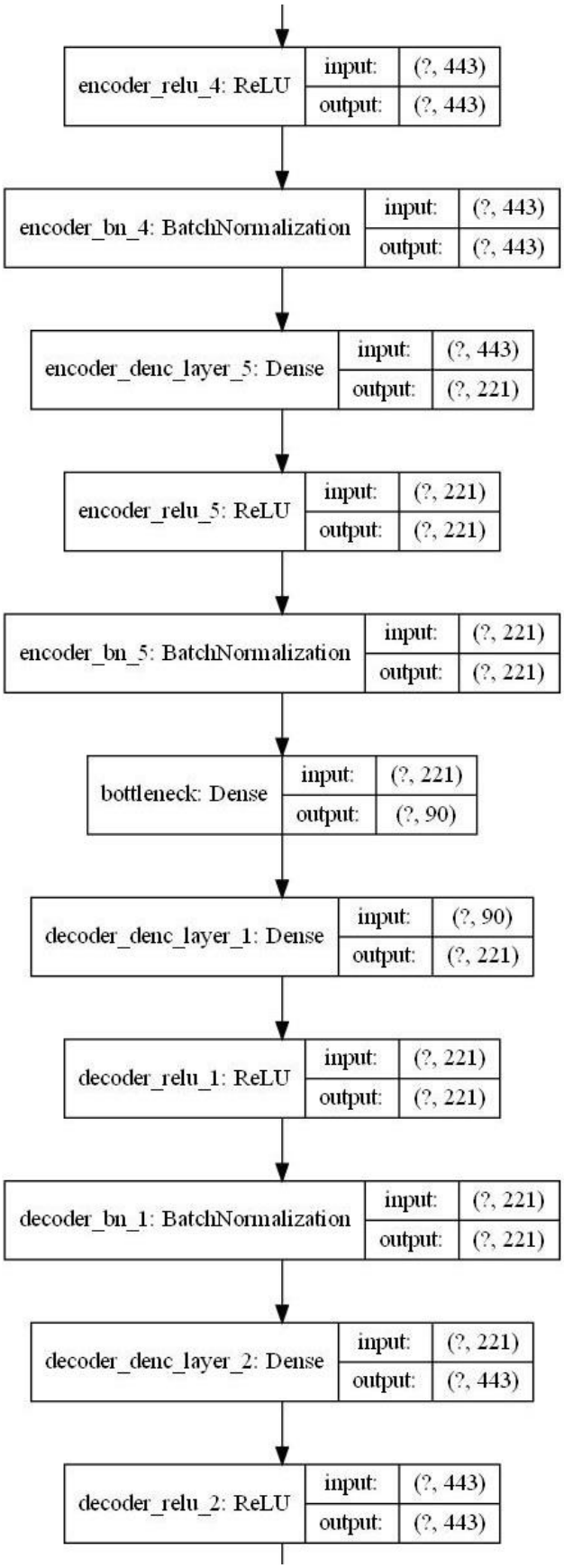


Figure (4.10): The implementation of AF5L_ Framework on the dataset.


encoder_relu_4: ReLU
input: (?, 443)
output: (?, 443)
encoder_bn_4: BatchNormalization
input: (?, 443)
output: (?, 443)
encoder_denc_layer_5: Dense
input: (?, 443)
output: (?, 221)
encoder_relu_5: ReLU
input: (?, 221)
output: (?, 221)
encoder_bn_5: BatchNormalization
input: (?, 221)
output: (?, 221)
bottleneck: Dense
input: (?, 221)
output: (?, 90)
decoder_denc_layer_1: Dense
input: (?, 90)
output: (?, 221)
decoder_relu_1: ReLU
input: (?, 221)
output: (?, 221)
decoder_bn_1: BatchNormalization
input: (?, 221)
output: (?, 221)
decoder_denc_layer_2: Dense
input: (?, 221)
output: (?, 443)
decoder_relu_2: ReLU
input: (?, 443)
output: (?, 443)

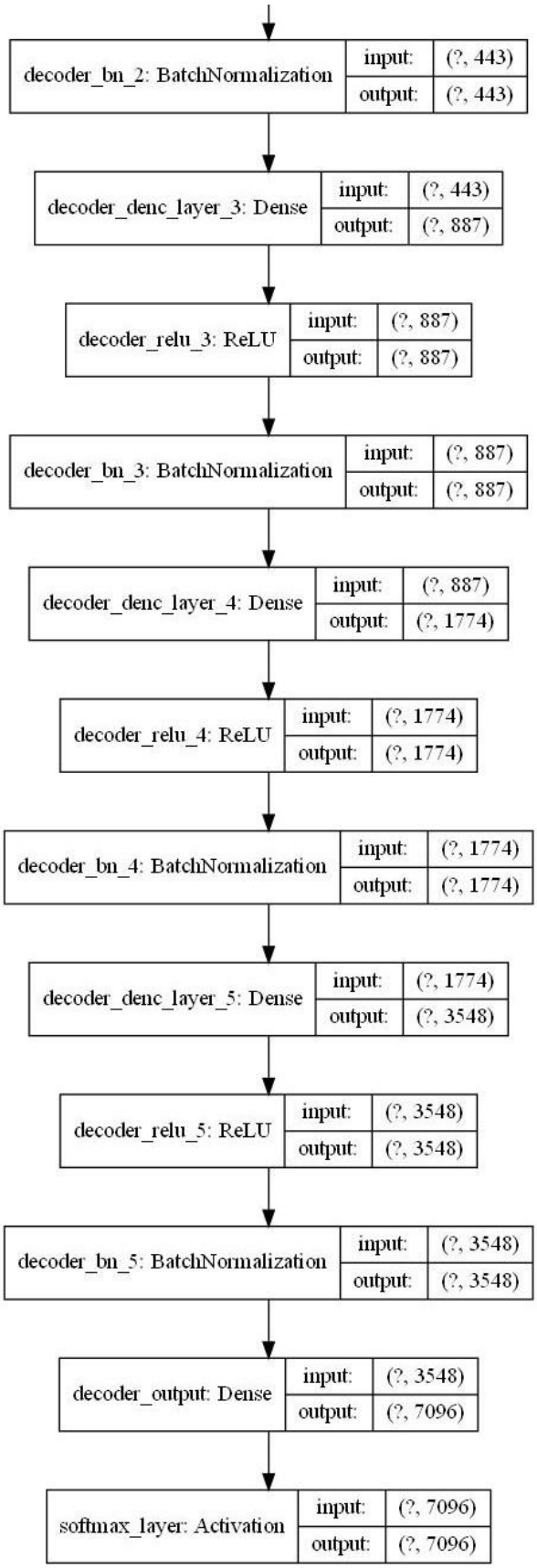

decoder_bn_2: BatchNormalization
input: (?, 443)
output: (?, 443)
decoder_denc_layer_3: Dense
input: (?, 443)
output: (?, 887)
decoder_relu_3: ReLU
input: (?, 887)
output: (?, 887)
decoder_bn_3: BatchNormalization
input: (?, 887)
output: (?, 887)
decoder_denc_layer_4: Dense
input: (?, 887)
output: (?, 1774)
decoder_relu_4: ReLU
input: (?, 1774)
output: (?, 1774)
decoder_bn_4: BatchNormalization
input: (?, 1774)
output: (?, 1774)
decoder_denc_layer_5: Dense
input: (?, 1774)
output: (?, 3548)
decoder_relu_5: ReLU
input: (?, 3548)
output: (?, 3548)
decoder_bn_5: BatchNormalization
input: (?, 3548)
output: (?, 3548)
decoder_output: Dense
input: (?, 3548)
output: (?, 7096)
softmax_layer: Activation
input: (?, 7096)
output: (?, 7096)

Figure (4.11) shows the Curves diagram of training/validating loss function of AF5L_ Framework when applied on Facebook government pages dataset.

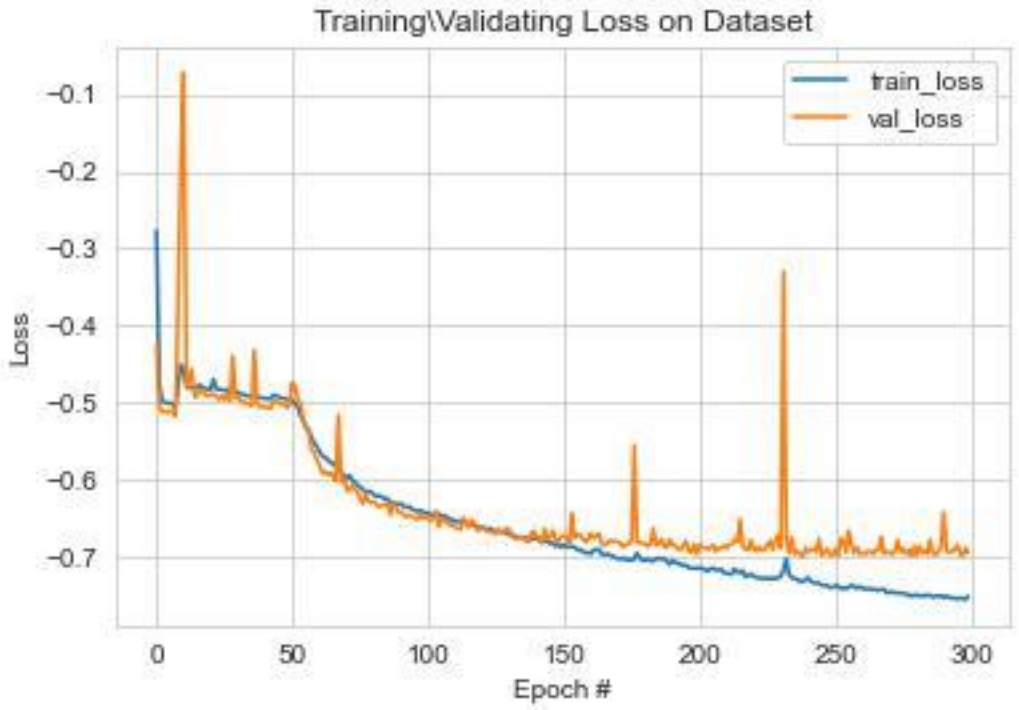


Figure (4.11): The diagram of the loss function curves of AF5L_ Framework

Figure (4.12) shows the encoder extracted from AF5L_ Framework after training it on Facebook government pages dataset**.**

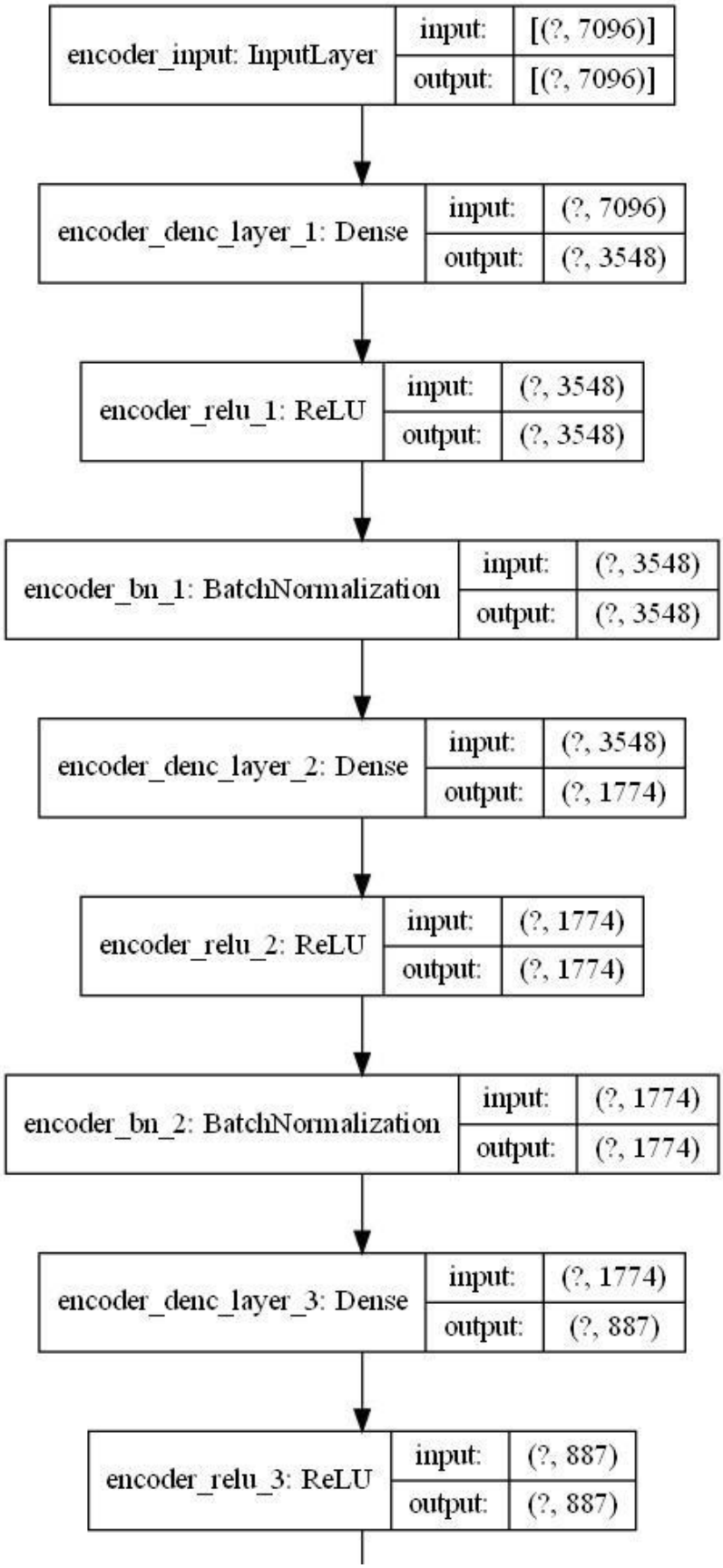
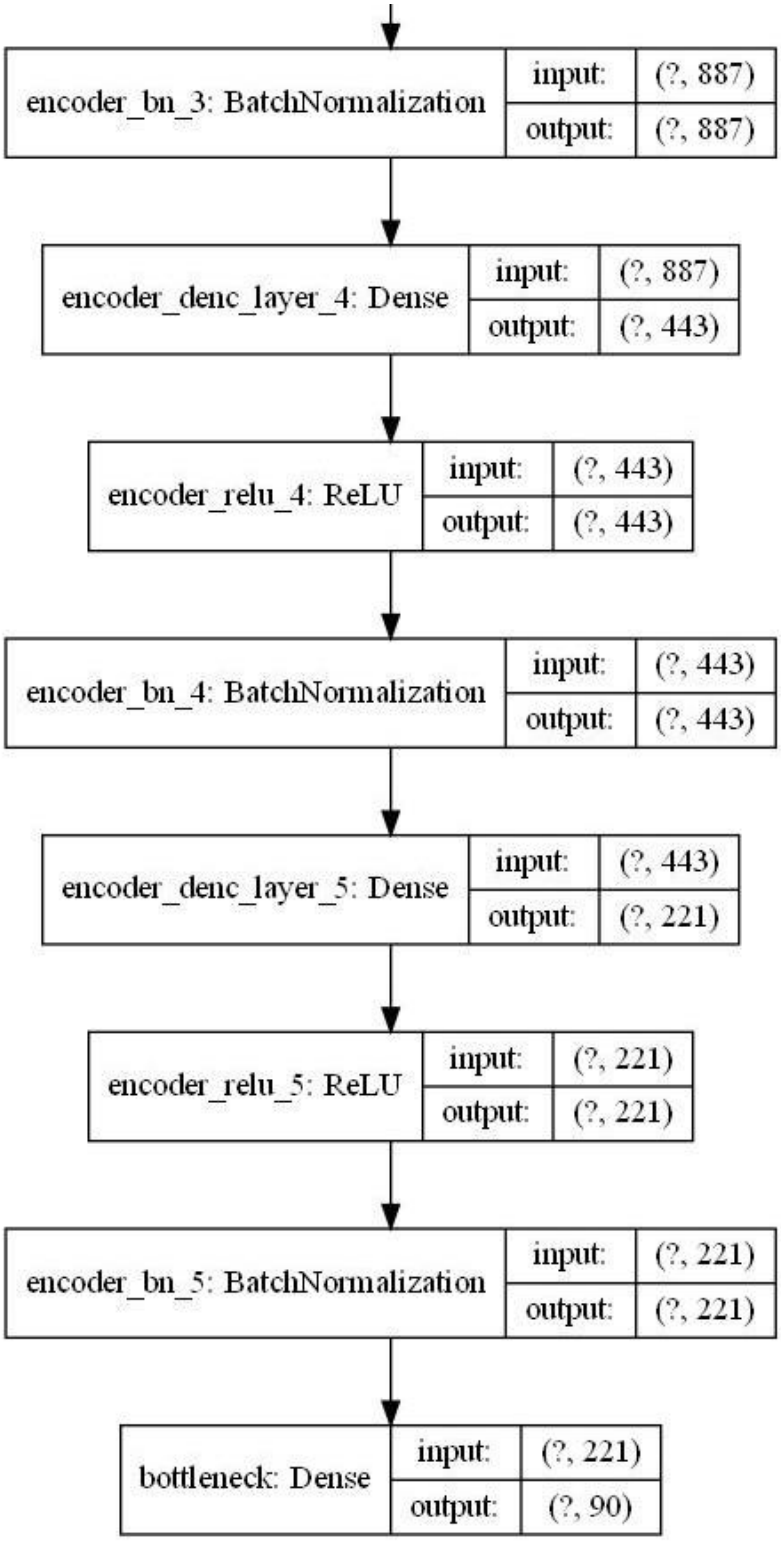


Figure (4.12): The encoder extracted from AF5L_ Framework


encoder_bn_3: BatchNormalization
input: (?, 887)
output: (?, 887)
encoder_denc_layer_4: Dense
input: (?, 887)
output: (?, 443)
encoder_relu_4: ReLU
input: (?, 443)
output: (?, 443)
encoder_bn_4: BatchNormalization
input: (?, 443)
output: (?, 443)
encoder_denc_layer_5: Dense
input: (?, 443)
output: (?, 221)
encoder_relu_5: ReLU
input: (?, 221)
output: (?, 221)
encoder_bn_5: BatchNormalization
input: (?, 221)
output: (?, 221)
bottleneck: Dense
input: (?, 221)
output: (?, 90)

4.2.4.4 VTF1L_ Framework

This CDGAE framework is the one-layer framework with the ĀVX inputs and output, and number of bottleneck layer neurons equal to the number of dataset's communities.

Figure (4.13) shows the comprehensive schematic diagram for implementation of VTF1L_Framework on Facebook government pages dataset, consist of (7057 node+26 feature =7083) Input and output, 1-encoder layer with number of neurons: (3541), 1-decoder layer with number of neurons: (3541), and the bottleneck layer with neurons (number of communities = 90).

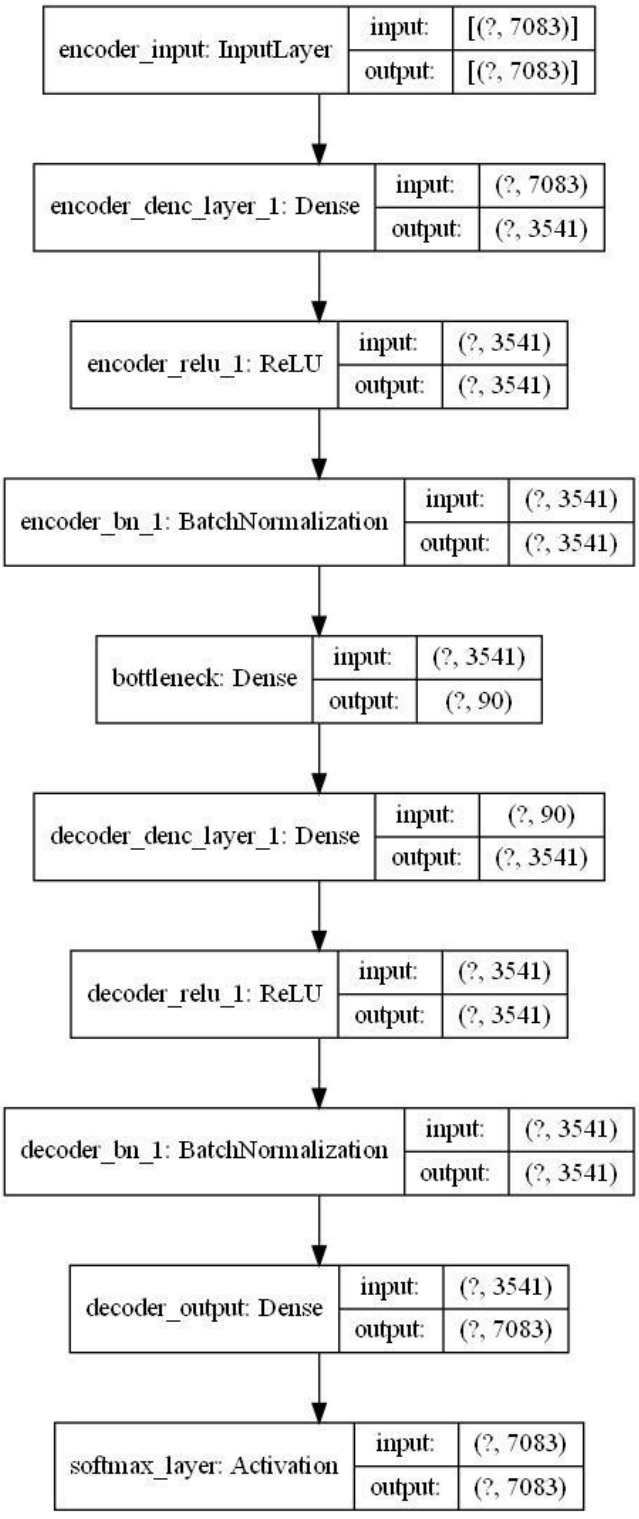


Figure (4.13): The implementation of VTF1L_ Framework on the dataset.

Figure (4.14) shows the Curves diagram of training/validating loss function of VTF1L _ Framework when applied on Facebook government pages dataset.

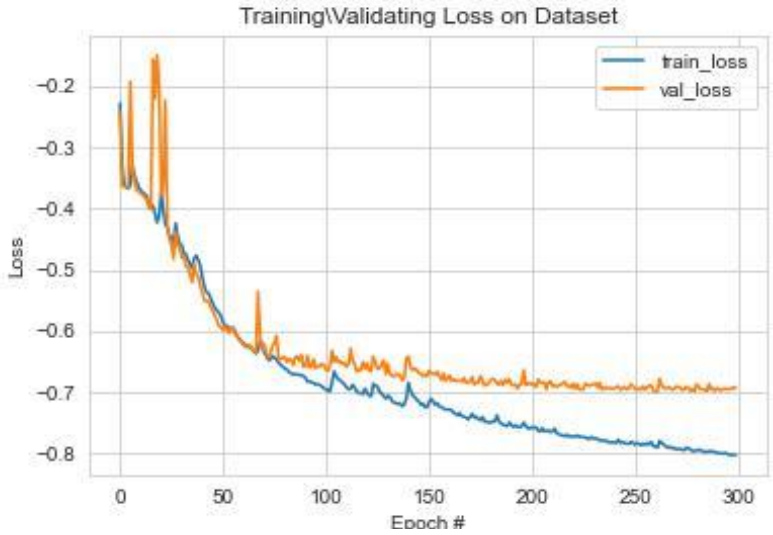


Figure (4.14): The diagram of the loss function curves of VTF1L _ Framework

Figure (4.15) shows the encoder extracted from VTF1L _ Framework after training it on Facebook government pages dataset**.**

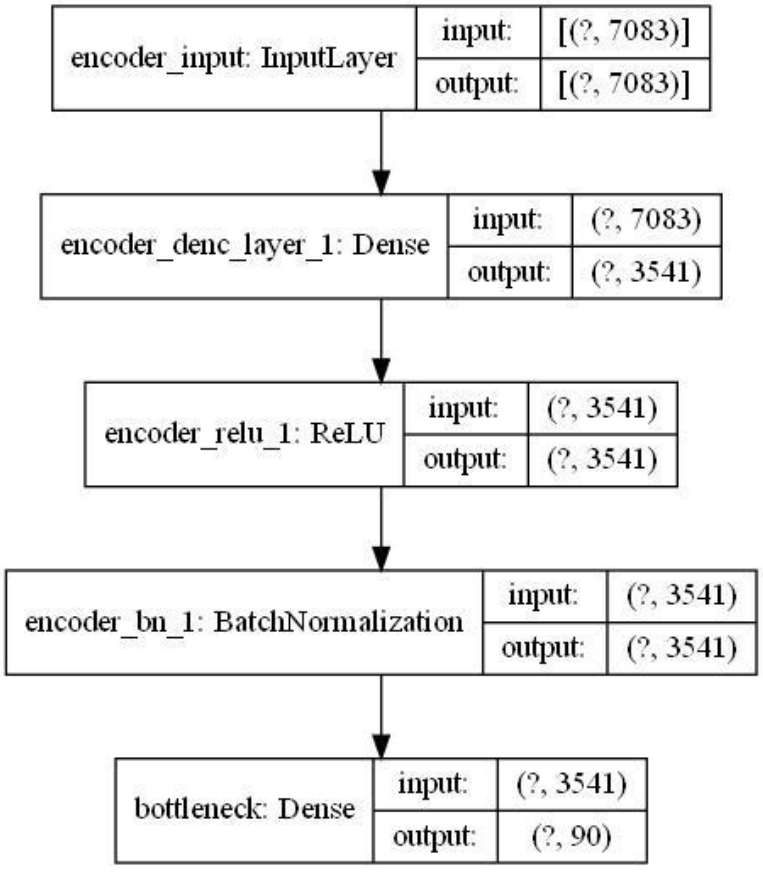


Figure (4.15): The encoder extracted from VTF1L _ Framework

4.2.4.5 VTF3L_ Framework

This CDGAE framework is the three-layers framework with the ĀVX inputs and output, and number of bottleneck layer neurons equal to the number of dataset's communities.

Figure (4.16) shows the comprehensive schematic diagram for implementation of VTF3L_framework on Facebook government pages dataset, consist of (7057 node+26 feature =7083) Input and output, 3-encoder layer with number of neurons: (3541, 1770, 885), 3-decoder layer with number of neurons: (885, 1770, 3541), and the bottleneck layer with neurons (number of communities = 90).

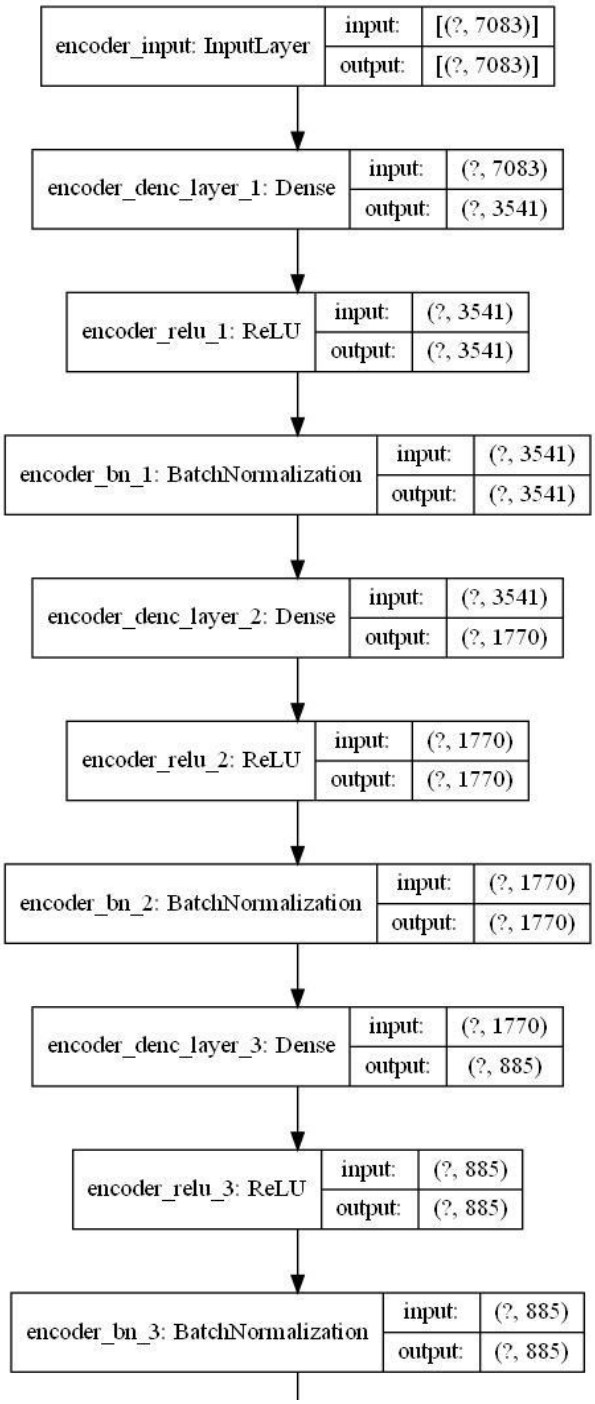


Figure (4.16): The implementation of VTF3L_ Framework on the dataset.

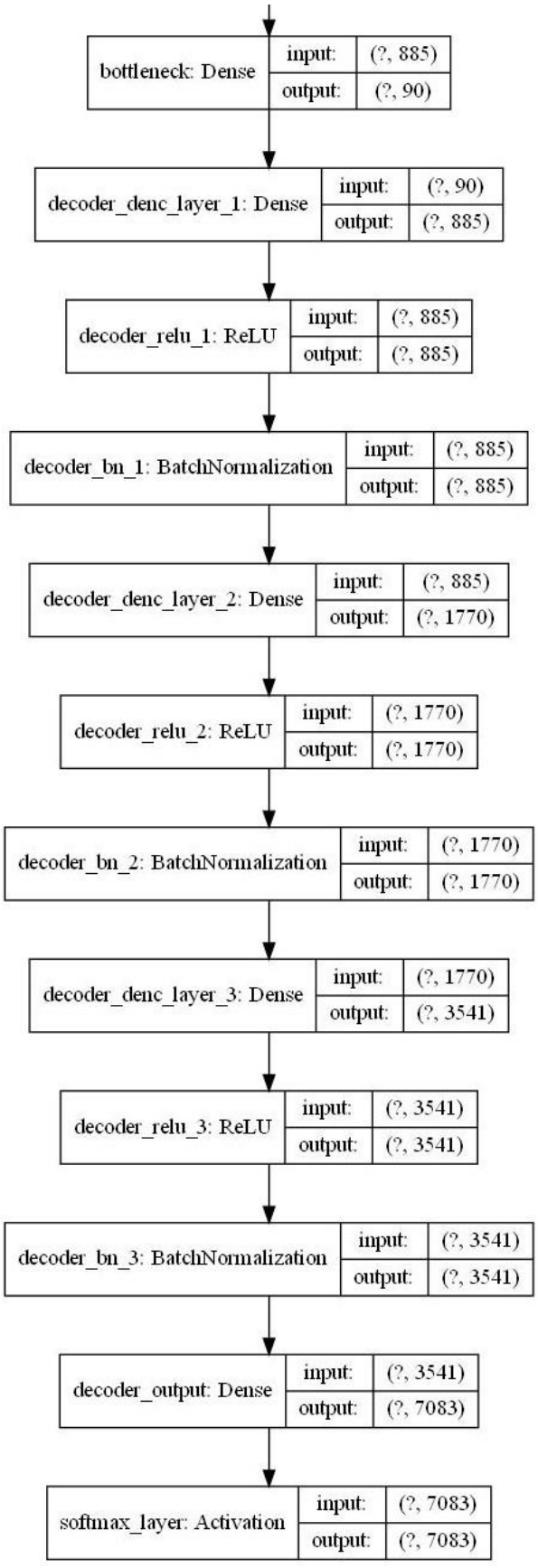

bottleneck: Dense
input: (?, 885)
output: (?, 90)
decoder_denc_layer_1: Dense
input: (?, 90)
output: (?, 885)
decoder_relu_1: ReLU
input: (?, 885)
output: (?, 885)
decoder_bn_1: BatchNormalization
input: (?, 885)
output: (?, 885)
decoder_denc_layer_2: Dense
input: (?, 885)
output: (?, 1770)
decoder_relu_2: ReLU
input: (?, 1770)
output: (?, 1770)
decoder_bn_2: BatchNormalization
input: (?, 1770)
output: (?, 1770)
decoder_denc_layer_3: Dense
input: (?, 1770)
output: (?, 3541)
decoder_relu_3: ReLU
input: (?, 3541)
output: (?, 3541)
decoder_bn_3: BatchNormalization
input: (?, 3541)
output: (?, 3541)
decoder_output: Dense
input: (?, 3541)
output: (?, 7083)
softmax_layer: Activation
input: (?, 7083)
output: (?, 7083)

Figure (4.17) shows the Curves diagram of training/validating loss function of VTF3L _ Framework when applied on Facebook government pages dataset.

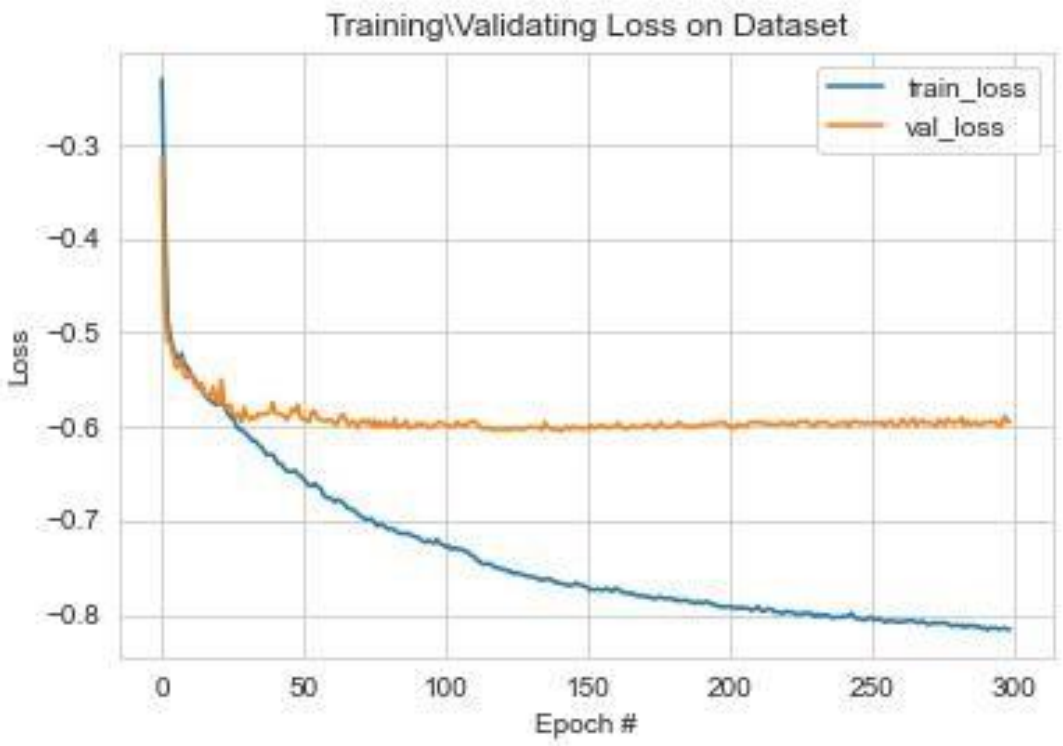


Figure (4.17): The diagram of the loss function curves of VTF3L_ Framework

Figure (4.18) shows the encoder extracted from VTF3L_ Framework after training it on Facebook government pages dataset**.**

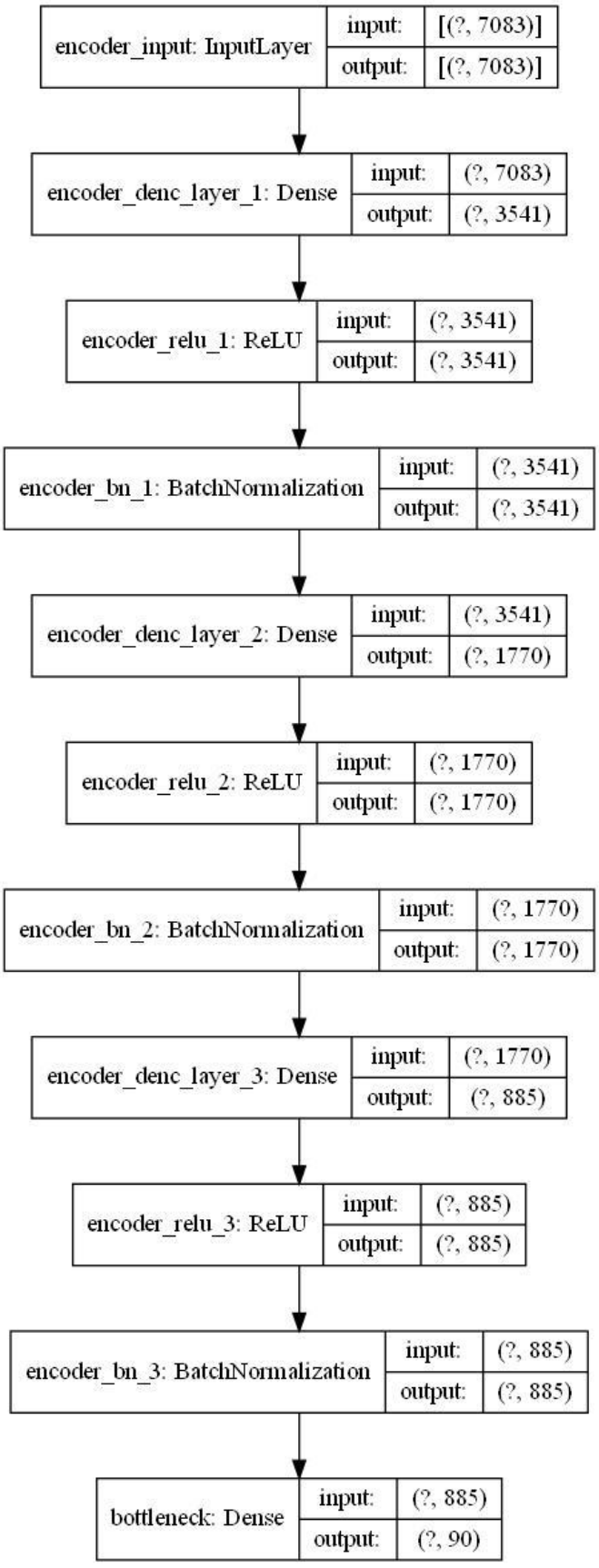


Figure (4.18): The encoder extracted from VTF3L_Framework

4.2.4.6 VTF5L_ Framework

This CDGAE framework is the five-layers framework with the ĀVX inputs and output, and number of bottleneck layer neurons equal to the number of dataset's communities.

Figure (4.19) shows the comprehensive schematic diagram for implementation of VTF5L_Framework on Facebook government pages dataset, consist of (7057 node+26 feature =7083) Input and output, 5-encoder layer with number of neurons: (3541, 1770, 885, 442, 221), 5-decoder layer with number of neurons: (221, 442, 885, 1770, 3541), and the bottleneck layer with neurons (number of communities = 90).

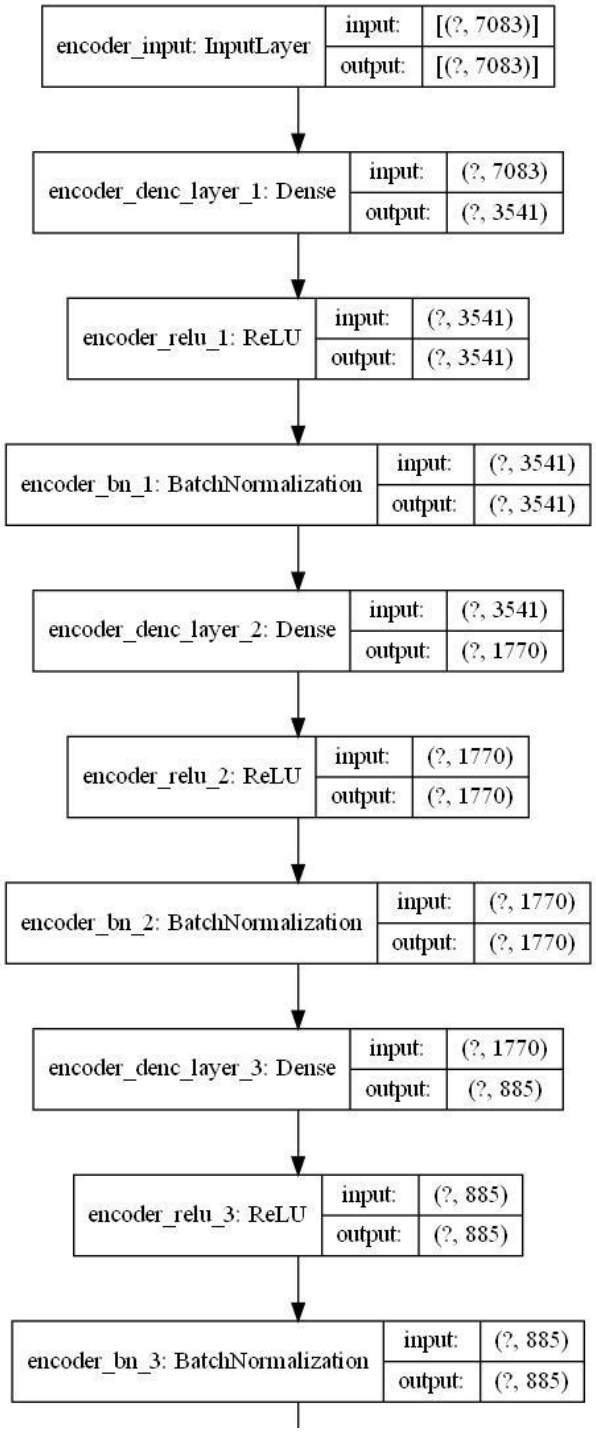
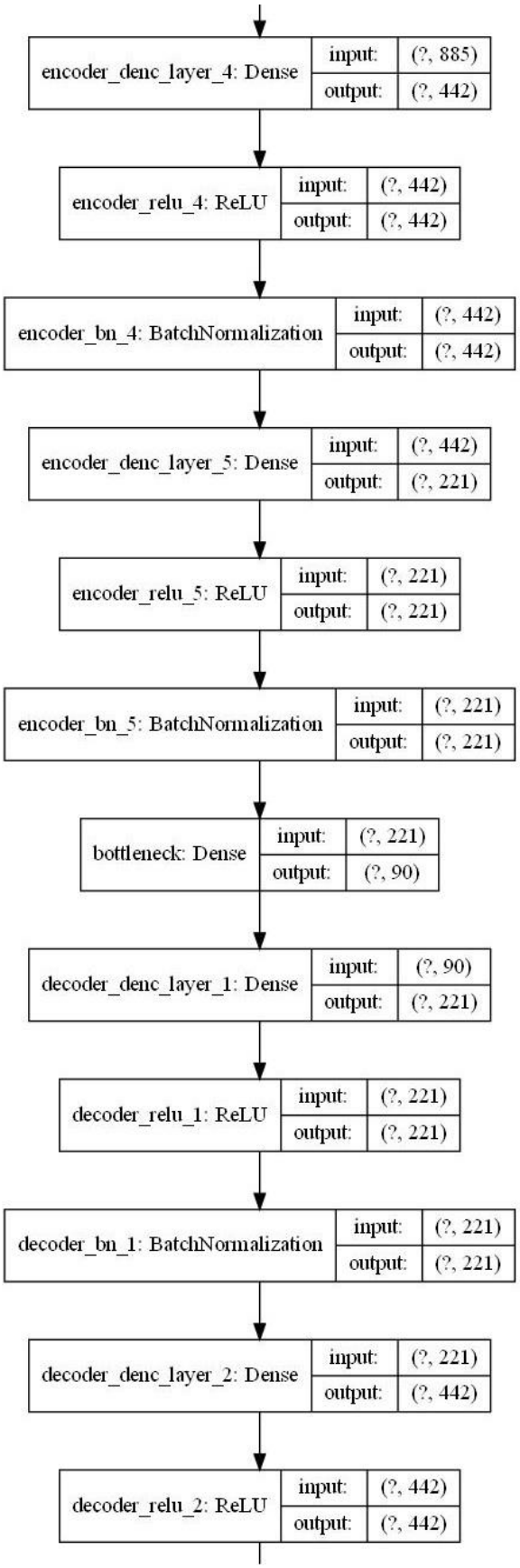


Figure (4.19): The implementation of VTF5L_ Framework on the dataset.


encoder_denc_layer_4: Dense
input: (?, 885)
output: (?, 442)
encoder_relu_4: ReLU
input: (?, 442)
output: (?, 442)
encoder_bn_4: BatchNormalization
input: (?, 442)
output: (?, 442)
encoder_denc_layer_5: Dense
input: (?, 442)
output: (?, 221)
encoder_relu_5: ReLU
input: (?, 221)
output: (?, 221)
encoder_bn_5: BatchNormalization
input: (?, 221)
output: (?, 221)
bottleneck: Dense
input: (?, 221)
output: (?, 90)
decoder_denc_layer_1: Dense
input: (?, 90)
output: (?, 221)
decoder_relu_1: ReLU
input: (?, 221)
output: (?, 221)
decoder_bn_1: BatchNormalization
input: (?, 221)
output: (?, 221)
decoder_denc_layer_2: Dense
input: (?, 221)
output: (?, 442)
decoder_relu_2: ReLU
input: (?, 442)
output: (?, 442)

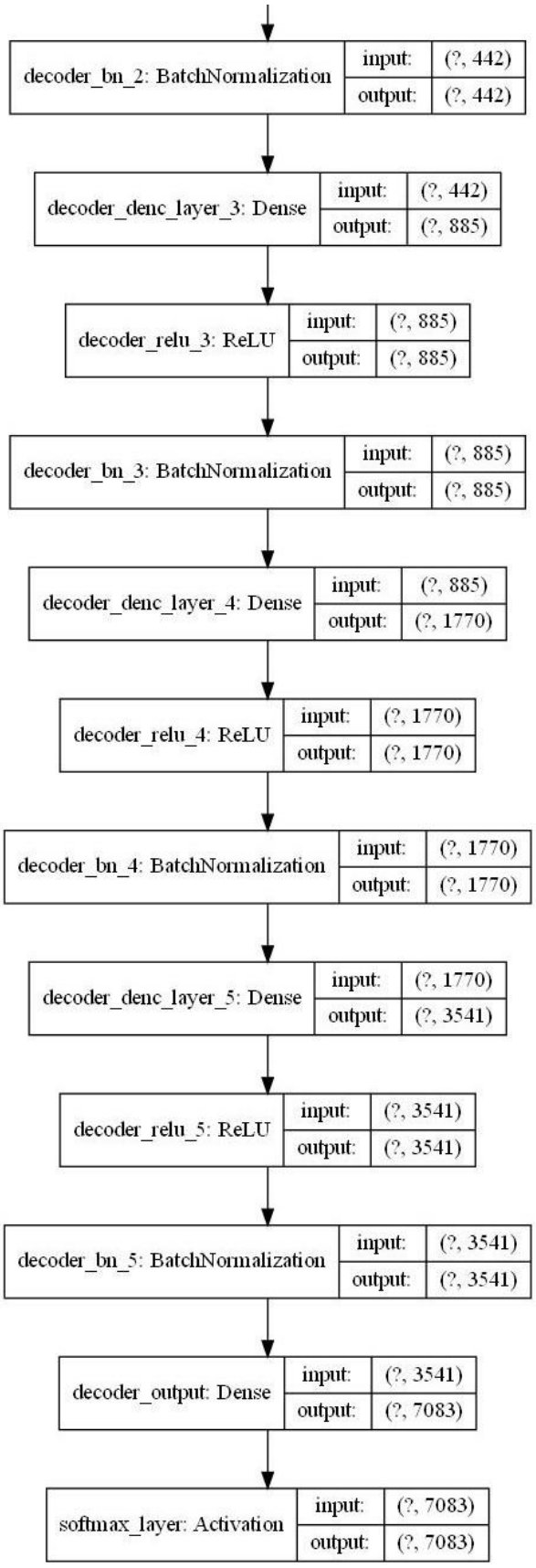

decoder_bn_2: BatchNormalization
input: (?, 442)
output: (?, 442)
decoder_denc_layer_3: Dense
input: (?, 442)
output: (?, 885)
decoder_relu_3: ReLU
input: (?, 885)
output: (?, 885)
decoder_bn_3: BatchNormalization
input: (?, 885)
output: (?, 885)
decoder_denc_layer_4: Dense
input: (?, 885)
output: (?, 1770)
decoder_relu_4: ReLU
input: (?, 1770)
output: (?, 1770)
decoder_bn_4: BatchNormalization
input: (?, 1770)
output: (?, 1770)
decoder_denc_layer_5: Dense
input: (?, 1770)
output: (?, 3541)
decoder_relu_5: ReLU
input: (?, 3541)
output: (?, 3541)
decoder_bn_5: BatchNormalization
input: (?, 3541)
output: (?, 3541)
decoder_output: Dense
input: (?, 3541)
output: (?, 7083)
softmax_layer: Activation
input: (?, 7083)
output: (?, 7083)

Figure (4.20) shows the Curves diagram of training/validating loss function of VTF5L _ Framework when applied on Facebook government pages dataset.

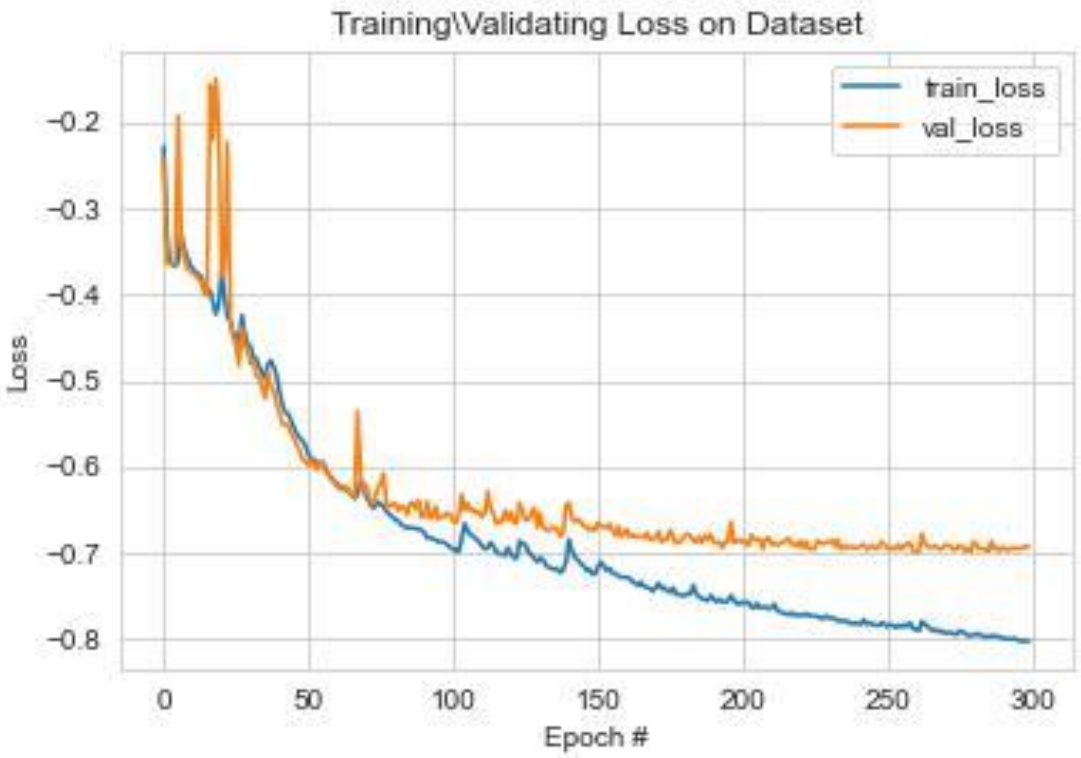


Figure (4.20): The diagram of the loss function curves of VTF5L_ Framework

Figure (4.21) shows the encoder extracted from VTF5L_ Framework after training it on Facebook government pages dataset**.**

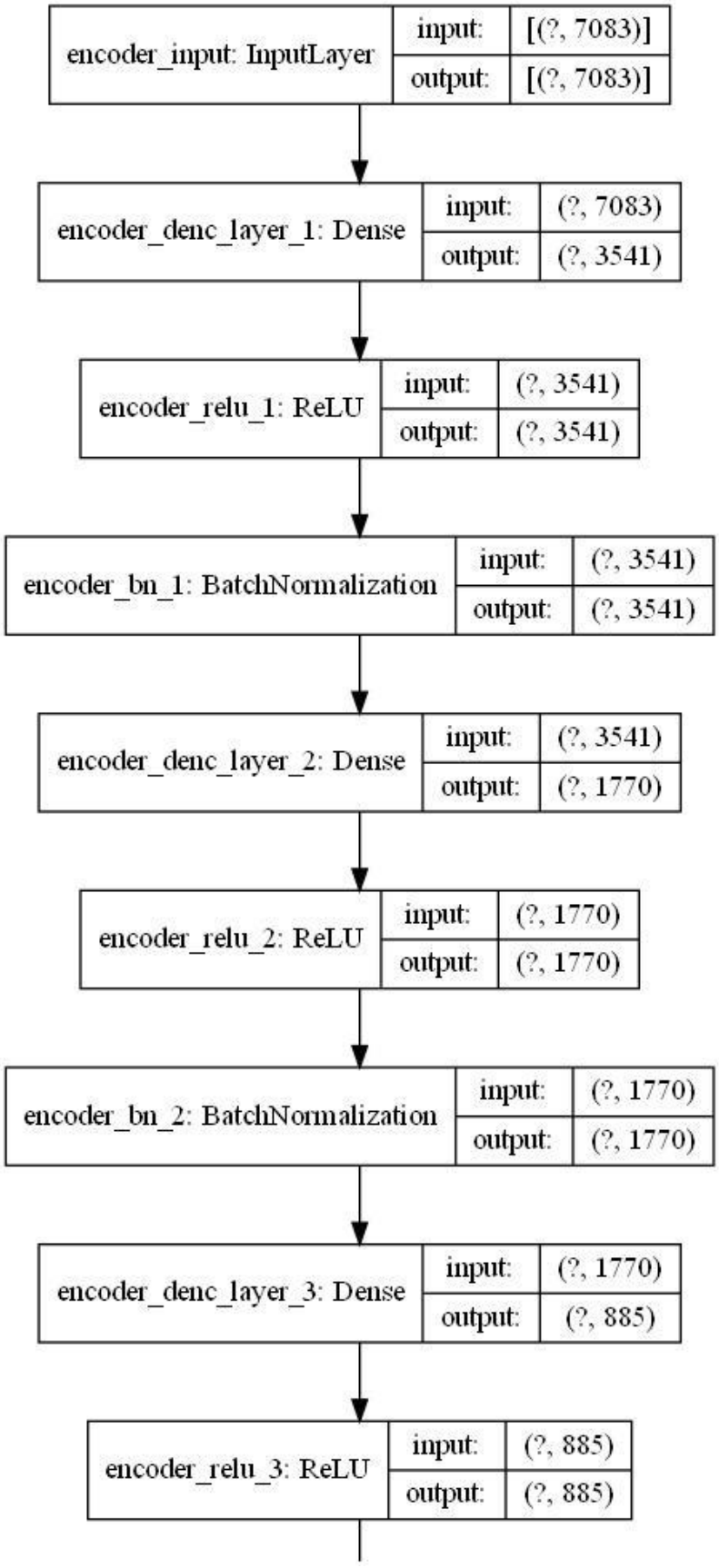
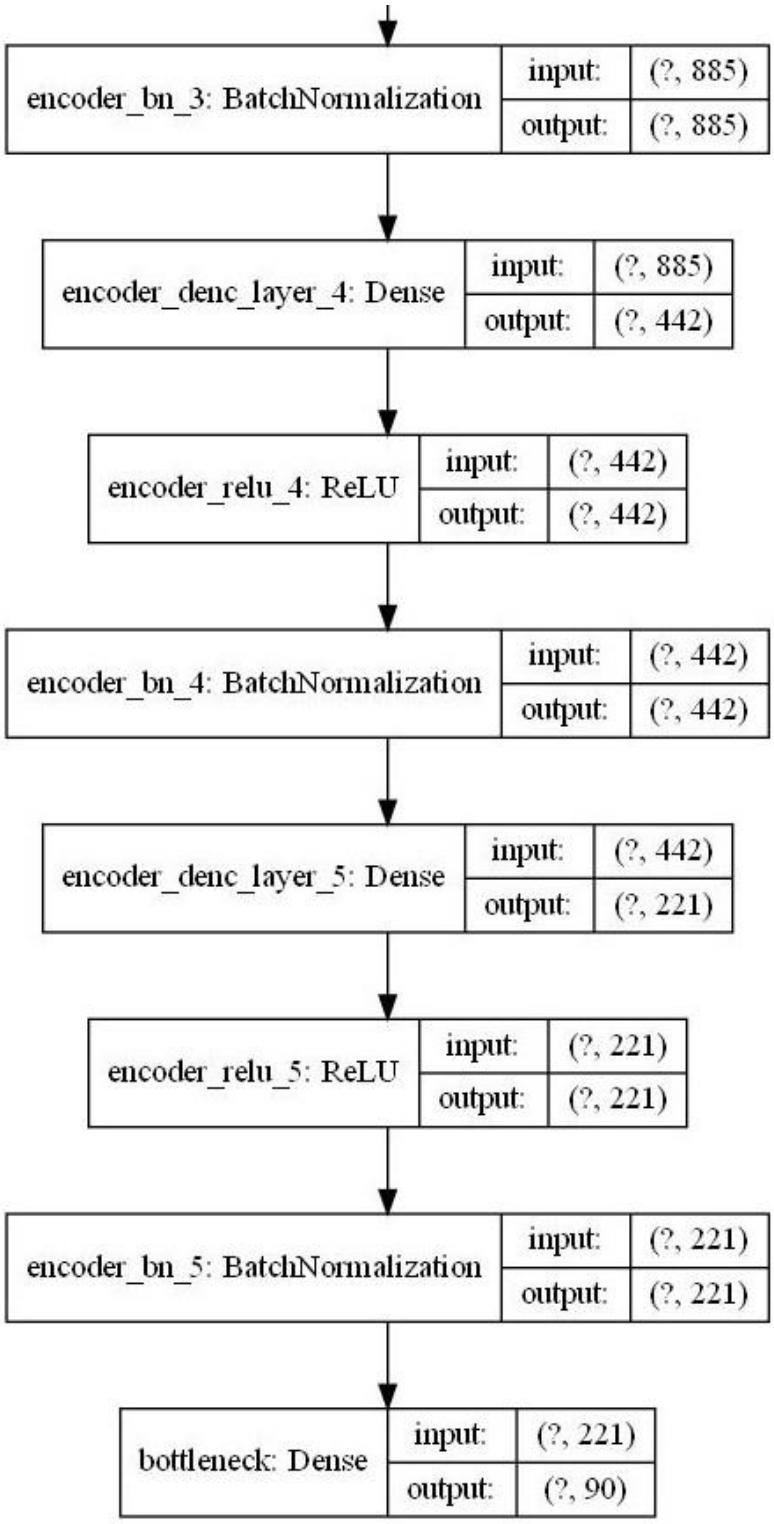


Figure (4.21): The encoder extracted from VTF5L_Framework


encoder_bn_3: BatchNormalization
input: (?, 885)
output: (?, 885)
encoder_denc_layer_4: Dense
input: (?, 885)
output: (?, 442)
encoder_relu_4: ReLU
input: (?, 442)
output: (?, 442)
encoder_bn_4: BatchNormalization
input: (?, 442)
output: (?, 442)
encoder_denc_layer_5: Dense
input: (?, 442)
output: (?, 221)
encoder_relu_5: ReLU
input: (?, 221)
output: (?, 221)
encoder_bn_5: BatchNormalization
input: (?, 221)
output: (?, 221)
bottleneck: Dense
input: (?, 221)
output: (?, 90)

4.2.4.7 CF1L _ Framework

This CDGAE framework is the one-layer framework with the ĀCX inputs and output, and number of bottleneck layer neurons equal to the number of dataset's communities.

Figure (4.22) shows the comprehensive schematic diagram for implementation of CF1L_Framework on Facebook government pages dataset, consist of (7057 node+12 feature =7069) Input and output, 1-encoder layer with number of neurons: (3534), 1-decoder layer with number of neurons: (3541), and the bottleneck layer with neurons (number of communities = 90).

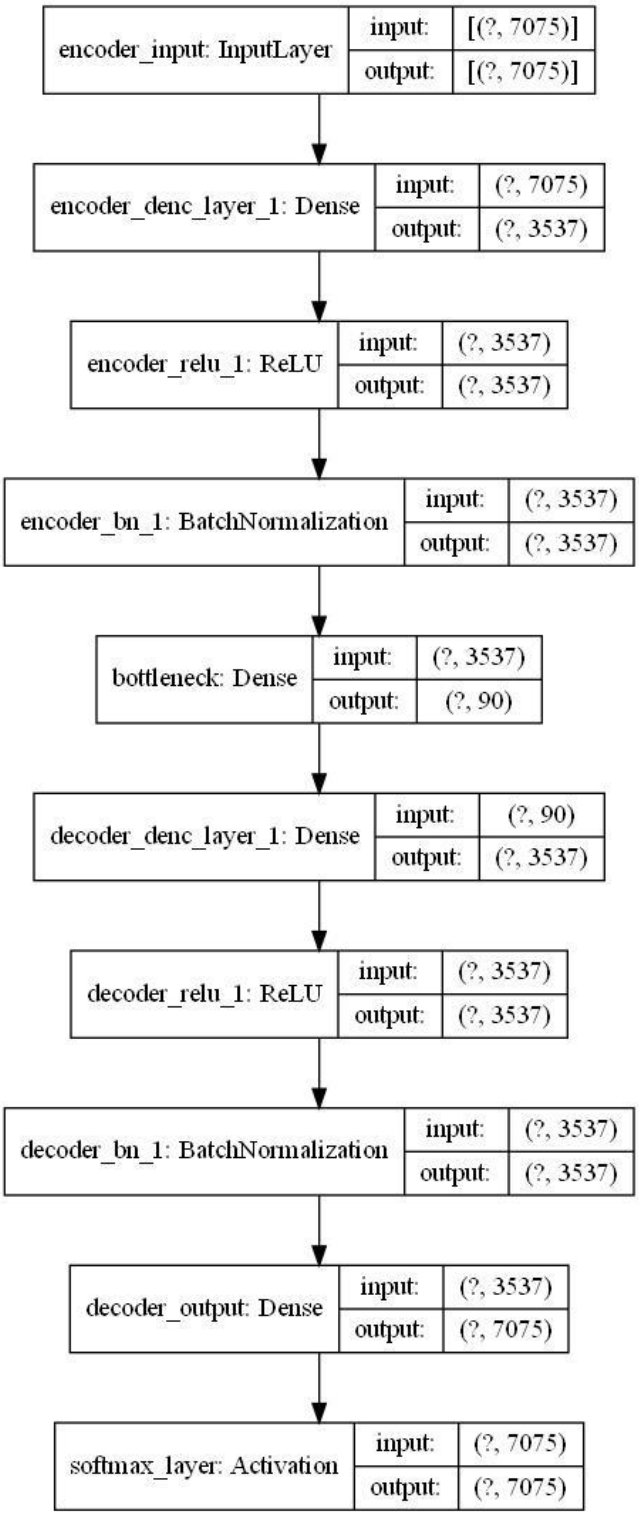


Figure (4.22): The implementation of CF1L_ Framework on the dataset.

Figure (4.23) shows the Curves diagram of training/validating loss function of CF1L_ Framework when applied on Facebook government pages dataset.

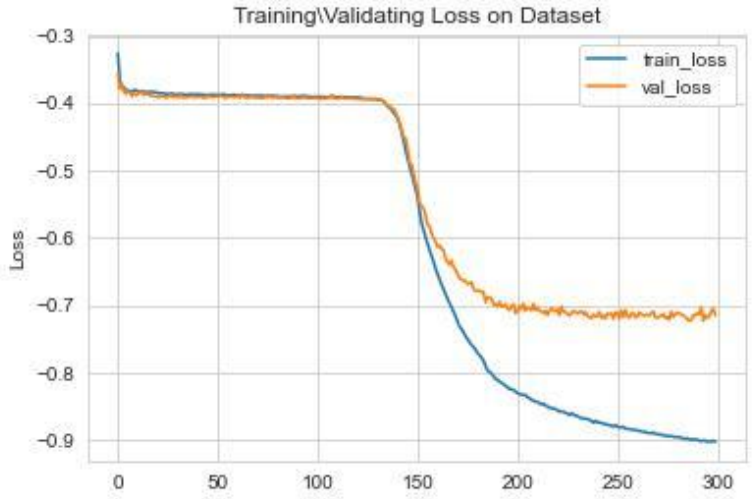


Figure (4.23): The diagram of the loss function curves of CF1L _ Framework

Figure (4.24) shows the encoder extracted from CF1L _ Framework after training it on Facebook government pages dataset.

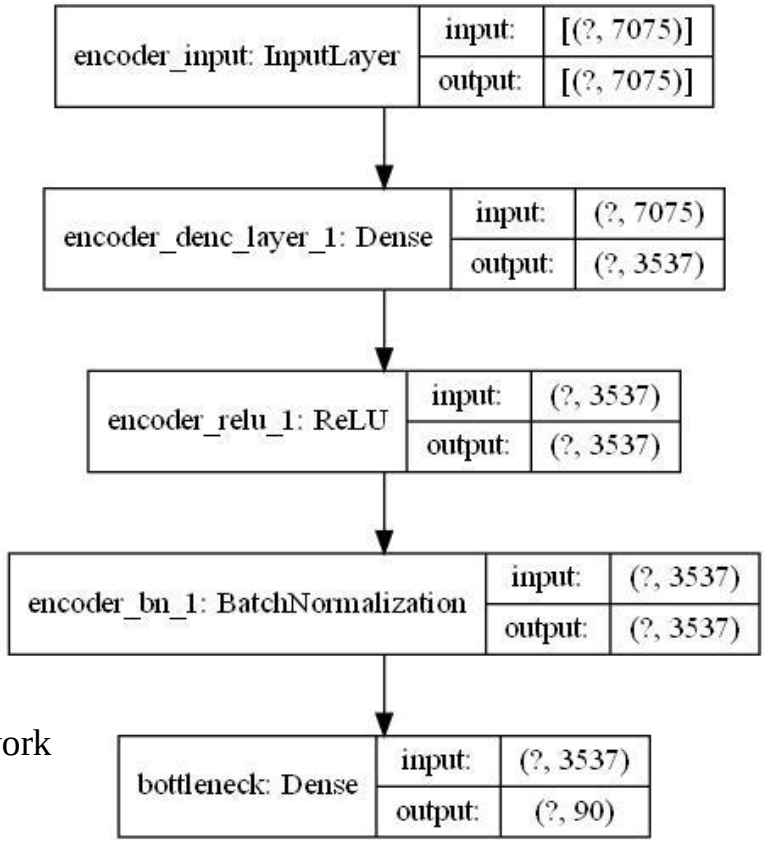


Figure (4.24): The encoder extracted from CF1L _ Framework

4.2.4.8 CF3L _ Framework

This CDGAE framework is the three-layers framework with the ĀCX inputs and output, and number of bottleneck layer neurons equal to the number of dataset's communities.

Figure (4.25) shows the comprehensive schematic diagram for implementation of CF3L_Framework on Facebook government pages dataset, consist of (7057 node+12 feature =7069) Input and output, 3-encoder layer with number of neurons: (3534, 1767, 883), 3-decoder layer with number of neurons: (883, 1767, 3541), and the bottleneck layer with neurons (number of communities = 90).

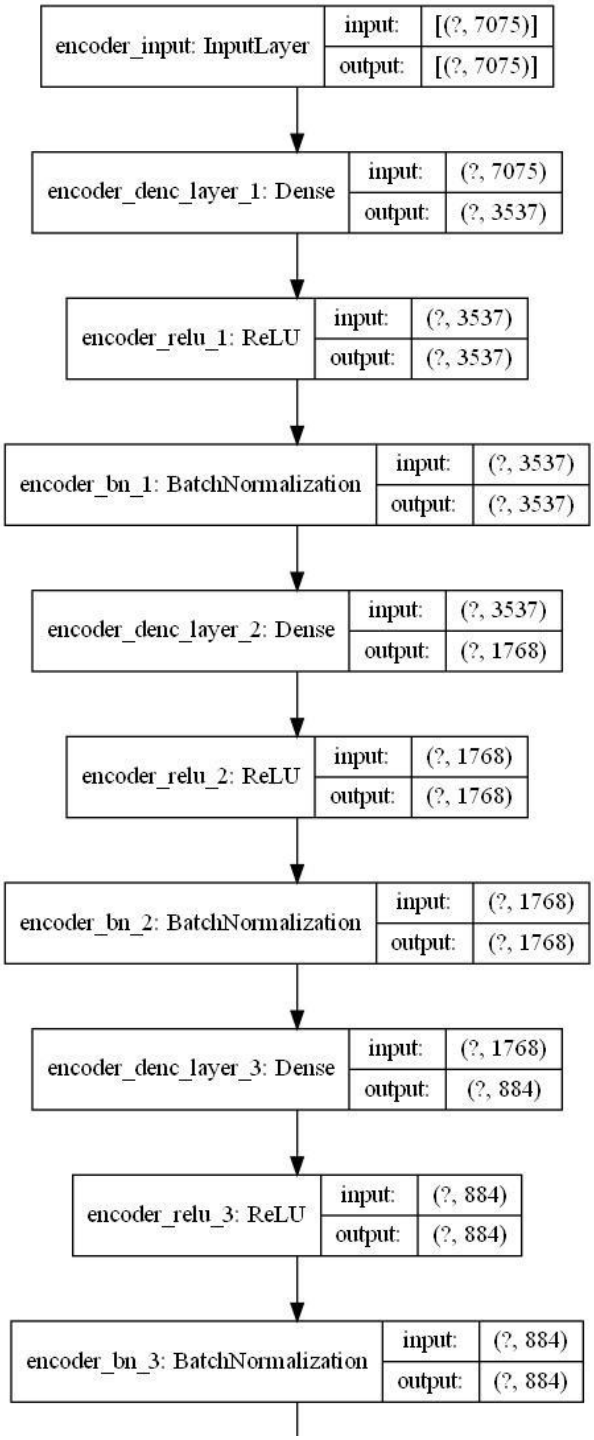


Figure (4.25): The implementation of CF3L_framework on the dataset.

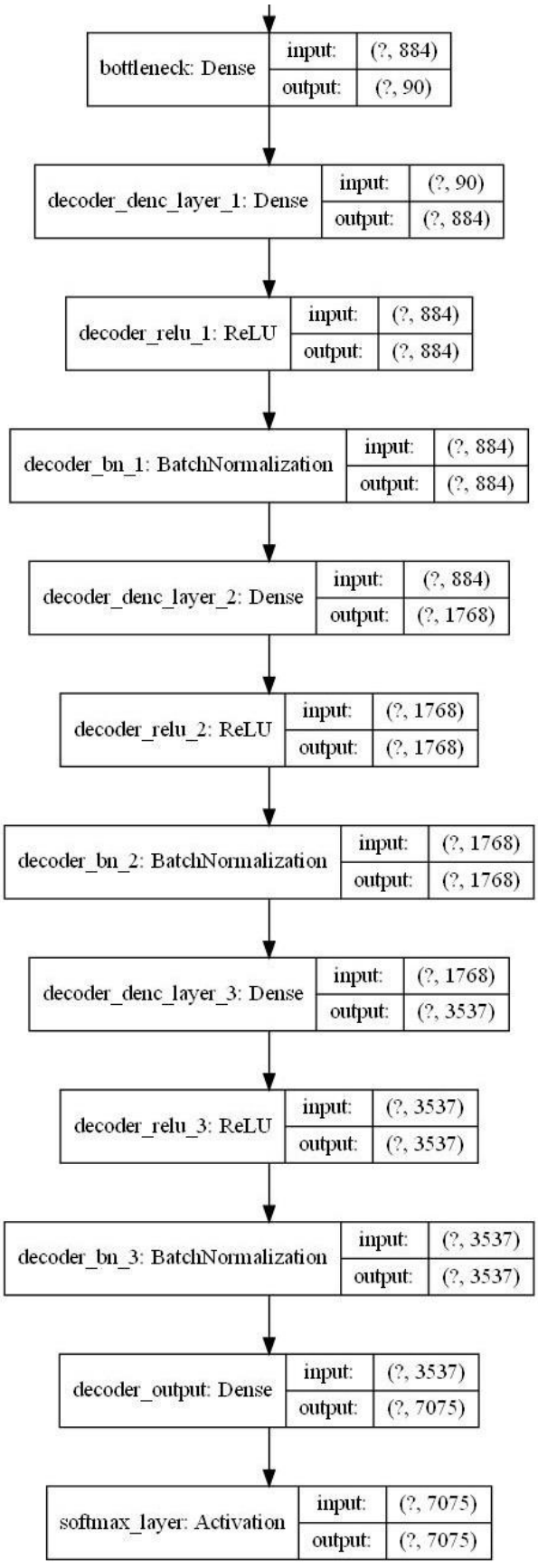

bottleneck: Dense
input: (?, 884)
output: (?, 90)
decoder_denc_layer_1: Dense
input: (?, 90)
output: (?, 884)
decoder_relu_1: ReLU
input: (?, 884)
output: (?, 884)
decoder_bn_1: BatchNormalization
input: (?, 884)
output: (?, 884)
decoder_denc_layer_2: Dense
input: (?, 884)
output: (?, 1768)
decoder_relu_2: ReLU
input: (?, 1768)
output: (?, 1768)
decoder_bn_2: BatchNormalization
input: (?, 1768)
output: (?, 1768)
decoder_denc_layer_3: Dense
input: (?, 1768)
output: (?, 3537)
decoder_relu_3: ReLU
input: (?, 3537)
output: (?, 3537)
decoder_bn_3: BatchNormalization
input: (?, 3537)
output: (?, 3537)
decoder_output: Dense
input: (?, 3537)
output: (?, 7075)
softmax_layer: Activation
input: (?, 7075)
output: (?, 7075)

Figure (4.26) shows the Curves diagram of training/validating loss function of CF3L_ Framework when applied on Facebook government pages dataset.

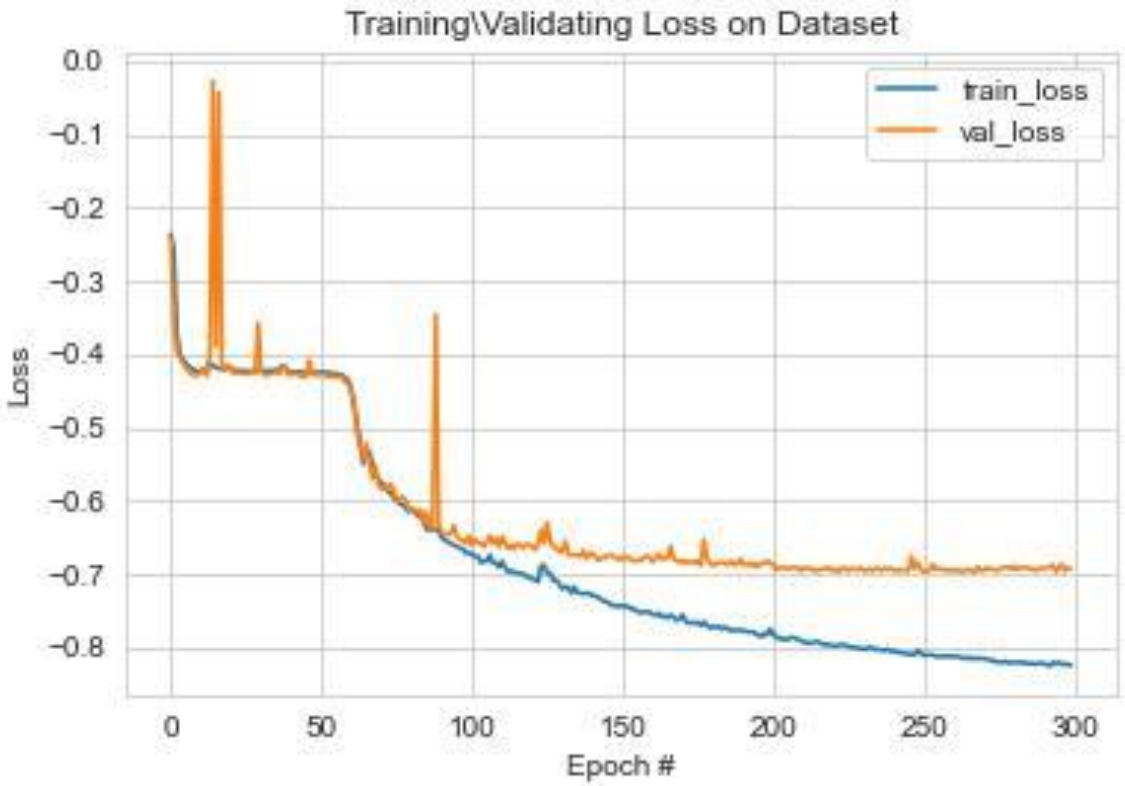


Figure (4.26): The diagram of the loss function curves of CF3L _ framework

Figure (4.27) shows the encoder extracted from CF3L _ Framework after training it on Facebook government pages dataset.

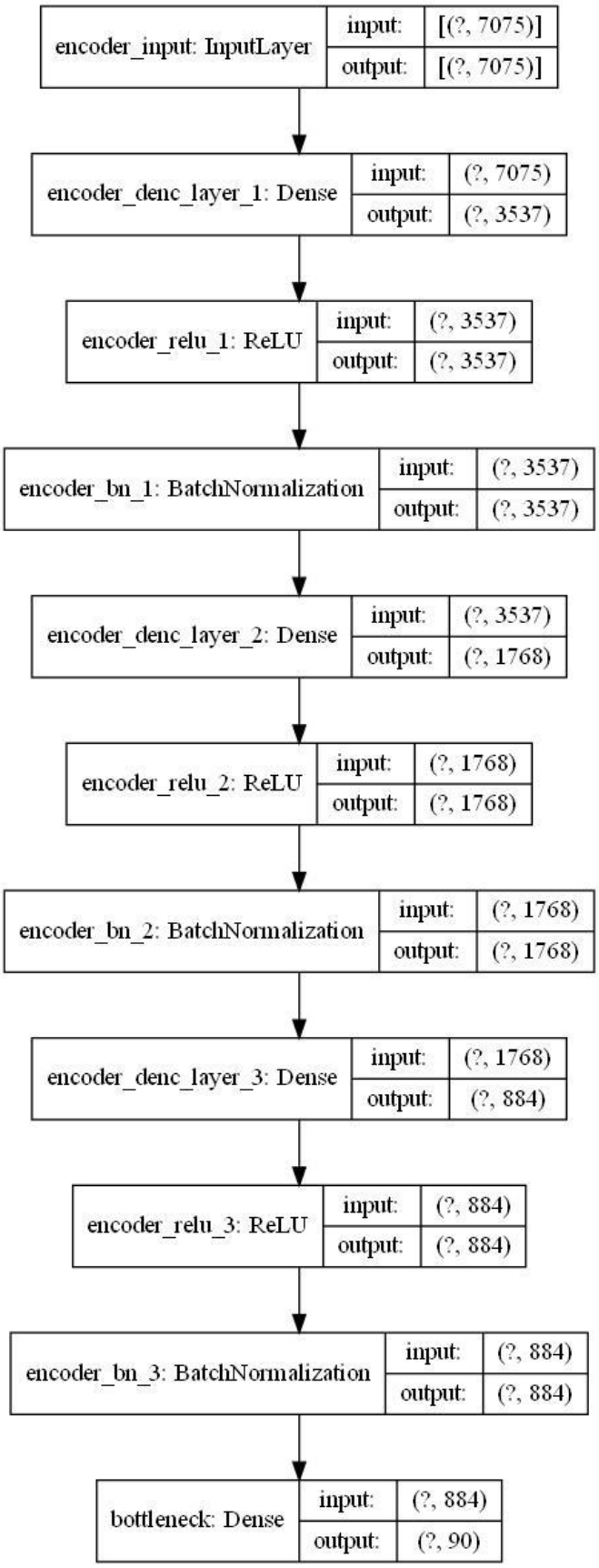


Figure (4.27): The encoder extracted from CF3L _ Framework

4.2.4.9 CF5L _ Framework

This CDGAE framework is the five-layers framework with the ĀCX inputs and output, and number of bottleneck layer communities.

Figure (4.28) shows the comprehensive schematic diagram for implementation of CF5L_Framework on Facebook government pages dataset, consist of (7057 node+12 feature =7069) Input and output, 5-encoder layer with number of neurons: (3534, 1767, 883, 441, 220), 5-decoder layer with number of neurons: (220, 441, 883, 1767, 3541), and the bottleneck layer with neurons (number of communities = 90).

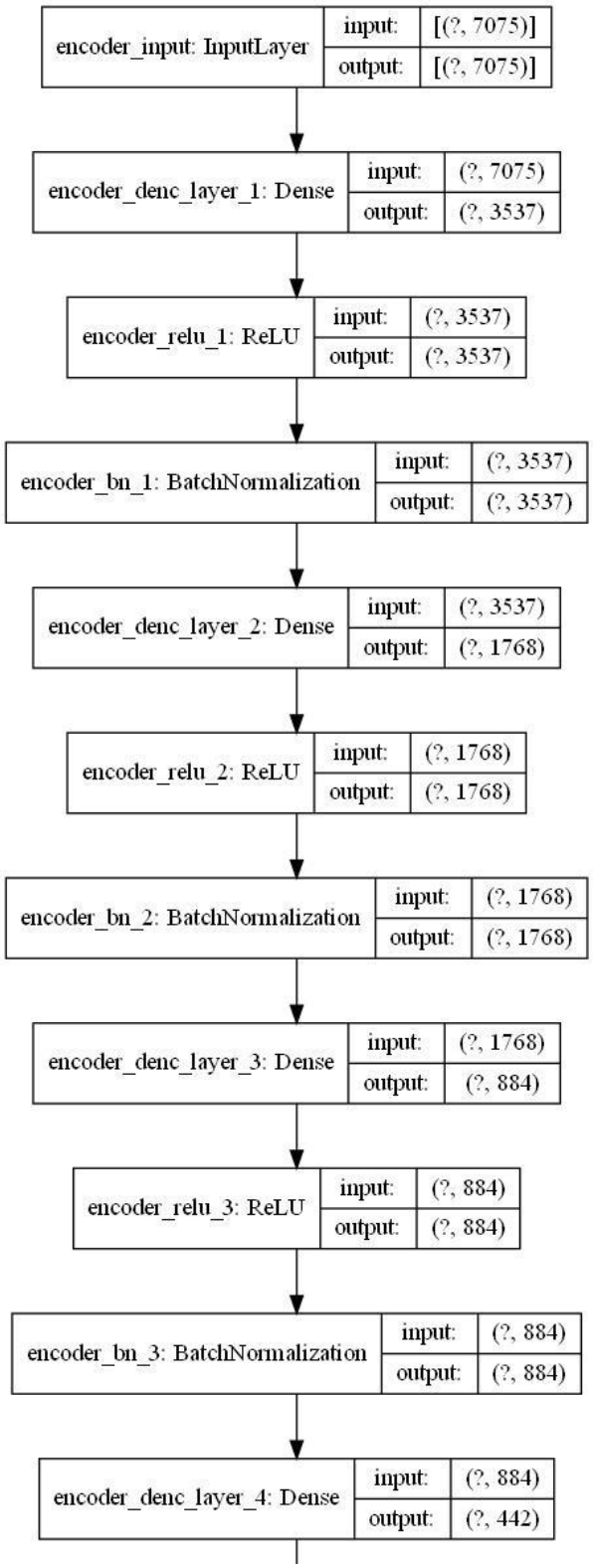
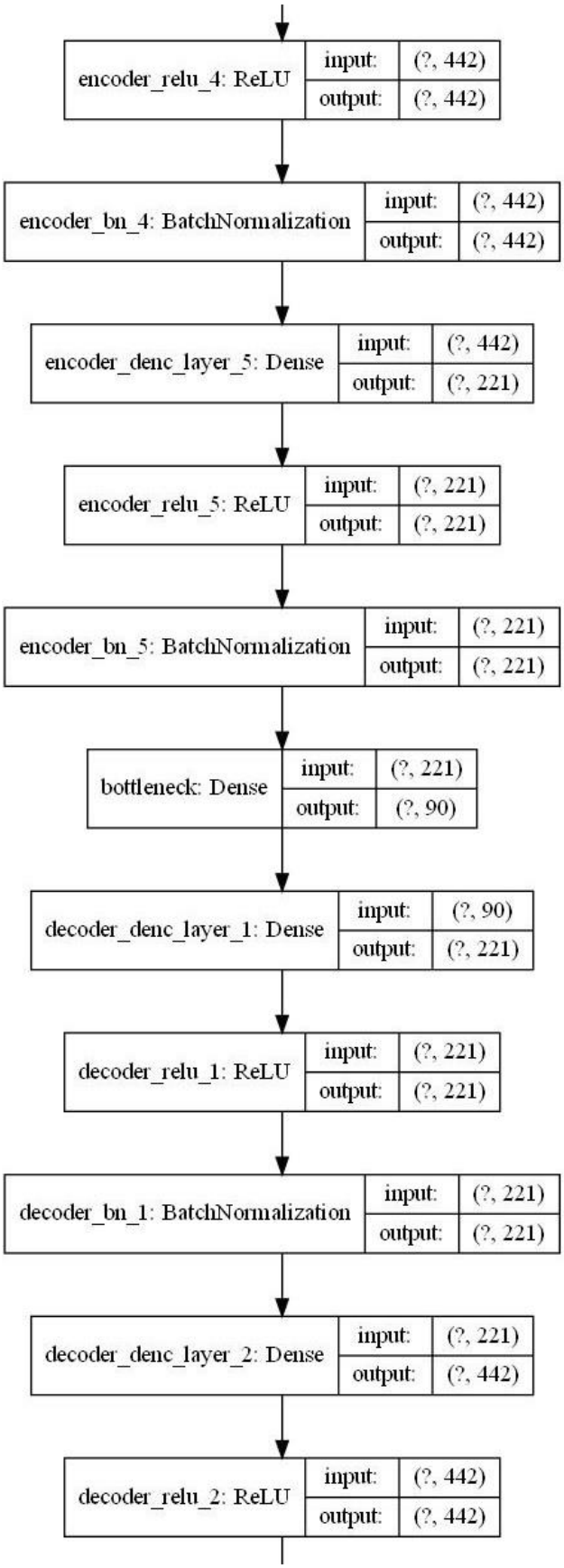


Figure (4.28): The implementation of CF5L_Framework on the dataset.


encoder_relu_4: ReLU
input: (?, 442)
output: (?, 442)
encoder_bn_4: BatchNormalization
input: (?, 442)
output: (?, 442)
encoder_denc_layer_5: Dense
input: (?, 442)
output: (?, 221)
encoder_relu_5: ReLU
input: (?, 221)
output: (?, 221)
encoder_bn_5: BatchNormalization
input: (?, 221)
output: (?, 221)
bottleneck: Dense
input: (?, 221)
output: (?, 90)
decoder_denc_layer_1: Dense
input: (?, 90)
output: (?, 221)
decoder_relu_1: ReLU
input: (?, 221)
output: (?, 221)
decoder_bn_1: BatchNormalization
input: (?, 221)
output: (?, 221)
decoder_denc_layer_2: Dense
input: (?, 221)
output: (?, 442)
decoder_relu_2: ReLU
input: (?, 442)
output: (?, 442)

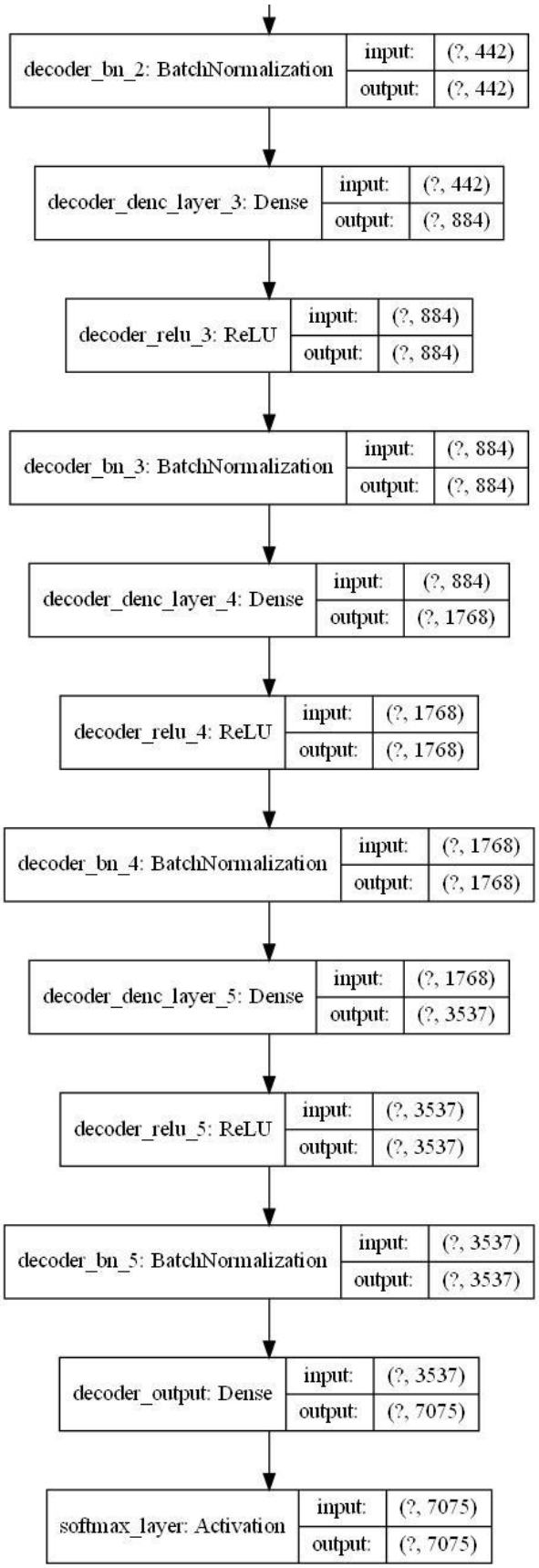

decoder_bn_2: BatchNormalization
input: (?, 442)
output: (?, 442)
decoder_denc_layer_3: Dense
input: (?, 442)
output: (?, 884)
decoder_relu_3: ReLU
input: (?, 884)
output: (?, 884)
decoder_bn_3: BatchNormalization
input: (?, 884)
output: (?, 884)
decoder_denc_layer_4: Dense
input: (?, 884)
output: (?, 1768)
decoder_relu_4: ReLU
input: (?, 1768)
output: (?, 1768)
decoder_bn_4: BatchNormalization
input: (?, 1768)
output: (?, 1768)
decoder_denc_layer_5: Dense
input: (?, 1768)
output: (?, 3537)
decoder_relu_5: ReLU
input: (?, 3537)
output: (?, 3537)
decoder_bn_5: BatchNormalization
input: (?, 3537)
output: (?, 3537)
decoder_output: Dense
input: (?, 3537)
output: (?, 7075)
softmax_layer: Activation
input: (?, 7075)
output: (?, 7075)

Figure (4.29) shows the Curves diagram of training/validating loss function of CF5L_ Framework when applied on Facebook government pages dataset.

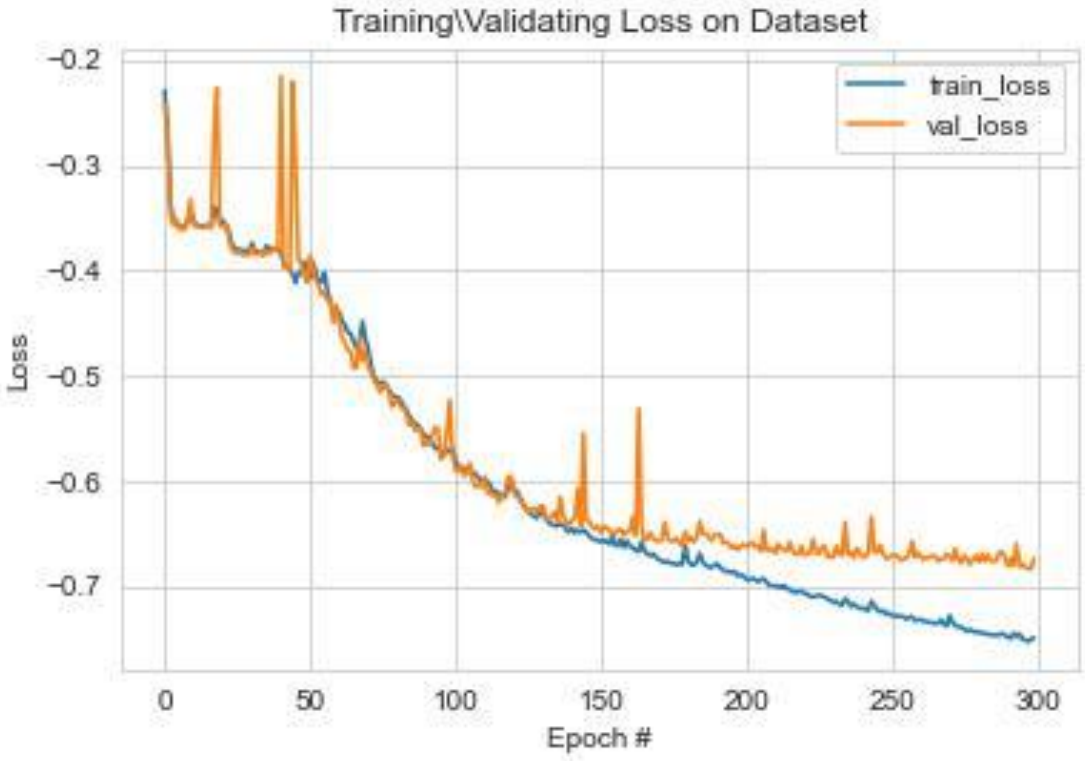


Figure (4.29): The diagram of the loss function curves of CF5L _ Framework

Figure (4.30) shows the encoder extracted from CF5L _ Framework after training it on Facebook government pages dataset.

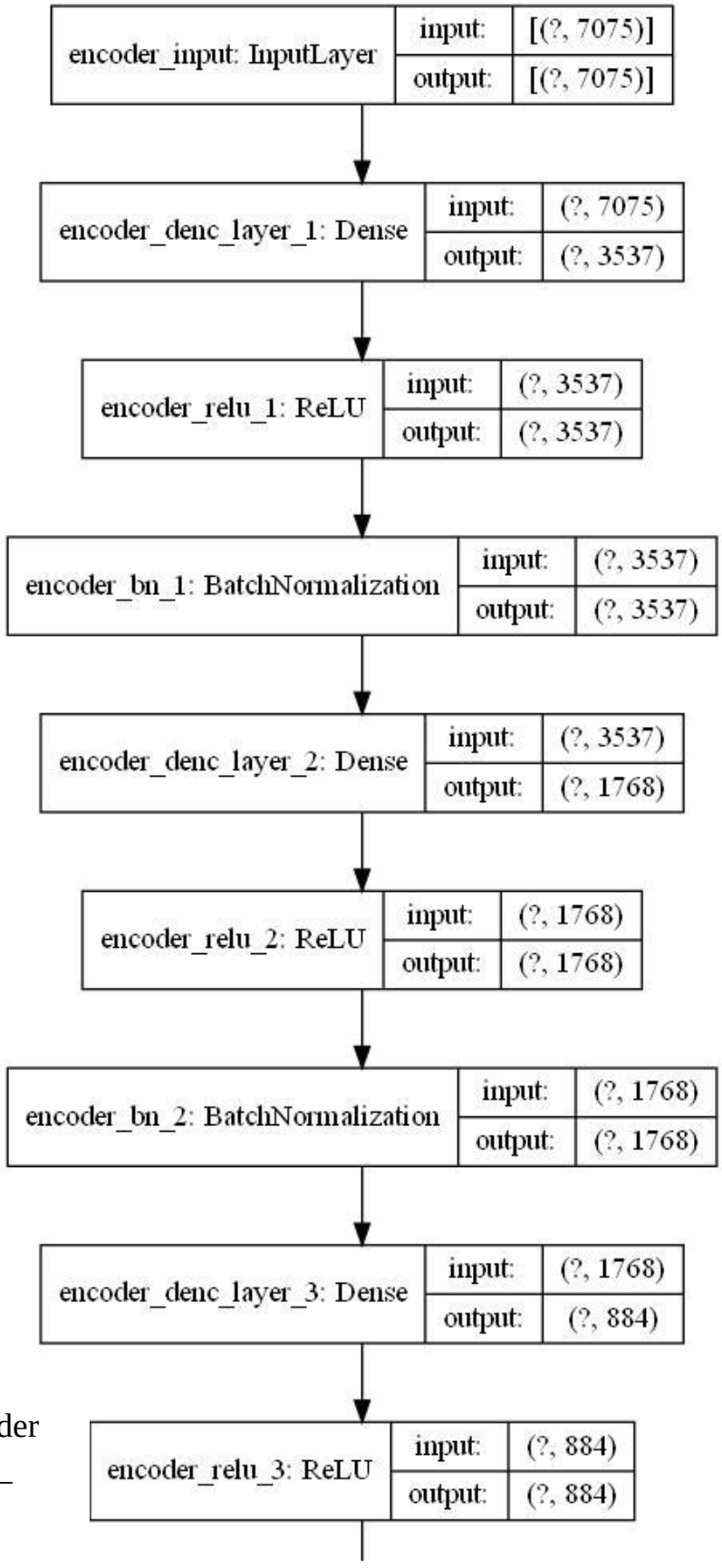
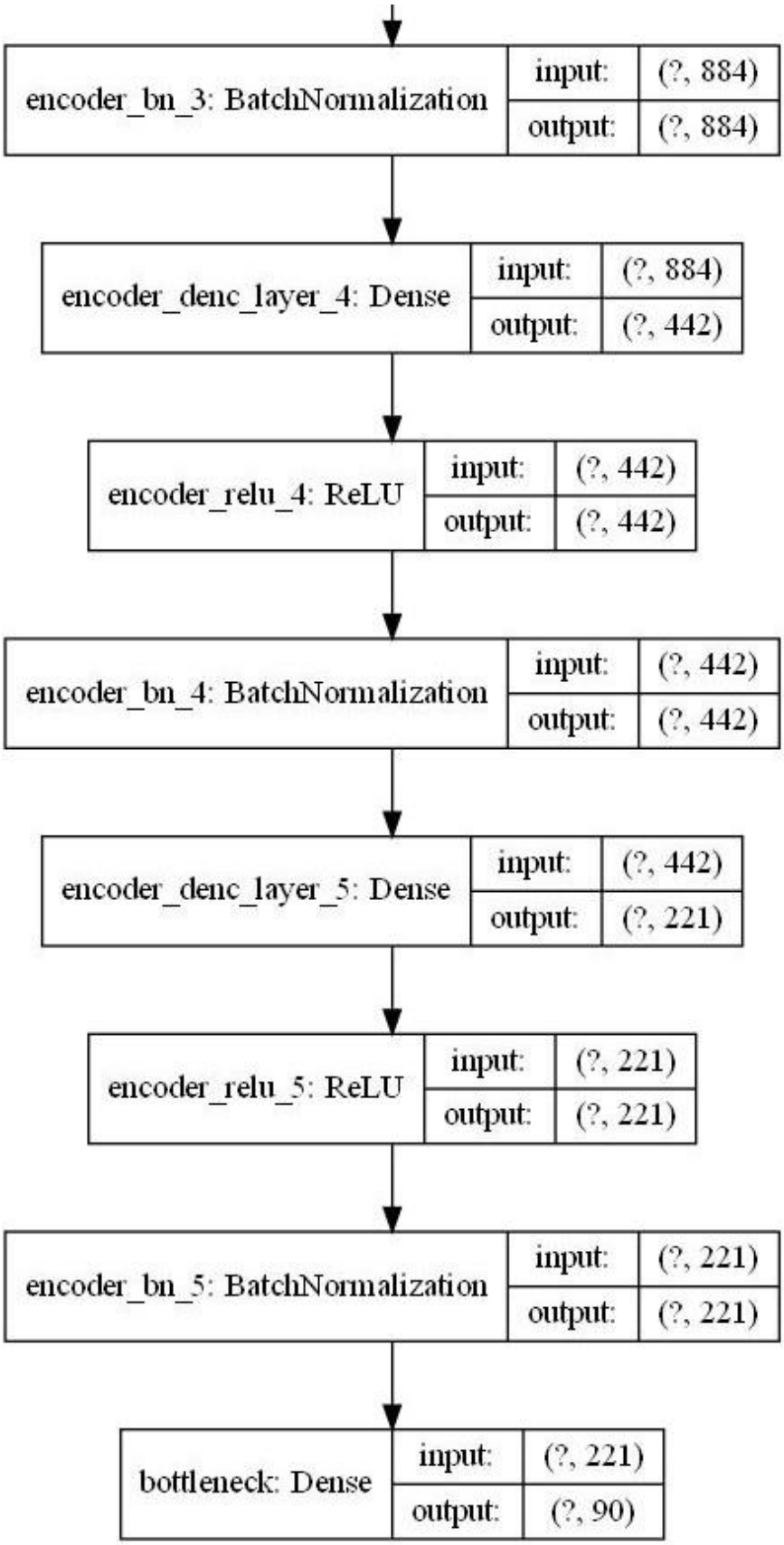


Figure (4.30): The encoder extracted from CF5L _ Framework


encoder_bn_3: BatchNormalization
input: (?, 884)
output: (?, 884)
encoder_denc_layer_4: Dense
input: (?, 884)
output: (?, 442)
encoder_relu_4: ReLU
input: (?, 442)
output: (?, 442)
encoder_bn_4: BatchNormalization
input: (?, 442)
output: (?, 442)
encoder_denc_layer_5: Dense
input: (?, 442)
output: (?, 221)
encoder_relu_5: ReLU
input: (?, 221)
output: (?, 221)
encoder_bn_5: BatchNormalization
input: (?, 221)
output: (?, 221)
bottleneck: Dense
input: (?, 221)
output: (?, 90)

4.2.4.10 FL1L_ Framework

This CDGAE framework is the one-layer framework with the A inputs and output, and number of bottleneck layer neurons equal to the number of dataset's communities.

Figure (4.31) shows the comprehensive schematic diagram for implementation of FL1L_Framework on Facebook government pages dataset, consist of 7057 nodes input and output, 1-encoder layer with number of neurons: (3528), 1-decoder layer with number of neurons: (3528), and the bottleneck layer with neurons (number of communities = 90).

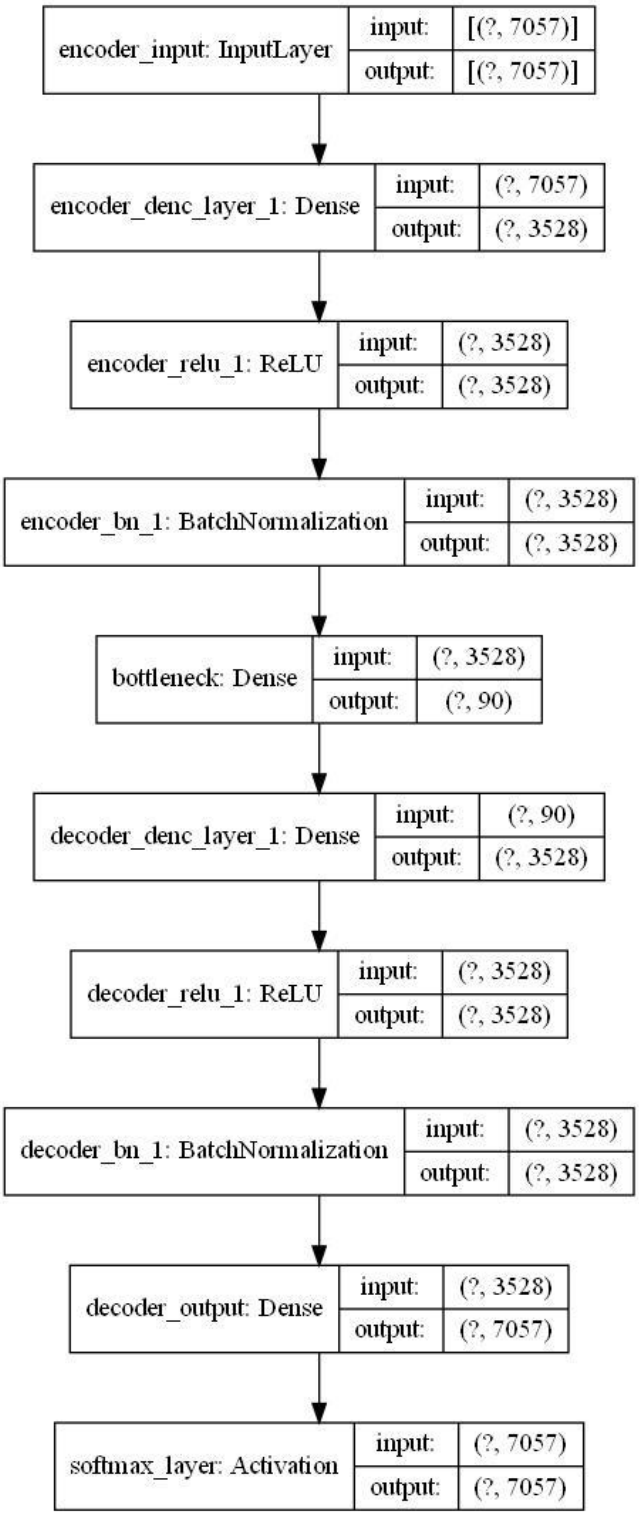


Figure (4.31): The implementation of FL1L_Framework on the dataset.

Figure (4.32) shows the Curves diagram of training/validating loss function of FL1L_ Framework when applied on Facebook government pages dataset.

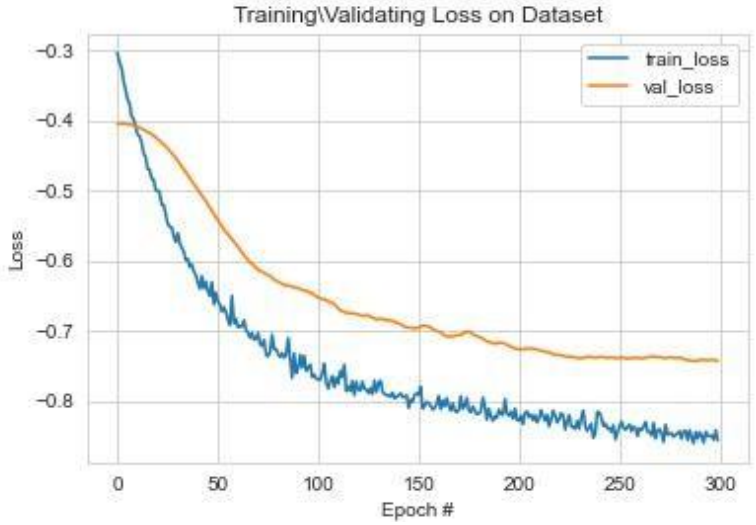


Figure (4.32): The diagram of the loss function curves of FL1L _ Framework

Figure (4.33) shows the encoder extracted from FL1L _ Framework after training it on Facebook government pages dataset.

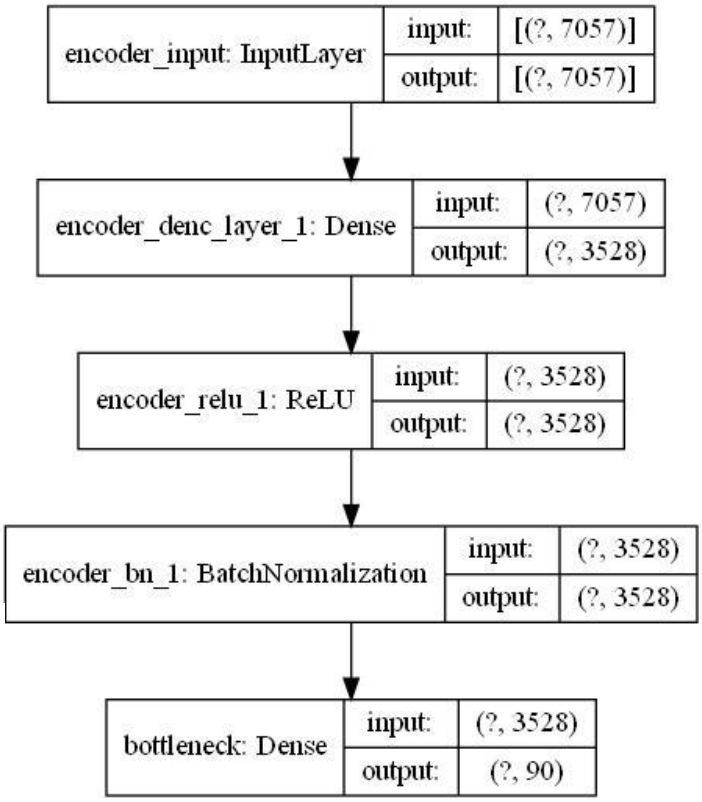


Figure (4.33): The encoder extracted from FL1L_Framework

4.2.4.11 FL3L_ Framework

This CDGAE framework is the one-layer framework with the A inputs and output, and number of bottleneck layer neurons equal to the number of dataset's communities.

Figure (4.34) shows the comprehensive schematic diagram for implementation of FL3L_framework on Facebook government pages dataset, consist of 7057 nodes input and output, 3-encoder layer with number of neurons: (3528, 1764, 882), 3-decoder layer with number of neurons: (882, 1764, 3528), and the bottleneck layer with neurons (number of communities = 90).

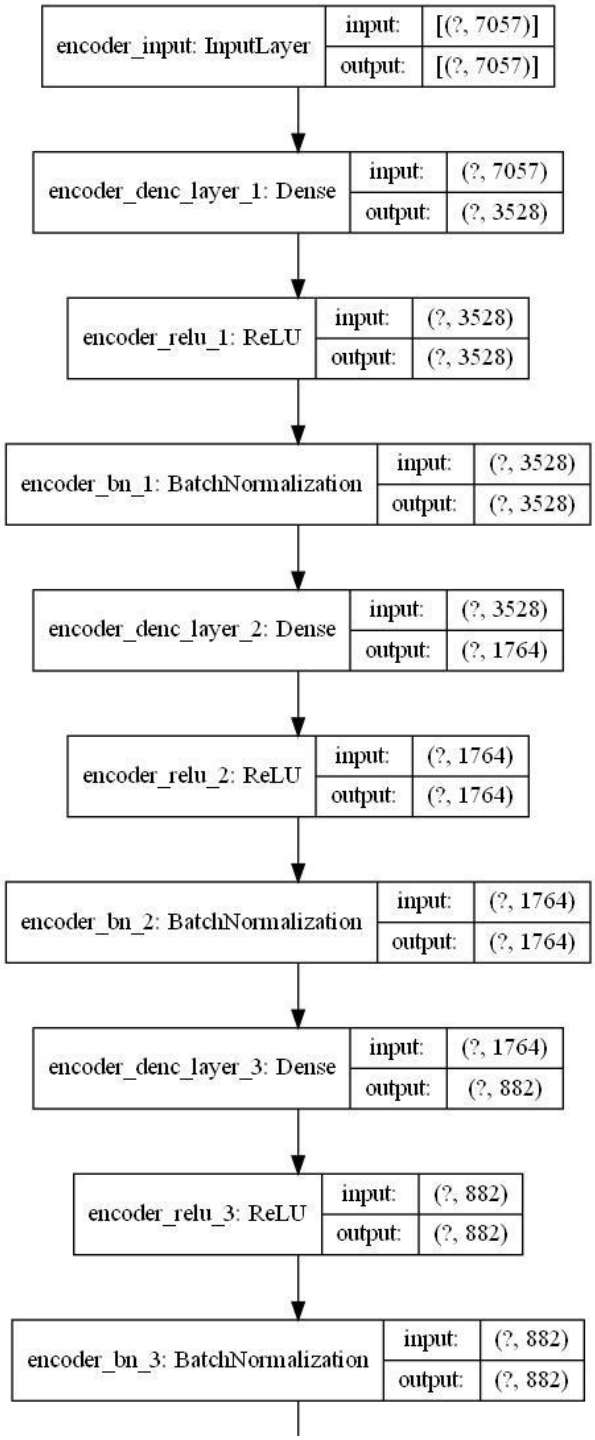


Figure (4.34): The implementation of FL3L_ framework on the dataset.

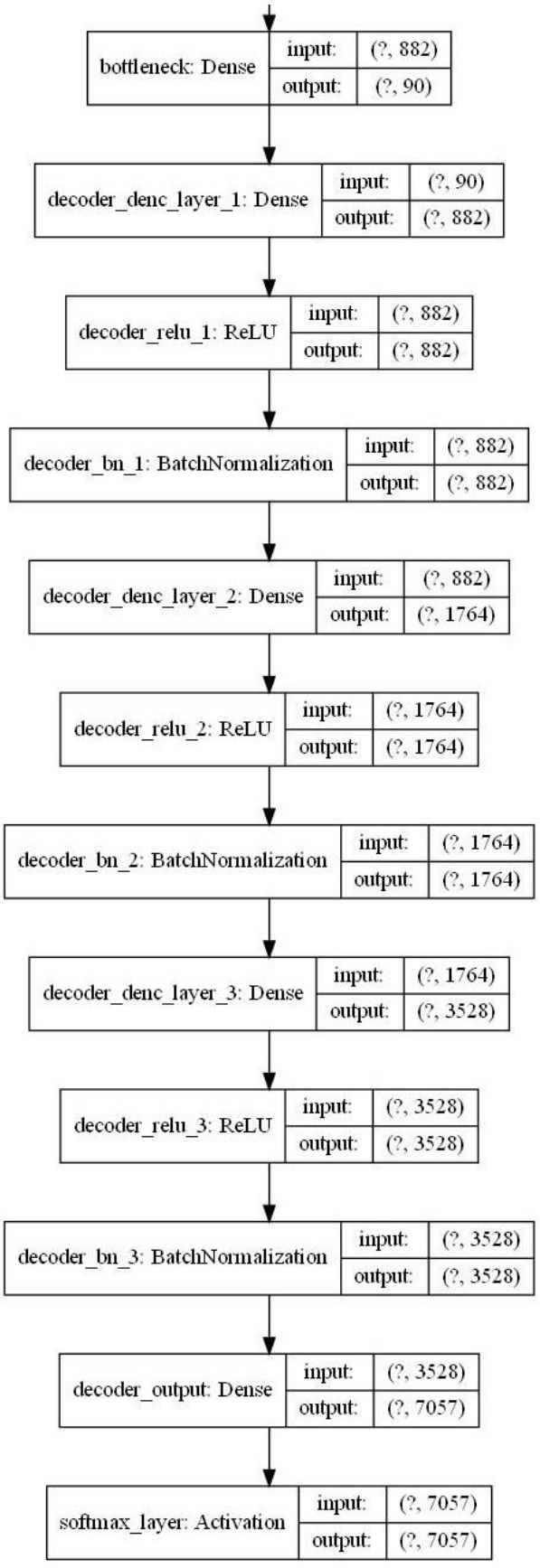

bottleneck: Dense
input: (?, 882)
output: (?, 90)
decoder_denc_layer_1: Dense
input: (?, 90)
output: (?, 882)
decoder_relu_1: ReLU
input: (?, 882)
output: (?, 882)
decoder_bn_1: BatchNormalization
input: (?, 882)
output: (?, 882)
decoder_denc_layer_2: Dense
input: (?, 882)
output: (?, 1764)
decoder_relu_2: ReLU
input: (?, 1764)
output: (?, 1764)
decoder_bn_2: BatchNormalization
input: (?, 1764)
output: (?, 1764)
decoder_denc_layer_3: Dense
input: (?, 1764)
output: (?, 3528)
decoder_relu_3: ReLU
input: (?, 3528)
output: (?, 3528)
decoder_bn_3: BatchNormalization
input: (?, 3528)
output: (?, 3528)
decoder_output: Dense
input: (?, 3528)
output: (?, 7057)
softmax_layer: Activation
input: (?, 7057)
output: (?, 7057)

Figure (4.35) shows the Curves diagram of training/validating loss function of FL3L_ Framework when applied on Facebook government pages dataset.

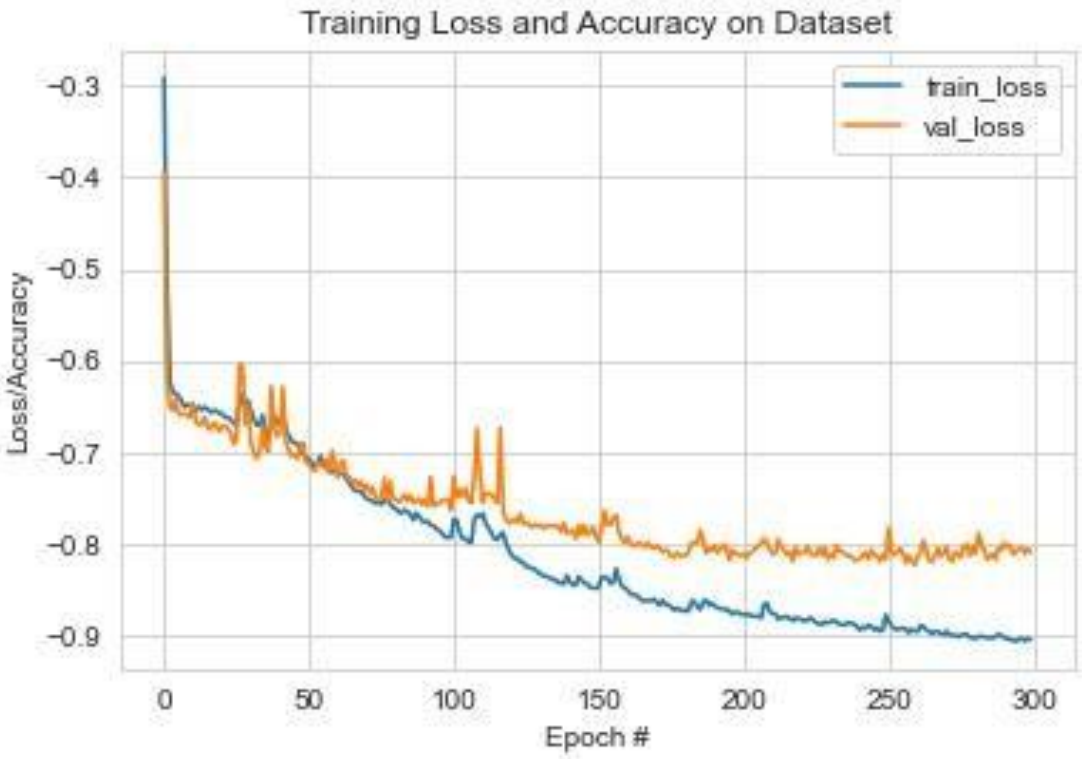


Figure (4.35): The diagram of the loss function curves of FL3L _ framework

Figure (4.36) shows the encoder extracted from FL3L _ Framework after training it on Facebook government pages dataset.

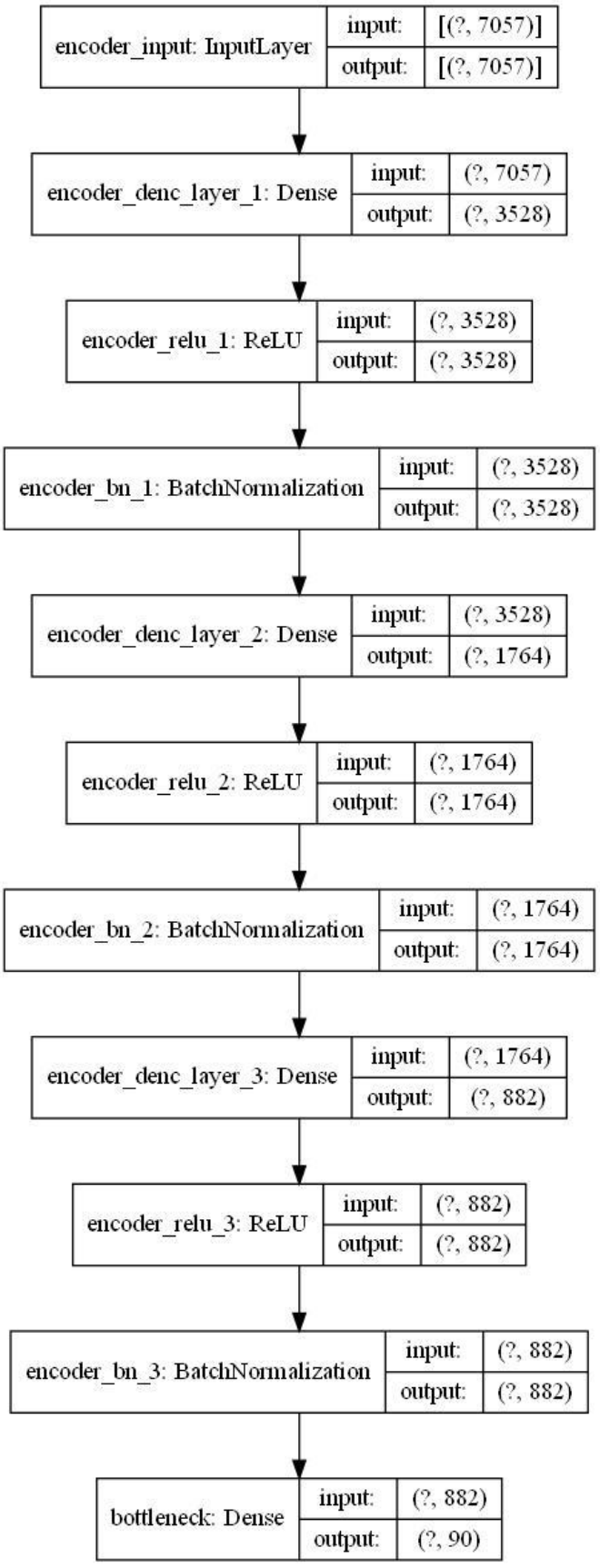


Figure (4.36): The encoder extracted from FL3L_framework

4.2.4.12 FL5L_ Framework

This CDGAE framework is the one-layer framework with the A inputs and output, and number of bottleneck layer neurons equal to the number of dataset's communities

Figure (4.37) shows the comprehensive schematic diagram for implementation of FL5L_framework on Facebook government pages dataset, consist of 7057 nodes input and output, 5-encoder layer with number of neurons: (3528, 1764, 882, 441, 220), 5-decoder layer with number of neurons: (220, 441, 882, 1764, 3528), and the bottleneck layer with neurons (number of communities = 90).

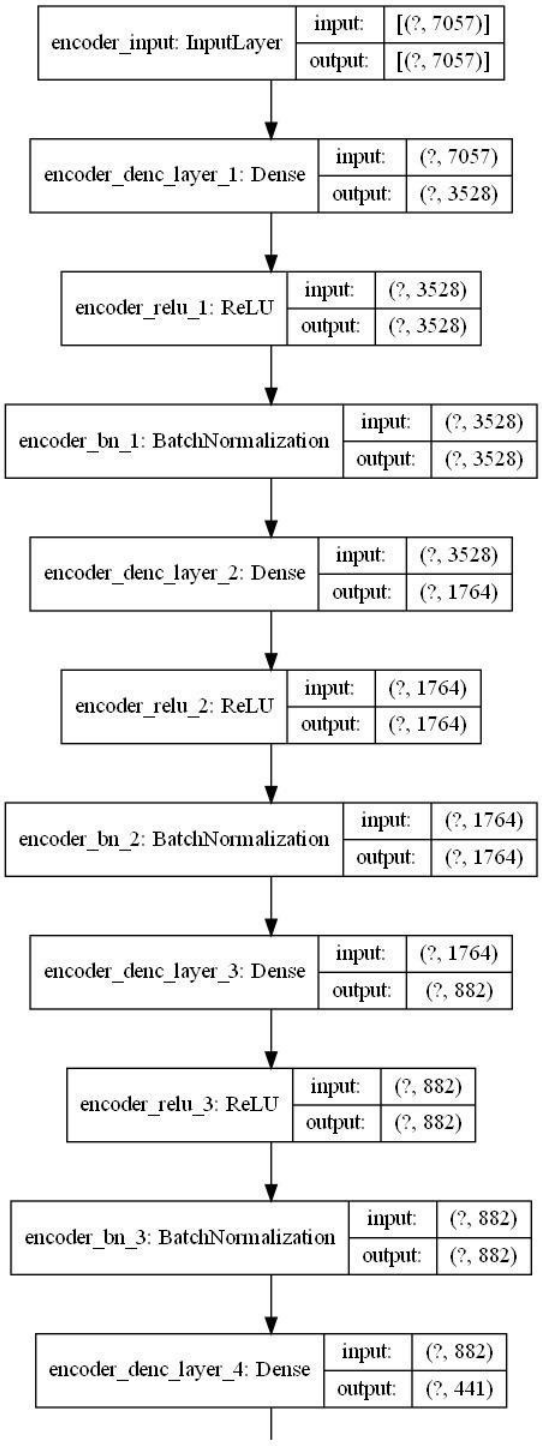
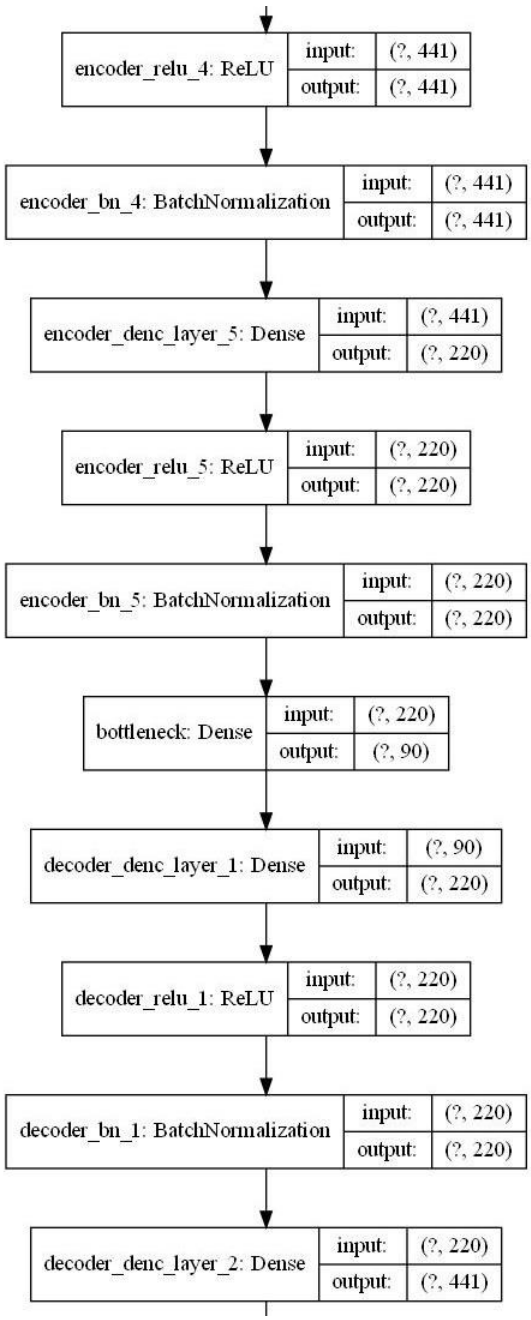


Figure (4.37): The implementation of FL5L_ Framework on the dataset.


encoder_relu_4: ReLU
input: (?, 441)
output: (?, 441)
encoder_bn_4: BatchNormalization
input: (?, 441)
output: (?, 441)
encoder_denc_layer_5: Dense
input: (?, 441)
output: (?, 220)
encoder_relu_5: ReLU
input: (?, 220)
output: (?, 220)
encoder_bn_5: BatchNormalization
input: (?, 220)
output: (?, 220)
bottleneck: Dense
input: (?, 220)
output: (?, 90)
decoder_denc_layer_1: Dense
input: (?, 90)
output: (?, 220)
decoder_relu_1: ReLU
input: (?, 220)
output: (?, 220)
decoder_bn_1: BatchNormalization
input: (?, 220)
output: (?, 220)
decoder_denc_layer_2: Dense
input: (?, 220)
output: (?, 441)

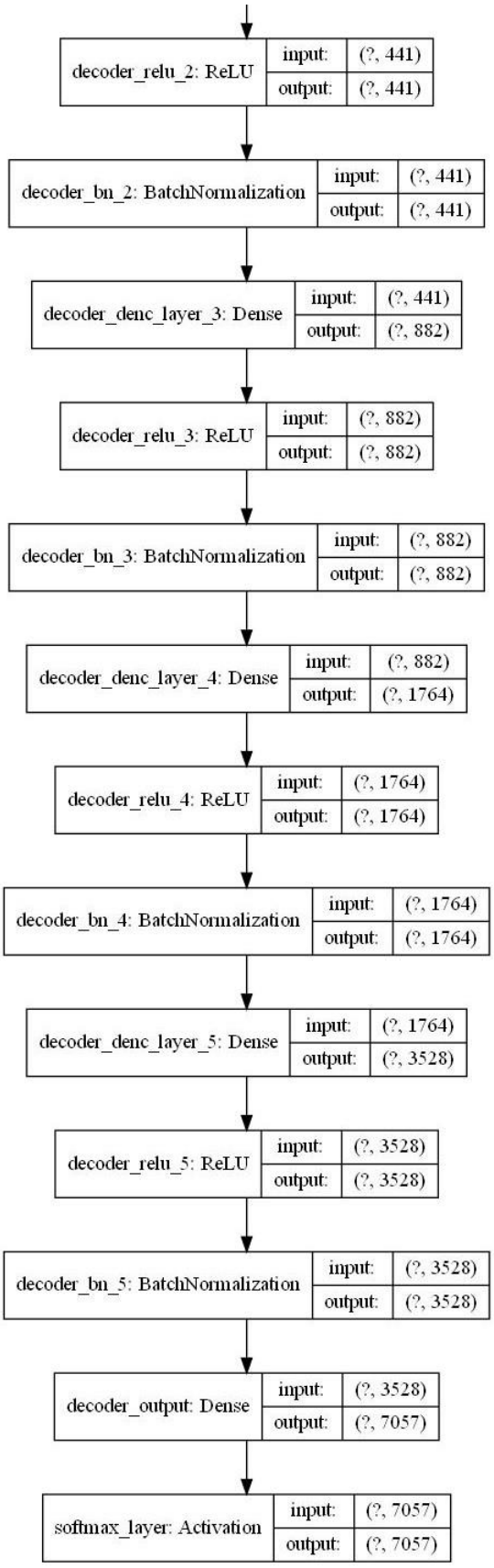

decoder_relu_2: ReLU
input: (?, 441)
output: (?, 441)
decoder_bn_2: BatchNormalization
input: (?, 441)
output: (?, 441)
decoder_denc_layer_3: Dense
input: (?, 441)
output: (?, 882)
decoder_relu_3: ReLU
input: (?, 882)
output: (?, 882)
decoder_bn_3: BatchNormalization
input: (?, 882)
output: (?, 882)
decoder_denc_layer_4: Dense
input: (?, 882)
output: (?, 1764)
decoder_relu_4: ReLU
input: (?, 1764)
output: (?, 1764)
decoder_bn_4: BatchNormalization
input: (?, 1764)
output: (?, 1764)
decoder_denc_layer_5: Dense
input: (?, 1764)
output: (?, 3528)
decoder_relu_5: ReLU
input: (?, 3528)
output: (?, 3528)
decoder_bn_5: BatchNormalization
input: (?, 3528)
output: (?, 3528)
decoder_output: Dense
input: (?, 3528)
output: (?, 7057)
softmax_layer: Activation
input: (?, 7057)
output: (?, 7057)

Figure (4.38) shows the Curves diagram of training/validating loss function of FL5L_ Framework when applied on Facebook government pages dataset.

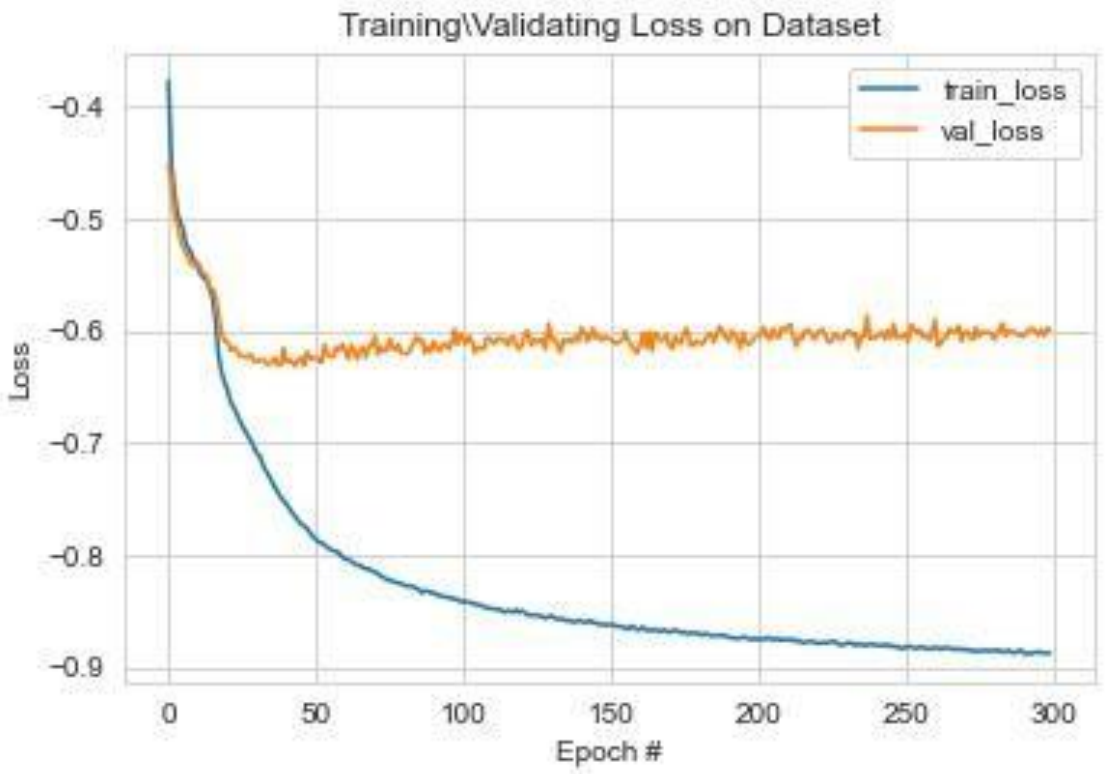


Figure (4.38): The diagram of the loss function curves of FL5L _ Framework

Figure (4.39) shows the encoder extracted from FL5L_Framework after training it on Facebook government pages dataset.

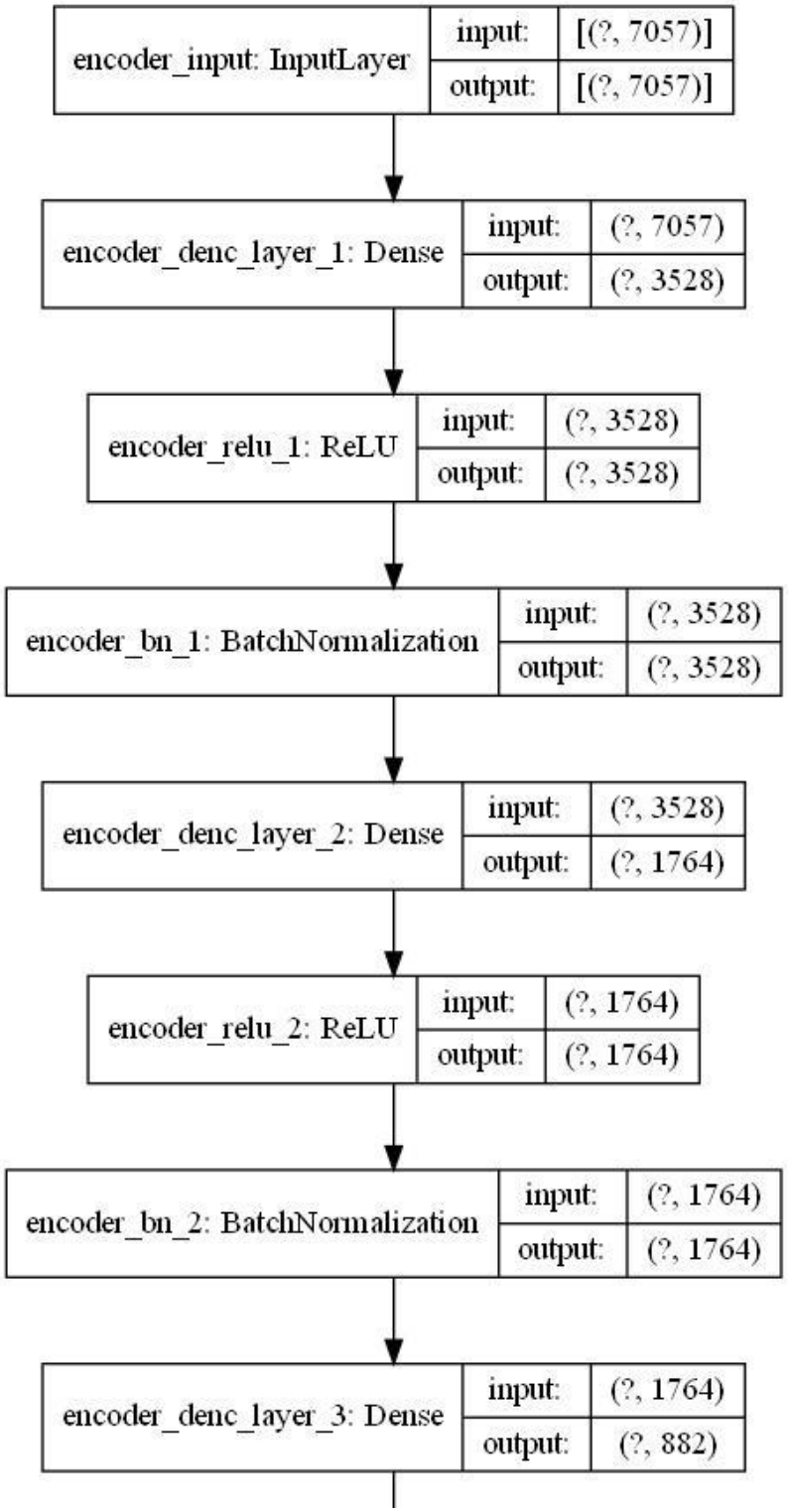
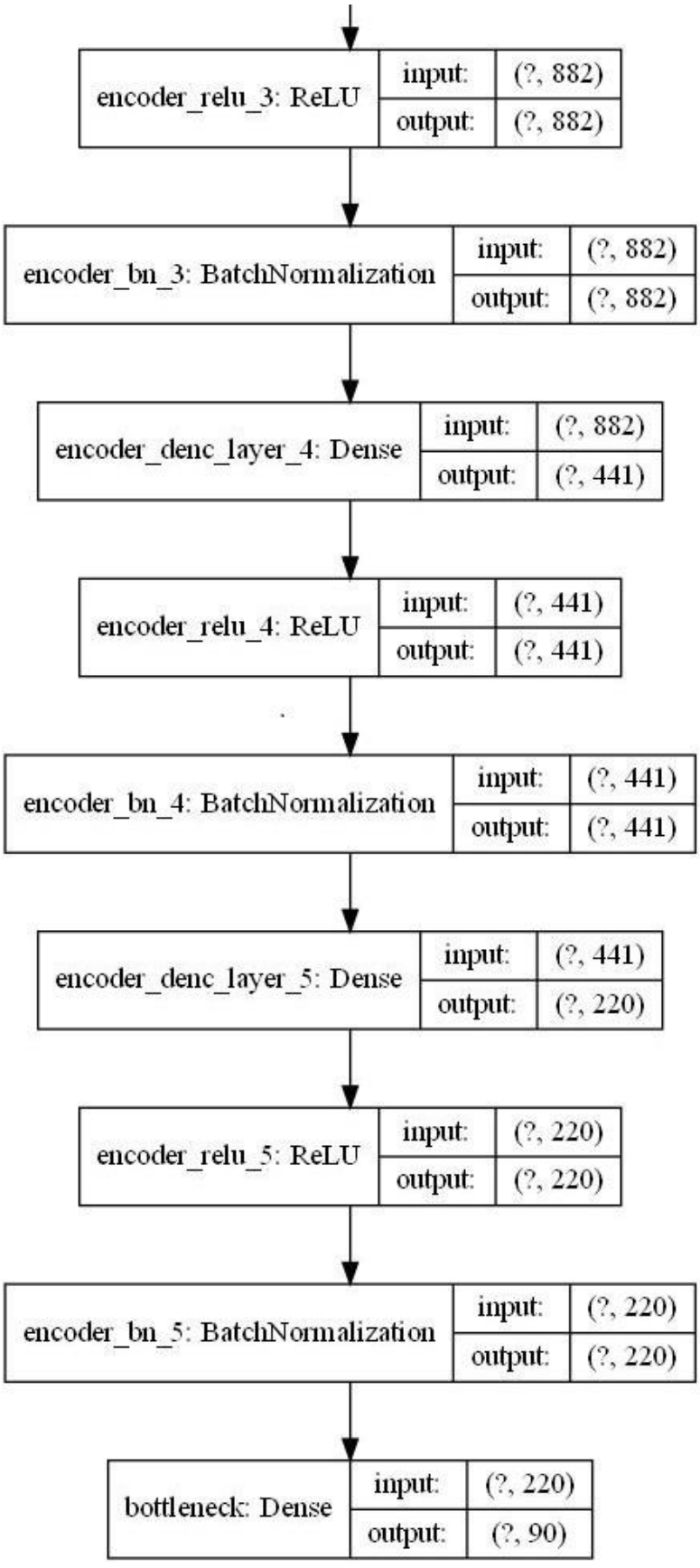


Figure (4.39): The encoder extracted from FL5L_framework


encoder_relu_3: ReLU
input: (?, 882)
output: (?, 882)
encoder_bn_3: BatchNormalization
input: (?, 882)
output: (?, 882)
encoder_denc_layer_4: Dense
input: (?, 882)
output: (?, 441)
encoder_relu_4: ReLU
input: (?, 441)
output: (?, 441)
encoder_bn_4: BatchNormalization
input: (?, 441)
output: (?, 441)
encoder_denc_layer_5: Dense
input: (?, 441)
output: (?, 220)
encoder_relu_5: ReLU
input: (?, 220)
output: (?, 220)
encoder_bn_5: BatchNormalization
input: (?, 220)
output: (?, 220)
bottleneck: Dense
input: (?, 220)
output: (?, 90)

4.2.5 The number of communities after applying the CDGAE Frameworks

When CDGAE Framework s are applied to datasets, a different number of communities is produced, where it is less than or equal to the number communities computed, the number of communities resultant each CDGAE framework for each dataset are listed in table (4.4).

Table (4.4): Communities numbers according to CDGAE Framework s

CDGAE frameworks	Facebook government pages
FL1L	90
FL3L	41
FL5L	15
AF1L	66
AF3L	75
AF5L	25
VTF1L	65
VTF3L	24
VTF5L	31
CF1L	53
CF3L	69
CF5L	45

4.3 Evaluation the community detection Framework.

After detecting the nodes for each community, the performance of the framework was evaluated by comparing its results of community with the LFR benchmark graphs communities.

4.3.1 The LFR benchmark graphs

A group of LFR benchmark graphs were defined that its formula is mentioned in (2.54) with characteristics similar to the dataset used in our experiments for use in evaluating our Framework, as shown in table (4.45).

Table (4.5): LFR benchmark graphs properties that used and set the k_{min} by 1

Nodes	γ	β	μ	k_{max}	s_{min}	s_{max}	seed
7057	1.55	3.5	0.7	697	48	776	60

4.3.2 Experimental results of the evaluation metrics applied to the CDGAE Frameworks

To evaluate the efficiency of CDGAE framework, the metrics NMI, and accuracy-score were used.

In table (4.6), the experimental results have recorded applying these metrics to the test set of datasets with the LFR benchmark graphs, that are mentioned their characteristics in table (4.5).

Datasets	**CDGAE Framework s**	**NMI**	**Accuracy**
Facebook government pages	FL1L_Framework	65.23	69.19
	FL3L_Framework	22.66	30.72
	FL5L_Framework	13.03	26.98
	AF1L_Framework	73.29	78.67
	AF3L_Framework	47.02	62.97
	AF5L_Framework	84.86	87.42
	VTF1L_Framework	77.29	78.67
	VTF3L_Framework	72.93	77.63
	VTF5L_Framework	33.9	50.02
	CF1L_Framework	57	65.88
	CF3L_Framework	47.48	77.54
	CF5L_Framework	38.79	54.47

4.4. The Discussion of the Experimental Results of the CDGAE Framework.

The experimental is resulting from the Framework are discussed from three aspects: effect of adding the features to the nodes, the feature selection effect and the model depth effect represented by the number of its layers.

There was mentioned in section (3.2.2.1) that the Centrality-based measurements were utilized for the features initialization and assigned their as node features, here, the impact of its application on the framework was discussed through the practical results obtained.

There mentioned in section (3.2.2.2) two types of feature selection: the correlation-based feature selection (CFS) and the Variance Threshold-based feature selection (VTFS) were used, here, the impact of its application on the framework was discussed by the practical results obtained.

The framework was built with (one layer, three layers, and five layers), as previously explained in section (3.2.3.3), which represents the depth of the framework. Also, its impact on the framework was discussed.

4.4.1. Measuring the experimental results of applying CDGAE Framework to Facebook government pages dataset.

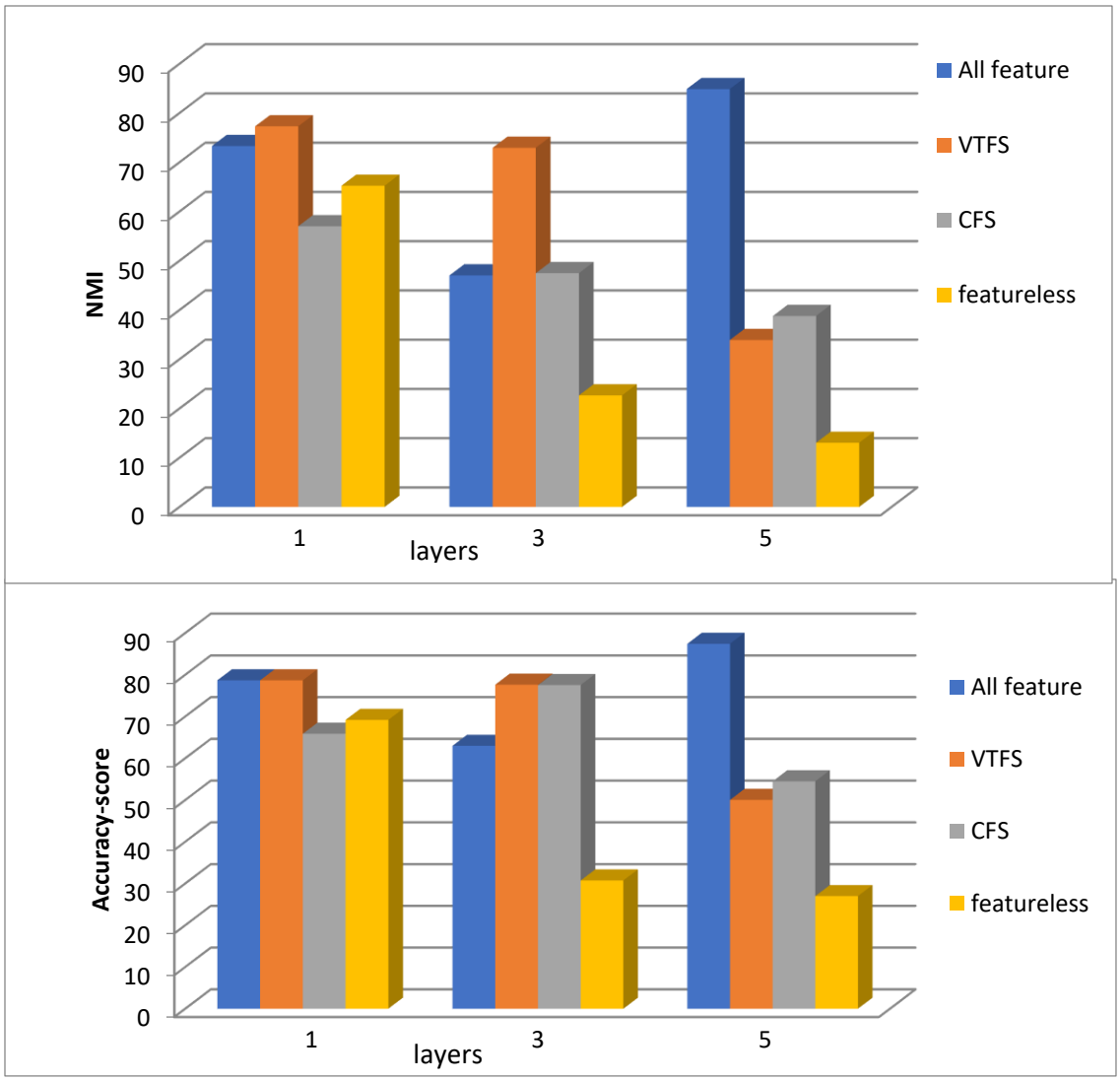


Figure (4.40): The charts of evaluation metrics results of Facebook government pages dataset.

Figure (4.40) shows the charts of MNI, and accuracy-score that describes CDFGAE Framework 's behavior under the effect of adding feature, Framework depth and feature selection for Facebook government pages dataset.

- Adding the features to the nodes of the dataset led to a noticeable improvement in the Framework performance.

- The framework performance improved with VTFS rather than CFS in comparison to all feature in the four matrices.
- The depth of framework showed marked improvement in the results of all features rather than CFS and VTFS.

4.5. Communities Simulation of the dataset.

In this section, the communities' graph of the best results of CDGAE framework for the dataset were represented.

- For Facebook government pages dataset, figure (4.41) shows the AF5L-Framework with 57 communities that was gave the best result of the experimental metrics.

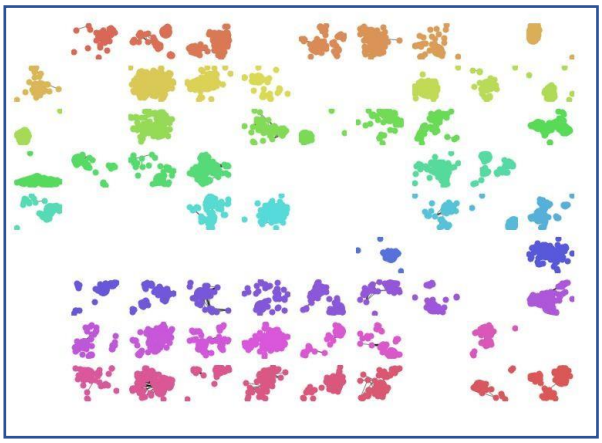


Figure (4.41): the communities of Facebook government pages dataset.

4.6. Time implementation of the CDGAE

The time required by the CDGAE Framework to execute centrality measurement computing, correlation-based, variance threshold-based feature selection, community number extracting and community detection are shown in table (4.7) and the time required to execute the training data are shown in table (4.8). The executing time was computed in seconds.

Table (4.7): The Time implementation of the CDGAE Framework

Dataset	**Centrality measurement computing**	**Correlation-based feature selection**	**Variance threshold-based feature selection**	**numbers of communities counting**	**Community detection**
Facebook government pages	6839.8 s	0.066 s	0.0156 s	634.58 s	4.436 s

Table (4.8): The Time implementation for training the CDGAE Framework

Dataset		**Training**			
		Featureless	**All Feature**	**VTFS**	**CFS**
Facebook government pages	1-layer	16863.4s	15935.75s	15357.97s	15616.78s
	3-layer	20628.46s	21093.207s	20995.87s	21166.46s
	5-layer	22707.441s	21069.468s	21207.44s	21234.58s

Chapter Five

Conclusions and Future Works

Chapter Five

Conclusions and Future Works

5.1 Conclusions

After implementing the proposed Framework, some conclusions can be registered in the following:

1. The graphs Autoencoder GAE was used for learning from structured data, by augmenting the graph features of the featureless graph nodes using centrality measurement. where community detection showed better performance with the use of features.
2. Improving the performance of community detection has been obtained by deepening the CDGAE Framework by increasing its layers, but this lead to increase the time of implementation and amount of computation.
3. Variance threshold-based feature Selection gave better results in less time, in contrast to the correlation-based feature Selection that did not significantly improve performance, this has been proven in implementing the CDGAE framework on the Facebook government pages dataset.

5.2 Future Works

1. Building an application that can be implemented to identify Influence nodes of the communities detected with the CDGAE Framework.
2. Development of the CDGAE Framework to become more powerful to deal with the most complex graph, i.e., big data from social media platforms, in a way that does not lose the original data and uses less amount of resources, and consumes less time.
3. Development of the CDGAE Framework to deal with the dynamic graphs.

4. Reconstructing the CDGAE by using deeper architecture can adaptively exploit the deep structural patterns of graphs to improve representational learning without compromising performance.

References

References

[1] W. R. Scott and G. F. Davis, “Organizations and Organizing Rational, Natural, and Open System Perspectives,” 2007, Accessed: Sep. 02, 2022. [Online]. Available: www.irpublicpolicy.ir

[2] P. Bedi and C. Sharma, “Community detection in social networks,” *Wiley Interdiscip Rev Data Min Knowl Discov*, vol. 6, no. 3, pp. 115–135, 2016.

[3] J. Sun, W. Zheng, Q. Zhang, and Z. Xu, “Graph Neural Network Encoding for Community Detection in Attribute Networks,” *IEEE*, vol. 52, no. 8, pp. 7791–7804, Jun. 2021, doi: 10.1109/TCYB.2021.3051021.

[4] J. Cao, D. Jin, L. Yang, and J. Dang, “Incorporating network structure with node contents for community detection on large networks using deep learning,” *Neurocomputing*, vol. 297, pp. 71–81, Jul. 2018, doi: 10.1016/j.neucom.2018.01.065.

[5] B. S. Khan, · Muaz, and A. Niazi, “Network Community Detection: A Review and Visual Survey.”

[6] D. Jin *et al.*, “A Survey of Community Detection Approaches: From Statistical Modeling to Deep Learning,” Jan. 2021, [Online]. Available: http://arxiv.org/abs/2101.01669

[7] D. Singh and R. Garg, “Issues and Challenges in Community Detection Algorithms” International Journal of Management, Technology And Engineering, 2018, p1375-1382, 8 (XII, DECEMBER), doi: 10.1109/ICTAI.2017.00035.

[8] V. Bhatia and R. Rani, "DFuzzy: a deep learning-based fuzzy clustering model for large graphs," *Knowl Inf Syst*, vol. 57, no. 1, pp. 159–181, Oct. 2018, doi: 10.1007/s10115-018-1156-3.

[9] L. Yang, X. Cao, D. He, C. Wang, X. Wang, and W. Zhang, "Modularity Based Community Detection with Deep Learning," ACM Journal, 2016. doi: 10.1016/j.ins.2016.03.090.

[10] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, "MGAE: Marginalized Graph Autoencoder for Graph Clustering," *CIKM'17 - Proceedings of the 2017 ACM Conference on Information and Knowledge Management*, pp. 889–898, 2017, doi: 10.1145/3132847.

[11] D. Jin, M. Ge, Z. Li, W. Lu, D. He, and F. Fogelman-Soulie, "Using deep learning for community discovery in social networks," in *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, Jun. 2018, vol. 2017-November, pp. 160–167. doi: 10.1109/ICTAI.2017.00035.

[12] J. Cao, D. Jin, and J. Dang, "Autoencoder based community detection with adaptive integration of network topology and node contents," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018, vol. 11062 LNAI, pp. 184–196. doi: 10.1007/978-3-319-99247-1_16.

[13] Y. Xie, X. Wang, D. Jiang, and R. Xu, "High-performance community detection in social networks using a deep transitive autoencoder," *Inf Sci (N Y)*, vol. 493, pp. 75–90, Aug. 2019, doi: 10.1016/j.ins.2019.04.018.

[14] J. J. Choong, X. Liu, and T. Murata, "Optimizing Variational Graph Autoencoder for Community Detection with Dual Optimization," *2020*, doi: 10.3390/e22020197.

[15] R. Xu, Y. Che, X. Wang, J. Hu, and Y. Xie, "Stacked autoencoder-based community detection method via an ensemble clustering framework," *Inf Sci (N Y)*, vol. 526, pp. 151–165, Jul. 2020, doi: 10.1016/j.ins.2020.03.090.

[16] D. Jin *et al.*, "A Survey of Community Detection Approaches: From Statistical Modeling to Deep Learning," Jan. 2021, doi: arXiv:2101.01669v3.

[17] O. Shchur and S. Günnemann, "Overlapping Community Detection with Graph Neural Networks," *Deep Learning on Graphs Workshop*, pp. 112–119, Sep. 2019, doi: 10.48550/arXiv.1909.12201 Focus to learn more.

[18] C. T. Duong *et al.*, "On Node Features for Graph Neural Networks," 33rd Conference on Neural Information Processing Systems, 2019. doi: 10.1145/3511808.3557661.

[19] C. M. Le and E. Levina, "Estimating the number of communities in networks by spectral methods," Jul. 2015, Electronic Journal of Statistics, 2015, doi: 10.1214/21-EJS1971.

[20] A. Saade, F. Krzakala, and L. Zdeborová, "Spectral Clustering of Graphs with the Bethe Hessian," *Advances in Neural Information Processing Systems 27* , vol. 1, pp. 406–414, Jun. 2014, doi: 10.1145/3072847.

[21] S. Abadal, A. Jain, R. Guirado, J. López-Alonso, and E. Alarcón, "Computing Graph Neural Networks: A Survey from Algorithms to Accelerators," *ACM Comput. Surv.*, vol. 54, Sep. 2020, doi: 10.1145/3477141.

[22] J. Zhou *et al.*, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020, doi: 10.1016/j.aiopen.2021.01.001.

[23] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," Jan. 2019, doi: 10.1109/TNNLS.2020.2978386.

[24] S. Sobolevsky, "Recurrent Graph Neural Network Algorithm for Unsupervised Network Community Detection," *EURASIP J Wirel Commun Netw*, Mar. 2021, doi: 10.1109/TNNLS.2020.2978386.

[25] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: a comprehensive review," *Comput Soc Netw*, vol. 6, no. 1, Dec. 2019, doi: 10.1186/s40649-019-0069-y.

[26] J. Zhou *et al.*, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020, doi: 10.1016/j.aiopen.2021.01.001.

[27] C. Song, F. Liu, Y. Huang, L. Wang, and T. Tan, "LNCS 8258 - Auto-encoder Based Data Clustering," Lecture Notes in Computer Science, 2013. doi: 10.1121/TNNLS.2020.2978386.

[28] R. Fei, J. Sha, Q. Xu, B. Hu, K. Wang, and S. Li, "A new deep sparse autoencoder for community detection in complex networks," *EURASIP J Wirel Commun Netw*, vol. 2020, no. 1, Dec. 2020, doi: 10.1186/s13638-020-01706-4.

[29] P. V. Tran, "Learning to Make Predictions on Graphs with Autoencoders," Feb. 2018, doi: 10.1109/DSAA.2018.00034.

[30] YUGESH VERMA, "A Complete Understanding of Dense Layers in Neural Networks," *DEVELOPERS CORNER*, Sep. 19, 2021. https://analyticsindiamag.com/a-complete-understanding-of-dense-layers-in-neural-networks/ (accessed Aug. 16, 2022).

[31] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning - whole book," *Nature*, vol. 521, no. 7553, p. 800, 2016, Accessed: Aug. 16, 2022. [Online]. Available: http://goodfeli.github.io/dlbook/%0Ahttp://dx.doi.org/10.1038/nature14539

[32] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks." JMLR Workshop and Conference Proceedings, pp. 315–323, Jun. 14, 2011. Accessed: Aug. 16, 2022. [Online]. Available: https://proceedings.mlr.press/v15/glorot11a.html

[33] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *32nd International Conference on Machine Learning, ICML 2015*, vol. 1, pp. 448–456, Feb. 2015, doi: 10.48550/1502.03167.

[34] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *ArXiv*, pp. 456–471, Dec. 2015, doi: 10.48550/ 1412.6980.

[35] J. Han, M. Kamber, and J. Pei, "Data Mining. Concepts and Techniques, 3rd Edition (The Morgan Kaufmann Series in Data Management Systems)," 2011.

[36] "tf.keras.losses.CosineSimilarity | TensorFlow Core v2.9.1." https://www.tensorflow.org/api_docs/python/tf/keras/losses/CosineSimilarity (accessed Aug. 17, 2022).

[37] Z. Wan, Y. Mahajan, B. W. Kang, T. J. Moore, and J. H. Cho, "A Survey on Centrality Metrics and Their Network Resilience Analysis," *IEEE Access*, vol. 9, pp. 104773–104819, 2021, doi: 10.1109/ACCESS.2021.3094196.

[38] A. Landherr, B. Friedl, and J. Heidemann, "A Critical Review of Centrality Measures in Social Networks," *Business & Information Systems Engineering*, vol. 2, no. 6, pp. 371–385, Dec. 2010, doi: 10.1007/s12599-010-0127-3.

[39] A. Saxena and S. Iyengar, "Centrality Measures in Complex Networks: A Survey," Nov. 2020, [Online]. Available: http://arxiv.org/abs/2011.07190

[40] K. Das, S. Samanta, and M. Pal, "Study on centrality measures in social networks: a survey," *Social Network Analysis and Mining*, vol. 8, no. 1. Springer-Verlag Wien, Dec. 01, 2018. doi: 10.1007/s13278-018-0493-2.

[41] J. Gräßler, D. Koschützki, and F. Schreiber, "CentiLib: Comprehensive analysis and exploration of network centralities," *Bioinformatics*, vol. 28, no. 8, pp. 1178–1179, Apr. 2012, doi: 10.1093/bioinformatics/bts106.

[42] M. Ashtiani *et al.*, "A systematic survey of centrality measures for protein-protein interaction networks," *BMC Syst Biol*, vol. 12, no. 1, Jul. 2018, doi: 10.1186/s12918-018-0598-2.

[43] X. Zhang, J. Zhu, Q. Wang, and H. Zhao, "Identifying influential nodes in complex networks with community structure," *Knowl Based Syst*, vol. 42, pp. 74–84, Apr. 2013, doi: 10.1016/j.knosys.2013.01.017.

[44] T. B. Kirigin, S. B. Babić, and B. Perak, "Semi-Local Integration Measure of Node Importance," *Mathematics*, vol. 10, no. 3, Feb. 2022, doi: 10.3390/math10030405.

[45] Z. Wan, Y. Mahajan, B. W. Kang, T. J. Moore, and J. H. Cho, "A Survey on Centrality Metrics and Their Network Resilience Analysis," *IEEE Access*, vol. 9, pp. 104773–104819, 2021, doi: 10.1109/ACCESS.2021.3094196.

[46] U. Brandes and D. Fleischer, “LNCS 3404 - Centrality Measures Based on Current Flow,” Springer, Berlin, Heidelberg, 2005. doi: 10.1007/978-3-540-31856-9_44.

[47] A. Hagberg, D. Schult, and P. Swart, “NetworkX Reference Release 2.5,” the organization source, 2020. doi: s12599-010-0127-1.

[48] B. Cai, L. Zeng, Y. Wang, H. Li, and Y. Hu, “Community detection method based on node density, degree centrality, and k-means clustering in complex network,” *Entropy*, vol. 21, no. 12, Dec. 2019, doi: 10.3390/e21121145.

[49] by Shlomi Dolev, Y. Elovici, R. Puzis, S. Dolev, and R. Puzis Ben-, “Routing Betweenness Centrality,” Journal of the ACM, 2009. doi: 10.1145/1734213.1734219.

[50] A. Landherr, B. Friedl, and J. Heidemann, “A Critical Review of Centrality Measures in Social Networks,” *Business & Information Systems Engineering*, vol. 2, no. 6, pp. 371–385, Dec. 2010, doi: 10.1007/s12599-010-0127-3.

[51] C. Duron, “Heatmap centrality: A new measure to identify super-spreader nodes in scale-free networks,” *PLoS One*, vol. 15, no. 7 July, Jul. 2020, doi: 10.1371/journal.pone.0235690.

[52] C. Feng, J. Ye, J. Hu, and H. L. Yuan, “Community Detection by Node Betweenness and Similarity in Complex Network,” *Complexity*, vol. 2021, 2021, doi: 10.1155/2021/9986895.

[53] J. G. Dy and C. E. Brodley, “Feature Selection for Unsupervised Learning,” Journal of Machine Learning Research, 2004. doi: 1-58113-233-6/00/08.

[54] S. Solorio-Fernández, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad, "A review of unsupervised feature selection methods," *Artif Intell Rev*, vol. 53, no. 2, pp. 907–948, Feb. 2020, doi: 10.1007/s10462-019-09682-y.

[55] J. Tang, S. Alelyani, and H. Liu, "Feature Selection for Classification: A Review," CRC Press, 2014. doi: 10.1201/b17320.

[56] M. Mohammadi, H. Sharifi Noghabi, G. Abed Hodtani, and H. Rajabi Mashhadi, "Robust and stable gene selection via Maximum-Minimum Correntropy Criterion," *Genomics*, vol. 107, no. 2–3, pp. 83–87, Mar. 2016, doi: 10.1016/j.ygeno.2015.12.006.

[57] M. A. Hall, "Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning," *University of Waikato, Department of Computer Science*, 2005, doi: 10.48550/324/1.

[58] T. Chakraborty, A. Dalmia, A. Mukherjee, and N. Ganguly, "Metrics for Community Analysis: A Survey," *ArXiv*, pp. 345–392, Apr. 2016, doi: 10.48550/arXiv.1604.03512.

[59] V. Labatut, "Generalized Measures for the Evaluation of Community Detection Methods," *International Journal of Social Network Analysis and Mining (SNAM)*, vol. 2, no. 1, pp. 44–63, 2015, doi: 10.1504/IJSNM.2015.069776ï.

[60] X. Liu, H.-M. Cheng, and Z.-Y. Zhang, "Evaluation of Community Detection Methods," *IEEE*, pp. 1736–1746, Jul. 2019, doi: 10.1109/TKDE.2019.2911943.

[61] A. Lancichinetti and S. Fortunato, “Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities,” Apr. 2009, doi: 10.1103/PhysRevE.80.016118.

[62] “(PDF) A fusion condition for community detection with modularity.” https://www.researchgate.net/publication/324215908_A_fusion_condition_for_community_detection_with_modularity (accessed Aug. 25, 2022).

[63] “Normalized Mutual Information. A measure to evaluate network… | by Luís Rita | Medium.” https://luisdrita.com/normalized-mutual-information-a10785ba4898 (accessed Aug. 25, 2022).

[64] M. Sokolova and G. Lapalme, “A systematic analysis of performance measures for classification tasks,” *Inf Process Manag*, vol. 45, no. 4, pp. 427–437, Jul. 2009, doi: 10.1016/j.ipm.2009.03.002.

[65] “Datasets.” https://github.com/benedekrozemberczki/datasets#github-stargazer-graphs.

الملخص

في الآونة الأخيرة ، انتشر استخدام منصات الشبكات الاجتماعية ، مثل الفيسبوك و تويتر و الويشات ، بشـــكل كبير. يعد اكتشـــاف المجتمع ، وهو تجميع المســـتخدمين ذوي الاهتمامات والميول المتشـــابهة في مجموعات ، جانبًا مهمًا لفهم البنية المعقدة للشبكات الاجتماعية.

يتم تمثيل المسـتخدمين كعقد في الشـبكة الاجتماعية. تصـف ميزات العقد ، بالإضـافة إلى طوبولوجيا الشـبكة ، علاقة التشـابه بينها. يتمثل التحدي في اكتشـاف المجتمع في التعامل مع الشـبكات المفتقرة للميزات. تتميز تقنية التشفير التلقائي للرسم البياني بدقة منخفضة في اكتشاف المجتمع. لذلك يكمن التحدي البرمجي في تطوير بنية هذه التقنية لإعطاء نتائج عالية الدقة.

اسـتخدمت هذه الأطروحة الطريقة الهجينة ، اكتشـاف المجتمع باسـتخدام إطار عمل التشـفير التلقائي للرسـم البياني (CDGAE). حيث تم اسـتخدام الطريقة القائمة على قياس المركزية لتعيين الميزات ، للتعامل مع الشـبكة الخالية من الميزات. ثم تم اسـتخدام طريقتين لاختيار الميزة ، تحديد ميزة الارتباط و تحديد ميزة عتبة التباين ، لتجنب التكرار والحشـو في الميزات. تمت زيادة طبقات GAE لزيادة دقة اكتشـــاف المجتمع. تمت مســـاواة الرقم العصـــبي لطبقة عنق الزجاجة في المشـــفر التلقائي للرســـم البياني (GAE) مع عدد المجتمعات ، حيث تم تحديده بواسـطة الطريقة الطيفية لمصـفوفة بيثا حسـين ، لتحديد عضـوية كل عقدة في مجتمعها مباشرةً دون استخدام خوارزمية التجميع.

تم تطبيق CDGAE على مجموعة بيانات صـــفحات الحكومة على الفيســـبوك. أظهرت النتائج التجريبية أفضـل أداء لـــــ CDGAE بخمس طبقات لمجموعة بيانات صـفحات حكومة الفيسـبوك بعد زيادة عقدها بالميزات ، حيث سجل مقياس المعلومات المتبادلة الموحدة (NMI-Score) ومقياس الدقة القيم المئوية 84.86 و 87.42 على التوالي.

وزارة التعليم العالي والبحث العلمي
جامعة بابل
كلية علوم البنات
قسم الحاسوب

اكتشاف المجتمع الاجتماعي باستخدام قياس المركزية وتقنية الترميز الثلقائي

رسالة مقدمة إلى مجلس كلية العلوم للبنات - جامعة بابل وهي
جزء من متطلبات نيل درجة الماجستير في العلوم / علوم
الحاسوب

من قبل
حوراء زهير احمد
بإشراف
أ.م.د. آسيا مهدي ناصر الزبيدي

1444 هـ

2022 م