Adaptive Graph Attention Network in Person Re-Identification

Final Year Project Report

A 4th Year Student Name

Le Dinh Duy
HE130655

Under the supervision of

Dr. Phan Duy Hung

Bachelor of Computer Science

Hoa Lac Campus – FPT University

Spring 2021

DECLARATION

Project Title	Adaptive Graph Attention Network in Person Re-Identification
Author	*Le Dinh Duy*
Student ID	HE130655
Supervisor	Dr. Phan Duy Hung

I declare that this thesis entitled *Adaptive Graph Attention Network in Person Re-Identification* is the result of my own work except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Le Dinh Duy
HE130655

Department of Computer Science
Hoa Lac Campus – FPT University

Date: April 27, 2021

ACKNOWLEDGEMENTS

I would like to express my deep and sincere gratitude to my research supervisor, *Dr. Phan Duy Hung*, for giving me the opportunity to do research and providing invaluable guidance throughout this work. His dynamism, vision, sincerity and motivation have deeply inspired me. He has taught me the methodology to carry out the work and to present the works as clearly as possible. It was a great privilege and honor to work and study under his guidance.

I am greatly indebted to my honorable teachers of the Department of Computer Science at the Hoa Lac Campus – FPT University who taught me during the course of my study. Without any doubt, their teaching and guidance have completely transformed me to a person that I am today.

I am extremely thankful to my parents for their unconditional love, endless prayers, caring and immense sacrifices for educating and preparing me for my future. I would like to say thanks to my friends and relatives for their kind support and care.

Finally, I would like to thank all the people who have supported me to complete the project work directly or indirectly.

Le Dinh Duy
Hoa Lac Campus – FPT University
Date: April 27, 2021

Dedicated to
Everyone interested in the applications of graph learning in the artificial intelligence word

– Le Dinh Duy

ABSTRACT

This paper approaches a significant problem in computer vision: re-identifying a person when having groups of people. Re-identifying (Re-ID) by group context is a new direction for improving traditional single-object Re-ID task by additional information from group layout and group member variations. Adding new improvements in the graph convolution layer structure or using more powerful theories boosts the model's accuracy. To handle these issues, we proposed to leverage the information of group objects: person and subgroups of two or three people inside a group image. We have reorganized the load data structure to improve training efficiency. Organizing the data is based on the relational representation of the central node, and the observed nodes further incorporate their features extracted through the backbone is Resnet. We propose a graph convolution model that uses the Selu activation function to study this data. The key challenge in implementing is to define the optimal group-wise matching using adaptive graph attention based on a graph convolution network modified and training techniques. However, we only did this training for a limited amount of time with 20 epochs per training due to resource limitation. The key challenge in implementing is to define the optimal group-wise matching using adaptive graph attention based on a graph convolution network modified and training techniques. However, we only did this training for a limited amount of time with 20 epochs per training due to resource limitation.

Keywords: Graph Convolution Network, Graph learning, Graph attention, Activation function, Context Interaction, Neural Networks, Person Re-Identification

Contents

1 Introduction **1**

1.1 Background 1
1.2 Motivation 3
1.3 Objectives and Contributions 3
1.4 Organization 3

2 Related work **4**

2.1 Graph neural networks 4
2.1.1 Graph convolutional networks (GCN) 4
2.1.1.1 Definitions 4
2.1.1.2 Preprocessing Graphs 5
2.1.1.3 Propagation Rule 5
2.1.1.4 The number of layers 6
2.1.2 Contextual Graph Representation Learning 7
2.1.3 Graph attention 9
2.1.4 Graph Learning Network 10
2.2 Loss function 12
2.2.1 Cosine embedding loss 12
2.2.2 Graph learning loss 12
2.3 Activation methods 13
2.3.1 Rectified Linear Units 13
2.3.2 Scaled Exponential Linear Unit 13
2.3.3 Continuously Differentiable Exponential Linear Units 13
2.4 Evaluation measures (information retrieval) 15
2.4.1 MAP (Mean Average Precision) 15
2.4.2 Cumulative Matching Characteristics (CMC) 17

2.5 Regularization . . . 17
2.5.1 Dropout . . . 17
2.5.2 Stochastic Depth . . . 18

3 Adaptive graph attention **20**
3.1 Overview . . . 20
3.2 The attention central node matrix . . . 21
3.3 Adaptive graph model . . . 21
3.3.1 Graph convolution modified activation network . . . 22
3.3.2 Graph attention with graph convolution network . . . 23
3.3.3 Graph learning combined graph convolution network . . . 23
3.4 Noise learning . . . 24
3.4.1 Dropout base on level hops . . . 24
3.4.2 Stochastic Depth in Graph Convolution network . . . 25

4 Experimental Results **27**
4.1 Experimental setup . . . 27
4.1.1 Handle dataset . . . 27
4.1.1.1 Traindata . . . 28
4.1.1.2 Test dataset . . . 28
4.1.2 Evaluation metric . . . 29
4.1.3 Implementation Details . . . 29
4.2 Performance . . . 29
4.2.1 Performance change after re-make data-loader . . . 29
4.2.2 Comparison with GCN base [1] Methods . . . 30
4.2.3 Comparison model when apply noisy learning . . . 30
4.3 Analysis . . . 31
4.3.1 Results on re-make data loader . . . 31
4.3.2 Results Comparison model . . . 31
4.3.3 Results when apply noisy learning . . . 33
4.4 Discussion . . . 33

5 Conclusion **35**

6 Appendix **37**
6.1 process data 37
6.1.1 filter 37
6.1.2 getIndex 37
6.1.3 getItem 38
6.1.4 getGallery 39
6.1.5 gallery gcn 39
6.2 Model diagram 40

References **46**

List of Figures

2.1 Example: Gathering info process with 2 layers of target node i. 7
2.2 Performance over layers. Picture from the paper [2] 7
2.3 The attention mechanism $\vec{a}(W_{\vec{h}i}, W_{\vec{h}j})$ employed by GAT model, parametrized by a weight vector $\vec{a} \in \mathbb{R}^{2F'}$, applying a LeakyReLU activation. 9
2.4 The author proposed method is a recurrent block. They create a set of node embeddings $\{H_i^{(l)}\}_{i=1}^{k}$ that are later combined to produce an intermediary representation $H_{int}^{(l)}$. Then use the updated node information with the adjacency information to produce a local embedding of the nodes information $H_{local}^{(l)}$ thats is also the output $H^{(l+1)}$. Also broadcast the information of the local embedding to produce a global embedding $H_{global}^{(l)}$. And combine the local and global embeddings to predict the next layer adjacency $A^{(l+1)}$[3]. 11
2.5 Rectified Linear Units distribution 14
2.6 Scaled Exponential Linear Units distribution 14
2.7 Continuously Differentiable Exponential Linear Units distribution . 15
2.8 Computing the Intersection over Union is as simple as dividing the area of overlap between the bounding boxes by the area of union . . 16
2.9 Dropout Neural Net Model. Left: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the **left**. Crossed units have been dropped..[4] 18
2.10 The linear decay of p_l illustrated on a ResNet with stochastic depth for $p_0 = 1$ and $p_L = 0.5$. Conceptually, we treat the input to the first ResBlock as H_0, which is always active.[4] 19

3.1 GCN model structure with 3 layer graph convolution 22
3.2 Graph attention with graph convolution network structure 23

3.3 Graph learning combined graph convolution network structure . . . 24
3.4 Out GCN + dropout model structure with 3 layer graph convolution 25
3.5 Out GCN + StoDepth model structure with 3 layer graph convolution 26

6.1 Out GCN model structure with 3 layer graph convolution 41
6.2 GAT + GCN model diagram 42
6.3 Graph learning with GCN network structure 43
6.4 Graph convolution network add Dropout 44
6.5 Graph convolution network and StoDepth 45

List of Tables

4.1 Performance evaluate Data processing base on Gcn base model with 3 layer and CUHK-SYSU dataset [1]. 30
4.2 Performance evaluate our model Gcn base model on CUHK-SYSU dataset with gallery size of 100 30
4.3 Performance evaluate our model with noisy training. 30

Chapter 1

Introduction

1.1 Background

In computer vision, person re-identification (re-id) is a fundamental and significant field of research. Its objective is to reidentify individuals across multi-camera surveillance systems. Person re-identification has a huge potential in applications related to video surveillance, for instance searching for lost people or suspects. In recent years, re-identification has been receiving increasing attention as an effect of its excellent applications in jobs directly related to security and safety. For a typical person re-id pipeline, the re-id system is provided with a target person as a probe. Therefore, the re-id system seeks to search through a gallery of known ID recordings to find the matched ones.

However, there are several following reasons why the re-id individual has to face numerous difficult issues. First, the distributions of probe and gallery are multi-modal as a result of the variety of data sources. To illustrate, pedestrians can be captured by surveillance cameras or smartphones. Second, vastly differing illuminations and human poses will amplify intra-class variations. Third, erroneous detection/tracking, occlusions, and background clutter lead to massive appearance changes, which further increases the difficulty for a person to re-id.

Typically, traditional person re-id tasks have focused exclusively on manually cropped image snapshots or video clips from multiple cameras. These techniques consider learning distance metrics based on individual features. Therefore, one critical precondition is that the foreground pedestrian should be accurately detected

or annotated in the scene. Alternatively, inaccuracies in detection or annotation will cause heavy noise into individual appearances, which makes this problem setting unfeasible in real scenarios.

To close this gap, several researchers [5, 1] have recently introduced the person search setting into this domain. Basically, the idea is to simultaneously manage two tasks (i.e., pedestrian detection and person re-identification) within a single framework. This setting is more representative of real-world scenarios and enables the system to operate without the use of off-line pedestrian detectors. Nevertheless, these methods still employ individual features as appearance cues. Thus, it is difficult to distinguish people who are dressed similarly, especially in situations where we have to search through a large gallery set.

Although utilizing context/group information is a potential approach to tackle real-world person re-identification, but recent works suffer from the following problems. To begin, determining how to identify a group is not a simple process. Existing techniques [6] rely heavily on manual annotations to recognize semantic groups, which entails a significant amount of human labor. Alternative methods [7, 8] make use of spatial and temporal cues like velocity and relative position in the incident, which are regarded as social constraints, to model group behaviors in order to assist in person re-id. These social force models make extensive use of intricately designed constraints to simulate social influences in a scene, which ordinarily does not have trivial solutions and is difficult for optimization.

Graph Convolutional Networks [2] have recently gained prominence with applications in several domains, from node classification [2] to physical systems simulation [9]. Learning dense and low-dimensional node representations from graph-structured data has become the keystone in many practical application scenarios, including typical user-item collaborative filtering [10]. Their main advantage is their ability to deal with node information and relational structure at the same time. GCNs learn node representations as the summary of the node's neighborhood, which allows them to effectively encode the graph's (higher-order) topological structure and connectivity information [2, 11, 12]. Moreover, one of the advantages of GCN is that they seamlessly incorporate graph structure information in the

form of the graph adjacency matrix into the model. Hence GCNs do not need to learn that structure and can focus their modeling capacity on learning good node representations.

1.2 Motivation

Previous research has been done in the area of context-based re-identification. Notably, several graph approaches have demonstrated their superiority in aggregating context information. However, these works have been condensed in-depth because they rely on primitive ideas that ignore the relationship between recent technological advances. Besides, the organization of data also plays a vital role in the training process is not well organized. The research has focused on implement combinations of information convolution architectures, training methods, and improving the sampling method while loading the data set.

1.3 Objectives and Contributions

The goal of this research is to improve the learning efficiency of the graph approach with the re-id task of the application of the contextual. Notably, this thesis proposes changes in the data loader engine and the base graph convolution thesis. This thesis contributes by combining multiple learning methods based on the combination of graph-based architectures with the customization we propose.

1.4 Organization

This paper is structured as follows. Chapter 2 introduces studies of the literature studies related to the thesis. Chapter 3 details our strategies, which include a customized simulation architecture and teaching methods. Chapter 4 presents the design implementation and the experiments. Chapter 5 concludes the thesis and suggests future work.

Chapter 2

Related work

2.1 Graph neural networks

2.1.1 Graph convolutional networks (GCN)

Graph convolutional networks (GCN) [2] has been proposed to learn graph relations with convolution, which facilitates the optimization of traditional graph model. GCN has been applied to various tasks [13, 14, 15, 16, 17]. Several recent works [18, 19] employ graph model for person re-id, but the graphs are constructed to model relationships between probe-gallery pairs, and no context information is considered. In this thesis, we design a target-context graph and employ a pairwise GCN to learn visual relations in the scene.

2.1.1.1 Definitions

Currently, most graph neural network models have a somewhat universal architecture in common. For these models, the goal is then to learn a function of signals/ features on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ which takes as input:

- A feature description x_i for every node i; summarized in a $N \times D$ feature matrix X (N: number of nodes, D: number of input features)
- A representative description of the graph structure in matrix form; typically in the form of an adjacency matrix A (or some function thereof)

and produces a node-level output Z (an $N \times F$ feature matrix, where F is the number of output features per node). Graph-level outputs can be modeled by introducing some form of pooling operation (see, e.g. [20]).

2.1.1.2 Preprocessing Graphs

The general idea of GCN: For each node, we get the feature information from all its neighbors and of course, the feature of itself [21].

Assume we use the average() function. We will do the same for all the nodes. Finally, we feed these average values into a neural network. A is typically not normalized and therefore the multiplication with A will completely change the scale of the feature vectors (we can understand that by looking at the eigenvalues of A).

In graph theory, the Laplacian matrix [22] is defined as $L = D - A$, where

- A, the adjacency matrix, indicates for each pairs of vertices whether they are connected by an edge;
- D, the degree matrix, is the diagonal matrix containing the number of edges attached to each vertex.

Multiplying with $D^{-1}A$ now corresponds to taking the average of neighboring node features.

To put more weights on the nodes that have low-degree and reduce the impact of high-degree nodes, Symmetric normalized Laplacian are apply [2] as the "weighted" average. The idea of this weighted average is that we assume low-degree nodes would have bigger impacts on their neighbors, whereas high-degree nodes generate lower impacts as they scatter their influence at too many neighbors. The normalization equation is modified from:

$$normalization\ term = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}, \tag{2.1}$$

2.1.1.3 Propagation Rule

Every neural network layer can then be written as a non-linear function

$$H^{(l+1)} = f\left(H^{(l)}, A\right), \tag{2.2}$$

with $H^{(0)} = X$ and $H^{(L)} = Z$ (or z for graph-level outputs), L being the number of layers. The specific models then differ only in how $f(\cdot, \cdot)$ is chosen and parameterized.

let's consider the following simple form of a layer-wise propagation rule:

$$f(H^{(l)}, A) = \sigma\left(AH^{(l)}W^{(l)}\right), \tag{2.3}$$

Combining these 2.1.1.2, we essentially arrive at the propagation rule introduced in [2]:

$$f(H^{(l)}, A) = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right), \tag{2.4}$$

with $\hat{A} = A + I$, where I is the identity matrix and $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$.

2.1.1.4 The number of layers

The number of layers is the farthest distance that node features can travel. For example, with 1 layer GCN, each node can only get the information from its neighbors. The gathering information process takes place **independently, at the same time** for all the nodes.

When stacking another layer on top of the first one, we repeat the gathering info process, but this time, the neighbors already have information about their own neighbors (from the previous step) represent in figure 2.1. It makes the number of layers as **the maximum number of hops** that each node can travel. So, depends on how far we think a node should get information from the networks, we can config a proper number for layers. **But again, in the graph, normally we don't want to go too far. With 6–7 hops, we almost get the entire graph which makes the aggregation less meaningful**.

In the paper [2], the authors also conducted some experiments with shallow and deep GCNs. From the figure 2.2, we see that **the best results are obtained with a 2 or 3-layer model**. Besides, with a deep GCN (more than 7 layers), it tends to get bad

performances (dashed blue line). One solution is to use **the residual connections between hidden layers** (purple line).

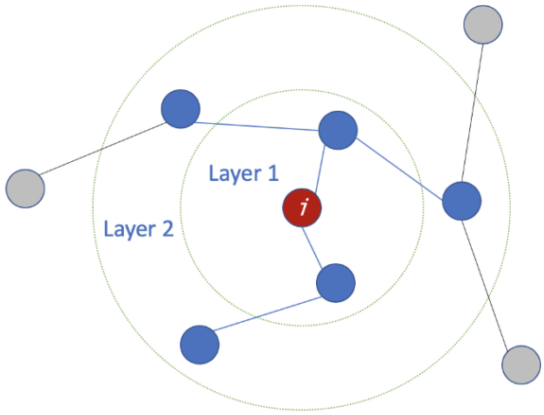


Figure 2.1: Example: Gathering info process with 2 layers of target node i.

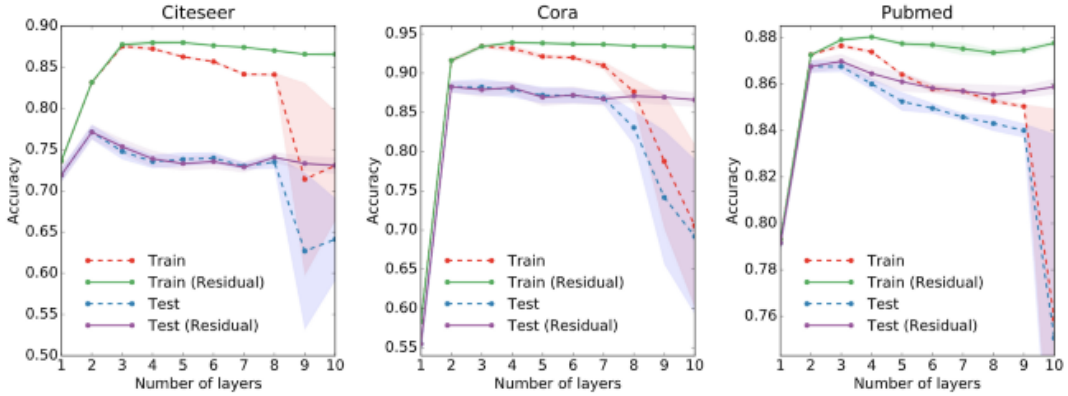


Figure 2.2: Performance over layers. Picture from the paper [2]

2.1.2 Contextual Graph Representation Learning

In this section, we introduce the detailed structure of the proposed context graph as well as the GCN model employed to learn graph parameters. The overall structure is illustrated on the right side of Figure 3. Given two images A and B. The objective of our model is to judge whether target pair appeared in the images A0

and B0 (in red bounding boxes) belong to the same identity, given K context pairs $(A_i, B_i), i \in \{1, .., K\}$. The objective is to construct a graph to jointly take the target pairs and the context information into consideration, and eventually outputs the similarity score. A straightforward solution is to employ two graphs to model each image, and to utilize a Siamese GCN structure to extract features of both graphs, as in [20]. However, the Siamese structure prevents the contextual information to propagate between graphs, which leads to significant information loss. In our situation, the targets and contexts all appear in pairs. Therefore, we build a graph whose nodes consist of instance pairs. In this graph, the target node is the center of the graph, which is connected to all the context nodes for information propagation and feature updation.

In particular, considering a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consisting of N vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$. We assign each node with a pair of features $(x_{A_j}, x_{B_j}), j \in \{0, ..., K\}$. If the images have K context pairs, then $N = K + 1$. We use $X \in \mathbb{R}^{N \times 2d}$, where d is the instance-level feature dimension. We use $A \in \mathbb{R}^{N \times N}$ to denote the adjacent matrix associated with graph G. If we assign the target node as the first node in the graph, then the adjacent matrix is:

$$A_{i,j} = \begin{cases} 1 & if \quad i = 1 \quad or \quad j = 1 \quad or \quad i = j \\ 0 & otherwise \end{cases}, \tag{2.5}$$

where $i, j \in \{1, ..., N\}$. If we use $\hat{A}$ to denote the normalized adjacency matrix, layer-wise GCN propagates as follows:

$$Z^{(l+1)} = \sigma\left(\hat{A} Z^{(l)} W^{(l)}\right), \tag{2.6}$$

where $Z^{(l)}$ is the matrix activation of the $l - th$ layer, and $Z^{(0)} = X$ as input. $W^{(l)}$ is the learnable parameter matrix and σ is the ReLU activation function in our framework. Finally, we utilize a fully connected layer to merge all the vertices features into 1024-dimensional feature vector. And a binary Softmax loss layer is employed supervise network training.

2.1.3 Graph attention

A Graph Attention Network (GAT) [23] is a neural network architecture that operates on graph-structured data, leveraging masked self-attentional layers to address the shortcomings of prior methods based on graph convolutions or their approximations. By stacking layers in which nodes are able to attend over their neighborhoods' features, a GAT enables (implicitly) specifying different weights to different nodes in a neighborhood, without requiring any kind of costly matrix operation (such as inversion) or depending on knowing the graph structure upfront.

GAT introduces the attention mechanism as a substitute for the statically normalized convolution operation. Below are the equations to compute the node embedding $h_i^{(l+1)}$ of layer $l+1$ from the embeddings of layer l.

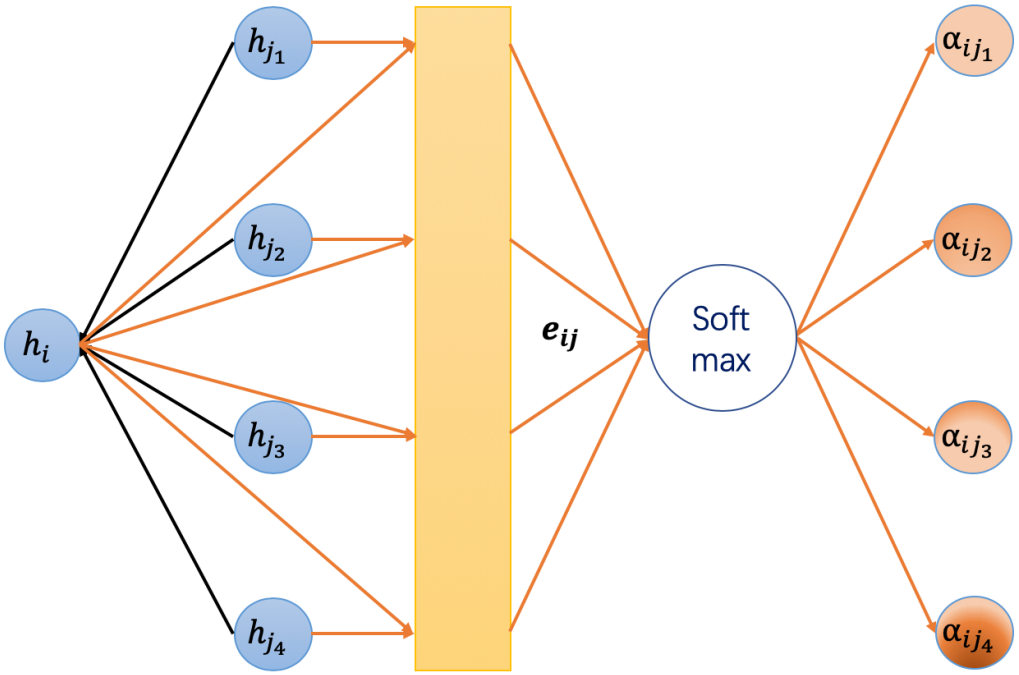


Figure 2.3: The attention mechanism $\vec{a}(W_{\vec{h}i}, W_{\vec{h}j})$ employed by GAT model, parametrized by a weight vector $\vec{a} \in \mathbb{R}^{2F'}$, applying a LeakyReLU activation.

$$Z_i^{(l)} = W^{(l)} h_i^{(l)}, \tag{2.7}$$

$$e_{ij}^{(l)} = LeakyReLU(\vec{a}^{(l)^T}(z_i^{(l)}||z_j^{(l)})), \tag{2.8}$$

$$a_{ij}^{(l)} = \frac{exp(e_{ij}^{(l)})}{\sum_{k \in N_i} exp(e_{ik}^{(l)})}, \tag{2.9}$$

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in N_i} a_{ij}^{(l)} Z_j^{(l)}\right), \tag{2.10}$$

Explanations:

- Equation (2.7) is a linear transformation of the lower layer embedding $h_i^{(l)}$ and $W^{(l)}$ is its learnable weight matrix.
- Equation (2.8) computes a pair-wise un-normalized attention score between two neighbors. Here, it first concatenates the Z embeddings of the two nodes, where $||$ denotes concatenation, then takes a dot product of it and a learnable weight vector $\vec{a}^{(l)}$, and applies a LeakyReLU in the end. This form of attention is usually called additive attention, contrast with the dot-product attention in the Transformer model.
- Equation (2.9) applies a softmax to normalize the attention scores on each node's incoming edges.
- Equation (2.10) is similar to GCN. The embeddings from neighbors are aggregated together, scaled by the attention scores.

2.1.4 Graph Learning Network

Given a set of vertices $V = \{v_i\}$, such that every element vi is a feature vector, we intend to predict its structure as a set of edges between the vertices, $E = \{(v_i, v_j) : v_i, v_j \in V\}$. In other words, we want to learn the edges of the graph $G = (V, E)$ that maximize the relations between the vertices given some prior patterns, i.e., a family of graphs.

The sequential application of these steps recover effective relations on nodes, even when trained on families of graphs. We represent this process in 2.4.

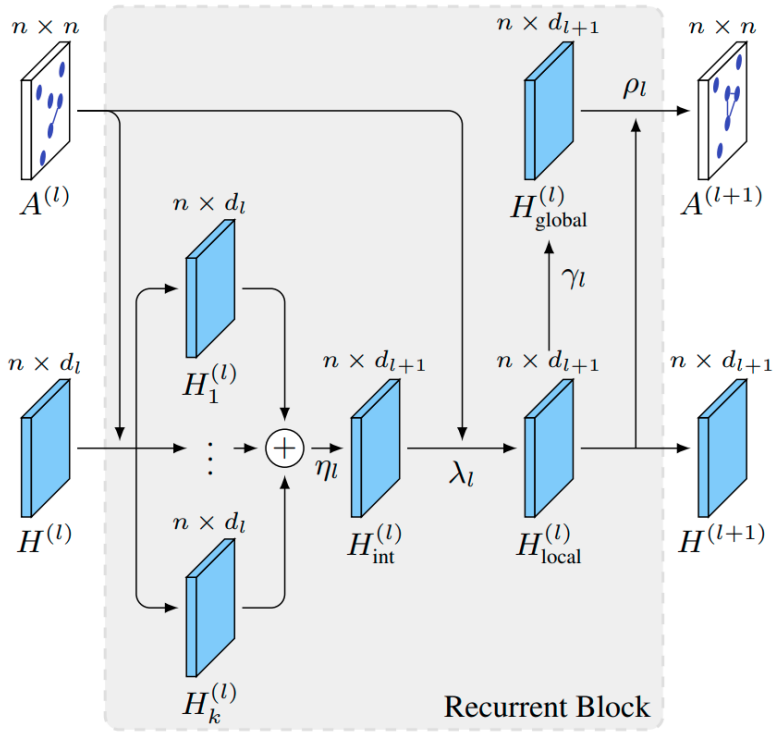


Figure 2.4: The author proposed method is a recurrent block. They create a set of node embeddings $\{H_i^{(l)}\}_{i=1}^k$ that are later combined to produce an intermediary representation $H_{int}^{(l)}$. Then use the updated node information with the adjacency information to produce a local embedding of the nodes information $H_{local}^{(l)}$ thats is also the output $H^{(l+1)}$. Also broadcast the information of the local embedding to produce a global embedding $H_{global}^{(l)}$. And combine the local and global embeddings to predict the next layer adjacency $A^{(l+1)}$[3].

Node Embeddings. At a given step l on the alternating process, we have d_l hidden features, $H^{(l)} \in \mathbb{R}^{n \times d_l}$, for each of our n nodes, and the set of edges (structure) encoded into an adjacency matrix $A^{(l)} \in [0,1]^{n \times n}$ that represents our graph. As introduced, our first step is to produce the features of the next step, $H^{(l+1)}$, through the embedding.

Adjacency Matrix Prediction. After obtaining the nodes embeddings, $H_{local}^{(l)}$ (5), we use them to predict the next adjacency matrix $A^{(l+1)}$ through

$$A^{(l+1)} = \rho_l\left(H_{local}^{(l)}\right) = \sigma_l\left(M^{(l)}\alpha_l\left(H_{local}^{(l)}\right)M^{(l)^T}\right), \tag{2.11}$$

where $M^{(l)} \in \mathbf{R}^{n \times n}$ is the weight matrix that produces a symmetric adjacency, α_l is a transformation that mixes global and local information within the graph, and $\cdot^T$ denotes the transposition operator.

The intuition is that nodes similar to the global and local context should receive higher attention weights for the projection of a new adjacency graph within the graph creation (2.11).

2.2 Loss function

2.2.1 Cosine embedding loss

This is used for measuring whether two inputs are similar or dissimilar, using the cosine distance, and is typically used for learning nonlinear embeddings or semi-supervised learning.

The loss function for each sample is:

$$\mathcal{L}(x, y) = \begin{cases} 1 - \cos(x_1, x_2) & \text{if } y = 1, \\ \max(0, \cos(x_1, x_2)) & \text{if } y = -1, \end{cases} \tag{2.12}$$

2.2.2 Graph learning loss

We use the modified loss function [24] based on [25] to optimize the learnable weight vector w_i as follows:

$$\mathcal{L}_{GL} = \frac{1}{N^2} \sum_{i,j=1}^{N} \exp\left(A_{i,j} + \eta ||v_i - v_j||_2^2\right) + \gamma ||A||_F^2 \tag{2.13}$$

where $\| \cdot \|_F$ represents Frobenius-Norm. Intuitively, the first item means that nodes v_i and v_j are far apart in higher dimensions encouraging a smaller weight value A_{ij} , and the exponential operation can enlarge this effect. Similarly, nodes that are close to each other in higher dimensional space can have a stronger connection weight. This process can prevent graph convolution aggregating information of noise node. η is a tradeoff parameter controlling the importance of nodes of the graph. We also average the loss due to the fact that the number of nodes is dynamic on the different

documents. The second item is used to control the sparsity of soft adjacent matrix A. γ is a tradeoff parameter and larger γ brings about more sparsity soft adjacent matrix A of graph.

2.3 Activation methods

2.3.1 Rectified Linear Units

Rectified Linear Units [26], or ReLUs, are a type of activation function that are linear in the positive dimension, but zero in the negative dimension. The kink in the function is the source of the non-linearity. Linearity in the positive dimension has the attractive property that it prevents non-saturation of gradients (contrast with sigmoid activations), although for half of the real line its gradient is zero. RELUs is show as Fig.2.5

The RELU activation function is given by:

$$RELU(x) = (x)^+ = \max(0, x) \tag{2.14}$$

2.3.2 Scaled Exponential Linear Unit

Scaled Exponential Linear Units [27], or SELUs, are activation functions that induce self-normalizing properties. SELUs is show as Fig.2.6

The SELU activation function is given by:

$$SELU(x) = \begin{cases} \lambda x & \text{if } x \geq 0 \\ \lambda\alpha\left(\exp\left(x\right) - 1\right) & \text{if } x < 0 \end{cases} \tag{2.15}$$

with $\alpha \approx 1.6733$ and $\lambda \approx 1.0507$

2.3.3 Continuously Differentiable Exponential Linear Units

The Continuously Differentiable Exponential Linear Units [28], or CELU, is is simply the ELU [29] where the activation for negative values has been modified to ensure

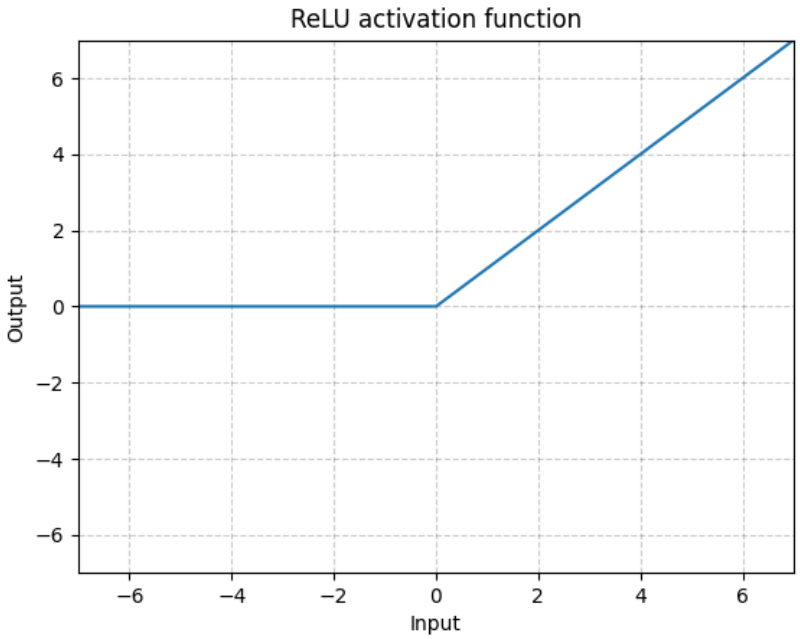


Figure 2.5: Rectified Linear Units distribution

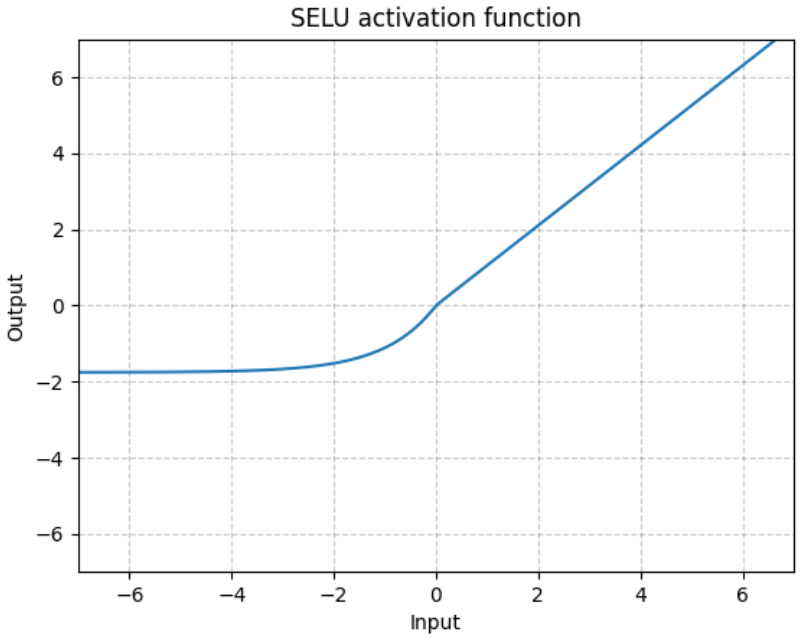


Figure 2.6: Scaled Exponential Linear Units distribution

that the derivative at $x = 0$ for all values of α is 1. CELUs is show as Fig.2.7 The CELU activation function is given by:

$$CELU(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha \left(\exp\left(\frac{x}{\alpha}\right) - 1 \right) & \text{otherwise} \end{cases} \tag{2.16}$$

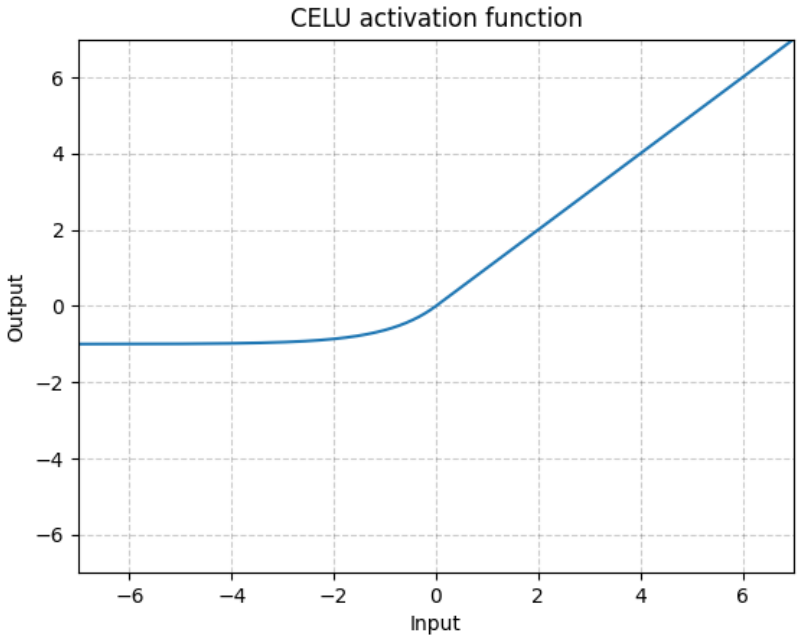


Figure 2.7: Continuously Differentiable Exponential Linear Units distribution

2.4 Evaluation measures (information retrieval)

2.4.1 MAP (Mean Average Precision)

AP (Average precision) is a popular metric in measuring the accuracy of object detectors like Faster R-CNN, SSD, etc. Average precision computes the average precision value for recall value over 0 to 1.

Precision & recall

Precision of a given class in classification, also known as positive predicted value,

is given as the ratio of true positive (TP) and the total number of predicted positives. The formula is given as such:

$$Precistion = \frac{TP}{TP + FP} = \frac{TP}{total\ positive\ results}, \tag{2.17}$$

Recall also known as true positive rate or sensitivity, of a given class in classification, is defined as the ratio of TP and total of ground truth positives. The formula is given as such:

$$Recall(TPR) = \frac{TP}{TP + FN} = \frac{TP}{total\ cancer\ cases}, \tag{2.18}$$

where: TP = True positive; TN = True negative; FP = False positive ; FN = False negative;

IoU (Intersection over union)

IoU measures the overlap between 2 boundaries. We use that to measure how much our predicted boundary overlaps with the ground truth (the real object boundary) follow by 2.8

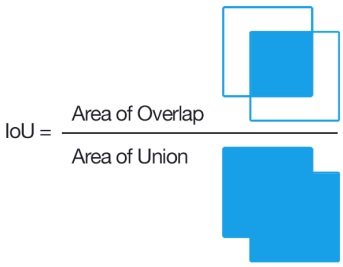


Figure 2.8: Computing the Intersection over Union is as simple as dividing the area of overlap between the bounding boxes by the area of union

Traditionally, we define a prediction to be a TP if the IoU is > 0.5. The are possible scenarios are described below:

True Positive $IoU > 0.5$

False Positive There are two possible scenarios where a BB would be considered as FP:

- $IoU < 0.5$
- Duplicated BB

False Negative When our object detection model missed the target, then it would be considered as a False Negative. The two possible scenarios are as follows:

- When there is no detection at all.
- When the predicted BB has an $IoU > 0.5$ but has the wrong classification, the predicted BB would be FN.

Average precision [30] Formula of mAP:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \,, \tag{2.19}$$

2.4.2 Cumulative Matching Characteristics (CMC)

Cumulative Matching Characteristics (CMC) curves are the most popular evaluation metrics for person re-identification methods. Consider a simple single-gallery-shot setting, where each gallery identity has only one instance. For each query, an algorithm will rank all the gallery samples according to their distances to the query from small to large, and the CMC top-k accuracy is

$$Acc_k = \begin{cases} 1 & \text{if top-k ranked gallery samples contain the query identity} \\ 0 & \text{otherwise} \end{cases} , \tag{2.20}$$

which is a shifted step function. The final CMC curve is computed by averaging the shifted step functions over all the queries. While the single-gallery-shot CMC is well defined, it does not have a common agreement when it comes to the multi-gallery-shot setting, where each gallery identity could have multiple instances.

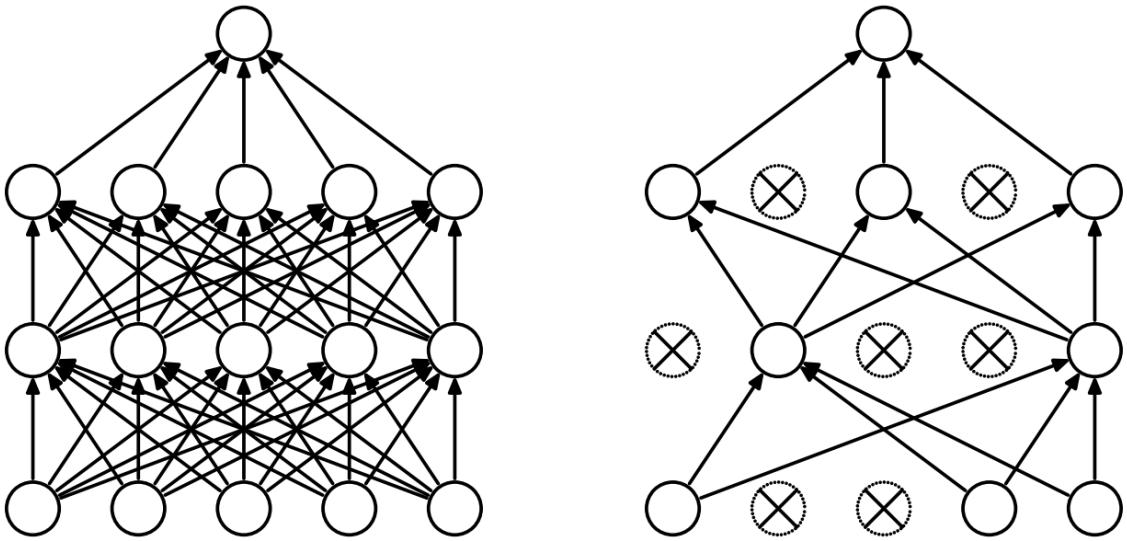
2.5 Regularization

2.5.1 Dropout

Dropout is a regularization technique for neural networks that drops a unit (along with connections) at training time with a specified probability (a common value is

$p = 0.5$)[31]. At test time, all units are present, but with weights scaled by p (i.e. w becomes pw).

The idea is to prevent co-adaptation, where the neural network becomes too reliant on particular connections, as this could be symptomatic of overfitting. Intuitively, dropout can be thought of as creating an implicit ensemble of neural networks.



(a) Standard Neural Net (b) After applying dropout.

Figure 2.9: Dropout Neural Net Model. Left: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the **left**. Crossed units have been dropped..[4]

2.5.2 Stochastic Depth

Stochastic Depth[4] aims to shrink the depth of a network during training, while keeping it unchanged during testing. This is achieved by randomly dropping entire ResBlocks [32] during training and bypassing their transformations through skip connections.

Let $b_l \in \{0, 1\}$ denote a Bernoulli random variable, which indicates whether the th ResBlock is active ($b_l = 1$) or inactive ($b_l = 0$). Further, let us denote the "survival"

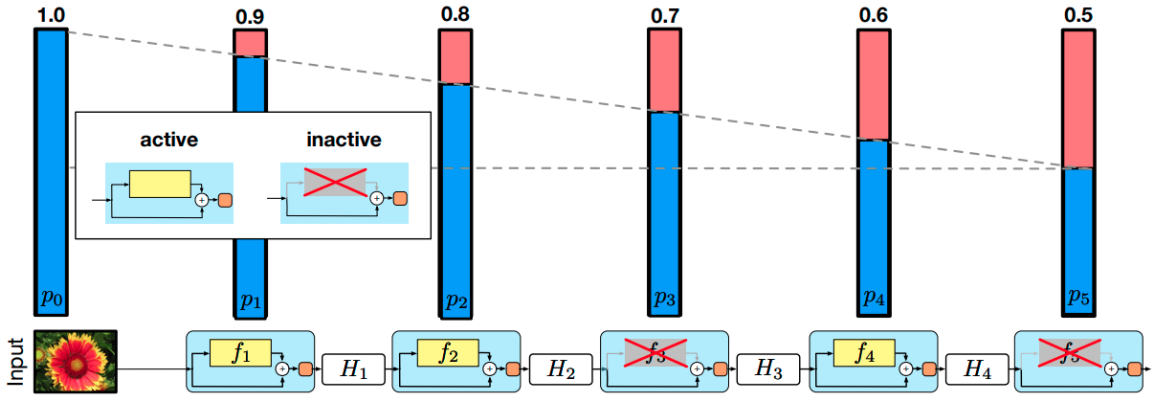


Figure 2.10: The linear decay of p_l illustrated on a ResNet with stochastic depth for $p_0 = 1$ and $p_L = 0.5$. Conceptually, we treat the input to the first ResBlock as H_0, which is always active.[4]

probability of ResBlock as $p_l = \Pr(b_l = 1)$. With this definition we can bypass the lth ResBlock by multiplying its function f_l with b_l and we extend the update rule to:

$$H_l = \text{ReLU}(b_l f_l(H_{l-1}) + \text{id}(H_{l-1})) \tag{2.21}$$

If $b_l = 1$, this reduces to the original ResNet update and this ResBlock remains unchanged.

If $b_l = 0$, the ResBlock reduces to the identity function, $H_l = \text{id}(H_l - 1)$.

Chapter 3

Adaptive graph attention

3.1 Overview

From the base model structure described in [1], we saw several opportunities for change. Therefore, our central concept is to enhance the model layout by either changing the nonlinear transformations or entirely altering the embedding design. Additionally, we experiment with noise learning techniques in the hope that the model can increase prediction accuracy. All of these three components are explained in detail as follows.

The attention central node matrix In this stage, we instantiate the attention relational matrix, through which the model can retain the observer nodes' feature to the central node. Except for the model that uses graph learning, we applied this attention matrix for the models we generate.

Adaptive graph model This stage is a critical component of our approach, designed to enhance the in-stance feature with context information to provide a more accurate representation. Given a probe-gallery pair, we construct a graph to measure the similarity of the target pair. Graph nodes consist of target persons and the associated context pairs, which represent in the relational matrix. We generate our model, which modifies the structure, to learn the similarity between probe-gallery pairs.

Noise learning This stage is an addition to our strategy. We apply noise components to the model based on the restriction of the knowledge collected from the

hops in the graph model. That helps increase the model's learning by self-limiting the information in the early layers, allowing the subsequent layer to learn more.

3.2 The attention central node matrix

The adjacent matrix plays a vital role in describing the relationships between points in the graph while training the graph model. In this experiment, we would like to retain the relationship between the central and observer nodes because we want to gather the feature from the reference nodes to assemble on a central node. To emphasize the knowledge of the central node, we normalize it such that the sum of information from the central node is distributed evenly among the nodes to observe. Then the observer nodes retransmit the amount of information by the inverse quantity of information received.

In particular, considering a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consisting of N vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$. We assign each node with a pair of features $(x_{A_j}, x_{B_j}), j \in \{0, ..., K\}$. If the images have K context pairs, then $N = K + 1$. We use $X \in \mathbb{R}^{N \times 2d}$, where d is the instance-level feature dimension. We use $A \in \mathbb{R}^{N \times N}$ to denote the adjacent matrix associated with graph G. We assign the target node as the first node in the graph, and normalize by assume above, then add self-loop, the adjacent matrix will become:

$$\tilde{A}_{i,j} = \begin{cases} \sqrt{2/N} & if \quad j = 1 \\ \sqrt{N/2} & if \quad i = 1 \\ 1 & if \quad i = j \\ 0 & otherwise \end{cases}, \tag{3.1}$$

Where $i, j \in \{1, ..., N\}$.

3.3 Adaptive graph model

In this section, from the base model described in [1], a three-layer graph convolution network is used to share weight between the two inputs with a header layer used to calculate similarity (ref-image), that we propose three approaches to improve the learning of the model. The first is to change the activation function in the

graph convolution core to help model learning retain more information after each layer. Next is to implement graph learning to help model and self-capture the relationships between nodes (which does not match the attention matrix). The last is to use attention graphs to improve information to the central node.

3.3.1 Graph convolution modified activation network

In this segment, we will change the activation base (ReLU) to SeLU or CeLU based on theoretical for a particular graph-based neural network model [2]. Then we consider a multi-layer Graph Convolution Network (GCN) with two the layer-wise propagation rules from 2.3 as follows:

$$f(H^{(l)}, \tilde{A}) = SeLU\left(\tilde{A}H^{(l)}W^{(l)}\right), \tag{3.2}$$

$$f(H^{(l)}, \tilde{A}) = CeLU\left(\tilde{A}H^{(l)}W^{(l)}\right), \tag{3.3}$$

Here, $\tilde{A} = \hat{A} + I$, is the attention matrix 3.2 of the undirected graph $\mathcal{G}$ with added self-connections. I is the identity matrix, and $\mathcal{W}^{(l)}$ is a layer-specific trainable weight matrix. $H^{(l)} \in R^{N \times D}$ is the matrix of activations in the l^{th} layer; $H^{(0)} = X$. In the following, we show that the structure of GCN model 3.1.

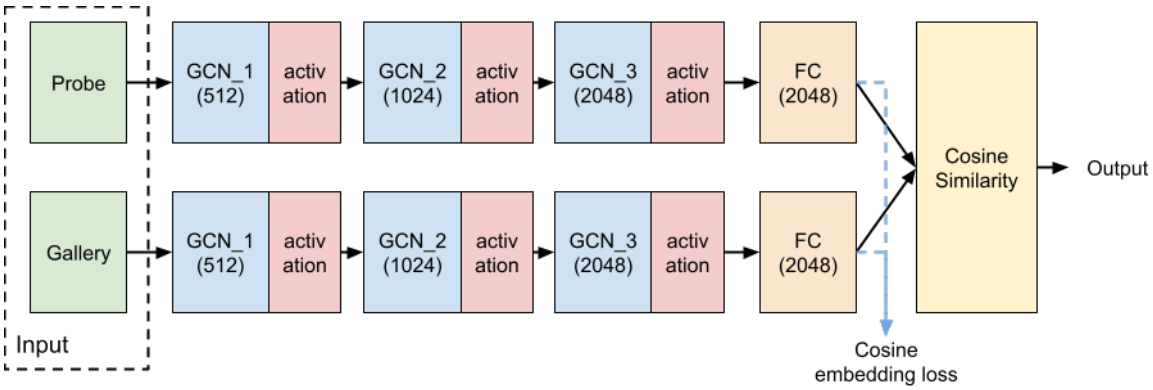


Figure 3.1: GCN model structure with 3 layer graph convolution

3.3.2 Graph attention with graph convolution network

In this segment, to effectively apply the advantages of the graph attention network, we propose a network with three layers. In the first layer, we use the GAT layer 2.1.3 leveraging masked self-attention layers to focus attention on the central node with their neighborhoods' features. By that, the GCN layer deeper to focus on information enrichment in the central node.

However, the use of an additional GAT layer does not achieve the improvement effect. The limit has been mentioned in hops that the number of convolution layers should be equal to the number of hops 2.1.1.4 to ensure that the model can learn sufficiently the relationships between the nodes shown in the adjacency matrix. From there, we propose to use two more GCN layers to focus on information enrichment in the central node.

The setup in detail as seen in the figure 3.2.

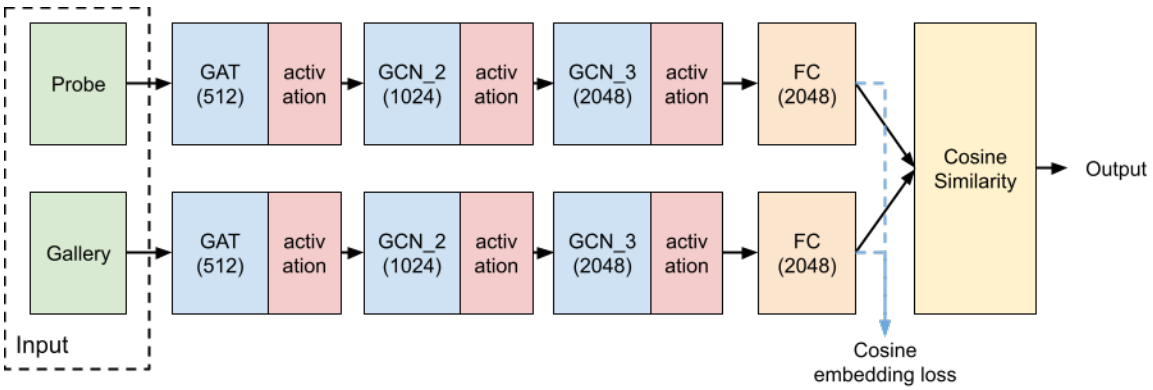


Figure 3.2: Graph attention with graph convolution network structure

3.3.3 Graph learning combined graph convolution network

To increase the model's adaptive ability, we propose to use graph learning 2.1.4 to generate adjacency matrices. In comparison with the attention matrix used previously, the adjacency matrix created by graph learning will be more adaptive for each probe input. Additionally, we add a graph convolution layer to aggregate the

network knowledge. Also, the formula for calculating similarities remains the same as for the previous models.

In the model we propose, the loss function update in two parts, the similarity loss 2.2.1 and the loss for graph learning 2.2.2. The general loss is averaging the component loss.

The setup in detail as seen in the figure 3.3.

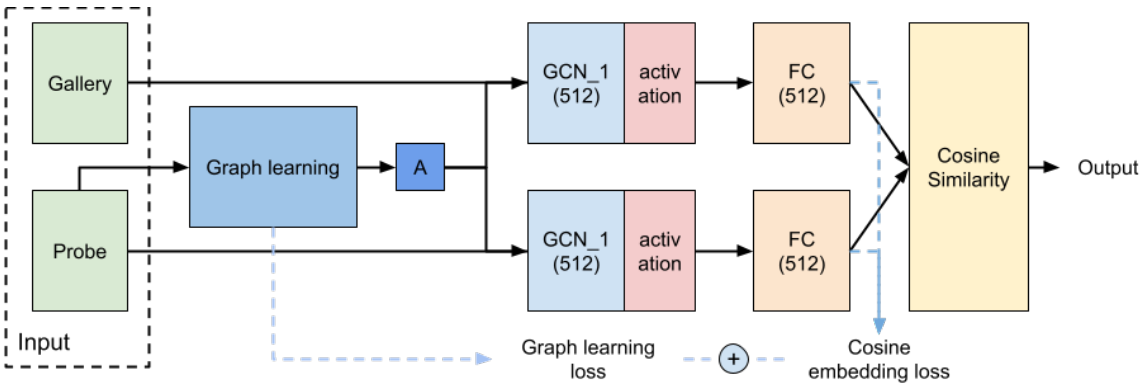


Figure 3.3: Graph learning combined graph convolution network structure

3.4 Noise learning

As mentioned in 2.1.1.4, the number of hops depends on the number of layers of the knowledge graph. In our case, the attention matrix 3.2 only captures the relationship of the central node to the observer nodes, so the maximum number of hops would be 2. According to the theory mentioned, setting the number of layers to accumulate features would not bring more efficiency. Therefore, we exported the addition of noise components to our model structure. The amount of information of the early layers is limited, allowing the subsequent layer to learn more.

3.4.1 Dropout base on level hops

As shown in 2.1.1.4 when the model gets the max number of layers GCN as max level hops, there is no information to collect. It can be lead to the scoring metric

not improve for long epochs of training. While adding a Dropout in the previous layers eventually breaks the amount of information to be learned, this helps the subsequent layers have more information for better learning.

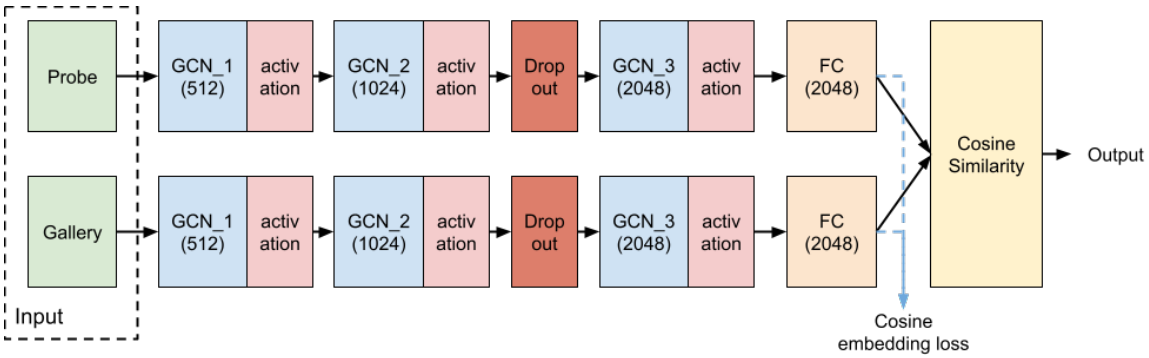


Figure 3.4: Out GCN + dropout model structure with 3 layer graph convolution

3.4.2 Stochastic Depth in Graph Convolution network

Stochastic Depth helps to reduce the network's depth during training by using residual blocks. That is particularly well suited to graph networks with a small number of hops, as in our case.

The Bernoulli random variable mentioned in 2.5.2 will assist the layers in taking turns studying, which is intended to increase each layer's learning level. Base on 2.21, we modify activation to the Selu, and the downsample, which use when our layer inactive, is the algorithm used for interpolation is determined by nearest. The new formula is:

$$H_l = \text{SeLU}\left(b_l GCN_l\left(H_{l-1}\right) + \text{downsample}\left(H_{l-1}\right)\right) \tag{3.4}$$

We substitute existing blocks with new ones to construct a three-layer architecture while preserving the model parameter configuration 3.5.

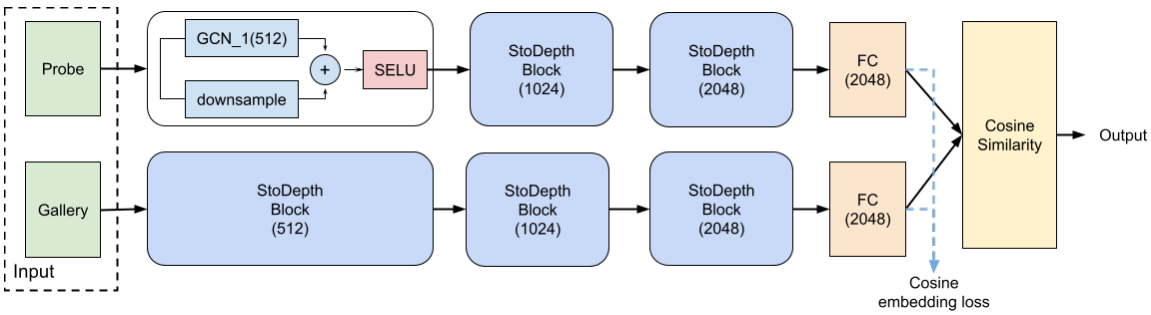


Figure 3.5: Out GCN + StoDepth model structure with 3 layer graph convolution

Chapter 4

Experimental Results

In this section we describe the method of conducting the test. We reworked the load data with the CUHK-SYSU data-set, in order to improve the training efficiency and the correctness of the data. From the models in chapter 3 and the base model defined in [1], we designed the framework to train under the same timing, configuration and data-sets have been reworked. The accuracy of the model will be assessed by the mAP and K-rank scores, then from the results obtained, we conduct analysis and evaluation and make our conclusions.

4.1 Experimental setup

4.1.1 Handle dataset

We evaluate the proposed method on CUHK-SYSU dataset [1]. CUHK-SYSU dataset is a large-scale person search dataset which contains totally 18184 images, with 8432 different persons and 96143 annotated bounding boxes indicating the locations of different pedestrians. This dataset covers diverse scenes, where 12490 images are collected from real street snap and the rest images comes from movies or TVs. For each query person, there exists at least two images containing the target person in gallery set. In addition, this dataset includes large variations in lighting, occlusion, background and resolution, which are close to the real application scenarios.The whole dataset are officially split into training set and testing set. The training set contains 5532 persons in 11206 images, and the testing set contains 2900 query persons and 6978 images.

4.1.1.1 Traindata

Training data is an indispensable part of model training. In this experiment based on the CUHK-SYSU data set and the feature data extracted through RestNet, we process it to form data that the proposed models can learn. In this section we will go through four stages, which is to load the associated datasets, preprocess, select sample the probe object, and finally create the context for the training.

In traindata has 3 sub-file, and all of sub-file has 11206 samples:

- psdb_train_gt_roidb.pkl: Origin data contains the boxes and person had detected in 5532 persons.
- gallery_detections.pkl: The location of bbox with score iou for each image.
- gallery_features.pkl: The feature of bbox.

After loading the relevant files, we remove the bound box samples whose IOU scores are lower than the threshold to filter out noise 6.1.1. At the same time, it is a combination of IDs of related people appearing in the same frame 6.1.2.

When in the interactive, the loader generator randomly the index samples of probe with their bound box features. Then select contextual with each probe by pass all through 6.1.3 pair group with N node to assume the context.

To generator label and gallery for each sample training, we select random person, who not in that frame as neighbor. The frame that probe person in ignore frame sample strain will mask as label, the another will be select randomly 6.1.4.

At last, with probe and gallery select before, we calculate each person feature to each feature of bound box of each frame to fine the most matching 6.1.5.

4.1.1.2 Test dataset

The process has the same with train dataset. In sthis experiment, we use the test set with 100 gallery for each query.

4.1.2 Evaluation metric

We utilize the same evaluation protocol as in [1] with two kinds of evaluation metrics—cumulative matching characteristics (CMC top-K) and mean averaged precision (mAP). The first one is inherited from the person re-id problem, where a matching is counted if there is at least one of the top-K predicted bounding boxes overlaps with the ground truths with intersection-over-union (IoU) greater or equal to 0.5. The second one is inspired from the object detection tasks. The main difference is that a matching is accepted only if the overlap between the bounding box of ground truth person and top-1 matching box is larger than 0.5.

4.1.3 Implementation Details

To train the graph model, we describe the initial structure as the thesis mentioned in chapter 3. We experiment on four groups structure are: the base model with 2 and 3 GCN layers; our model with GCN modified activation; the attention graph concatenates 2 GCN layers; and graph learning concatenate GCN layer. The initial learning rate is 0.1, which decreases by a factor of 2 after 10 epochs, and the total training epoch is 20. We implement our model on Pytorch, and all the models are trained and tested on a GTX 3060Ti GPU. About data training, we create a data loader that follows 4.1.1 for both training and testing.

4.2 Performance

In this section, we report the performance results of our re-make data-loader and compare our model with author [1] methods on CUHK-SYSU datasets. We also compare the results using some noisy training technical.

4.2.1 Performance change after re-make data-loader

After optimize train and test dataset loader performane increasing about 1 to 1.5%, and time training residue 1.5 to 2 times, detail has show in 4.1.

Table 4.1: Performance evaluate Data processing base on Gcn base model with 3 layer and CUHK-SYSU dataset [1].

Data processing	mAP (%)	top-1 (%)	top-5 (%)
Original	80.30	82.00	91.79
Re-make	81.88	83.14	92.76

4.2.2 Comparison with GCN base [1] Methods

After training all the models under the same conditions is the CUHK-SYSU dataset [1] with 20 epochs. The table 4.2 summarizes the results. A brief examination reveals that altering the activation function improved the model's learning capacity.

Table 4.2: Performance evaluate our model Gcn base model on CUHK-SYSU dataset with gallery size of 100

Model	mAP (%)	top-1 (%)	top-5 (%)	top-10 (%)
Gcn 3 layer [1]	81.80	83.14	92.59	95.07
Gcn 2 layer [1]	81.88	82.66	92.86	95.03
Gcn 3 remove ReLU	82.57	83.90	93.10	**95.48**
Gcn 2 remove ReLU	82.63	83.86	93.52	95.38
Gcn 3 CeLU	82.62	84.00	93.31	95.34
Gcn 3 SeLU	82.80	84.21	93.28	95.31
Gcn 2 SeLU	**82.88**	**84.34**	93.24	95.28
GAT + GCN	81.45	82.34	92.76	94.34
GLCN	82.06	83.72	**93.66**	95.24

4.2.3 Comparison model when apply noisy learning

The results of the model training show in the table 4.3. The rapid assessment and training results with noise did not achieve the expected results but did contribute to proving the Hops hypothesis.

Table 4.3: Performance evaluate our model with noisy training.

Regularization	mAP (%)	top-1 (%)	top-5 (%)	top-10 (%)
Gcn 3 SeLU	**82.80**	**84.21**	93.28	95.31
Gcn 3 Stochastic Depth	81.51	82.55	92.48	94.59
Gcn 3 Dropout	82.06	83.14	92.97	94.90

4.3 Analysis

This section focuses on analyzing and evaluating the research outcomes from the testing process. We concentrate on reiterating the efficiency of load, doing an aspect analysis on each sample we have, and eventually examining how the impact of training with noise affects the model's training.

4.3.1 Results on re-make data loader

Apart from that, the index deviation is initially set to 1 rather than 0. As a consequence of this circumstance, a wide-ranging bias developed. In light of these shortcomings, we reorganized the data structure to ensure the accuracy of the training dataset.

After remaking, data loading output was increased by 1.5-2 times due to eliminating data replication errors. This data is a better fit for the node id, allowing for a more accurate learning model. The results shown in table 4.1 have shown a marked effect after the modification, increasing by 1.5% mAP and 1% for top-1. These results undoubtedly provide a great basis for us to continue our experiments.

4.3.2 Results Comparison model

We summarize the comparative results in Table 4.2. The gallery size of all methods is 100. In comparison to the label using hand-crafted features, we observe that all our learning methods achieve slightly better performance improvements. The model with Gcn 2 and 3 layers [1] can be viewed as an overall baseline of the proposed framework, our framework achieves at least 0.2% improvements on mAP and 0.6% on top-1 matching rate, which demonstrates the effectiveness of the proposed frameworks.

Changing the activation function has brought a number of positive effects. That was demonstrated by the results of the table, specifically an increase of about 1% for both mAP and top-1 scores. This effect arises from changing the activation from relu to ones of celu and selu. With relu, the non-linear space is confined to the positive, which makes the negative components cancel out. However, as shown

above, we have seen that the negative components have a beneficial impact on the model's learning. The inclusion of two activation functions, celu or selu, proved that with a nonlinear expansion of the negative part. With celu the negative part space is smaller than selu, which can be thought to be the reason for the better use of selenium model. Nevertheless, it is not desirable to maintain negative values. When we removed the activation layer in our test, i.e., no nonlinear spaces, the model with a maximum of 82.63 mAP was still lower than the 82.88 mAP of the model using selu. We conclude from this that the excitation or selu usage model is the optimal one for our test.

The results affirm once again that the number of the graph convolution layers is dependent on the number of hops 2.1.1.4. Specifically, the maximum number of hops specified with the adjacency matrix we have is two, which also explains why a network with two graph convolutional layers outperformed a network with three layers in terms of performance before, and after modifying the activation function.

The expectation with Model GAT+GCN 3.3.2 is to use the GAT layer to navigate features and funnel them to a central node where the GCN layer can learn more effectively. In comparison to the baseline model, though, this model is inefficient. Consequently, it can be inferred that using the adjacency matrix as an attention matrix is more efficient than using graph attention to navigate features to the central node in the Re-ID task with contextual.

In the results of the graph learning model, basically we can see the self-study of adjacency matrices has proven its effectiveness, as it has improved by 0.2% mAP and led in top-5. The application of graph learning in learning adjacency matrices without using the attention matrix 3.2 is extremely promising because it is capable of adapting itself to various contexts appropriately. This strength is advantageous for the Re-ID problem on multiple objects simultaneously, as it can help self-learn complexity in the context and optimize multiple output targets. However, the adjacency matrix posed has not really worked out superior to the attention matrix initiated from the beginning for the issue we set out.

4.3.3 Results when apply noisy learning

The purpose of this experiment is to increase the noise in order to assess the deeper layers' ability to learn. Additionally, the findings demonstrated that adding noise simply makes learning more difficult because the simulations do not yield optimal results.

The most significant effect is the use of stochastic depth when the network using this training gets the lowest score of 81.51 mAP. This can be explained by the skip layer in stochastic depth.

When dropout is applied to the experiment, we can see from the result of the baseline model that the hops limit actually occurs since the 2-layer model performs better than the 3-layer model. Thus, in this experiment, we attempted to improve learning performance by adding a dropout between grades 2 and 3. The result is partly positive when it has learned better the baseline model.

To be honest, our experiment's use of two distinct methods of learning did not turn out well. The explanation for this is that, according to the studies we refer to, it all needs to take more than one hundred training rounds for the effect to become evident.

4.4 Discussion

Modifying the prototype data structure created a meaningful contribution to our experiments. This improvement serves as the foundation and central value for our evaluation of the systems we propose.

The use of relu by the original GCN greatly limits the ability of our model to learn, which our experiments have shown. Depending on the purpose towards that we have a suitable solution. But in this thesis we have created a new architecture using selu.

Other learning models, such as GAT or graph learning, are highly appreciated but

do not bring too much efficiency. Assuming our issue aims at multiple output targets, the superior adaptability of a GAT or learning graph can be advantageous. Additionally, it is also our future test orientation.

The application of training methods with noise are being applied very effectively can be mentioned in the APT article. Unfortunately, due to the limited trial time of the study, we did not have the resources to evaluate the effectiveness of the method. We will improve this issue in the subsequent report.

Chapter 5

Conclusion

The technology landscape is continuously developing and changing. Demand for advanced technologies never stops increasing as people constantly tackle new challenges or make new discoveries. Besides that, the demand for identification technology will be increasing, as it is the basis for user support applications as well as collective management. To keep up with that trend, the optimization of the computational complexity of the RE-ID task is critical. It can be said that the application of the context has actually opened a path toward where we can make more accurate predictions at a low computational cost.

We have made two propositions in this thesis. First is the improvement of the data loading method described in Chapter 4. Second, studies were conducted to expand and improve the model's learning capabilities through the graphs' application. Along with the coupling from existing models, the proposed models include a GCN architecture based on selu rather than the conventional GCN architecture based on Relu stated in Chapter 3. This method gives better identification results than the traditional method mentioned in the author's research.

We discovered during the experiment that maintaining a definition of the directed adjacency matrix assists the model in learning. The result is superior in studies using adaptive models such as GAT or GLCN. This is due to the orientation supporting the model in learning effectively from the beginning, allowing it to progress further under the same time limit. Besides, the expansion of the nonlinear space from relu to selu has also shown a positive effect as the model can learn better.

In future work, we will develop our simultaneous multi-label recognition institute deeply with the desire to maximize the subject defined per frame. Then the use of more adaptive networks will be able to exert its effectiveness in learning complex contexts instead of the oriented contexts as in the experiment. On the other hand, there is the application of noise learning methods to improve model quality by breaking down the hops restriction. The limitation of Hops can only be solved when we can represent the relationships of the nodes referenced through an adjacency matrix or have to pass information from the head layers straight to the deeper layers. That is a direction that we rationally expect will improve the ability of the model to learn.

Chapter 6

Appendix

6.1 process data

6.1.1 filter

```
def filter(traindetections,trainfeatures,thresh=0.0):
    det = []
    feat = []
    for item,item2 in zip(traindetections,trainfeatures):
        ids = item[:,4]>thresh
        item1 = item[ids]
        item2 = item2[ids,:,0,0]
        det.append(item1)
        feat.append(item2)
return det, feat
```

6.1.2 getIndex

```
def getIndex(self):
    personSet = {}
    for i in range(len(self.traindata)):
        ids = self.traindata[i]['gt_pids']
        detection = self.det[i]
        if len(detection)==0: continue
        for j,person in enumerate(ids):
            if person==-1: continue
            try:
                tmp = [i,detection[j]]
            except:
                continue
```

```
            try:
                personSet[person].append(tmp)
            except:
                personSet[person] = [tmp]
    return personSet
```

6.1.3 getItem

```
def select_p_n(det,feat_p,roi,num=4):
    def _compute_comp(a, b,iou_thresh=0.0):
        x1 = max(a[0], b[0])
        y1 = max(a[1], b[1])
        x2 = min(a[2], b[2])
        y2 = min(a[3], b[3])
        inter = max(0, x2 - x1) * max(0, y2 - y1)
        union = (a[2] - a[0]) * (a[3] - a[1]) + \
                (b[2] - b[0]) * (b[3] - b[1]) - inter
        iou = inter * 1.0 / union
        if iou>iou_thresh:
            return 0
        else:
            return a[4]
    detN = det.shape[0]
    if detN==1:
        feat1 = feat_p[0]
        featn = [feat_p[0]]*num
        return feat1,np.asarray(featn)

    index = np.zeros(detN)
    for i in range(detN):
        index[i] = _compute_iou(det[i],roi)
    index = [[x, i] for i, x in enumerate(index)]
    index = sorted(index, key=lambda x: x[0], reverse=True)
    feat1 = feat_p[index[0][1]]

    index = np.zeros(detN)
    for i in range(detN):
        index[i] = _compute_comp(det[i], roi)
    index = [[x, i] for i, x in enumerate(index)]
    index = sorted(index, key=lambda x: x[0], reverse=True)

    featn = [feat1]
    for i in range(num-1):
        if i>=detN-1:
```

```
            featn.append(feat_p[index[1][1]])
        else:
        featn.append(feat_p[index[i][1]])
    return feat1,np.asarray(featn)
```

6.1.4 getGallery

```
def getGallery(probe, randIdx, sample, feat):
    slice = random.sample(sample,100)
    label = [0]*100
    gallery = []
    idx = list(range(len(probe)))
    idx.remove(randIdx)

    for i in range(100):
        tmp = feat[slice[i]]
        gallery.append(tmp)

    if len(probe)==1:
        return label,gallery

    for i, ix in enumerate(idx):
        item = probe[ix]
        if item[0] in slice:
            id = slice.index(item[0])
            tmp = random.sample(sample,1)[0]
            gallery[id] = feat[tmp]
        gallery[i] = feat[item[0]]
        label[i] = 1
    if not sum(label) == sum(label[:sum(label)]):
        print("error")
    return label, gallery
```

6.1.5 gallery gcn

```
def gallery_gcn(probe, gallery_feat, num=3):
    if len(gallery_feat) == 99:
        gallery_feat.append(gallery_feat[-1])
    featuresave = [[]] * self.neibor

    for i in range(num):
        tmp = []
```

```
        for j, item in enumerate(gallery_feat):
            sim0 = -1
            try:
                nowfeature = item[0]
            except Exception:
                continue
            for id, person in enumerate(item):
                sim = Sim1(probe[i], person)
                if sim > sim0:
                    sim0 = sim
                    nowfeature = person
            tmp.append(nowfeature)
            del nowfeature
        featuresave[i] = tmp

    gallery = []
    for i in range(len(gallery_feat)):
        feature = []
        for j in range(num):
            try:
                feature.append(featuresave[j][i])
            except Exception:
                break
        else:
            gallery.append(feature)

    if len(gallery) == 99:
        gallery.append(gallery[-1])
        gallery = np.asarray(gallery)
    return np.asarray(gallery)
```

6.2 Model diagram

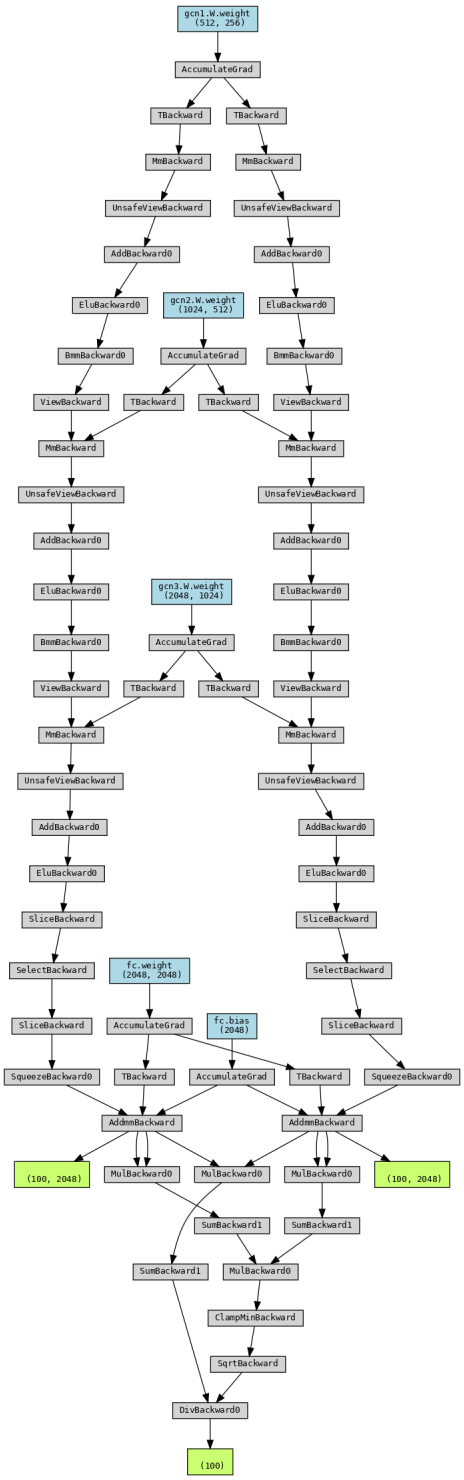


Figure 6.1: Out GCN model structure with 3 layer graph convolution

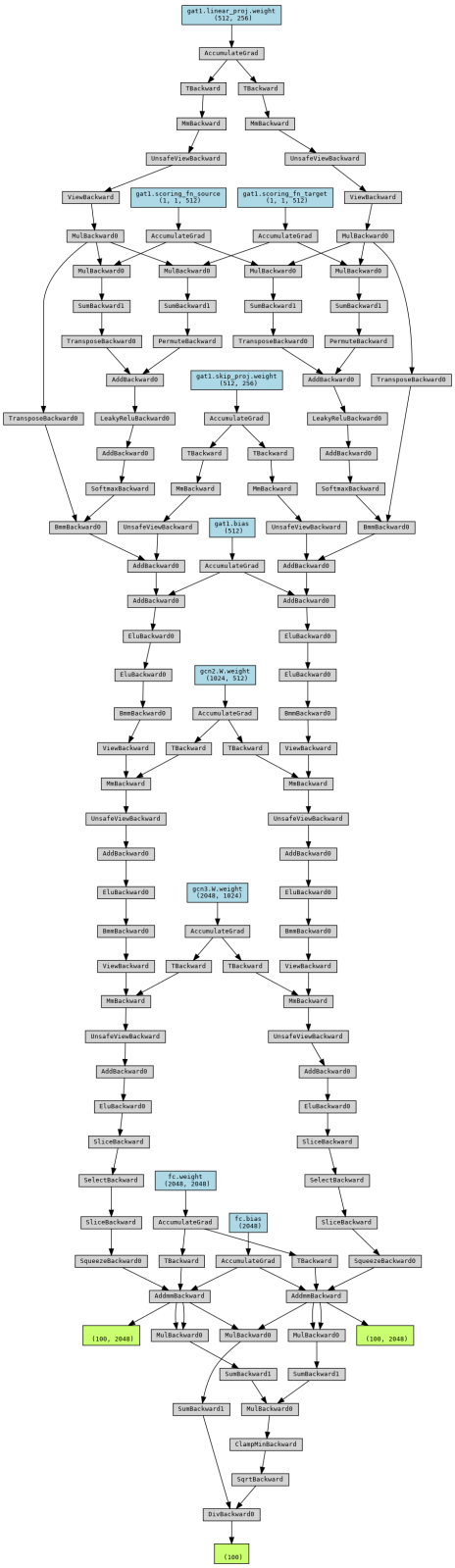


Figure 6.2: GAT + GCN model diagram

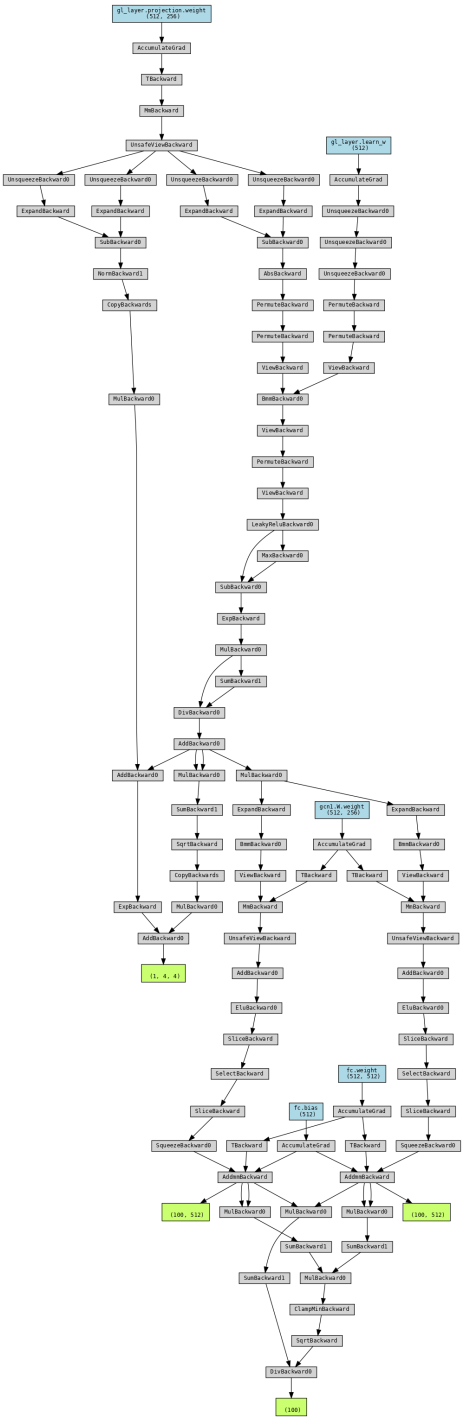


Figure 6.3: Graph learning with GCN network structure

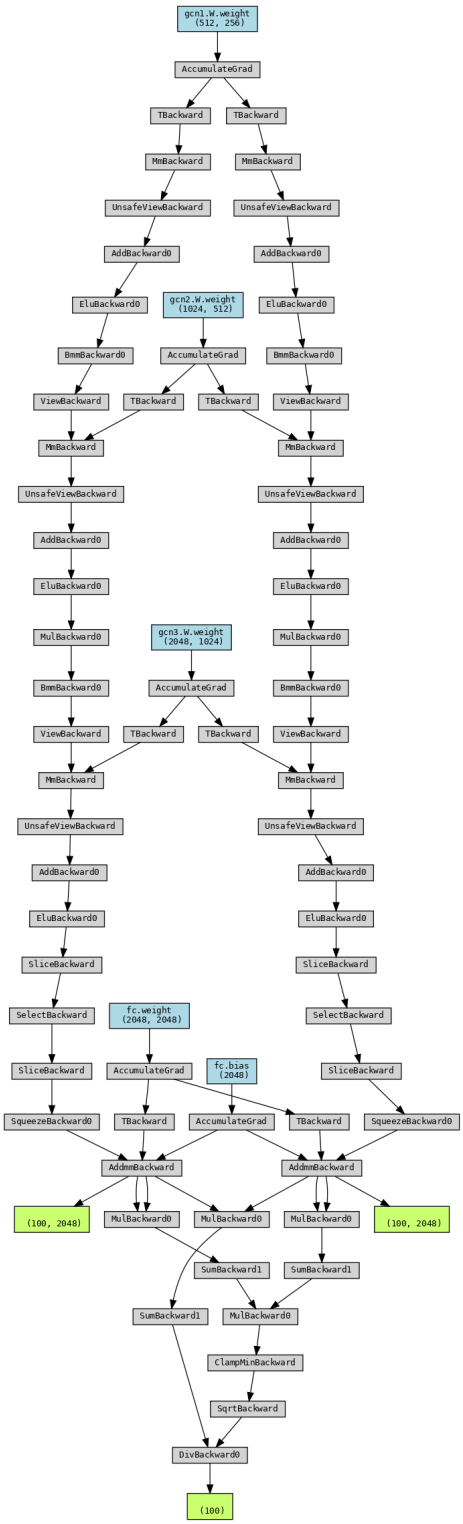


Figure 6.4: Graph convolution network add Dropout

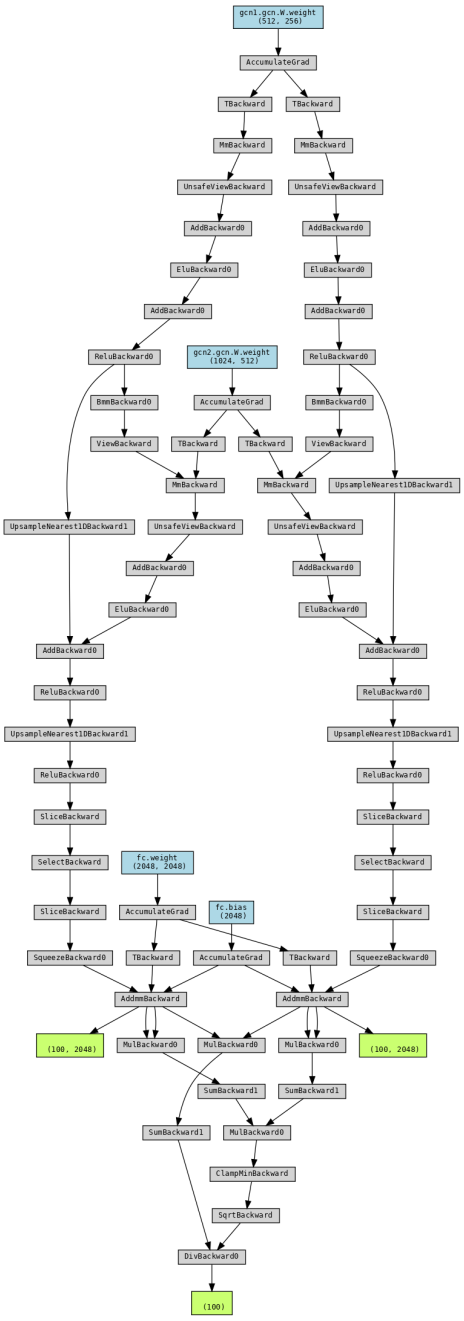


Figure 6.5: Graph convolution network and StoDepth

References

[1] Tong Xiao, Shuang Li, Bochao Wang, Liang Lin, and Xiaogang Wang. Joint detection and identification feature learning for person search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3415–3424, 2017.

[2] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017.

[3] D. Saire and A. Ramírez Rivera. Graph learning network: A structure learning algorithm. In *Workshop on Learning and Reasoning with Graph-Structured Data (ICMLW 2019)*, June 2019.

[4] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. Deep networks with stochastic depth, 2016.

[5] Yuanlu Xu, Bingpeng Ma, Rui Huang, and Liang Lin. Person search in a scene by jointly modeling people commonness and person uniqueness. In *Proceedings of the 22nd ACM International Conference on Multimedia*, MM '14, page 937–940, New York, NY, USA, 2014. Association for Computing Machinery.

[6] G. Lisanti, N. Martinel, A. Del Bimbo, and G. L. Foresti. Group re-identification via unsupervised transfer of sparse features encoding. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2468–2477, 2017.

[7] Shayan Modiri Assari, Haroon Idrees, and Mubarak Shah. Human re-identification in crowd videos using personal, social and environmental constraints. In *ECCV (2)*, pages 119–136, 2016.

[8] M. Cao, C. Chen, X. Hu, and S. Peng. From groups to co-traveler sets: Pair matching based person re-identification framework. In *2017 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 2573–2582, Los Alamitos, CA, USA, oct 2017. IEEE Computer Society.

[9] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks, 2020.

[10] Rianne van den Berg, Thomas N. Kipf, and Max Welling. Graph convolutional matrix completion, 2017.

[11] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs, 2014.

[12] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data, 2015.

[13] Sofia Ira Ktena, Sarah Parisot, Enzo Ferrante, Martin Rajchl, Matthew Lee, Ben Glocker, and Daniel Rueckert. Distance metric learning using graph convolutional networks: Application to functional brain networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 469–477. Springer, 2017.

[14] Bang Liu, Ting Zhang, Di Niu, Jinghong Lin, Kunfeng Lai, and Yu Xu. Matching long text documents via graph convolutional networks. *arXiv preprint arXiv:1802.07459*, pages 2793–2799, 2018.

[15] Siyuan Qi, Wenguan Wang, Baoxiong Jia, Jianbing Shen, and Song-Chun Zhu. Learning human-object interactions by graph parsing neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 401–417, 2018.

[16] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *Proceedings of the European conference on computer vision (ECCV)*, pages 399–417, 2018.

[17] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*, 2017.

[18] Dapeng Chen, Dan Xu, Hongsheng Li, Nicu Sebe, and Xiaogang Wang. Group consistent similarity learning via deep crf for person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8649–8658, 2018.

[19] Yantao Shen, Hongsheng Li, Shuai Yi, Dapeng Chen, and Xiaogang Wang. Person re-identification with deep similarity-guided graph neural network. In *Proceedings of the European conference on computer vision (ECCV)*, pages 486–504, 2018.

[20] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[21] William L. Hamilton. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159, 2020.

[22] Eric W Weisstein. Laplacian matrix. *https://mathworld. wolfram. com/*, 1999.

[23] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[24] Wenwen Yu, Ning Lu, Xianbiao Qi, Ping Gong, and Rong Xiao. Pick: Processing key information extraction from documents using improved graph learning-convolutional networks, 2020.

[25] Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo. Semi-supervised learning with graph learning-convolutional networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11305–11312, 2019.

[26] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress.

[27] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks, 2017.

[28] Jonathan T. Barron. Continuously differentiable exponential linear units, 2017.

[29] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2016.

[30] Mu Zhu. Recall, precision and average precision. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, 2(30):6, 2004.

[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

Adaptive Graph Attention Network in Person Re-Identification

Final Year Project Report

A 4th Year Student Name

Le Dinh Duy
HE130655

Under the supervision of

Dr. Phan Duy Hung

Bachelor of Computer Science

Hoa Lac Campus – FPT University

Spring 2021