Advancing Skeleton-based Action Recognition: Exploring New Loss Functions and Architectures

Project Report submitted by

Nalla Amulya (420211)
Veeramalla Sumukh (420249)
Kondapalli Jagadeesh (420149)

Under the supervision of

Dr. Nagesh Bhattu Sristy
Assistant Professor

Department of Computer Science and Engineering
National Institute of Technology, Andhra Pradesh

9 May, 2024

Advancing Skeleton-based Action Recognition: Exploring New Loss Functions and Architectures

Project Report submitted by

Nalla Amulya (420211)
Veeramalla Sumukh (420249)
Kondapalli Jagadeesh (420149)

Under the supervision of

Dr. Nagesh Bhattu Sristy
Assistant Professor

Department of Computer Science and Engineering
National Institute of Technology, Andhra Pradesh

9 May, 2024

APPROVAL SHEET

This project work entitled “Advancing Skeleton-based Action Recognition: Exploring New Loss Functions and Architectures” worked out by Nalla Amulya(420211), Veeramalla Sumukh(420249), and Kondapalli Jagadeesh(420149) is approved for the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

Examiners

_____

_____

_____

Supervisor(s)

_____

_____

Chairman

_____

Date : _____

Place : _____

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Nalla Amulya	Veeramalla Sumukh	Kondapalli Jagadeesh
420211	420249	420149
Date:	Date:	Date:

Department of Computer Science and Engineering

NATIONAL INSTITUTE OF TECHNOLOGY, ANDHRA PRADESH

Certificate

This is to certify the project report entitled "Advancing Skeleton-based Action Recognition: Exploring New Loss Functions and Architectures" submitted by Nalla Amulya, Roll No. 420211, Veeramalla Sumukh, Roll No. 420249, Kondapalli Jagadeesh, Roll No. 420149 to National Institute of Technology, Andhra Pradesh in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a record of bonafide research work carried out by them under my supervision and guidance. This work has not been submitted elsewhere for the award of any degree.

Dr. Nagesh Bhattu Sristy
(Project Guide)

Place: Tadepalligudam
Date:

Table of Contents

List of Figures

List of Tables

Abstract

The project aims to develop an efficient and robust model for skeleton-based activity recognition which can recognize and classify different actions, given video footage. The development of video capture technologies has led to the emergence of video representations. Since human skeletons offer a condensed data format to illustrate how the human body moves dynamically, skeleton-based representations[1] are currently highly popular for human action recognition. Even though we've made progress in understanding the 3D skeleton data, we still struggle to capture all the important information it holds. This is especially true when it comes to figuring out how different joint movements are connected and what they mean together.

Recently, the dominant approach for identifying actions based on skeletal information involves using Graph Neural Networks (GNNs), specifically leaning towards a type known as Graph Convolutional Networks (GCNs)[2]. Although these models can achieve high accuracy on benchmark datasets, they come with several constraints and limitations that make these models impractical for usage in real-world scenarios.

Nevertheless, three common problems are noted:(1) Because the models are typically sophisticated, their computational complexity is also higher. (2) Relying on labels during training is a constant downside for supervised learning models[3, 2]. (3) Large model implementations are not helpful for real-time applications. In order to address these issues, we aim to develop novel architectures that effectively model dependencies between joints, allowing for a more comprehensive understanding of how different joint movements are connected and their collective significance in activity recognition, along with an aim to limit the model's complexity and its parameters.

Chapter 1

Introduction

Since video data records many frames over time and image data only records one frame, video data is richer than image data. A video's individual frames can reveal additional information on the positions, motions, and interactions of the objects in the scene. A video can offer a deeper comprehension of the scene's dynamics and context by analyzing many frames. Additionally, temporal data from video recordings can be utilized to identify activities and track objects across time. This temporal data includes things like the sequence and length of actions. In addition, video data is capable of extracting both spatial and temporal information, unlike image data which is only able to extract one of these. The ability to record timely changes, like motions of objects or people and weather or lighting changes, is another benefit of video data over image data. This can be used to examine how items behave over time and can provide more details about the scene than one image could.

Human activity recognition, within the field of computer vision, involves a process to identify and comprehend human actions depicted in video data. Our project's goal is to develop an approach that can classify and recognize different actions performed by humans in videos with reduced consumption of system resources (Infrastructure). This can have various applications, such as sports analysis, surveillance, and human-computer interaction.

The foundation of our work lies in the ConMLP framework[17] which includes Self-Supervised Contrastive Learning for Action Recognition[1, 5]. Building upon this base paper, we delve into a series of experiments aimed at augmenting the capabilities of the model and improving the model's performance on challenging datasets. Our research endeavors cover a wide variety of methodologies, including the exploration of alternative loss functions, integration of graph-based representations, and the incorporation of recurrent neural network architectures. By systematically examining these approaches, we aim to push the boundaries of skeleton-based activity recognition and prepare a path for more accurate and robust systems.

Furthermore, we experiment with the integration of graph-based representations into recurrent neural network architectures, aiming to capture intricate dependencies between skeletal joints and improve the temporal modeling of human actions. Leveraging graph neural networks and recurrent units such as LSTM and GRU, we seek to extract more informative features from the skeletal data, thereby improving the model's capacity to identify intricate activities with greater accuracy and robustness. Through comprehensive experimentation and analysis, our objective is to contribute towards the development of more efficient and robust models for skeleton-based activity recognition, thereby advancing the state-of-the-art in this domain.

One of the primary objectives of our project is to investigate the efficacy of different loss

functions in enhancing the discriminative power of the model. In particular, we explore the substitution of the contrasting loss function with a tuned variant, as well as the addition of MSE(mean squared error) loss to the supervised contrasting loss[14]. Through empirical evaluation of benchmark datasets like the NTU RGB+D, we aim to ascertain the effect of these modifications on the model's performance and identify avenues for improvement.

Furthermore, we introduce an entirely novel approach that combines a self-reconstruction task with contrastive learning[4, 12]. It has been confirmed that the pairing of a contrastive loss and a self-reconstruction loss produces representations of sufficiently high granularity. The suggested learning paradigm produces superior performance in many fine-grained categorization tasks. We present Contrastive Learning + Reconstruction loss - a learning algorithm that is self-supervised and simultaneously optimizes a contrastive loss and a self-reconstruction loss to produce representations. We demonstrate that current cutting-edge techniques for contrastive learning[4, 12] struggle to accurately describe fine-grained visual features. We extend the SimCLR framework by including a self-reconstruction task. This is achieved by using a straightforward two-head encoder-decoder architecture. We show that this extension contributes toward a more refined vector representation with detailed visual characteristics. Combining these concepts, SimCLR + Reconstruction Loss performs better than SimCLR for skeleton-based action recognition.

Chapter 2

Literature Review

2.1 Extensive Survey of Vision-Based Methods for Recognizing Human Actions

Here, the primary goal is to obtain features using a variety of techniques. These characteristics can then be utilized to train a classifier or regressor to recognize or predict actions as mentioned in [19].

2.1.1 Human Action Feature Representation Methods:

This paper [19] explains a number of feature extraction techniques. The following are a few of the paper's detailed techniques:

1. Methods for Representing Action Features Using Deep Learning
2. Manually Designed Action Features for RGB Data
3. Manually Designed Action Features for Depth and Skeleton Data

2.1.2 Manually Designed Action Features for RGB Data:

Methods Based on Spatiotemporal Volumes: This is a template matching technique, however instead of employing image processing to recognize objects, they use a computationally costly three-dimensional spatiotemporal pattern. Trajectory-based features- Using an optical flow to track these feature points allows for the calculation of displacement information from video frame-sampled dense point clouds. Difficulties: This approach requires an accurate two- or three-dimensional model of human skeleton to match moving object pixel intensities with the background.

2.1.3 Manually Designed Action Features for Depth and Skeleton Data:

1. **Depth sequence-based methods**: This method describes actions mainly by describing the human body's depth map changes with motion.

2. **Skeleton-based methods**: Techniques built on the human skeletal sequence [1] employ alterations in the joints of humans throughout video frames to depict the activity, encompassing shifts in the joint points' positions and appearances. Occlusion in the scene will cause the joint point estimate to be either missing or inaccurate.

3. **Feature fusion methods :** In order to eliminate occlusion and perspective changes-related mistakes in the skeleton feature, this method combines the joint feature with the depth information feature. The action recognition algorithm's computational complexity is heightened by these considerations.

2.2 NTURGB+D: A Large-Scale Dataset for 3D Human Activity Analysis

Shahroudy et al. (2016) present NTU RGB+D, a novel large-scale dataset for 3D human activity recognition, in their work "NTURGB+D: A Large-Scale Dataset for 3D Human Activity Analysis" [11]. NTU RGB+D is made up of more than 4 million frames and 56,000 video clips that were gathered from 40 different participants executing 60 different activity classes. The dataset is captured using both RGB and depth cameras, providing rich information about the 3D pose and motion of the subjects.

At the time of publishing, NTU RGB+D was the largest dataset available for 3D human activity recognition. It was also one of the first datasets to include both RGB and depth data, which made it possible to develop new methods for 3D human activity recognition that was more robust to noise and variability than the previous methods that relied on RGB data alone.

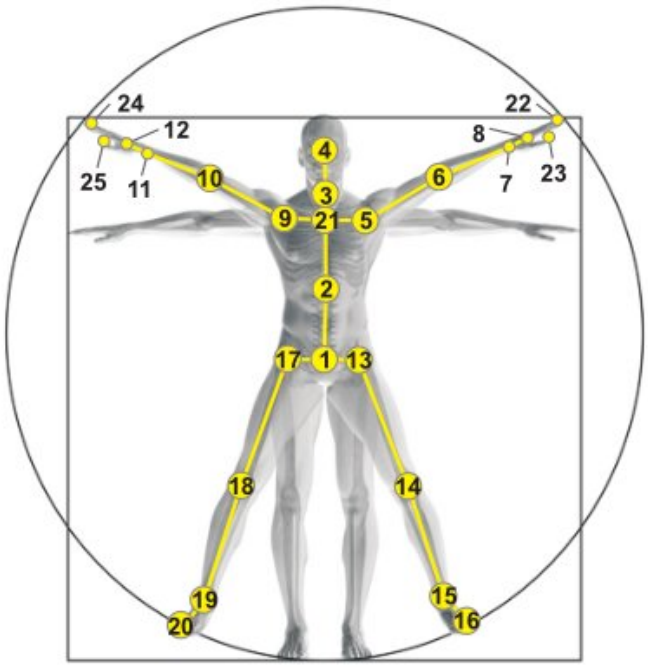

Figure 2.1: Configuration of NTURGB+D's 25 body joints

2.3 ConMLP for Skeleton Action Recognition

The ConMLP (Contrastive Multilayer Perceptron) model[17] has garnered attention in the literature as a novel approach that leverages contrastive learning[4, 12] techniques within the context of multilayer perceptron architectures[6]. This fusion of contrastive learning and MLP principles aims to enhance representation learning and feature extraction capabilities.
The ConMLP architecture typically comprises multiple layers, each equipped with non-linear activation functions to capture intricate relationships within the data. The contrastive loss function[4] is instrumental in guiding the model to align representations of similar samples while pushing apart those of dissimilar ones. This results in a feature space where semantically similar instances are clustered together.

2.4 Tuned-Contrastive Learning

Tuned-Contrastive Learning (TCL) [15] has proven itself to be a significant development in contrastive learning, introducing key enhancements to address specific challenges and limitations inherent in traditional contrastive loss functions. This literature review summarizes TCL's foundational principles, advancements, and applications within the context of machine learning research.

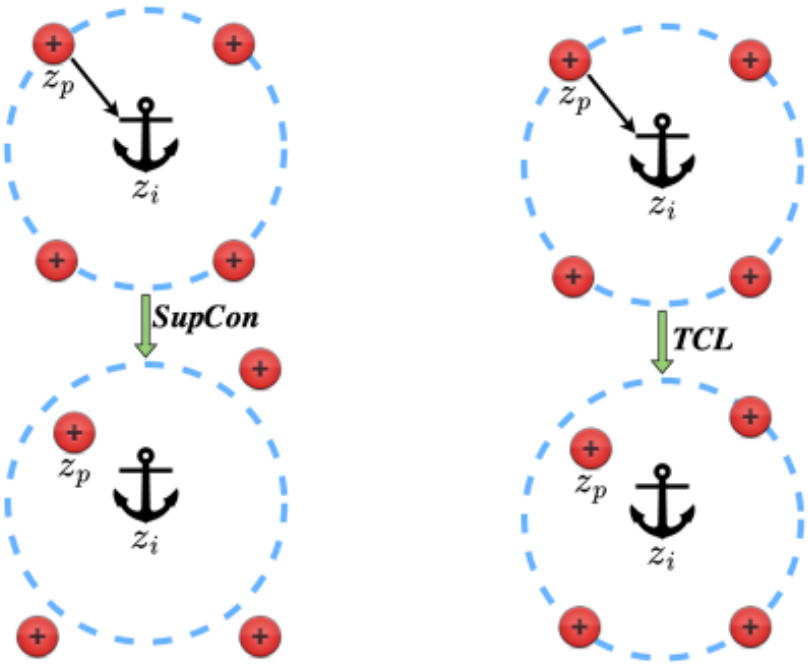

Figure 2.2: TCL differs from SupCon loss.

The Foundations of Tuned Contrastive Learning:

- Multitude of Positives and Negatives: TCL[15] distinguishes itself by accommodating a

batch with multiple positives and negatives during training. This adaptability allows the model to harness a richer set of relationships within the data, providing more comprehensive and nuanced representations.

- Addressing Implicit Negatives as Positives:One notable limitation in traditional contrastive losses is the implicit consideration of positives as negatives. TCL introduces mechanisms to rectify this issue, ensuring that hard positives are not inadvertently treated as negatives. This correction leads to more accurate and discriminative representations.
- Regulating Hard Negative Gradient Response: TCL goes a step further by offering parameters to regulate the gradient response from hard negatives. This additional tuning facilitates better control over the learning dynamics, preventing issues associated with overly aggressive negative gradient responses.

2.5 graph2vec: Learning Distributed Representations of Graphs

The approach employed in the Graph2Vec model[16], is founded on the concept of treating graphs akin to "documents," drawing inspiration from techniques in natural language processing (NLP) to derive continuous representations for entire graphs. This process involves encoding both the structural attributes of the graph, extracted through graph kernels, and the features associated with individual nodes, into fixed-length feature vectors. This amalgamation yields comprehensive representations that encapsulate both the graph's local and global properties, facilitating a deeper understanding of its semantics.

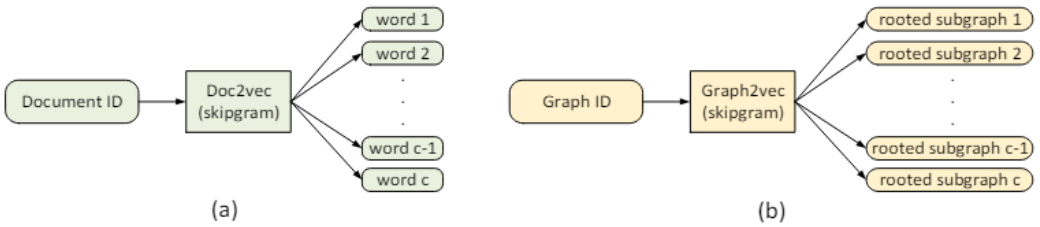

Figure 2.3: (a) doc2vec's skipgram model vs (b) graph2vec

Integral to the graph2Vec approach is the incorporation of graph kernels, which play a pivotal role in capturing the underlying structural patterns and relationships within the graph. By extracting these patterns, the model can generate embeddings that possess enhanced discriminative power, enabling effective graph classification and similarity estimation tasks. Subsequently, these feature vectors are inputted into a neural network model, which learns to map the input graph representations to continuous embeddings. This neural network architecture enables the model to capture intricate relationships within the graph data, facilitating the generation of informative and meaningful embeddings.

The versatility of the learned graph embeddings extends to various downstream applications, including graph classification, clustering, and similarity estimation. These embeddings serve as

compact yet comprehensive representations of the graph data, empowering more effective analysis and interpretation across diverse application domains. Through the fusion of graph kernel techniques, neural network modeling, and NLP-inspired methods, the graph2Vec approach[16] offers a holistic framework for learning continuous representations of entire graphs, heralding new possibilities in graph analytics and machine learning research.

Chapter 3

Approaches

3.1 ConMLP-with Contrastive Learning

For action recognition and prediction based on skeletons, as mentioned in the literature review section above, we have chosen to employ a skeleton-based model. This model's simplicity in architecture and reduced processing requirements for handling skeleton data are the primary motivations for its use. We used this naive approach in the domain of self-supervised contrastive learning[4, 5, 10] for action recognition and skeleton data analysis tasks. Through the fusion of multilayer perceptrons and contrastive learning, this framework excels in capturing intricate spatial-temporal patterns inherent in skeleton data, paving the way for more robust and adaptive action classification models.

3.1.1 Architecture

We have a series of skeletal representations, denoted as $\{\mathrm{x}\} = \{x_1..x_T\}$. Each x_i contains the S subjects' spatial 3D coordinates with J distinct 3D body joints, forming a sequence of T consecutive frames. Our training dataset $\Phi = \{\mathbf{x}_i\}_{i=1}^{N}$ is comprised of N such sequences, each corresponding to different actions. It's worth noting that these sequences may differ in viewpoints and subjects. Every sequence x_i is associated with a label y_i, where y_i indicates the specific action class it belongs to, out of a total of c classes.The main objective here is to derive meaningful representations from the skeletal sequences x_i without relying on explicit labels during the learning process. In other words, the goal is to capture the inherent features and patterns in the data without using predefined class labels.

Initially, the input mini-batch data undergoes data augmentations[9] for diversification. Two augmented versions of the mini-batch data are then passed through the foundational encoder network. Further, the projection head fine-tunes the final representations,a part not used in the inference stage. On the projection network's output, a contrastive loss is then calculated. Finally, these representations are used in training a linear classifier, with the condition that the representations themselves stay fixed, or "frozen," throughout this training stage.

Data Augmentation : The field of contrastive learning[4, 11] depends heavily on data augmentation[9], contributing significantly to the robustness and generalization capabilities of models. By applying diverse augmentations to the input data within a mini-batch, the model

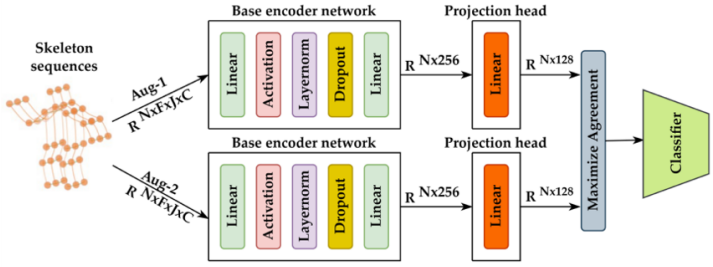

Figure 3.1: An overview of the ConMLP framework

is exposed to a richer set of variations, enhancing its ability to discern meaningful patterns and features. This augmented data serves as a more comprehensive training set, enabling the model to learn invariant representations across different transformations. Consequently, the contrastive learning framework benefits from this increased variability during training, enhancing model adaptability to changes in the input data that it might encounter in real-world scenarios. In essence, data augmentation acts as a catalyst, fostering the development of more resilient and versatile representations, ultimately improving the model's performance and transferability.

1. **Shear Transformation:** When dealing with skeleton data, the dynamics are distinct. In this scenario, each joint is projected in a predetermined direction, essentially dictating the projection of a skeleton frame into a specific viewpoint. To execute this transformation for skeleton data, a generic 3D shear transformation matrix proves instrumental. This transformation matrix is explicitly designed to induce shearing effects in sequences of skeleton data. Its definition is articulated as follows:

$$\text{Shear} = \begin{bmatrix} 1 & S_X^Y & S_X^Z \\ S_Y^X & 1 & S_Y^Z \\ S_Z^X & S_Z^Y & 1 \end{bmatrix}$$

2. **Reverse Transformation:** In the context of the dataset-NTU RGB+D with structures (N, F, J, C), where N is the size of the mini-batch, F represents the number of frames, J is the number of joints, and C is the 3D coordinates, the reversal of the temporal sequence of frames is referred to as "reverse.".

Base encoder network: The encoder network's primary function is to convert input into representative vectors, achieved by forwarding two sets of augmented input through the encoder network and generating a representation vector pair. A simple Multilayer Perceptron (MLP) serves as the base encoder in this architecture with 256 hidden layers. Following typical MLP design principles, the encoder involves linear layers, Gelu activation[7], layer normalization[8] (instead of using batch normalization for stability during training), and dropout that prevents overfitting.
According to [7], the GELU activation function is:

$$\text{GELU}(x) = \frac{x}{2}\left(1+\tanh\left(\sqrt{\frac{2}{\pi}}\left(2x+\frac{2}{\pi}x^3\right)\right)\right), \tag{3.1}$$

Projection head: Integrating a projection head serves the purpose of translating the base encoder network-generated representations, into a 128-dimensional latent space, which forms the basis for further classification efforts. Generally, it is possible to model the projection head as an MLP with just one hidden layer or as a single linear layer network. However, in this investigation, the default choice is a linear layer network for the projection head, emphasizing computational efficiency and showcasing the advantages of the proposed framework.

3.1.2 Contrastive Loss Function

The Normalized Temperature-scaled Cross-Entropy Loss (NT-Xent Loss)[20] serves as the basis of self-supervised learning within the ConMLP framework. Utilizing the contrastive loss function for self-supervised learning [4, 5], the index of one random augmented sample (the anchor) is $i \in \mathbb{I} \equiv \{1,2,...,2N\}$, and the 128-dimensional latent space derived from the augmented sample is z_i. The augmented samples x_i and $z_{j(i)}$ are samples produced from the i-indexed sample(the positive); the $2(N-1)$ remaining indices are referred to as the negative samples. The temperature parameter $\tau \in \mathbb{R}^+$ is represented by the symbol $\cdot$, which stands for the inner product.

$$\mathbb{L}_{\text{self}} = -\sum_{i\in\mathbb{I}} \log \frac{\exp(z_i \cdot z_{j(i)}/\tau)}{\sum_{a\in\mathbb{A}(i)} \exp(z_i \cdot z_a/\tau)} \tag{3.2}$$

Consider an arbitrary augmented sample, denoted by i, which serves as the anchor and belongs to the set $I \equiv \{1,2,...,2N\}$. Here, N represents the size of the mini-batch. The augmented sample $\bar{X}_l$ yields a 128-dimensional latent space, $z_l = \text{PROJ}(\text{ENC}(_l)) \in \mathbb{R}^{D_p}$, obtained through processing by the projection network, which was followed by the encoder network. The positive sample produced from the i-indexed sample is denoted by $z_{j(i)}$, whereas the remaining 2(N-1) indices are regarded as the negative samples. The $\bullet$ denotes the inner product. Introducing a temperature-related parameter $\tau \in \mathbb{R}^+$, and defining $A(I)$ as $I \setminus i$, representing the index values of all elements in set I excluding i.

Both the augmented sample and identically labeled samples inside a mini-batch are treated as positive samples[9] at the same time when using the supervised contrastive loss function[14]. Furthermore, the gradient computation built into SupCon has an inherent capacity to effectively mine both hard positives and negatives, ensuring optimal utilization of challenging samples.

$$\mathbb{L}_{\text{sup}} = \sum_{i\in\mathbb{I}} -\frac{1}{|\mathbb{P}(i)|} \sum_{p\in\mathbb{P}(i)} \log \frac{\exp(z_i \cdot z_p/\tau)}{\sum_{a\in\mathbb{A}(i)} \exp(z_i \cdot z_a/\tau)} \tag{3.3}$$

The supervised contrastive loss function[4, 12, 14] employed in this approach is a synthesis of the aforementioned contrastive losses, where the set of identifiers for all of the positive samples in $\mathbb{A}(i)$ except i, is represented by the expression $\mathbb{P}(i) \equiv \{p \in \mathbb{A}(i) : \tilde{y}_p = \tilde{y}_i\}$, and $|\mathbb{P}(i)|$ denotes the cardinality.

3.2 ConMLP- with Tuned Contrastive Learning

In this section, we experimented on how to enhance the ConMLP framework by incorporating tuned contrastive learning[15]. Instead of the usual supervised contrastive (SupCon) loss[4, 12],

we test the effectiveness of a tuned contrastive loss. This change aims to make the model better at recognizing detailed features from skeletal data, thus improving its performance in identifying different activities. By examining this adjustment within the ConMLP architecture, we aim to understand if using tuned contrastive learning can benefit skeleton-based activity recognition. Nevertheless, the parameter count remains identical to that of the base model, ConMLP.

3.2.1 Architecture

The architecture of our proposed approach closely mirrors that of the ConMLP model, with the primary distinction lying in the choice of loss function. Specifically, we have replaced the conventional supervised contrastive (SupCon) loss with a tuned contrastive loss[15] in our framework. This modification allows us to leverage the benefits of contrastive learning while fine-tuning the loss function to better suit the intricacies of our dataset and the task at hand. By optimizing the contrastive loss[4, 12] in a more tailored manner, we aim to expand the model's capacity for discrimination and improve its ability to capture subtle variations in skeletal data representations. Despite this change, the core architecture remains intact, featuring multi-layer perceptron (MLP) modules for feature extraction and a contrastive learning mechanism for training.

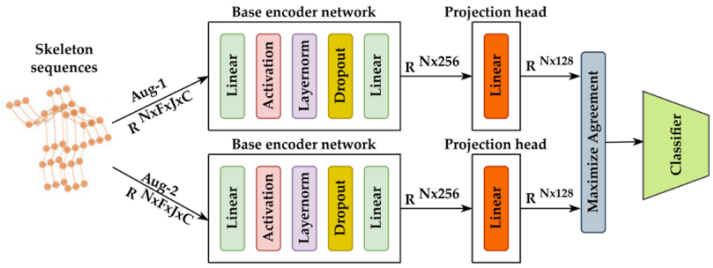

Figure 3.2: Tuned contrastive learning in supervised set-up

3.2.2 Tuned-Contrastive Loss

Tuned Contrastive Learning (TCL)[15] introduces a loss function that extends to multiple positives and negatives within a batch. It provides parameters for tuning, with the objective of improving gradient responses resulting from hard negatives and hard positives. The fundamental idea behind contrastive loss involves treating each transformed sample as a reference, known as an anchor. The network is then trained to push negatives away from it and bring positives closer to it in the latent space using this anchor.

TCL Loss is designed to handle a batch concurrently with multiple positives and negatives, applicable in both self-supervised and supervised scenarios[10]. It addresses limitations observed in SupCon loss, the absence of mechanisms to regulate hard negative gradient responses and specifically the implicit consideration of positives as negatives. As a result, TCL loss pro-

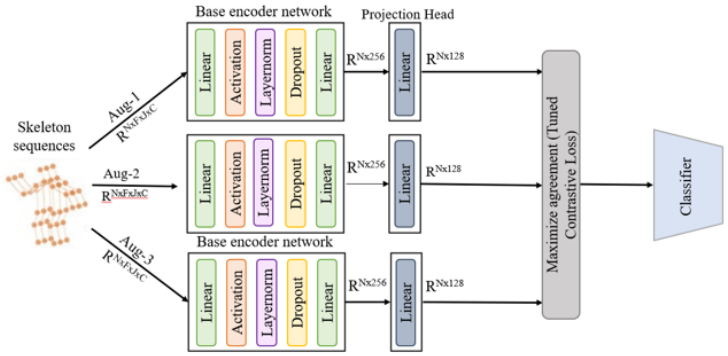

Figure 3.3: Tuned contrastive learning in self-supervised set-up

duces improved gradient responses for challenging positives and negatives, leading to consistent but modest enhancements in performance, with improvements over both cross-entropy loss and SupCon loss. According to[15], TCL is defined as:

$$L_i^{tcl} = \frac{-1}{|P(i)|} \sum_{p \in P(i)} log(\frac{exp(z_i.z_p/\tau)}{D(z_i)})$$

where

$$D(z_i) = \sum_{p' \in P(i)} exp(z_i.z'_p/\tau) + k_1(\sum_{p' \in P(i)} exp(-z_i.z'_p)) + k_2(\sum_{n \in N(i)} exp(z_i.z_n/\tau)) \tag{3.4}$$

$$k_1, k_2 \geq 1$$

Here, the scalar parameters k_1 and k_2 are pre-fixed before training. The batch of samples obtained after augmentation is shown by I, and it will be twice as large as the initial input batch. $i \in \{I\}$ indicates an anchor or sample inside it. The normalized projection network embedding for sample i given by the projector network is indicated by z_i. $P(i)$ is the collection of all positive samples for the anchor i (apart from the anchor i itself). This includes positives from the augmentation module and positives in batch I that have the same label as anchor i. The set of negatives in a batch denoted by $N(i)$ is such that $N(i) \equiv I \setminus (P(i) \cup \{i\})$.

The anchor z_i will pull the positive z_p but somewhat push the other positives out of the embedding space in order for the SupCon loss per sample $L_i(sup)$ to decrease. To minimize this impact and enhance performance, TCL loss introduces parameters.

It's important to note that introducing an additional term to enhance the gradient response from hard positives in the TCL loss is distinct from the approach of boosting the gradient response through a simple increase in the learning rate. The key distinction lies in the fact that TCL loss, under the same fixed learning rate, elevates the gradient signal's magnitude by modifying the coefficient of the positive term in the loss equation (equation 11). This adjustment not only increases the magnitude but also alters the gradient direction. Consequently, TCL loss consistently exhibits improved performance. The parameter k2 in TCL allows for the regulation, or increase, of the gradient signal from hard negatives. This addresses the second limitation of the SupCon loss, providing a direct solution to enhance the gradient response and contribute to improved overall performance.

3.3 graph2vec + LSTM Model

Skeleton based action recognition task using graph2vec[16] requires the entire skeleton graph to be the input to the model and we might want to train a deterministic model on top of the graph. This indicates that a fixed-length feature vector is how we wish to represent a graph. Which is where Graph2vec comes into play. It is the first work that talks about embedding the entire graph in an unsupervised manner and classifying the action using LSTM for skeleton-based activity recognition.

3.3.1 Architecture

In this model architecture, we begin by representing each skeleton graph using graph2vec, a method that generates embeddings for graphs represented as an edge list along with the features of each node in the graph. Each continuous skeleton graph, representing a sequence of frames, is transformed into a fixed-size embedding vector of 128 dimensions capturing its structural and relational information.

Next, we organize these graph embeddings into a sequence of inputs and feed them into an LSTM network [18] for action classification. RNNs such as LSTM are designed to handle sequential input by storing information for long periods of time. It processes the sequence of graph embeddings, one at a time, and maintains an internal state that captures the temporal dependencies between consecutive frames.

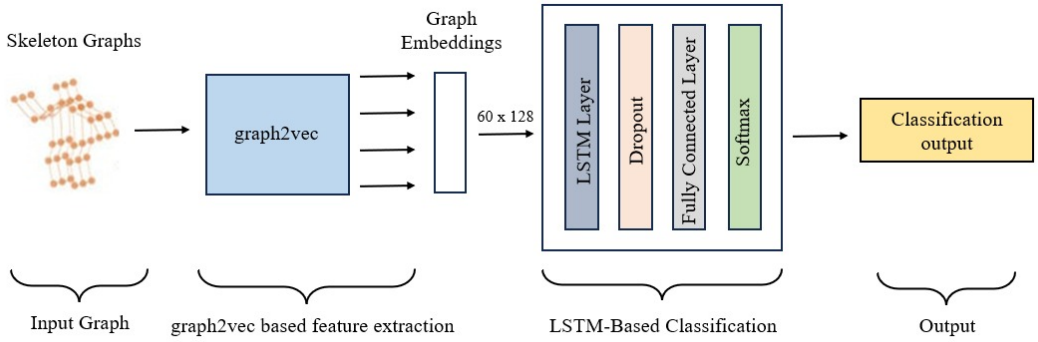

Figure 3.4: Graph2vec - LSTM model Architecture

In this specific architecture, we pass 60 continuous skeleton graph embeddings representing a sample, which collectively contain 300 frames, to the LSTM. As the LSTM iterates through the sequence of graph embeddings, it gradually learns to extract relevant temporal patterns and features that are indicative of different actions or activities. Finally, the LSTM's output flows through a classification layer, which typically consists of a softmax activation function is placed after fully connected layers. This layer maps the learned features from the LSTM to the corresponding action classes, producing a probability distribution over the possible actions. As the classification result, the action with the highest probability is then predicted for the given sequence of skeleton movements.

3.4 graph2Vec + GRU Model

Each skeleton sequence is converted into a graph representation, where joints are nodes and connections between joints are edges. This edge list information along with features is stored as json objects then given as input to graph2vec. In this modified architecture, we retain the core concept of using graph2vec[16] for graph embedding generation, but instead of employing an LSTM for action classification, we utilize a GRU model. GRU is a specific kind of RNN that works effectively with sequential data processing. It can capture long-range dependencies in temporal sequences because of gating mechanisms that let it selectively update and forget information with time.

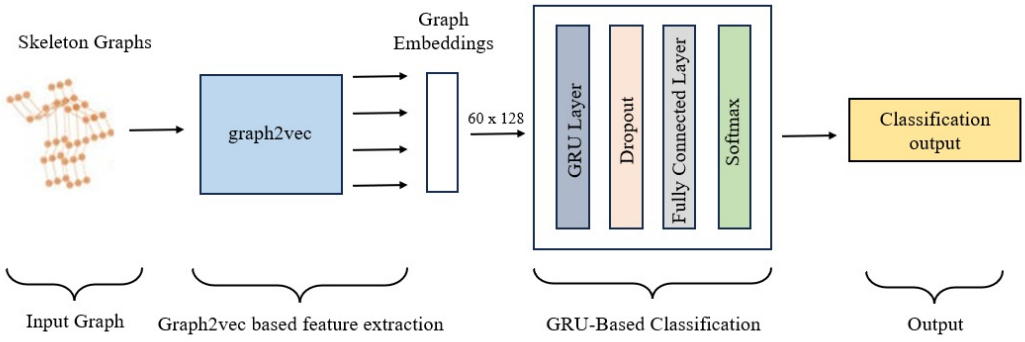

Figure 3.5: Graph2vec - GRU model Architecture

3.5 ConMLP-with added Reconstruction Loss

This section presents Contrastive Loss + Reconstruction Loss algorithm - - a learning algorithm that is self-supervised and simultaneously optimizes a contrastive loss and a self-reconstruction loss to produce representations. The variances inside a class may be larger than the variations across the classes in fine-grained classification tasks. In reality, it is possible to notice a high degree of similarity between categories, and modeling these minute differences requires a huge number of features. Therefore, these challenges create difficulty for models that rely on representations obtained from unsupervised pretraining. We discovered that this restriction is effectively addressed by including a self-reconstruction loss. Additionally, we introduce a novel approach that adds a self-reconstruction exercise to contrastive learning. Better results in skeleton-based action recognition are obtained with the suggested learning paradigm, indicating that combining a contrastive loss function with a self-reconstruction loss produces representations with a high enough granularity.

3.5.1 Architecture

This model comprises an encoder network $e(.)$, a projection head $p(.)$ for contrastive learning, and a decoder network $d(.)$ for reconstruction. For the encoder and decoder in this model, we used ConMLP as the primary architecture. In the process of training, the model receives two augmented inputs as in ConMLP and outputs the reconstructed image $x_i = d(e(\bar{x}_i))$ as well as the

contrastive vector representation z_i = p(e($\bar{x}_i$)). The contrastive loss L_{self} and the reconstruction loss L_r make up the two components of the training loss.

$$L_{total} = L_{self} + L_r \tag{3.5}$$

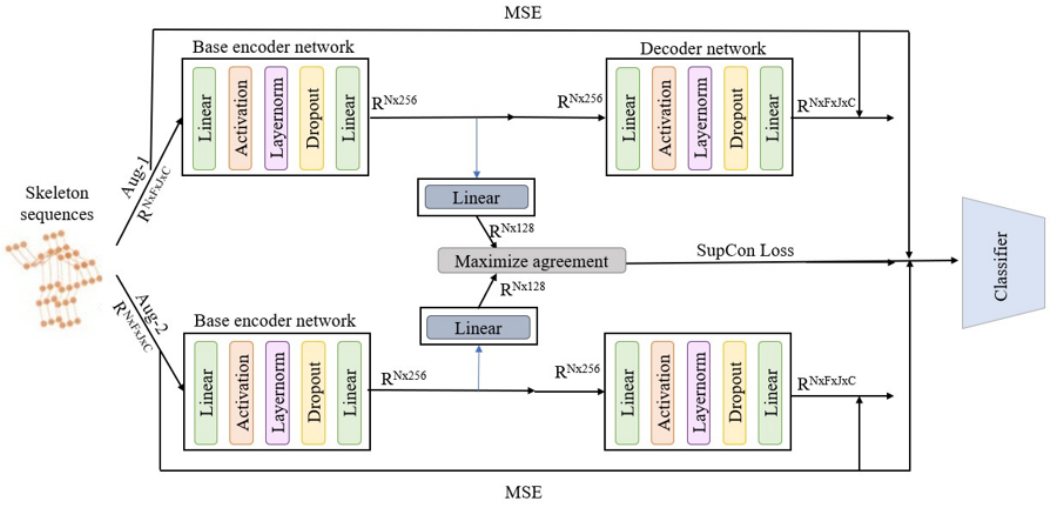

Figure 3.6: ConMLP-with added Reconstruction Loss Architecture

The element-wise MSE between the expected reconstruction and the input is used by reconstruction loss L_r to assess the fidelity of the reconstruction. As stated in eq: 3.2, we employ the NT-Xent loss as contrastive loss L_{self}.

$$L_r = \frac{1}{N}\sum_{i}^{N}(d(e(\bar{x}_i)) - x_i)^2 \tag{3.6}$$

Chapter 4

Experimentation

4.1 Setup:

Software and hardware specifications that are used in this project are briefly listed below:

1. Programming language used: Python3.
2. Framework for deep learning: PYTORCH.
3. We made use of CPU in the local system for training ConMLP-based models.
4. We made use of GPU P100 in Kaggle for training our custom Graph2vec models.

4.2 Dataset:

NTU RGB+D Datasets - This dataset contains 56,880 samples. These motion clips were concurrently recorded by three camera viewpoints for 60 classes. Cross-View(X-View) and Cross-Subject(X-Sub) are the two evaluation protocols used.

In this dataset, there are 37,646 training and 18,932 testing clips in X-View, and there are 40,091 training and 16,487 testing clips for X-Sub, respectively. 56,880 action samples make up this dataset. 60 action categories, completed by 40 subjects, are included in the "NTU RGB+D" dataset. The dataset contains 3 divisions of actions: 40 daily actions like drinking, and eating, nine health-related actions like sneezing, and 11 mutual interactions like hugging. These activities occur in 17 distinct scenario settings, which correlate to 17 film segments (S001–S017). The 25 main body joints' 3D positions are contained in 3D skeletal data for every frame. In the two datasets, every file or folder name is written in the pattern SsssCcccPpppRrrrAaaa (e.g., S001C002P003R002A013), here sss stands for setup number, ppp for the performer (subject) ID, ccc for camera ID,aaa for action class label, and rrr for replication number (1 or 2). Skeletal data is missing or incomplete for 302 samples in the "NTU RGB+D" dataset.

4.3 Data Preperocessing

Preparing the data is crucial for skeletal-based action recognition, especially nturgb+d dataset[11]. A training and a test set were created from the raw data used in this study. 37,646 training samples and 18,932 test samples for X-View were used.

According In the context of our project, the utilization of Graph2vec necessitates representing data as graphs, where each graph encapsulates the relationships between entities. Specifically, the skeleton data is structured into graphs, with the edge list encoding connections between nodes, and the coordinates of the nodes serving as their features. The edge list and the features were encoded as json objects in .json files corresponding to each skeleton frame(graph). Given that graph2vec operates on a per-graph basis, we've adopted a preprocessing step to refine our dataset. Specifically, we've excluded mutual actions from the dataset for graph2vec based models, ensuring that each graph represents a distinct scenario or interaction.

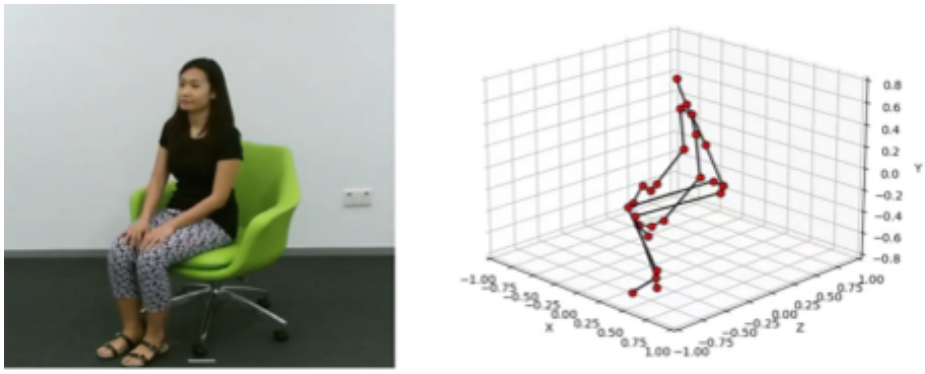

Figure 4.1: RGB Video and corresponding 3D skeleton data.

The official NTU dataset statement states that certain data in the dataset was incomplete and had to be removed. Seq_transformation.py, get_raw_denoised_data.py, and get_raw_skes_data.py are the three main preprocessing scripts utilized as mentioned in ConMLP. The get_raw_skes_data.py script retrieves essential skeleton data, while get_raw_denoised_data.py focuses on reducing noise and artifacts, ensuring cleaner input. Subsequently, seq_transformation.py performs sequence transformations, including resizing, to standardize the dataset. The coordinates are flattened and fed as input to ConMLP based models as NxFxJxC where J, C, F, and N stand for joints, frames in a skeleton sequence, mini-batch size, and 3D coordinates, respectively.

4.4 Default Metrics:

- **For graph2vec based models:** The models use the PyTorch framework. graph2vec was trained for 10 epochs, transforming the skeleton graphs into a 128-dimension embedding. With 0.025 as the learning rate and 0.0001 as the downsampling rate, graph2vec was optimized using SGD[13]. The LSTM and GRU classification layers are trained for 150 epochs with a randomly generated batch size of 256. The LSTM and GRU blocks were optimized by Adam optimizer with 0.001 as the learning rate. Dropout with a dropout rate of 0.2 is applied within the LSTM classification blocks. Cross-entropy loss is the classification loss used for this task, in the end.

- **For ConMLP based models:** The encoder network at the base used was MLP, which is SGD[13] optimized, using 0.001 as the learning rate and 0.0005 as weight decay. In

addition, the representations were converted to a 128-dimensional latent space using a linear project head. The temperature coefficient of 0.07 was used in the loss function. For training, a mini-batch of size 512 was created at random and kept for 5000 epochs. A cosine schedule was used to gradually reduce the learning rate without any restarts.

Chapter 5

Results and Discussion

In our evaluation, we assessed the four distinct models proposed above, namely ConMLP with tuned contrastive loss, ConMLP with added reconstruction loss, graph2vec-LSTM, and graph2vec-GRU, across NTURGB+D dataset[11]. We chose to explore graph2vec initially because we found some problems with basic ConMLP architecture. The basic ConMLP model did well in cross view, but in cross-subject, it didn't perform consistently. This inconsistency made us think about trying different approaches. Also, we considered how complex the ConMLP models were, especially in terms of how many parameters they had. So, we wanted to see if there were simpler ways to do things, which led us to try out graph2vec. graph2vec-LSTM and graph2vec-GRU produced the following results.

Model	**X-View(%)**
graph2vec + LSTM model	62.67
graph2vec + GRU model	49.96

Table 5.1: Top-1 accuracy scores - graph2vec based models in Supervised setup

However, the underperformance of the graph2vec-LSTM model can be attributed to the ineffective integration of task-agnostic graph representations with the LSTM network, limited generalization capabilities of the representations, and suboptimal fine-tuning due to the absence of end-to-end training. Addressing these issues through better integration strategies, end-to-end training, and fine-tuning procedures could potentially enhance the model's performance.

Due to the limitations observed with graph2vec, we decided to reconsider the strengths of ConMLP architecture, we conducted experiments with different loss functions to address the shortcomings encountered with graph2vec. W investigated the efficacy of tuned contrastive loss and supervised contrastive (SupCon) loss[4] added with reconstruction loss in enhancing the performance of ConMLP models.

Model	**X-View(%)**
ConMLP + Sup Con Loss	88.28
ConMLP + TCL Loss	**68.16**
ConMLP + Reconstruction Loss	**93.55**

Table 5.2: Top-1 accuracy scores - ConMLP based models in Supervised setup

Our exploration with ConMLP models revealed that incorporating reconstruction loss led to remarkable advancements, establishing a new state-of-the-art (SOTA) in graph representation learning. The ConMLP model augmented with reconstruction loss achieved an outstanding top-1 accuracy of 93.55%, showcasing its effectiveness in capturing fine-grained details and structural nuances in the data. This approach, combines contrastive learning with reconstruction techniques to improve representation quality. In contrast, experiments with tuned contrastive[15] loss didn't yield comparable results due to the challenge of selecting optimal hyperparameters. The sensitivity of tuned contrastive loss in hyperparameter selection hindered its effectiveness, resulting in suboptimal outcomes.

Model	**X-View(%)**
ConMLP + Sup Con Loss	90.40
ConMLP + TCL Loss	**62.75**
ConMLP + Reconstruction Loss	**92.18**

Table 5.3: Top-1 accuracy scores - ConMLP based models in Self - Supervised setup

Furthermore, the additional decoder in the ConMLP+ reconstruction loss architecture results in a two-fold increase in training time and model size (number of parameters) over the standard conMLP architecture.

Model	**Parameters(M)**
graph2vec+LSTM	76.2
graph2vec+GRU	74.8
ConMLP + Sup Con Loss	11.4
ConMLP + TCL Loss	11.4
ConMLP + Reconstruction Loss	23.01

Table 5.4: Parameters comparision

Our findings highlight the significance of reconstruction loss in enhancing the performance of ConMLP models, achieving state-of-the-art performance in skeleton-based activity recognition domain.

Chapter 6

Conclusion

In conclusion, our exploration of different methodologies highlights both challenges and chances for improvement. Firstly, in our experimentation with graph2vec+LSTM and graph2vec+GRU models, we encountered subpar performance due to limited integration of task-agnostic graph representations with LSTM and absence of end-to-end training. Improving integration, end-to-end training, and fine-tuning can enhance graph2vec based models performance.

While we aimed to enhance the ConMLP model with the Tuned Contrastive Learning (TCL) loss function, our findings suggest that this approach didn't lead to significant performance improvements. Experiments with tuned contrastive loss highlighted the challenges in hyperparameter selection, limiting its effectiveness.

We showcased that the basic ConMLP framework with supervised contrastive loss has drawbacks in capturing fine-grained features in its representations. We added a reconstruction task to the foundational ConMLP architecture in order to solve this problem. Adding the reconstruction loss method increases self-supervised learning performance significantly by 4% on the NTURGB+D dataset for skeleton-based activity recognition. Reconstruction task addition results in additional performance gain, reaching SOTA performance in skeleton-based activity recognition challenge.

Bibliography

[1] Ke, Q.; Bennamoun, M.; An, S.; Sohel, F.; Boussaid, F. A New Representation of Skeleton Sequences for 3d Action Recognition. In Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21– 26 July 2017.

[2] Yan, S.; Xiong, Y.; Lin, D. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. In Proceedings of the 32nd AAAI Conference on Artificial Intelligence, AAAI 2018, New Orleans, LA, USA, 2–7 February 2018

[3] Kolesnikov, A.; Zhai, X.; Beyer, L. Revisiting Self-Supervised Visual Representation Learning. In Proceedings of the 32nd IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, 15–20 June 2019.

[4] Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G. A Simple Framework for Contrastive Learning of Visual Representations. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, Virtual, 12–18 July 2020.

[5] Thoker, F.M.; Doughty, H.; Snoek, C.G.M. Skeleton-Contrastive 3d Action Representation Learning. In Proceedings of the 29th ACM International Conference on Multimedia, MM 2021, Virtual, 20–24 October 2021.

[6] Tolstikhin, I.; Houlsby, N.; Kolesnikov, A.; Beyer, L.; Zhai, X.; Unterthiner, T.; Yung, J.; Steiner, A.; Keysers, D.; Uszkoreit, J.; Lucic, M.; Dosovitskiy, A. Mlp-Mixer: An All-Mlp Architecture for Vision. In Proceedings of the 35th Conference on Neural Information Processing Systems, NeurIPS 2021, Virtual, 6–14 December 2021.

[7] Hendrycks, D.; Gimpel, K. Gaussian Error Linear Units (Gelus). arXiv 2016, arXiv:abs/1606.08415.

[8] . Lei Ba, J.; Ryan Kiros, J.; Geoffrey Hinton, E. Layer Normalization. arXiv 2016, arXiv:abs/1607.06450.

[9] Rao, H.; Xu, S.; Hu, X.; Cheng, J.; Hu, B. Augmented Skeleton Based Contrastive Action Learning with Momentum Lstm for Unsupervised Action Recognition. Inf. Sci. 2021, 569, 90–109.

[10] Budisteanu, E.A.; Mocanu, I.G. Combining Supervised and Unsupervised Learning Algorithms for Human Activity Recognition. Sensors 2021, 21, 630

[11] Shahroudy, A.; Liu, J.; Ng, T.T.; Wang, G. Ntu Rgb+D: A Large Scale Dataset for 3d Human Activity Analysis. In Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016.

[12] ng, F.; Liu, H. Understanding the Behaviour of Contrastive Loss. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2021, Nashville, TN, USA, 20–25 June 2021.

[13] oshchilov, I.; Hutter, F. Sgdr: Stochastic Gradient Descent with Warm Restarts. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017.

[14] Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C. and Krishnan, D., 2020. Supervised contrastive learning. Advances in neural information processing systems, 33, pp.18661-18673.

[15] Animesh, C., 2023. Tuned Contrastive Learning (Doctoral dissertation, University of California, San Diego).

[16] Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., Jaiswal, S. (2017, July 17). graph2vec: Learning Distributed Representations of Graphs. arXiv.org. https://arxiv.org/abs/1707.05005).

[17] Dai, C.; Wei, Y.; Xu, Z.; Chen, M.; Liu, Y.; Fan, J. ConMLP: MLP-Based Self-Supervised Contrastive Learning for Skeleton Data Analysis and Action Recognition. Sensors 2023, 23, 2452. https://doi.org/10.3390/s23052452.

[18] Staudemeyer, R. C., Morris, E. R. (2019, September 12). Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks. arXiv.org. https://arxiv.org/abs/1909.09586

[19] Zhang, H.-B.; Zhang, Y.-X.; Zhong, B.; Lei, Q.; Yang, L.; Du, J.-X.; Chen, D.-S. A Comprehensive Survey of Vision-Based Human Action Recognition Methods. Sensors 2019, 19, 1005. https://doi.org/10.3390/s19051005

[20] Papers with Code - NT-Xent Explained. (n.d.). https://paperswithcode.com/method/nt-xent

Acknowledgment

We owe our sincere gratitude to our project guide Dr. S. Nagesh Bhattu, Department of Computer Science, National Institute of Technology, Andhra Pradesh, who took a keen interest and guided us all along, till the completion of our project work by providing all the necessary information required for the design and development of this model and his timely reviews on project outcomes and suggesting improvements in further fine-tuning the model.

We are also thankful to Dr.Karthik Sheshadri and Dr.K.Himabindu who were part of our project review committee and have spent a significant amount of their valuable time understanding and reviewing our project outcomes and sharing their feedback.

Further, the success and outcome of this project required a lot of guidance and assistance from many people, and we are privileged to have got this all along with the completion of our project. Everything we have done is because of such guidance and help, and we will never hesitate to thank them.

We are grateful and lucky enough to receive consistent motivation, support, and guidance from all the staff of the Computer Science Department. They have helped us to complete our project work successfully. We would also like to sincerely thank all my friends for their timely support.