École Polytechnique Fédérale de Lausanne

# Semester project

## VITA lab EPFL

Killian Hinard

**Supervisor** : Mohamed Ossama Ahmed Abdelfattah

Autumn 2023

## 1 Introduction

Action recognition has been an important and popular field of study for deep learning models for the last years, with a lot of application such as : security and video surveillance, sport performance analysis, video games, human/computer interface and healthcare. Moreover we can observe that recently, self-supervised learning method for skeletal action recognition have demonstrated promising result with the rapid advance of contrastive learning and, since the first apparition of attention based architecture, transformers based network are showing more and more promising result compare to the previous technologies.

Knowing those facts, our goal through this project is to explore the impact and the performances of a specific contrastive learning framework, with extreme data augmentation when used to train a state of the art transformer based network for action recognition.

This is an interesting problem because nowadays, we can find an almost infinite amount of data on the internet, but the major part of those data is composed of unlabelled data. However, all the best current model use supervised learning and thus need labelled data. Self-supervised learning allows the model to learn with a way smaller amount of labelled data which can be really time consuming and expensive to obtain. The field of action recognition, which we're interested in, required a huge load of data and improving self-supervised learning method can help us achieve good performances without this need of tons of data.

## 2 Related Work

**AimCLR (Contrastive Learning from Extremely Augmented Skeleton Sequences for Self-supervised Action Recognition)[1] :** AimCLR is a method that implement a self-supervised learning framework for skeletal action recognition. They introduced a new pipeline to handle extreme data augmentations. The idea behind is that traditional data augmentation methods are carefully designed to not hurt the learning process. This limit the potential novel movement pattern that can emerge if you use more aggressive augmentations, but extreme augmentations induce dramatic changes in the movement pattern of a sequence an thus can hurt the learning process. With AimCLR they proposed a framework to solve those issues by using both a normally augmented sequence (same pipeline for this sequence as CrosSCLR[2]) and an extremely augmented sequence. They train a common query encoder with a combination of an InfoNCE loss between normally augmented query and key and a new loss called Dual Divergence Minimization loss that will handle the extremely augmented data (more details on the losses later in the report). On top of that they added an Energy-based Attention-guided Drop Module to challenge the network that will learn even when important features are dropped, which will lead to reduce over-fitting problem.

The strength of this method is that it bring a new way to do data augmentations. The extreme augmentations bring novel movement pattern and help the network learn with less data, and by introducing the $D^3M$, the extremely augmented sequence is handled properly without hurting the learning process.

The limitations are that in the original version of the paper, they use a GCN-based network as the query and key encoder, we might be able to achieve better performances with a transformer based encoder, which is more recent and shows better overall performances.

**MotionBERT (A Unified Perspective on Learning Human Motion Representations)[3] :** MotionBERT is a method that present a unified perspective to tackle different human-centric video task, by learning a representation from different sources. In the original paper it can do 3D pose estimation, Action Recognition and Mesh recovery. The method is based on a first phase of pretraining, in which we train the network to recover 3D skeletal joint motion from 2D partial observations and a second phase where we finetune the network on one of the different task it can achieve. They introduced an encoder for this network with the Dual-stream Spatio-temporal Transformer (DSTformer) which is a network composed of spatial attention layer and temporal attention layer, allowing the network to capture both the spatial connection between the joint and the temporal connection between the frame of the motion.

The strength of this model is that : it is showing impressive result for the different task it can tackle, which confirm the fact that the representation the network is learning is pretty robust. It brings also a way to learn from different sources for the pretraining, without the need of labelled data which is a huge advantage, especially for computer vision task. Finally, it gives us a way to do a lot of different human skeletal motion task by learning only one representation.

The limitations are that it is a massive network, with millions of parameters, we need a huge computing power to pretrain it and so using it can become both pretty expensive and time-consuming.

**ActCLR (Actionlet-Dependent Contrastive Learning for Unsupervised Skeleton-Based Action Recognition)[4] :** ActCLR is a method that propose another way to do data augmentation in the context of Action Recognition. It is pretty similar to AimCLR for the way it uses a GCN network as a backbone encoder but it differs in this way : now the data augmentation are applied differently to the part of the skeletal joints that are moving and to the part that stay static. This is done in order to introduce more diversity while maintaining their own characteristics. They introduce a feature pooling method to differentiate the static and mooving regions in order to apply different transformations.

The strength is that this way of doing the augmentation leads to better result than when we applied the same transformation everywhere : we have now a way to bring novel motion patterns but keeping the intrinsic characteristic of the movement.

The limitation is that the backbone is still a GCN as in AimCLR and better performances might be achievable with transformer based backbone.

## 3 Method Implemented

The method we will use is based on the two recent work MotionBERT and AimCLR. We wanted to study the impact of extreme data augmentation on transformer based architecture. As we saw before AimCLR implement an architecture showing really good performances that uses a GCN based backbone. Moreover MotionBERT is implementing an attention based network that shows really good result for action recognition when pretrained on skeletal joint. Thus, we will try to use AimCLR framework for data augmentation to pretrain MotionBERT.

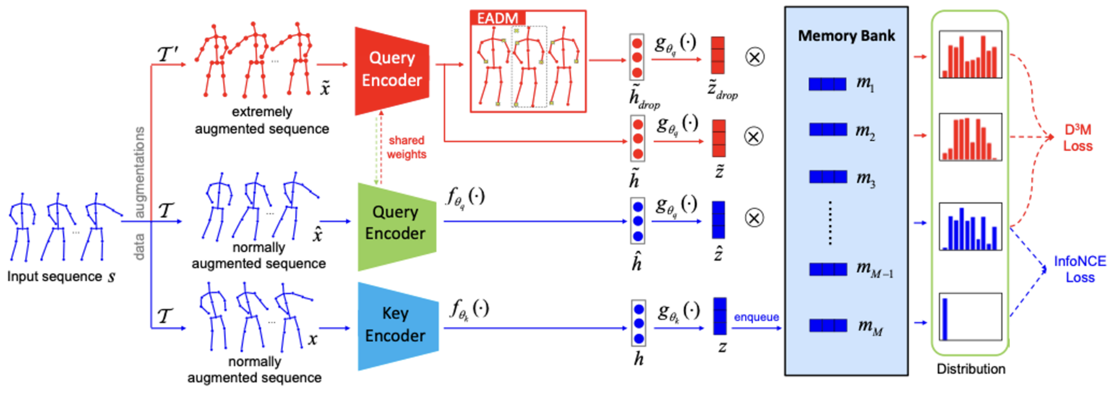

Figure 1: Pipeline of the proposed method, here the query encoder are the DSTFormer block from MotionBERT. (This figure is a slightly modified version of the one from AimCLR paper)

**Data Augmentation :** The pipeline of AimCLR introduces two different types of augmentations for the input sequence. First type, the **normal augmentations** $\mathcal{T}$ which apply one spatial augmentation *Shear* and one temporal augmentation *Crop*. The second type, **extreme augmentations** $\mathcal{T}'$ which applies 8 different augmentations to the data : 4 spatial augmentations *Shear, Spatial Flip, Rotate* and *Axis Mask*, 2 temporal augmentations *Crop* and *Temporal Flip* and 2 spatio-temporal augmentations *Gaussian Noise* and *Gaussian Blur*. The goal of the extreme augmentations is to bring to the model more novel movement patterns than the classic normal augmentations.

**Query and Key Encoder :** The query and key encoder we use in our architecture are the DSTformer from MotionBERT. It is a transformer based network composed of an alternance of subblock each composed of spatial and temporal attention layers (see the figure)

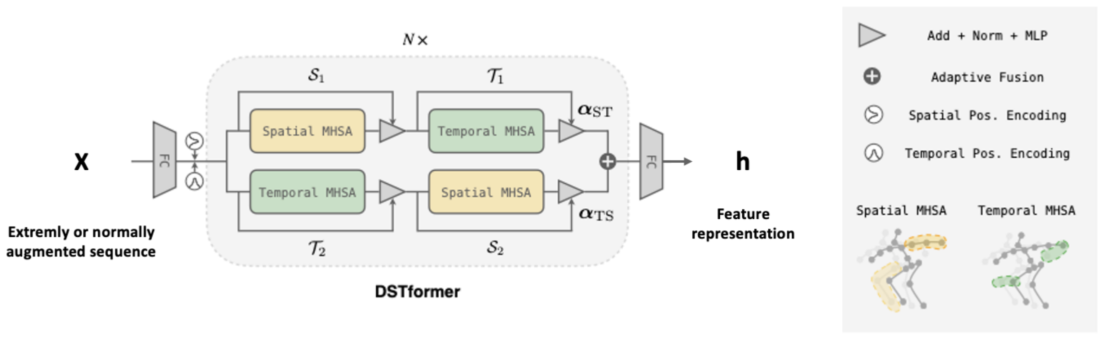

Figure 2: DSTformer architecture. (This figure is a slightly modified version of the one from MotionBERT paper

We use the same encoder to get the representation of the normally augmented sequence and the extremely augmented sequence. The weight for the key encoder are computed as a moving average of the query encoder coefficient such as :

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q$$

with m being the momentum coefficient. This way of updating the key encoder coefficient is meant to ensure stable key representations.

**Losses :** To pretrain this network, we are using two different losses : the InfoNCE loss and the Dual Distributional Divergence Minimization ($D^3M$) loss. First, the InfoNCE loss, which is a common loss in contrastive learning framework, is used between the normally augmented query and key. It is computed as follow :

$$\mathcal{L}_{Info} = -q(z|\hat{z})\log p(z|\hat{z}) - \sum_{i=1}^{M} q(m_i|\hat{z})\log p(m_i|\hat{z})$$

where $p(x|\hat{z})$ represent the likelihood of the query $\hat{z}$ being assigned to x, it is the distribution learned by the network and is computed as follow :

$$p(x|\hat{z}) = \frac{\exp(\hat{z}\cdot x/\tau)}{\exp(\hat{z}\cdot z/\tau) + \sum_{i=1}^{M}\exp(\hat{z}\cdot m_i/\tau)}$$

M is the memory bank containing all the $m_i$ key from the previous sample considered as negative. Finally $q(x|\hat{z})$ represent the ideal distribution of the likelihood, but to avoid the problem of having unknown ideal distribution, the InfoNCE considered it as one-hot distribution such that : $q(z|\hat{z}) = 1$ for positives pair and $q(m_i|\hat{z}) = 0$ for negative pair.

For the extremely augemented sequence, we can't just do InfoNCE loss between the extremely augmented sequence as a query and the normally augmented sequence as a key as we might have the idea first since the the extremely augmented sequence may not keep the identity of the original sequence. For that reason we use the $D^3M$ loss introduce in the AimCLR method, which is basically an InfoNCE loss, but considering the distribution of the normally augmented sequence to approximate the ideal likelihood distribution $q(x|\hat{z})$. We can compute it as follow :

$$\mathcal{L}_{\tilde{z}} = -p(z|\hat{z}) \log p(z|\tilde{z}) - \sum_{i=1}^{M} p(m_i|\hat{z}) \log p(m_i|\tilde{z}).$$

We compute it the same way for $\tilde{z}_{drop}$ which gives us $\mathcal{L}_{\tilde{z}_{drop}}$ and then we have that the total $D^3M$ is given by :

$$\mathcal{L}_{D^3M} = \frac{1}{2}(\mathcal{L}_{\tilde{z}} + \mathcal{L}_{\tilde{z}_{drop}})$$

## 4 Experiment

### 4.1 Dataset

**NTU RGB+D 60 Dataset :** This is a dataset composed 56 880 video samples from people doing an action from 60 different action classes. It contains 3D skeletal joint, and each skeletal graph contains V = 25 joints. There are two evaluation protocols recommended : xsub (Cross-subject) where training set and testing set are sequences from different subject and xview (Cross-view) where training and testing data are from different camera views.

**NTU RGB+D 120 Dataset :** An extended version of the NTU RGB+D 60 dataset with 57 600 sequences added to the 56 880 original ones, moreover we have now 120 action classes. There are also two different evaluation protocols recommended : xsub (Cross-subject) same principle as before and xset (Cross-setup) the training and testing data are from different setup IDs.

### 4.2 Experimental Settings

All the experiment were conducted with the Pytorch framework. The lentgh of each video sequence is reduced to 50 frame, as it was the case in AimCLR.

**Self-supervised Pretraining :** To pretrain the different network we will evaluate, we use the AimCLR framework with one variation : first we will use the full framework, with the non-mention above Nearest Neighbour Mining Loss, which is a loss they introduced to gather more positive sample by assuming that the samples in the memory bank are not to be considered all as negative samples compared to the current samples you are looking. To solve this issue, they look with a nearest neighbour mining algorithm the k closest sample from the memory bank to the query and they consider them as positive samples in the loss computation formula. In a second time, we will not use anymore the NNM loss, after looking at its effect.

**Caveat** : Each network was pretrained on a different number of epoch unfortunately because of missing time : the MotionBERT based network are taking a real long time to train. It will be precise for each experiment result the number of epochs.

**Linear-evaluation :** To asses the performances of our network we used to different method : first, as for AimCLR, a classic linear evaluation where we take the pretrained weight and we train only a fully-connected layer a the end of the pipeline that would predict the action class from the representation learned during the pretraining. We also applied the same strategy but using the action-head from MotionBERT instead of just a fully-connected layer at the end in order to predict the action class. For the two protocol we train the final classifier for 100 epoch with a decrease in the learning rate at epoch 80.

## 4.3 Results :

All the first experiment are run on the NTU-RGB+D 60 dataset with the xsub evaluation protocol.

**AimCLR framework with Vanilla transformer encoder with NNM loss:** First, before using a really complexe netwrok architecture such as MotionBERT for the encoder, we wanted to see the performances for a simple vanilla transformer. The implementation we used for the code is the one from "Masked Motion Predictors are Strong 3D Action Representation Learners"[5], that has a pytorch implementation.

For this experiment we used first the full AimCLR framework with the NNM loss that start at epoch 150. For the model hyperparameters, we took the one from the original implementation as well as the optimizer parameter and learning rate. This experiment was conducted on NTU-RGB+D 60 xsub dataset.

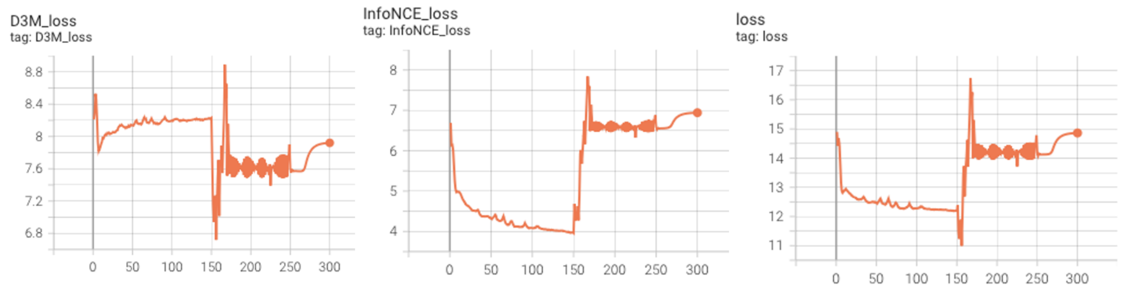

Figure 3: The InfoNCE, D3M and total loss during pretraining for vanilla transformer with Nearest Neighbour Mining

As we can see on the above figure, the network seems to learn well until the start of the nearest neighbour mining at epoch 150. From there it is showing a oscillatory behaviour without any improve in the training loss. This suggest that the nearest neighbour is not appropriated when using a transformer encoder, or at least the way we compute it right now hurt the learning process. This is confirmed by the fact that after the pretraining, the learned model does really bad in the linear evaluation as it achieves an accuracy of : 1.67% which correspond for 60 classes to a random guess. To dig more in this problem we made a loop evaluation protocol that will run the linear evaluation protocol for every 5 epoch of pretraining.

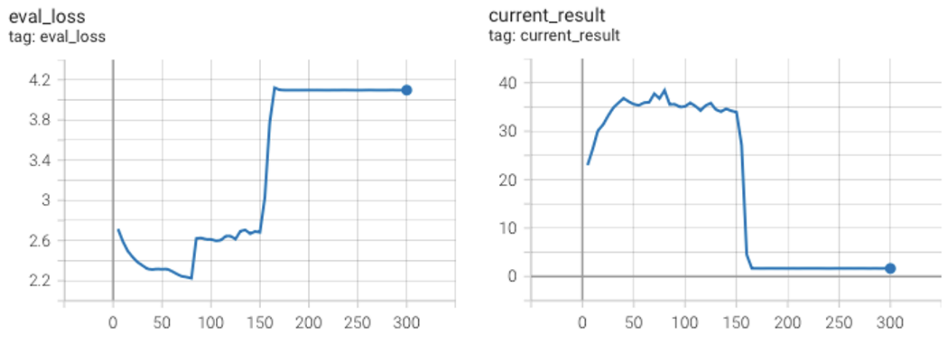

Figure 4: The CrossEntropy evaluation Loss (left) and the evaluation accuracy (right) through the pretraining process for vannilla transformer backbone

This clearly confirm that we have a problem with the NNM. To be sure that it is due to the transformer encoder we try to run the same loop evaluation for AimCLR classic network to see if we observe the same kind of behavior with the ST-GCN encoder :

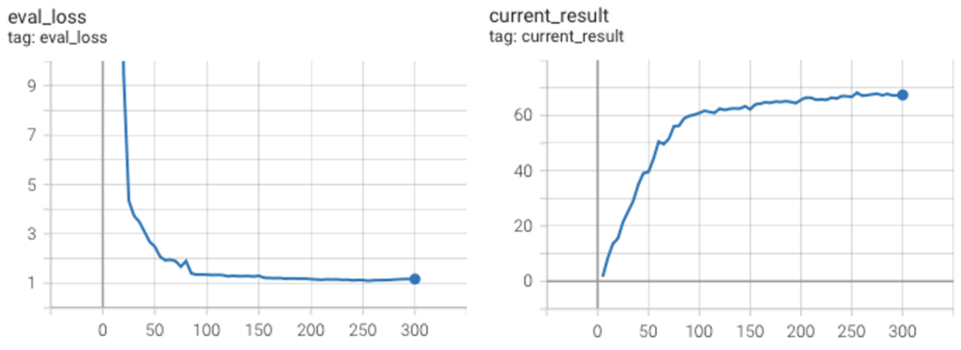

Figure 5: The CrossEntropy evaluation Loss (left) and the evaluation accuracy (right) through the pretraining process for ST-GCN backbone

we can clearly see that in the case of the st-gcn network, we don't have any drop in the prediction quality when the NNM start (epoch 150). Knowing that, and by acknowledging the fact that in the result of AimCLR paper the NNM does not have a huge impact, we decide to get rid of it to improve the quality of our model.

**AimCLR framework with Vanilla transformer encoder:** We train this time the network with the vanilla transformer but without the NNM. We pretrain it for 500 epochs and train the head for 100 epochs with the same learning rate and optimizer parameter as previously and this are the results :

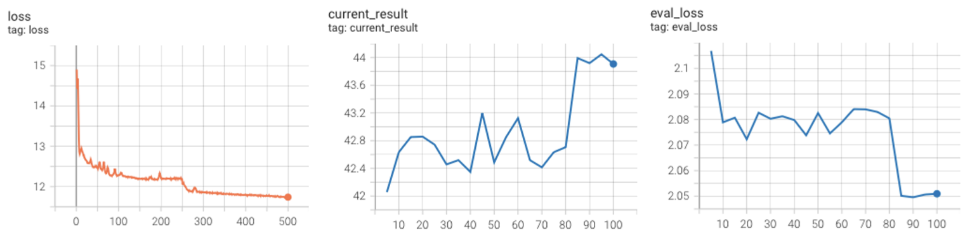

Figure 6: The pretraining loss evolution (left and orange), the linear evaluation (blue) evolution of the test loss and accuracy

As expected, without the NNM we have much better result and we achieve a top accuracy of 44.05%. These result are encouraging and we will next try to improve those result by using the more complex model MotionBERT as encoder.

**MotionBERT as the encoder with the same classifying head as AimCLR :** For this experiment, we replace the ST-GCN encoder from AimCLR by the DSTformer from MotionBERT. We use the same model hyperparameters as in MotionBERT, but with 4 as block depth instead of 5, because one depth more increase a lot the number of parameters and in the original MotionBERT paper, we can see that it doesn't lead to a huge improvement. The optimizer is still SGD with the same parameters as in MotionBERT. For this experiment we had to decrease a lot the batch size to not overload the GPU and thus it takes a lot of time to train. For this reason we only pretrained for 150 epochs instead of 300. Here are the results (We can also see on this graph that after the epoch 150, when the NNM starts the result are getting worst so it shows the same problem for an other transformer base architecture) :

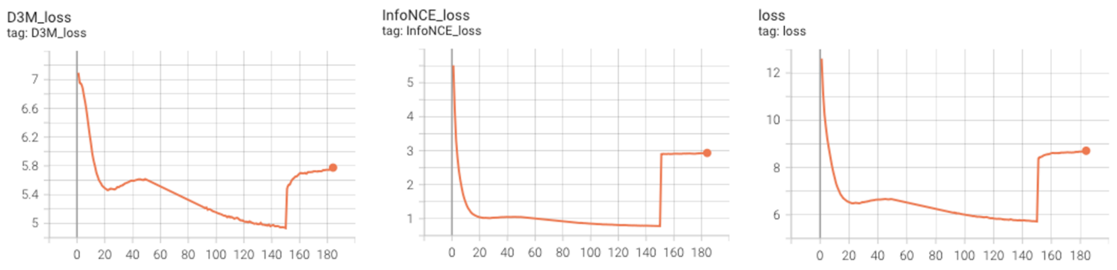

Figure 7: The loss evolution of MBAimCLR during the pretraining

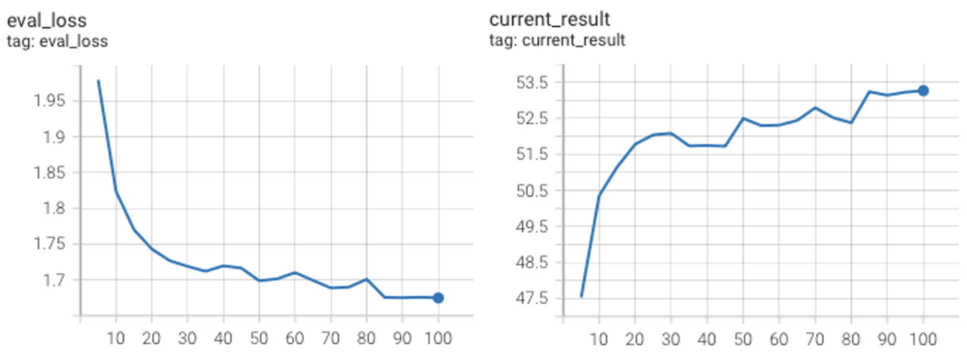

Figure 8: Evolution of the evaluation loss and the accuracy during the linear evaluation

We observe that we have better result with MotionBERT and we achieve a top accuracy of 53.37%.

**MotionBERT as the encoder with the action-head from MotionBERT :** In order to improve the result with MotionBERT as the encoder, we now try to do the linear evaluation with the action-head from MotionBERT which is mad to work with the representation learned with MotionBERT and which is slightly more complexe than the simple fully connected head from AimCLR. We use the same pretrain model as before. The results are :

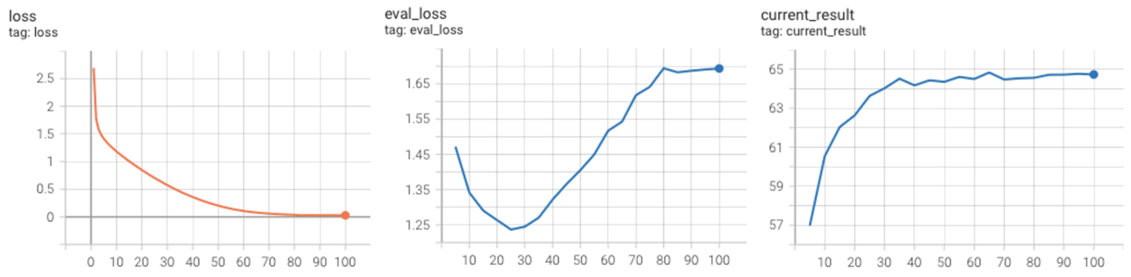

Figure 9: Evolution of the training loss (orange), the evaluation loss (blue) and the accuracy during the linear evaluation with the action head

First, we can observe that we achieve better performances than with the classic classification head : the top accuracy is now 64.83%. Moreover we can also see that the model start over-fitting the training set, we have almost zero loss at the end, and thus that the result on the test set can't improve since the model don't learn anything new after a certain amount of epoch. This would suggest that we need either more data or more generalization mechanisms.

**Other datasets :** Now that we know what seems to be our best model, MotionBERT as the encoder and action-head for the evaluation without NNM, we will pretrain it and evaluate it on the others dataset : NTU RGB+D 60 xview and NTU RGB+D 120 xset and xsub. All the results are summarized in this table :

| **Method** | NTU-60(%) | | NTU-120(%) | |
|---|---|---|---|---|
| | xsub | xview | xsub | xset |
| **TransAimCLR** | 44.05 (500e) | - | - | - |
| **MBAimCLR** | 64.83 (150e) | 75.85 (50e) | 59.52 (15e) | 59.34 (15e) |
| **AimCLR** | 74.34 (300e) | 79.68 (300e) | 63.4 (300e) | 63.4 (300e) |

Table 1: Performance comparison of the different methods on the NTU-60 and NTU-120 datasets. The number in parenthesis are the number of epoch the model was pretrained on

All the result are top-1 accuracies. As we can see, our model never beat the AimCLR model. This need to be still mitigated by the fact that we trained our model for way less epoch, especially for the NTU-120 dataset since it was really big, but even with only few pretraining epochs we achieve similar accuracies than what AimCLR achieves on those dataset. We could then expect that with more pretraining epochs, we could observe better results for those datasets.

## References

[1] Tianyu Guo, Hong Liu, Zhan Chen, Mengyuan Liu, Tao Wang, and Runwei Ding. Contrastive learning from extremely augmented skeleton sequences for self-supervised action recognition, 2021.

[2] Mohammadreza Zolfaghari, Yi Zhu, Peter Gehler, and Thomas Brox. Crossclr: Cross-modal contrastive learning for multi-modal video representations, 2021.

[3] Wentao Zhu, Xiaoxuan Ma, Zhaoyang Liu, Libin Liu, Wayne Wu, and Yizhou Wang. Motionbert: A unified perspective on learning human motion representations, 2023.

[4] Lilang Lin, Jiahang Zhang, and Jiaying Liu. Actionlet-dependent contrastive learning for unsupervised skeleton-based action recognition, 2023.

[5] Yunyao Mao, Jiajun Deng, Wengang Zhou, Yao Fang, Wanli Ouyang, and Houqiang Li. Masked motion predictors are strong 3d action representation learners, 2023.