Actional-Structural Graph Convolution Networks Applied to Martial Arts, Dancing and Sports Datasets for Action Recognition

By Adam Russell 12711142
Supervisor: Cen Wan

Contents

1. Abstract

This project applies Actional-Structural Graph Convolution Networks (AS-GCN) to Martial Arts, Dancing and Sports (MADS) Datasets for action recognition. Initially the model was deployed on an AWS cloud deep learning machine image (DLMI) but this was found to be unfeasible for long term testing. The AS-GCN pipeline (Li, et al., 2019) was deployed on a local hardware set-up and trained on a reduced size NTU RGB+D 120 (Liu, et al., n.d.) data set referenced in the paper. The accuracy was found to be comparable with the results listed in the AS-GCN paper. The MADS dataset was pre-processed and used to train a new AS-GCN model which classifies a sequence as Tai-chai, Karate, Jazz style dance or Hip-hop style dance. The cross-view accuracy was found to be comparable to the results listed in the AS-GCN paper, but the cross-subject accuracy was significantly lower. The AS-GCN code base had many errors and was absent of some the main features discussed in the main paper which meant some of the targets set out in the proposal were not achievable. In lieu of this further development work for this project are outlined including the incorporation of one-shot and transfer learning techniques.

All Pytorch output files and batch files are included in the project repository:
https://github.com/Birkbeck/msc-data-science-project-2020_21---files-arusse13

2. Introduction

2.1. What constitutes an action?

Action recognition seeks to classify sequences of human body part motions. What constitutes as an action is very broad even with this posterior statement. In this proposal we regard an action as all the four subcategories of physical human activity as defined below (Jegham, et al., 2020):

'Gesture: It is a visible bodily action representing a specific message. It is not a matter of verbal or vocal communication but rather a movement made with the hands, face or other parts of the body such as Okay gestures and thumbs up.

Action: It is a set of physical movements conducted by only one person like walking and running.

Interactions: It is a set of actions executed by at most two actors. At least one subject is a person and the other one can be a human or an object (hand shaking, chatting, etc).

Group activities: It is a mixture of gestures, actions, or interactions. The number of performers is at least two plus one or more interactive objects (playing volleyball, obstacle racing, etc).'

2.2. Why work with Sequences?

Of course, many actions can be recognized from a single frame using now well-established deep learning convolution neural networks (CNN). These will often detect subtle features of the image around humans such as objects and environment (Konushin, 2017). For example, a model may classify an image of a human playing football by detecting the features of the football on a green field. Further many actions share similar poses as part of the sequence. For instance, an image of footballer with the ball at their foot may be of them receiving a pass or attempting a pass. Therefor we further elaborate that action recognition in the context of this proposal is purely based on human pose sequences. All environmental and objects are to be filtered out. Processing sequences of frames in theory mitigate the ambiguity of processing single static frames.

2.3. Challenges working with Sequences

Working with sequences of frames directly is unfeasible as the size of the inputs to the neural network increases rapidly with the number of frames in a sequence (Konushin, 2017). Various filtering techniques have been developed to reduce the input size such as foveated architectures, optical flow and dense trajectories. This project will implement structural graphs whereby a graph with edges for bones and vertices for joints is extracted for each human in each frame for example Figure 1. The resulting lower resolution, sparse space time volumes are significantly more efficient to compute and store.

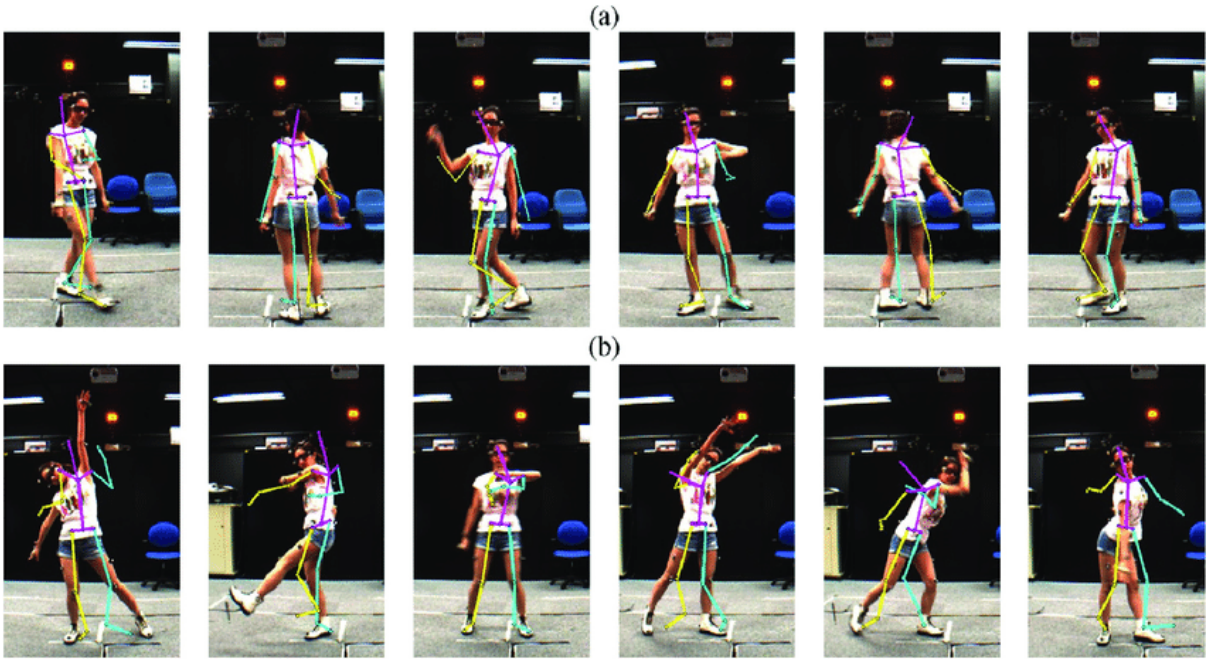


Figure 1 Example of skeletal graph extraction (Zhang, et al., 2017)

2.4. Applications of Action Recognition

Human action recognition is an active area of computer vision research for both real time and recorded data applications (Ni, n.d.). Deploying action recognition models to existing CCTV infrastructures have high value for all emergency services. Police, military and security organisations seek automated detection of crime and terrorism events. Fire services seek to minimise response times to road traffic accidents. Automated recognition of trauma/causality type (for example stroke/heart attack) would improve triage and save valuable time in deploying the correct treatment. Automated Factory's and warehouses seek to monitor the safety of human employees from the hazardous of collaborative robots (cobots). Self-driving car manufacturers would see large gains in pedestrian safety if they implemented action recognition models that detect if a pedestrian was about to step out into the street. Film, television, sports and media seek to use this technology to automatically time stamps actions of interest in sequence such as headers in a game of football or handshakes at a political conference for a news video archive indexing and retrieval.

3. Literature Review

3.1. Actional-Structural Graph Convolution Network (AS-GCN)

(Li, et al., 2019) propose that actions depend on richer joint dependencies than fixed skeletal graphs portray between joints that are far apart. Even basic actions such as walking or sitting mobilize many far apart joints and appendages in complex ways. They propose these Actional Links (A-Links) supplement traditional structural links of a fixed skeletal graph as shown in Figure 2. To generate these A-links the first stage of the Actional-Structural Graph Convolution Network (AS-GCN) proposed by (Li, et al., 2019) is the A-link inference module (AIM) consisting of an encoder and decoder. Skeletal space time volumes are input. The encoder 'exacts link features from 3D joint position and then converts the link features to the linking probabilities'. As information is propagated between joints and links the module learns link features. Concurrently Structural-links

(S-Links) are extracted at high order polynomials than the original to increase the set of 'local' neighbours of each joint/link.

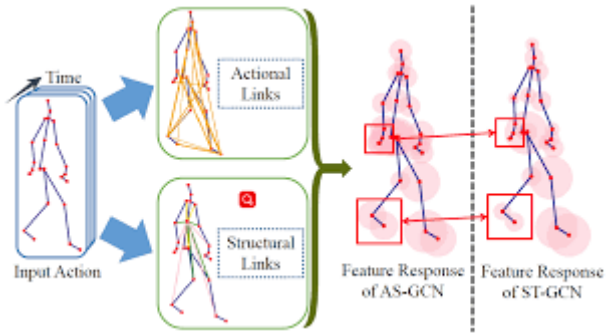


Figure 2 Graph Development (Li, et al., 2019)

The Actional-Structural Graph Convolution Block combine the actional graph convolution (AGC) and the Structural Graph Convolution (SGC) for each time stamp. A hyper parameter on the AGC trades off the importance between structural features and actional features. There are nine blocks of AS-GCN followed by one layer of Temporal Convolution to capture inter-frame action features along the time axis. This is the backbone of the network.

The recognition head of the network classifies actions using global averaging pooling and a SoftMax classifier with a standard cross entropy loss function. The Prediction head features the back-bone network in reverse with a standard l2 loss function for prediction. The entire pipeline is shown is shown in Figure 3.

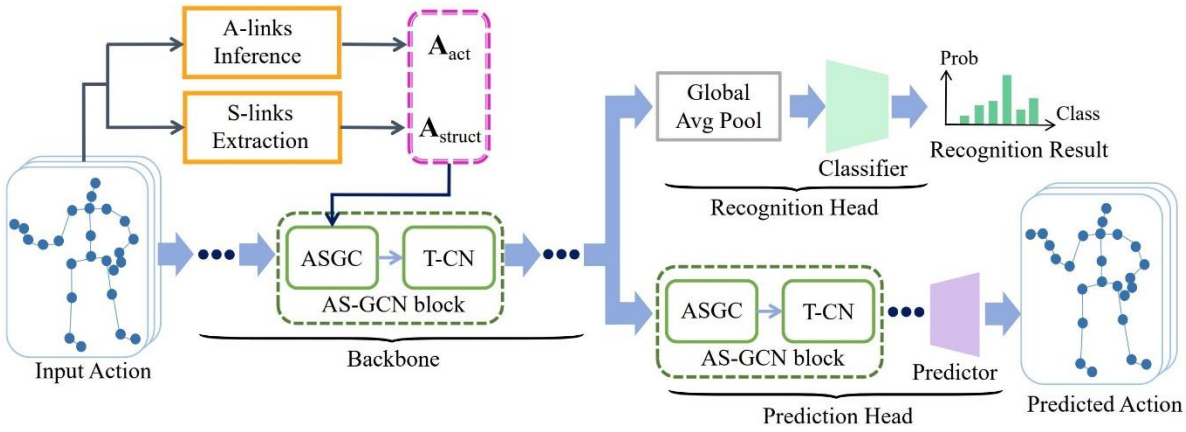


Figure 3 AS-GCN Pipeline (Li, et al., 2019)

3.2. NTU RGB+D 120 Data Set

The NTU RGB+D 120 (Liu, et al., 2019) data set features 114480 samples and 120 action classes sub categorised into 82 Daily Actions Table 1, 12 Medical Conditions Table 2 and 26 Mutual Actions/Two Person interactions Table 3. This dataset has been captured using three Kinect V2 cameras concurrently. The three kinetic v2 cameras were all set at the same height at -45, 0, +45 horizontal angles with the actors asked to perform each action towards both the left and right cameras resulting in 6 perspectives per action per set up. The samples are available in a variety of formats (RGB video, IR, Full depth maps etc.) but on 3D skeletons (body joints) will be required for training.

A1: drink water	A2: eat meal	A3: brush teeth	A4: brush hair
A5: drop	A6: pick up	A7: throw	A8: sit down
A9: stand up	A10: clapping	A11: reading	A12: writing
A13: tear up paper	A14: put on jacket	A15: take off jacket	A16: put on a shoe
A17: take off a shoe	A18: put on glasses	A19: take off glasses	A20: put on a hat/cap
A21: take off a hat/cap	A22: cheer up	A23: hand waving	A24: kicking something
A25: reach into pocket	A26: hopping	A27: jump up	A28: phone call
A29: play with phone/tablet	A30: type on a keyboard	A31: point to something	A32: taking a selfie
A33: check time (from watch)	A34: rub two hands	A35: nod head/bow	A36: shake head
A37: wipe face	A38: salute	A39: put palms together	A40: cross hands in front
A61: put on headphone	A62: take off headphone	A63: shoot at basket	A64: bounce ball
A65: tennis bat swing	A66: juggle table tennis ball	A67: hush	A68: flick hair
A69: thumb up	A70: thumb down	A71: make OK sign	A72: make victory sign
A73: staple book	A74: counting money	A75: cutting nails	A76: cutting paper
A77: snap fingers	A78: open bottle	A79: sniff/smell	A80: squat down
A81: toss a coin	A82: fold paper	A83: ball up paper	A84: play magic cube
A85: apply cream on face	A86: apply cream on hand	A87: put on bag	A88: take off bag
A89: put object into bag	A90: take object out of bag	A91: open a box	A92: move heavy objects
A93: shake fist	A94: throw up cap/hat	A95: capitulate	A96: cross arms
A97: arm circles	A98: arm swings	A99: run on the spot	A100: butt kicks
A101: cross toe touch	A102: side kick	-	-

Table 1 Daily Actions (82) (Liu, et al., n.d.)

A41: sneeze/cough	A42: staggering	A43: falling down	A44: headache
A45: chest pain	A46: back pain	A47: neck pain	A48: nausea/vomiting
A49: fan self	A103: yawn	A104: stretch oneself	A105: blow nose

Table 2 Medical Conditions (12) (Liu, et al., n.d.)

A50: punch/slap	A51: kicking	A52: pushing	A53: pat on back
A54: point finger	A55: hugging	A56: giving object	A57: touch pocket
A58: shaking hands	A59: walking towards	A60: walking apart	A106: hit with object
A107: wield knife	A108: knock over	A109: grab stuff	A110: shoot with gun
A111: step on foot	A112: high-five	A113: cheers and drink	A114: carry object
A115: take a photo	A116: follow	A117: whisper	A118: exchange things
A119: support somebody	A120: rock-paper-scissors	-	-

Table 3Mutual Actions / Two Person Interactions (26) (Liu, et al., n.d.)

3.3. Martial Arts, Dancing and Sports (MADS)

The MADS contain sequences of Tai-chai, Karate, Jazz style dance, Hip-hop style dance and other sports such as football, tennis and basketball performed by a single actor at a time. It features >53000 video frames and 30 action classes sub categorised into Tai-chi, Karate, Jazz, Hip-Hop and sports each with 6 actions. The video was shot in a studio environment using point grey bumble-II cameras. Each sequence is 60 or 80 seconds long with frame rates of 15 or 10 or 20.

Action category	No. Action Sequence
Tai-Chi	1. Commencing Form, Buddha's Warrior Attendant Pounds Mortar (I) (Qishi,Jin Gang Dao Dui (1))
	2. Single Whip, Lazy about Tying Coat (Dan bian Xia Shi,Lan Zha Yi)
	3. White Crane Spreads its Wings, Buddha's Warrior Attendant Pounds Mortar (II) (Bai He Liang Chi,Jin Gang Dao Dui (2))
	4. Walk Obliquely, Twist Step (I) (Xie Xing Ao Bu (1))
	5. Walk Obliquely, "Twist Step (II) (Xie Xing Ao Bu (2))
	6. Fist of Covering Hand and Arm, Crossing Hand (Yan Shou Gong Quan,Shi Zi Shou)
Karate	1. Second Basic Form (Fukyugata Ni)
	2. Third Basic Form (Fukyugata Sandan)
	3. Third Peace Form (Pinan Sandan)
	4. Fifth Peace Form (Pinan Godan)
	5. Straddle Stance (Naihanchi Nidan)
	6. Horse Riding Stance (Naihanchi Nidan Sandan)
Jazz	1. Jazz action1
	2. Jazz action2
	3. Jazz action3
	4. Jazz action4
	5. Jazz action5
	6. Jazz action6
HipHop	1. Hip-hop action1
	2. Hip-hop action2
	3. Hip-hop action3
	4. Hip-hop action4
	5. Hip-hop action5
	6. Hip-hop action2
Sports	1. Badminton
	2. Basketball
	3. Football
	4. Rugby
	5. Tennis
	6. Volleyball

Table 4 Action sequences in MADS (Zhang, et al., 2017)

4. Project Description, Aims and Objectives

Initially the AS-GCN pipeline proposed by (Li, et al., 2019) will be deployed on a cloud deep learning machine image (DLMI) and trained on the NTU RGB+D 120 data set referenced in the paper. The computation time shall be recorded for bench marking. Concurrently the skeletal models of MADS data set will be investigated for compatibility as inputs to the AS-GCN trained model. If the provided skeleton models are not directly compatible OpenPose toolbox (Hidalgo, et al., n.d.) may be implemented to estimate skeleton data.

The trained AS-GCN model recognition and prediction head model shall be applied to the MADS data set for temporal localization of component action recognition. The training data action classes are different from MADS, therefore NTU RGB+D 120 action classes shall manually be applied to MADS footage to assess the accuracy of the pre-trained AS-GCN model.

Finally, the MADS data set will be used to train a new AS-GCN model either as a binary classifier or with the 30 action sequence labels as the output to the network. This will require the author to deeper understand the model architecture of AS-GCN (Li, et al., 2019). The MADS data set may require pre-processing to; extract compatible skeletal data, reshape the data for the AS-GCN architecture and split the sequences out into shorter sequences.

In summary the objectives of the project are:

1. Deploy the AS-GCN model on cloud deep learning machine image and train
2. Manually assign labels actions from the ROSE dataset to the MADS dataset footage and assess the AS-GCN model accuracy
3. Pre-process the MADS for training a new AS-GCN with MADS action labels

5. Tools and Programming Languages

The accompanying code repository (Li, et al., 2019) for the AS-GCN model lists the following requirements:

- Python 3.6
- Pytorch 0.4.1
- pyyaml
- argparse
- numpy

The AS-GCN paper notes that models were trained using 8 GTX-1080Ti GPUs utilising the Pytroch CUDA integration. As comparable hardware is not directly available to the author, Amazon Web Service (AWS) cloud deep learning machine image (DLMI) shall be used for model training. Initial selection of DLMI and instances are shown below. The DLMI selected is includes Pytorch and NVIDIA CUDA frameworks and Conda environments. Amazon EC2 G4 Instances are entry level machine learning instances with NVIDIA T4 GPUs (Amazon Web Services, n.d.).

DLMI: Deep Learning AMI (Ubuntu 16.04), MXNet-1.6.0, Tensorflow-2.1.0 & 1.15.2, PyTorch-1.4.0, Keras-2.2, & other frameworks, configured with Neuron, NVIDIA CUDA, cuDNN, NCCL, Intel MKL-DNN, Docker & NVIDIA-Docker. (Amazon Web Services, n.d.)

Instance Size: g4dn.2xlarge, vCPUs=16, Memory (GB)=64, GPU=1, Storage (GB)=225, Network Bandwidth (Gbps)=Up to 25, EBS Bandwidth (GBps)=4.75.

6. Objective 1 – Deployment and Training

6.1. The Problem with AWS

As per the project proposal an AWS DLMI instance was set-up using a Putty interface. Immediately several impracticalities of this set up became evident. Firstly, the persistent storage and runtime costs – sometimes amounting to several hundred USD per month – were unfeasible for such an independently funded project. Of course, the main reason this instance was selected was for the NVIDIA T4 GPU as the AS-GCN pipeline is designed for GPU training. The high CPU count, high RAM and other features of the instance were of little value to the project. AWS were approached to request a reduction in surplus instance features to reduce cost, but such re-configurations were not available.

The other critical issue uncovered upon initial model training runs was that the amount of video memory was insufficient. The AS-GCN paper used a set-up with 8 GTX-1080Ti GPUs with a total video memory capacity of 88 GB (11 GB per card) and CUDA core count of 28672 (3584 per card). This is vastly greater than the AWS DLMI NVIDIA T4 GPU with 16GB video memory and 2560 CUDA cores. The initial assumption had been that most of that GPU processing power was used to keep the training times down but it's now clear that an element of GPU parallel processing was required as minimum. Given that initial instance size was already economically unfeasible, to move to computationally more powerful GPU instances to match the paper would have been further unfeasible.

6.2. Improved hardware set-up

Given the experience of attempting to deploy the model on AWS, a local hardware set-up was deemed a better approach forward both technically and economically. With a local set-up data and code changes could be uploaded/applied instantly (as opposed to a bottle necked Putty interface). And the cost of purchasing a GPU would amount to several months of projected AWS costs anyway. The details of the final local set-up are listed below. This set-up had slightly more total video memory at 20GB (AWS g4dn.2xlarge: 16GB vram) but significantly more total CUDA core count of 5888 (AWS g4dn.2xlarge: 2560) and much less CPU/RAM resource which is sufficient for this projects experimental needs. Comparatively speaking this equated to around 20% processing capacity compared with the system used for the reference paper.

HP Omen 15 2020, Intel® Core™ i7-10750H 6 core processor, 16 GB memory, 1 TB SSD display, NVIDIA® GeForce RTX™ 2070 with Max-Q design (8 GB GDDR6, 2304 CUDA Cores), Windows 10 Home 64

Connected via thunderbolt 3 port; GeForce RTX™ 3060 GAMING OC 12GB GDDR6 (3584 CUDA Cores)

6.3. The infrastructure for the pipeline

Below are the pipeline packages and versions used for this project:

Python 3.8.8 (note that the original paper specified a pytorch version of 0.4.1)

Anaconda 2.0.4 – opensource conda package and environment manager

Pytorch 1.9.0 (note that the original paper specified a pytorch version of 0.4.1)

MATLAB R2021a update 4 - required to visualise the skeletons

6.4. Structure of the NTU RGB+D 120 dataset

Of the NTU RGB+D 120 dataset, only the Kinetics V2 Skeletons were downloaded (RGB, Depth and IR sequences were not required). The skeletons files contain information from each of the 0-24 joints at each frame. The detected joints are shown in Figure 4 below. Each joint has 11 associated properties: colour [x, y]; depth [x, y]; camera [x, y, z]; and orientation [x, y, z, w] (Jamhoury, 2018) as shown in . Each skeleton file is a sequence of frames thought the number of frames varies across the sequences in the dataset. Each skeleton file is named such that the coding represents all the information about the sequence such Subject, Camera & Action Class. The maximum performer count in any dataset sequence is two. Where there are more than one performers in a sequence, the graphs for all persons are captured in the skeletons files and there is unique identifier for each skeleton at each frame e.g. (frame 1 (Performer ID: 1(SKELETON), Performer ID: 2(SKELETON),..), frame 2 (Performer ID: 1(SKELETON), Performer ID: 2(SKELETON),..),..

File naming convention – SsssCcccPpppRrrrAaaa
sss is the setup number, ccc is the camera ID, ppp is the performer (subject) ID, rrr is the replication number (1 or 2), and aaa is the action class label.

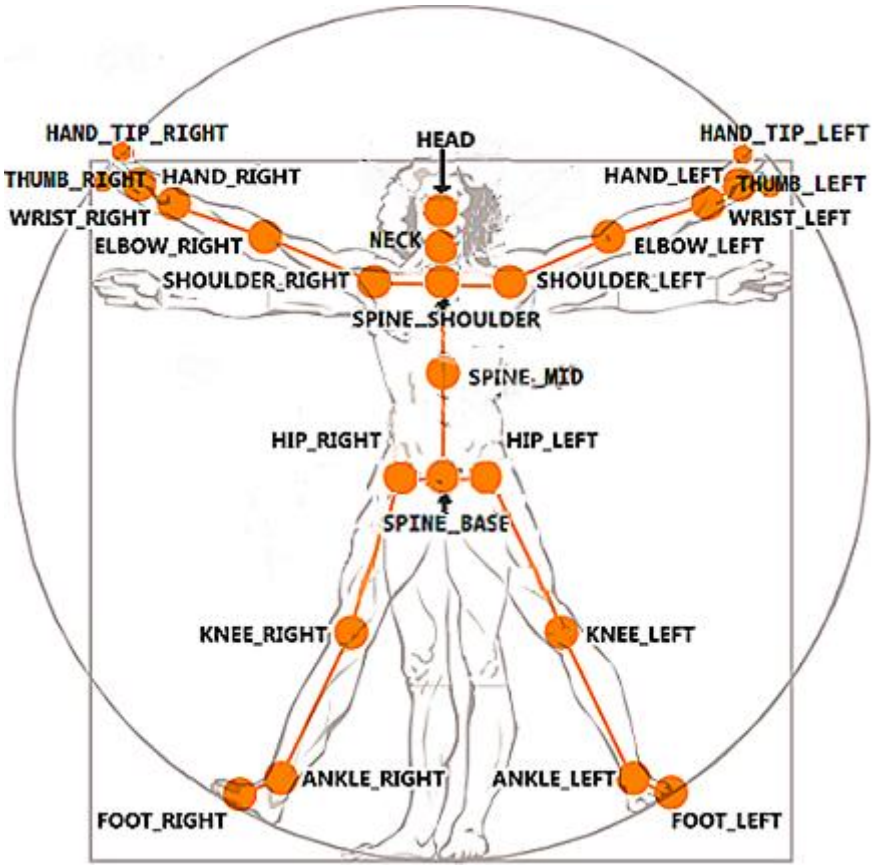
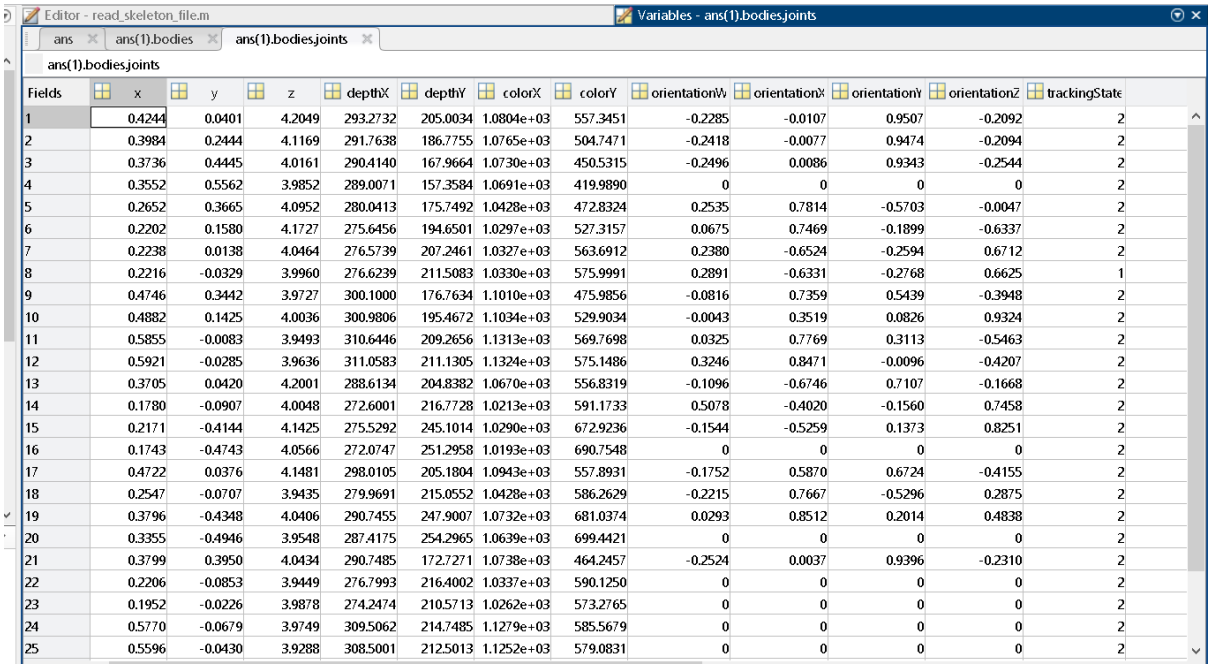


Figure 4 Kinect v2 25-joint skeleton. Image from Stack Overflow.



Editor - read_skeleton_file.m

Variables - ans(1).bodies.joints

ans | ans(1).bodies | ans(1).bodies.joints

ans(1).bodies.joints

Fields	x	y	z	depthX	depthY	colorX	colorY	orientationW	orientationX	orientationY	orientationZ	trackingState
1	0.4244	0.0401	4.2049	293.2732	205.0034	1.0804e+03	557.3451	-0.2285	-0.0107	0.9507	-0.2092	2
2	0.3984	0.2444	4.1169	291.7638	186.7755	1.0765e+03	504.7471	-0.2418	-0.0077	0.9474	-0.2094	2
3	0.3736	0.4445	4.0161	290.4140	167.9664	1.0730e+03	450.5315	-0.2496	0.0086	0.9343	-0.2544	2
4	0.3552	0.5562	3.9852	289.0071	157.3584	1.0691e+03	419.9890	0	0	0	0	2
5	0.2652	0.3665	4.0952	280.0413	175.7492	1.0428e+03	472.8324	0.2535	0.7814	-0.5703	-0.0047	2
6	0.2202	0.1580	4.1727	275.6456	194.6501	1.0297e+03	527.3157	0.0675	0.7469	-0.1899	-0.6337	2
7	0.2238	0.0138	4.0464	276.5739	207.2461	1.0327e+03	563.6912	0.2380	-0.6524	-0.2594	0.6712	2
8	0.2216	-0.0329	3.9960	276.6239	211.5083	1.0330e+03	575.9991	0.2891	-0.6331	-0.2768	0.6625	1
9	0.4746	0.3442	3.9727	300.1000	176.7634	1.1010e+03	475.9856	-0.0816	0.7359	0.5439	-0.3948	2
10	0.4882	0.1425	4.0036	300.9806	195.4672	1.1034e+03	529.9034	-0.0043	0.3519	0.0826	0.9324	2
11	0.5855	-0.0083	3.9493	310.6446	209.2656	1.1313e+03	569.7698	0.0325	0.7769	0.3113	-0.5463	2
12	0.5921	-0.0285	3.9636	311.0583	211.1305	1.1324e+03	575.1486	0.3246	0.8471	-0.0096	-0.4207	2
13	0.3705	0.0420	4.2001	288.6134	204.8382	1.0670e+03	556.8319	-0.1096	-0.6746	0.7107	-0.1668	2
14	0.1780	-0.0907	4.0048	272.6001	216.7728	1.0213e+03	591.1733	0.5078	-0.4020	-0.1560	0.7458	2
15	0.2171	-0.4144	4.1425	275.5292	245.1014	1.0290e+03	672.9236	-0.1544	-0.5259	0.1373	0.8251	2
16	0.1743	-0.4743	4.0566	272.0747	251.2958	1.0193e+03	690.7548	0	0	0	0	2
17	0.4722	0.0376	4.1481	298.0105	205.1804	1.0943e+03	557.8931	-0.1752	0.5870	0.6724	-0.4155	2
18	0.2547	-0.0707	3.9435	279.9691	215.0552	1.0428e+03	586.2629	-0.2215	0.7667	-0.5296	0.2875	2
19	0.3796	-0.4348	4.0406	290.7455	247.9007	1.0732e+03	681.0374	0.0293	0.8512	0.2014	0.4838	2
20	0.3355	-0.4946	3.9548	287.4175	254.2965	1.0639e+03	699.4421	0	0	0	0	2
21	0.3799	0.3950	4.0434	290.7485	172.7271	1.0738e+03	464.2457	-0.2524	0.0037	0.9396	-0.2310	2
22	0.2206	-0.0853	3.9449	276.7993	216.4002	1.0337e+03	590.1250	0	0	0	0	2
23	0.1952	-0.0226	3.9878	274.2474	210.5713	1.0262e+03	573.2765	0	0	0	0	2
24	0.5770	-0.0679	3.9749	309.5062	214.7485	1.1279e+03	585.5679	0	0	0	0	2
25	0.5596	-0.0430	3.9288	308.5001	212.5013	1.1252e+03	579.0831	0	0	0	0	2

Figure 5 Example of a single frame parsed into MATLAB where each row is a joint

Below are examples of the visualisations of frames from sequences.

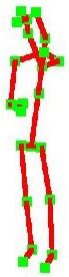


Figure 6 Example frame skeleton visualisation for action A001 - drink water

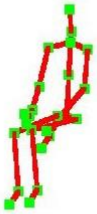


Figure 7 Example frame skeleton visualisation for action A008 - sit down

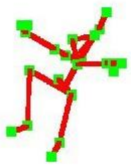


Figure 8 Example frame skeleton visualisation for action A016 - wear shoe. Note the confusion in joint detection when the joints are spatially compact

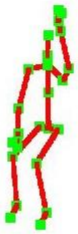


Figure 9 Example frame skeleton visualisation for action A023 – hand waving

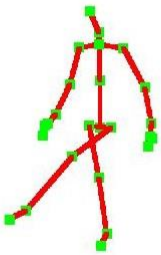


Figure 10 Example frame skeleton visualisation for action A024 - kicking

6.5. Model Evaluation: Cross View Vs. Cross Subject

Cross subject evaluation is the model's performance at action classification across different subjects or actors. The actors featured in either the training or validation set are exclusive. Cross subject evaluation can be interpreted as the model's ability to normalise variations in skeleton sizes. For example, an adult and a child will have different size skeletons but common everyday actions (e.g. drinking water, kicking a ball, shaking hands etc.) will use the same joint dependencies.

Cross View evaluation is the model's performance at action classification when the action is captured from different viewpoints. Every sequence in the NTU RGB+D dataset is captured in 3 different angles by 3 Kinetic V2. The camera samples used in the training or validation set are exclusive. The default for model was trained with samples from cameras 2 & 3 and validated with samples from camera 1. Cross view evaluation can be interpreted as the model's ability to normalise actor orientation to the camera. A model that is sensitive to orientation may inherently 'occlude' the joints furthest from the camera disregarding vital joint dependencies.

6.6. Project Pipeline

What follows is an in-depth description of the main AS-GCN pipeline.

1. *Data Pre-processing* – all the sequences/skeletons comprising the dataset, which have been pre-labelled by file name are merged with the AS-GCN clone repository. The pre-processing module [ntu_gen_preprocess.py] processes the data in the following ways.
    a. The 3D spatial data for each joint [x, y, z] for each performer in each frame of each sequence is extracted from the skeleton files.
    b. Each sequence is 'padded' to be 300 frames long by repeating the sequence from the start. At this point each sample is structured as [25 *joints* x 3 *spatial coordinates* x 300 *frames*]
    c. The entire dataset is compiled into a Numpy data frame and separate actions label s file.
    d. The dataset is split in a training/testing split ratio of 70%/30%
2. *Pre training of Actional Inference Module (AIM)* – At the onset of training, the method for model evaluation is selected; *Cross Subject* or *Cross View*.
   AIM is pre-trained for 10 epochs by default. This 'warms up' the encoder side of AIM which propagates between joints and links iteratively to learn link features (Li, et al., 2019). AIM is further trained during main pipeline training.
3. Main pipeline training – As for step2 the main training pipeline is separated into *cross-view* or *cross subject*. The main AS-GCN back bone is mini-batched trained using stochastic gradient descent (SGD) for 100 epochs by default. The model is evaluated every 10 Epochs. The Loss function is standard cross entropy loss. At every Epoch two models (.pt files)are output to the work directory for the recognition head and prediction head respectively.
4. Testing – the action classification for the recognition head is evaluated accordingly with either the *cross-view* or *cross-subject* evaluation performance. The performance metrics are displayed for the top-1 and top-5 accuracys. Top-1 is the accuracy of the model's highest probability predictions (otherwise known as conventional accuracy). Top-5 is the accuracy that the correct class is in the models 5 highest classes for prediction by probability.

6.7. Project pipeline configuration

The project is setup to configure Pytorch by way of YAML files. There are 2 corresponding YAML files (cross view and cross subject) for each of Steps 2,3,4 above. The pertinent variables and their default values that were amended for this project were;

device: [0,1,2,3] – Specifies which GPU's to be used based on torch.cuda.device() numbering

batch_size: 32 – Specifies the number of samples/sequences for DataLoader to load per iteration. Note this parameter is limited by GPU memory capacity.

test_batch_size: 32 – As above but for testing.

start_epoch: 10 – Indicates which model weights should be called from the working directory by Epoch.

num_epoch: 100 – Indicates the number of times to process the entire the dataset once. Note the time per epoch is related to the size of the dataset.

num_worker: 4 – if this is greater than zero, Pytorch uses multi process data loading.

6.8. Findings about the paper repositories

Unfortunately, the AS-GCN paper main repositories were not well maintained and there was an extensive amount of debugging required for the project. There were several technical issues that were beyond the scope of this project to surmount which limited the range of the project;

1. No pretrained model weights were supplied on the AS-GCN Github repositories.
2. When attempting to use multiple GPU's to train the main pipeline the following error was encountered; "Caught RuntimeError in replica 0 on device 0". This appears to be an issue with Pytorch parallelising the model across multiple GPU's possibly occurring due to the mismatch in Pytorch versions (paper version: 0.4.1, project version: 1.9.0). As such main pipeline training was only possible on 1 GPU so the batch size had to be significantly reduced to 4 (default 32).
3. The AS-GCN repository did not include any code fragments or instruction relating to the 'prediction head'.
4. Amending the number of classes in the model (via *num_class* in the YMAL files) seemed to break the model so was kept at the default value of 60.

6.9. Runtime

Given the reduced GPU capacity for main pipeline training, it was decided that the data set should be reduced from 60 actions to 5. This bought Epoch training times down from approximately 40-50 minutes to 3 minutes per Epoch. The number of Epochs was also reduced from 100 to 30-40. The selected actions were: A001 - drink water, A008 - sit down, A016 - put on a shoe, A023 - hand waving, A024 - kicking something.

# actions: 5, # samples: 4740, Dataset size: 1.05 GB	
A001 - drink water	948 samples
A008 - sit down	948 samples
A016 - put on a shoe	948 samples
A023 - hand waving	948 samples
A024 - kicking something	948 samples

Table 5 Objective1 dataset

The Pytorch configuration files (.ymal), model files (.pt) and log files (.txt) are included in the project github repository in *Objective1*:
https://github.com/Birkbeck/msc-data-science-project-2020_21---files-arusse13

6.10. Results

The final accuracy of the trained models is shown below:

Dataset	Top-1 Accuracy (%)	Top-5 Accuracy (%)
Complete NTU-RGB+D Cross Subject (Li, et al., 2019)	86.8%	
Complete NTU-RGB+D Cross View (Li, et al., 2019)	94.2%	
Kinetics (Li, et al., 2019)	34.8%	56.5%
Project Results NTU-RGB+D [A001, A008, A016, A023, A024 classes only] Cross Subject	88.18%	100%
Project Results NTU-RGB+D [A001, A008, A016, A023, A024 classes only] Cross View	80.74%	100%

Table 6 Objective1 action recognition classification accuracy

6.11. Conclusions

Upon first viewing the results the 100% top-5 accuracy immediately seems exciting however, the top-5 accuracy can be disregarded as the limited number of 5 classes means the correct class will always be present in the top 5 highest probability classes.

The cross-subject project results are comparable to (Li, et al., 2019) results which we would expect given Objective1 is repeating their work albeit with a reduced dataset size. However, the cross-view project results show significantly less accuracy (-13.46%) in contrast. Both the cross-subject and cross-view project results accuracy are well beyond that of the (Li, et al., 2019) Kinetics results. A value of 20% or less would suggest the model is performing less accurately then selecting a class at random. Higher accuracy might have been expected given the reduced dataset size thought this may be a symptom of not reducing the learning rate despite a reduced batch size.

8. Objective 2 – Model Evaluation by Hand

The significant amount of debugging/re-deployment time for objective 1 significantly constrained the amount of time available for this project and so unfortunately this objective was not achieved.

9. Objective 3 – New Data Training

9.1. Structure of MADS Data

The MADS dataset is supplied as a combination of video files (.avi), sensory information files e.g. depth, IR (.m) and associated code to parse the information into MATLAB.

# actions: 5, # samples: 60, Dataset size: ~25.0 GB	
Tai-chi	12 samples
Karate	12 samples
Jazz	12 samples
Hip-hop	12 samples
Sports	12 samples

Table 7 MADS dataset

Upon review of the MADS datasets applicability to the AS-GCN pipeline three problems were evident.

1. The joint information for each sequence is stored as MATLAB (.m) files. Recall that the supplied pre-processor for AS-GCN accepts only Kinetic V2 skeleton files.
2. The MADS dataset captured only 19 joints per frame where the NTU RGB+D dataset captured 25 joints per frame meaning the AS-GCN pipeline is configured for 25 joints.
3. Each MADS Sequence is 80seconds long on average and there are in total 30 sequences captured across 3 cameras. In contrast the NTU RGB+D dataset has approximately 1000 skeleton files per action which vary in frame count from 50 to over 100 frames.

Due to time constraints, it was deemed more efficient to create new joint data from the video sequences of the MADS dataset rather than attempt to wrangle the depth data from the MATLAB files. Fortunately, ST-GCN (Yan, et al., 2018) – which the AS-GCN project is forked from – has an Openpose pre-processor available. AS-GCN is the same pipeline as ST-GCN (the difference being that AS-GCN features actional links as opposed to just structural links) and so the same Openpose pre-processor is compatible. The solutions to above problems were as follows.

1. Divide each MADS sequence into shorter sequences such that the number of training sample per action were of a similar order to NTU RGB+D dataset.
2. Extract joint data from the sequences using Openpose.

It was decided that the ‘Sports’ category of the MADS dataset not be used as it was comprised up 6 separate sports and so seemed to broad a class label in contrast to the other martial arts and dance category’s

9.2. Splitting the Sequences Using FFmpeg

FFmpeg is an opensource suite of library's for multi-media (video, audio, pictures etc.) processing. It is command line based so simple batch files can quickly be written to automate the processing across whole data repositories. FFmpeg's processing capabilities cover decoding, encoding, transcoding, muxing, demuxing, streaming, filtering and playing (Anon., n.d.).

The batch files commands for splitting and reducing the resolution of each sequence are included in the project GitHub repository. The scripts operate as follows.

1. Manually assign AS-GCN compatible labels to each sequence ready to be splitting as shown below. Where ?? are wild cards ready for indexing.
2. Recursively find all .avi action sequences in a directory (including all sub directories).
3. For each found file; split and then reduce resolution of the action sequence

9.3. Introduction to Openpose

Openpose is an open source, real time multi person 2D pose estimation toolkit developed by (Cao, et al., 2019). Openpose takes each frame from a sequence as input to a CNN which in parallel predicts confidence maps for body part detection (graph joints) and Part Affinity Fields (PAF) for part association (graph edges). Biparte matching is used to parse the predictions to a complete graph. The success of Openpose is down to its bottom-up approach – predicting the graph before predicting the associated subjects (actors) – which allows for both increased accuracy and decreased computation time even on consumer grade GPU hardware over top-down systems. (Top-down systems tend to estimate the subject first and then apply a filter so that each subject can be graphed individually).

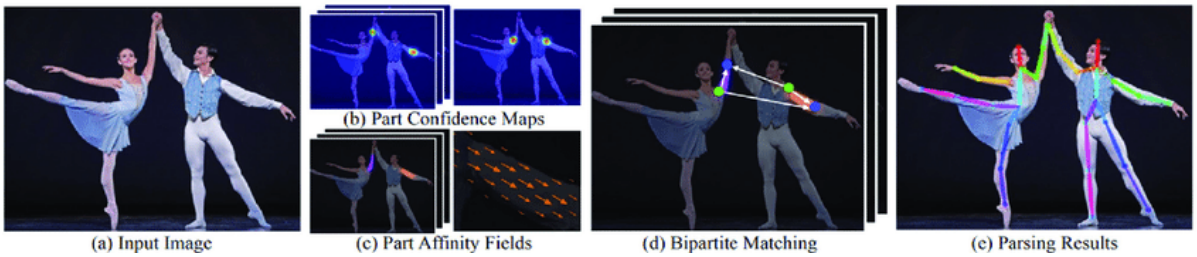


Figure 11 Pipeline of Openpose (Sabater, et al., 2021)

9.4. From Openpose to AS-GCN

Each MADS sequences was processed with the Openpose BODY_25 model. The 2D locations for each of the 25 joints were parsed out as JSON files for each frame. Note that the Openpose BODY_25 defines graph nodes (or joints) differently to the Kinetic V2 skeletons (the impacts of this are discussed in section 9.8.14). As the AS-GCN paper specifically references using Openpose as a way to input non-Kinetic V2 skeletal shaped data, it is assumed that the difference in skeletal graph forms will at least not dramatically reduce the accuracy of AS-GCN.

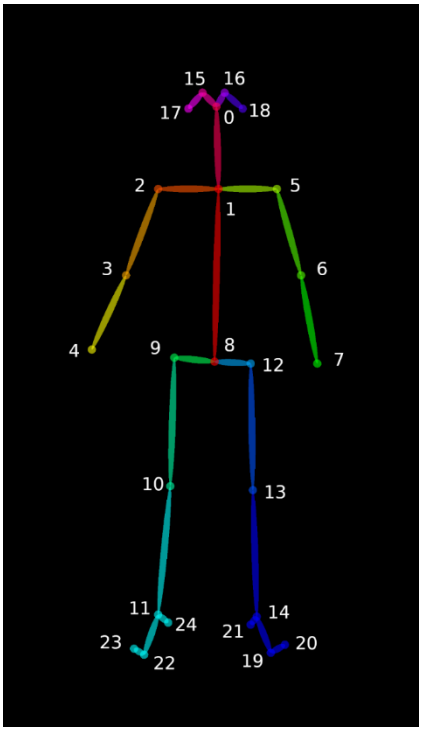


Figure 12 Pose Output Format (BODY_25) REF

Presenting the Openpose, 2D keypoint information directly to the AS-GCN network would not be possible as it requires 3D node (joint) information [x, y, z]. In lieu of a depth co-ordinate the confidence factor (from confidence mapping phase) for each Openpose node (joint) location was used. Thought not explicitly stated anywhere, it is assumed this acts as proxy for depth given subjects closer to the camera will have higher confidence scores for joint locations and vice versa.

Once all the MADS sequences had been output as JSON files for each frame it was required that the frames be remerged into single sequence files. This presented a convenient point to separate each action sequence into shorter sequences. It was decided that each sequence would be broken into the 5 frame sequences.

To automate this process as much as possible batch files were written and can be found on the front page of the project Github repository:

https://github.com/Birkbeck/msc-data-science-project-2020_21---files-arusse13

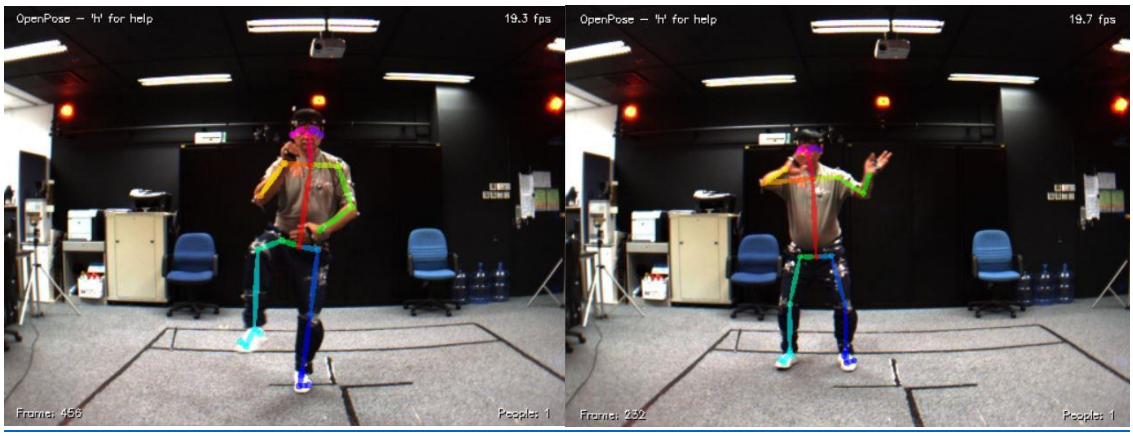


Figure 13 two frames of Tai-Chi Openpose processing

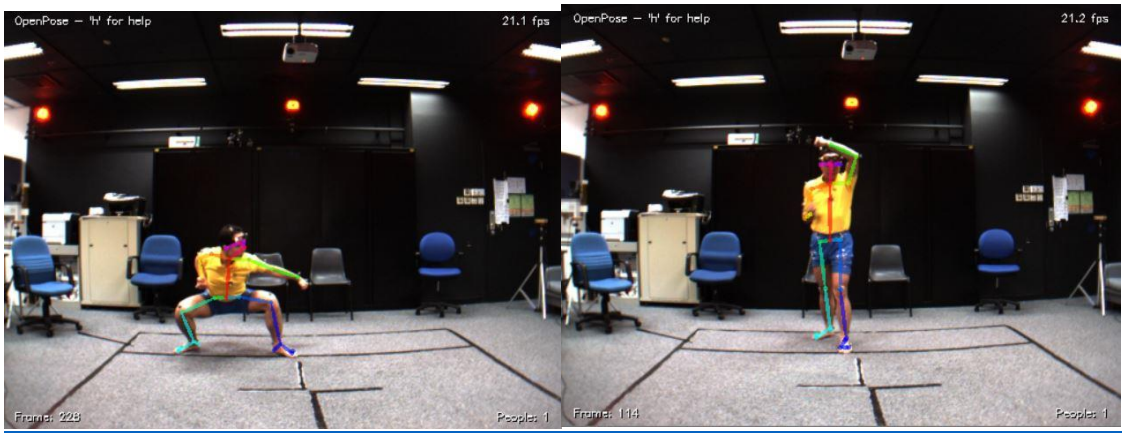


Figure 14 two frames of Karate Openpose processing

Figure 15 two frames of jazz dance Openpose processing

Figure 16 two frames of Hiphop Dance Openpose processing

9.5. Runtime

Once the data pre-processing steps were completed the data set looked as follows;

# actions: 4, # samples: 648, Dataset size: ~47.0 MB	
Tai-chi	82 samples
Karate	226 samples
Jazz	166 samples
Hip-hop	174 samples
Sports	Omitted

Table 8 Objective3 dataset

The training time for each Epoch was 1-2 minutes.

The Pytorch configuration files (.ymal), model files (.pt) and log files (.txt) are included in the project github repository in *Objective3*:
https://github.com/Birkbeck/msc-data-science-project-2020_21---files-arusse13

9.6. Results

Dataset	Top-1 Accuracy	Top-5 Accuracy
Complete NTU-RGB+D Cross Subject (Li, et al., 2019)	86.8%	
Complete NTU-RGB+D Cross View (Li, et al., 2019)	94.2%	
Kinetics (Li, et al., 2019)	34.8%	56.5%
Project Results NTU-RGB+D [A001, A008, A016, A023, A024 classes only] Cross Subject	88.18%	100%
Project Results NTU-RGB+D [A001, A008, A016, A023, A024 classes only] Cross View	80.74%	100%
Project Results MADS [Karate, Tai-Chi, Jazz, HipHop classes only] Cross Subject	35.19%	100%
Project Results MADS [Karate, Tai-Chi, Jazz, HipHop classes only] Cross view	94.36%	100%

Table 9 Objective3 action recognition classification accuracy

9.7. Conclusions

Objective 3 project results show comparable accuracy on cross-view evaluation to (Li, et al., 2019) but significantly less accuracy on cross-subject evaluation. This is highly likely a symptom of having only 1 subject/performer per action for the MADS dataset so the model will have only learnt very weak generalisation between different subjects. Note also that the dataset for objective3 featured less than 14% (objective1:4740, objective3:648) the number of training samples then objective1. As for the 6.10 the Top-5 accuracy is redundant information in this experiment. When compared to (Li, et al., 2019) Kinetics results the cross-subject performance is comparable and the cross-view performance is far more greater. Given the 3 datasets presented in Table 9 (MADS, NTU RGB+D and Kinetics) it is apparent that AS-GCN performers much better in studio footage settings (MADS, NTU RGB+D) then on footage from 'the wild' (Kinetics is a dataset formed of Youtube videos).

9.8. Further Experiments & Development

This project had a very limited scope given the time and resources available, this section outlines what further experiments might be attempted to gain deeper insight into the performance of AS-GCN. Each proposal is written in the format: idea, usefulness, implementation, measurement.

9.8.1. Adjust the division of the data

In this project the MADS dataset samples were split in 5 second sequences to increase the number of samples for each action and resize each sample to be a similar number frames as the NTU RGB+D dataset. As the AS-GCN data pre-processor pads out each sequence so that all samples are 300 frames the theoretical time length for any sequence is 12.5 seconds (assuming 24fps frame rate). By creating new MADS split datasets with sequence lengths of less then 5 seconds (3 seconds) and greater then 5 seconds (8 seconds, 10 seconds) on sequences, experiments in this project can repeated in order to establish the effect of sequence length on the AS-GCN generalisation performance (by way of classification accuracy). Such experiments should include sequence length variations between the training and validation sets (e.g. train using 8-second sequence length data set, validate using 3-second sequence length data set). There would need to be some consideration about the impact of reducing the number of samples in the training set. A reasonable prediction would be that as the number of frame in a sequence approaches 0, the closer to a static 3D image is the sequence and so AS-GCN approaches the performance of a static pose classifier.

9.8.2. Attempt a higher resolution

this is purely for the situation where Openpose is used to extract 2D keypoint data in-lieu of direct Euclidean 3D joint positions (from 3D sensing devices). In the above project each sample was reduce in resolution to 340x256 in line with the ST-GCN pipeline (Yan, et al., 2018). The MADS dataset was originally at resolution of 512x380. It can reasonably be assumed that higher resolution sequences result in higher confidence scores for joint locations from Openpose. This may raise the confidence score of all joints proportionally or may amplify the difference between joints that are close or far to the camera (close=high confidence, far=low confidence). The former effect would be suboptimal as a higher resolution processing requires greater computational effort (especially time and video memory, ideally there is a limit were a higher resolution doesn't affect confidence mappings). As the confidence score for each joint position at each frame is used as a proxy for true depth this may have a significant impact on the AS-GCN classification performance. Such an experiment should also vary the resolution between the training samples and the validation set e.g. (training samples=high resolution, validation samples=low resolution) to ascertain if harmonised data resolution is required for AS-GCN application deployment.

9.8.3. Overlap Splits during sample division

the below splitting sequence may serve to generate more data samples without compromising the classification performance. This theory is because the data pre-processor implicitly repeats data during padding anyway. Such an experiment should seek to establish a value of overlap as a ratio of overlapped frames to non-overlapped frames (e.g. 1second of overlapped frame for 5seconds of non-overlapped frames). This would expand the number of training samples for each action class.

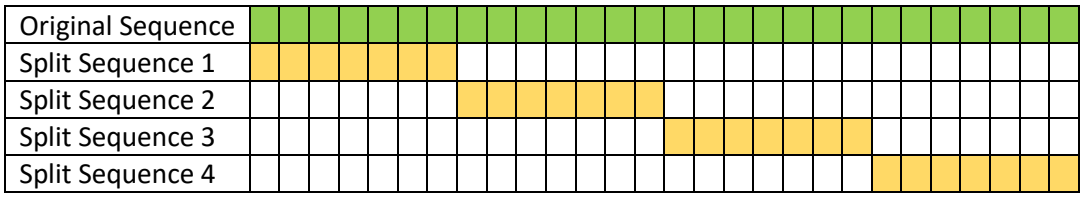


Table 10 Objective3 sample splitting

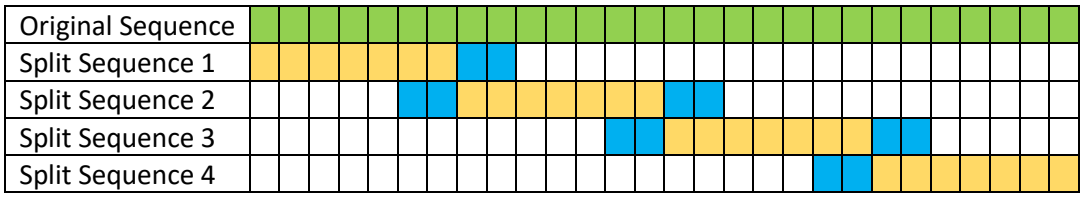


Table 11 Proposed sample splitting overlapping frames (blue)

9.8.4. Expand out each action into smaller component actions

in this project it was decided that the 'sports' action be excluded for several reasons. Firstly, the other actions are also classable as 'sports'. Secondly the 'sports' action samples were comprised of 6 distinct sports (badminton, football, rugby, tennis, volleyball, basketball, volleyball) despite the category of 'sports' having the same number of samples as the other categories. So, if the individual sub-sports activities had been trained as individual classes, they contain only 1/6 of the training samples than the other prime classes (tai-chi, hip-hop, jazz, karate). The martial arts categories of karate and tai-chi also had labels for sub sequences. To use this sub-data would require manually relabelling the data. To counter the imbalance/lack of resulting data sample one might introduce other datasets or even a totally different learning paradigm such as one-shot learning as postulated by (Sabater, et al., 2021) or transfer learning as postulated by (ABDULAZEEM, et al., 2021). These Paradigms are discussed in deeper depth in section 10.

9.8.5. Incorporate depth data from MADS

the decision to not use the depth data from the MADS dataset was purely based on time constraints. To parse the data from MATLAB files into JSON files could be done by reverse engineering the NTU RGB+D MATLAB parser (Shahroudy, 2019). Implementing this would take a significant amount of development time.

9.8.6. Increase batch size

Using a batch size of 4 – down from the default of 32 – pushes the learning process more towards online learning which means significantly more randomness in batch steps (BILOGUR, 2018). This decision was driven predominately by a critical error in the AS-GCN codebase which did not accept GPU parallelism which meant that all main pipeline training was restricted to a single GPU. A full debug of the AS-GCN pipeline is required to correct this problem on the most recent versions of Pytorch. Based on the experience of this project and literature on AS-GCN, it reasonable to assume that 2.75GB of video memory is required per unit of mini batch for training the main pipeline (e.g. batch=4 requires total video memory=4*2.75=11GB, batch=32 requires total video memory=32*2.75=88GB).

9.8.7. Improved hardware setup

An improved hardware setup can be either an extension to the local hardware set-up or a more suitable cloud service. For the former, entry level GPU's on the market feature large video memory (VRAM) and motherboards usually feature multiple PCI-Ex16 slots for direct card mounting as well multiple PCI-Ex 4 slots for riser mounting. It is estimated that an equivalent machine setup to that used in AS-GCN paper could assembled for around £4000. However, at the time of writing (Summer 2021) there is major chip shortage worldwide leading to supply issues with GPU and inflated prices. As an alternative, recently several cloud GPU services have become available such as Lambda Labs Inc. which are specifically developed for scaled training requirements. At the time of writing the below product offering from Lambda (Lambda Labs, Inc., 2021) is extremely competitive when compared with DLMI from AWS. Such an instance would be comparable to local set-up used in for the AS-GCN paper;
2x A6000 GPUs (48 GiB VRAM each), NVLINK=Yes, VCPUS=28, AMD EPYC™ (2.50 GHz) CPUs, 200 GiB, 1 TiB, ON-DEMAND PRICE= $4.50 / hr

9.8.8. Adjust learning rate

Although the batch size the was adjusted, the learning rate was kept at the default values for both the AIM training (0.005) and the main pipeline training (0.1). As is well known in the field of deep learning, larger learning rates work well with larger batch sizes as there is greater confidence in the direction of the gradient (Michaus, 2017). The randomness in small batch sizes requires more iterations and smaller steps to reach minima of the loss function. Thought decayed learning rates are used (0.1 every 20 epochs) it would be prudent to rerun the experiments with smaller initial learning rate.

9.8.9. More Action Classes

In this project the base line training experiment used 5 action classes and the MADS data set action classes used 4 action classes. The number of action classes for the baseline experiment was reduced to be in line with the action classes for the MADS dataset and to keep the time for each Epoch down to feasible level (initial full NTU RGB+D dataset epoch time 40-50 minutes, reduced 5 class dataset NTU RGB+D epoch time = 2-3 minutes). The more action classes trained using AS-GCN with a resulting reasonable level accuracy, the greater the generalization performance of the network and the greater the application potential of AS-GCN. Of more technical value to the field would be further studies for specific types of actions within activities (e.g. football actions models, actions = volleys, step-over, pass etc. application = kicks for an individual player in football).

9.8.10. Resolve the fixed classed output (adjust number of end nodes of network)

the inability to adjust the number of output classes of the final layer of network suggests that large parts of the network were effectively 'dead'. Thought the neurons of the 'dead' classes in final layer

would never see any activation – as there were no labels to trigger them in the forward pass – these nodes are still being initialized with associated edges which will consume amounts of computation resources. This may explain why the foot print of the batch size = 4 in the GPU vram was the same whether it was full NTU RGB+D dataset (with all action classes being trained) or the smaller the subset comprised of 5 actions. This again is something for further debugging of the AS-GCN.

9.8.11. Integrate the Model into a demo application (offline and/or real time)

ST-GCN (Yan, et al., 2018) – which is the frame work for AS-GCN – features both an offline (no video playback with skeleton plotting) and real time pose estimation using the Openpose python API. Adapting this source code would be the most efficient way to get any trained AS-GCN models into deployment. Note that ST-GCN has now been developed into a full opensource tool box under the name of MMSkeleton (Multimedia Laboratory, CUHK, 2021). The action list for this project includes 3D pose estimation which is assumed to be the formal integration of AS-GCN.

9.8.12. Deep dive into feature maps for each action class

To the consider the inference potential of AS-GCN the feature maps of each layer must be interpretable. As the AS-GCN uses Euclidean joint positions in time to build spatial and temporal feature maps of the structural and actional links the interpretable feature maps will be relative to individual joints e.g one might predict the action of handshaking would rely almost entirely on joints below the shoulder of the shaking hand and so all joints. This would be best represented as a heat map on a neutral image of the skeleton such as Figure 4 or Figure 12 probably using the MATLAB. This inferential insight has promising applications in biomechanics and sports.

9.8.13. Expanded dataset

A natural progression of this study is to expand out the datasets to further reinforce the selected action classes. At the time of writing paperswithcode.com listed 50 video data sets related to action recognition (Facebook AI Research, 2021). Of those 3 datasets are specific to sports action recognition; sports-1M (Karpathy, et al., 2014), UCF101 (Soomro, et al., 2012) and TAPOS (Shao, et al., 2020). The former 2 datasets are over seven years old and compilations from YouTube. The latter dataset is relatively recent and features labelling at the sub-action level thought is limited to footage taken from athletics and gymnastic events only. The datasets used in this project have been created in controlled-repeatable environments featuring fixed camera positions, blank back grounds and specific short actions by one or two actors at most per sequence. When incorporating a new dataset consideration should be given as to a similar repeatable control video dataset or ‘in the wild’ footage is preferential.

9.8.14. Assessment of structural links

The Kinetics V2 (NTU RGB+D) and Openpose extracted skeletons used significantly different key points for joints (see Figure 4 and Figure 12). Contrastingly Openpose skeletons considers eyes/ears/heel/big toe/small toe as joints and Kinetics V2 considers much more details in the hands. Both of these formats use 25 joints. A comparison of which is better can be achieved easily by applying Openpose to NTU RGB+D video sequences, training a new AS-GCN on this new data set and comparing accuracy to the AS-GCN paper.

Also note that Openpose has the ability to add vastly greater joint densities at the hands and around the face as shown below. Both hand and face gestures can form apart of or exclusively be used as part of actions (e.g. sign language, lip reading). Applying these joint formats would be an interesting natural development of AS-GCN.

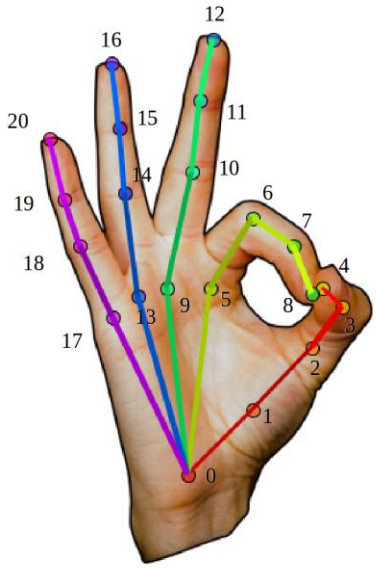


Figure 17 Openpose Hand Output Format (Hidalgo, et al., n.d.)

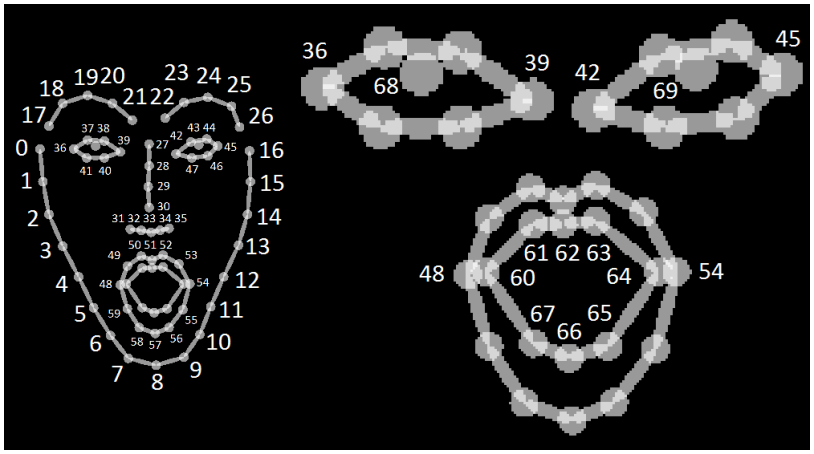


Figure 18 Openpose Face Output Format (Hidalgo, et al., n.d.)

9.8.15. Prediction Head Development:

The AS-GCN paper outlines the prediction head as the decoder to the A-link inference module. Unfortunately, the github repository is completely absent of any coding for this decoder pipeline. Given that the authors of AS-GCN released a concurrent paper at the same time titled 'Symbiotic Graph Neural Networks for 3D Skeleton-based Human Action Recognition and Motion Prediction' (Li, et al., 2019) and also an upcoming publication by the lead author titled 'Class-Agnostic Action-Skeleton Graph Neural Networks for 3D Skeleton-based Human Motion Prediction' (listed as in submission on authors website) (Li, 2021), it would seem that section of the code base was deliberately omitted to be further developed for the specific area of research. In this authors opinion at the time of writing, future action/pose prediction will be a hot area of computer vision research in the future with applications being built upon action classification usages e.g. Insight into what an opposing player will action next given a pose in sports.

10. Further Developments in the Field of Graph-Action Recognition

10.1. One-Shot Learning for Action Recognition

One-shot learning in the context of machine learning is when a model is trained on just a single sample. It is an active field of study with applications in physio/medical therapy and virtual/augmented reality. Such applications require the ability to learn on-the-fly whilst being view in variant but not subject invariant. This is because generally these applications are applied to single subjects at a time e.g. tracking movements of a patient for physiotherapy, calibrating a user's actions for virtual reality user interface etc.

The work by (Sabater, et al., 2021) presents a one-shot learning framework for one-shot action recognition using joint graphs (skeletons) of the same format as AS-GCN. The foundation of the framework is that a single training action sample or anchor sample is encoded and the model checks for the same encoding pattern in the target motion sequence as shown in Figure 19.

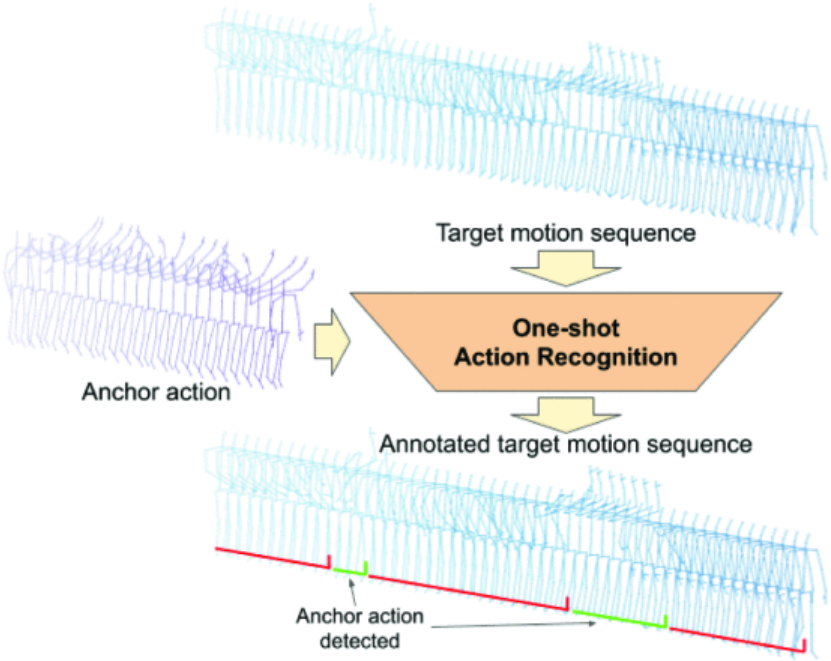


Figure 19 Overview of the proposed action recognition method (Sabater, et al., 2021)

The proposed framework is comprised of three stages; pose normalization, pose features and motion descriptor generation from a TCN, One-shot action recognition. Pose normalization transforms each frame 3D skeleton space to be centred at the midpoint between the two hip joints. This makes the encoding view and location invariant. Pose features calculates the pair-wise Euclidean distances between points and bone angles. These features are then feed to a motion descriptor temporal convolution neural network for temporal feature extraction of the sample which outputs the anchor sample embeddings. Finally, the anchor embeddings are compared to target embeddings by way of the cosine distance and the Jensen-Shannon divergence. This framework is shown diagrammatically as Figure 20.

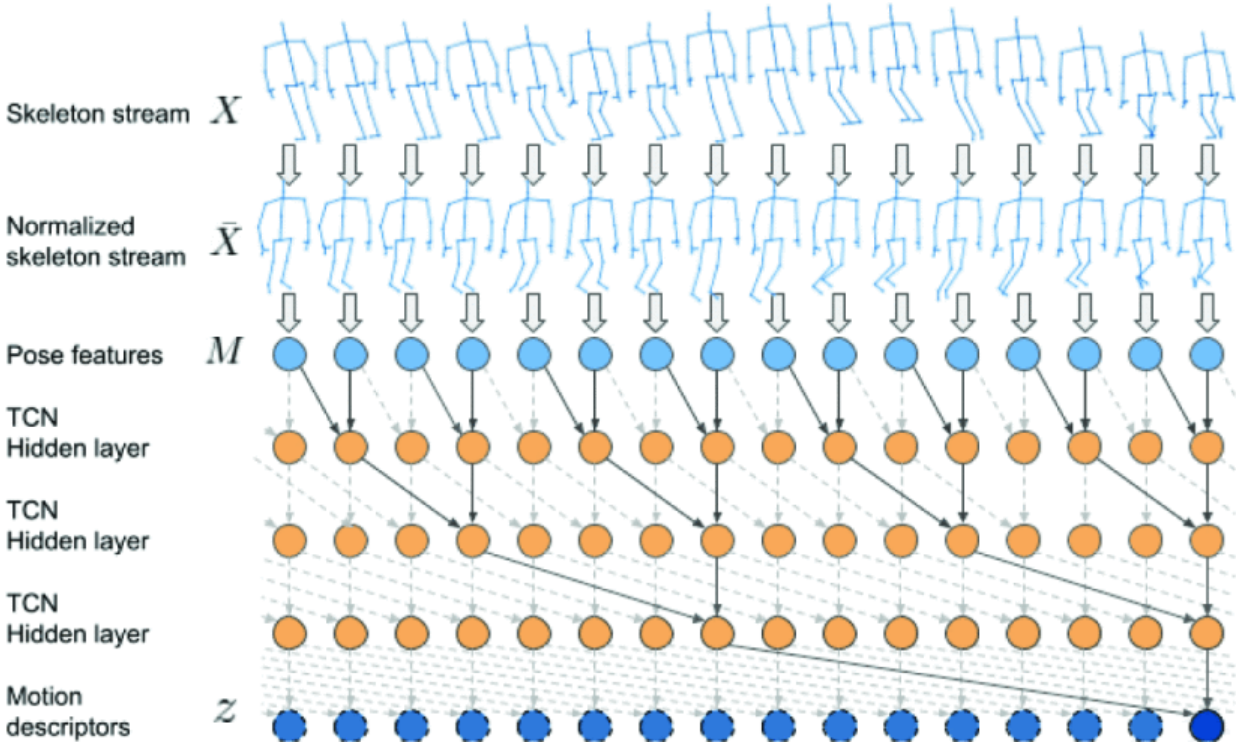


Figure 20 Motion descriptor generation (Sabater, et al., 2021)

Given that the size of the MADS dataset was so small relative to NTU RGB+D this approach would be prove an interesting counter development strategy for this project. As MADS featured numerous samples it would be more appropriate to consider such a study few-shot learning. It is envisioned that such a project would progress as follows; for each action an anchor sequence – which is a shorter frame sequence from the one of original samples – is trained and tested against all of samples, this is repeated for all actions across all possible anchor sequences, the accuracy would be the cosine distance and Jensen-Shannon divergence and precision/recall/F1 across these results. Note that the original paper by (Sabater, et al., 2021) used NTU RGB+D for training so a direct counter study with same pre-processed data from this project may easily implemented. Note that a major advantage to this one-shot technique is the reduced hardware requirements (NVIDIA GeForce RTX 2080 Ti (GPU) and a Intel Core i7-9700K (CPU)) being approximately 1/8 than that of the original AS-GCN paper and even being trainable on the CPU. It would offer an interesting insight into which is better for graph-based action recognition given a dataset of a set size:

Structural & Actional Links trained by stochastic gradient decent
OR
One-shot embeddings for each possible anchor sequence

10.2. Transfer Learning for Action Recognition

Transfer learning in the context of machine learning is when knowledge from one domain (or model-dataset) is used in similar-target domain (or model-dataset) to improve/augment performance and significantly reduce training time. The theory being that similar models extract the same features. For instance, it is well documented that many image recognition models extract edge features from an image independent of model architecture or dataset. This concept is depicted in the Figure 21.

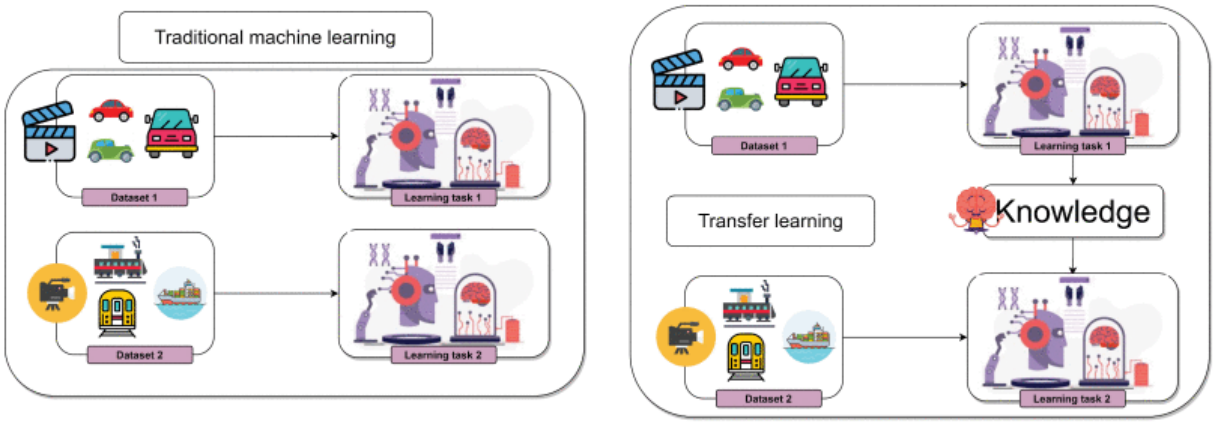


Figure 21 Transfer learning vs. non-transfer learning graphical illustration (ABDULAZEEM, et al., 2021)

The recent work by (ABDULAZEEM, et al., 2021) explains the three principle methods of applying transfer learning as being;

- Fixed Feature Extraction; this is where a model (neural network) which has been pretrained on the knowledge domain dataset is exported for use persevering all network structures, weights and bias except the final fully-connected layer is replaced with one or more layers to suit the target domain dataset. The preservation of the architecture, weights and bias perseveres or 'fixes' the feature extraction knowledge of the network across domains.
- Fine-tuning and Layer Freezing; this is where a model (neural network) which has been pretrained on the knowledge domain dataset is exported for use persevering all network structures, weights and biases. The exported network is then further trained on the target domain dataset to 'fine tune' the weights and biases. In some instances, it is advantages to freeze layers to persevere that area of knowledge. If the required feature knowledge for the target domain is similar enough to the knowledge domain, then this fine-tuning technique should offer greatly reduced training times and reduced computational power requirements.
- Pre-trained models; this is where a models architecture only – not pre-trained – is exported for use from the knowledge domain. The exported model is then fully trained on the target domain dataset. If the required feature knowledge for the target domain is similar enough to the knowledge domain, then this fine-tuning technique should offer greatly reduced research and development time compared with designing and cross validating new model. However the training time and associated computational power requirements will be as per the original model.

The above three methods may be considered a scale of knowledge transfer with fixed feature extraction being the maximum, fine-tuning and layer freezing being around the mid-point and pre-trained models being the minimum. All feature a common foundation of adopting a pre-validated network structure. The right transfer learning method for the objective can offer: significantly

reduced research & development time, reduced training time, increased model generalisation and increased knowledge accessibility.

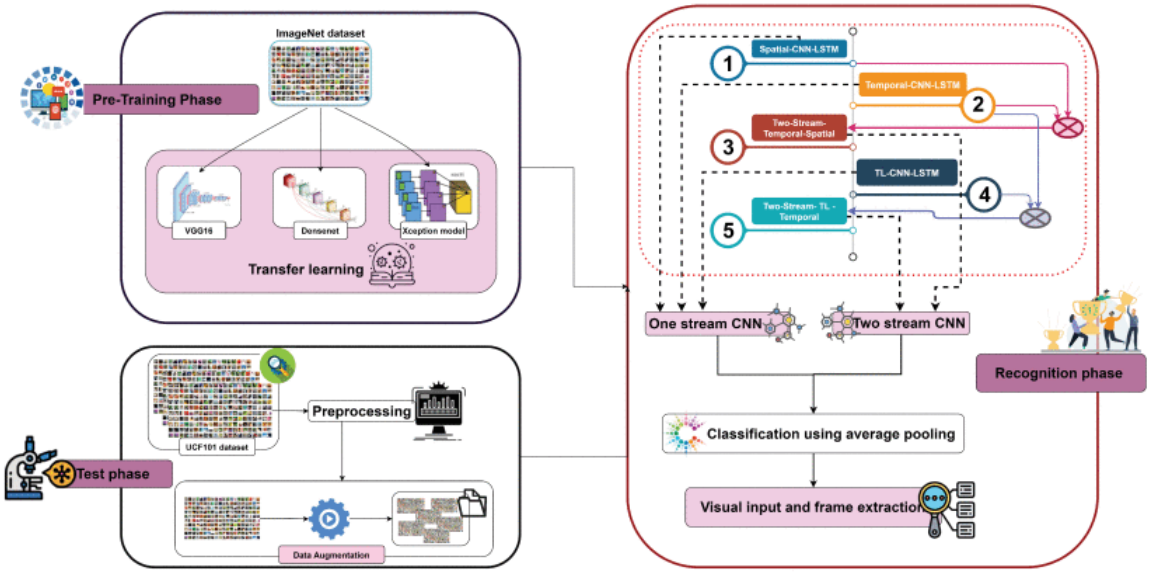


Figure 22 The transfer learning-based human action recognition (TL-HAR) framework. (ABDULAZEEM, et al., 2021)

(ABDULAZEEM, et al., 2021) present a combination of the above methodologies to 'assemble' a network for action recognition by using *fixed feature extraction* image classifier models which are incorporated into a *pre-trained* two stream, CNN and long short-term memory network for temporal feature extraction to the model (see Figure 22). The image recognition-based modelling presented in this paper is not applicable to AS-GCN which is graphed based; however, the transfer learning methodologies are applicable. A transfer learning development on this project would be a *Fine-tuning and Layer Freezing* where the original AS-GCN pre-trained model is exported for use on MADS by replacing the final fully connected layer with the correct amount of classes, and then further training on MADS for fine tuning.

11. References

Li, M. et al., 2019. Symbiotic Graph Neural Networks for 3D Skeleton-based Human Action Recognition and Motion Prediction. *IEEE-TPAMI.*

ABDULAZEEM, Y., BALAHA, H. M., BAHGA, W. M. & BADAWY, M., 2021. Human Action Recognition Based on Transfer Learning Approach. *IEEE Access,* Volume 9.

Amazon Web Services, n.d. *Amazon EC2 G4 Instances.* [Online]
Available at: https://aws.amazon.com/ec2/instance-types/g4/
[Accessed May 2020].

Amazon Web Services, n.d. *Deep Learning AMI (Ubuntu 16.04).* [Online]
Available at: https://aws.amazon.com/marketplace/pp/Amazon-Web-Services-Deep-Learning-AMI-Ubuntu-1604/B077GCH38C
[Accessed May 2020].

Anon., n.d. *FFmpeg.* [Online]
Available at: https://www.ffmpeg.org/
[Accessed August 2021].

BILOGUR, A., 2018. *Full batch, mini-batch, and online learning.* [Online]
Available at: https://www.kaggle.com/residentmario/full-batch-mini-batch-and-online-learning
[Accessed 2021].

Cao, Z. et al., 2019. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

Facebook AI Research, 2021. *Datasets - Action Recognition.* [Online]
Available at: https://paperswithcode.com/datasets?task=action-recognition-in-videos
[Accessed August 2021].

Hidalgo, G. et al., n.d. *openpose.* [Online]
Available at: https://github.com/CMU-Perceptual-Computing-Lab/openpose
[Accessed May 2020].

Jamhoury, L., 2018. *Understanding Kinect V2 Joints and Coordinate System.* [Online]
Available at: https://lisajamhoury.medium.com/understanding-kinect-v2-joints-and-coordinate-system-4f4b90b9df16
[Accessed July 2021].

Jegham, I., Khalifa, A. B., Alouani, I. & Mahjoub, M. A., 2020. *Vision-based human action recognition: An overview and real world challenges.* [Online]
Available at: https://www.sciencedirect.com/science/article/pii/S174228761930283X
[Accessed May 2020].

Karpathy, A. et al., 2014. Large-scale Video Classification with Convolutional Neural Networks. *CVPR.*

Konushin, A., 2017. *Action classification with convolutional neural networks.* [Online]
Available at: https://www.coursera.org/learn/deep-learning-in-computer-vision/lecture/0R0ds/action-classification-with-convolutional-neural-networks
[Accessed May 2020].

Lambda Labs, Inc., 2021. *GPU Cloud - Pricing & Specifications.* [Online]
Available at: https://lambdalabs.com/service/gpu-cloud/pricing
[Accessed August 2021].

Li, M., 2021. *Maosen Li - About Me.* [Online]
Available at: https://limaosen0.github.io/about/
[Accessed August 2021].

Li, M. et al., 2019. *Actional-Structural Graph Convolutional Networks for Skeleton-based Action Recognition.* [Online]
Available at: https://arxiv.org/abs/1904.12659
[Accessed May 2020].

Li, M. et al., 2019. *The model architecture of AS-GCN (for human action recognition).* [Online]
Available at: https://github.com/limaosen0/AS-GCN
[Accessed May 2020].

Liu, J. et al., 2019. *NTU RGB+D 120: A Large-Scale Benchmark for 3D Human Activity Understanding.* [Online]
Available at: https://arxiv.org/pdf/1905.04757.pdf
[Accessed May 2020].

Liu, J. et al., n.d. *Action Recognition Datasets: "NTU RGB+D" Dataset and "NTU RGB+D 120" Dataset.* [Online]
Available at: http://rose1.ntu.edu.sg/datasets/actionrecognition.asp
[Accessed May 2020].

Michaus, M. P., 2017. *Visualizing Learning rate vs Batch size.* [Online]
Available at: https://miguel-data-sc.github.io/2017-11-05-first/
[Accessed 2021].

Multimedia Laboratory, CUHK, 2021. *MMSkeleton.* [Online]
Available at: https://github.com/open-mmlab/mmskeleton
[Accessed 2021].

Ni, H., n.d. *Human action recognition.* [Online]
Available at: https://www.turing.ac.uk/research/research-projects/human-action-recognition
[Accessed May 2020].

Sabater, A. et al., 2021. One-shot action recognition in challenging therapy scenarios. *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW).*

Shahroudy, A., 2019. *NTURGB-D/Matlab/.* [Online]
Available at: https://github.com/shahroudy/NTURGB-D/tree/master/Matlab
[Accessed 2021].

Shao, D., Zhao, Y., Dai, B. & Lin, D., 2020. Intra- and Inter-Action Understanding via Temporal Action Parsing. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2020.*

Soomro, K., Zamir, A. R. & Shah, M., 2012. UCF101: A Dataset of 101 Human Action Classes From Videos in The Wild. *CRCV-TR-12-01.*

Yan, S., Xiong, Y. & Lin, D., 2018. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. *AAAI.*

Zhang, W. et al., 2017. *Martial Arts, Dancing and Sports dataset: A challenging stereo and multi-view dataset for 3D human pose estimation.* [Online]
Available at: https://www.sciencedirect.com/science/article/abs/pii/S026288561730046X
[Accessed May 2020].

Zhang, W. et al., 2017. *Martial Arts, Dancing and Sports Dataset.* [Online]
Available at: http://visal.cs.cityu.edu.hk/research/mads/
[Accessed May 2020].