
Article:
Mohottala, S. orcid.org/0000-0002-6196-2161, Gawesha, A. orcid.org/0000-0001-8946-5629, Kasthurirathna, D. orcid.org/0000-0001-8820-9033 et al. (2 more authors) (2025) Spatio-temporal graph neural network based child action recognition using data-efficient methods: A systematic analysis. Computer Vision and Image Understanding, 259. 104410. ISSN 1077-3142

https://doi.org/10.1016/j.cviu.2025.104410

Computer Vision and Image Understanding
journal homepage: www.elsevier.com

Spatio-Temporal Graph Neural Network based Child Action Recognition using Data-Efficient Methods: A Systematic Analysis - Supplementary Material

Sanka Mohottala[a,b,**], Asiri Gawesha[a], Dhrashana Karshutrirathna[a], Charith Abhayaratne[d], Pradeepa Samarasinghe[c]

[a]*Department of Software Engineering, Faculty of Computing, Sri Lanka Institute of Information Technology, Sri Lanka*
[b]*Department of Electrical and Electronic Engineering, Faculty of Engineering, Sri Lanka Institute of Information Technology, Sri Lanka*
[c]*Department of Information Technology, Faculty of Computing, Sri Lanka Institute of Information Technology, Sri Lanka*
[d]*School of Electrical and Electronic Engineering, The University of Sheffield, United Kingdom*

ABSTRACT

This paper presents implementations on child activity recognition (CAR) using spatial-temporal graph neural network (ST-GNN)-based deep learning models with the skeleton modality. Prior implementations in this domain have predominantly utilized CNN, LSTM, and other methods, despite the superior performance potential of graph neural networks. To the best of our knowledge, this study is the first to use an ST-GNN model for child activity recognition employing both in-the-lab, in-the-wild, and in-the-deployment skeleton data. To overcome the challenges posed by small publicly available child action datasets, transfer learning methods such as feature extraction and fine-tuning were applied to enhance model performance. As a principal contribution, we developed an ST-GNN-based skeleton modality model that, despite using a relatively small child action dataset, achieved superior performance (94.81%) compared to implementations trained on a significantly larger (x10) adult action dataset (90.6%) for a similar subset of actions. With ST-GCN-based feature extraction and fine–tuning methods, accuracy improved by 10%-40% compared to vanilla implementations, achieving a maximum accuracy of 94.81%. Additionally, implementations with other ST-GNN models demonstrated further accuracy improvements of 15%-45% over the ST-GCN baseline. The results on activity datasets empirically demonstrate that class diversity, dataset size, and careful selection of pre-training datasets significantly enhance accuracy. In-the-wild and in-the-deployment implementations confirm the real-world applicability of above approaches, with the ST-GNN model achieving 11 FPS on streaming data. Finally, preliminary evidence on the impact of graph expressivity and graph rewiring on accuracy of small dataset-based models is provided, outlining potential directions for future research.

1. Initial Implementations

Initial implementations on reproducing the NTU-60 dataset based results with ST-GNN models as stated in the journal paper are given in Table 2. All implementation were done only with cross-subject protocol and Top-1 accuracy as used as the accuracy metric. Due to computation limitations, RA-GCN was only done with 2 streams. Original batch sizes and number of epochs could not be used with these due to computational limitations and that might explain the shortcomings of the reproduced results. TensorFlow based re-implementation of ST-GCN was done based on (Yan et al., 2018). However, the Top-1 accuracy achieved was lower than the reported accuracy. This could be attributed to differences in hyperparameters, which arose due to ambiguity in the paper and our computational limitations. In our subsequent ST-GCN implementations, dropout

**Corresponding author
e-mail: divandyasm@gmail.com (Sanka Mohottala)

and layer specific edge weighting (i.e.,learnable Mask) were not implemented since that could potentially result in low performance in transfer learning. With these changes, our ST-GCN accuracy was also improved as shown in Table 1.

Table 1: ST-GCN Implementation using NTU-60 Dataset

Implementation	Accuracy	
	Top-1	Top-5
ST-GCN original	81.5%	-
ST-GCN our imple.	75.16%	94.47%
ST-GCN w/o dropout+mask	78.19%	96.01%

While the ST-GCN implementation with NTU-120 dataset resulted in 69.04%, a comparison could not be done since implementation was not done based on a standard protocol and original ST-GCN (Yan et al., 2018) haven't used this dataset. This implementation was used to select the classes for the NTU-44 protocols.

Initial implementations were done with roto-translation invariance being introduced to the dataset as done by (Kofinas et al., 2021). This can be seen as introducing a strong prior into the model. This resulted in improved accuracy but only by a small margin. Implementations were done using NTU-44 dataset and NTU-60 datasets as in Table 3. Since resultant improvement was marginal, original pre-processing was used in all subsequent implementations.

Skeleton structural implementations as detailed in Jornal paper were done using the NTU-5 dataset. Since there are two candidate structures, four train-test skeleton combinations were used in four different implementations and results are given in Table 4. KinectV2 refers to the 25 joint structure and kinectV1 refers to the modified 20 joint structure. Results suggest that while the ST-GCN performs well even with structural changes done to both train-test sets, loss of information affect the model performance. But given that kinectV1-kinectV2 train-test combination performs better than kinectV2-kinectV1 train-test combination, we have used the modified structure in our subsequent transfer learning implementations.

Initial implementations based on kinetics-400 dataset were done to determine the optimal ST-GCN model configurations. ST-GCN (Yan et al., 2018) reported results for OpenPose based kinetics-400 (i.e., kinetics-skeleton) as well as a motion oriented 30 class subset from this dataset called kinetics-motion. Thus we attempted to reproduce the results with same datasets and the best results are given in Table 5. Since KS-KSS dataset share data with kinetics-skeleton dataset, data from those 8 classes were not used in the implementation. Original hyper-parameters were used where it was possible but some (e.g., batch size) were not used due to computational limitations. This and/or the differences in pre-processing may have lead to the differences between original and reproduced results with kinetics-{skeleton,motion} datasets.

Because of the unconstrained nature of the kinetics-400 videos, ST-GCN authors (Yan et al., 2018) introduced a new pre-processing stage that is different from the one used with constrained scenario based datasets such as NTU-60. Primary pre-processing stage contains skeleton normalization, skeleton centering and tracking of individuals in multi-person scenarios. Secondary pre-processing stage contains random frame selection from raw frames as well as camera motion simulation process. Since exact configurations of this pre-processing stage is not given in the paper, experiments with kinetics-motion subset were done to determine the optimal configuration. Result from these experiments (Table 6) suggest that combination of frame selection along with random skeleton movement (i.e., camera motion simulation) outperforms each individual implementations. In addition, a new sub-sampling approach with frame dropping was also implemented, but due to low performance it was not added to the final secondary pre-process stage. Based on these results, 150 frame window based combined approach was used in all future implementations.

Comparisons were done between different skeleton structures (Table 7) using KS-balance and KS-full datasets which were introduced in Journal paper. Feet related joints were removed from G_{body} resulting in $G_{body}*$ skeleton structure with 19 vertices (V_3). Considering the overall performance, G_{body} was used for the vanilla implementations while G_{coco} was used in

Table 2: ST-GNN implementations with NTU-60

	ST-GCN	2s-AGCN	MS-AAGCN	RA-GCN	ST-GAT
Original	81.5%	88.5%	90.0%	87.3%	92.8%
Reproduced	78.2%	86.1%	88.7%	84.4%	90.4%

Table 3: Pre-processing Implementations

Dataset	Implementation	Accuracy
NTU 44	Original	80.78%
	Roto-translational invariance	79.55%
	Roto-trainslational invariance*	81.31%
NTU 60	Original	78.70%
	Roto-translational invariance*	80.49%

Table 4: Skeleton structure selection for CWBG dataset

Test \ Train	KinectV2	KinectV1
KinectV2	94.16%	91.73%
KinectV1	89.32%	93.30%

Table 5: ST-GCN Implementations based on kinetics-400 dataset

Implementation	Accuracy	
	Top- 1	Top-5
kinetics-skeleton (Yan et al., 2018)	30.7%	52.8%
kinetics-skeleton [our imple.]	21.16%	41.7%
kinetics-motion (Yan et al., 2018)	72.4%	-
kinetics-motion [our imple.]	68.01%	-

Table 6: Secondary pre-processing configuration results

Implementation	Accuracy
Random frame selection [w = 150]	65.99%
Random skeleton movement	67.54%
Combined approach [w = 150]	68.01%
Combined approach [w = 128]	67.87%
Sub-sampling approach	49.06%
Without secondary pre-processing stage	63.17%

transfer learning implementations. As detailed in our previous paper (Mohottala et al., 2022), a modified graph structure was used instead of the default OpenPose-COCO graph structure.

Table 7: Skeleton Structure Selection for KS-KSS dataset

Skeleton structure	KS-balance	KS-Full
G_{body} , $\|V_2\| = 25$	69%	75%
G_{body}* , $\|V_3\| = 19$	67%	74%
G_{coco} , $\|V_1\| = 18$	66.45%	74%

2. Inter-class variation effect

For inter-class variation analysis, we use CWBG-D and CWBG-S protocols since they both contains similar number of classes, similar number of samples and balanced datasets. Each sample contains similar number of features where sample shape is (T, V, C) resulting in $T \cdot V \cdot C$ number of features. Based on the pre-processing done, T=300, V=20 and C=3 thus resulting in 18000 features in each sample skeleton sequence. We utilized t-SNE, a dimensionality reduction method, to visualize and analyze datasets residing in this high-dimensional feature space in a two-dimensional space. This technique preserves the pairwise similarities and dissimilarities between data points. The resulting visualizations are presented in Figure 1a and 1b. To analyse resultant model performance, we use the LOOCV evaluation method based model implementation and the resultant Top-1 accuracy and the confusion matrix which are derived considering all 30 implementations the samples from the corresponding CWBG protocol.

Considering the CWBG-D protocol, from Figure 1a, ”crouch”, ”draw flower”, ”fly like a bird” and ”hands up” classes can be identified as the ones with highest inter-class variation, as they exhibit lower overlap and greater distance

Table 8: Publicly available Child Datasets Summary

Name	Sensors	Subjects	Actions	Samples	Data	Year
Kinder-Gator(Aloba et al., 2018)	Kinect	10	58	580	RGB+D	2018
Kinder-Gator 2.0 (Dong et al., 2020)	MoCap	8	21	538	Depth	2020
CWBG (Vatavu, 2019)	Kinect	30	15	1313	Depth	2019
APSICA (Olalere et al., 2021)	RGB	NA	21	1592	RGB	2021
SSBD (Rajagopalan et al., 2013)	RGB	NA	3	75	RGB	2012
KS-KSS (Mohottala et al., 2022)	RGB	NA	8	2396	RGB	2022

from other classes.Additionally, lower intra-class variation can be observed in the ”crouch” and ”draw flower” classes due to their relatively tight clustering. Relatively better performance for these classes as evident from the confusion matrix Figure 1c corroborate these interpretations. On the other hand, higher overlap among samples in ”jump”, ”stand on one leg” and ”throw ball” result in lower inter-class variations and the low accuracy results from the confusion matrix corroborate the conclusion that low inter-class variation result in low accuracy. Considering the CWBG-S protocol, lower inter-class variation between the ”draw circle”, ”draw flower” and ”draw square” are quite apparent because of the overlapping of samples from these 3 classes. Corresponding LOOCV model output in confusion matrix Figure 1d corroborate that low inter-class variation resulting in low accuracy as well as misclassification among the overlapping classes.

3. Extended Age-wise and Gender-wise Implementations

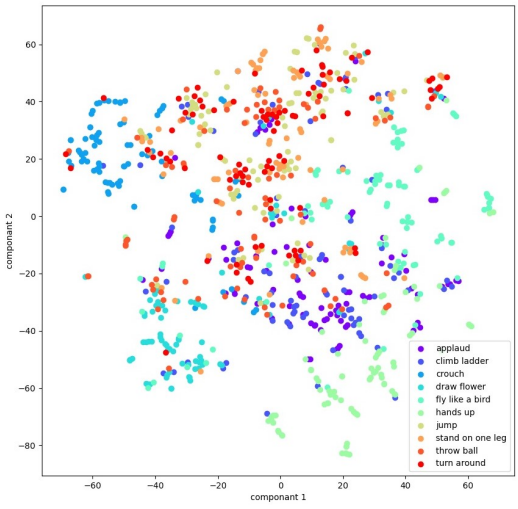
To further study the effect of age and gender of children on the ST-GNN model performance, we introduce several new protocols. For the gender factor, we introduces 4 protocols, one only containing boys in the training, another only containing girls and final one containing a combination of boys and girls.

- CWBG-G-B: Contains all the boys in training phase and all the girls in test phase.
- CWBG-G-G: Contains all the girls in training phase and all the boys in test phase.
- CWBG-G-M-1: Train set contains 8 boys and girls, with a near-uniform age distribution. test set contains rest of the 7 boys and 7 girls.
- CWBG-G-M-2: Train set from CWBG-G-M-1 is used as the test set and test set is used as the train set.

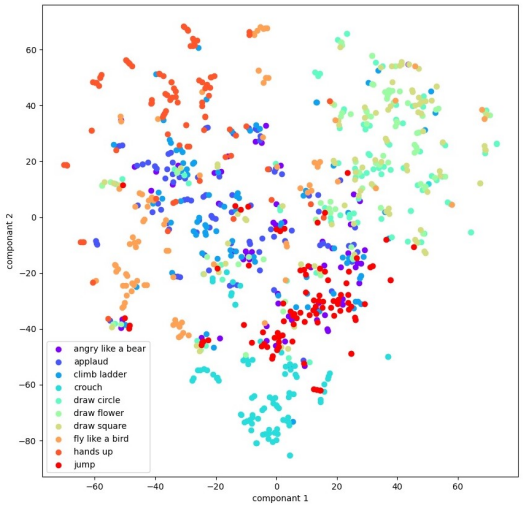
For the age factor, we introduce 5 protocols as given below.

- CWBG-A-3: Contains all the children (10) in Group 3 - in testing phase and all the other children in training phase.
- CWBG-A-4: Contains all the children (10) in Group 4 in testing phase and all the other children in training phase
- CWBG-A-5: Contains all the children (10) in Group 5 in testing phase and all the other children in training phase
- CWBG-A-M-1: Contains a mixture of children, 7 from Group 3 and 5 and 6 from Group 4, with near-uniform gender distribution in training phase and all others in test phase.
- CWBG-A-M-2: similar to CWBG-A-M-1 but children IDs are different.

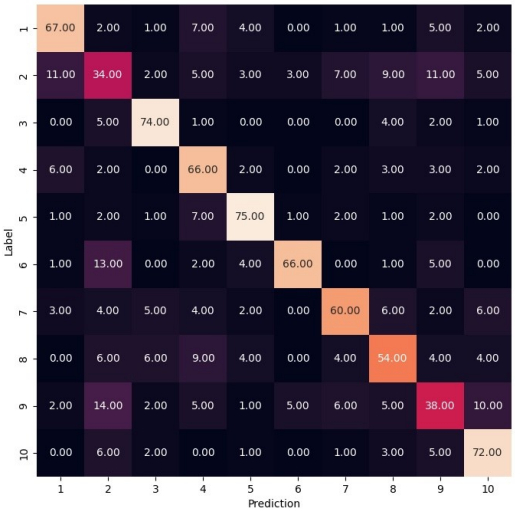
Implementations were done under CWBG-F and CWBG-D subset protocols and the results are shown in Table 9. Results from CWBG-A-M-1 and CWBG-A-M-2 were averaged since both are mixed age protocols and the results are given under ’Mixed’ protocol. When a higher age group is used in testing, a sharp increase in accuracy can be observed across both CWBG-F and CWBG-D protocols. This is to be expected since



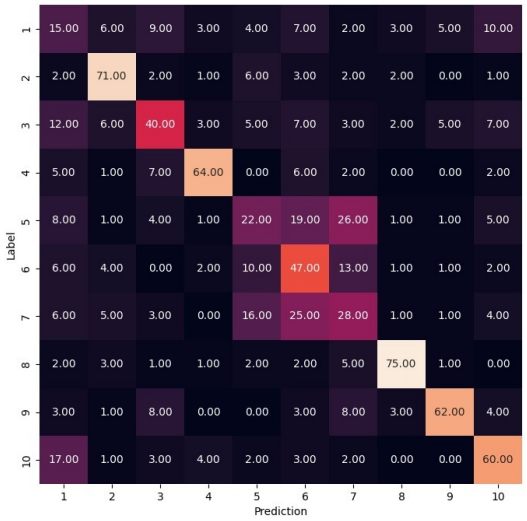
(a) t-SNE visualization of CWBG-Dissimilar dataset



(b) t-SNE visualization of CWBG-Similar dataset



(c) CWBG-Dissimilar confusion matrix



(d) CWBG-Similar confusion matrix

Fig. 1: ST-GCN input and output visualization

Table 9: Age-wise implementation results

Protocol	CWBG-A-3	CWBG-A-4	CWBG-A-5	Mixed
CWBG-F	37.49 ± 2.2	40.00 ± 2.7	42.76 ± 2.5	43.13 ± 5.4
CWBG-D	55.96 ± 3.1	56.54 ± 2.4	63.47 ± 2.6	65.33 ± 3.5

Table 10: Gender-wise implementation results

Protocol	Boys	Girls	Mixed
CWBG-F	36.39 ± 3.0	37.18 ± 3.1	41.68 ± 4.1
CWBG-D	56.43 ± 3.7	53.29 ± 3.0	59.04 ± 2.7

with lower intra-class variations in test set can result in better model performance due to smaller differences in test set as well as due to better distribution capturing by the higher intra-class variation present in training dataset. Best accuracy is achieved with 'Mixed' protocol as expected since it contains a similar distribution in both train and test sets and able to capture it well during the training phase.

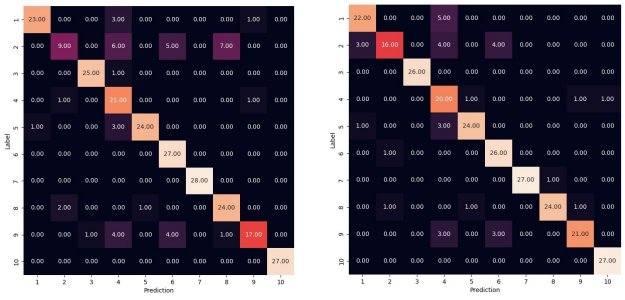
Implementations were also done on age factor as detailed in journal paper and the results are given in Table 10 under both CWBG-F and CWBG-D subset protocols. CWBG-G-B protocol based results are given under 'Boys' column and CWBG-G-G protocol based results are given under 'Girls' column. Since CWBG-G-M-1's test set is used as train set in CWBG-G-M-2 and vice versa, averaged results for both of these protocols were calculated and given under the 'Mixed' column. Across both CWBG-D protocol, 'Girls' test set performs better than 'Boys' test set but with CWBG-D protocol, the opposite is true. Thus these results are not conclusive enough to make a strong claim. When compared with 'Mixed' implementation, it's performance is superior to both Girls only and Boys only trained models, thus suggesting the previous conclusion that both train and test sets capture the distribution equally well.

Original CWBG paper (Vatavu, 2019) uses the dataset to study children's gesture preferences by introducing an algorithm called 'dissimilarity-consensus' to calculate the similarity-dissimilarity of actions they perform. In that they apply the algorithm to actions of each class independent of other action classes. Since this considers the similarity of actions within each class, this algorithm can be considered as a method to measure intra-class variation and the results from this study can be used as a quantitative measure of intra-class variation. An anlysis done in (Vatavu, 2019) under the same age groups gives conclusive results that increased age group result in decreased intra-class variation when all classes are considered together thus giving conclusive evidence for age being a factor in intra-class variation. Our results of higher performance with higher age group corroborate these findings conclusively and further showing that the lower performance of CAR model is influenced by the varying levels of motor skill development in children.

4. Transfer Leraning with ST-GCN for CAR

Number of samples (n) and number of features in each sample (p) in a dataset can have a significant effect on model performance. Generally, $n >>> p$ leads to well-fitted models but $n \approx p$ can lead to overfitting. In non-FRA source datasets in Table **??**, sample shape is $(3, 20, 300)$ thus $p = 18000$ while FRA source datasets result in $p = 6000$ due to down-sampling resulting in $(3, 20, 100)$. Down sampling was done to emulate a 10FPS frame rate in the source dataset because of the frame rate mismatch in CWBG (10FPS) and NTU (30FPS) datasets. While the increased $\frac{n}{p}$ positively contribute to the model performance, loss of information as a result of frame dropping contributed negatively. Net effect of this can be observed in Table **??** as a drop in accuracy between down-sampled and original dataset based implementations. Thus we can conclude there is considerable loss of information in FRA source datasets.

5. ST-GNN Implementations



(a) ST-GAT (b) MS-AAGCN

Fig. 2: CWBG-D classwise comparison

Both of these models misclassified data from the 'throw ball' and 'climb ladder' classes as 'draw flower' and 'hands up' in

a similar manner (figure 2). Additionally, the ST-GAT model further misclassified some data from 'climb ladder' class as belonging to the 'throw ball' class. In order to identify potential reasons for these shortcomings, misclassified skeleton sequences were visualized in a 3D space. While it is hard to exactly interpret the reasons behind misclassification as 'draw flower' class, it seems similarity in hand movement is a one reason. When considering the samples that were misclassified as 'hands up', the visualizations leads to better interpretations. Misclassified samples from 'climb ladder' class can be attributed to participants raising there hands with little to no movement in legs. Misclassified samples from 'throw ball' class can be attributed to the overhead throwing of the ball. These samples are shared in both models as well. ST-GAT misclassification of some 'climb ladder' samples as 'stand on one leg' can be attributed to lack of hand movements. Misclassification of 3 instances from 'fly like a bird' class as 'draw flower' is present in both of these models and confusion matrices from RA-GCN and 2s-AGCN also show the same behavior. Further analysis show that these 3 instances are from the same participant (ID: 28) and the visualization show that participant is doing a single-handed circular motion similar to drawing a imaginary circle rather than the two-handed steady movement that is present in other samples. Furthermore, comparison of ST-GAT and MS-AAGCN models probability distributions (Figure 3) suggest both of these models are over-confident models compared to other models such as 2s-AGCN.

Experiments done with NTU-60 and NTU-120 under cross-subject approach in (Song et al., 2021) show that RA-GCN model is robust to truncation, occlusion and jitter compared to other models such as ST-GCN and 2s-AGCN. Since the CWBG dataset was created in a constrained environment (i.e., lab environment), we assume there is no occlusion/truncation present. Visualization of CWBG skeleton sequences corroborate this assumption. While there is no apparent reason for presence of jitter in CWBG, analysis of noise in kinect sensors (Pagliari and Pinto, 2015) show that sensor noise is more pronounced in Kinect V1 than Kinect V2. Since participants performed action at a 3m distance from sensor, standard deviation of noise resulted can be assumed to be $\approx$ 0.015m. While this is much higher than the standard deviation of noise from a Kinect V2 sensor ($\approx$ 0.002m), resulted from jitter experiments with $\approx$ 0.05m standard deviation of jitter in (Song et al., 2021) suggest that 2s-AGCN is still robust to kinect v1 sensor noise just as it is to kinect v2 sensors. Thus the better performance of 2s-AGCN compared to RA-GCN with CWBG-F and CWBG-D can be attributed to the lack of presence of jitter along with occlusion and truncation. However, the slightly superior performance of RA-GCN compared to 2s-AGCN with CWBG-S and CWBG-Sh contradicts this conclusion, and it may be explained by the presence of jitter in the classes within these two subsets.

6. In-the-wild Implementations

Table 11: Transfer learning results for KS-KSS datasets

Dataset	Method	Full	Balance	Large	Small
KS	Propagation	84.3	83.38	86.03	76.92
KSS	Propagation	81.26	87.68	87.92	82.60
	Fine-Tuning	80.47	89.85	86.51	82.60
	Feature Extraction	79.15	89.13	87.92	86.95

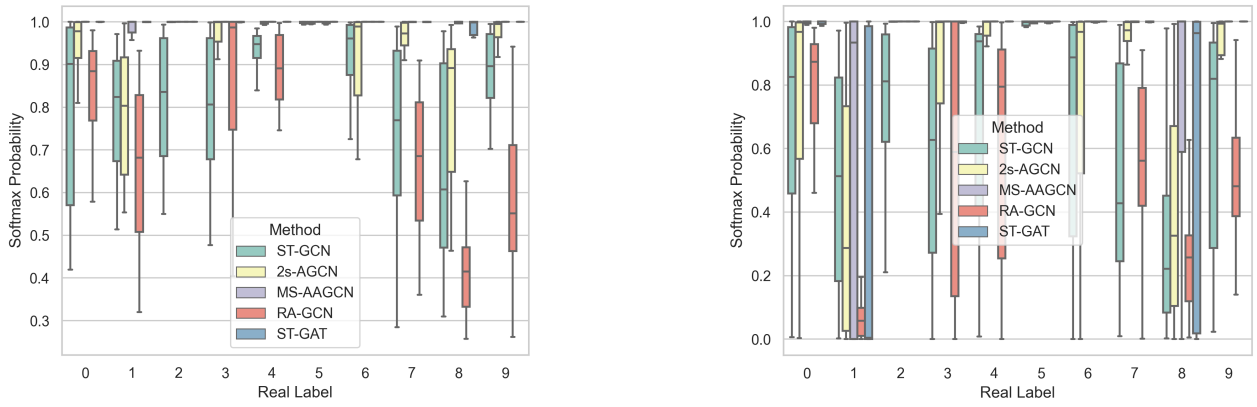
Table 12: Comparison of RGB modality and skeleton modality of KS-KSS dataset

	Modality	Full	Balance	Large	Small
KS	RGB	86.62	88.64	87.02	73.07
	Skeleton	84.3	83.38	86.03	76.92
KSS	RGB	82.57	86.23	79.72	78.26
	Skeleton	81.26	89.85	87.92	86.95

Table 13: Vanilla implementation results

Accuracy	Full	Large	Balance	Small
KS	75.29	77.83	69.32	69.23
KSS	60.68	64.88	59.43	86.95

When comparing the class-wise accuracy between KS/KSS-Balance protocols, performance of each class has increased,



(a) CWBG-D: All correctly classified samples

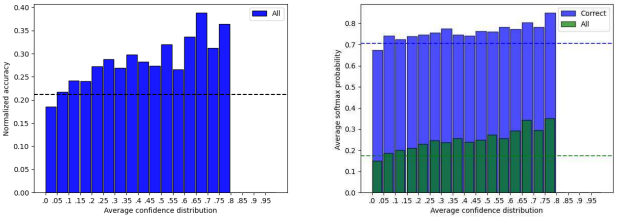
(b) CWBG-D: All samples

Fig. 3: ST-GNN performance with CWBG datasets

yet there is no relative improvement between classes.'Clapping' class perform the best while 'baseball throw' perform the worst. Result also suggest there is no strong connection between classwise confidence value and classwise accuracy given that 'climbing tree' class performs second best even though the average confidence value is the lowest and 'hopscotch' performs second worst even though the average confidence is the highest.

7. Accuracy vs Confidence Comparison

Figure 4 contains additional results for the second most active person.



(a) Normalized accuracy

(b) Softmax probability

Fig. 4: Comparison results for second most active person ($p = 1$)

8. In-the-Deployment Additional Results

In-the-deployment experiments, latency results were taken with respect to each module in the system pipeline along with the number of people in sliding window as detailed in the Section 3.6 of the main manuscript. Some of the results are given in the Results and Discussion (Section 5.9) including the Table 15 where average latency results are given in milliseconds. To add error bounds to these results, we have calculated the standard deviations for each of these results and are given in the Table 14. In this Table, each row label represent a specific module in the system module as in the main manuscript and each column label represent the number of people in a given sliding window.

9. Hyper-parameter tuning and Results

Hyper-parameter tuning was a major part of most of the implementations and detailed of different hyper-parameters are given in the Section 9.1. Final selected hyper-parameters for CWBG protocols and the performance difference between different hyper-parameters for the KS protocol are given in Section 9.2.

9.1. Hyper-parameter details

9.1.1. Base Optimizer Parameters

1. base lr: The initial learning rate used for training. Affects how much model weights are updated during each optimization step.

2. batch size: Number of samples processed before the model is updated. Batch size of 4 was the highest we could go due to storage limitations.

Time (ms)	1	2	3	4	5	6	7	8
inference_time	0.0624 ± 0.0046	0.0838 ± 0.0084	0.1040 ± 0.0092	0.1187 ± 0.0109	0.1223 ± 0.0134	0.1330 ± 0.0088	0.1759 ± 0.0187	0.1910 ± 0.0071
detection_time	0.0071 ± 0.0006	0.0072 ± 0.0007	0.0073 ± 0.0007	0.0071 ± 0.0005	0.0070 ± 0.0004	0.0069 ± 0.0005	0.0072 ± 0.0004	0.0080 ± 0.0000
pose_time	0.0256 ± 0.0028	0.0326 ± 0.0033	0.0384 ± 0.0035	0.0409 ± 0.0033	0.0448 ± 0.0031	0.0491 ± 0.0030	0.0595 ± 0.0036	0.0665 ± 0.0007
track_time	0.0004 ± 0.0006	0.0007 ± 0.0006	0.0008 ± 0.0006	0.0009 ± 0.0006	0.0010 ± 0.0007	0.0011 ± 0.0007	0.0015 ± 0.0006	0.0010 ± 0.0000
action_time	0.0153 ± 0.0016	0.0298 ± 0.0039	0.0429 ± 0.0064	0.0543 ± 0.0099	0.0551 ± 0.0125	0.0612 ± 0.0075	0.0914 ± 0.0174	0.0960 ± 0.0000

Table 14: Mean ± standard deviation of processing times for each metric vs. number of people in the frame

3. epochs: Total number of passes over the entire training dataset.

4. Optimizer: Optimization algorithm used to update model parameters; Stochastic Gradient Descent (SGD) and Adam were mainly explored.

5. SGD momentum: A momentum value used with SGD to accelerate convergence by smoothing the gradient updates.

6. SGD nesterov: Boolean indicating whether Nesterov momentum is used, a variation of standard momentum that anticipates future gradients.

9.1.2. Learning Rate Scheduler Parameters

7. learning rate scheduler: Defines how the learning rate is adjusted during training.
    - PiecewiseConstDecay: The learning rate stays constant for specified intervals and then drops.
    - ExponentialDecay: The learning rate decays exponentially over time.

8. steps: Epochs at which the learning rate changes in piecewise constant decay.

9. iterationNum: Total number of iterations used for one epoch of training. Possibly around the value obtained as dataset size divided by batch size.

10. values: List of learning rate values to be used at different training steps when using PiecewiseConstDecay.

11. stepdecay: Indicates whether step-based decay is applied. Boolean.

12. steps decay: Steps at which decay is applied in step-based decay. Only relevant if stepdecay is set to TRUE.

13. decay rate: Decay rate for exponential or step-based learning rate schedules.

9.1.3. Regularization Parameters

14. Weight decay: Type of weight regularization used. L2 corresponds to L2 regularization, which discourages large weights.

15. Weight decay value: Strength of the L2 penalty added to the loss function to prevent overfitting.

9.1.4. Weight Initialization Parameters

16. Weight initializer (WI): Method used to initialize the weights of the model. VarianceScaling adjusts the scale based on the number of input or output units.

17. WI scale: A scale factor applied during weight initialization. Might control the variance level of the initial weights

18. WI mode: Mode for variance scaling. 'Fan out' scales weights based on the number of output units (neurons) in a layer.

19. WI distribution: Distribution used to sample initial weights. 'Truncated normal' means values are drawn from a normal distribution, but any value more than two standard deviations from the mean is discarded and redrawn.

9.2. *Hyper-parameters and the model performance*

The first part of this section presents the selected hyper-parameters for the CWBG protocols. The second part details the various hyper-parameter configurations used with the ST-GCN model in the in-the-wild experiments, along with a comparative analysis of the model's performance across these configurations.

In the first part, for the cross-subject protocol, final selected hyper-parameters were given in the main manuscript (Section 5.3) under all four CWBG protocols. For other three protocols,selected hyper-parameters are given in the Tables 15 and 16. Performance of the ST-GCN model on these protocols are given in the Table 3 of the main manuscript.

In the second part, for the in-the-wild implementations, KS-X protocols were used. Details of these protocols are given in Section 3.5 of the main manuscript. Hyper-parameter details of finally selected hyper-parameters as well as some of the other sub-optimal hyper-parameters are given in Tables 17, 18, 19 and 20.

Model performance for these combination of hyper-parameters are also given in the subsequent Figures. For KS-Full protocol, ST-GCN performance for selected hyper-parameters are given in Figure 5. Performance for other hyper-parameters are given in Figures 6 and 7. Model performance of KS-Large protocol for the selected hyper-parameters is given in Figure 8. Similarly, for KS-Balanced protocol,model performance for selected hyper-parameters is given in Figure 9 while for other hyper-parameter combinations, model performance is given in Figures 10 and 11. For KS-Small protocol, model performance for the selected hyper-parameters is given in Figure 12 while Figures 13 and 14 contains the model performance for other hyper-parameter sets.

Table 15: Final hyper-parameter values for Random Hyper parameters

hyperparam	full	dissimilar	similar	shared
base lr	0.01	0.01	0.01	0.01
batch size	4	4	4	4
epochs	30	30	30	30
Optimizer	SGD	SGD	SGD	SGD
opt-SGD-moment	0.9	0.9	0.9	0.9
opt-SGD-nestrov	TRUE	TRUE	TRUE	TRUE
learning rate schedular	PiecewiseConstantDecay	PiecewiseConstantDecay	PiecewiseConstantDecay	PiecewiseConstantDecay
steps	[10,20]	[10,20]	[10,20]	[10,20]
iterationNum	916	628	628	312
constantBaseLearnigRate	FALSE	FALSE	FALSE	FALSE
values	[0.01,0.001,0.0001]	[0.01,0.001,0.0001]	[0.01,0.001,0.0001]	[0.01,0.001,0.0001]
stepdecay				
steps_decay				
decay-rate				
regularizer	l2	l2	l2	l2
regularizer val	0.0001	0.0001	0.0001	0.0001
weight initilizer	VarianceScaling	VarianceScaling	VarianceScaling	VarianceScaling
scale,mode,distribution	2,fan_out,truncated_normal	2,fan_out,truncated_normal	2,fan_out,truncated_normal	2,fan_out,truncated_normal
learningMode	'from_scratch'	'from_scratch'	'from_scratch'	from_scratch
pretrainedClasses	0	0	0	0
childDatasetClasses	15	10	10	5
protocolTraining	random_full	random_dissimilar	random_similar	random_shared
FPS	30	30	30	30
save freq	2	2	2	2
acc	44.82	66.22	53.1	80.74
mean time	0.007880198	0.007766723	0.007908626	0.008125316
std time	0.000901624	0.000807083	0.00090555	0.001110671

Table 16: Final hyper-parameter values for LOOCV Hyper parameters

hyperparam	full	dissimilar	similar	shared
base lr	0.01	0.01	0.01	0.01
batch size	4	4	4	4
epochs	30	30	30	30
Optimizer	SGD	SGD	SGD	SGD
opt-SGD-moment	0.9	0.9	0.9	0.9
opt-SGD-nestrov	TRUE	TRUE	TRUE	TRUE
learning rate schedular	ExponentialDecay	ExponentialDecay	ExponentialDecay	ExponentialDecay
steps	[10,20]	[10,20]	[10,20]	[10,20]
iterationNum	1268	868	840	432
constantBaseLearnigRate	FALSE	FALSE	FALSE	FALSE
values	[0.01,0.001,0.0001]	[0.01,0.001,0.0001]	[0.01,0.001,0.0001]	[0.01,0.001,0.0001]
stepdecay	TRUE	TRUE	TRUE	TRUE
steps_decay	1268//4	868//4	840//4	432//4
decay-rate	0.96	0.96	0.96	0.96
regularizer	l2	l2	l2	l2
regularizer val	0.0001	0.0001	0.0001	0.0001
weight initilizer	VarianceScaling	VarianceScaling	VarianceScaling	VarianceScaling
scale,mode,distribution	2,fan_out,truncated_normal	2,fan_out,truncated_normal	2,fan_out,truncated_normal	2,fan_out,truncated_normal
learningMode	'person_{child+1}'	'person_{child+1}'	'person_{child+1}'	'person_{child+1}'
pretrainedClasses	0	0	0	0
childDatasetClasses	15	10	10	5
protocolTraining	loocv_new	loocv_diss	loocv_sim	loocv_shared
FPS	30	30	30	30
save freq	2	2	2	2
acc	48.93	67.53	55.79	81.13
mean time	0.00835618	0.008330888	0.010205285	0.012963511
std time	0.002472371	0.002750763	0.004082358	0.007184704

Table 17: KS-ALL Hyperparameters

hyperparam	Journal added	other 1	other 2
base lr	0.001	0.001	0.001
batch size	4	4	4
epochs	50	50	30
Optimizer	SGD	SGD	SGD
opt-SGD-moment	0.9	0.9	0.9
opt-SGD-nestrov	TRUE	TRUE	TRUE
learning rate schedular	PiecewiseConstantDecay	PiecewiseConstantDecay	PiecewiseConstantDecay
steps	[20,40]	[20,40]	[20,40]
iterationNum	1716	1792	1792
constantBaseLearnigRate	FALSE	FALSE	FALSE
stepdecay			
steps_decay			
decay-rate			
regularizer	l2	l2	l2
regularizer val	0.0001	0.0001	0.0001
weight initilizer	VarianceScaling	VarianceScaling	VarianceScaling
scale,mode,distribution	2,fan_out,truncated_normal	2,fan_out,truncated_normal	2,fan_out,truncated_normal
learningMode	'From_Scratch'	From_Scratch	From_Scratch
pretrainedClasses	0	0	0
childDatasetClasses	8	8	8
protocolTraining	BODY_25_child_8	BODY_25_child_8	BODY_25_child_8
FPS	30	30	30
save freq	5	5	5
acc	75.29	74.12	74.12

Table 18: KS-Large Hyperparameters

hyperparam	Journal added
base lr	0.001
batch size	4
epochs	50
Optimizer	SGD
opt-SGD-moment	0.9
opt-SGD-nestrov	TRUE
learning rate schedular	PiecewiseConstantDecay
steps	[20,40]
iterationNum	1716
constantBaseLearnigRate	FALSE
stepdecay	
steps_decay	
decay-rate	
regularizer	l2
regularizer val	0.0001
weight initilizer	VarianceScaling
scale,mode,distribution	2,fan_out,truncated_normal
learningMode	'From_Scratch'
pretrainedClasses	0
childDatasetClasses	5
protocolTraining	BODY_25_child_5
FPS	30
save freq	5
acc	77.83

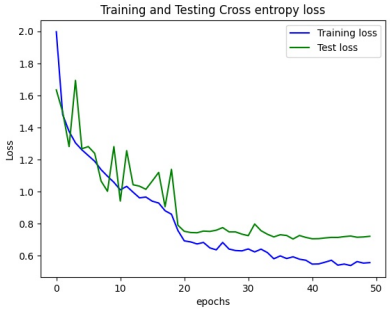
Table 19: KS-Balanced Hyperparameters

hyperparam	Journal added	other 1	other 2
base lr	0.001	0.001	0.001
batch size	4	4	4
epochs	50	50	50
Optimizer	SGD	SGD	SGD
opt-SGD-moment	0.9	0.9	0.9
opt-SGD-nestrov	TRUE	TRUE	TRUE
learning rate schedular	PiecewiseConstantDecay	PiecewiseConstantDecay	PiecewiseConstantDecay
steps	[20,40]	[20,40]	[20,40]
iterationNum	936	936	936
constantBaseLearnigRate	FALSE	FALSE	FALSE
stepdecay			
steps_decay			
decay-rate			
regularizer	l2	l2	l2
regularizer val	0.0001	0.0001	0.0001
weight initilizer	VarianceScaling	VarianceScaling	VarianceScaling
scale,mode,distribution	2,fan_out,truncated_normal	2,fan_out,truncated_normal	2,fan_out,truncated_normal
learningMode	'From_Scratch'	From_Scratch	From_Scratch
pretrainedClasses	0	0	0
childDatasetClasses	5	5	5
protocolTraining	BODY_25_child_5_balanced	BODY_25_child_5_balanced	BODY_25_child_5_balanced
FPS	30	30	30
save freq	5	5	5
acc	69.32	67.41	66.45

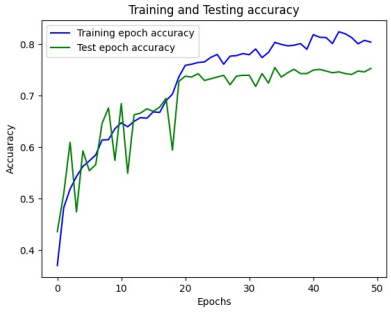
Table 20: KS-Small Hyperparameters

hyperparam	Journal added	other 1	other 2
base lr	0.001	0.001	0.001
batch size	4	4	4
epochs	50	50	30
Optimizer	SGD	SGD	SGD
opt-SGD-moment	0.9	0.9	0.9
opt-SGD-nestrov	TRUE	TRUE	TRUE
learning rate schedular	PiecewiseConstantDecay	PiecewiseConstantDecay	PiecewiseConstantDecay
steps	[15,25]	[20,40]	[15,25]
iterationNum	76	76	76
constantBaseLearnigRate	FALSE	FALSE	FALSE
stepdecay			
steps_decay			
decay-rate			
regularizer	l2	l2	l2
regularizer val	0.0001	0.0001	0.0001
weight initilizer	VarianceScaling	VarianceScaling	VarianceScaling
scale,mode,distribution	2,fan_out,truncated_normal	2,fan_out,truncated_normal	2,fan_out,truncated_normal
learningMode	'From_Scratch'	From_Scratch	From_Scratch
pretrainedClasses	0	0	0
childDatasetClasses	3	3	3
protocolTraining	BODY_25_child_3	BODY_25_child_3	BODY_25_child_3
FPS	30	30	30
save freq	5	5	5
acc	69.23	57.69	53.84

(a) Confusion Matrix

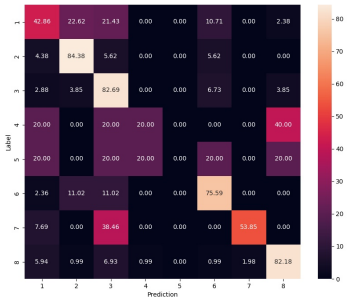


(b) Cross-Entropy Loss

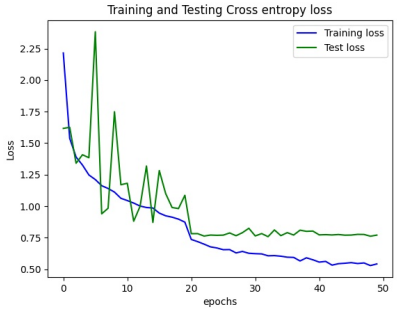


(c) Train Test Plot

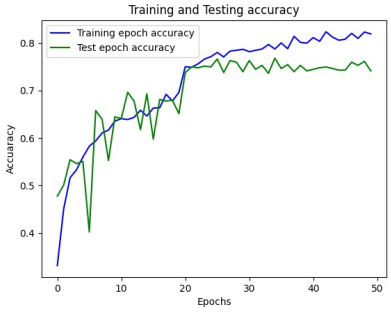
Fig. 5: Journal added- KS Full



(a) Confusion Matrix

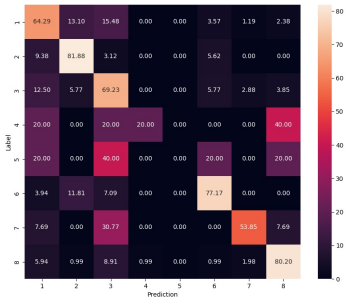


(b) Cross-Entropy Loss

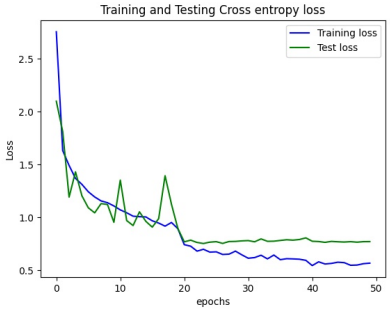


(c) Train Test Plot

Fig. 6: Other 1 - KS Full



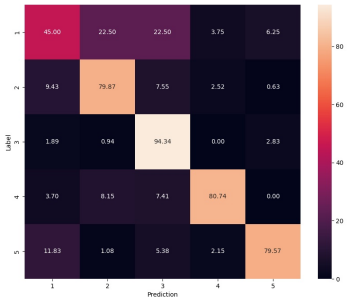
(a) Confusion Matrix



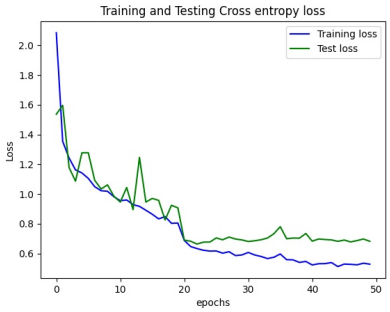
(b) Cross-Entropy Loss

(c) Train Test Plot

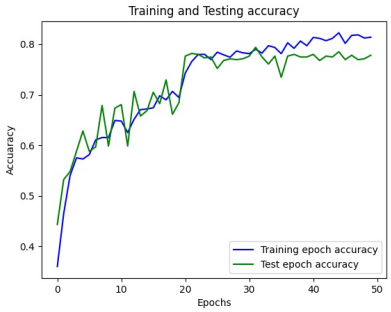
Fig. 7: Other 2- KS Full



(a) Confusion Matrix

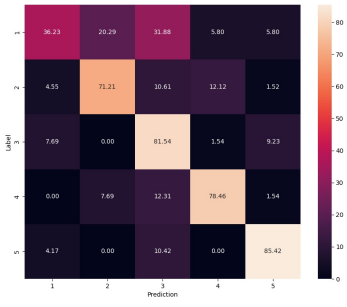


(b) Cross-Entropy Loss

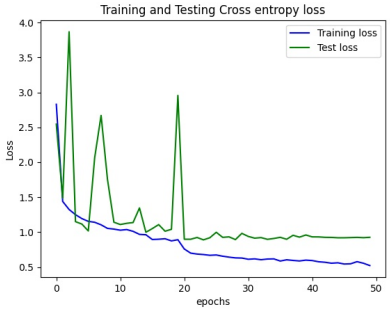


(c) Train Test Plot

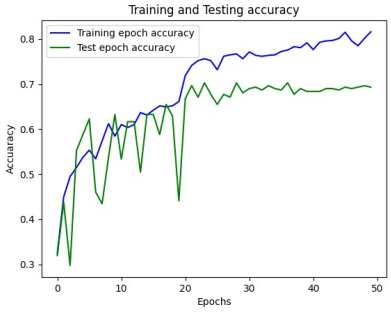
Fig. 8: Journal added- KS Large



(a) Confusion Matrix

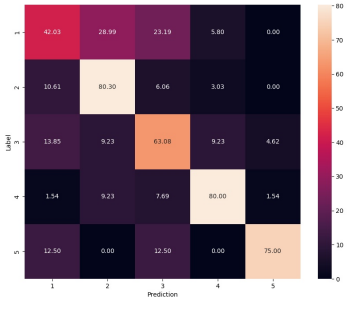


(b) Cross-Entropy Loss

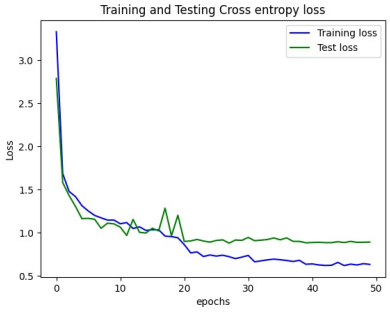


(c) Train Test Plot

Fig. 9: Journal added- KS Balanced



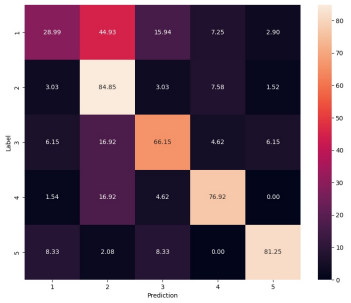
(a) Confusion Matrix



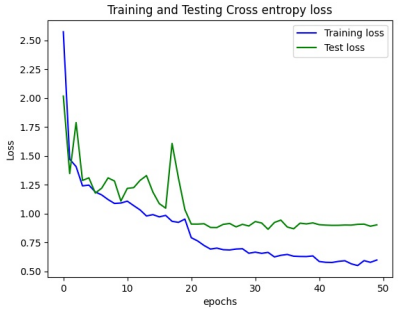
(b) Cross-Entropy Loss

(c) Train Test Plot

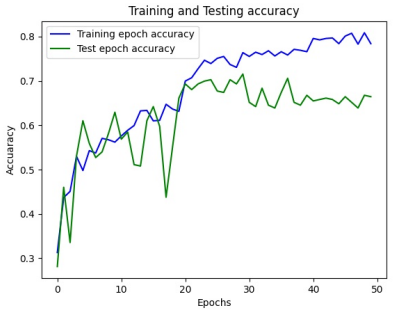
Fig. 10: Other 1 - KS Balanced



(a) Confusion Matrix



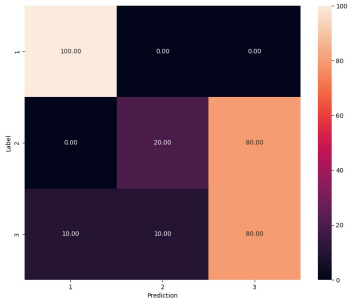
(b) Cross-Entropy Loss



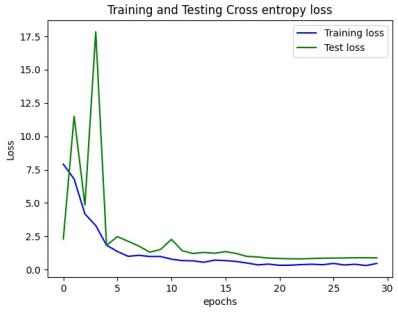
(c) Train Test Plot

c

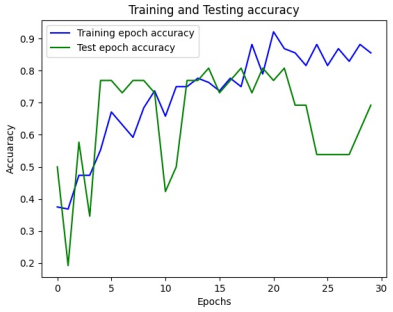
Fig. 11: Other 2 - KS Balanced



(a) Confusion Matrix

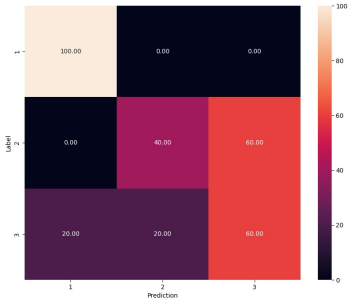


(b) Cross-Entropy Loss

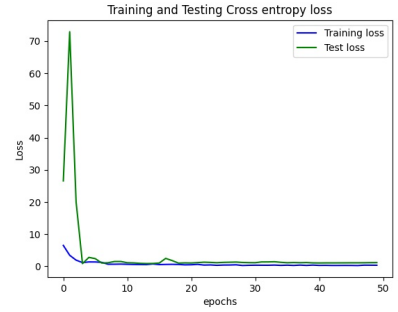


(c) Train Test Plot

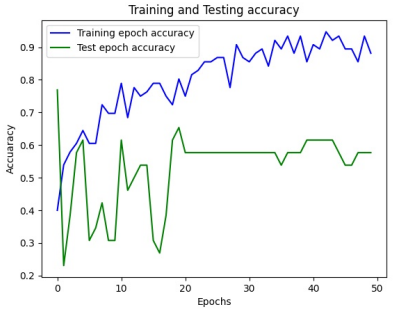
Fig. 12: Journal added - KS Small



(a) Confusion Matrix

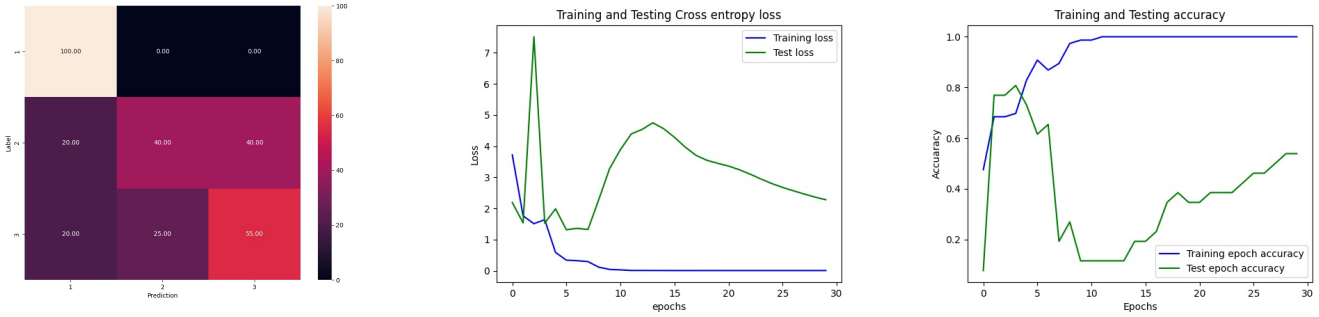


(b) Cross-Entropy Loss



(c) Train Test Plot

Fig. 13: Other 1 - KS Small



(a) Confusion Matrix

(b) Cross-Entropy Loss

(c) Train Test Plot

Fig. 14: Other 2 - KS Small

References

Aloba, A., Flores, G., Woodward, J., Shaw, A., Castonguay, A., Cuba, I., Dong, Y., Jain, E., Anthony, L., 2018. Kinder-Gator: The UF Kinect Database of Child and Adult Motion, in: Diamanti, O., Vaxman, A. (Eds.), EG 2018 - Short Papers, The Eurographics Association. doi:10.2312/egs.20181033.

Dong, Y., Aristidou, A., Shamir, A., Mahler, M., Jain, E., 2020. Kinder-Gator 2.0, Optical motion capture, Dataset, MIG2020. URL: https://doi.org/10.5281/zenodo.4079507, doi:10.5281/zenodo.4079507.

Kofinas, M., Nagaraja, N., Gavves, E., 2021. Roto-translated local coordinate frames for interacting dynamical systems. Advances in Neural Information Processing Systems 34, 6417–6429.

Mohottala, S., Abeygunawardana, S., Samarasinghe, P., Kasthurirathna, D., Abhayaratne, C., 2022. 2D Pose Estimation based Child Action Recognition, in: TENCON 2022 - 2022 IEEE Region 10 Conference (TENCON), IEEE, Hong Kong, Hong Kong. pp. 1–7. URL: https://ieeexplore.ieee.org/document/9977799/, doi:10.1109/TENCON55691.2022.9977799.

Olalere, F., Brouwers, V., Doyran, M., Poppe, R., Salah, A.A., 2021. Video-based sports activity recognition for children, in: 2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), IEEE. pp. 1563–1570.

Pagliari, D., Pinto, L., 2015. Calibration of kinect for xbox one and comparison between the two generations of microsoft sensors. Sensors 15, 27569–27589.

Rajagopalan, S., Dhall, A., Goecke, R., 2013. Self-stimulatory behaviours in the wild for autism diagnosis, in: Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 755–761.

Song, Y.F., Zhang, Z., Shan, C., Wang, L., 2021. Richly Activated Graph Convolutional Network for Robust Skeleton-based Action Recognition. IEEE Transactions on Circuits and Systems for Video Technology 31, 1915–1925. URL: http://arxiv.org/abs/2008.03791, doi:10.1109/TCSVT.2020.3015051. arXiv:2008.03791 [cs].

Vatavu, R.D., 2019. The dissimilarity-consensus approach to agreement analysis in gesture elicitation studies, in: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, Association for Computing Machinery, New York, NY, USA. p. 1–13. URL: https://doi.org/10.1145/3290605.3300454, doi:10.1145/3290605.3300454.

Yan, S., Xiong, Y., Lin, D., 2018. Spatial temporal graph convolutional networks for skeleton-based action recognition, in: Thirty-second AAAI conference on artificial intelligence.