TeleoWatch: Pose-Transformer-Based Advanced Action Recognition

Hanno Jacobs[1] [0009−0002−3251−7832] and Thambo Nyathi[1] [0000−0003−4676−6063]

Department of Computer Science, University of Pretoria, Pretoria, South Africa
https://www.up.ac.za/

Abstract. Recognition of human actions from videos is a valuable application for building management, security systems, accident prevention, accident intervention and several other applications. This study provides a framework for joint-based action recognition for human and non-human action recognition and common use cases for advanced intelligent closed-circuit television management systems. This study offers a scaling and normalization algorithm that makes the model position agnostic for the person's position in the frame and allows for generalization for actions performed facing the camera at various angles. It also improves on previous pose-based action recognition systems (Bidirectional-LSTM) by using a transformer encoder architecture for the recognition task. The transformer encoder architecture uses the joint locations output from pose estimation computer vision models to train the model to recognise the common joint trajectories that constitute human actions. The proposed transformer encoder model provides better performance than bidirectional-LSTM models in both speed and accuracy and provides better generalisation to novel tasks on benchmark action recognition datasets such as the KTH dataset and the UR-Fall dataset.

Keywords: Computer Vision · Machine Learning · Action Recognition · Transformer · Bidirectional-LSTM · CCTV · Pose-Detection

1 Introduction

Human action recognition is valuable in building management, security systems, accident prevention, and intervention. This is essential for applications where recognizing human actions in real time can capture significant value for customers. The ability to recognise actions using purely vision-based approaches is essential for monitoring actions in the closed-circuit television (CCTV), context where unknown people perform actions in public spaces. Previous studies in computer vision-based approaches for action recognition largely fall into two categories: First, purely single-frame image-based object detection models [23] that have been trained to recognise a particular state in an action as valuable and second, video-based methods that take a certain number of frames in a rolling window to detect the temporal context of an action. A good example of single-frame-based methods would be fall detection which is based on categorising all people in a frame into either the state of "standing" or "fallen" (and sometimes even "falling" in between the two). This approach often performs worse than recurrent-based models that have access to the temporal information which is lost in single-frame methods [14][1][17]. Some of the reasons for these incorrect detections are primarily because the temporal context of why the person is in the "falling" or "fallen" state is lost. Therefore, someone who is lying down on a couch or sitting on the floor is often automatically categorised as "fallen" when the context of a few frames could reveal the truth. Another problem with this approach is that a purely Convolutional Neural Network (CNN) vision-based approach struggles to generalise to different people with different appearances in either the "standing" or "fallen" positions [16]. Another problem is that a person is essentially the same "object" for CNN networks whether they are standing or lying down. Therefore class categorisation mistakes are prevalent.

The problem of generalization to people in different positions is somewhat addressed in methods that use pose models to extract the skeletal joint locations of people in the different positions [14][1] and then train another machine learning model to recognise the relative joint locations that constitute fallen or standing. This approach can generalize better than the purely CNN-based approach but still suffers from the lack of context which is inherent in recognising an action from a single frame that inherently has no temporal context.

Other commonly used methods are dynamic image networks [2], optical flow (moving images) [22] or texture descriptors [10] that blend multiple images into one image and then perform object detection on the blended image. This method allows for temporal context that allows for better recognition of the context of the action. However, the image then still suffers from the problem of CNN generalization to different people with different appearances performing the actions.

Current state-of-the-art methods for video action recognition rely on the use of pose-based computer vision models to extract the skeletal joints of a person performing a given action and feed them into a neural network that has a form of recurrence (LSTM/RNN/GRU) [4][20][15] or attention (Transformer) [6]. One of the tasks of this paper is to compare the performance of the recurrent Bi-LSTM [4] to the attention-based transformer method and evaluate their ability to generalize when adding new features (such as bounding box speed). We compare the two approaches' robustness for generalizing in CCTV environments where camera frame rates often vary due to dropped frames or varying camera frame rates and finally, we also evaluate our Transformer model on fall detection after training on our own custom dataset that contains falls and four other adversarial classes.

The goals of our action recognition system are: First, be able to recognise arbitrary actions that can generalize to work for humans and non-humans. Second, be able to generalize to work and recognize actions performed from a variety of different camera angles and people facing different directions (away from the camera, towards the camera and all the angles in between). Third, performs better than another state-of-the-art Bi-LSTM model. Fourth, ensure that the action recognition model works for fall detection such that it can work on unseen fall datasets and fifth, be able to work well for CCTV-specific situations and deployments (AI-on-camera deployment, multi-person batching, and be MLOps [12] pipeline friendly).

Human action recognition use-cases: Fall detection as well as identification of real vs fake falls, running detection (useful for indoors), aggression detection, ensuring factory tasks such as assembly of products are done correctly, ensuring that cleaning and janitorial tasks are done correctly, identifying poor lifting posture for manual labourers, recognition of exercises done in gyms for workout tracking, recognition of various actions taken on sports fields for automated statistics capture, recognition of poor throw/lift/run technique on sports field for team management and injury assessments, recognition of anomalous actions in public spaces, recognition of poor hand washing technique for surgeons or other high stakes environments, patient getting into or out of bed incorrectly, human-re-identification through gait.

Non-human action recognition: The modular approach of our approach which uses an easily retrainable pose detection model such as the YOLOv8 pose allows for this action recognition model to be used to identify non-human actions if the pose model is trained to place key points on non-human objects. This approach can be used to perform action recognition on any non-human object that can either be manipulated in space by an agent or on the non-human object itself as long as it has "joints" that can perform relative motion with other parts of itself. Some examples of use cases for non-human action recognition are recognition of different robot arm motions in a factory (recognition of faulty motion), recognition of non-round ball motion (tumbling vs spiralling), recognition of boom-gate opening and closing for facilities management, recognition of animal-actions for conservation.

2 Related Work

2.1 Action recognition for intelligent CCTV systems

Intelligent CCTV systems are becoming ubiquitous in the area of building and facilities management and safety and security systems. The area of intelligent CCTV systems incorporating action recognition systems that go far beyond basic action recognition is necessary to enable insights into facility usage, security and accident prevention/response. An action recognition model that can be trained on any arbitrary set of actions and then deployed on a large scale for several edge devices is something that is all but unknown in the industry. The goal of this work is to create a model that can be trained on an arbitrary set of actions, and deployed at a large scale on several cameras with different camera angles, lighting conditions and volumes of people walking through the view of the camera. A successful action recognition model should be created with parallelization and batching strategies in mind, given that a large number of people should be able to be analysed at once. With this in mind, speed, accuracy, and model sizes must be able to work on varying image sizes, camera angles, camera frame rates, edge devices and lighting conditions. The ideal model should be able to be retrained on labelled false-positive and false-negative data to aid the improvement that MLOps [12] processes pipelines provide in optimizing models for the particular cameras that they are run on. To aid in the MLOps process there should be the ability to adjust model sizes for deployment on low-powered edge devices as well as the ability to scale up the model architecture to build a large model that can run on a powerful GPU cluster to be used to relabel false-positive detections for adding to the training dataset (as proposed in [26]). Therefore, models like Transformers that can be more easily scaled up to create large models like those used in large language models are better for the task than recurrent models like the LSTM which often suffer from the vanishing or exploding gradient problem for long sequences. Transformers further benefit from being more easily parallelizable than LSTMs or other recurrent networks due to the lack of recurrence which is intrinsically harder to parallelize than Transformers.

2.2 Fall detection in public spaces

The purpose of fall detection systems is to automatically identify falls, prompt appropriate responses, facilitate quicker medical attention and reduce the delay in accessing help after a fall. Traditional monitoring methods often depend on manual intervention such as emergency buttons [9], which may not be readily available after a fall, or wearable sensors such as watches that are required to be worn by each person for whom falls are wished to be detected. These methods are not useful when posed in the context of CCTV or building management systems. The ideal fall detection system for CCTV-based systems should be able to recognise falls, in public spaces, and have a near-zero false-negative as well as a low false positive rate. The vision-based solution allows for falls to be detected in public spaces, even in persons who are not expected to be fall risks.

2.3 LSTM action recognition model

Previous work in pose-based action recognition done by Ramirez et al. [20] illustrates how a successful application of skeleton-based action recognition can be performed to recognise different actions, including falls, based on the temporal positions of human joints. This paper provides the state-of-the-art for fall detection for joint locations fed into a recurrent network (LSTM). The work by Chen et al. [4] shows how an attention-guided Bi-LSTM can detect actions without using the skeleton points. Current state of the art methods use pose-skeleton LSTM/Bi-LSTM models for recognition tasks (such as fall detection) as shown by Inturi et al. [8]. Therefore this will be the benchmark that our Transformer model should beat. Therefore the rest of our work will compare Bi-LSTM model to our proposed Transformer model.

The Bidirectional-LSTM action recognition model enables the recurrence relationship of the joints to be captured for the actions to be recognised. The bi-directionality of the model allows the model to learn recurrence relationships in both the forward and reverse directions. The problem with using a recurrence model such as an LSTM is the common vanishing [7] and exploding gradient problem [18] that hampers the model's ability to generalize for both long and short sequences of data with the same model architecture (number of layers and layer sizes). This problem is avoided by using the attention-based Transformer model. The LSTM model's sensitivity to sequence lengths means it has tremendous trouble generalising to actions performed over varying frames per second with the same models. This makes it unsuitable for most CCTV corner cases where frame rates can vary due to dropped frames and different default camera frame rates that are common for large buildings with cameras that are often supplied by several different vendors.

2.4 Transformers in action recognition

The Transformer model makes use of the multi-head attention mechanism that is introduced by Vaswani et al. [25]. Previous work for action recognition using Transformer models [6] [19] [11] demonstrate that Transformers show promising results for action recognition when combined with temporal joint information and the powerful attention mechanism. The usefulness of Transformers in action recognition tasks for CCTV systems extends beyond the promised accuracy and speed increases suggested [6] [19] [11]. The Transformer-based implementation by Do and Kim [6] relies on grouping human joints for optimization which results in high accuracy on human action recognition at the cost of generalizability to non-human action recognition tasks. The transformer architecture with its self-attention mechanism can tend to different sequence lengths which provide promise for CCTV systems which commonly drop frames and have varying frame rates for different cameras that are deployed on the same site.

3 Proposed Approach

3.1 Pose detection model

The proposed pose detection model that is used to extract key points (joints) is a YOLOv8x-pose model [24] which is among the state of the art for pose detection models. The model returns a bounding box along with coordinates for the following key points (referred to as joints) on the people in the frame: nose, eyes, ears, shoulders, elbows, wrists, hips, knees and ankles. These key points are fed into the Transformer model one frame at a time after they are scaled and normalized to remove positional information. This model can be swapped out for any other pose model that returns key points.

3.2 Scaling and Normalization algorithm

An important pre-processing step that is imposed on the key points generated by the pose detection model is the scaling and normalization of the data. This is important seeing that the data should be represented in as uniform a format as possible given the model will try to learn the underlying patterns and trends in the pose estimation key points. The method taken to ensure that the data for actions is represented constantly is described in the scaling and normalization algorithm in algorithm 1. This algorithm and pre-processing step are essential for making sure that the location of the person in the frame and the relative orientation of the person acting are not taken into account to aid in the generalization of the model over the same action done in other places in the frame and also being done in other directions concerning the camera angle (front on vs facing away from the camera and all the other angles in between). This scaling and normalization algorithm is designed to be pose model agnostic and work on non-human objects, unlike the setup by Do and Kim [6] which relies on grouping human joints for optimization.

Algorithm 1 Normalize and Scale Keypoints

```
 1: norm_val_x, norm_val_y ← joints_coordinates[T=0][norm_index]
 2: for each time_step in joints_coordinates do
 3:     for each keypoint in time_step do
 4:         keypoint_x ← keypoint_x - norm_val_x
 5:         keypoint_y ← keypoint_y - norm_val_y
 6:     end for
 7: end for
 8: norm_val_x, norm_val_y ← joints_coordinates[T=end][norm_index]
 9: for each time_step in joints_coordinates do
10:     for each keypoint in time_step do
11:         keypoint_x ← keypoint_x / norm_val_x
12:         keypoint_y ← keypoint_y / norm_val_y
13:     end for
14: end for
15: signed_max_val ← max(abs(joints_coordinates))
16: for each time_step in joints_coordinates do
17:     for each keypoint in time_step do
18:         keypoint_x ← keypoint_x / signed_max_val
19:         keypoint_y ← keypoint_y / signed_max_val
20:     end for
21: end for
22: return joints_coordinates
```

3.3 Examples of the scaling and normalization algorithm in action

The selected normalization keypoint should ideally be one of the shoulders since the shoulders' keypoints are the most commonly visible keypoints when taking videos of people whether from the front or the back. When a key point such as the nose or ear is selected then the pose models often output no point and then give a poor output guess when the person is looking away from the camera and these facial features are not visible. This would result in poor performance and generalization on the scaling and normalization algorithm. Using a shoulder joint location is almost always output by the model (as long as the shoulders are framed) since the training data that pose models are trained on has labelled joint data for people when they are looking away of the camera as well. This is not the case for facial features such as nose or ear locations.

An example of the scaling and normalization algorithm in action in figure 1 conveys both relative position information as well as velocity information to the model that receives the keypoint information as input. The velocity information is carried over by the relative distance between the key points for motions that are performed since the frame rate is constant.

Note that the action looks upside down when due to the relative distance of the ankle's key points being far away from the normalization joint which is selected (left shoulder) thus they dominate the third portion of the scaling and normalization algorithm that normalizes the points to be kept between -1 and +1 for x and y coordinates. If the normalization keypoint selected for normalization were in the bottom half of the body then the plots would look right-side up.

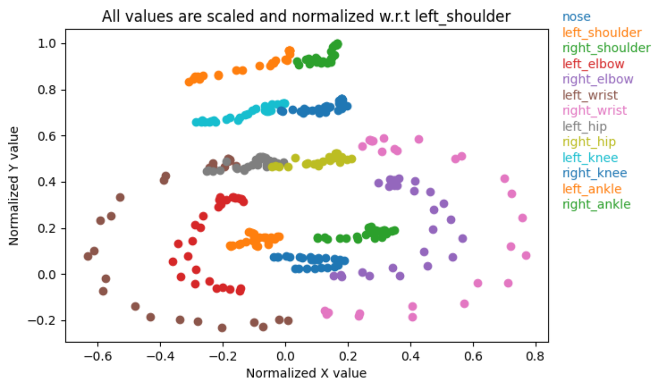

Fig. 1. Scaled and normalized jumping jack keypoints

3.4 Transformer action recognition model

The Transformer model proposed makes use of the same architecture as the pose-based Bi-LSTM but with the Bi-LSTM model swapped out by Transformer encoder layers. The encoder section of the transformer model is the necessary section of the model that enables using multi-head attention on a recognition task such as the task of action recognition in the problem at hand. The architecture used for the action recognition application is shown in figure 2. This architecture can effectively attend to vastly varying sequence lengths as long as the model is at least large enough to handle the longest sequence.

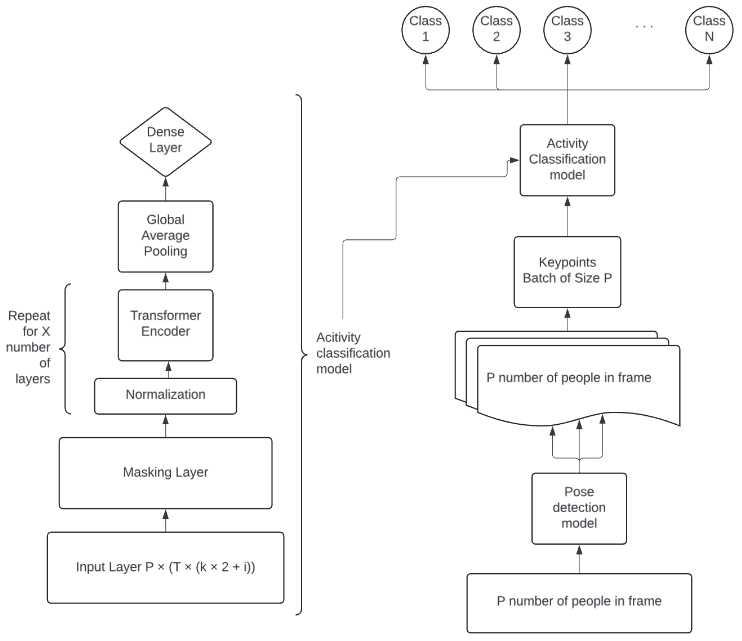
Class 1
Class 2
Class 3
. . .
Class N
Dense Layer
Global Average Pooling
Transformer Encoder
Repeat for X number of layers
Normalization
Masking Layer
Input Layer P × (T × (k × 2 + i))
Acitivity classification model
Activity Classification model
Keypoints Batch of Size P
P number of people in frame
Pose detection model
P number of people in frame

Fig. 2. Transformer encoder model used

4 Experimental Setup

Each of the clips in all of the datasets is limited in duration to an arbitrary max duration of 5 seconds per repetition. This can be adjusted to any length that makes sense for the actions that are to be recognised, however, keeping the max length of actions at 3 to 5 seconds provides good performance. A max duration of 5 seconds is set, given that most actions that humans perform which provide value to action recognition systems take less than 5 seconds per repetition. Training data for our custom dataset is captured such that each training clip constitutes exactly one repetition of the action. Training data is purposely done at different speeds to have a wide variety of durations between 0 and 5 seconds. This aids the model to recognise the same action regardless of how long it takes to complete a repetition of the action. The sequences are all right-zero-padded so as to allow for seamless masking using the Keras input masking library [5].

4.1 Own custom dataset

For the development of a fall detection model, a custom dataset is created of the following actions, performed by one person from one camera angle: fall, squat, punching (a "1-2-punch combination"), jumping jacks, and kicking. The purpose of this dataset is twofold, firstly, create a comprehensive set of examples for true falls as well as a set of adversarial actions. This ensures that the model is able do differentiate falls from other actions and secondly see if the actions performed by one person from one camera angle can generalize to the same actions performed by other people from other camera angles. The actions are performed with the person facing towards as well as away from the camera and all the angles in between. This is done so that the action can be recognised regardless of which direction the person is facing when they act.

The primary adversarial action that is of the most value is the squat class. Squats mimic falls in the sense that they both have a strong downward motion by the person acting, but the fall is more noisy in terms of the joints of the person acting and squats have the upward motion during the concentric portion of the repetition. In CCTV systems people who squat down to pick something up off the floor or people bending down to tie shoes are commonly mistaken for falls, therefore adding the squat class as an adversarial class is a smart choice. The custom dataset consists of example videos of each of the five categories (100-150 examples per action) done by one person from one angle. The goal is to see if a single person performing different falls, from a single camera angle can create a model that can generalize to the UR-Fall dataset which consists of multiple people doing different types of falls from a different camera angle than that used for recording the falls on our custom dataset that include falls. The models are trained on the custom dataset with an 80/20 train-test split. The 80% that is used for training is then further split up to an 80/20 train-validation split (resulting in a final split of 64/16/20 train, val, test split).

4.2 KTH dataset

The KTH dataset created by Schüldt et al. [21] is a dataset that consists of 6 different actions performed by 25 different people taken from several different angles: boxing, handclapping, handwaving, walking, jogging, and running. These actions allow for discrimination between the actions that the hands make for the first 3 classes and discrimination between bipedal motion done at different speeds and intensities for the last 3 classes. Good performance on this benchmarking dataset suggests a model that is good at discriminating between similar actions done by different people and therefore will be able to generalize well to arbitrary actions done by different people from different camera angles.

4.3 UR-Fall dataset

The UR-Fall dataset [13] is a dataset of 30 falls that are performed by 5 different people from one camera angle. The falls are performed from standing/walking to falling position and also from seated to fallen position. Good performance on this dataset indicates that the model can identify falls that occur from different initial positions.

4.4 Hardware

The experiments were developed on a computer with the following specifications: MacBook Pro (14-inch, 2021) with an Apple M1 Pro chip, which includes an 8-

core CPU (6 performance cores and 2 efficiency cores) and a 14-core GPU. The system has 16 GB of unified memory (RAM) and a 512 GB SSD for storage, running macOS Monterey. The approach was developed using Python 3.10 in Visual Studio Code (VSCode).

5 Results

5.1 Inference speed vs Accuracy comparisons

A summary of the model's performance for different training batch sizes is shown in figure 3 for testing done on the KTH dataset (64/16/20 train, val, test split). From this figure, it is clear that the performance of the Transformer model is more accurate than that of the Bi-LSTM model when tuned for accuracy and similar speeds and also faster when tuned for similar accuracy and better speeds (The models are trained on a 14" M1-pro MacBook pro, the absolute speeds are not important, note the relative speeds differences). The figure shows that the transformer can be tuned to either be of relatively similar accuracy as the Bi-LSTM with much better speeds or similar speeds with much better accuracy. It is worth noting that the Transformer model performs relatively flat and predictably for both accuracy and speed when trained on different batch sizes whilst the LSTM is very sensitive to batch sizes (and sequence lengths) and has poor accuracy when trained at batch sizes below 32 as well as at a batch size of 90. This is indicative of training dynamics that are very sensitive to gradient updates which the transformer model is not.

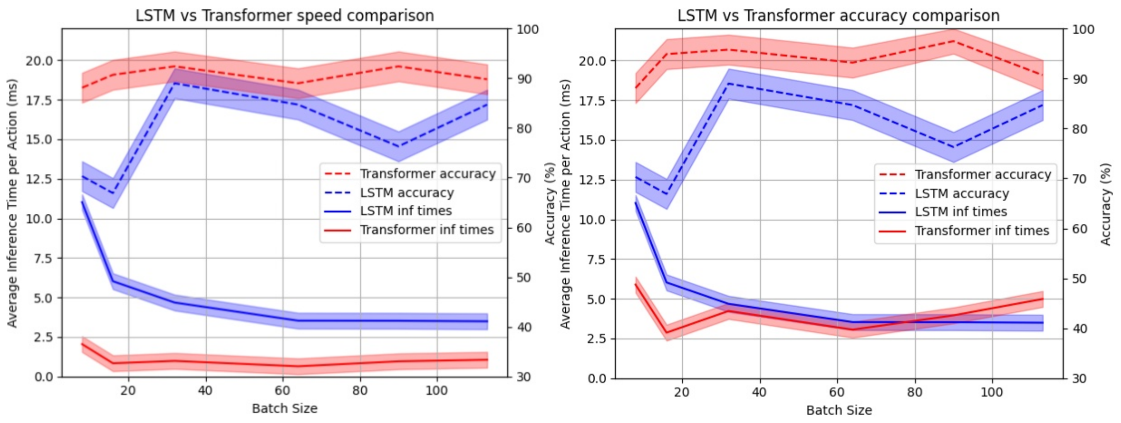

Fig. 3. Bi-LSTM vs Transformer

The results of testing the Bi-LSTM and Transformer models at their optimal batching numbers (batch size is 32 for Bi-LSTM and 90 for Transformer) for 5 training runs each yields average accuracies that are shown in table 1. This table suggests that the Transformer is the better model overall when comparing accuracy for all the tested datasets, not just the KTH dataset as displayed in figure 3. Note that the UR-Fall dataset is not trained on at all, the models are trained on our dataset with the specified training split (64/16/20 train, val, test split) and then the UR-Fall dataset is used completely as an unseen test dataset. Therefore the ability of the Bi-LSTM and the Transformer models to perform well on the UR-Fall dataset at a high confidence level (set high 85% to prevent false positives) suggests that the scaling and normalization algorithm allows for the action recognition models to generalize to never before seen actions done from different camera and body angles, by different people, as long as they perform the same action.

The Transformer model's results on our own dataset, KTH dataset and UR-Fall datasets are shown in confusion matrices and Metric-Confidence level curves in figures 4 to 6.

	Bi-LSTM	Transformer
Our own dataset	77.0%	95.5 %
KTH dataset	88.1%	97.5%
UR-Fall dataset (at 85% confidence)	83%	97%

Table 1. Accuracies of Bi-LSTM vs Transformer models on the tested datasets

5.2 Model size comparisons

When running a model on the edge for CCTV applications model size is a very important factor to consider. The advancements of AI-on-camera devices, such as the Sony IMX500 [3], allow for lightweight machine learning models to be run directly on the camera without streaming the video to a more powerful device for inference. These cameras often have a maximum model size limit that is 10MB [3]. The size of the Bi-LSTM model is 16.45MB compared to the Transformer model which is 1.77MB. This means that the Transformer model is not only faster, and more accurate but also an order of magnitude smaller, enabling edge-deployment of several different Transformer action recognition models directly on a single camera.

6 Limitations

The classification model has decreased precision when a high percentage of the keypoints obtained from the pose model are occluded. This can be solved by only making predictions when a high enough percentage of keypoints are present over the rolling window. The classification model can be trained to work well for extreme camera angles (very high/low w.r.t the subject) but this will require extra representative data in the training set. The model is limited by the performance

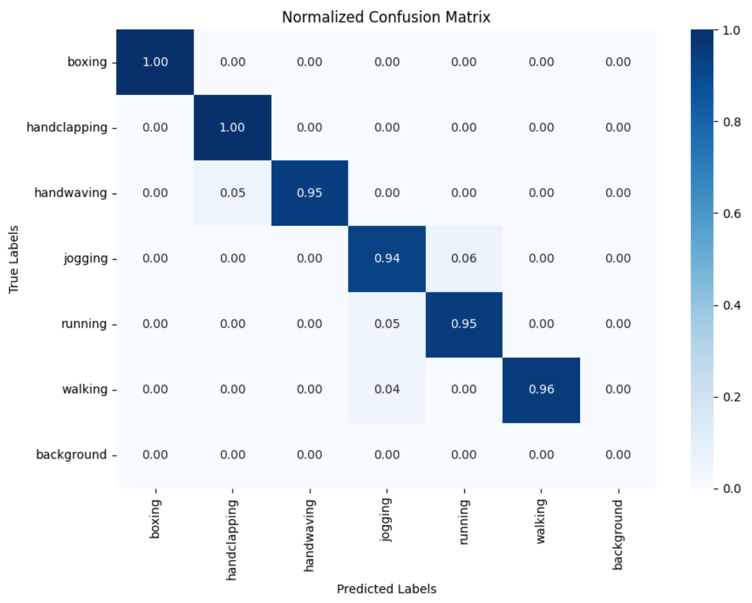

Fig. 4. Transformer confusion matrix on the KTH dataset for a 40% confidence level

of the pose model. This limitation can be mitigated by retraining the pose model on more data that is representative of difficult cases.

7 Conclusion

The Transformer model implemented in this paper accomplishes all the goals of the paper as stated in section 1. The model can generalise to non-human applications due to its pose model agnostic approach that is not specially tuned for humans, seeing as the scaling and normalization algorithm is made to be pose model agnostic. The model can generalize well to arbitrary actions as demonstrated by its performance on our dataset as well as the KTH dataset [21] and UR-Fall dataset [13] as is demonstrated by the model's performance on different camera angles, body angles and people. This is shown by its ability to perform well on our dataset (95.5% test set accuracy) which contains multiple body angles, the KTH dataset (97.5% test set accuracy) which contains multiple people and camera angles and the 97% accuracy on the unseen UR-Fall dataset which has multiple people and different fall types that were not trained on. The model has better accuracy than the Bi-LSTM model on all of the tested datasets as shown in table 1, better batching performance as shown in figure 3 and smaller model size as discussed in section 5.2. The model is also designed with CCTV deployments in mind and allows for easy scaling up of the model for MLOps relabelling. The model is easily parallizable which aids in better batch processing

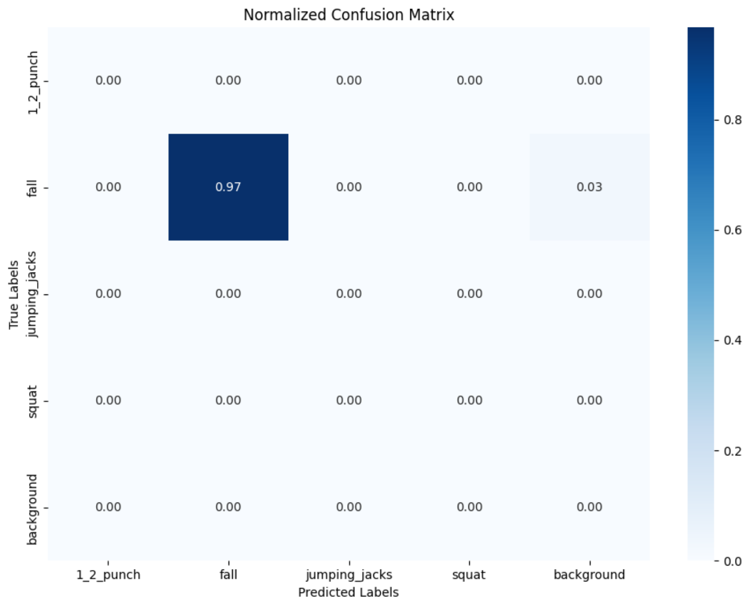

Fig. 5. Transformer confusion matrix on the UR-Fall dataset for a 85% confidence level

and faster inference than the recurrent LSTM models due to the Transformer architecture.

8 Future Work

Further work can be done on hyperparameter tuning and model architecture tuning which may involve adding residual layers to the transformer model. Further work can be done on creating an augmentation scaling and normalization algorithm that can translate training pose data from one angle to another to simulate different camera angles. This algorithm could also include further augmentation that could be used to "speed up" or "slow down" actions, drop certain joints add noise to the joints to create augmented datasets and even create fake adversarial joints to actions to force the model to learn the underlying joints and motions that are most important for each actions.

References

1. Ali, A., Pinyonatpong, E., Wang, P., Dorodchi, M.: Skeleton-based human action recognition via convolutional neural networks (cnn). arXiv **abs/2301.13360** (2023), `https://arxiv.org/abs/2301.13360`, accessed: 2023-01-31

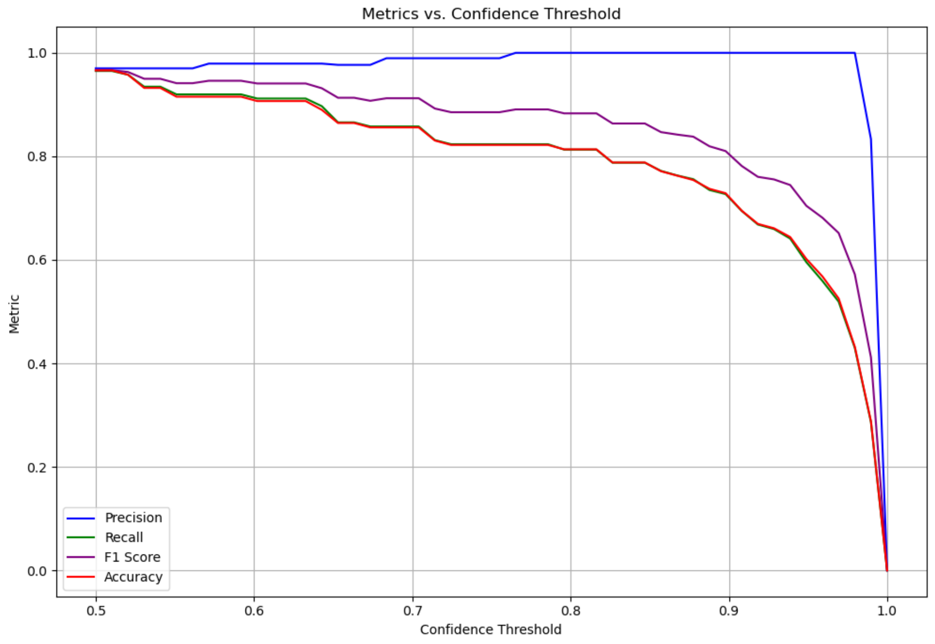

Fig. 6. Transformer metrics on the KTH dataset

2. Bilen, H., Fernando, B., Gavves, E., Vedaldi, A., Gould, S.: Dynamic image networks for action recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3034–3042. IEEE (2016). https://doi.org/10.1109/CVPR.2016.331
3. Bonazzi, P., Rüegg, T., Bian, S., Li, Y., Magno, M.: Tinytracker: Ultrafast and ultra-low-power edge vision in-sensor for gaze estimation (11 2023). https://doi.org/10.1109/SENSORS56945.2023.10325167
4. Chen, Y., Li, W., Wang, L., Hu, J., Ye, M.: Vision-based fall event detection in complex background using attention guided bi-directional lstm. IEEE Access **8**, 161133–161147 (2020). https://doi.org/10.1109/ACCESS.2020.3021795
5. Chollet, F., et al.: Keras. `https://keras.io` (2015)
6. Do, J., Kim, M.: Skateformer: Skeletal-temporal transformer for human action recognition. arXiv **abs/2403.09508** (2024), `https://arxiv.org/abs/2403.09508`, accessed: 2024-03-14
7. Hochreiter, S.: The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **6**, 107–116 (04 1998). https://doi.org/10.1142/S0218488598000094
8. Inturi, A.R., Manikandan, V.M., Garrapally, V.: A novel vision-based fall detection scheme using keypoints of human skeleton with long short-term memory network. Arabian Journal for Science and Engineering **48**, 1143–1155 (2023). https://doi.org/10.1007/s13369-022-06684-x
9. Karar, M., Shehata, H., Reyad, O.: A survey of iot-based fall detection for aiding elderly care: Sensors, methods, challenges and future trends. Applied Sciences **12**,

3276 (03 2022). https://doi.org/10.3390/app12073276
10. Kellokumpu, V., Zhao, G., Pietikäinen, M.: Recognition of human actions using texture descriptors. Machine Vision and Applications **22**(5), 767–780 (2009). https://doi.org/10.1007/s00138-009-0233-8
11. Khan, S., Naseer, M., Hayat, M., Zamir, S.W., Khan, F.S., Shah, M.: Transformers in vision: A survey. arXiv **abs/2101.01169** (2022), `https://arxiv.org/abs/2101.01169`, accessed: 2022-01-10
12. Kreuzberger, D., Kühl, N., Hirschl, S.: Machine learning operations (mlops): Overview, definition, and architecture. arXiv **abs/2205.02302** (2022), `https://arxiv.org/abs/2205.02302`, accessed: 2023-05-31
13. Kwolek, B., Kepski, M.: Human fall detection on embedded platform using depth maps and wireless accelerometer. Computer Methods and Programs in Biomedicine **117**(3), 489–501 (2014), `https://home.agh.edu.pl/~bkw/research/pdf/2014/KwolekKepski_CMBP2014.pdf`
14. Li, C., Zhong, Q., Xie, D., Pu, S.: Skeleton-based action recognition with convolutional neural networks. In: 2017 IEEE International Conference on Multimedia Expo Workshops (ICMEW). pp. 597–600 (2017). https://doi.org/10.1109/ICMEW.2017.8026285
15. Lin, C.B., Dong, Z., Kuan, W.K., Huang, Y.F.: A framework for fall detection based on openpose skeleton and lstm/gru models. Applied Sciences **11**(1), 329 (2021). https://doi.org/10.3390/app11010329, `https://doi.org/10.3390/app11010329`
16. Long, K., Haron, H., Ibrahim, M., Eri, Z.: An image-based fall detection system using you only look once (yolo) algorithm to monitor elders' fall events (02 2021)
17. Orozco, C.I., Xamena, E., Buemi, M.E., Berlles, J.J.: Human action recognition in videos using a robust cnn lstm approach. Ciencia y Tecnología **20**, 23–36 (2020)
18. Philipp, G., Song, D., Carbonell, J.G.: The exploding gradient problem demystified - definition, prevalence, impact, origin, tradeoffs, and solutions (2018)
19. Plizzari, C., Cannici, M., Matteucci, M.: Spatial temporal transformer network for skeleton-based action recognition. arXiv **abs/2012.06399** (2020), `https://arxiv.org/abs/2012.06399`, accessed: 2020-12-11
20. Ramirez, H., Velastin, S.A., Aguayo, P., Fabregas, E., Farias, G.: Human activity recognition by sequences of skeleton features. Sensors **22**(22), 3991 (2022). https://doi.org/10.3390/s22113991, `https://doi.org/10.3390/s22113991`
21. Schüldt, C., Laptev, I., Caputo, B.: Recognizing human actions: A local svm approach. In: Proc. ICPR (2004)
22. Sevilla-Lara, L., Liao, Y., Güney, F., Jampani, V., Geiger, A., Black, M.J.: On the integration of optical flow and action recognition. arXiv **abs/1712.08416** (2017), `https://arxiv.org/abs/1712.08416`, accessed: 2017-12-22
23. Shinde, S., Kothari, A., Gupta, V.: Yolo based human action recognition and localization. In: Procedia Computer Science, International Conference on Robotics and Smart Manufacturing (RoSMA2018). pp. 831–838. Elsevier (2018). https://doi.org/10.1016/j.procs.2018.07.112
24. Ultralytics, et al.: ultralytics/yolov8: Yolov8 - new state-of-the-art yolo model. `https://github.com/ultralytics/ultralytics` (2023), accessed: 2024-05-31
25. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need (2023)
26. Zhang, X., Zhou, Z., Chen, D., Wang, Y.E.: Autodistill: An end-to-end framework to explore and distill hardware-efficient language models. arXiv **abs/2201.08539** (2022), `https://arxiv.org/abs/2201.08539`, accessed: 2023-05-31