2023 International Conference on Machine Learning and Applications (ICMLA) | 979-8-3503-4534-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICMLA58977.2023.00039

Will Poppy Fall? Predicting Robot Falls in Advance Based on Visual Input

1st Borui He
Dept. of EECS
Syracuse University
Syracuse, USA
bhe100@syr.edu

2nd Garrett E. Katz
Dept. of EECS
Syracuse University
Syracuse, USA
gkatz01@syr.edu

***Abstract*—Falling is a critical problem for both people and robots, which may cause bodily harm to the elderly or prevent robots from executing issued orders. This motivates applications of machine learning to recognize and detect falls. Many datasets have been collected for this purpose, but primarily for detecting human falls after they occur. In this paper, we contribute simulated and real training data for robotic fall prediction in advance, based on egocentric video. We also compare an existing fall recognition model with a custom deep architecture we designed, to establish baseline performance on our datasets. We find that our architecture performs well for various prediction spans that can shift between training and testing.**

***Index Terms*—fall, robots, prediction**

I. Introduction

Falling can be a severe accident for both humans and humanoid robots. Egocentric cameras mounted on humans and robots are a common and informative source of observation that can be used to predict and detect falls. For robots in particular, if a fall is predicted in the near future, there is an opportunity to respond and prevent the fall by issuing appropriate motor commands. Action recognition and anomalous behavior detection [1] are two fields highly related to fall detection. However, most work in this area focuses on detecting events as they occur, rather than predicting them in advance.

A key challenge for robotic fall prediction is collecting sufficient training labels, since capturing a single fall episode can be time-consuming and risky for robotic hardware. On the other hand, a high volume of unlabeled image data can be captured from a robot's camera during episodes. Therefore, we propose a robotic fall prediction method combining self-supervised training of convolutional autoencoders for feature extraction, with a sample-efficient linear classifier on those features for fall prediction. Based on previous work [2], we hypothesize that the extracted features will be sufficient for good performance, and this hypothesis is borne out empirically. Our robotic platform is the Poppy Humanoid [3] for real-world data collection. Our code and training data is available online[1].

[1]Code is available at https://github.com/BoruiHe/Will-Poppy-fall and the real Poppy dataset is available at https://drive.google.com/file/d/1gnMWdRNPNHEHCsTN40nOruPfZXFxTSHg/view?usp=sharing

II. Related Work

Action recognition aims to classify actions in videos. Methods in this field include both convolutional neural networks (CNNs) [4]–[6] and vision transformers [7]–[9]. A general pattern is self-supervised learning of video representations used for classification or other downstream tasks.

Temporal action forecasting predicts the next object to interact with or the next action in videos. Previous works [10]–[12] estimate human actions and interactions, while [13] uses human action prediction to guide hunan-robot interaction.

Self-supervised learning extracts features without labels. For example, contrastive learning [14], [15] uses input similarity measures, and autoencoders [16], [17] reconstruct input data from low-dimensional representations.

Unimodal learning uses a single input data modality. Multimodal fall detection [18]–[20] fuses cameras, gyroscopes, accelerometers, and/or radars, but a single modality (e.g. RGB video) can also be effective [21]–[25]. We focus on unimodal learning since our robot has limited sensor hardware.

There are many datasets involving egocentric RGB video, which matches the sensor equipment on Poppy, but they focus on human behaviors other than falling [26], [27]. JRDB [28] and RICA [29] are both egocentric fall datasets but focus on humans rather than robots. JPL [30] and the dataset from [2] involve robotic falls in response to external forces applied by humans, but not falls due to the robot's own failure to keep its balance while walking. In [20], accelerometer data is used to estimate precisely when a fall occurs, but in platforms like Poppy where this modality is lacking, automatic fall labeling is not possible. Because we could not find a directly applicable 3rd party dataset, we collected our own simulated and real training data for egocentric fall prediction on the Poppy robot.

III. Methods

A. Datasets

Simulated Data: We simulated the Poppy robot using its open-source unified robot description file in conjunction with the Pybullet physics simulator [31]. Since we are more concerned with robot balance during normal operation rather than unforeseen external events, we induce falls through joint position control rather than external forces. Specifically, we

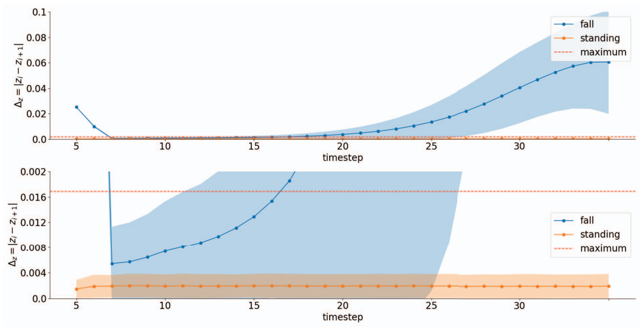

Fig. 1. Mean and standard deviation of the change in z-coordinate over time. z_i refers to the z-coordinate at timestep i. $|z|$ refers to the absolute value of z. A magnified version (the lower plot) of the upper one depicts the difference in detail. The red breaking line refers to the maximum absolute values of the change along z-axis.

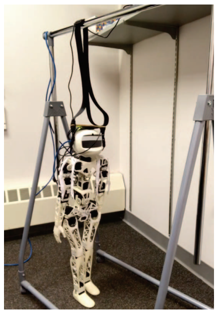

Fig. 2. The real Poppy Humanoid robot

apply Gaussian noise to the robot's target joint angles at every time-step and let the noise accumulate to eventually result in a fall. Letting θ_t denote the target angles at simulation time-step t, we set θ_t as follows:

$$\theta_{t+1} = \begin{cases} \theta_t + \mathbf{p}_t, & \text{fall episodes} \\ \theta_0 + \mathbf{p}_t, & \text{standing episodes} \end{cases} \tag{1}$$

where $\mathbf{p}_t$ is a perturbation vector with each component sampled from a normal distribution $\mathcal{N}(0, \sigma^2)$. We pad the episode with an initial pre-fall period by only applying perturbations after $t > 50$. In fall episodes, we use an adaptive σ value sampled from the following uniform distribution whose constants were tuned by hand to consistently produce falling behavior:

$$\sigma \sim \mathcal{U}\left(0.0015, 0.006 \cdot \max\left\{\frac{1}{4}, (t-188)/3\right\}\right), \tag{2}$$

so that t determines the upper bound on σ. In standing episodes we fixed $\sigma = 0.0015$. This reliably produced falling or standing as needed so we did not fine-tune further.

The simulated head camera captures videos with a resolution of 128×128 and 240 frames per second (fps). We generate 5k episodes each for falling vs. standing and extract only one video clip from each episode to maintain independent, identically distributed samples. In the simulation (but not real world) we have the ability to extract the current z-coordinate of the head camera, and the change in this coordinate over time can be used for fine-grained labeling of frames when a fall has occurred. The time evolution of z-coordinate change is shown in Fig. 1. By manual inspection of this plot, we set a threshold for change in z-coordinate to 0.00169 to detect falls, which is slightly greater than the maximum of all standing episodes.

To avoid very repetitive visual cues and potential overfitting, we placed the simulated robot in a four-walled room, with each wall's "wall-paper" set to a picture randomly selected from an arbitrary set of 28 pictures downloaded from the internet.

Real Data: We also collected data from our physical robot, shown in Fig. 2. Each episode consists of a single step forward using a joint trajectory with five waypoints optimized so that the center of mass was always contained in the support polygon. For simplicity we used this kinematic constraint rather than a more sophisticated trajectory optimization with dynamics constraints, and moved the robot slowly (several seconds per step) to mitigate the lack of dynamics modeling. This trajectory has a success rate (taking a step without falling) of roughly 80%. Ten images with resolution 96×128 are captured from the head camera during the transition between each waypoint. Episodes where the robot lost balance and the support strap became taut were manually labeled as falls. These labels are more coarse than the virtual data, only specifying the transition when a fall occurred and not a specific frame, and also unbalanced due to the 80% success rate.

B. Baseline approach

We use the fall detection system proposed in [32] as a baseline for comparison. Although originally designed for human falls, it is the closest prediction task to ours and the most directly applicable model to our datasets that we found in the literature. In this baseline, 10 frames are extracted from each video. A pretrained ResNet-50 [33] is used to extract learned features for each frame, which are then concatenated into a 2048-element feature vector for the video. This feature vector is then classified using a linear SVM [34].

C. Custom approach

We also evaluate our own custom architecture inspired by ResNet [33] and ConvNets [35], shown in Fig. 3. We learn a low-dimensional latent representation of a video clip with a 3D-convolutional autoencoder, trained to minimize reconstruction error. Then a linear support vector classifier (SVC) predicts fall or stand labels using a clip's latent representation as input. The autoencoder is trained with self-supervision on the large amount of unlabeled video data, while the linear SVC is more appropriate for the small set of labeled data.

Unlike the ResNet, our autoencoder uses 3D modules (convolution, batch normalization [36], and pooling) to integrate temporal information in a video. The latent dimension was a hyper-parameter we varied over $[256, 512, 1024, 2048]$. Like ResNet, we use identity shortcuts to help training. The decoder contains seven 3D transposed convolution layers to reconstruct the video clip from its latent representation. Mean squared

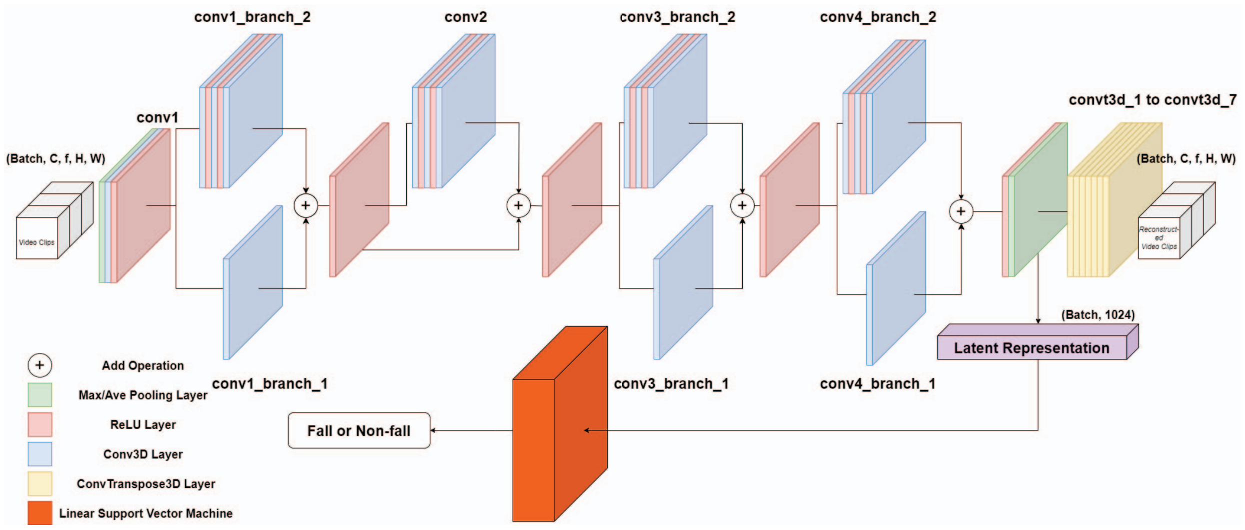

Fig. 3. An overview of the architecture. $Batch, C, f, H, W$ refers to batch size, the number of feature channels, the number of frames in a video clip, height and width, respectively. The number of feature channels of latent representations increases from 256 to 2048 by doubling itself.

error (MSE) between the original clip and its reconstruction is optimized with Adam [37] using a learning rate of 10^{-4} that decays at 10, 30, 60, 100 and 150 epochs, and a weight decay of 5×10^{-4}. We use 200 epochs total and a batch size of 40, and an 80/10/10% training/validation/testing split over 10K episodes in the virtual data and 30 episodes in the real data. Other layer parameters are listed in Table I.

The linear SVM [38] uses tolerance 10^{-5} and regularization parameter 1.0, and is trained separately and only on labeled data. For falls in the real data, we used the video clip that ends immediately before the transition labeled as fall. For virtual data, we used the clip a certain number of frames before the first frame labeled as fall; this number is another hyperparameter we call "prediction span" (described further below).

For standing episodes, we sample the time-point of the input video clip uniformly at random from the full video (excluding the first 50 frames in the simulated data, before joint perturbations are applied). Since the simulated data has a higher frame rate and many more frames, we also downsample by a factor of 8 before extracting the input video clips.

The real data is small (30 episodes) and unbalanced, so using a shallow SVM was especially important. We randomly divide the real dataset into a training set of 25 training samples and 5 testing samples. The real data also has a variable framerate because it has 10 frames per transition, but different transitions had different durations. We interpolated the real data across time to approximate a constant frame rate.

IV. Experiments

A. Experiment Design

Due to the coarse-grained labels in the real data, we used an experimental design with the virtual data that measures robustness to deviations in prediction span. To that end, we independently varied the prediction span during training (between 1, 5, or 9 frames) and testing (1, 3, 5, 7, or 9 frames). This means that in some experimental conditions, a network trained on one prediction span was tested on another.

Another experimental condition we varied was the source of standing examples, either for training or testing. Even in episodes where the robot eventually falls, there are early portions of the episode where the robot is still standing which could in principle be used for standing examples. We were interested in whether this sampling strategy significantly affected performance, so in one condition we sampled standing examples from standing episodes (denoted **SF**), and in another condition we sampled them from early in fall episodes before the fall occurred (denoted **F**). This could be done in training and/or testing, resulting in a 2×2 experimental design. We denote each possibility with the format <*training*> _ <*testing*>, so for example **F_SF** denotes the condition where standing examples are sampled from fall episodes in training but not testing. We also abbreviate **F_F** by **2F** and likewise for **SF**.

Lastly, as described above, we experimentally vary the size of the autoencoder latent representation. We compare the baseline and custom architectures on the basis of F1-score and balanced accuracy, each averaged over 3 independent repetitions of the experiments.

B. Results

We first evaluated the baseline model on a recognition task using our datasets (i.e., prediction span = 0), with the results shown in Table II, which shows that the baseline model works well on our virtual dataset but much worse on the real one. Next, although the baseline model is not originally intended for prediction tasks, we use it for the prediction task entailed by our datasets. Its performance on the real Poppy dataset is high only when training and testing prediction spans match (Table IV). We define "standard" experiments on the real Poppy dataset as those with equivalent prediction spans and highlight

TABLE I
PARAMETERS OF THE AUTOENCODER

Layer	Output Channels	Kernel Size	Stride	Padding	Output Padding
conv1	64	(2,3,3) [a]	(1,2,2)	(0,1,1)	-
conv1_branch_1	256	1	1	0	-
conv1_branch_2	128	1	1	0	-
	128	3	1	0	-
	256	1	1	0	-
conv2	128	1	1	0	-
	128	3	1	1	-
	256	1	1	0	-
conv3_branch_1	512	(3,1,1)	(1,2,2)	0	-
conv3_branch_2	128	(3,1,1)	(1,2,2)	0	-
	128	3	1	1	-
	512	1	1	0	-
conv4_branch_1	latent size [b]	(2,1,1)	(1,2,2)	0	-
conv4_branch_2	256	(2,1,1)	(1,2,2)	0	-
	256	3	1	1	-
	latent size	1	1	0	-
convt3d_1	512	3	2	1	1
convt3d_2	256	3	2	1	1 or (1,0,1) [c]
convt3d_3	256	3	2	1	1
convt3d_4	64	(4,3,3)	(1,2,2)	1	(0,1,1)
convt3d_5	64	(4,3,3)	(1,2,2)	1	(0,1,1)
convt3d_6	3	3	(1,2,2)	1	(0,1,1)
convt3d_7	3	3	(1,2,2)	1	(0,1,1)

[a] Different numbers for each dimension are organized into a tuple. Otherwise, one single number is used as the shorthand notation.
[b] Latent size could be 256, 512, 1024 or 2048.
[c] The second 3D transposed convolution layer (convt3d_2) may choose a proper output padding for reconstruction because of the different image resolutions.

TABLE II
EVALUATION FOR THE BASELINE MODEL

Datasets	F1-score (std)	Balanced Accuracy (std)
virtual Poppy	0.9985 (0.0003)	99.85 (0.03)
real Poppy	0.7967 (0.0324)	60.83 (7.17)

them in bold. The baseline model is better in the standard experiments, but when testing prediction spans exceed training prediction spans, it exhibits 0 F1-score and 50% balanced accuracy, entailing failure to detect true positives. And, with a long concatenated feature vector of 20480, it outperforms our model with the smallest latent size (256) only for other less challenging cases, i.e., training prediction span 9, testing prediction span 1 and 5 in Table IV and Table V. Table III and Table VI show that the baseline model outperforms our model in the scenario "2F", but does not transfer to the condition where standing examples come from standing videos (F_SF). The linear SVC of the baseline model converges within 10000 iterations and a tolerance of (10^{-4}).

Results on the virtual dataset are shown in Table VI. Instead of short-sightedness here, performance decreases with longer testing prediction span, and longer spans are more difficult even when they match in testing and training. On the other hand, compared to the real dataset, the performance gap between different latent sizes becomes negligible.

Based on comparisons between scenario 2F and SF_F, we find that training with extra "Standing" video clips from standing episodes does not always improve the results. Given a testing set of videos clips from both types of episodes, our model reaches a satisfying balanced accuracy even if it is trained on fall episodes only.

Fig. 4 compares balanced accuracy of experiments on the real Poppy dataset. We find that our model is sensitive to

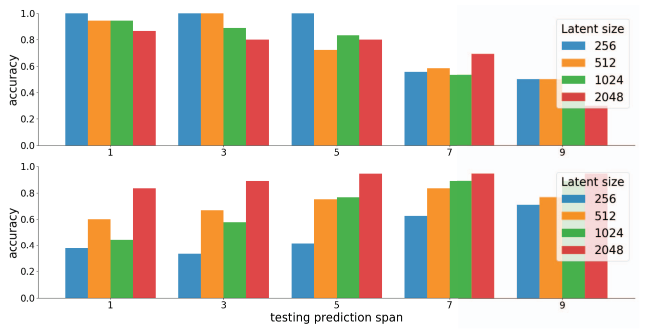

Fig. 4. Comparison of balanced accuracy between training prediction span 1 (top) and 9 (bottom). Balanced accuracy is converted to decimals.

TABLE III
EVALUATION FOR THE BASELINE MODEL TRAINED ON THE VIRTUAL POPPY DATASET

Scenario	PS [a]	F1-score			Balanced Accuracy		
		1	5	9	1	5	9
2F [b]	1	0.64	0.53	0.33	63.9	60.4	55.6
	5	0.67	0.64	0.44	63.2	64.6	59.6
	9	0.67	0.69	0.71	57.2	61.3	71.4
F_SF [c]	1	0.58	0.49	0.30	68.2	62.4	54.8
	5	0.58	0.55	0.38	66.6	65.1	56.9
	9	0.53	0.54	0.54	59.1	60.6	62.6
SF_F [d]	1	0.66	0.51	0.29	65.9	60.7	55.4
	5	0.65	0.62	0.41	62.6	63.3	59.0
	9	0.62	0.63	0.65	55.8	58.7	67.7
2SF [e]	1	0.94	0.91	0.81	93.9	91.1	83.6
	5	0.94	0.92	0.88	93.9	92.5	88.5
	9	0.91	0.90	0.91	90.7	89.8	90.7

[a] Training prediction span
[b] The highest standard deviation for F1-score and balanced accuracy are 0.0143 and 1.28.
[c] The highest standard deviation for F1-score and balanced accuracy are 0.0150 and 0.79.
[d] The highest standard deviation for F1-score and balanced accuracy are 0.0151 and 0.92.
[e] The highest standard deviation for F1-score and balanced accuracy are 0.0067 and 0.66.

prediction span. A model trained with a short prediction span but tested with longer span has a much worse performance, which we call "short-sightedness". Likewise, long-sightedness is also observed in the opposite condition. We report some representative conditions in Table V. The middle ground, training with prediction span of 5, is a reasonable trade-off especially with a greater latent size. We also find that a greater latent size is only necessary for longer prediction spans. It mitigates long-sightedness and improves the performance on all testing prediction spans. However, compared to significantly degrading performance with a shorter prediction span, a greater latent size does help improving the F1-score. This indicates that the greater latent size could mitigate short-sightedness problems by increasing true positives. The short/long-sightedness conditions have large standard deviations, so we can not draw a firm conclusion. We only observe a trend that could be explored further in future work.

TABLE IV
EVALUATION FOR THE BASELINE MODEL TRAINED ON THE REAL POPPY DATASET

PS [a]	F1-score			Balanced Accuracy		
	1	5	9	1	5	9
1 [b]	**1.00**	0.67	0.00	**100.0**	77.8	50.0
5	0.42	**1.00**	0.00	66.7	**100.0**	50.0
9	0.22	0.50	**1.00**	58.3	72.2	**100.0**

[a] Training prediction span
[b] The highest standard deviation of F1-score and balanced accuracy for all standard experiments are 0 and 0.

TABLE V
EVALUATION FOR OUR MODELS TRAINED ON THE REAL POPPY DATASET

PS [a]	LS [b]	F1-score			Balanced Accuracy		
		1	5	9	1	5	9
1 [c]	256	**1.00**	1.00	0.00	**100.0**	100.0	50.0
	512	**0.93**	0.50	0.00	**94.4**	72.2	50.0
	1024	**0.93**	0.67	0.11	**94.4**	83.3	40.0
	2048	**0.93**	0.88	0.19	**86.7**	80.0	30.0
5 [d]	256	1.00	**1.00**	0.62	100.0	**100.0**	73.6
	512	0.85	**0.85**	0.30	79.2	**79.2**	55.6
	1024	1.00	**1.00**	0.67	100.0	**100.0**	77.8
	2048	0.90	**0.90**	0.27	83.3	**83.3**	44.4
9 [e]	256	0.19	0.25	**0.74**	37.5	41.7	**70.8**
	512	0.63	0.86	**0.93**	60.0	75.0	**76.7**
	1024	0.39	0.79	**0.89**	44.4	76.7	**88.9**
	2048	0.77	0.93	**0.93**	83.3	94.4	**94.4**

[a] Training prediction span
[b] Latent size
[c] The highest standard deviation of F1-score and balanced accuracy for standard experiments are 0.09 and 9.4.
[d] The highest standard deviation of F1-score and balanced accuracy for standard experiments are 0.14 and 21.2.
[e] The highest standard deviation of F1-score and balanced accuracy for standard experiments are 0.16 and 20.5.

V. CONCLUSION

In this work, we aim to predict robotic falls in advance, in contrast with much existing work on visual action recognition which only applies after an action has already occurred. We employed a simple architecture composed of a convolutional autoencoder and linear SVC, trained from scratch for fall prediction with egocentric visual sensors only. Our results show that with an appropriate prediction span during training, the extracted latent features contain useful appearance and motion information that can be leveraged to predict falls. Our approach follows two well-known ideas, skip connections and self-supervised learning, and reaches promising results even without state-of-the-art architecture such as masked autoencoders.

VI. FUTURE WORK

A major issue in our present approach was short- and long-sightedness when training and testing prediction spans did not match. In future work, we plan to train one network on multiple prediction spans to make it more robust. Another opportunity in future work is to include joint angle trajectories as additional input to the prediction model. In contrast with human data, robotic joint angle readings are readily available from the motor hardware without a need for additional sensors or motion capture. This includes recent joint angles the robot has already visited, but can also include upcoming joint targets that are planned but have not yet been issued, as both may contain useful information about impending falls. By incorporating more feature extractors, our method could also be used to support more humanoid robots with trajectory planning in the future.

TABLE VI
EVALUATION FOR MODELS TRAINED ON THE VIRTUAL POPPY DATASET

S [a]	PS [b]	LS [c]	F1-score					Balanced Accuracy				
			1	3	5	7	9	1	3	5	7	9
2F [d]	1	256	0.57	0.53	0.48	0.43	0.34	60.15	58.77	57.65	56.45	54.60
		512	0.57	0.53	0.48	0.42	0.34	60.08	59.02	57.68	55.82	54.01
		1024	0.59	0.55	0.50	0.44	0.34	61.33	**60.12**	58.62	57.22	54.91
		2048	0.59	0.54	0.49	0.43	0.32	**61.50**	59.88	58.32	56.67	51.60
	5	256	0.61	0.58	0.55	0.50	0.41	60.70	59.87	**58.95**	57.98	55.58
		512	0.62	0.60	0.56	0.52	0.44	60.82	60.05	58.45	**58.37**	56.15
		1024	0.62	0.59	0.54	0.50	0.42	60.72	60.03	57.93	57.58	55.78
		2048	0.61	0.58	0.55	0.49	0.42	60.22	58.92	57.57	56.57	55.24
	9	256	0.65	0.63	0.60	0.57	0.52	59.65	58.78	58.03	57.70	56.36
		512	0.65	0.63	0.61	**0.58**	0.52	59.63	58.95	58.05	57.92	56.18
		1024	**0.66** [h]	0.63	0.60	0.57	0.51	60.53	59.33	58.55	57.67	56.23
		2048	0.65	**0.63**	**0.61**	0.57	**0.63**	59.73	58.72	58.45	57.13	**56.50**
F_SF [e]	1	256	0.54	0.50	0.46	0.41	0.32	66.02	63.43	61.99	59.72	56.38
		512	0.54	0.50	0.46	0.40	0.33	66.04	63.61	61.77	59.28	56.48
		1024	0.57	0.53	0.48	0.42	0.32	67.78	65.33	62.87	60.18	56.75
		2048	0.56	0.52	0.47	0.41	0.31	57.55	54.02	52.50	60.24	56.38
	5	256	0.59	0.56	0.53	0.47	0.40	68.83	66.78	65.28	62.17	59.30
		512	0.59	0.56	0.53	0.48	0.41	68.69	66.78	64.87	63.18	59.67
		1024	0.59	0.56	0.53	0.48	0.41	68.48	67.04	65.09	62.64	59.51
		2048	0.57	0.55	0.52	0.47	0.40	67.20	65.85	64.31	61.77	58.80
	9	256	0.61	0.60	0.57	0.54	0.49	70.25	68.95	67.18	65.33	62.70
		512	0.62	0.59	0.57	**0.55**	0.49	70.52	68.53	66.97	**65.68**	62.47
		1024	**0.62**	**0.60**	0.57	0.54	0.49	**70.65**	**69.34**	67.16	65.29	62.18
		2048	0.61	0.59	**0.58**	0.54	**0.50**	69.99	68.30	**67.55**	64.96	**62.84**
SF_F [f]	1	256	0.63	0.61	0.58	0.55	0.52	58.12	57.00	56.33	55.43	54.82
		512	0.63	0.60	0.59	0.56	0.52	58.10	56.63	56.70	55.48	54.74
		1024	0.65	0.62	0.59	0.56	0.50	59.93	58.77	57.32	56.35	54.77
		2048	0.63	0.61	0.58	0.56	0.51	58.40	57.60	56.48	55.83	54.19
	5	256	0.65	0.63	0.61	0.58	0.55	57.43	57.20	56.43	55.37	54.58
		512	0.66	0.64	0.62	0.59	0.56	58.55	57.22	57.38	56.08	55.59
		1024	0.64	0.63	0.61	0.58	0.53	58.05	57.48	57.30	56.62	54.93
		2048	0.63	0.61	0.60	0.57	0.53	57.32	56.63	56.45	55.53	53.92
	9	256	0.65	0.64	0.62	0.61	0.58	55.73	56.05	55.25	55.17	54.18
		512	0.65	0.64	0.62	0.61	0.58	56.68	56.35	55.70	55.30	54.91
		1024	0.66	0.65	0.63	0.61	0.57	57.52	57.48	56.45	56.20	54.66
		2048	0.66	0.65	0.63	0.61	0.57	57.63	57.22	56.70	56.03	54.62
2SF [g]	1	256	0.79	0.76	0.72	0.69	0.63	81.52	79.12	76.62	74.47	71.02
		512	0.78	0.76	0.73	0.68	0.62	80.62	78.85	77.05	74.20	70.68
		1024	0.81	0.77	0.73	0.68	0.60	82.42	79.78	76.82	74.02	69.87
		2048	0.79	0.76	0.73	0.68	0.61	81.28	79.15	46.87	74.07	69.93
	5	256	0.83	0.81	0.78	0.74	0.68	84.32	82.28	80.68	77.80	74.25
		512	0.84	0.82	0.79	0.75	0.69	84.85	82.95	80.73	78.22	74.67
		1024	0.81	0.78	0.75	0.72	0.65	82.42	80.53	78.23	76.25	72.05
		2048	0.79	0.76	0.74	0.70	0.63	80.22	78.80	76.88	74.28	70.57
	9	256	0.83	0.82	0.80	0.77	0.73	83.67	82.92	81.43	79.48	76.35
		512	0.82	0.81	0.78	0.75	0.71	82.10	81.45	79.63	77.72	74.57
		1024	0.84	0.82	0.79	0.76	0.70	83.87	82.33	80.32	77.80	74.12
		2048	0.82	0.81	0.79	0.75	0.69	82.53	81.53	79.87	77.67	73.53

[a] Scenario
[b] Training prediction span
[c] Latent size
[d] The highest standard deviation for F1-score and balanced accuracy are 0.0369 and 2.69.
[e] The highest standard deviation for F1-score and balanced accuracy are 0.0321 and 2.01.
[f] The highest standard deviation for F1-score and balanced accuracy are 0.0293 and 1.60.
[g] The highest standard deviation for F1-score and balanced accuracy are 0.0443 and 3.55.
[h] Bold values indicate the highest score or accuracy.

REFERENCES

[1] O. Boiman and M. Irani, "Detecting irregularities in images and in video," *International journal of computer vision*, vol. 74, pp. 17–31, 2007.

[2] L. Xia, I. Gori, J. K. Aggarwal, and M. S. Ryoo, "Robot-centric activity recognition from first-person rgb-d videos," in *2015 IEEE winter conference on applications of computer vision*. IEEE, 2015, pp. 357–364.

[3] M. Lapeyre, P. Rouanet, and P.-Y. Oudeyer, "The poppy humanoid robot: Leg design for biped locomotion," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 349–356.

[4] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497.

[5] J. Lin, C. Gan, and S. Han, "Tsm: Temporal shift module for efficient video understanding," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 7083–7093.

[6] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 6450–6459.

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[8] Y. Li, C.-Y. Wu, H. Fan, K. Mangalam, B. Xiong, J. Malik, and C. Feichtenhofer, "Improved multiscale vision transformers for classification and detection," *arXiv preprint arXiv:2112.01526*, 2021.

[9] Z. Liu, J. Ning, Y. Cao, Y. Wei, Z. Zhang, S. Lin, and H. Hu, "Video swin transformer," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 3202–3211.

[10] S. Liu, S. Tripathi, S. Majumdar, and X. Wang, "Joint hand motion and interaction hotspots prediction from egocentric videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3282–3292.

[11] M. Liu, S. Tang, Y. Li, and J. M. Rehg, "Forecasting human-object interaction: joint prediction of motor attention and actions in first person video," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*. Springer, 2020, pp. 704–721.

[12] E. Dessalene, C. Devaraj, M. Maynord, C. Fermuller, and Y. Aloimonos, "Forecasting action through contact representations from first person video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[13] H. S. Koppula and A. Saxena, "Anticipating human activities using object affordances for reactive robotic response," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 1, pp. 14–29, 2015.

[14] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[15] X. Chen and K. He, "Exploring simple siamese representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 15 750–15 758.

[16] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 000–16 009.

[17] C. Wei, H. Fan, S. Xie, C.-Y. Wu, A. Yuille, and C. Feichtenhofer, "Masked feature prediction for self-supervised visual pre-training," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 14 668–14 678.

[18] K. Ozcan and S. Velipasalar, "Wearable camera-and accelerometer-based fall detection on portable devices," *IEEE Embedded Systems Letters*, vol. 8, no. 1, pp. 6–9, 2015.

[19] F. Hussain, F. Hussain, M. Ehatisham-ul Haq, and M. A. Azam, "Activity-aware fall detection and recognition based on wearable sensors," *IEEE Sensors Journal*, vol. 19, no. 12, pp. 4528–4536, 2019.

[20] F. Liang, R. Hernandez, J. Lu, B. Ong, M. J. Moore, W. Sheng, and S. Zhang, "Collaborative fall detection using a wearable device and a companion robot," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 3684–3690.

[21] W. Li, D. Zhang, Y. Li, Z. Wu, J. Chen, D. Zhang, Y. Hu, Q. Sun, and Y. Chen, "Real-time fall detection using mmwave radar," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 16–20.

[22] F. J. Abdu, Y. Zhang, and Z. Deng, "Activity classification based on feature fusion of fmcw radar human motion micro-doppler signatures," *IEEE Sensors Journal*, vol. 22, no. 9, pp. 8648–8662, 2022.

[23] F. A. S. F. de Sousa, C. Escriba, E. G. A. Bravo, V. Brossa, J.-Y. Fourniols, and C. Rossi, "Wearable pre-impact fall detection system based on 3d accelerometer and subject's height," *IEEE Sensors Journal*, vol. 22, no. 2, pp. 1738–1745, 2021.

[24] G. L. Santos, P. T. Endo, K. H. d. C. Monteiro, E. d. S. Rocha, I. Silva, and T. Lynn, "Accelerometer-based human fall detection using convolutional neural networks," *Sensors*, vol. 19, no. 7, p. 1644, 2019.

[25] M. Saleh, M. Abbas, and R. B. Le Jeannes, "Fallalld: An open dataset of human falls and activities of daily living for classical and deep learning applications," *IEEE Sensors Journal*, vol. 21, no. 2, pp. 1849–1858, 2020.

[26] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu *et al.*, "Ego4d: Around the world in 3,000 hours of egocentric video," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18 995–19 012.

[27] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price *et al.*, "Scaling egocentric vision: The epic-kitchens dataset," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 720–736.

[28] R. Martin-Martin, M. Patel, H. Rezatofighi, A. Shenoi, J. Gwak, E. Frankel, A. Sadeghian, and S. Savarese, "Jrdb: A dataset and benchmark of egocentric robot visual perception of humans in built environments," *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[29] V. Schmuck and O. Celiktutan, "Rica: Robocentric indoor crowd analysis dataset," *IMU*, vol. 127, no. 74,234, pp. 31–172, 2020.

[30] M. S. Ryoo and L. Matthies, "First-person activity recognition: What are they doing to me?" in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2730–2737.

[31] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[32] X. Wang, E. Talavera, D. Karastoyanova, and G. Azzopardi, "Fall detection and recognition from egocentric visual data: A case study," in *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part I*. Springer, 2021, pp. 431–443.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[34] M. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[35] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[36] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.