Activity Recognition: A Deep Learning Approach

Sharareh Sayyad

Washington State University, USA

Overview of Our Project

H. Khaled, et al. Complex Intelligence. Syst. 9, 3535-3546 (2023)

Data pre-processing

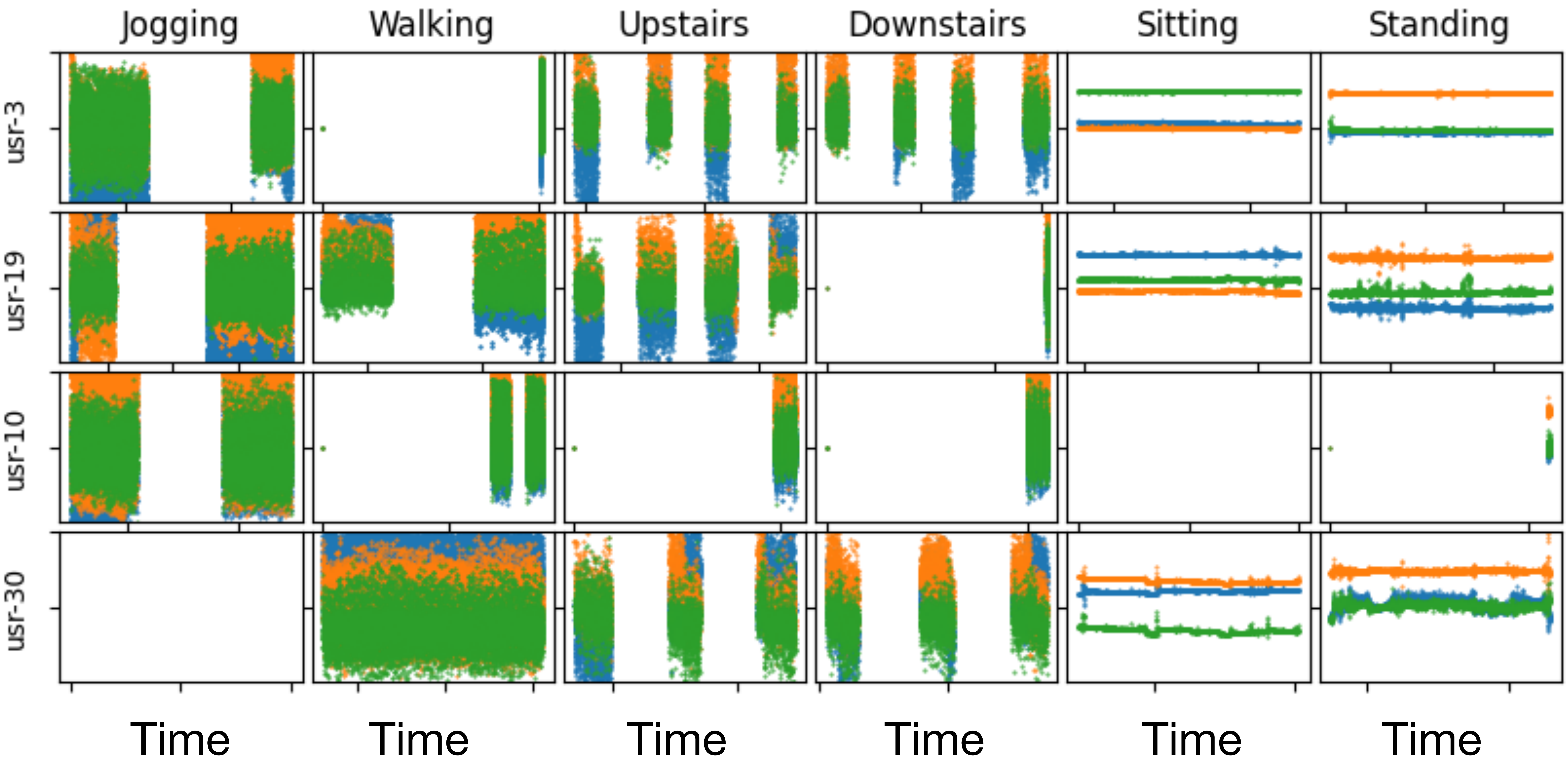
Neural Network models

Classifying Activities

Comparing model performances

WISDM Dataset

samples:1,098,207



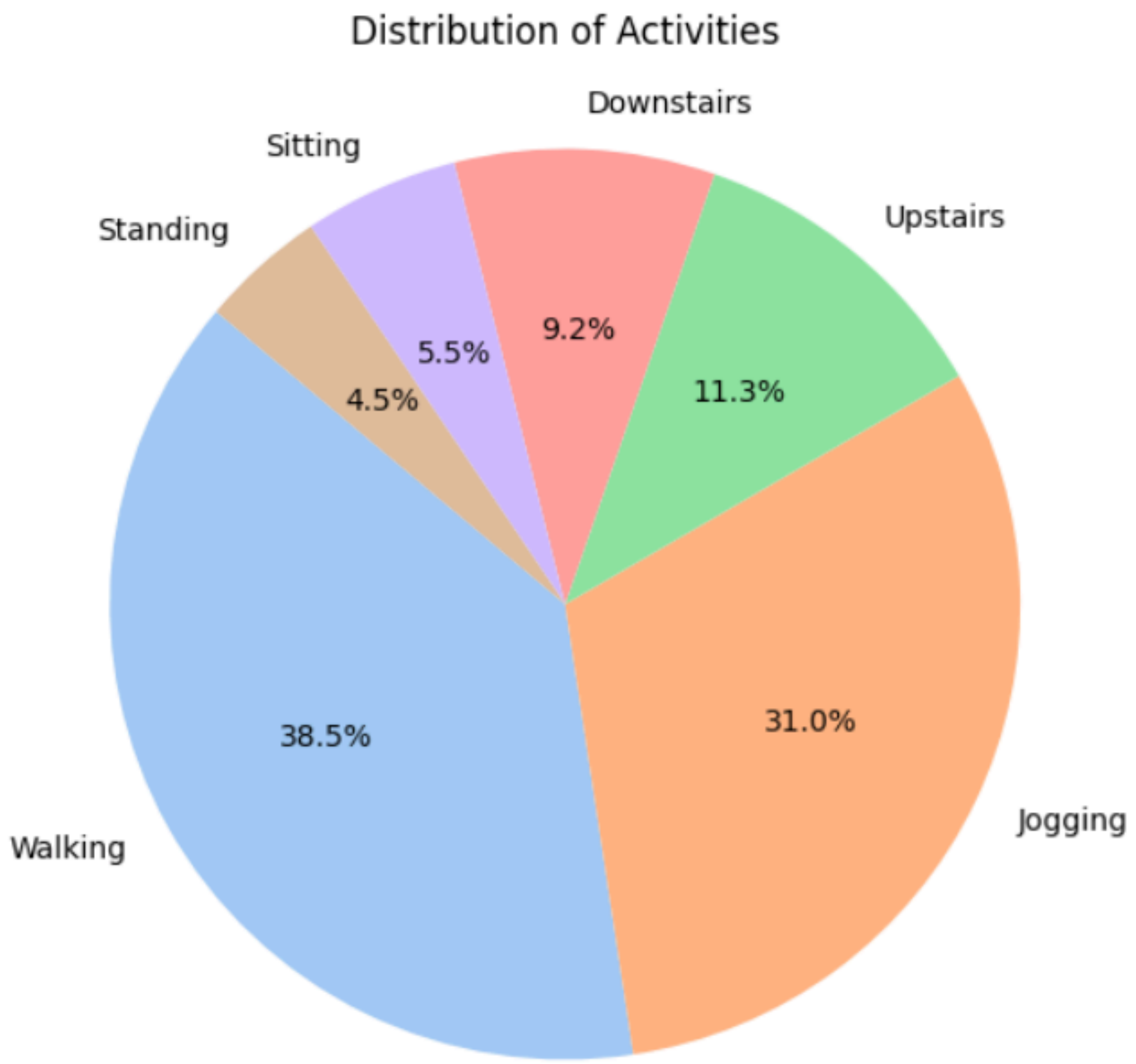
Features:

- timestamp
- x-accel
- y-accel
- z-accel
- user

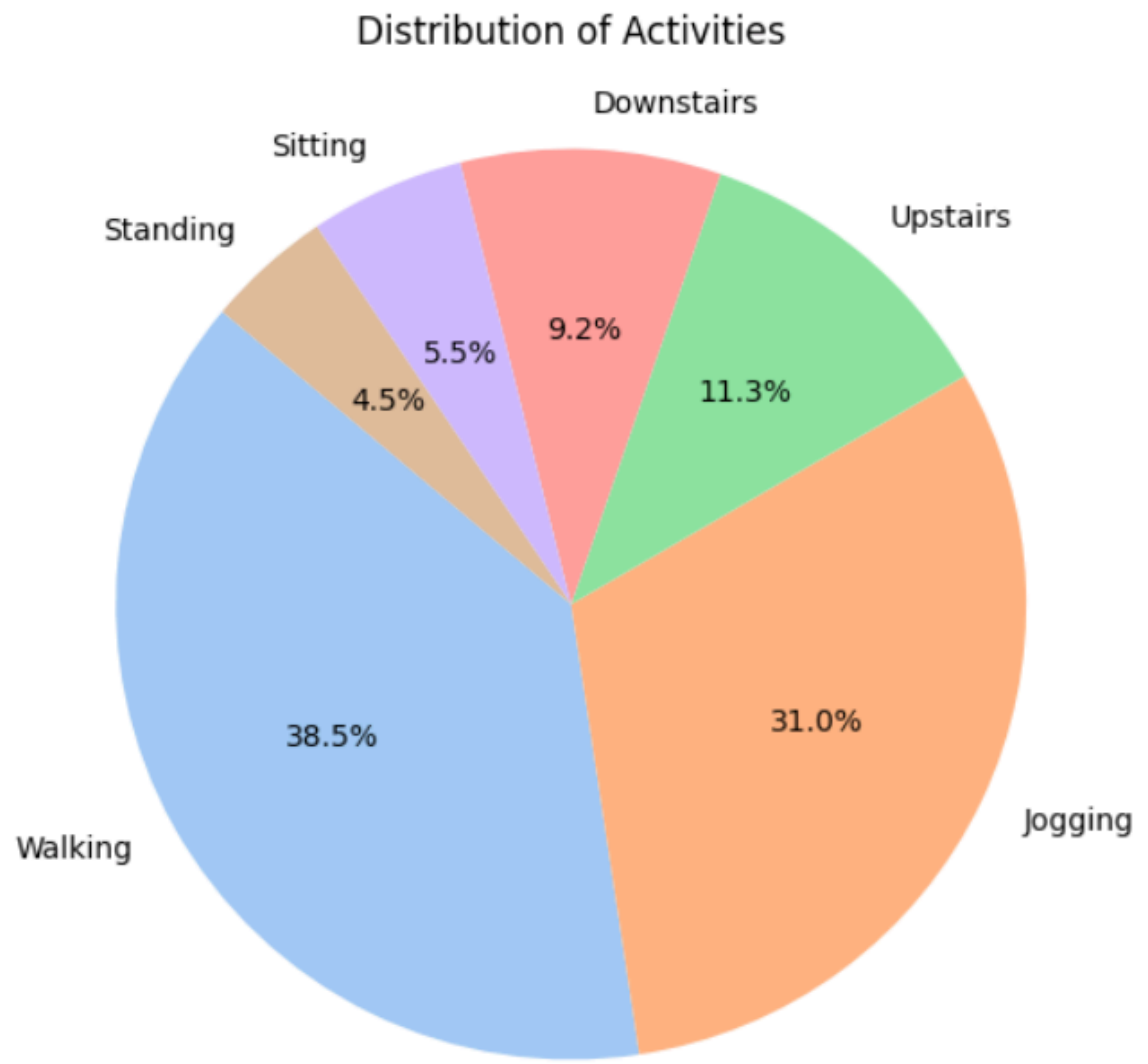
Labels(activity):

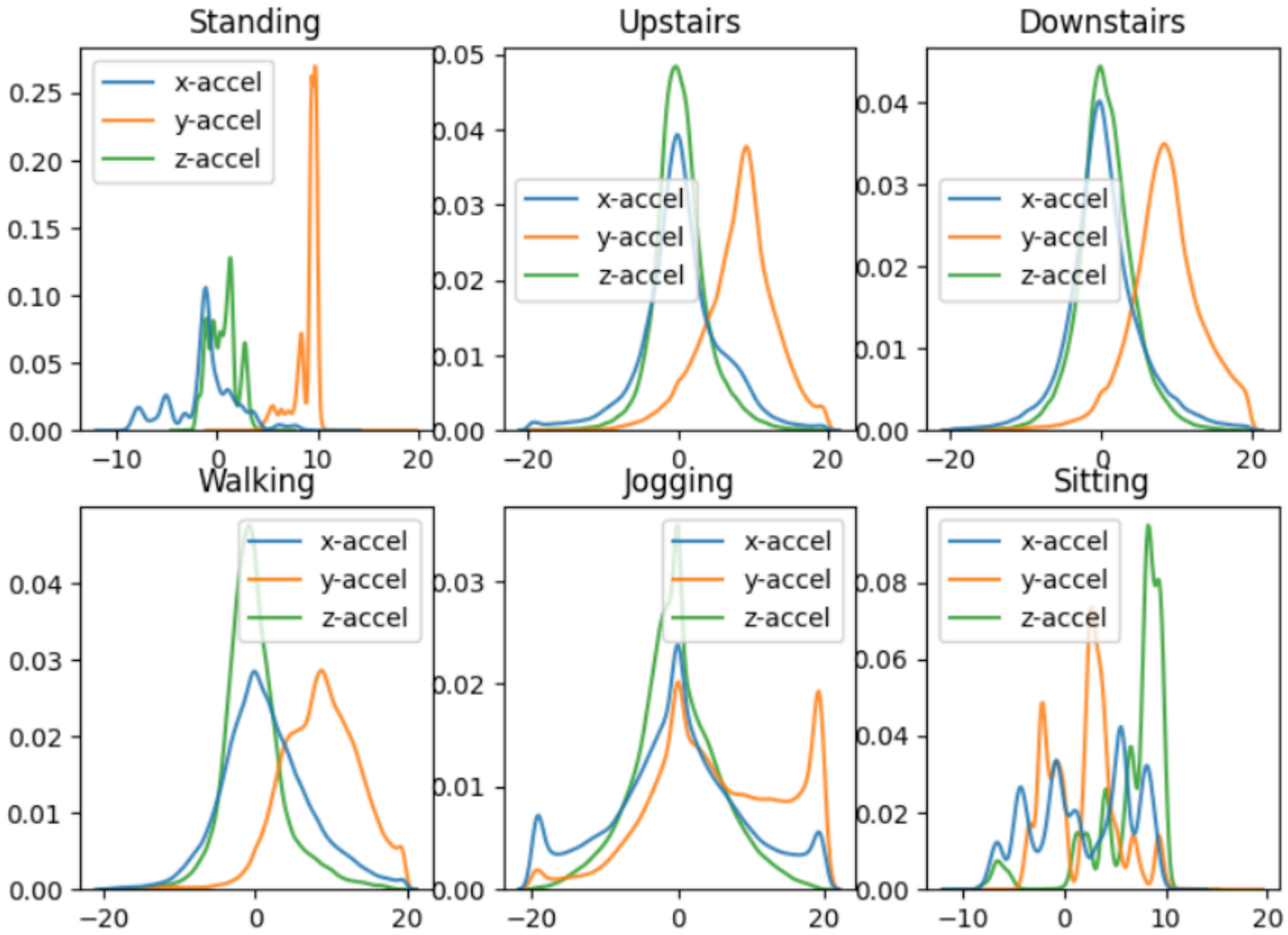
- Walking
- Jogging
- Upstairs
- Downstairs
- Sitting
- Standing

Distributions of samples for activities



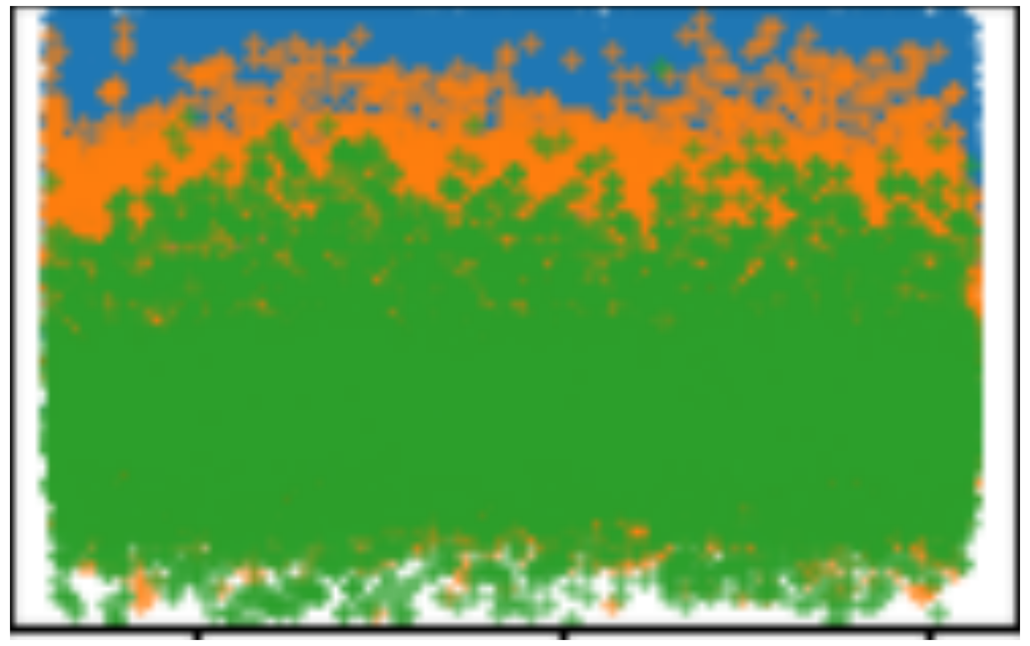
Distributions of samples for activities





Preparing inputs for neural network training

Passing raw samples
<u>with no time-ordering</u>

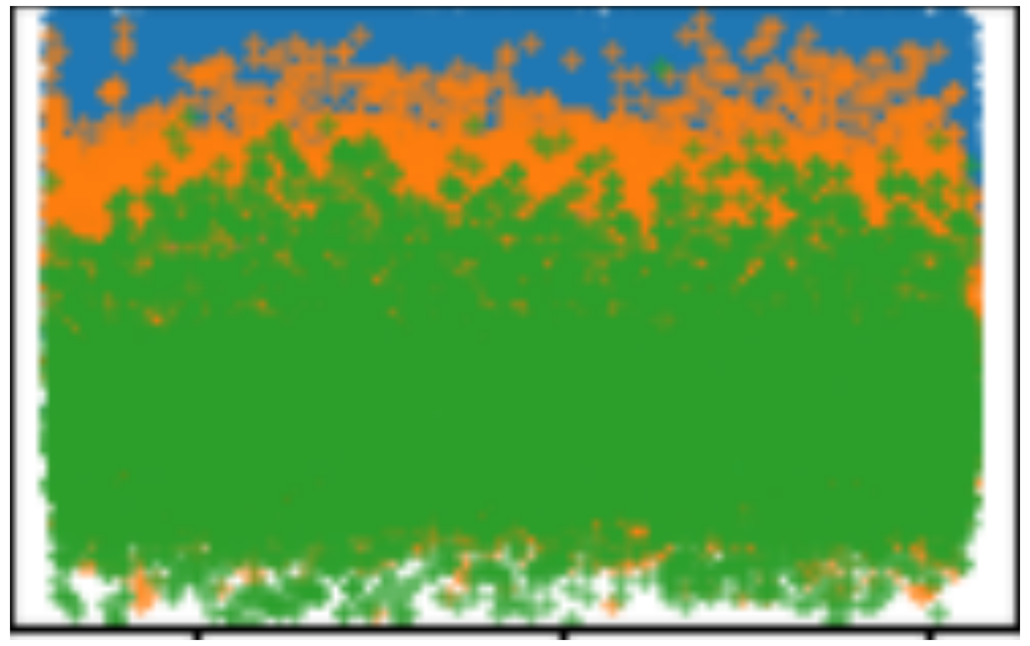


Tot. samples:1,098,207

Poor Performance!

Preparing inputs for neural network training

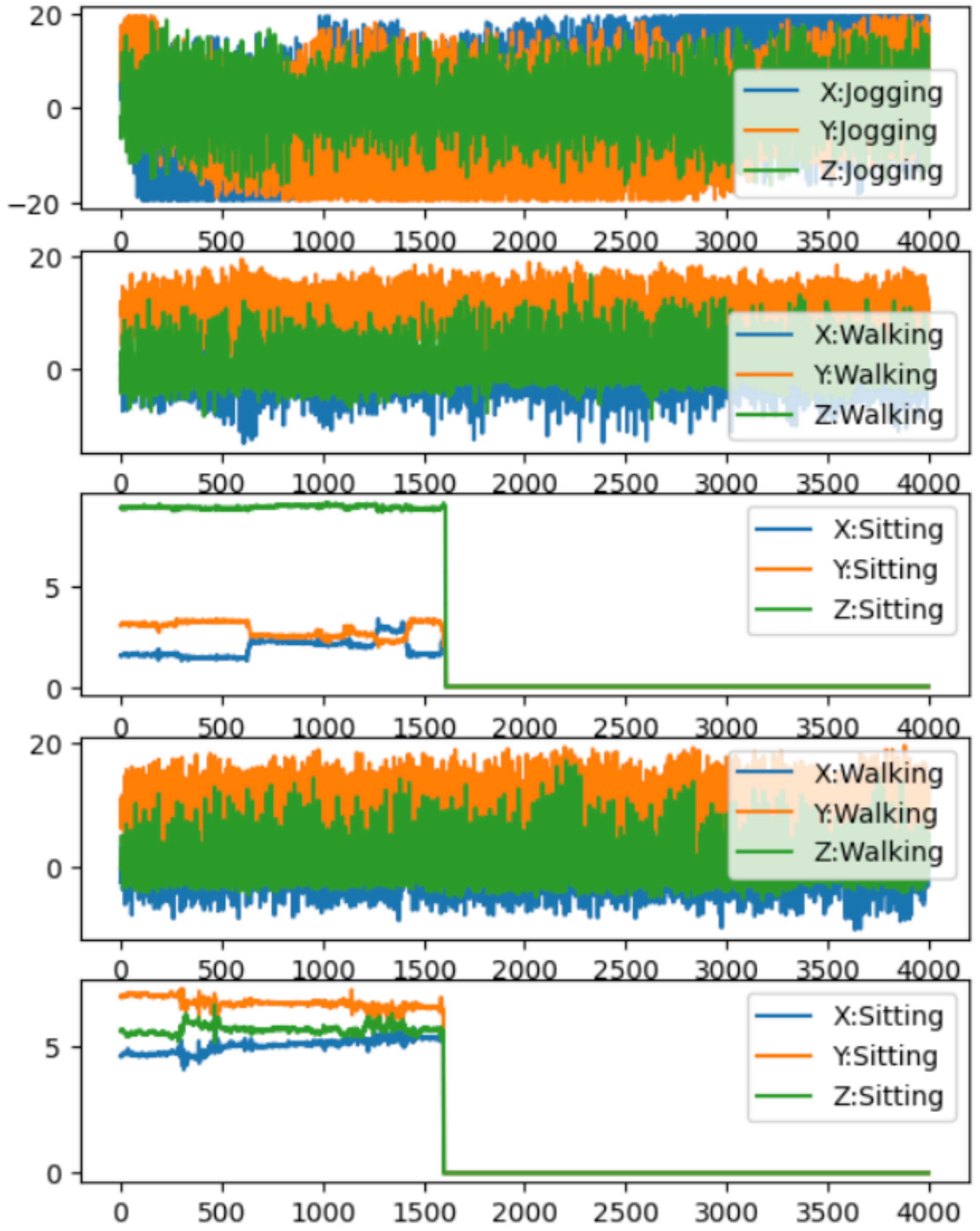
Passing raw samples
<u>with no time-ordering</u>



Tot. samples:1,098,207

Poor Performance!

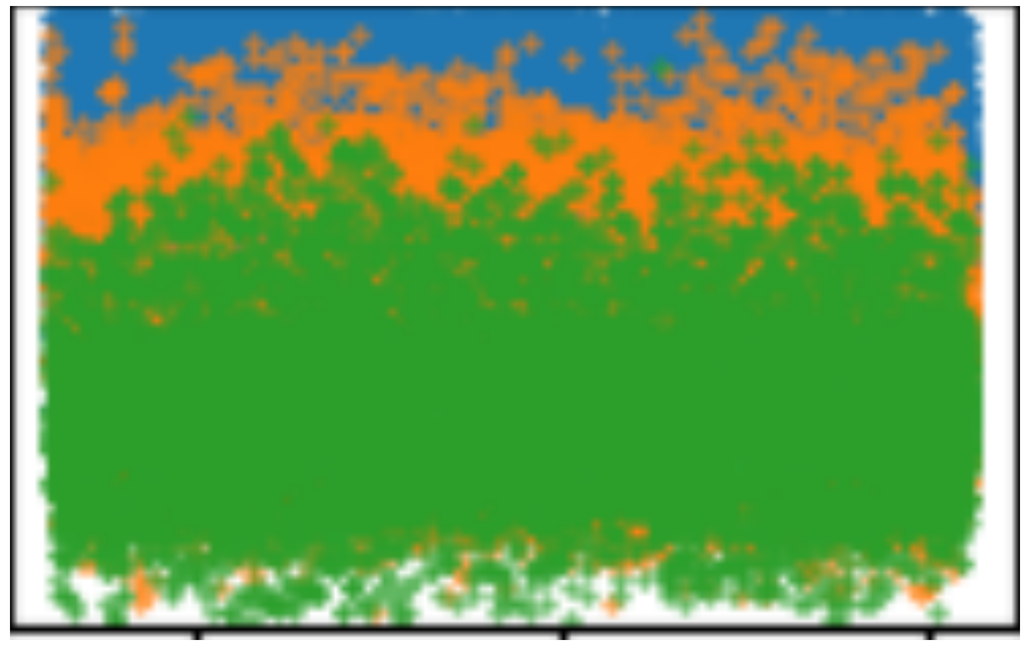
Passing samples as
<u>time-series</u>



Tot. Samples:179

Preparing inputs for neural network training

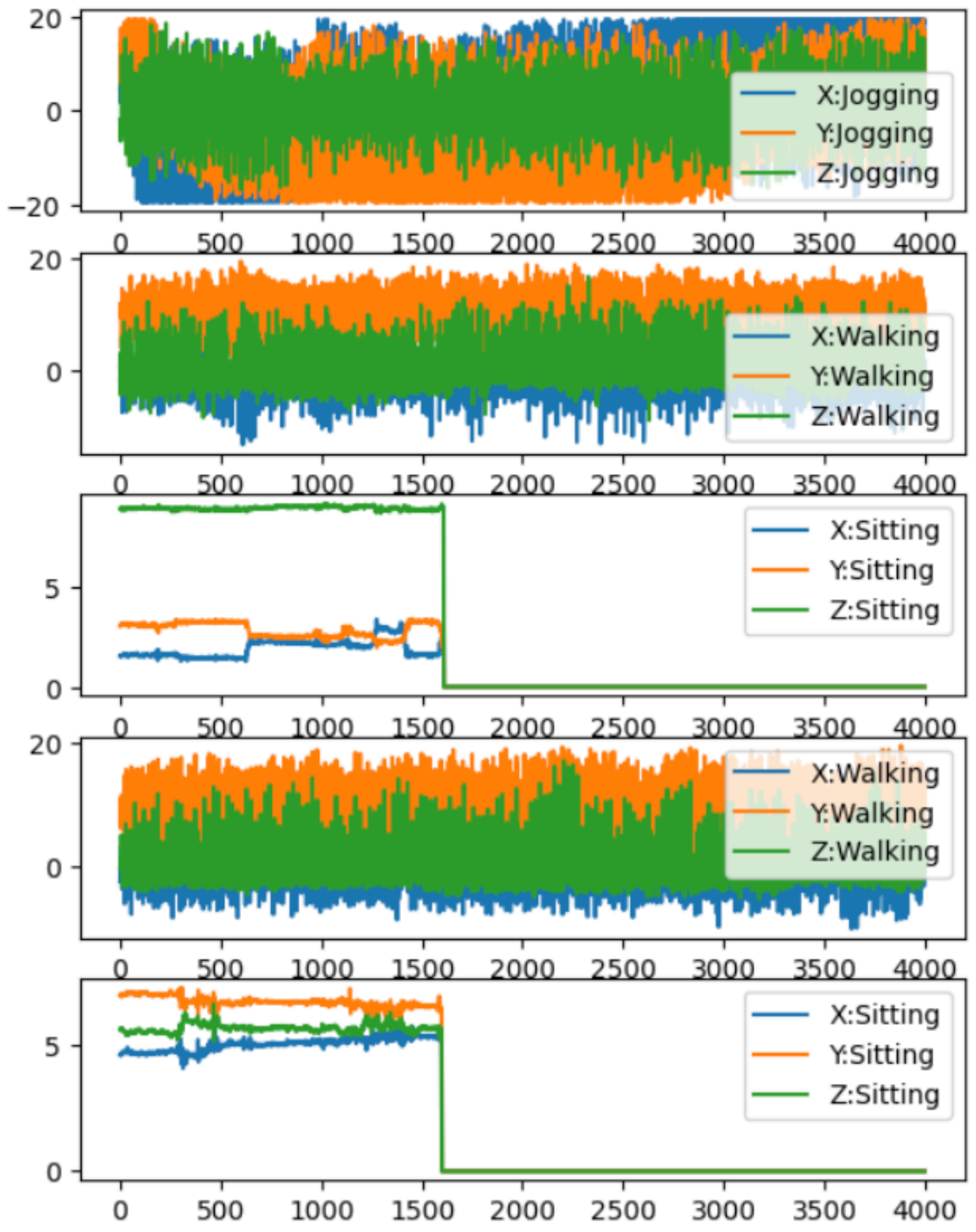
Passing raw samples
with no time-ordering



Tot. samples:1,098,207

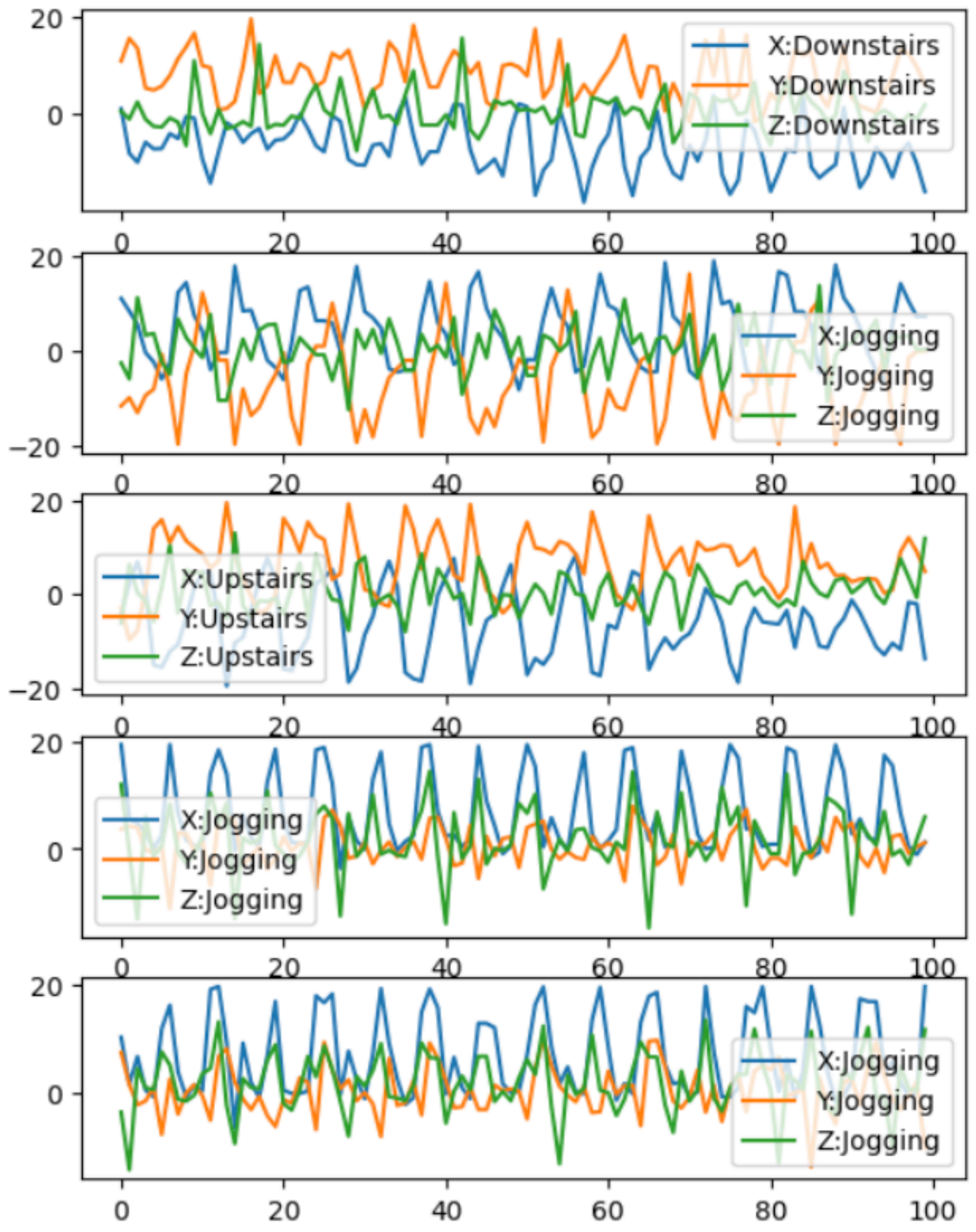
Poor Performance!

Passing samples as
time-series



Tot. Samples:179

Passing samples as
segmented time-series



Tot. Samples:5725

Dealing with data imbalance at the input level

Th/ Barros, et al. DOI:10.3390/educsci9040275

Training Neural Network Models

Training Neural Network Models

Designing models

Which architecture is best for classifying <u>time-series</u>?

Training Neural Network Models

Designing models

Training models

Which architecture is best for classifying <u>time-series</u>?

How to deal with an <u>imbalanced dataset</u>?

Training Neural Network Models

Designing models

Which architecture is best for classifying <u>time-series</u>?

Training models

How to deal with an <u>imbalanced dataset</u>?

What is included in training steps

Training imbalance dataset

- Kaiming weight initialization
- Weighted cross-entropy loss
- Stratified sampling
- AdamW as optimizer

What is included in training steps

Training imbalance dataset

- Kaiming weight initialization
- Weighted cross-entropy loss
- Stratified sampling
- AdamW as optimizer

Regularization to prevent overfitting

- Dropout layers in the model
- Learning-rate Scheduler
- Early stopping
- L2 norm regularization
- Batch normalization

What is included in training steps

Training imbalance dataset

- Kaiming weight initialization
- Weighted cross-entropy loss
- Stratified sampling
- AdamW as optimizer

Regularization to prevent overfitting

- Dropout layers in the model
- Learning-rate Scheduler
- Early stopping
- L2 norm regularization
- Batch normalization

Metrics to evaluate model performance

- Loss
- Confusion matrix

Measures	Formulas
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1-score	$\frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$
G-mean	$\sqrt{\frac{TN\times TP}{(TP+FN)\times(TN+FP)}}$

Models: CNN, LSTM

CNN models:

- **CNN1**: N (conv1d & maxpooling) + 2 linear layers
- **CNNwithBatchNorm**: N (conv1d & batch norm & maxpooling) + 2 linear layers
- **CNNwithSkip**: N residual blocks, each block with M (conv1d & batch norm) + Avg pooling + 2 linear layers

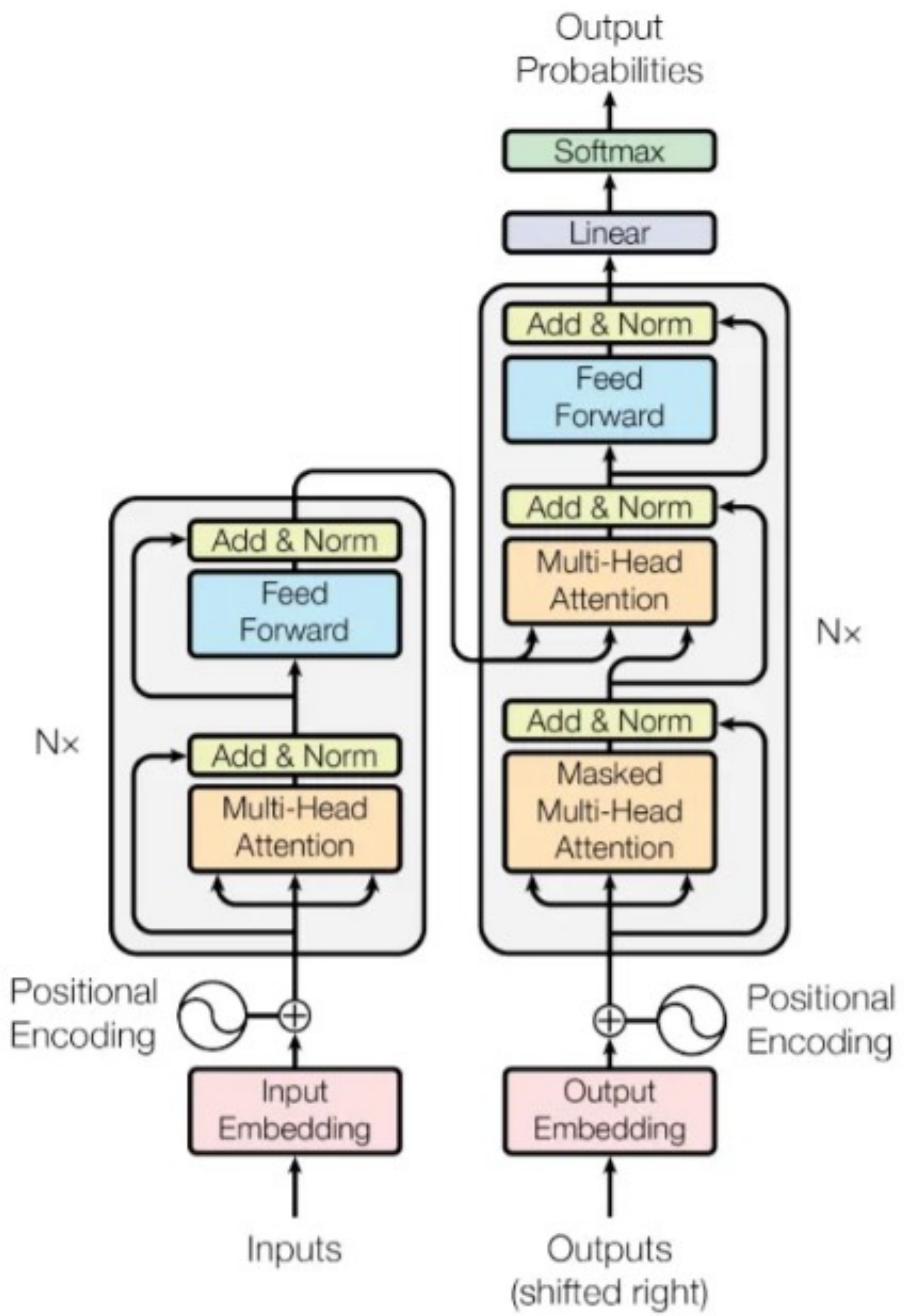
LSTM models:

- **LSTM1**: N (LSTM) + 2 linear layers

Models: CNN-LSTM

- **CNNLSTM1**: M(conv1d & maxpool) + N (LSTM) + 2 linear layers
- **CNNLSTMwithBatchNorm**: M (conv1d+batch norm+maxpooling)+ N (LSTM) + 2 linear layers
- **CNNwithSkipLSTM**: K residual blocks, each block with M(conv1d+batch norm) + avg pooling + N (LSTM) + 2 linear layers
- **CNNwithBatchNormLSTMParallel**: [N (conv1d+batch norm+maxpooling)+1linear lyr] + [M (LSTM)+1 linear lyr] + 1 linear layer on concat. outputs

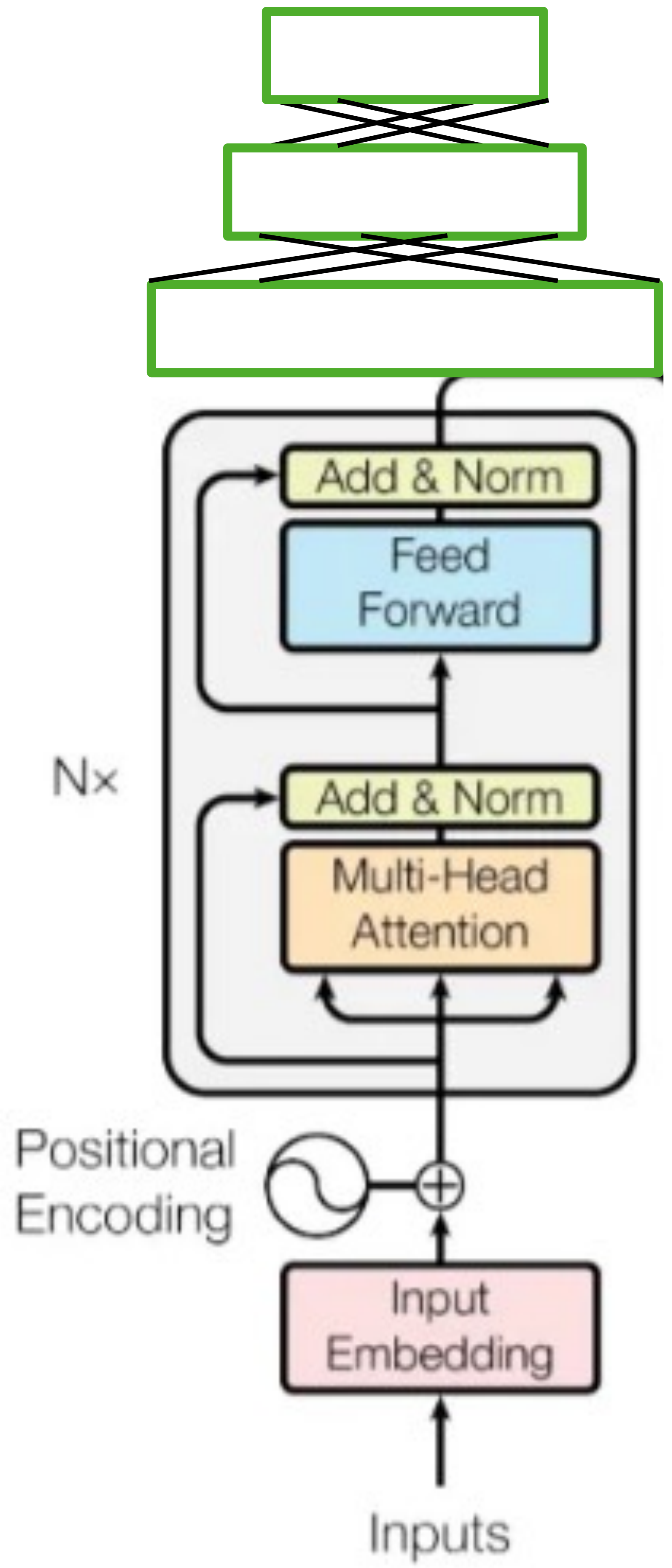
Models: Transformer



https://towardsdatascience.com/the-time-series-transformer-2a521a0efad3/

Models: Transformer

- **Trans1**: Input embedding(conv1d/linear layers) + positional encoding + N [transformer encoder layer(embed size, nhead, dim feedforward)] + 3 linear layers



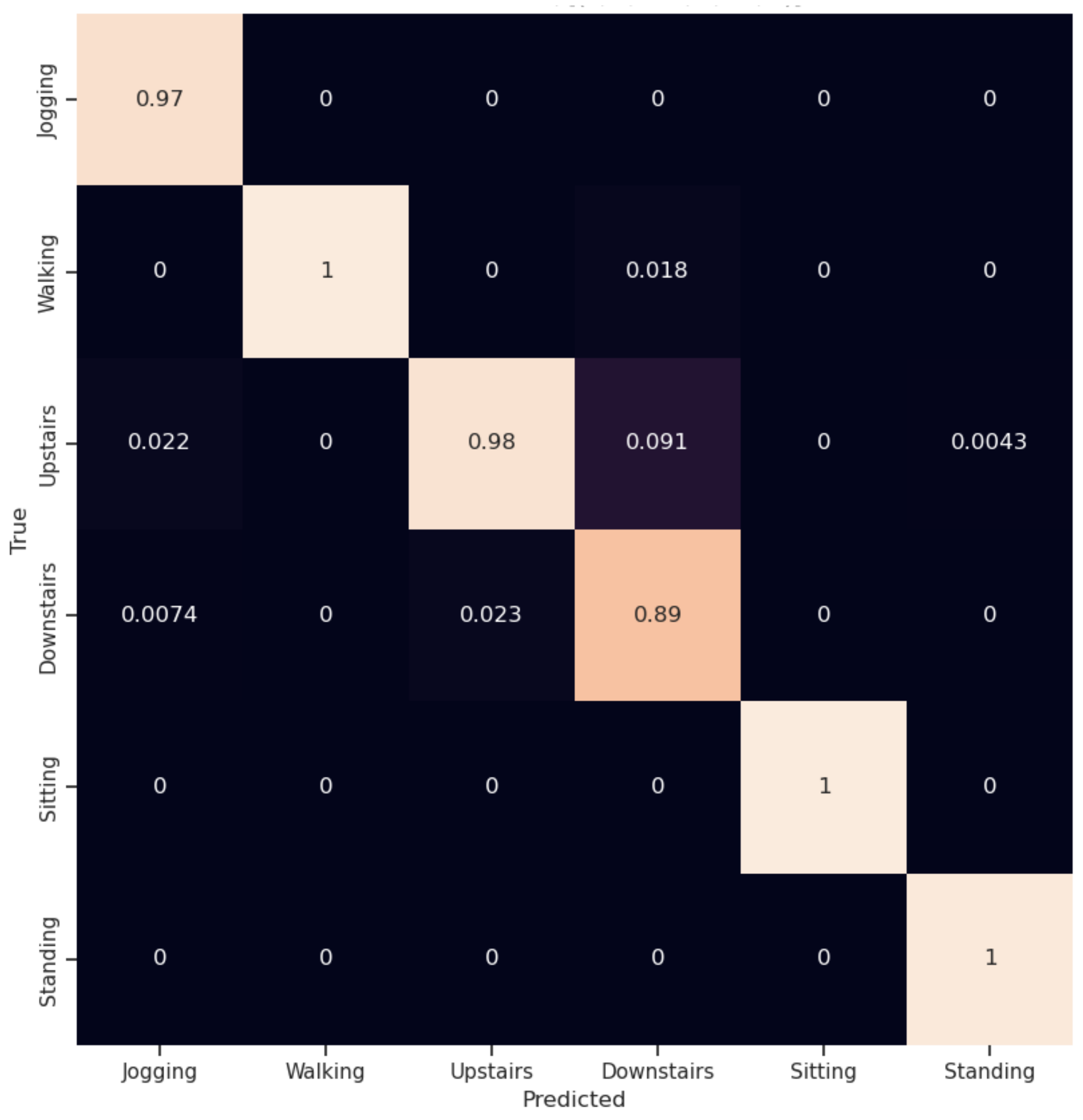
https://towardsdatascience.com/the-time-series-transformer-2a521a0efad3/

Best Results:

CNN

Segmented data & with down-upsampling

CNNwithBatchNorm



Accuracy on test set: 0.9890

F1 score on test set: 0.9812

Gmean on test set: 0.9897

Precision on test set: 0.9817

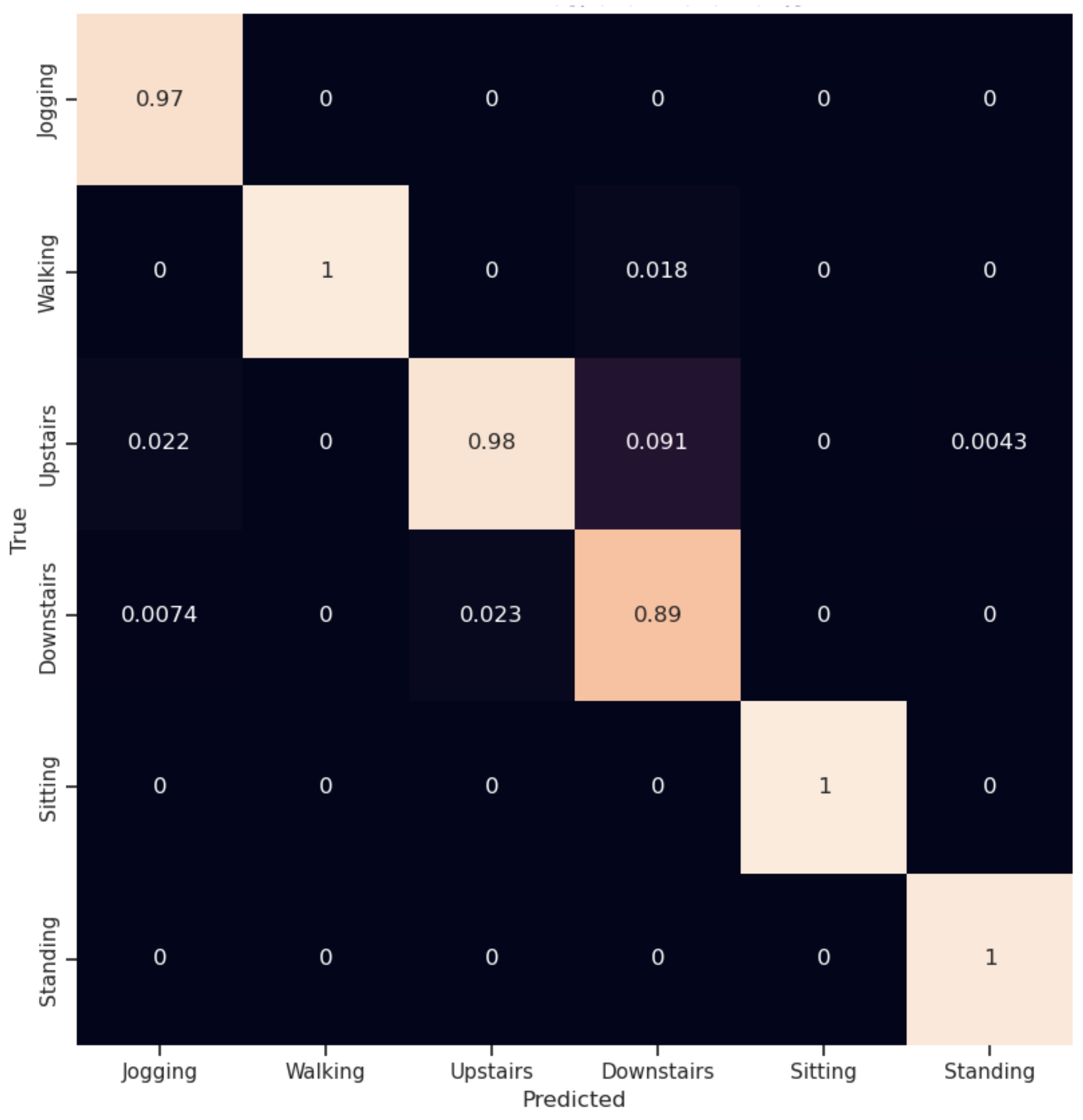
Recall on test set: 0.9809

Best Results:

CNN

Segmented data & with down-upsampling

CNNwithBatchNorm



Accuracy on test set: 0.9890

F1 score on test set: 0.9812

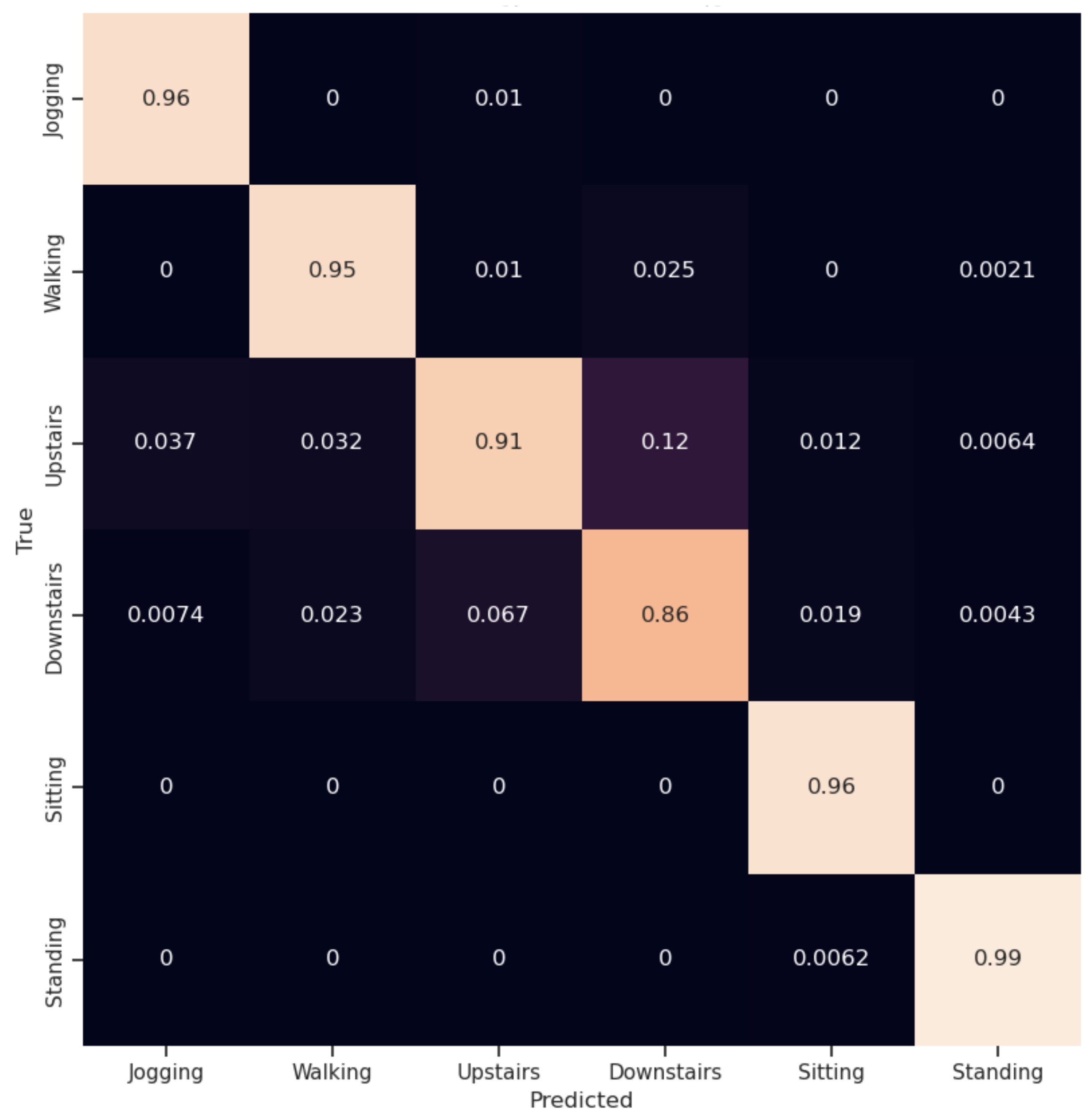
Gmean on test set: 0.9897

Precision on test set: 0.9817

Recall on test set: 0.9809

LSTM

Segmented data & with upsampling



Accuracy on test set: 0.9554

F1 score on test set: 0.9533

Gmean on test set: 0.9722

Precision on test set: 0.9541

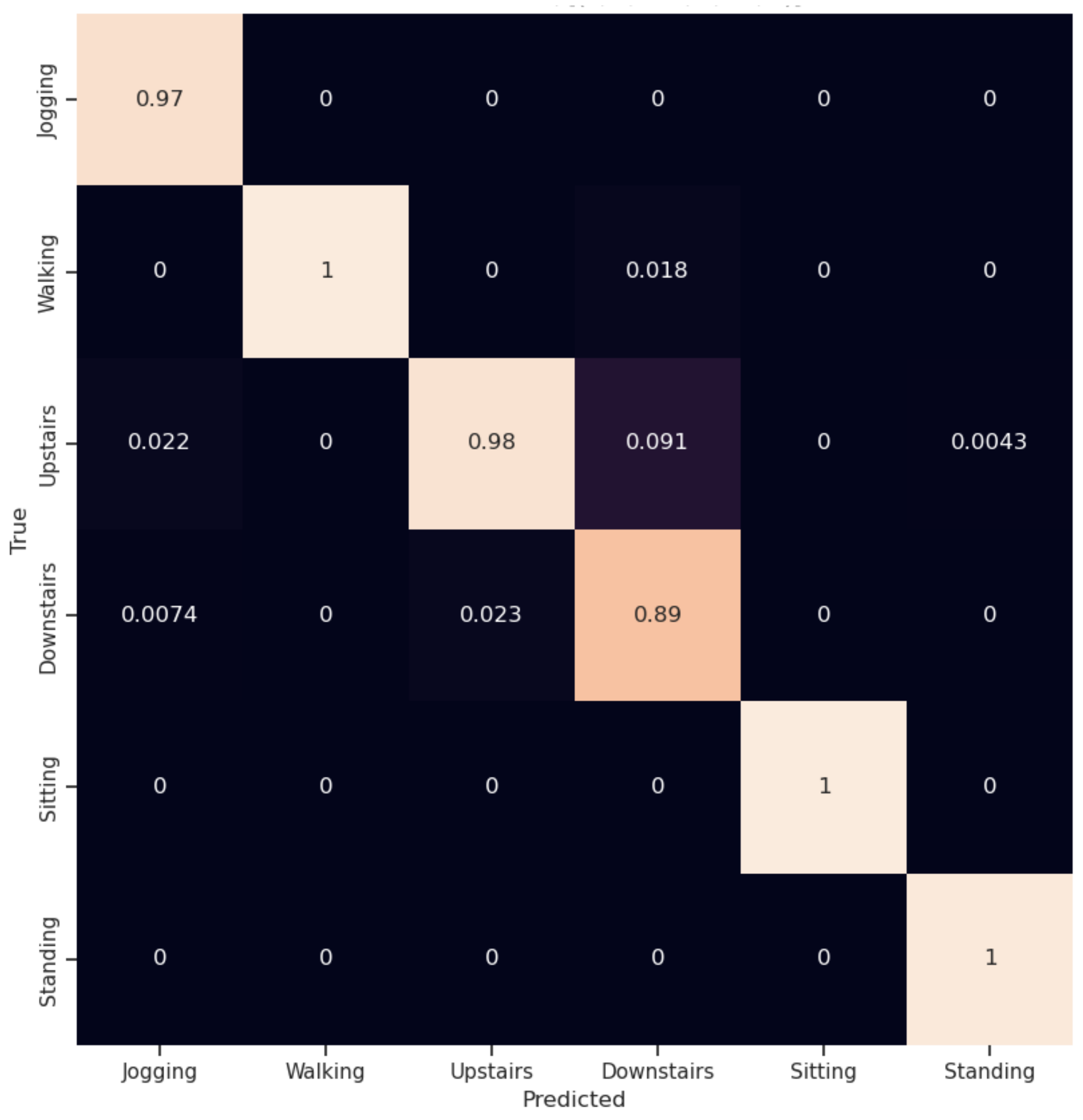
Recall on test set: 0.9527

Best Results:

CNN

Segmented data & with down-upsampling

CNNwithBatchNorm



Accuracy on test set: 0.9890

F1 score on test set: 0.9812

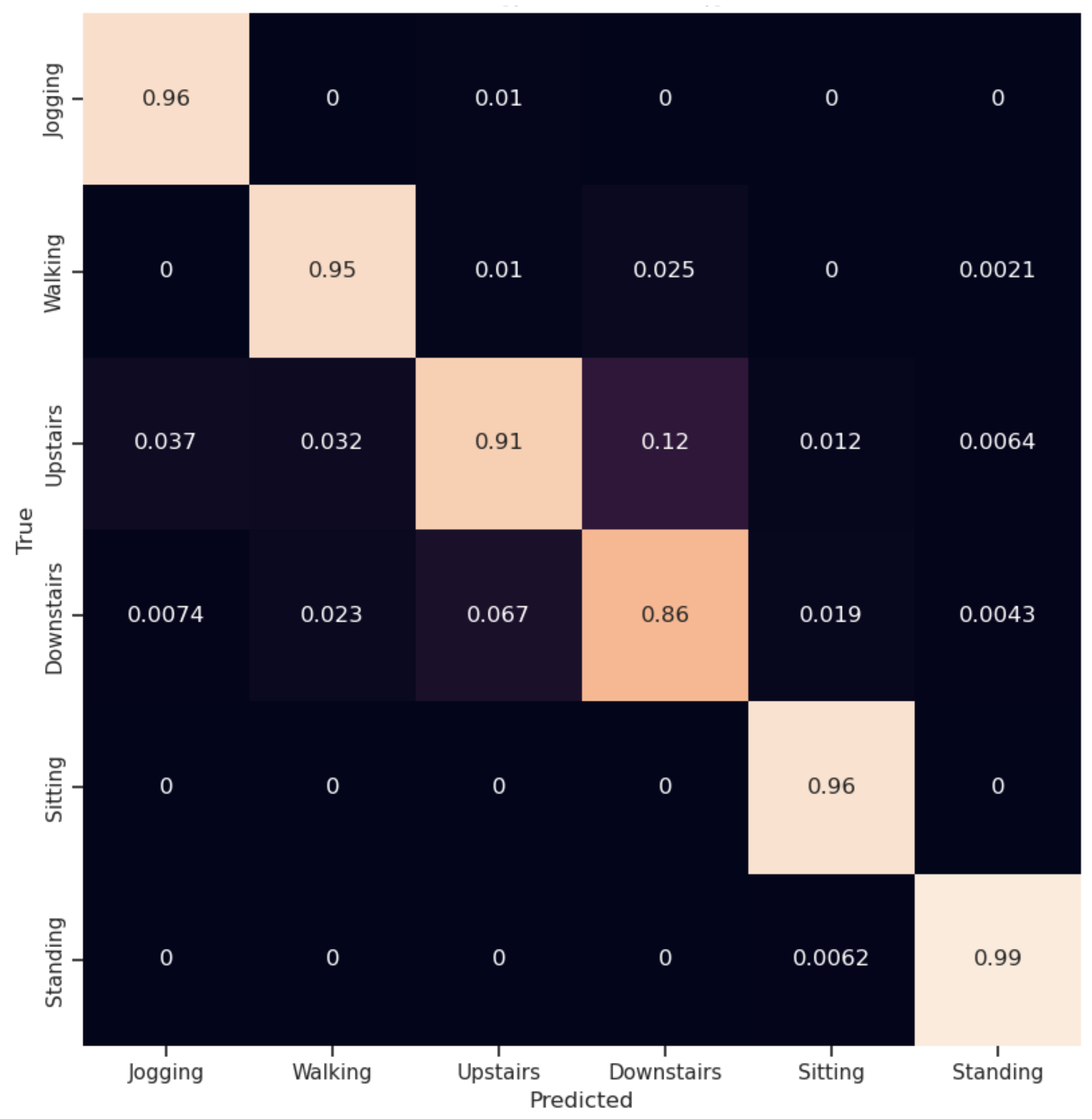
Gmean on test set: 0.9897

Precision on test set: 0.9817

Recall on test set: 0.9809

LSTM

Segmented data & with upsampling



Accuracy on test set: 0.9554

F1 score on test set: 0.9533

Gmean on test set: 0.9722

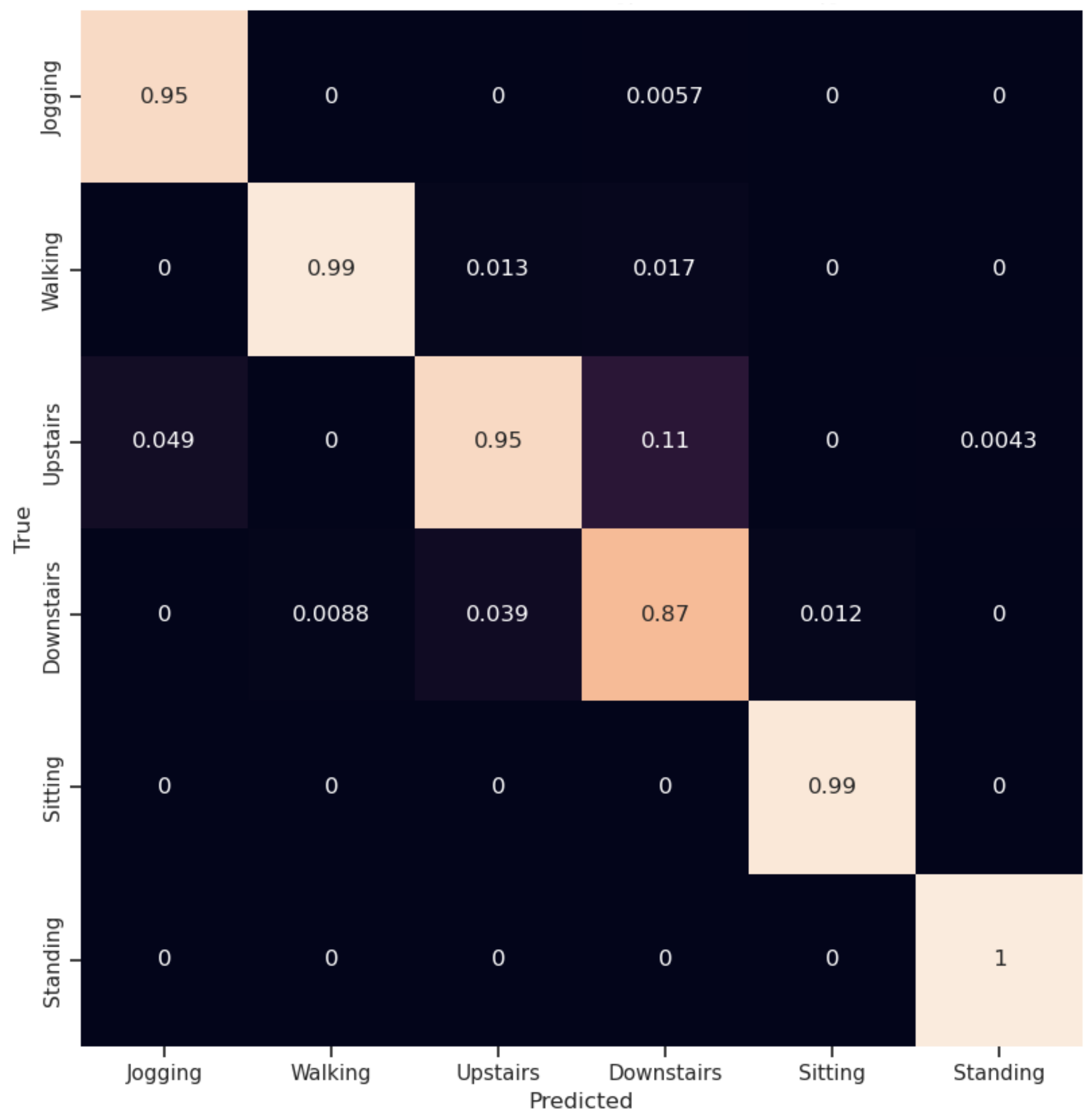
Precision on test set: 0.9541

Recall on test set: 0.9527

CNN-LSTM

Segmented data & with upsampling

CNNLSTMwithBatchNorm



Accuracy on test set: 0.9589

F1 score on test set: 0.9560

Gmean on test set: 0.9728

Precision on test set: 0.9544

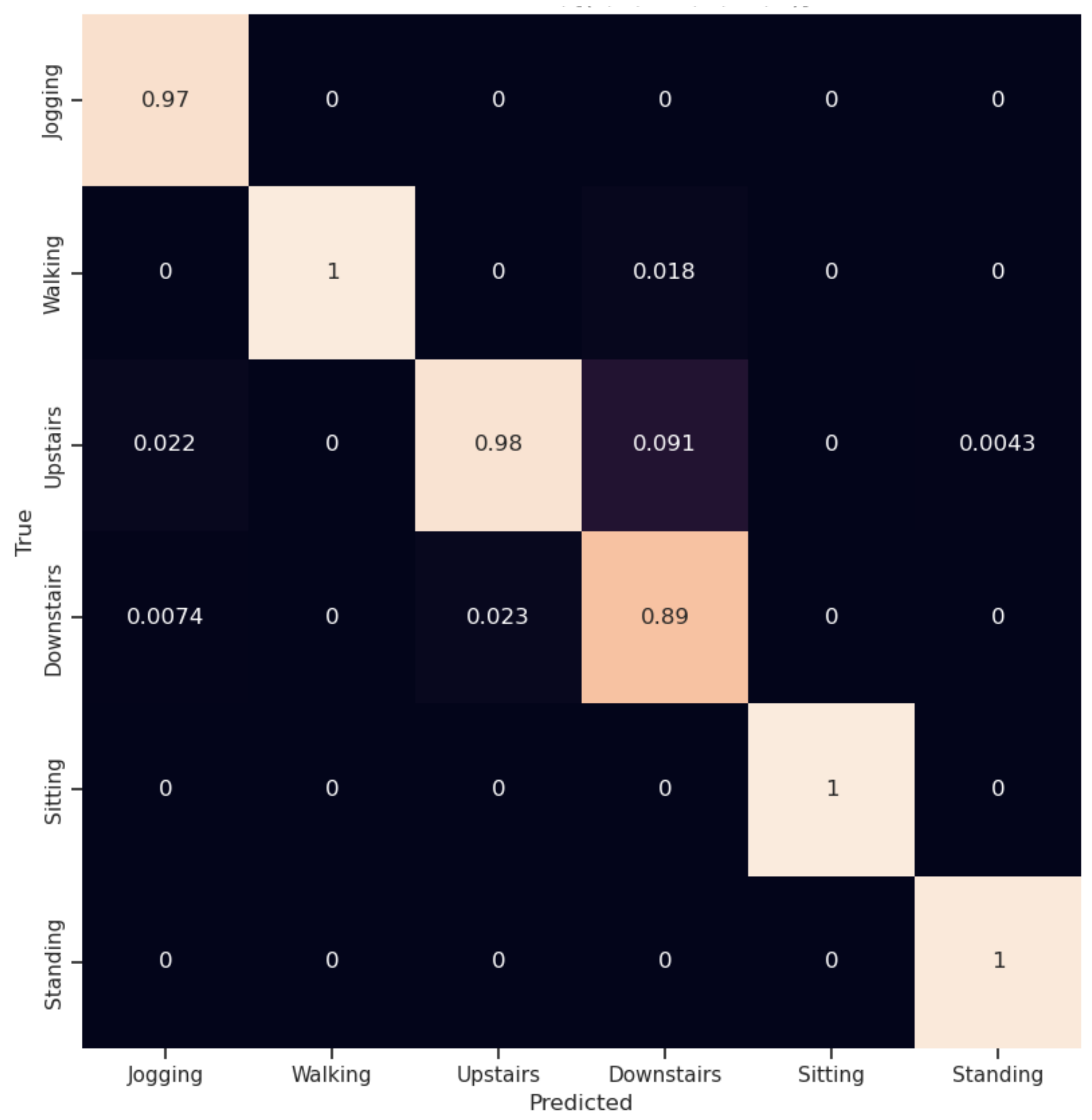
Recall on test set: 0.9577

Best Results:

CNN

Segmented data & with down-upsampling

CNNwithBatchNorm



<u>Accuracy</u> on test set: 0.9890

<u>F1 score</u> on test set: 0.9812

<u>Gmean</u> on test set: 0.9897

<u>Precision</u> on test set: 0.9817

<u>Recall</u> on test set: 0.9809

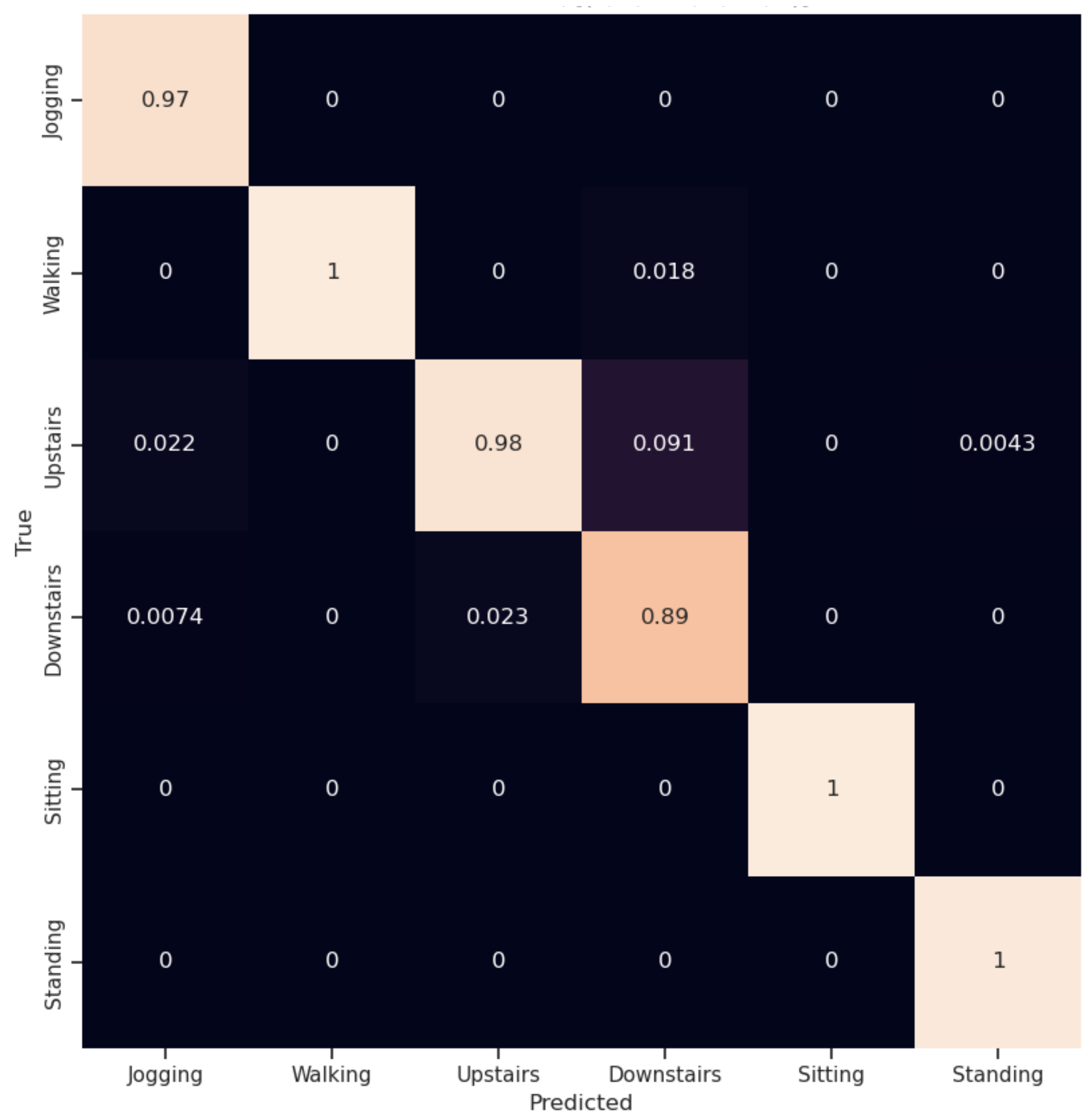
Best Results:

Best performance!

CNN

Segmented data & with down-upsampling

CNNwithBatchNorm



<u>Accuracy</u> on test set: 0.9890

<u>F1 score</u> on test set: 0.9812

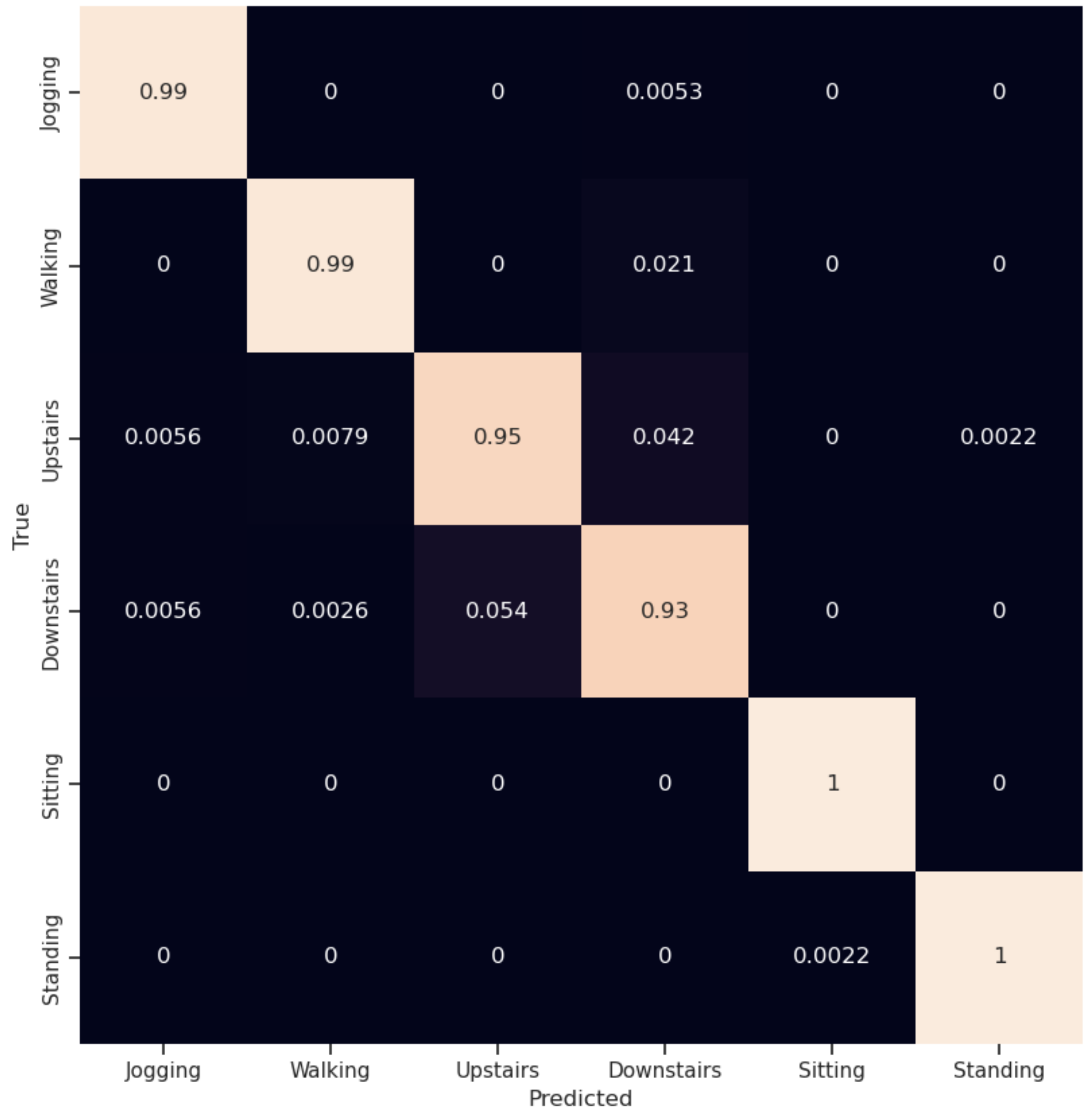
<u>Gmean</u> on test set: 0.9897

<u>Precision</u> on test set: 0.9817

<u>Recall</u> on test set: 0.9809

Transformer

Segmented data & with down-upsampling



<u>Accuracy</u> on test set: 0.9904

<u>F1 score</u> on test set: 0.9860

<u>Gmean</u> on test set: 0.9922

<u>Precision</u> on test set: 0.9862

<u>Recall</u> on test set: 0.9858

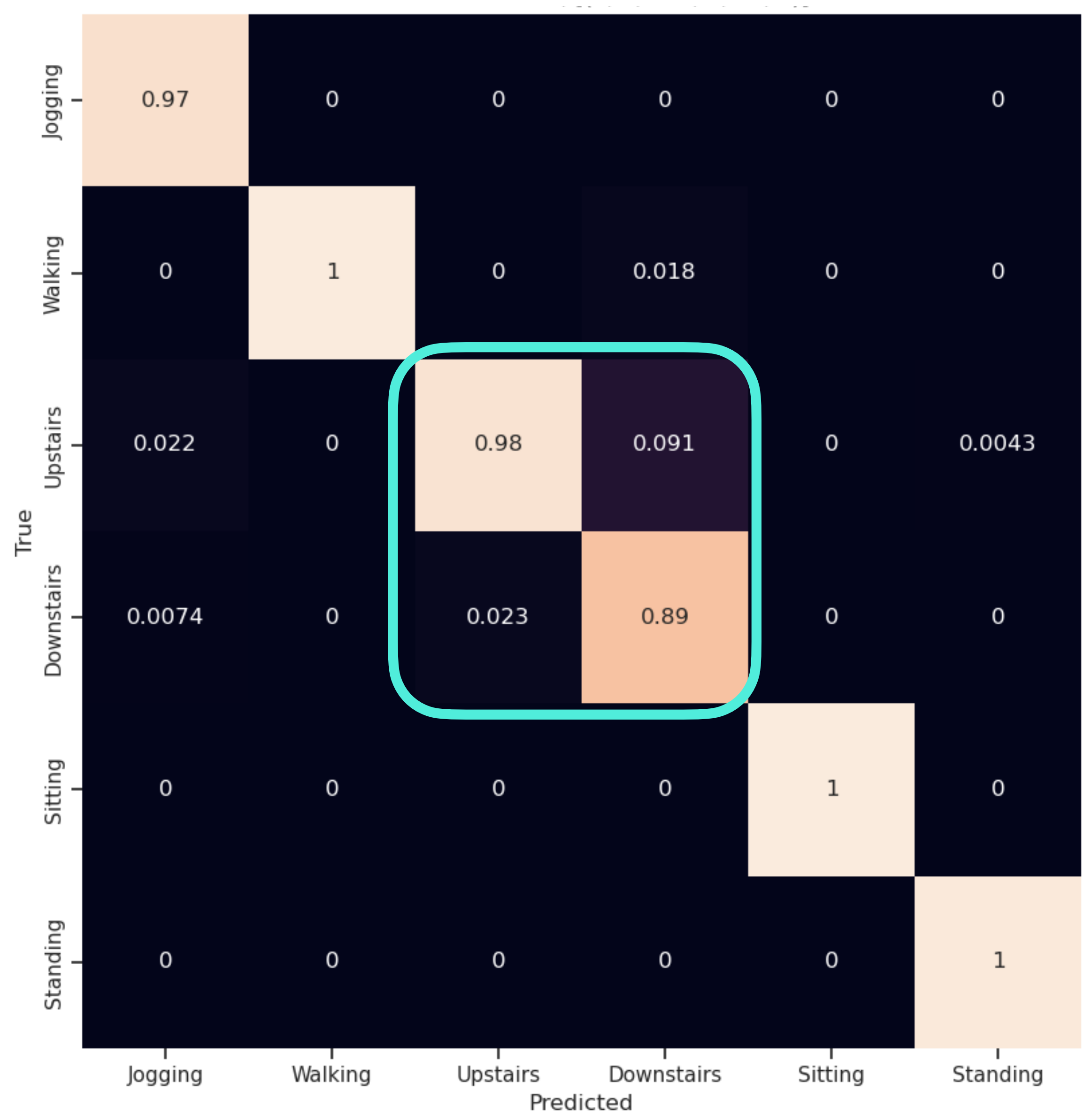
Best Results:

Best performance!

CNN

Segmented data & with down-upsampling

CNNwithBatchNorm



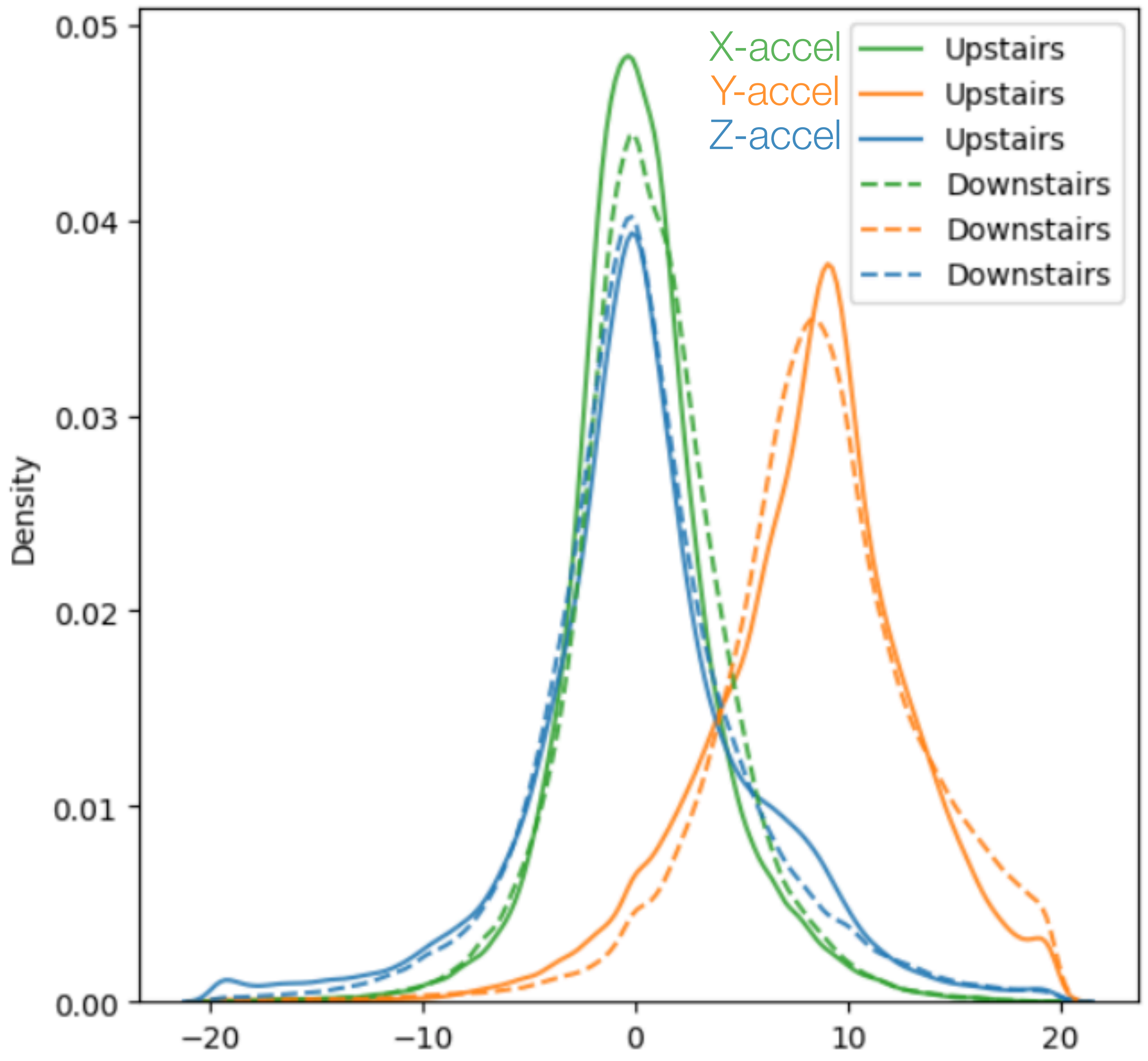
Accuracy on test set: 0.9890

F1 score on test set: 0.9812

Gmean on test set: 0.9897

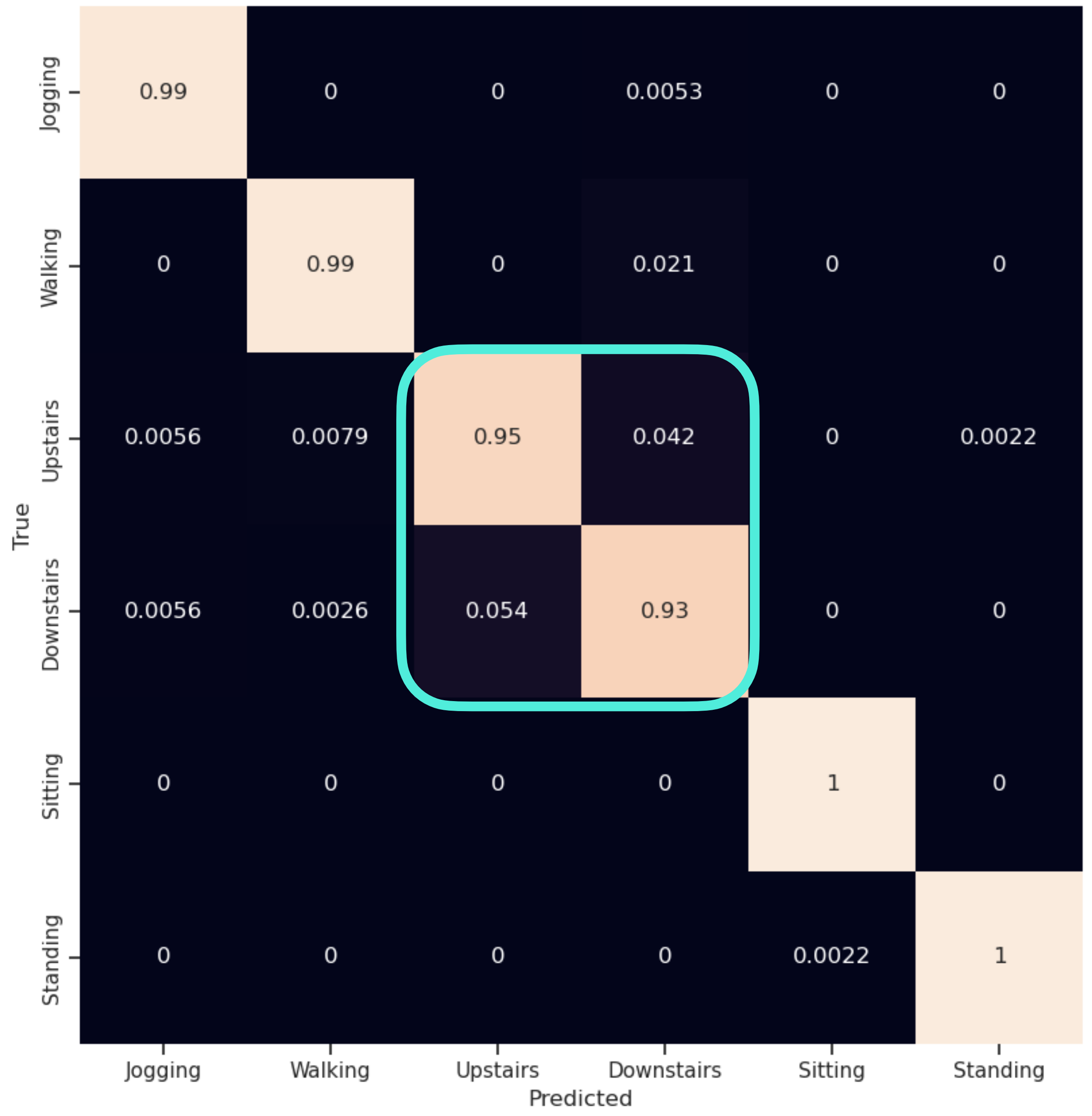
Precision on test set: 0.9817

Recall on test set: 0.9809



Transformer

Segmented data & with down-upsampling



Accuracy on test set: 0.9904

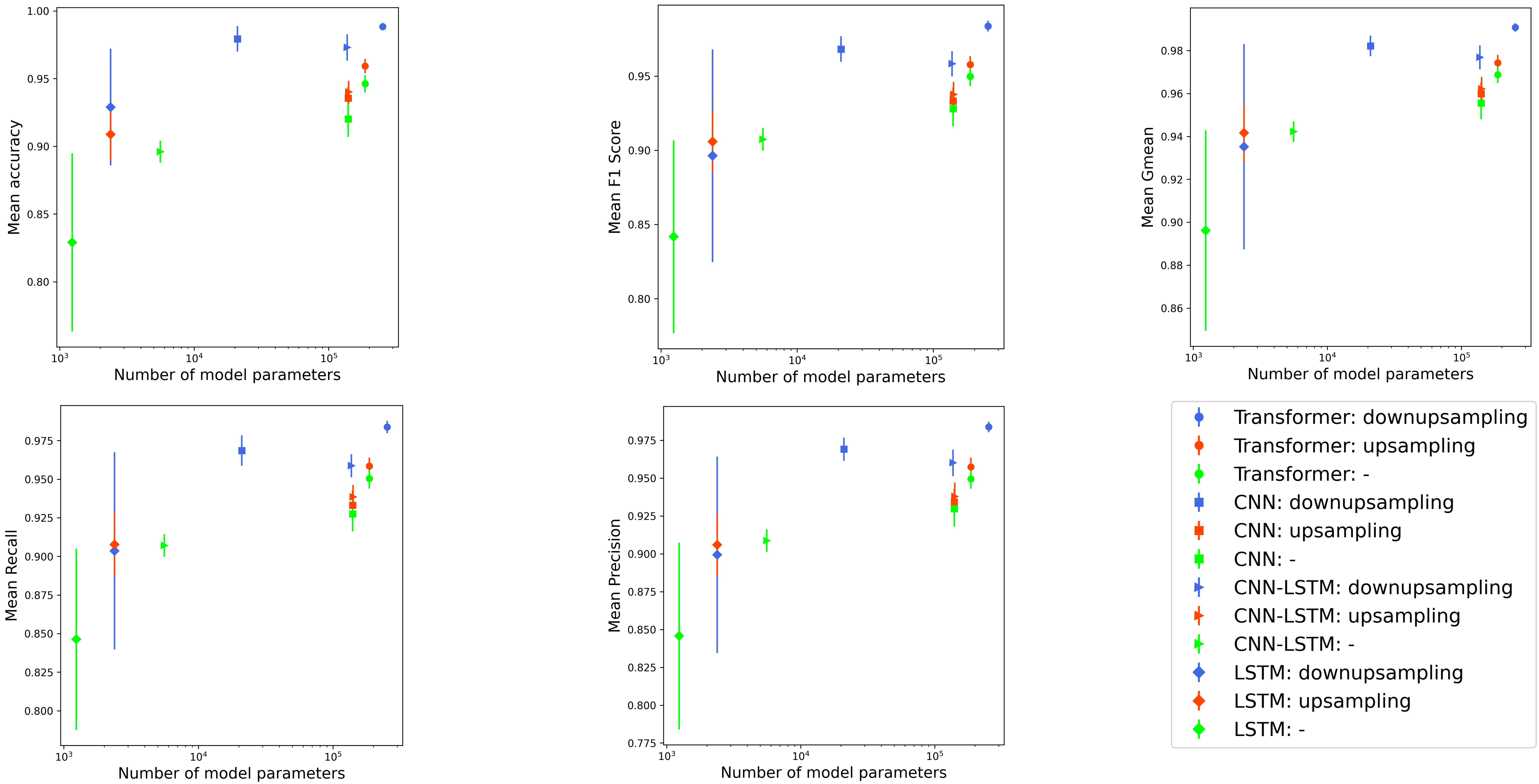
F1 score on test set: 0.9860

Gmean on test set: 0.9922

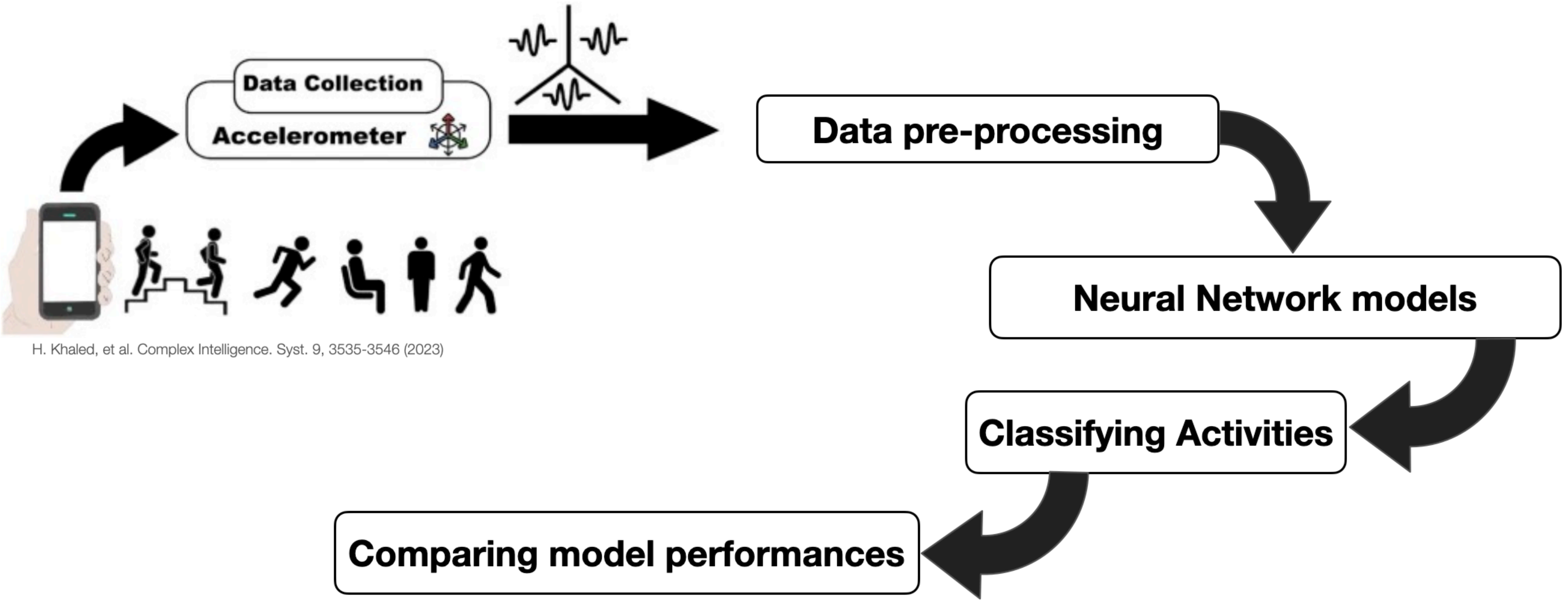
Precision on test set: 0.9862

Recall on test set: 0.9858

Averaging 20 metrics of the best-performing models



Summary



H. Khaled, et al. Complex Intelligence. Syst. 9, 3535-3546 (2023)

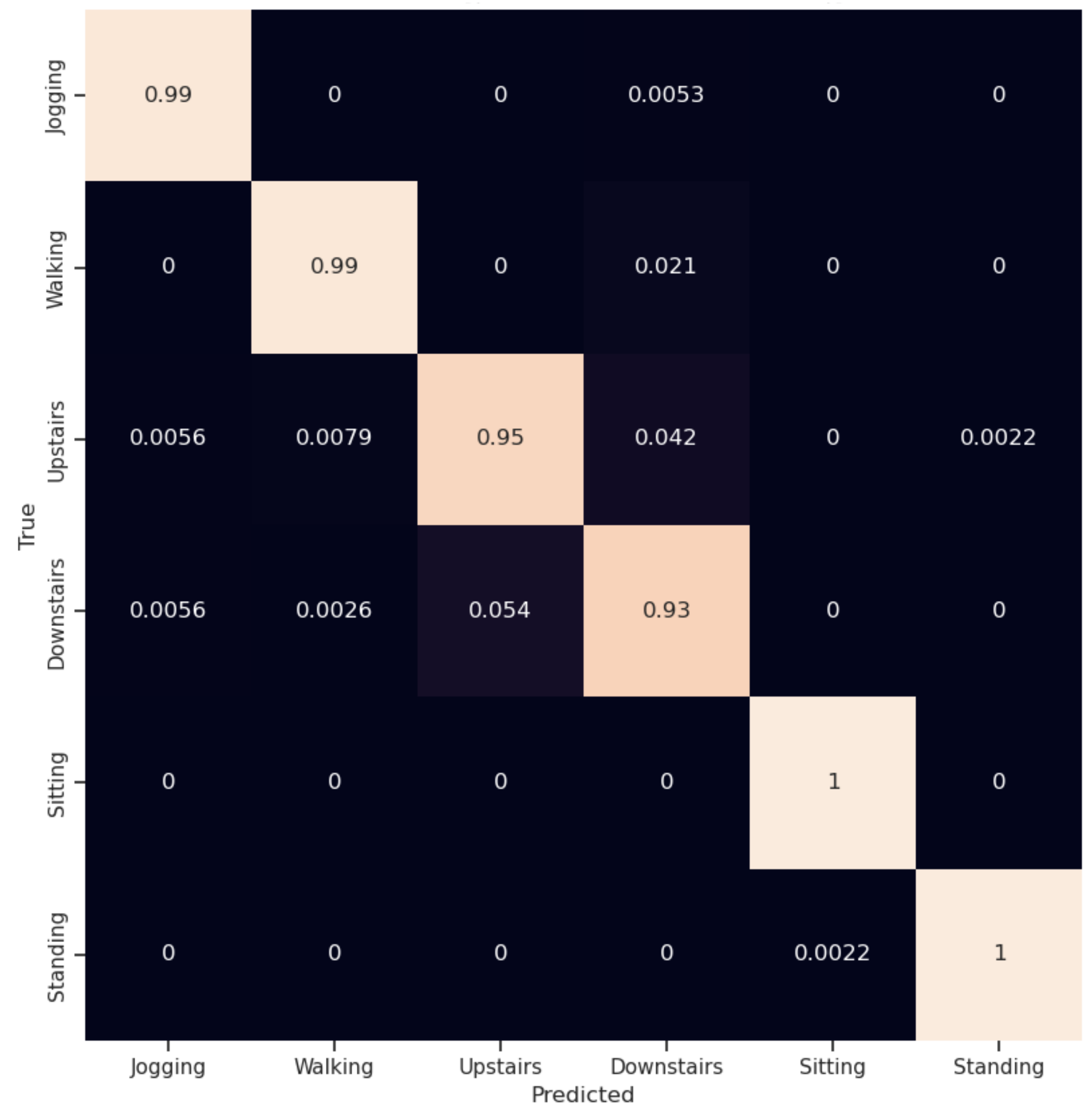
Employed upsampling and down-upsampling techniques

Implemented CNN, LSTM, CNN-LSTM & Transformer model

Best performance!

Transformer

Segmented data & with down-upsampling



Accuracy on test set: 0.9904

F1 score on test set: 0.9860

Gmean on test set: 0.9922

Precision on test set: 0.9862

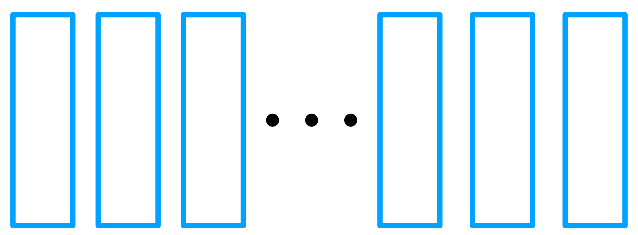
Recall on test set: 0.9858

Thank you for your attention!

Model architecture: MLP

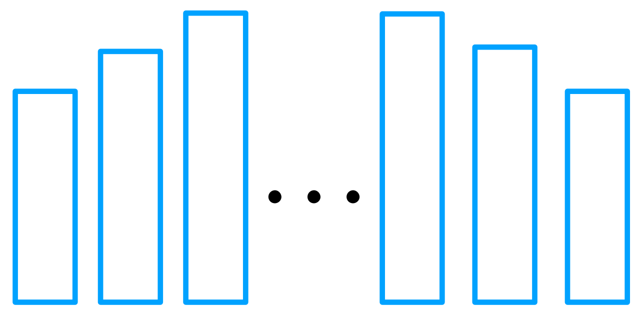
Two different models:

Mlp1



N hidden layers with size M

Mlp2



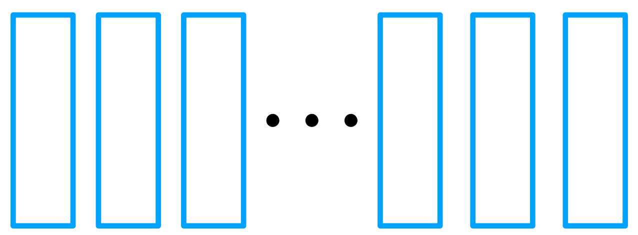
N hidden layers with a list of sizes for each layer

Model architecture: MLP

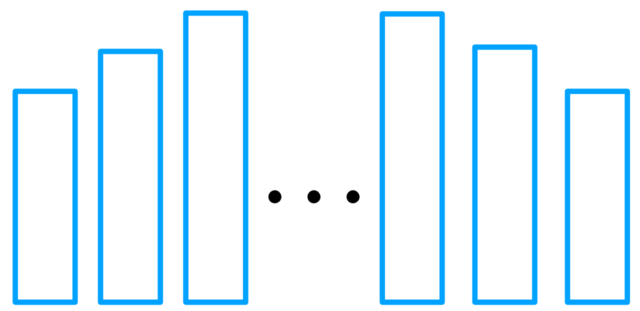
Mlp1

Mlp2

Two different models:



N hidden layers with size M



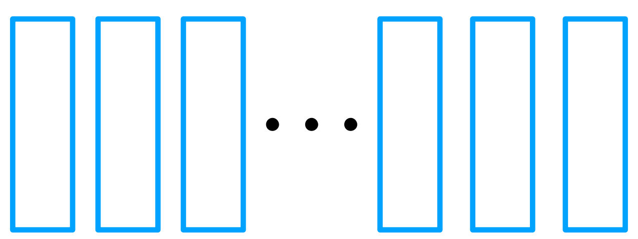
N hidden layers with a list of sizes for each layer

No time-ordering in input data

Model architecture: MLP

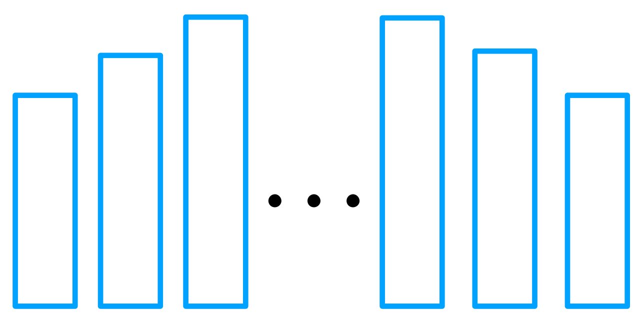
Two different models:

Mlp1



N hidden layers with size M

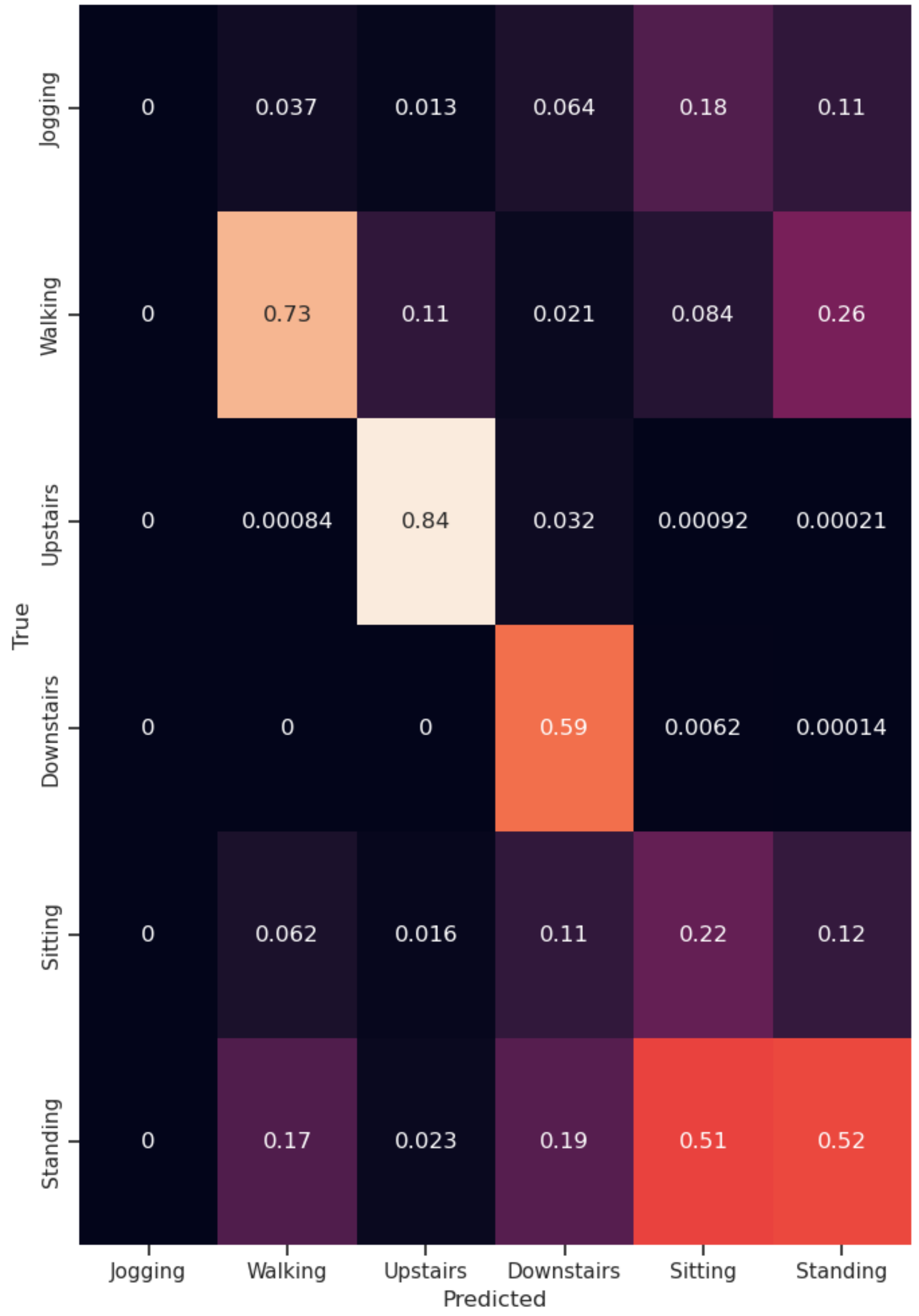
Mlp2



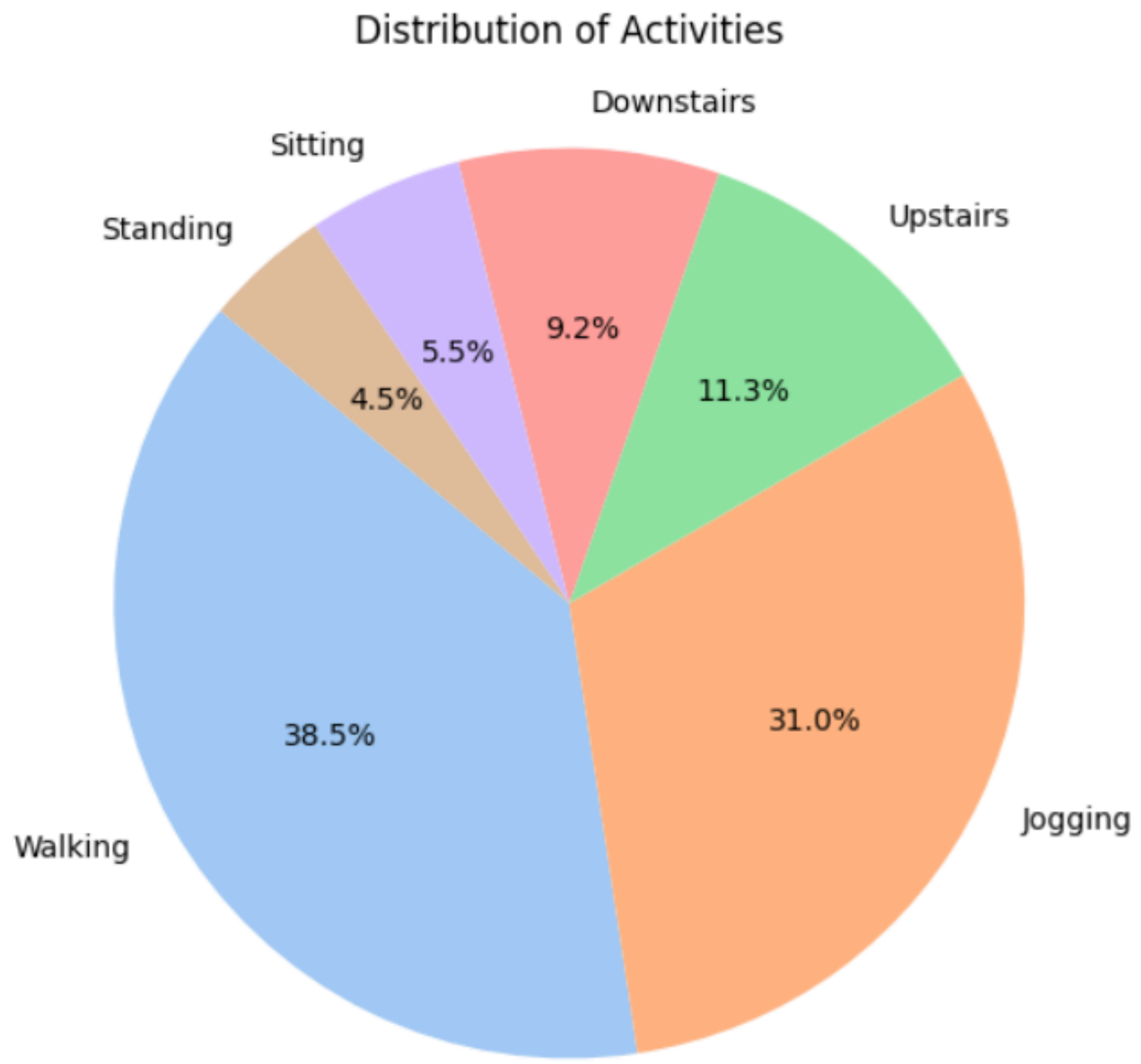
N hidden layers with a list of sizes for each layer

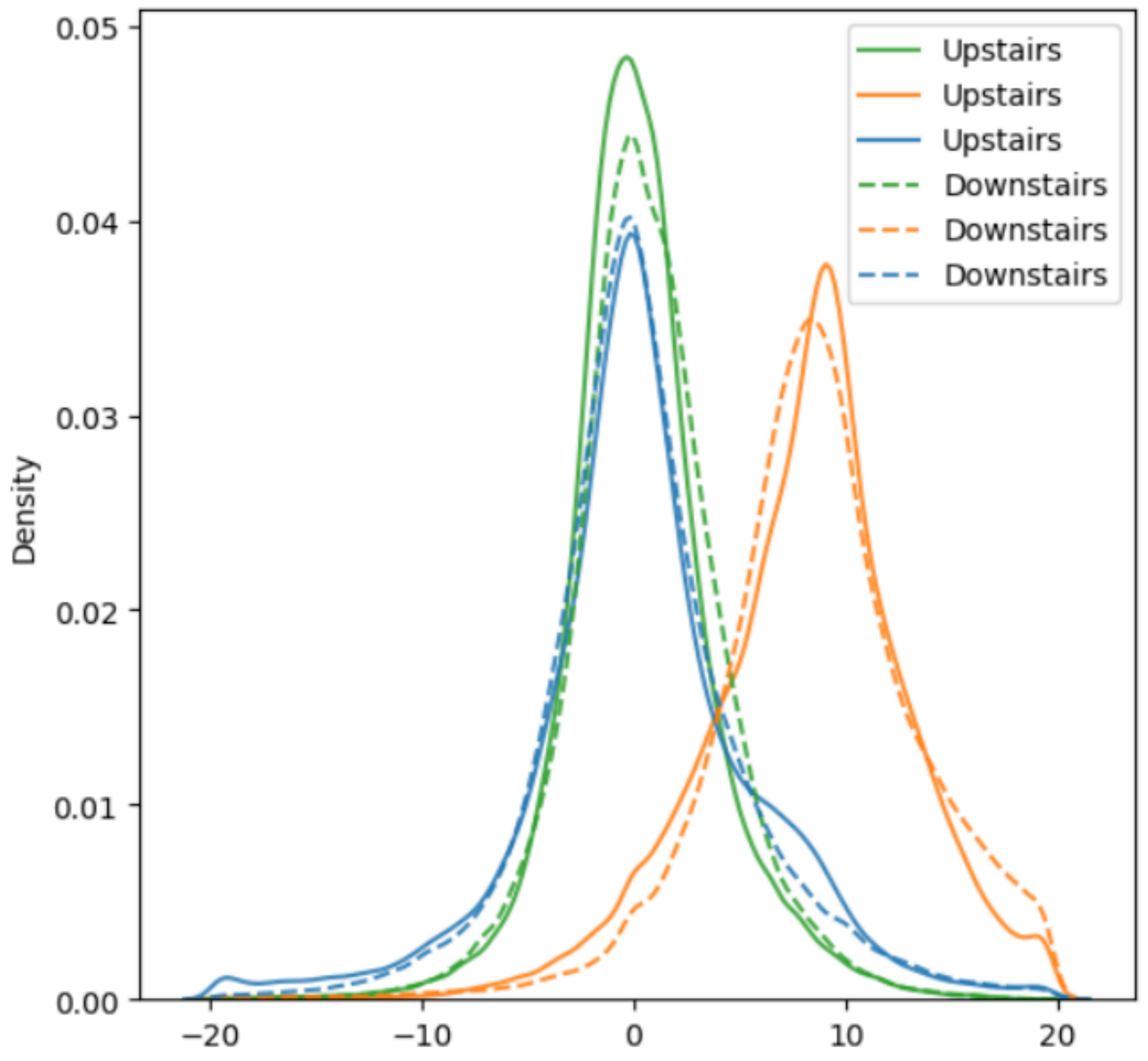
No time-ordering in input data

Batch size	Model type	Learning rate	# hidden layers	sizehidden layers	Accuracy	F1score
16	mlp2	0.0001	8	26	0.5679	0.2880
16	mlp2	0.0001	8	24	0.5669	0.3160
16	mlp2	0.0001	16	20	0.5621	0.2770
16	mlp1	0.0001	8	26	0.5512	0.2810
16	mlp1	0.0001	8	20	0.5429	0.3030



Distributions of samples for each activity





Distributions of samples for each activity