FastPoseGait: A Toolbox and Benchmark for Efficient Pose-based Gait Recognition

Shibei Meng[1*], Yang Fu[1*], Saihui Hou[1,2†], Chunshui Cao[2], Xu Liu[2], Yongzhen Huang[1,2†]
[1] School of Artificial Intelligence, Beijing Normal University [2] WATRIX.AI
{mengshibei, yangfu}@mail.bnu.edu.cn, {chunshui.cao, xu.liu}@watrix.ai
{housaihui, huangyongzhen}@bnu.edu.cn

Abstract

We present FastPoseGait, an open-source toolbox for pose-based gait recognition based on PyTorch. Our toolbox supports a set of cutting-edge pose-based gait recognition algorithms and a variety of related benchmarks. Unlike other pose-based projects that focus on a single algorithm, FastPoseGait integrates several state-of-the-art (SOTA) algorithms under a unified framework, incorporating both the latest advancements and best practices to ease the comparison of effectiveness and efficiency. In addition, to promote future research on pose-based gait recognition, we provide numerous pre-trained models and detailed benchmark results, which offer valuable insights and serve as a reference for further investigations. By leveraging the highly modular structure and diverse methods offered by FastPoseGait, researchers can quickly delve into pose-based gait recognition and promote development in the field. In this paper, we outline various features of this toolbox, aiming that our toolbox and benchmarks can further foster collaboration, facilitate reproducibility, and encourage the development of innovative algorithms for pose-based gait recognition. FastPoseGait is available at `https://github.com//BNU-IVC/FastPoseGait` *and is actively maintained. We will continue updating this report as we add new features.*

1. Introduction

Pose-based gait recognition [1, 15, 16, 25, 26, 29] aims to identify individuals based on their gait patterns captured through human pose sequences, even from a distance. Unlike the silhouette modality [3, 5, 9, 10, 14, 17], poses offer keypoint representations that are more robust to changes in appearance. As a result, pose-based gait recognition has gained significant interest in recent years. Several algorithms have been developed to achieve pose-based gait recognition, including methods based on Graph Convolutional Networks (GCN) [25, 26] and Transformer [29].

GaitGraph [26] was the pioneering work that introduced Graph Convolutional Networks (GCN) [21] into pose-based gait recognition. This novel approach demonstrated a notable improvement in performance compared to previous methods [1, 16]. Building upon the success of GaitGraph, GaitGraph2 [25] was presented to further enhance the performance of pose-based methods, particularly on the large dataset of OUMVLP-Pose [1]. GaitTR [29] introduced the use of Transformer [19] to improve the performance even further by leveraging the global relationship modeling capabilities of self-attention. For instance, on the CASIA-B benchmark, the rank-1 accuracy has already achieved up to 90.0% in CL conditions.

However, these existing advanced methods are conducted on only one or two benchmarks, lacking a comprehensive comparison across indoor and outdoor scenarios and datasets of varying scales. Furthermore, none of the existing approaches have compared these architectures under a unified setting, hindering the ability to make direct performance assessments. Toward the goal of addressing these limitations and offering a well-structured, highly modular and easily understandable codebase for researchers, we have built FastPoseGait. This toolbox offers several major features to enhance its usability and effectiveness:

(1) Alignment of experimental settings. FastPoseGait ensures a unified experimental setting, enabling quick reproduction of results and fair performance comparisons among different algorithms. This alignment facilitates efficient research and evaluation.

(2) Modular architecture. We decompose the gait recognition framework into distinct components, allowing users to easily construct a customized gait recognition algorithm by combining different modules.

(3) Support of multiple algorithms and benchmarks. FastPoseGait supports a range of cutting-edge gait recognition approaches and benchmarks with diverse research interests and different scales.

*Equal contribution
†Corresponding Author

(4) Availability of a large number of trained models and detailed results. The toolbox contains a large-scale model zoo and corresponding comprehensive results on different datasets to help researchers acquire a comprehensive understanding of pose-based gait recognition in an efficient manner.

(5) High efficiency. The code described in this report is compatible with multi-GPU setups utilizing Distributed Data Parallel (DDP)[1]. Additionally, it supports model training with Auto Mixed Precision (AMP)[2]. These features are implemented to accelerate the training process.

2. Supported Frameworks and Datasets

2.1. Supported Frameworks

GaitGraph [26] adopts the ResGCN model [21] as a backbone, which is originally used in action recognition. By regarding the human pose as a graph and employing graph convolution operations, the proposed approach achieves notable enhancements in accuracy compared to earlier methods for pose-based gait recognition [15, 16]. Furthermore, GaitGraph utilizes Supervised Contrastive Loss [11] for optimization.

GaitGraph2 [25] is built upon GaitGraph and introduces a significant improvement by incorporating a multi-input mechanism. Additionally, certain modifications have been made to the number of layers in order to better accommodate the scale of datasets. Supervised Contrastive Loss [11] is employed in this method with a much larger batch size compared with GaitGraph.

GaitTR [29] is the first algorithm to introduce Spatial Transformer [19] into pose-based gait recognition, enabling the establishment of comprehensive spatial relationships. This algorithm has achieved state-of-the-art performance on the CASIA-B benchmark [28]. Triplet Loss [8] is adopted to optimize the training procedures.

2.2. Supported Datasets

CASIA-B [28] consists of 124 subjects under three distinct walking conditions. 11 sequences of each subject on every walking condition are captured from different viewpoints. HRNet [23] and SimCC [13] are employed to extract pose data of 17 keypoints as shown in Figure 2 (a).

OUMVLP-Pose [1] is built on the large-scale OUMVLP dataset [24], which comprises 10,307 subjects from 14 viewpoints. OUMVLP-Pose provides 18-keypoint pose data shown in Figure 2 (b) extracted by AlphaPose [6] and OpenPose [2].

GREW [31] is a large-scale dataset including 26,345 individuals captured by 882 cameras in real-world settings. GREW provides pose information comprising 17 keypoints, which is obtained using HRNet pose estimator [23].

Gait3D [30] is an in-the-wild dataset collected by 39 cameras from a fixed real-world environment, consisting of 4,000 subjects. Gait3D also includes pose information of 17 keypoints extracted by HRNet pose estimator [23].

3. Architecture

Figure 1 illustrates the complete architecture of FastPoseGait, containing four distinct modules. Initially, **Sampler** selects sequences of skeletons, which are then preprocessed with the **Data Transform** module. **Model** encodes the inputs and is composed of various backbones. These backbones are comprised of blocks, which in turn consist of units with either residual connections or combinations of spatial and temporal convolutional layers. Different **Loss** functions are implemented to supervise the training process. Furthermore, FastPoseGait incorporates specific scripts that guarantee the proper formatting of data, including the required file structures and keypoint orders. TensorBoard[3] is utilized to conveniently visualize training logs and keypoint heatmaps, facilitating better record-keeping of the model optimization.

3.1. Sampler

Random Sampler Random sampler is utilized in GaitGraph [26] and GaitGraph2 [25] which adopt the supervised contrastive loss [11]. It randomly selects sequences from the whole dataset without considering the proportion of the positive and negative samples, allowing for a more diverse representation of data in one batch during training.

Triplet Sampler Triplet sampler is used in GaitTR [29] which employs the triplet loss [8] to supervise the training process. The batch size of this sampler is defined as (P, K), in which P refers to the number of subjects in a batch and K is the sequence number of every subject. This sampling strategy ensures a relatively balanced proportion of positive and negative samples.

Random Triplet Sampler Random Triplet Sampler is a *trade-off* of Random Sampler and Triplet Sampler. To ensure the presence of both positive and negative pairs in a single batch, we propose the following constraints on P and K:

$$\begin{cases} P \geqslant 2, \\ K \geqslant 2, \\ PK + c = BatchSize, \end{cases} \tag{1}$$

where $BatchSize$ is the total number of sequences in a batch. c denotes the remainder when $BatchSize$ is divided

[1] https://pytorch.org/tutorials/intermediate/ddp_tutorial.html

[2] https://pytorch.org/tutorials/recipes/recipes/amp_recipe.html

[3] https://www.tensorflow.org/tensorboard

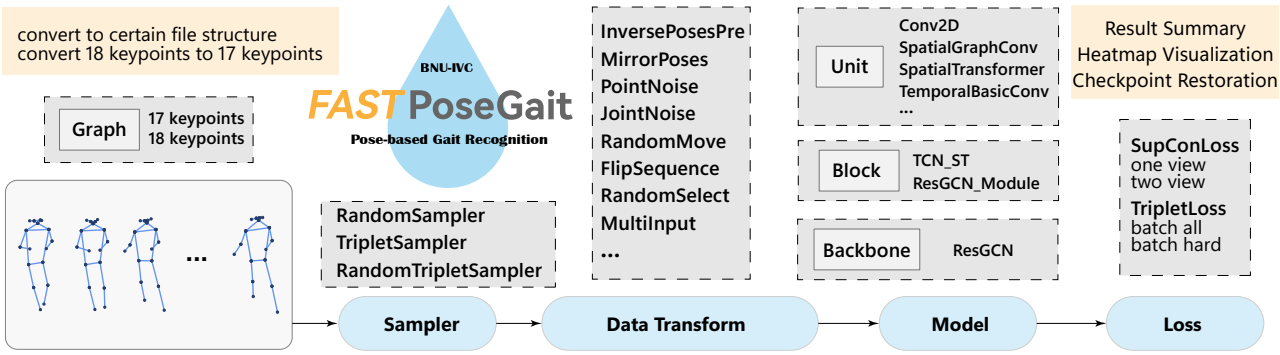

Figure 1: The Pipeline and Structure of FastPoseGait.

by P. After sampling P subjects with K sequences, c sequences are randomly selected from the remaining training set. Additionally, P and K can be further limited within an upper bound based on the characteristics of the dataset. *e.g.*, the training set of CASIA-B consists of 74 subjects and each subject is with 110 sequences, so $P \leqslant 74$ and $K \leqslant 110$.

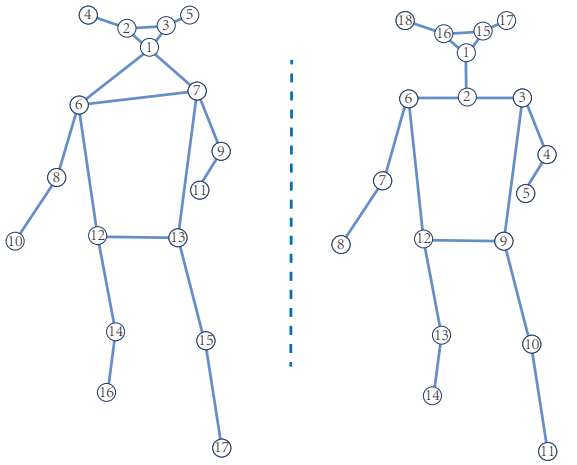

Figure 2: The illustration of keypoint orders and pre-defined connections.

3.2. Data Transform

Spatial Augmentations We implement five commonly employed spatial augmentations in FastPoseGait. They can be divided into two categories:

(1) Spatial Flip. (i) Pre-defined Inverse: It adopts the pre-defined symmetrical structure of keypoints to perform the left-right flipping operations. *e.g.*, in Figure 2 (a), we exchange the coordinate values of ⑥ and ⑦, ⑧ and ⑨, *etc*. (ii) Mirror Poses: The *centroid* coordinate values are acquired by calculating the mean of all keypoints in a skeleton graph. Then the symmetry points relative to these centroid values are obtained to mirror the skeletons.

(2) Spatial Noise. This type of transformation introduces random Gaussian noise at different levels, namely the point level, temporal level, and the entire sequence level.

Sequence Augmentations We implement two kinds of sequence-level augmentations: (1) Sequence Flip. It is to reverse the order of frames in a sequence. (2) Random Selection: It is to randomly select a fixed number of ordered frames from the entire sequence.

Multi-Input There are mainly four types of explicitly calculated input in action recognition [22], including joint, bone, angle and velocity. Various combinations are taken as input in different methods [25, 29].

3.3. Design of Model Framework

Graph Graph is the priori knowledge of human skeletons. It defines adjacency matrices of skeleton topologies and symmetrical structures of human keypoints. FastPoseGait supports the 17-keypoint and 18-keypoint graphs based on different pose estimators as shown in Figure 2. The pre-defined graph plays a significant role in defining the relationships and connectivities between keypoints, facilitating effective graph convolutions.

Unit Unit is a collection of basic layers. Different methods incorporate distinct types of basic layers, which encompass both spatial and temporal layers. *e.g.*, Graph Convolutional layer [27], Spatial Transformer layer [19], Temporal Convolutional layer [27], *etc*.

Block Block typically is comprised of one or two basic layers, representing a self-contained module repeated within a model backbone. It can consist of either a combination of a spatial and temporal layer or simply include a basic layer with residual connections. The block serves as a fundamental building component in the model architecture, enabling the flexible composition of these modules to capture complex spatial and temporal relationships.

Backbone Backbone is a model structure that is responsible for extracting discriminative features from transformed

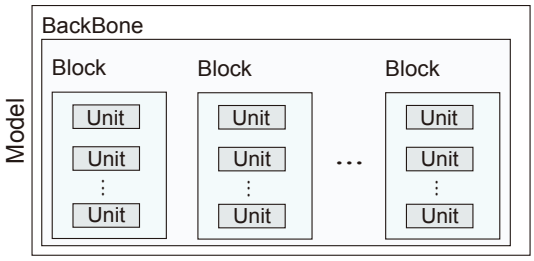

Figure 3: Framework of the model, illustrated with abstractions in FastPoseGait.

inputs. For instance, ResGCN [21] composed of multiple blocks with residual connections is employed as the backbone in both GaitGraph and GaitGraph2.

Model Model shown in Figure 3, serves as a method-specific definition of graphs, units, blocks and backbones. It outlines how these components are structured and combined to create an integrated framework to ensure consistency and effectiveness to achieve the desired objective.

3.4. Loss Function

Triplet Loss Triplet Loss [8] is employed in GaitTR [29] to decrease the intra-class distance and increase the inter-class distance of different identities. It can be divided into two types, one is named Batch All, in which all the triplets in a batch are considered:

$$L_{BA} = \frac{1}{N_{tri+}} \overbrace{\sum_{i=1}^{P}\sum_{a=1}^{K}}^{\text{all anchors}} \overbrace{\sum_{\substack{p=1\\p\neq a}}^{K}}^{\text{all pos.}} \overbrace{\sum_{\substack{j=1\\j\neq i}}^{P}\sum_{n=1}^{K}}^{\text{all neg.}} [m + d_{i,a,p}^{j,a,n}], \quad (2)$$

$$d_{i,a,p}^{j,a,n} = \mathrm{D}(f_i^a, f_i^p) - \mathrm{D}(f_i^a, f_j^n), \quad (3)$$

where the total number of triplets in a batch is $PK(PK-K)(K-1)$, N_{tri+} refers to the number of triplets with non-zero loss values, m is the margin threshold between positive and negative pairs. $\mathrm{D}(\cdot)$ is the calculation of distance, *e.g.*, Euclidean Distance, and f is final outputs of models.

The other type of triplet loss is named Batch Hard. For each anchor sequence within a batch, only the positive pair with the maximum distance is considered, as well as the negative pair with the minimum distance:

$$L_{BH} = \frac{1}{N_{tri+}} \overbrace{\sum_{i=1}^{P}\sum_{a=1}^{K}}^{\text{all anchors}} [m + d_{i,a}], \quad (4)$$

$$d_{i,a} = \underbrace{\max_{p=1\ldots K} (\mathrm{D}(f_i^a, f_i^p))}_{\text{hardest positive}} - \overbrace{\min_{\substack{j=1\ldots P\\n=1\ldots K\\j\neq i}} (\mathrm{D}(f_i^a, f_j^n))}^{\text{hardest negative}}, \quad (5)$$

where the number of generated hard triplets of L_{BH} in a batch is PK.

Supervised Contrastive Loss Supervised Contrastive Loss is originally proposed in [11] to handle the case that more than one sample is known as positive due to the presence of labels. It offers improved performance by avoiding the requirement for explicit hard mining, which is a sensitive yet crucial aspect of various loss functions like triplet loss:

$$L_{sup} = \sum_{i\in I} -\log \left\{ \frac{1}{|S(i)|} \sum_{s\in S(i)} \frac{\exp(f_i \cdot f_s/\tau)}{\sum\limits_{a\in A(i)} \exp(f_i \cdot f_a/\tau)} \right\}. \quad (6)$$

In one batch, I represents all the samples, while $A(\cdot)$ represents all the samples except the one at index i. $S(\cdot)$ represents the positive samples with the same label as i-th index, and $|S(\cdot)|$ denotes its cardinality. Specifically, in *one view* settings of GaitGraph2 [25], set I contains N elements, while in *two view* settings of GaitGraph [26], it contains $2N$ elements. The additional N elements are generated by different data augmentations.

4. Benchmark Results

4.1. Overall Experimental Settings

The initial learning rates for GaitGraph, GaitGraph2, and GaitTR are 0.01, 0.05, and 0.001, respectively, following the OneCycle [20] learning rate scheduler. GaitGraph2 on CASIA-B and Gait3D employ AdamW [18] and others utilize Adam [12] as the optimizer. All the models are trained and evaluated on four TITAN Xp 12G GPUs. More details about the sampler, batch size and model structure can be seen in the following sections.

Table 1: The batch size configurations in Vanilla Version.

Model	CASIA-B	OUMVLP-Pose	GREW	Gait3D
GaitGraph [26]	128	128	128	128
GaitGraph2 [25]	768	768	768	768
GaitTR [29]	(4,64)	(32,16)	(32,8)	(32,4)

4.2. Vanilla Version

For Vanilla Version, our goal is to maintain consistency with the original codebases or descriptions in the papers to the greatest extent possible. This involves preserving identical types of model structures, data augmentations, samplers, and batch sizes described in the original papers [25, 26, 29]. **(1)** Data Augmentations. As shown in Table 2, we employ the same types of data augmentations with original codebases and papers in Vanilla Version. **(2)** Samplers. Random Sampler is adopted in GaitGraph and GaitGraph2, while GaitTR utilizes Triplet Sampler. **(3)** Batch Sizes. Three models utilize different batch sizes as described in the papers, shown in Table 1.

Table 2: The data augmentation configurations in Vanilla Version.

	Spatial					Sequential	
Model	InversePosesPre	MirrorPoses	PointNoise	JointNoise	RandomMove	RandomSelectSequence	FlipSequence
GaitGraph [26]		✓	✓	✓		✓	✓
GaitGraph2 [25]	✓		✓	✓	✓	✓	✓
GaitTR [29]	✓		✓	✓		✓	

Table 3: Rank-1 (%) performance of three methods on CASIA-B. The results in parentheses are mentioned in the papers.

Version		Vanilla				Improved			
Model	Pose Estimator	NM	BG	CL	Mean	NM	BG	CL	Mean
GaitGraph [26]	HRNet [23]	86.37 (87.70)	76.50 (74.80)	65.24 (66.30)	76.04 (76.27)	88.47	77.52	67.95	77.98
GaitGraph2 [25]	HRNet [23]	80.29 (82.00)	71.40 (73.20)	63.80 (63.60)	71.83 (72.93)	83.60	72.80	67.01	74.47
GaitTR [29]	HRNet [23]	94.72	89.29	86.65	90.22	95.55	89.79	85.76	90.37
GaitTR [29]	SimCC [13]	94.91 (96.00)	88.82 (91.30)	90.34 (90.00)	91.35 (92.40)	95.02	90.70	89.67	91.80

Table 4: Rank-1 (%) performance of three methods on OUMVLP-Pose estimated by AlphaPose [6], excluding the identical-view cases.

Probe	Gallery(0°- 270°)					
	Vanilla			Improved		
	GaitGraph [26]	GaitGraph2 [25]	GaitTR [29]	GaitGraph [26]	GaitGraph2 [25]	GaitTR [29]
0°	1.88	53.19	29.59	41.27	54.88	33.04
15°	3.26	66.56	43.22	53.19	67.55	46.64
30°	4.01	71.04	47.77	57.69	71.98	50.96
45°	4.56	72.03	51.45	59.26	73.80	53.61
60°	3.66	67.10	50.81	58.19	70.80	53.41
75°	3.25	64.90	45.80	58.49	68.74	48.99
90°	2.31	62.35	39.47	55.18	65.62	42.99
180°	1.88	47.71	25.56	38.62	49.51	29.38
195°	2.69	56.41	33.50	43.99	58.71	37.29
210°	1.83	52.74	30.48	42.09	54.29	34.78
225°	3.08	67.41	46.10	54.45	69.70	48.49
240°	2.60	65.27	45.01	53.42	67.55	48.19
255°	2.30	63.93	40.05	52.47	66.98	44.64
270°	2.02	58.91	35.47	49.00	63.28	38.09
Mean	2.81	62.11	40.30	51.24	64.53	43.61

Table 5: Rank-1 (%), Rank-5 (%), Rank-10 (%) performance of three methods on GREW.

Version		Vanilla			Improved		
Model	Pose Estimator	Rank-1	Rank-5	Rank-10	Rank-1	Rank-5	Rank-10
GaitGraph [26]	HRNet [23]	10.18	19.57	24.73	36.08	56.69	64.89
GaitGraph2 [25]	HRNet [23]	34.78	49.69	55.51	44.41	59.04	64.69
GaitTR [29]	HRNet [23]	48.58	65.48	71.58	55.33	71.40	76.78

Table 6: Rank-1 (%), Rank-5 (%), Rank-10 (%) performance of three methods on Gait3D.

Version		Vanilla			Improved		
Model	Pose Estimator	Rank-1	Rank-5	Rank-10	Rank-1	Rank-5	Rank-10
GaitGraph [26]	HRNet [23]	8.6	17.3	23.8	14.6	31.3	38.8
GaitGraph2 [25]	HRNet [23]	11.2	24.0	31.2	12.5	24.7	30.6
GaitTR [29]	HRNet [23]	7.2	15.5	20.5	9.7	21.8	28.4

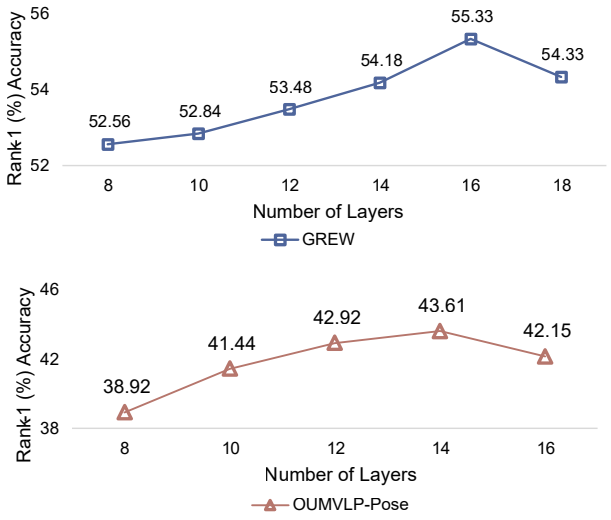

Figure 4: Impact of layer numbers on GaitTR.

The results can be seen in Tables 3 to 6. We achieve satisfactory results on CASIA-B in comparison to the results mentioned in the papers, with a maximum deviation of less than 2% in terms of Rank-1 accuracy. On OUMVLP-Pose, GREW and Gait3D, the reproduced algorithms still yield a relatively stable performance across indoor and outdoor scenarios and varying dataset scales, especially for GaitGraph2 and GaitTR.

4.3. Improved Version

Although Vanilla Version of the three methods demonstrates satisfactory performance, there is still plenty of room for improvement: **(1)** As shown in Tables 3 to 6, some results of GaitGraph are drastically inconsistent on untested benchmarks, *e.g.*, Rank-1 accuracy of only 2.81% on OUMVLP-Pose and 10.18% on GREW. **(2)** Pose-based gait recognition is sensitive to data augmentations, and thus original settings can be unsuitable for datasets with diverse scenarios. **(3)** Deep network structures have been proved to be effective in both deep learning [7] and appearance-based gait recognition [4]. Therefore, a deeper network is necessary to achieve optimal performance for larger-scale datasets. Taking these factors into account, we have introduced an Improved Version of FastPoseGait to fully unleash the model's potential on all four benchmarks. This version incorporates a more unified experimental setup and provides a comprehensive analysis of samplers, data augmentations, and model structures.

(1) Sampler. To tackle the issue of selecting positive pairs in GaitGraph and GaitGraph2, we have replaced Random Sampler with Triplet Sampler. This modification prevents the supervised contrastive loss from degrading to an unsupervised one, maintaining the integrity of the loss function's gradient structure. Additionally, to address overfitting on CASIA-B for GaitGraph, we utilize the Random Triplet Sampler described in Section 3.1. As shown in the preliminary study of Table 7, the performance of GaitGraph on OUMVLP-Pose can be improved by up to 48.4%.

Table 7: The impact of different sampler strategies in GaitGraph on the OUMVLP-Pose dataset.

Sampler	Batch Size	Rank-1
Random Sampler	128	2.81
Triplet Sampler	(32,16)	49.68
Triplet Sampler	**(256,2)**	**51.24**

(2) Data Augmentations. For pose-based gait recognition, appropriate data augmentation holds significant importance. We have extensively experimented with various data augmentations and explored different combinations on both indoor and outdoor benchmarks. The details of these experiments can be seen in Tables 8 to 13. Furthermore, due to the observed tendency of methods to overfit on CASIA-B, we have applied the same augmentations used on Gait3D to GREW and OUMVLP-Pose datasets. The maximum improvement achieved through data augmentations is 15.1%.

(3) Network Depth. In the case of GaitGraph and GaitGraph2, we adhere to the original network depth settings. Since GaitGraph2, as an enhanced version of GaitGraph, adjusts the model layers to accommodate different dataset scales. To determine the optimal number of layers of GaitTR, we examine the network depth specifically on the GREW and OUMVLP-Pose. As shown in Figure 4, GaitTR achieves the highest performance of 16 layers on GREW and 14 layers on OUMVLP-Pose with the improvement of 2.8% and 4.7%, respectively.

Compared with Vanilla Version, the overall performance of the Improved Version witnesses significant enhancements shown in Tables 3 to 6, *i.e.*, +1.9%, +48.4%, +25.9% and +6.0% for GaitGraph, +2.6%, +2.4%, +9.6% and +1.3% for GaitGraph2, +0.5%, +3.3%, +6.8% and +2.5% for GaitTR on CASIA-B, OUMVLP-Pose, GREW and Gait3D, respectively. The Improved Version offers comparable experimental results for pose-based methods across various datasets, showcasing both unleashed potentials of the models and effectiveness of pose-based approaches.

5. Conclusion

FastPoseGait is an open-source toolbox designed for pose-based gait recognition. Unlike other projects that focus on a single algorithm, FastPoseGait integrates three representative algorithms into a unified framework with a highly modular structure, which allows for easy comparison of effectiveness and efficiency. The toolbox encompasses diverse benchmarks, pre-trained models, and comprehensive results, providing valuable resources for further advancements in pose-based gait recognition research.

Table 8: The data augmentation study of GaitGraph on CASIA-B.

Spatial					Sequential	CASIA-B			
InversePosesPre	MirrorPoses	PointNoise	JointNoise	RandomMove	FlipSequence	NM	BG	CL	Mean
—	—	—	—	—	—	73.75	63.19	51.62	62.85
					✓	**74.32**	**63.95**	**52.05**	**63.44**
✓					✓	73.16	59.96	43.53	58.88
	✓				✓	71.63	58.87	42.37	57.62
		✓			✓	**85.74**	**71.54**	**61.78**	**73.02**
			✓		✓	**75.77**	**64.88**	**49.66**	**63.44**
				✓	✓	68.56	54.96	39.04	54.19
✓		✓			✓	84.85	73.53	62.28	73.55
✓			✓		✓	77.09	67.79	53.08	65.99
		✓	✓		✓	85.98	74.22	64.85	75.02
✓		✓	✓		✓	**88.47**	**77.52**	**67.95**	**77.98**

Table 9: The data augmentation study of GaitGraph2 on CASIA-B.

Spatial					Sequential	CASIA-B			
InversePosesPre	MirrorPoses	PointNoise	JointNoise	RandomMove	FlipSequence	NM	BG	CL	Mean
—	—	—	—	—	—	78.76	67.89	60.18	68.94
✓						**81.85**	**69.53**	**62.06**	**71.15**
	✓					**82.33**	**71.21**	**63.06**	**72.20**
		✓				**83.60**	**72.80**	**67.01**	**74.47**
			✓			**78.39**	**68.69**	**63.22**	**70.10**
				✓		**81.08**	**68.61**	**61.14**	**70.28**
					✓	**82.21**	**70.57**	**60.19**	**70.99**
	✓	✓				81.81	68.82	63.64	71.42
		✓		✓		**84.31**	**71.96**	**65.78**	**74.02**
		✓			✓	81.04	69.33	60.16	70.18
✓		✓				81.40	69.70	62.55	71.22
		✓	✓			81.40	69.48	64.12	71.67
	✓	✓			✓	84.31	72.27	63.20	73.26
		✓	✓		✓	82.62	70.87	62.88	72.12
	✓	✓	✓	✓	✓	84.48	72.75	64.54	73.92
		✓	✓	✓	✓	83.16	71.49	62.26	72.31

Table 10: The data augmentation study of GaitTR on CASIA-B.

Spatial					Sequential	CASIA-B			
InversePosesPre	MirrorPoses	PointNoise	JointNoise	RandomMove	FlipSequence	NM	BG	CL	Mean
—	—	—	—	—	—	92.75	88.78	88.41	89.98
✓						**94.31**	**89.68**	**88.97**	**90.99**
	✓					**94.03**	**90.04**	**88.34**	**90.80**
		✓				93.49	88.48	87.23	89.73
			✓			**93.95**	**88.88**	**89.72**	**90.85**
				✓		92.84	87.78	86.16	88.93
					✓	93.21	88.87	86.80	89.63
✓	✓		✓			**95.02**	**90.70**	**89.67**	**91.80**

Table 11: The data augmentation study of GaitGraph on Gait3D.

Spatial					Sequential	Gait3D		
InversePosesPre	MirrorPoses	PointNoise	JointNoise	RandomMove	FlipSequence	Rank-1	Rank-5	Rank-10
—	—	—	—	—	—	11.4	24.2	32.8
✓						10.3	23.6	31.3
	✓					**11.1**	**25.2**	**33.0**
		✓				**12.1**	**28.1**	**36.4**
			✓			**10.7**	**25.9**	**33.0**
				✓		10.3	24.8	32.2
					✓	**11.0**	**26.0**	**33.7**
	✓	✓				13.2	29.0	38.0
	✓	✓	✓			**14.6**	**31.3**	**38.8**
	✓	✓	✓		✓	13.2	28.2	35.5

Table 12: The data augmentation study of GaitGraph2 on Gait3D.

Spatial					Sequential	Gait3D		
InversePosesPre	MirrorPoses	PointNoise	JointNoise	RandomMove	FlipSequence	Rank-1	Rank-5	Rank-10
—	—	—	—	—	—	11.3	24.1	31.1
✓						**11.4**	**24.4**	**33.9**
	✓					10.5	24.4	32.6
		✓				**11.9**	**26.1**	**32.8**
			✓			10.3	23.1	31.8
				✓		10.8	24.5	32.0
					✓	**11.7**	**25.9**	**33.0**
✓		✓			✓	12.6	23.4	29.9
		✓			✓	**12.5**	**24.7**	**30.6**

Table 13: The data augmentation study of GaitTR on Gait3D.

Spatial					Sequential	Gait3D		
InversePosesPre	MirrorPoses	PointNoise	JointNoise	RandomMove	FlipSequence	Rank-1	Rank-5	Rank-10
—	—	—	—	—	—	4.0	12.9	18.0
✓						**7.8**	**15.4**	**21.2**
	✓					4.3	12.6	18.2
		✓				4.7	12.5	16.8
			✓			4.4	13.2	17.6
				✓		3.7	11.9	17.2
					✓	**9.7**	**21.8**	**28.4**
✓					✓	**8.6**	**19.2**	**26.0**

References

[1] Weizhi An, Shiqi Yu, Yasushi Makihara, Xinhui Wu, Chi Xu, Yang Yu, Rijun Liao, and Yasushi Yagi. Performance evaluation of model-based gait on multi-view very large population database with pose sequences. *IEEE transactions on biometrics, behavior, and identity science*, 2(4):421–430, 2020.

[2] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[3] Hanqing Chao, Yiwei He, Junping Zhang, and Jianfeng Feng. Gaitset: Regarding gait as a set for cross-view gait recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 8126–8133, 2019.

[4] Chao Fan, Saihui Hou, Yongzhen Huang, and Shiqi Yu. Exploring deep models for practical gait recognition. *arXiv preprint arXiv:2303.03301*, 2023.

[5] Chao Fan, Yunjie Peng, Chunshui Cao, Xu Liu, Saihui Hou, Jiannan Chi, Yongzhen Huang, Qing Li, and Zhiqiang He. Gaitpart: Temporal part-based model for gait recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14225–14233, 2020.

[6] Hao-Shu Fang, Jiefeng Li, Hongyang Tang, Chao Xu, Haoyi Zhu, Yuliang Xiu, Yong-Lu Li, and Cewu Lu. Alphapose: Whole-body regional multi-person pose estimation and tracking in real-time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[9] Saihui Hou, Chunshui Cao, Xu Liu, and Yongzhen Huang. Gait lateral network: Learning discriminative and compact representations for gait recognition. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX*, pages 382–398. Springer, 2020.

[10] Saihui Hou, Xu Liu, Chunshui Cao, and Yongzhen Huang. Gait quality aware network: toward the interpretability of silhouette-based gait recognition. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[11] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673, 2020.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[13] Yanjie Li, Sen Yang, Peidong Liu, Shoukui Zhang, Yunxiao Wang, Zhicheng Wang, Wankou Yang, and Shu-Tao Xia. Simcc: A simple coordinate classification perspective for human pose estimation. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VI*, pages 89–106. Springer, 2022.

[14] Junhao Liang, Chao Fan, Saihui Hou, Chuanfu Shen, Yongzhen Huang, and Shiqi Yu. Gaitedge: Beyond plain end-to-end gait recognition for better practicality. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part V*, pages 375–390. Springer, 2022.

[15] Rijun Liao, Chunshui Cao, Edel B Garcia, Shiqi Yu, and Yongzhen Huang. Pose-based temporal-spatial network (ptsn) for gait recognition with carrying and clothing variations. In *Biometric Recognition: 12th Chinese Conference, CCBR 2017, Shenzhen, China, October 28-29, 2017, Proceedings 12*, pages 474–483. Springer, 2017.

[16] Rijun Liao, Shiqi Yu, Weizhi An, and Yongzhen Huang. A model-based gait recognition method with body pose and human prior knowledge. *Pattern Recognition*, 98:107069, 2020.

[17] Beibei Lin, Shunli Zhang, and Xin Yu. Gait recognition via effective global-local feature representation and local temporal aggregation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14648–14656, 2021.

[18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[19] Chiara Plizzari, Marco Cannici, and Matteo Matteucci. Spatial temporal transformer network for skeleton-based action recognition. In *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part III*, pages 694–701. Springer, 2021.

[20] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pages 369–386. SPIE, 2019.

[21] Yi-Fan Song, Zhang Zhang, Caifeng Shan, and Liang Wang. Stronger, faster and more explainable: A graph convolutional baseline for skeleton-based action recognition. In *proceedings of the 28th ACM international conference on multimedia*, pages 1625–1633, 2020.

[22] Yi-Fan Song, Zhang Zhang, Caifeng Shan, and Liang Wang. Constructing stronger and faster baselines for skeleton-based action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 45(2):1474–1488, 2022.

[23] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5686–5696, 2019.

[24] Noriko Takemura, Yasushi Makihara, Daigo Muramatsu, Tomio Echigo, and Yasushi Yagi. Multi-view large population gait dataset and its performance evaluation for cross-view gait recognition. *IPSJ transactions on Computer Vision and Applications*, 10:1–14, 2018.

[25] Torben Teepe, Johannes Gilg, Fabian Herzog, Stefan Hörmann, and Gerhard Rigoll. Towards a deeper understanding of skeleton-based gait recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1569–1577, 2022.

[26] Torben Teepe, Ali Khan, Johannes Gilg, Fabian Herzog, Stefan Hörmann, and Gerhard Rigoll. Gaitgraph: Graph convolutional network for skeleton-based gait recognition. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2314–2318. IEEE, 2021.

[27] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[28] Shiqi Yu, Daoliang Tan, and Tieniu Tan. A framework for evaluating the effect of view angle, clothing and carrying condition on gait recognition. In *18th international conference on pattern recognition (ICPR'06)*, volume 4, pages 441–444. IEEE, 2006.

[29] Cun Zhang, Xing-Peng Chen, Guo-Qiang Han, and Xiang-Jie Liu. Spatial transformer network on skeleton-based gait recognition. *arXiv preprint arXiv:2204.03873*, 2022.

[30] Jinkai Zheng, Xinchen Liu, Wu Liu, Lingxiao He, Chenggang Yan, and Tao Mei. Gait recognition in the wild with dense 3d representations and a benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20228–20237, 2022.

[31] Zheng Zhu, Xianda Guo, Tian Yang, Junjie Huang, Jiankang Deng, Guan Huang, Dalong Du, Jiwen Lu, and Jie Zhou. Gait recognition in the wild: A benchmark. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14789–14799, 2021.