*Applied Computer Science, vol. 19, no. 1, pp. 1–14*
*doi: 10.35784/acs-2023-01*

*Submitted: 2023-02-13 | Revised: 2023-03-01 | Accepted: 2023-03-09*

*Keywords: human pose estimation, lightweight model, Edge AI, deep learning, computer vision*

*Lei LIU* [0000-0002-1807-6906][*,**], *Eric B. BLANCAFLOR* [0000-0002-7189-3040][***], *Mideth ABISADO* [0000-0003-4215-7260][****]

# A LIGHTWEIGHT MULTI-PERSON POSE ESTIMATION SCHEME BASED ON JETSON NANO

**Abstract**

*As the basic technology of human action recognition, pose estimation is attracting more and more researchers' attention, while edge application scenarios pose a higher challenge. The authors propose a lightweight multi-person pose estimation scheme to meet the needs of real-time human action recognition on the edge end. This scheme uses AlphaPose to extract human skeleton nodes and adds ResNet and Dense Upsampling Revolution to improve its accuracy. Meanwhile, YOLO is used to enhance AlphaPose's support for multi-person pose estimation and to optimize the proposed model with TensorRT. In addition, the authors set Jetson Nano as the Edge AI deployment device of the proposed model and successfully realize the model migration to the edge end. The experimental results show that the speed of the optimized object detection model can reach 20 FPS, and the optimized multi-person pose estimation model can reach 10 FPS. With the image resolution of 320×240, the model's accuracy is 73.2%, which can meet the real-time requirements. In short, our scheme can provide a basis for a lightweight multi-person action recognition scheme on the edge end.*

## 1. INTRODUCTION

In recent years, human action recognition technology based on computer vision has been widely used in the health, game, and medical areas (Kong & Fu, 2022). According to relevant research and market reports, it was estimated that by 2023, the share of the digital fitness market in the entire sports fitness field will reach 27.4 billion dollars (Jegham et al., 2020). Figure 1 presents a digital fitness sample, which is a mobile-oriented intelligent fitness software. The AI fitness trainer in the software can automatically identify the user's actions and give suggestions. This new fitness method is popular with young people. Meanwhile, the development of these mobile-oriented human action recognition applications needs the support of human action recognition technology and Edge AI technology. Some problems

[*] National University, College of Computing and Information Technologies, Philippines, liulei@hnnu.edu.com
[**] Huainan Normal University, School of Computer Science, China, liulei@hnnu.edu.com
[***] Mapua University, School of Information Technology, Philippines, ebblancaflor@national-u.edu.ph
[****] National University, College of Computing and Information Technologies, Philippines, mbabisado@national-u.edu.ph

in these two aspects restrict the development of this trend. Among them, in human motion recognition, the general model is mainly slow in human multi-object detection (Zhang et al., 2019). The model size makes the model migration to the mobile and the edge end more challenging. Regarding Edge AI, the main problems are limited hardware resources and poor software tool-chain compatibility (Cao et al., 2016).

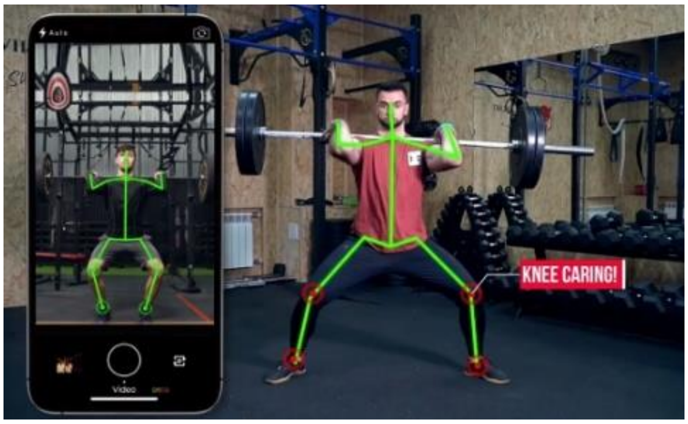

**Fig. 1. Illustration of intelligent fitness software**

In this regard, the authors first propose a hybrid human pose estimation model based on YOLO and AlphaPose to improve support for multiple human objects. Secondly, they enhance the structure of the model to obtain the trade-off between the accuracy and speed of the model. Finally, the authors compare and select a suitable Edge AI device, and migrate the lightweight model onto it to meet the need for the model to run in real-time on the edge end. Specifically, in optimizing the object detection model and multi-person pose estimation model, the authors mainly adjust the network structure of the model to improve the model speed. Meanwhile, the lightweight model with smaller model parameters is preferred in the consideration. The hardware, software, and performance are mainly compared in terms of Edge AI device performance analysis. All these specific evaluation criteria are limited to common indicators. For example, hardware includes CPU, GPU, memory, power consumption, etc.

The software consists of an operating system, SDK, and AI framework. Performance includes the frame rate. The rest of the paper is organized as follows: Section 2 reviews the research related to human action recognition and Edge AI. Section 3 presents the proposed scheme's overall design framework and each component's functions. Section 4 compares and analyzes the results of the proposed method in object detection and multi-person pose estimation. Finally, the conclusions are discussed.

## 2. REVIEW OF RELATED LITERATURE AND STUDIES

In this section, the authors first review related studies on human pose estimation, which provides essential support for the scheme. In multi-person pose estimation, the top-down method represented by AlphaPose will be highlighted. In Sec. 2.2 the authors mainly introduce the development status of Edge AI and the related work of the current mainstream lightweight Edge AI devices.

## 2.1. Human pose estimation

Human pose estimation is the primary technology in human action recognition, and its accuracy directly affects action recognition performance. According to the different data modes, human pose estimation methods can be divided into four categories: appearance-based, depth-based, optical flow-based, and skeleton-based (Chung et al., 2022). Skeleton-based schemes have recently attracted more researchers' attention due to their high recognition rate, fineness, and robustness (Sipola et al., 2022). Specifically, according to the processing sequence of human body parts, represent methods include two research branches: bottom-up and top-down (Gamra & Akhloufi, 2021).

Bottom-up methods are also known as part-based methods, represented by OpenPose (Chen et al., 2020). These methods first detect all human skeleton nodes in the image and then reassemble them into an individual human skeleton through the association between skeleton nodes. Pishchulin et al. (2016) proposes that the body parts of all people can be first detected by DeepCut and then tagged, filtered, and assembled by integral linear programming of these parts. A more substantial part detector based on ResNet (Tran et al., 2022) and a better incremental optimization strategy is proposed by (Gautam et al., 2020). Openpose introduces Part Affinity Fields (PAFs) to encode association scores between body parts with individuals and solves the matching problem by decomposing it into a set of bipartite matching subproblems. Kreiss et al. (2021) propose a method to enhance the correlation of human skeleton nodes by using a Part Intensity Field and a Part Association Field to locate and correlate body parts, respectively. Although the bottom-up method has proved its good performance because it mainly considers the local area of the image, their body part detectors may not be able to meet this challenge when the number of human objects in the image is small. The top-down methods have two stages, represented by AlphaPose (Fang et al., 2022). They first detect the human object in the image and mark the rectangular bounding box of each human area to eliminate the interference of non-human entities; Then, the skeleton points of each human body region are detected. Fang et al. (2017) propose a symmetric spatial transformer network to solve the problem of imperfect bounding boxes with huge noise given by human body detectors. Mask R-CNN (Nguyen et al., 2022) extends Faster R-CNN (Akshatha et al., 2022) by adding a pose estimation branch in parallel with the existing bounding box recognition branch after ROIAlign, enabling end-to-end training. Chen et al. (2018) use a feature pyramid network to localize simple joints and a refining network that integrates features of all levels from the previous network to handle complex joints. A simple-structured network (Xiao et al., 2018) with ResNet (Alnuaim et al., 2022) as the backbone and a few deconvolutional layers as upsampling head show effective and competitive results. Sun et al. (2019) present a robust high-resolution network, where a high-resolution subnetwork is established in the first stage, and high-to-low-resolution subnetworks are added one by one in parallel in subsequent steps, conducting repeated multi-scale feature fusions. Bertasius et al. (2019) extend from images to videos and propose a method for learning pose warping on sparsely labeled videos. Khirodkar et al. (2021) offer a Multi-Instance Pose Network (MIPNet) that predicts multiple 2D pose instances within a bounding box. It can overcome the limitations of crowded scenes with occlusions. The top-down methods can achieve remarkable precision on popular large-scale benchmarks. Moreover, it can further improve the rate and accuracy of pose estimation by optimizing the detector at the human object detection stage.

## 2.2. Edge AI

With the development of 5G, big data, and the industrial internet, the sinking of computing resources represented by edge computing has become a new development trend, which makes it possible for more artificial intelligence applications to appear at the edge (Shiraishi, 2020). With the support of Edge AI software technology and Edge AI hardware devices, it is possible to build a lightweight edge vision scheme that can meet the requirements of real-time data processing and response with low delay, and it will also provide critical support for the realization of edge oriented human pose estimation technology. In response, major hardware manufacturers have launched their own Edge AI hardware platform solutions. Figure 2 shows three common Edge AI devices. The GPU-based Jetson Nano launched by Nvidia has a sound software ecology (Süzen et al., 2020). The TPU-based Coral series development boards established by Google specifically for deep learning have high performance (Park et al., 2020). The latest version of Raspberry Pi (Dewangan & Sahu, 2021), which is widely used in the Internet of Things, has dramatically improved its hardware performance and can also provide good hardware support for Edge AI applications. Many computer vision schemes based on Edge AI have been successfully applied to various image and natural language processing scenes. However, Edge AI is still in its infancy. The software and hardware ecology are messy, the equipment performance is different, and the toolchain from data collection to model deployment and reasoning of each scheme still needs to be clarified.

**Fig. 2. Illustration of Edge AI devices**

To quickly and accurately deploy the human pose estimation model on the low-power and low-cost hardware platform, we focus on the optimization and migration of the human pose estimation model. It proposes to build a lightweight multi-person pose estimation scheme based on human skeleton nodes. Optimizing the object detection and pose estimation models allows the project to quickly obtain the image coordinates of human pose skeleton nodes, laying the foundation for human action recognition. Meanwhile, the system will use the model optimization framework to migrate the model to the Edge AI device.

## 3. METHODOLOGY

In this section, the authors first introduce the scheme's primary framework, which consists of YOLO and AlphaPose. Sections 3.2 and 3.3 present the optimization methods for object detection and multi-person pose estimation models, respectively. These optimization methods are processed based on the optimization framework-TensorRT.

## 3.1. Scheme design

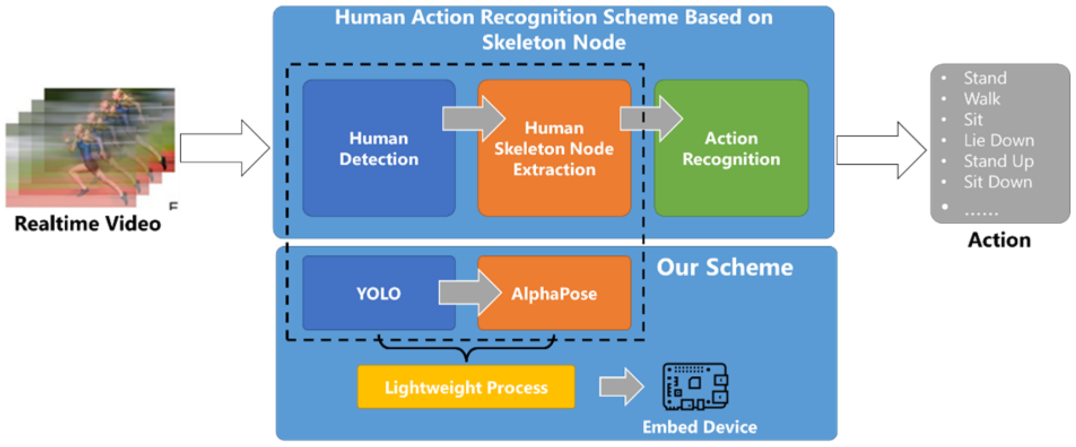

**Fig. 3. Illustration of human action recognition scheme & our scheme**

In Figure 3, the action recognition scheme based on human skeleton nodes includes three stages: the human object detection stage, the human skeleton node extraction stage, and the human action recognition stage. In the human object detection stage, YOLO series models are lightweight object detection models widely used on the mobile and edge end. They have achieved good performance in terms of mean AP, frame rate, and other indicators. Therefore, we used YOLO to process human object detection tasks. In addition, we used off-the-shelf YOLO, which was trained on the COCO dataset, and it already worked well in our scheme. In the multi-person skeleton node extraction stage, AlphaPose is superior to the comparison model in the AP@ indicator, as shown in Table 1. The AP@ represents the detection accuracy when Intersection Over Union (IOU) is in a specific value. Therefore, the authors use AlphaPose as the base model of the multi-person pose estimation scheme.

**Tab. 1. Comparison of multi-person pose estimation models**

| Model | AP@0.5:0.95 | AP@0.5 | AP@0.75 |
|---|---|---|---|
| OpenPose | 62.3 | 84.4 | 66.7 |
| Detection | 67.2 | 88.0 | 73.1 |
| AlphaPose | 73.3 | 89.2 | 79.1 |

To improve the performance of the scheme, the authors reconstruct the network of AlphaPose. The original AlphaPose uses Fast_Reset50 as the backbone to extract skeletal nodes from the image, and we replace it with ResNet. In addition, we adopt the Dense Upsampling Convolution (DUC) (Sediqi & Lee, 2021) to upsample the extracted features, as shown in Figure 4. The DUC introduces a deformable convolution module and deformable RoI pooling module to improve the model transformation capabilities of CNN. Both are based on adding additional offsets to the module at spatially sampled locations and learning the offsets from the target task without additional supervision. The new modules can easily replace the normal modules in existing CNN and can be trained end-to-end by standard back-propagation to produce deformable convolutional networks. The incorporation of DUC makes the improved AlphaPose more flexible in extracting skeleton features and can significantly reduce the scale of the model. We add three DUC modules to ResNet and used a 2D convolutional layer in 1×1 size to generate the heatmap. In addition, we first perform 2D convolution on a feature map of size c×h×w in the DUC module and then reshape it to c'×2h×2w via PixelShuffle (Liu et al., 2023).

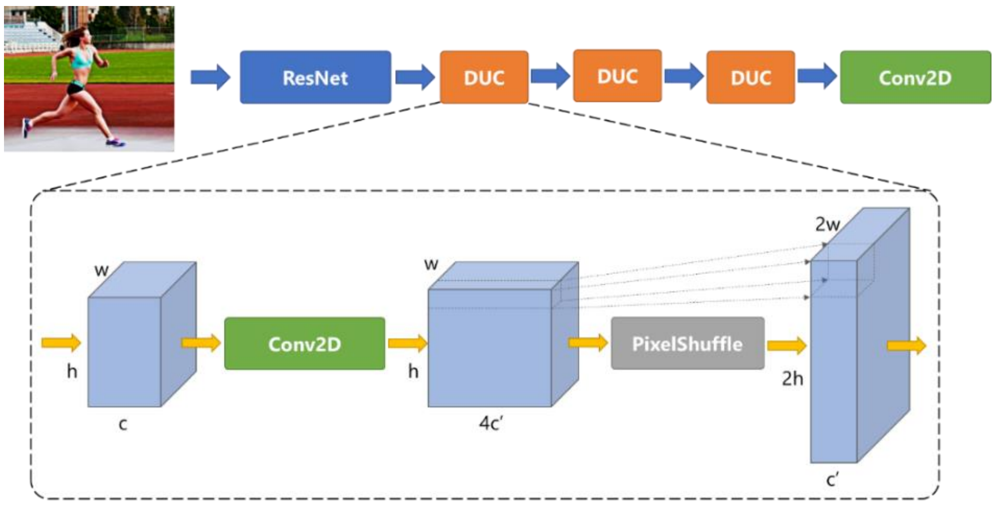

**Fig. 4. The network architecture of our pose estimation model**

## 3.2. Model optimization

TensorRT was mainly used to optimize YOLO and AlphaPose, optimization platforms and AI implementation, making NVIDIA for GPUs (Jeong et al., 2022). In Figure 5, it is both an inference optimization engine and a runtime execution engine. It provides optimal support for the inference phase of the model at the graphics optimization, operator optimization, memory optimization, and Int8 calibration levels. Specifically, it benefits from the fact that after training the neural network, TensorRT can compress, optimize and deploy the network at runtime without the overhead of a framework. TensorRT can also improve the latency, throughput, and throughput of the network through combining layers, kernel optimization selection, as well as performing optimization and conversion to optimal matrix math methods based on a specified precision. We first fine-tuned the network structure of the model by pruning and quantization and then further optimized the model with TensorRT.

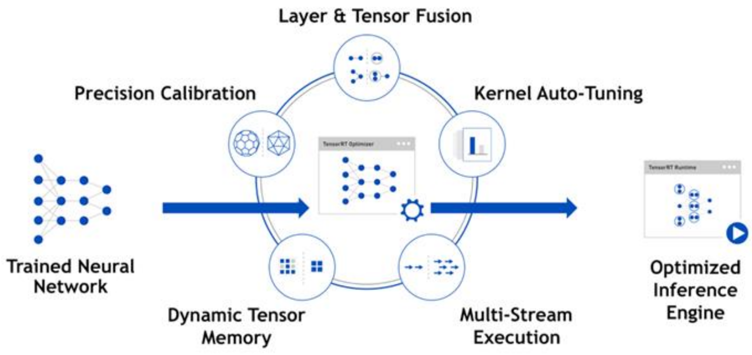

**Fig. 5. Model optimization processing flow of TensorRT**

### 3.2.1. Optimization of object detection model

In the YOLO series models (Ge et al., 2021), YOLOv4 outperforms YOLOv3 in terms of detection speed and accuracy and is comparable to YOLOv-5 in terms of real-time performance. This article uses YOLOv4-tiny-416, a lightweight version of YOLOv4, as the optimization target. Its parameter size is 24.3MB. YOLOv4-tiny-416 can achieve a good balance between the speed and accuracy of detection. The specific optimization process is shown in Figure 6. In the Tensor dimension initialization phase, set the layer of network '030_ Revolutionary = [*h//32, w//32, c*]', and ' 037_ Evolutionary = [*h//16, w//16, c*]'. The *h* and *w* represent the height and width of the input image, respectively. The *c* represents the channels of the input image, and the "//" represents an integer division operation. After conversion, we propose the optimized model YOLOv4-opt, and its parameter is 27.7MB.

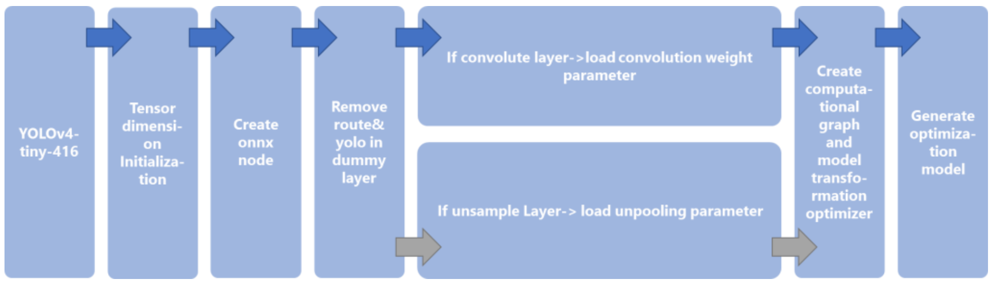
YOLOv4-tiny-416
Tensor dimension Initialization
Create onnx node
Remove route& yolo in dummy layer
If convolute layer->load convolution weight parameter
If unsample Layer-> load unpooling parameter
Create computational graph and model transformation optimizer
Generate optimization model

**Fig. 6. Object detection model optimize processing**

After the human object detection results are obtained, they need to be serialized. First, the result is converted into a two-dimensional tensor list. $T_{\_list} = [[x_1, y_1, h_1, w_1, score_1], [x_2, y_2, h_2, w_2, sc_2], \ldots, [x_i, y_i, w_i, h_i, score_i]]$, where $[x_i, y_i, h_i, w_i, score_i]$ represents the structural data of the $i_{th}$ human object, *x*, and *y* represent the abscissa and ordinate of the image at the upper left corner of the human body prediction frame, respectively. The *h* and *w* represent the height and width of the prediction frame, respectively. The *score* represents the confidence level of the human body. In addition, we transform the original image $I_{m\_t}$ into floating point 32-bit tensor data $I_{m\_t'}$, which performs normalization operations, as in Formula (1), (2), and (3).

$$I_{m\_t}[0] += -0.406 \quad (1)$$

$$I_{m\_t}[1] += -0.457 \quad (2)$$

$$I_{m\_t}[2] += -0.480 \quad (3)$$

Among them, the $I_{m\_t}[0]$ is the R-Channel data of $I_m$, the $I_{m\_t}[1]$ is the G-Channel data of $I_m$, and the $I_{m\_t}[2]$ is the B-Channel data of $I_m$. According to $T_{\_list}$, the human body region image is cut from the original image and arranged from high to low according to the *score* to obtain the serialized image list, which realizes the serialization of human body images and improves the data interaction efficiency between the object detection model and the human pose skeleton node detection model. According to $T_{\_lis}$, we cut the human body region images from the original images and arranges them in descending order according to the *score* to obtain the serialized image list. This serialization method of human body images improves the efficiency of data interaction between the object detection model and the human pose estimation model.

### 3.2.2. Optimization of multi-person pose estimation model

The original model parameter size of AlphaPose is 238.9MB, and the optimization method is shown in Figure 7. In the human pose estimation model, we initialize the input dimension of the dummy network layer and set it to the tensor type $(1,3,H_{dummy},W_{dummy},)$. Among them, $1$ represents the batch size, $3$ represents the number of image channels, and $H_{dummy}$ and $W_{dummy}$ represent the normalization scale of the input image. We set $H_{dummy}$ =224 and $W_{dummy}$ =160. An onnx network node is created for the dimension-initialized human pose estimation model, and the input-output network layer of it is customized, with the input layer set to "input1" and the output layer formed to "output1". We create the model calculation graph and set its input dimension to the tensor type $(1,3,H_d,W_d)$. The $1$ represents the batch size, and $3$ represents the number of image channels, $W_d$ and $H_d$ represent the normalized scale of the input image of the network layer. In our paper, we set $W_d$ = 160, $H_d$ = 224, and generate the optimization model AlphaPose-opt, with 184.5MB of model parameter size.

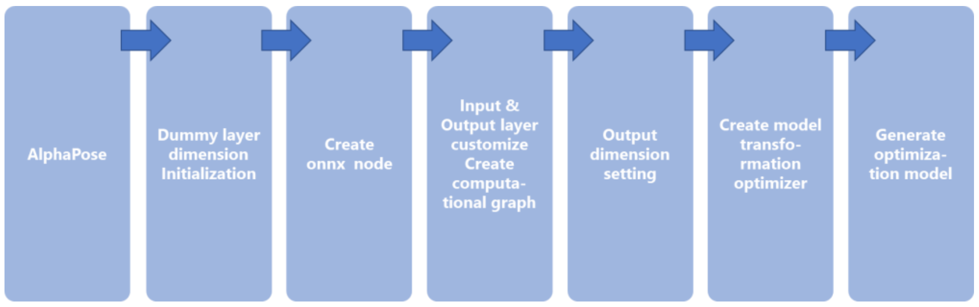
AlphaPose
Dummy layer dimension Initialization
Create onnx node
Input & Output layer customize Create computa-tional graph
Output dimension setting
Create model transfo-rmation optimizer
Generate optimiza-tion model

**Fig. 7. Pose skeleton node detection model optimizes processing**

## 3.3. Model migration

There are significant differences in hardware, software, and performance between the various types of Edge AI devices; it needs the proper edge devices to make our model work well on the edge end. In Tables 2 and 3, the Jetson series device performs well in terms of performance and cost of trade-off. Therefore, we choose Jetson Nano as the hardware platform of the proposed model.

**Tab. 2. Comparison of Edge AI devices in hardware**

| Edge AI Device | General Processor | AI Processor | Peak Performance | Memory Bandwidth | Power Consumption | Cost |
|---|---|---|---|---|---|---|
| Raspberry Pi(4B) | CPU:4*BCM2711 CPU:4*Cortex-A73 | Video Core VI | 735 GOP/s | 75GB/s | 7.5 Watts | $50 |
| Coral Dev Board | CPU:4*Cortex-A53 GPU: GC7000 Lite | TPU | 4 TOP/s | 34 GB/s | 2 TOPS per watt | $130 |
| Jetson Nano | CPU:4*Cortex-A57 | 128*CUDA GPU | 472 GOP/s | 25.6 GB/s | 5-10 Watts | $100 |

**Tab. 3. Comparison of Edge AI devices in software and performance**

| Edge AI Device | Operating System | SDK | AI Framework | Data type | Frequency |
|---|---|---|---|---|---|
| Raspberry Pi(4B) | Ubuntu CentOS Windows Raspbian | Raspberry Pi OS | TensorFlow, Caffe, PyTorch, Paddle Lite | FP16, Int8 | Low (<10fps) |
| Coral Dev Board | Mendel Linux OS | Edge TPU API | Tensor Flow Lite | FP32, FP16, Int8 | Moderate (>10fps&<20fps) |
| Jetson Nano | NVIDIA L4T | JetPack SDK | TensorFlow, Pytorch, MXNet, Caffe | FP32, FP16, Int8 | High (>20fps & <40fps) |

The operation of TensorRT for model migration consists of two main phases: build and deployment. The Build phase involves the conversion of the model from another model form to a TRT form. During the model conversion, the inter-layer fusion and accuracy calibration of the optimization mentioned above is completed. The output of this step is an optimized TRT model for the specific GPU platform and network model, serialized to disk or memory in the form of a plan file. The plan file from the previous step is first deserialized, a Runtime Engine is created, and the inference task can then be executed. In our paper, YOLO and STGCN are built in the PyTorch framework and converted into TRT models using ONNX intermediate conversions, as shown in Figure 8. After optimization, the model can reduce the model's parameters by 16%.

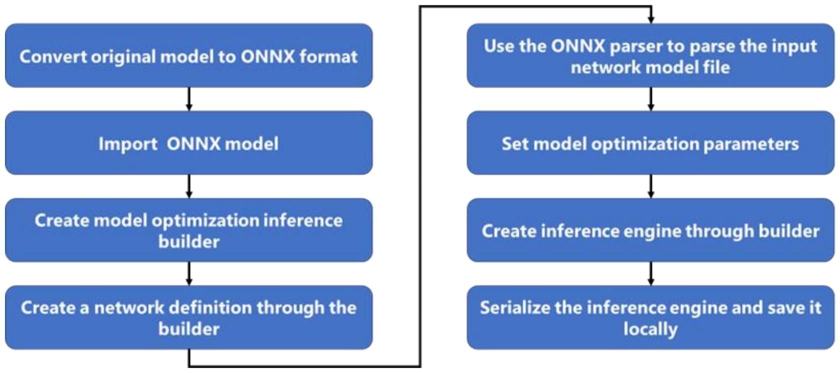
Convert original model to ONNX format
Import ONNX model
Create model optimization inference builder
Create a network definition through the builder
Use the ONNX parser to parse the input network model file
Set model optimization parameters
Create inference engine through builder
Serialize the inference engine and save it locally

**Fig. 8. TensorRT model migration processing**

## 4. EXPERIMENT AND ANALYSIS

The hardware environment used in the experiment is the NVIDIA Jetson Nano embedded development board, with 4-core 64bit ARM CPU, 128-core integrated NVIDIA GPU, 4GB DDR4 memory, and an apparent size of 80mm×100mm. The software environment is 64bit Ubuntu 18.04LTS, and the main tools relied on include OpenCV 4.5.1, Torch 1.7.0, and TensorRT 7.1.3.

The experiment is mainly divided into two parts. The first part compares the current widely used YOLO series model with the optimized YOLOv4-opt, and the second part compares the current typical lightweight attitude estimation model with the optimized AlphaPose-opt. Specifically YOLOv3-spp-416, YOLOv4-tiny-416, YOLOv5s, YOLOv6s, and YOLOv7-tiny. Meanwhile, Pose, OpenPose, trt_Pose, Tf Posert, HyperPose.

### 4.1. Dataset and Evaluation Metric

The experimental test data comes from the public dataset Microsoft COCO dataset and UR Fall Detection dataset [31]. The first one is for YOLOv4-tiny-416-opt, and the other is for Pose_opt. The UR Fall Detection dataset is a commonly used human fall behavior dataset that can be used for the experimental analysis of the object detection model and pose skeleton node detection model. In out paper, the original dataset is expanded by using three image conversion methods: symmetrical inversion, rotation, and brightness transformation, and finally, 100 human behavior videos are obtained, with the video resolution adjusted to 320×240, and the video frame rate is 30FPS.

We select two evaluation indexes that reflect the detecting object detection model and the attitude estimation model to evaluate the improved model. The first is the average precision mean (mAP), which is calculated from the average precision (AP), as shown in Formula (4) and (5). Where Nc represents the number of categories, AP represents the average precision of different types. After getting the AP values of each class, average them to get mAP. The second is the frame rate (FPS), which is usually used to balance the real-time performance of the model. It represents the number of pictures the neural network can process per second.

$$AP = \int_0^1 P(R)dR \tag{4}$$

$$mAP = \frac{\sum P_A}{N_c} \tag{5}$$

## 4.2. Object detection model performance analysis

In Table 4, the YOLOv4-opt has a detection frame rate of 20.3 FPS, 2.14 times that of the original model, and has little difference from other comparison methods in terms of mAP. The frame rate of YOLOv6s and YOLOv7-tiny can reach 10 frames/s, and YOLOv7-tiny is better than YOLOv4-tiny-416 in terms of the average frame rate. However, due to the complexity of deploying YOLOv7-tiny on JetsonNaon, it needs better support for edge devices. The mAP of YOLOv3-spp-416 is better than YOLOv4-opt, but its detection frame rate is only 3.8 frames/s, which cannot meet the real-time requirements. Therefore, considering the comprehensive performance of the model in terms of detection accuracy and frame rate, our model performs well in all comparison algorithms and applies to low-power embedded development boards.

**Tab. 4. Performance comparison of object detection models**

| Model | FPS | mAP | Image Resolution |
|---|---:|---:|:---:|
| YOLOv3-spp-416 | 3.8 | 85.5 | 320×240 |
| YOLOv4-tiny-416 | 9.5 | 83.3 | 320×240 |
| YOLOv5s | 6.7 | 82.4 | 320×240 |
| YOLOv6s | 13.0 | 83.0 | 320×240 |
| YOLOv7-tiny | 14.8 | 84.3 | 320×240 |
| YOLOv4-opt (ours) | 20.3 | 83.1 | 320×240 |

## 4.3. Multi-person pose estimation performance analysis

In Table 5, the detection frame rate of the AlphaPose-opt is 1.5 times that of the original model, and the frame rate reaches 9.8 FPS while maintaining the same mAP. Therefore, the AlphaPose-opt has a certain degree of real-time. Meanwhile, the AlphaPose-opt has a better detection frame rate and average accuracy than OpenPose. The mAP of OpenPose is higher than that of trt-Pose, and the detection frame rate is reduced to 1/2 of trt-Pose, but the image resolution is 224×224. The mAP of TF-Pose is better than our model, but its detection frame rate is low, so it does not have high practicability. The all-around performance of HyperPose is consistent with AlphaPose-opt, and both have high detection accuracy and practicability. Therefore, our model has a high application value on the edge end.

**Tab. 5. Performance comparison of pose skeleton node detection models**

| Model | FPS | mAP | Image Resolution |
|---|---|---|---|
| Pose | 6.5 | 74.3 | 320×240 |
| OpenPose | 3.2 | 66.7 | 320×240 |
| trt-Pose | 5.4 | 62.6 | 224×224 |
| TF-Pose | 4.6 | 72.6 | 320×240 |
| HyperPose | 11.3 | 69.5 | 320×240 |
| AlphaPose-opt(ours) | 9.8 | 73.2 | 320×240 |

## 5. CONCLUSIONS

The authors focused on the multi-person action recognition scenarios on the edge end. They propose a lightweight hybrid model based on human skeleton nodes to solve the difficulty of the AI model running in a limited resource environment. Their scheme adopts the combination of YOLO and AlphaPose to process the multi-object and human skeleton node detection task, respectively. Meanwhile, the authors realize the light weight of the model by improving the model structure and optimizing the framework, and successfully migrating the model to Jetson Nano. The experimental results show that the proposed method has high real-time accuracy and can provide a basis for human action recognition on the edge end.

However, the solution has some limitations, mainly: 1) the accuracy of the optimized human pose estimation model is not yet high enough and the frame rate is easily affected by the number of detected people, so the model is suitable for scenarios with a small number of people and a simple environment. 2) the resolution of the input video that the solution can handle is low, so it is more suitable for coarse-grained human pose estimation applications. Therefore, future research will continue on multi-target local occlusion, anti-light changeability, and complex environment adaptability to enhance the model's stability and robustness.

## Acknowledgments

*This study received support from the following sources: the University Natural Science Foundation of Anhui Province (Grant no. 2022AH010085), the University Natural Science Foundation of Anhui Province (Grant no. KJ2021A0970), the Huainan Normal University Scientific Research Project (Grant no. 2022XJZD030), and the Key Research and Development Plan Project Foundation of Huainan (Grant no. 2021A248).*

## Data Availability Statement

*The datasets used and analyzed during the current study are available from the corresponding author upon reasonable request.*

## Conflicts of Interest

*The authors declare that they have no conflicts of interest.*

## REFERENCES

Akshatha, K. R., Karunakar, A. K., Shenoy, S. B., Pai, A. K., Nagaraj, N. H., & Rohatgi, S. S. (2022). Human detection in aerial thermal images using faster R-CNN and SSD algorithms. *Electronics*, *11*(7), 1151. https://doi.org/10.3390/electronics11071151

Alnuaim, A. A., Zakariah, M., Hatamleh, W. A., Tarazi, H., Tripathi, V., & Amoatey, E. T. (2022). Human-computer interaction with hand gesture recognition using ResNet and MobileNet. *Computational Intelligence Neuroscience*, *2022*, 8777355. https://doi.org/10.1155/2022/8777355

Bertasius, G., Feichtenhofer, C., Tran, D., Shi, J., & Torresani, L. (2019). Learning temporal pose estimation from sparsely-labeled Videos. *ArXiv, abs/1906.04016*. https://doi.org/10.48550/arXiv.1906.04016

Cao, Z., Simon, T., Wei, S.-E., & Sheikh, Y. (2016). Realtime multi-person 2D pose estimation using part affinity fields. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* (pp. 1302–1310). IEEE. https://doi.org/10.1109/CVPR.2017.143.

Chen, W., Jiang, Z., Guo, H., & Ni, X. (2020). Fall Detection Based on Key Points of Human-Skeleton Using OpenPose. *Symmetry*, *12*(5), 744. https://doi.org/10.3390/sym12050744

Chen, Y., Wang, Z., Peng, Y., Zhang, Z., Yu, G., & Sun, J. (2018). Cascaded pyramid network for multi-person pose estimation. *Proceedings of the IEEE Computer Society Conference on Computer Vision Pattern Recognition* (pp. 7103–7112). IEEE. https://doi.org/10.1109/CVPR.2018.00742

Chung, J.-L., Ong, L.-Y., & Leow, M. C. (2022). Comparative analysis of skeleton-based human pose estimation. *Future Internet*, *14*(12), 380. https://doi.org/10.3390/fi14120380

Dewangan, D. K., & Sahu, S. P. (2021). Deep learning-based speed bump detection model for intelligent vehicle system using raspberry pi. *IEEE Sensors Journal*, *21*, 3570–3578. https://doi.org/10.1109/JSEN.2020.3027097

Fang, H., Li, J., Tang, H., Xu, C., Zhu, H., Xiu, Y., Li, Y.-L., & Lu, C. (2022). AlphaPose: Whole-Body Regional Multi-Person Pose Estimation and Tracking in Real-Time. *ArXiv, abs/2211.03375*. https://doi.org/10.48550/arXiv.2211.03375

Fang, H., Xie, S., Tai, Y.-W., & Lu, C. (2017). RMPE: Regional multi-person pose estimation. *IEEE International Conference on Computer Vision* (pp. 2353–2362). IEEE. https://doi.org/10.48550/arXiv.1612.00137

Gamra, M. B., & Akhloufi, M. A. (2021). A review of deep learning techniques for 2D and 3D human pose estimation. *Image Vis. Comput*, *114*, 104282. https://doi.org/10.1016/j.imavis.2021.104282

Gautam, B. P., Noda, Y., Gautam, R., Sharma, H. P., Sato, K., & Neupane, S. B. (2020). Body part localization and pose tracking by using deepercut algorithm for king cobra's BBL (Biting Behavior Learning). *International Conference on Networking Network Applications* (pp. 422–429). IEEE. https://doi.org/10.1109/NaNA51271.2020.00078

Ge, Z., Liu, S., Wang, F., Li, Z., & Sun, J. (2021). YOLOX: Exceeding YOLO series in 2021. *ArXiv, abs/2107.08430*. https://doi.org/10.48550/arXiv.2107.08430

Jegham, I., Khalifa, A. B., Alouani, I., & Mahjoub, M. A. (2020). Vision-based human action recognition: An overview and real world challenges. *Forensic Science International: Digital Investigation*, *32*, 200901. https://doi.org/10.1016/j.fsidi.2019.200901

Jeong, E., Kim, J., & Ha, S. (2022). TensorRT-Based framework and optimization methodology for deep learning inference on jetson boards. *ACM Transactions on Embedded Computing Systems*, *21*, 1–26. https://doi.org/10.1145/3508391

Khirodkar, R., Chari, V., Agrawal, A., & Tyagi, A. (2021). Multi-Instance pose networks: rethinking top-down pose estimation. *IEEE/CVF International Conference on Computer Vision* (pp. 3102-3111). IEEE. https://doi.org/10.48550/arXiv.2101.11223

Kong, Y., & Fu, Y. (2022). Human action recognition and prediction: A survey. *International Journal of Computer Vision*, *130*(5), 1366-1401. https://doi.org/10.48550/arXiv.1806.11230

Kreiss, S., Bertoni, L., & Alahi, A. (2021). OpenPifPaf: Composite fields for semantic keypoint detection and spatio-temporal association. *IEEE Transactions on Intelligent Transportation Systems*, *23*, 13498–13511. https://doi.org/10.48550/arXiv.2103.02440

Liu, M.-J., Wan, L., Wang, B., & Wang, T.-L. (2023). SE-YOLOv4: shuffle expansion YOLOv4 for pedestrian detection based on PixelShuffle. *Applied Intelligence*, *2023*. https://doi.org/10.1007/s10489-023-04456-0

Nguyen, S.-H., Le, T.-T.-H., Nguyen, H.-B., Phan, T.-T., Nguyen, C.-T., & Vu, H. (2022). Improving the Hand Pose Estimation from Egocentric Vision via HOPE-Net and Mask R-CNN. *International Conference on Multimedia Analysis Pattern Recognition* (pp. 1-6). IEEE. https://doi.org/10.1109/MAPR56351.2022.9924768

Park, K., Jang, W., Lee, W., Nam, K., Seong, K., Chai, K., & Li, W.-S. (2020). Real-time mask detection on google edge TPU. *ArXiv*, *abs/2010.04427*. https://doi.org/10.48550/arXiv.2010.04427

Pishchulin, L., Insafutdinov, E., Tang, S., Andres, B., Andriluka, M., Gehler, P., & Schiele, B. (2016). DeepCut: Joint subset partition and labeling for multi person pose estimation. *Conference on Computer Vision Pattern Recognition* (pp. 4929–4937). IEEE. https://doi.org/10.1109/CVPR.2016.533

Sediqi, K. M., & Lee, H. J. (2021). A novel upsampling and context convolution for image semantic segmentation. *Sensors, 21*(6), 2170. https://doi.org/10.3390/s21062170

Shiraishi, Y. (2020). Latest trend of edge aI devices. *Journal of The Japan Institute of Electronics Packaging, 23*(2), 145-149. https://doi.org/10.5104/jiep.23.145

Sipola, T., Alatalo, J., Kokkonen, T., & Rantonen, M. (2022). Artificial intelligence in the IoT Era: A Review of Edge AI Hardware and Software. *31st Conference of Open Innovations Association* (pp. 320-331). IEEE. https://doi.org/10.23919/FRUCT54823.2022.9770931

Sun, K., Xiao, B., Liu, D., & Wang, J. (2019). Deep high-resolution representation learning for human pose estimation. *IEEE/CVF Conference on Computer Vision Pattern Recognition* (pp. 5686–5696.) IEEE. https://doi.org/10.1109/CVPR.2019.00584.

Süzen, A. A., Duman, B., & Şen, B. (2020). Benchmark analysis of jetson TX2, jetson nano and raspberry PI using Deep-CNN. *International Congress on Human-Computer Interaction, Optimization Robotic Applications* (pp.1–5.) IEEE. https://doi.org/10.1109/HORA49412.2020.9152915

Tran, H. Y., Bui, T. M., Pham, T.-L., & Le, V.-H. (2022). An evaluation of 2D human pose estimation based on ResNet backbone. *Journal of Engineering Research and Sciences, 1*(2), 59–67. https://doi.org/10.55708/js0103007

Xiao, B., Wu, H., & Wei, Y. (2018). Simple baselines for human pose estimation and tracking. *European Conference on Computer Vision. Lecture Notes in Computer Science* (pp. 472–487). Springer. https://doi.org/10.1007/978-3-030-01231-1_29

Zhang, H.-B., Zhang, Y.-X., Zhong, B., Lei, Q., Yang, L., Du, J.-X., & Chen, D.-S. (2019). A comprehensive survey of vision-based human action recognition methods. *Sensors, 19*(5), 1005–1016. https://doi.org/10.3390/s19051005