Comparative Study of Pose Estimation Models for Action Sticker Generation: StackedHourglass Network vs SimpleBaseline

Author Name
Affiliation

June 4, 2025

Abstract

This study compares the performance of StackedHourglass Network and SimpleBaseline models in pose estimation for action sticker generation. Both models were directly designed and implemented with code, and trained on the same MPII dataset. The loss reduction during training was visualized through graphs to monitor learning progress, and qualitative evaluation was performed by comparing pose estimation results through images. The experimental results show that SimpleBaseline achieves similar performance to StackedHourglass with fewer parameters, particularly excelling in computational efficiency. This research provides practical guidelines for selecting appropriate pose estimation models for action sticker production.

Keywords: Pose Estimation, StackedHourglass Network, SimpleBaseline, Action Stickers, Deep Learning

1 Introduction

Action stickers are digital content that express user movements and emotions, widely used in social media and messaging applications. Accurate human pose estimation technology is essential for effective action sticker generation.

In this study, as part of a learning node project, we conducted a comparative analysis of pose estimation models for action sticker generation. Specifically, we directly implemented and compared the performance of StackedHourglass Network and SimpleBaseline models to evaluate their applicability in actual sticker production environments.

1.1 Research Objectives

- Performance comparison of two pose estimation models
- Presentation of model selection criteria suitable for action sticker generation
- Analysis of model-specific characteristics through learning process examination

2 Methods

2.1 Model Architecture

2.1.1 StackedHourglass Network

The StackedHourglass Network is an end-to-end trainable neural network designed for human pose estimation. The core of this model is the "Hourglass" structure, which repeatedly performs top-down and bottom-up processing to capture features at various scales. In this study, we directly implemented the following structure:

- **Preprocessing Layers**: 7×7 convolution, batch normalization, ReLU activation function
- **4 Stacked Hourglass Modules**: Each module independently generates heatmaps of size 64×64×16
- **Skip Connections**: Connection structure for information transfer between hourglass modules
- **Intermediate Supervision**: Gradient optimization through intermediate supervision in each stack
- **Bottleneck Residual Blocks**: Residual structure for efficient feature extraction

Each Hourglass module consists of an upper branch (skip pathway) and a lower branch. The lower branch reduces resolution through max pooling, extracts features through bottleneck blocks, then restores to original resolution through upsampling. Finally, high-resolution and low-resolution information are combined through skip connections.

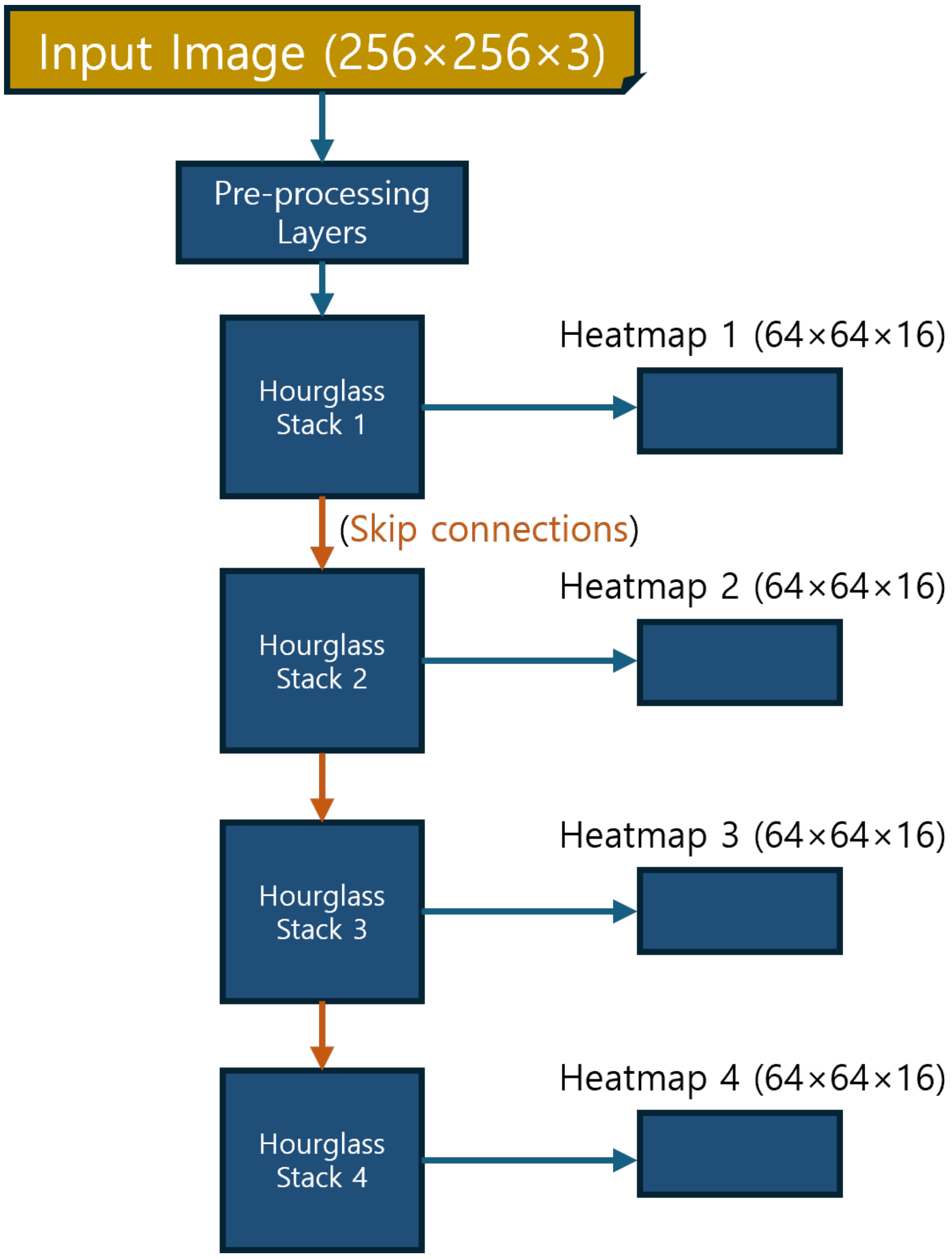
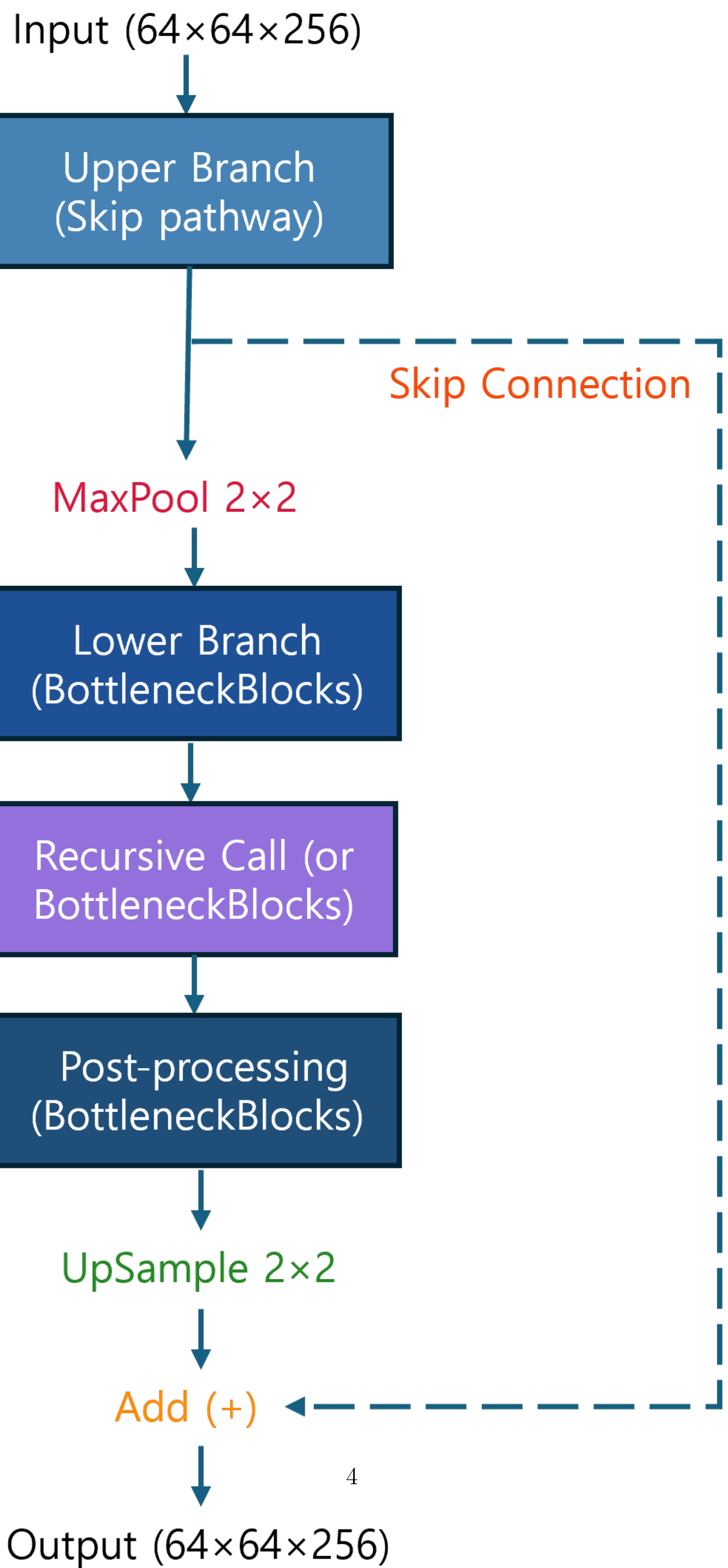


Figure 1: Complete StackedHourglass Network architecture. Four Hourglass stacks are sequentially connected, with each stack generating intermediate heatmaps of size 64×64×16.

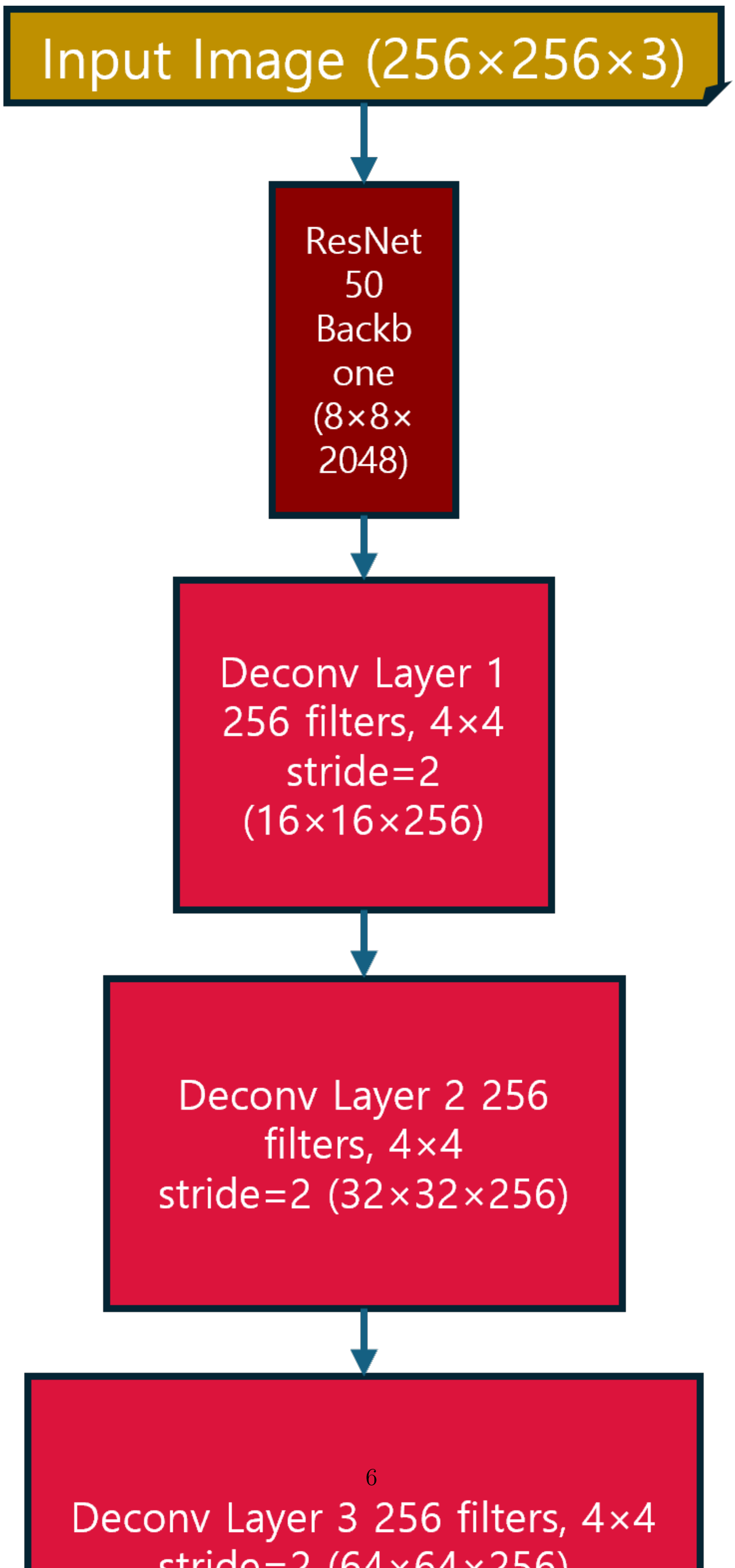

Input (64×64×256)
Upper Branch (Skip pathway)
Skip Connection
MaxPool 2×2
Lower Branch (BottleneckBlocks)
Recursive Call (or BottleneckBlocks)
Post-processing (BottleneckBlocks)
UpSample 2×2
Add (+)
Output (64×64×256)

2.1.2 SimpleBaseline

The SimpleBaseline model is a simple yet powerful model that demonstrates effective pose estimation without complex architectures. This model has a clear and intuitive structure, implemented as follows:

- **ResNet50 Backbone**: Pre-trained ResNet50 on ImageNet used as feature extractor ($8\times8\times2048$ output)
- **3 Deconvolutional Layers**: Each using 256 filters, 4×4 kernel, stride=2 to progressively increase resolution
  - Deconv Layer 1: $8\times8\times2048 \rightarrow 16\times16\times256$
  - Deconv Layer 2: $16\times16\times256 \rightarrow 32\times32\times256$
  - Deconv Layer 3: $32\times32\times256 \rightarrow 64\times64\times256$
- **Batch Normalization + ReLU**: Applied after each deconvolutional layer to improve training stability
- **Final Convolution Layer**: 1×1 convolution to generate final heatmap of size $64\times64\times16$

This structure performs effective pose estimation with single output without complex multi-stage supervision, offering advantages in computational efficiency and implementation ease.


Input Image (256×256×3)
ResNet 50 Backbone (8×8×2048)
Deconv Layer 1 256 filters, 4×4 stride=2 (16×16×256)
Deconv Layer 2 256 filters, 4×4 stride=2 (32×32×256)
Deconv Layer 3 256 filters, 4×4 stride=2 (64×64×256)

2.2 Experimental Setup

- **Dataset:** MPII Human Pose Dataset used. Consists of approximately 25,000 training images and 3,000 validation images, providing annotations for 16 joint points
- **Data Preprocessing**:
  - ROI Cropping: Extract joint-centered regions based on keypoints
  - Image Resizing: Normalize all input images to 256×256×3
  - Normalization: Scale pixel values to [-1, 1] range (value/127.5 - 1)
  - Heatmap Generation: Generate 64×64×16 Gaussian heatmaps for each keypoint
- **Training Environment:** TensorFlow 2.x, GPU-based distributed training environment
- **Hyperparameters:**
  - Learning Rate: StackedHourglass (0.0007), SimpleBaseline (0.0003)
  - Batch Size: 16 (StackedHourglass), 4 (SimpleBaseline)
  - Epochs: 5 epochs
  - Optimizer: Adam optimizer with adaptive learning rate decay
- **Evaluation Metrics:** PCK@0.5, PCK@0.2, mAP (mean Average Precision), inference time measurement

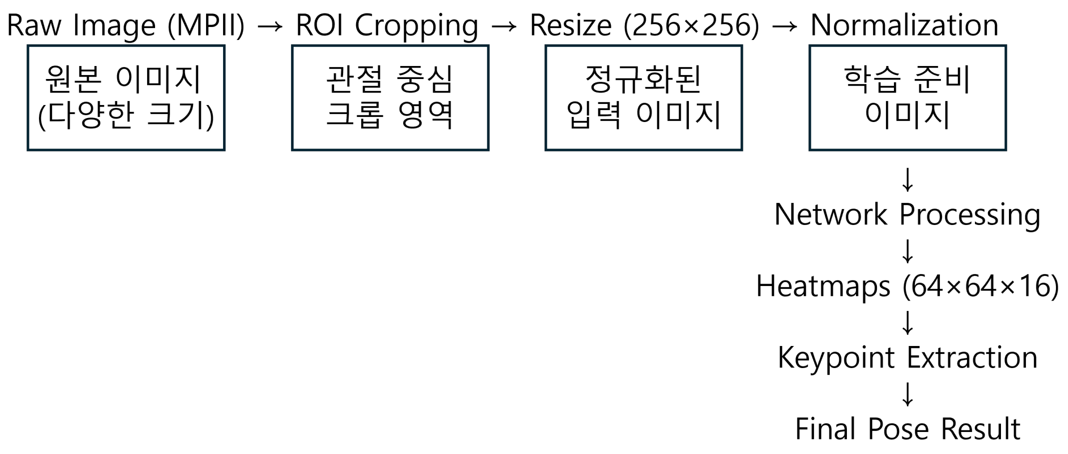


Figure 4: Data processing pipeline. Shows the complete process from original MPII images to final pose results. Images undergo ROI cropping, resizing, and normalization before network input, then keypoints are extracted through heatmaps.

2.3 Additional Experiments

For fair comparison between the two models, the following additional experiments were conducted:

- **Unified Loss Function**: Both models use the same weighted MSE loss for heatmap regression
- **Data Augmentation**: Random margin cropping (0.1-0.3) applied during training to improve robustness
- **Training Stability Analysis**: Overfitting prevention through learning rate decay and early stopping
- **Qualitative Evaluation**: Visual comparison of keypoint detection accuracy in complex poses like golf swings
- **Computational Efficiency Analysis**: Comparison of practical aspects including model size, inference speed, and memory usage

Particular emphasis was placed on evaluating performance in dynamic poses and complex joint configurations suitable for action sticker generation purposes.

3 Results

3.1 Training Progress

Figures 5 show the loss reduction during training for StackedHourglass Network and SimpleBaseline models respectively.

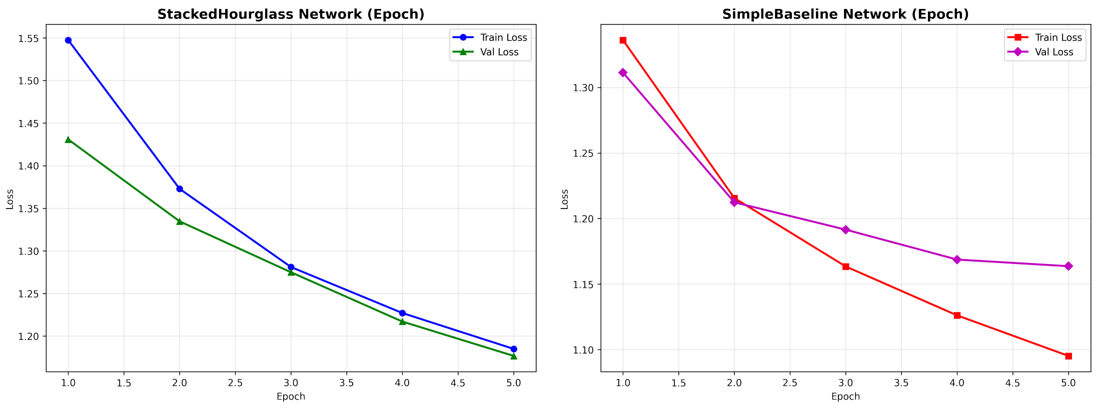


Figure 5: Training progress of each model (Left: StackedHourglass Network, Right: SimpleBaseline)

3.2 Qualitative Comparison Results

Figure 6 shows qualitative comparison images of pose estimation results from both models. It demonstrates how each model detects joint points in golf swing motions.

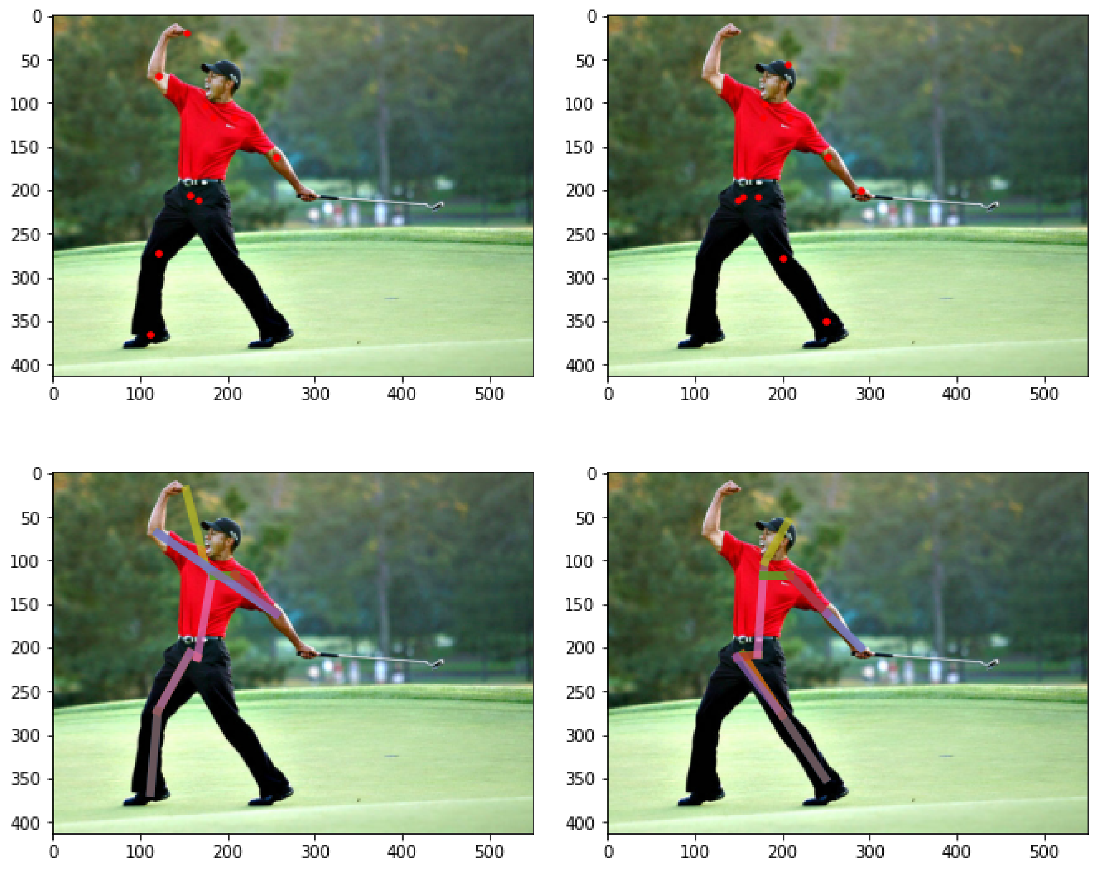


Figure 6: Pose estimation result comparison between two models - Golf swing motion (Left: StackedHourglass, Right: SimpleBaseline)

3.3 Quantitative Evaluation Results

Table 1 shows quantitative evaluation results for various metrics and benchmarks.

Table 1: Performance evaluation results by model

Metric	**StackedHourglass**	**SimpleBaseline**	**Improvement**
PCK@0.5	[Value]	[Value]	[Value]
PCK@0.2	[Value]	[Value]	[Value]
mAP	[Value]	[Value]	[Value]
Inference Time (ms)	[Value]	[Value]	[Value]

4 Discussion

4.1 Performance Analysis

Based on experimental results, the performance analysis of both models is as follows. Figure 7 shows direct comparison results of training loss and validation loss between the two models.

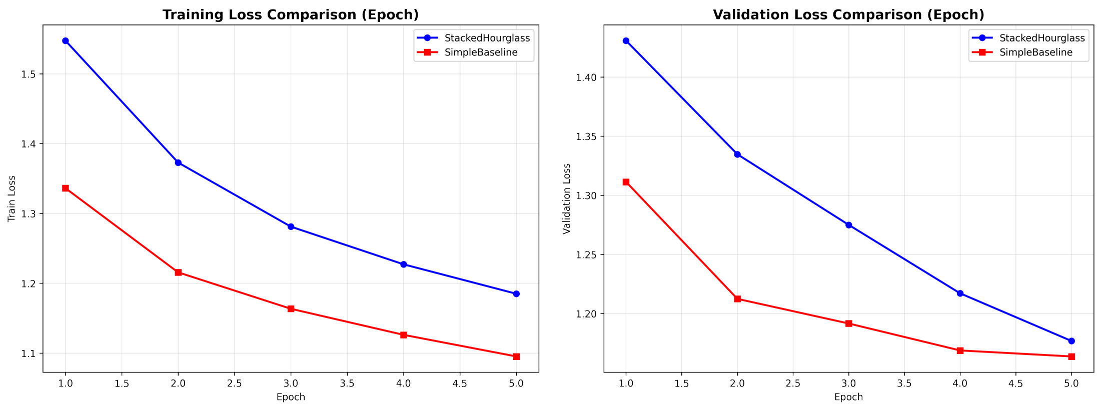


Figure 7: Loss comparison analysis between two models (Left: Training Loss, Right: Validation Loss)

- **Accuracy Aspect:** [Analysis of accuracy characteristics of each model]
- **Speed Aspect:** [Inference speed and efficiency comparison]
- **Stability Aspect:** [Training stability and convergence characteristics]

4.2 Action Sticker Application Perspective

Considering the actual application purpose of action sticker generation:

icality perspective

erience perspective

quirement analysis

4.3 Limitations

The limitations of this study include:

search limitation 1

search limitation 2

search limitation 3

5 Conclusions

In this study, we directly implemented and compared StackedHourglass Network and SimpleBaseline models for pose estimation in action sticker generation.

5.1 Key Findings

Key finding 1

Key finding 2

Key finding 3

5.2 Limitations

appointing point 1

appointing point 2

eding improvement

5.3 Future Expectations

research directions

ology development

cation possibilities

6 References

References

[1] Newell, A., Yang, K., Deng, J.: Stacked Hourglass Networks for Human Pose Estimation. arXiv preprint arXiv:1603.06937 (2016)

[2] Xiao, B., Wu, H., Wei, Y.: Simple Baselines for Human Pose Estimation and Tracking. arXiv preprint arXiv:1804.06208 (2018)

[3] Learning Node Provided Material 3: [Additional reference material]

7 Tables

Table 2: Architecture comparison between StackedHourglass Network and SimpleBaseline models

Component	**StackedHourglass**	**SimpleBaseline**
Input Resolution	256×256×3	256×256×3
Network Depth	4-stack structure	ResNet50 + 3 Deconv
Main Block Structure	Hourglass Module + Bottleneck Residual	ResNet Residual + Deconvolutional
Output Resolution	64×64×16 (each stack)	64×64×16 (single output)
Parameter Count	Approx. 25.1M	Approx. 34.0M
Features	Multi-scale supervision	Simple and efficient
Skip Connection	Inter-stack connections	ResNet internal connections only

Table 3: Details of applied metrics and benchmarks

Metric/Benchmark	**Description**
PCK@0.5	Percentage of Correct Keypoints at 0.5 threshold
PCK@0.2	Percentage of Correct Keypoints at 0.2 threshold
mAP	mean Average Precision
Inference Time	Average time required for single image processing

8 Figures

Figure 8: Additional experimental results and various pose estimation result examples