On the Design of 2D Human Pose Estimation Networks using Accelerated Neuroevolution and Novel Keypoint Representations

by

William McNally

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2022

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	James Little Professor, Dept. of Computer Science, University of British Columbia
Supervisors:	John McPhee Professor, Dept. of Systems Design Engineering, University of Waterloo
	Alexander Wong Professor, Dept. of Systems Design Engineering, University of Waterloo
Internal Member:	David Clausi Professor, Dept. of Systems Design Engineering, University of Waterloo
Internal Member:	John Zelek Associate Professor, Dept. of Systems Design Engineering, University of Waterloo
Internal-External Member:	William Melek Professor, Dept. of Mechanical and Mechatronics Engineering, University of Waterloo

Author's Declaration

Statement of Contributions

Chapters 3 and 4 contain material from two multi-author papers for which I was the lead author. As the lead author, I was responsible for conceptualizing the studies, carrying out all code development (except as indicated in the co-author contributions below), and drafting and submitting the manuscripts. My co-authors, which included my colleague Kanav Vats (PhD Candidate) and my supervisors Prof. McPhee and Prof. Wong, provided guidance throughout the process and feedback on draft manuscripts. The references for the two papers are provided below:

1. **McNally, W.**, Vats, K., Wong, A. and McPhee, J., 2021. EvoPose2D: Pushing the boundaries of 2D human pose estimation using accelerated neuroevolution with weight transfer. *IEEE Access*, *9*, pp. 139403-139414.

   Contribution of K. Vats: implemented the PoseFix post-processing algorithm.

2. **McNally, W.**, Vats, K., Wong, A. and McPhee, J., 2022. Rethinking keypoint representations: Modeling keypoints and poses as objects for multi-person human pose estimation. Submitted to ECCV 2022. *arXiv preprint arXiv:2111.08557*

   Contribution of K. Vats: technical review and assisted with exploratory experiments.

Abstract

Motion capture is a very useful technology that is employed across many industries. Biomechanical analysis, film production, video game development, and virtual reality are among its diverse applications. However, traditional *marker-based* motion capture systems are limited by their invasiveness, excessive cost, and lack of portability. Human pose estimation represents a promising *markerless* alternative, where 3D human poses are estimated from RGB images obtained using single or multi-camera setups. The estimation of 2D poses serves as the main foundation for these systems. As such, the development of accurate and efficient 2D human pose estimation algorithms is critical to the overall advancement of markerless motion capture, and that is the focus of this thesis.

Two novel convolutional neural networks for 2D human pose estimation are presented, one for each of the two multi-person estimation paradigms (*i.e.*, single-stage and two-stage). Motivated by the recent use of neural architecture search for convolutional neural network design, a novel neuroevolution framework is introduced and is leveraged in the design of a computationally efficient heatmap-based human pose estimation network for use in the two-stage paradigm. The neuroevolution was accelerated by a novel weight transfer scheme that relaxes the complete function-preservation constraint imposed by previous methods.

Recognizing the drawbacks of heatmaps, including the inherent issue of quantization error and the excessive computation required to generate and post-process large heatmap fields, two novel heatmap-free keypoint representations are introduced for modeling keypoint locations more efficiently. Drawing inspiration from single-stage object detectors, the representations are centered around modeling individual keypoints and sets of spatially related keypoints (*i.e.*, poses) as objects. It is found that pose objects lend themselves well to single-stage human pose estimation, and a method is introduced that jointly learns human pose objects and keypoint objects and fuses the detections to exploit the strengths of both object representations.

At the time of development, both networks achieved state-of-the-art accuracy in their respective categories on the most established multi-person human pose estimation benchmarks when scaled. Moreover, they are more computationally efficient than previous networks as shown by having fewer parameters, fewer floating-point operations, and faster inference speed. The two-stage model is recommended for accuracy-critical scenarios with large computational budgets, whereas the single-stage model is more efficient and has greater potential for future research. It is hoped that these new models advance the current state of markerless motion capture systems based on human pose estimation.

Acknowledgements

First and foremost, I would like to thank my supervisors, Prof. John McPhee and Prof. Alexander Wong, for their incredibly dependable support and invaluable mentorship. I would also like to thank my committee members – Profs. David Clausi, John Zelek, and William Melek for providing helpful feedback during my comprehensive exam and through the completion of my research. A special thanks to Profs. Clausi and Zelek for involving me in the Sports Analytics Research Group and its related projects – I thoroughly enjoyed the discussions held during the weekly meetings.

I also want to thank the current members and alumni of the Motion Research Group and the Vision and Image Processing Lab, especially Dr. Brokoslaw Laschowski and Kanav Vats, for their frequent and fruitful collaborations. Finally, I would like to acknowledge the organizations and programs who funded this research or provided computational resources, including the University of Waterloo, the Natural Sciences and Engineering Research Council of Canada (NSERC), the Canada Research Chairs (CRC) Program, Compute Canada, the Waterloo Artificial Intelligence Institute, the TPU Research Cloud (TRC) Program, Microsoft Azure, and NVIDIA.

Dedication

This thesis is dedicated to my family. To my parents Karen and John, my sister Tess, and my brother Ed: thank you for your support and encouragement over the years.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Motion capture is the process of digitally encoding the movements of humans and/or objects. Such data has numerous applications that can broadly be grouped into three categories: surveillance, control, and analysis [1]. Surveillance applications include intelligent security systems (*e.g.*, recognizing abnormal activities in crowds, like in an airport [2, 3]). Control applications are widespread in the entertainment industry, where motion capture is used to control digital avatars for animation [4], virtual reality [5], and filmmaking [6]. Analysis applications are diverse and exist across many industries. Examples include clinical gait analysis [7], the biomechanical analysis of athletes [8], next-generation sports analytics [9, 10], and the validation of mobile robots [11, 12]. The sheer number of potential applications makes motion capture data highly valuable. However, traditional marker-based motion capture systems possess several limitations that prevent many applications from coming to fruition.

1.1 Limitations of Marker-based Motion Capture

Traditional marker-based motion capture systems (*e.g.*, Vicon, Qualisys, OptiTrack, *etc.*) use numerous infrared cameras to triangulate the 3D positions of spherical retroreflective markers placed on target subjects and/or objects. Although these systems are considered the "gold standard" for motion capture technology [13], they suffer from three main drawbacks: lack of portability, invasiveness, and cost. These limitations are discussed in more detail below.

1. First, **the use of markers prohibits "in-the-wild" applications** such as biomechanical analysis outside the lab. In the field of sports biomechanics, athletes must be recruited to participate in motion capture experiments held in research laboratories. Besides being inconvenient for the athletes, these experiments may cause the athletes to behave differently than they would in their natural sporting environment. The ability to perform biomechanical analysis in the wild therefore has implications to biomechanical research, as well as sports broadcasting, with the potential to provide biomechanical analytics to commentators in real-time and enhance the overall viewing experience.

2. Second, **fitting markers on a human subject is invasive and time-consuming**. If markers are placed over clothing they can move relative to the body, leading to position error. To minimize position error, a subject may be asked to wear very tight clothing, or remove their clothing altogether such that the markers can be placed directly on the body. Even then, the skin can translate relative to anatomical landmarks. A less invasive alternative involves using custom-fit motion capture suits on which markers can be placed. However, these suits are expensive and do not fully resolve the issues with position error. Furthermore, there is potential for the markers/suit to interfere with the natural biomechanics of the subject. This raises questions regarding the integrity of the collected data.

3. Finally, **marker-based motion capture systems are prohibitively expensive**. Most applications require numerous redundant cameras to track markers that may disappear behind body parts or other obstructions. If the activity requires a large space, more cameras are required to cover the entire capture volume. The 2K Motion Capture Studio is one of the largest motion capture studios in North America and is used for recording video game animations. It consists of 144 Vicon Vantage cameras that cost thousands of dollars each [14].

1.2 Human Pose Estimation: Markerless Motion Capture using RGB Video and Deep Learning

Given the limitations of marker-based motion capture systems, *markerless* motion capture using RGB video and image processing algorithms is an attractive alternative, and is a highly researched area in the field of computer vision. Early approaches to markerless motion capture used hand-designed feature descriptors (*e.g.*, histograms of oriented gradients (HOG) [15], scale-invariant feature transforms (SIFT) [16], *etc.*) to extract local features from RGB images and detect human body parts [17, 18, 19]. These custom-engineered feature descriptors demanded careful designs that were sensitive to different body parts yet resistant to input variations (*e.g.*, skin pigment, lighting, clothing, *etc.*). Recent advances in computer hardware and software, as well as the ubiquity of imaging data available online, has fostered the development of methods based on *deep learning* [20], and more specifically, *convolutional neural networks* (CNNs) for artificial visual perception [21]. These data-driven approaches automatically learn image features that are more tolerant to input variations and thus have led to significant improvements over classical image processing techniques, including the early part-based models used for markerless motion capture. Indeed, deep learning and CNNs in particular have transformed the entire field of computer vision. It is widely contended that the momentous transition into the "deep learning era" was marked by the influential work of Krizhevsky *et al.* [22], who popularized the use of graphics processing units (GPUs) for large-scale CNN training, and ultimately developed a CNN (AlexNet) that went on to win the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [23] by an astounding margin.

CNNs have been central to the advancement of markerless motion capture technology. In essence, they are a type of artificial neural network that are specially equipped to process 2D data (*e.g.*, image data). In brief, they contain a hierarchical arrangement of image filters that produce features that are pertinent to a particular visual perception task (*e.g.*, locating human joints). The convolutional layer is the main component of a CNN. It applies a set of image filters (*i.e.*, kernels) to an input tensor (*e.g.*, an image) using discrete 2D convolution [24]. The output tensor contains the filtered results, called feature maps. The kernel weights control the filtering mechanism. Depending on the arrangement of its weights, a kernel may blur the input, sharpen it, or detect edges [24]. The power of deep learning is harnessed by learning the optimal kernel weights directly from the data by minimizing a loss function representing the error. The network optimization (*i.e.*, "training") is performed by iteratively updating the network weights using optimization algorithms like stochastic gradient descent (SGD) or Adam [25], which implement back-propagation [26] to efficiently compute the weight gradients. By stacking several convolutional layers in series, the intermediate feature maps become increasingly representative. In actuality, modern CNNs contain hundreds of convolutional layers and millions of learnable parameters (*i.e.*, weights) [27]. Remarkably, CNNs have surpassed human experts in their ability to understand images in several instances [28, 29, 30, 31, 32, 33]. It is worth noting that new deep learning architectures inspired by the Transformer [34] used in natural language processing and built entirely on attention mechanisms (*i.e.*, without convolution) are swiftly making their way into computer vision (*e.g.*, Vision Transformer (ViT) [35], Swin Transformer [36], *etc.*). However, CNNs maintain a dominant presence in computer vision at the time of writing due to their efficiency and simplicity. Furthermore, recent research suggests that when CNNs are “modernized” using the training algorithms and architectural traits of recent transformers, they can outperform said transformers on multiple benchmarks [37].

As it pertains to this thesis, researchers have exploited CNNs to perform markerless motion capture in the form of **human pose estimation** – the frame-by-frame detection and localization of human joints and other anatomical landmarks, often referred to as *keypoints*, in RGB images and video [38]. Traditionally, the study of human pose estimation has focused on estimating 2D poses by localizing sets of keypoints in the image coordinates (*i.e.*, 2D human pose estimation [39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59]). Keypoint locations are commonly modeled using heatmaps [41], which are 2D fields representing the per-pixel likelihood for the existence of keypoints. Target heatmaps are generated by centering Gaussians on the ground-truth keypoint locations, and then a human pose estimation network (*i.e.*, a CNN) is trained to predict the heatmaps by minimizing a mean squared error loss over thousands of training examples. The equation used to generate a target heatmap $\mathrm{S_k} \in \mathbb{R}^{h' \times w'}$ corresponding to a ground-truth keypoint of class k located at (x'_k, y'_k) is given by:

$$\mathrm{S_k}(x', y') = 255 \exp\left(\frac{-(x' - x'_k)^2 - (y' - y'_k)^2}{2\sigma^2}\right) \tag{1.1}$$

with $x', x'_k \in [0, w']$ and $y', y'_k \in [0, h']$. h' and w' are the height and width of the target heatmap, respectively. Heatmaps are commonly parameterized with $\sigma = \frac{h'}{32}$ pixels (px) [51]. Figure 1.1 illustrates the appearance of heatmap predictions using the human pose estimation network developed in Chapter 3. A separate heatmap is predicted for each keypoint type, and the arguments of the maximum heatmap response are used to predict the keypoint locations in the heatmap coordinates. The keypoints are then transformed back to the coordinates of the original input image $\mathbf{I} \in \mathbb{R}^{h \times w}$. An example 2D pose prediction is overlaid on the input image in the left of Figure 1.1.

Multiple people may appear in an image and 2D human pose estimation algorithms should be able to handle such cases. There are two main approaches that address multi-person estimation; they are categorized by whether they use one or two inference stages. Two-stage approaches detect all the people in the image first using a person detector and then estimate poses in the second stage using a single-person human pose estimation network (*i.e.*, like the one used to generate the heatmaps in Figure 1.1). In contrast, single-stage approaches estimate the poses for all the people in the image following a single forward pass. This gives rise to a trade-off: two-stage methods can achieve higher accuracy,

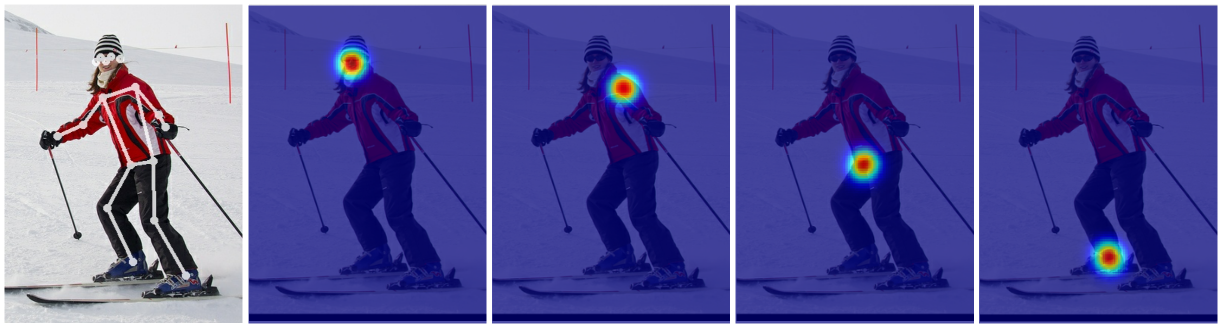


Figure 1.1: Sample keypoint heatmap predictions made using EvoPose2D, the 2D human pose estimation network developed in Chapter 3. From left to right: input image with the predicted 2D pose overlaid, sample heatmap predictions for the nose, left shoulder, right hip, and right ankle. The input image was sourced from the Microsoft COCO dataset [60].

but single-stage methods are more computationally efficient, especially when the number of people in the image is large.

The focus on 2D human pose estimation as opposed to 3D has largely been a consequence of data availability: gathering large amounts of diverse images from online sources and manually labeling the 2D keypoint locations is relatively straightforward, but this process cannot be extended to 3D keypoints since it is not feasible for humans to precisely annotate the depths of image keypoints. Instead, marker-based motion capture systems have been used to build datasets (*e.g.*, Human3.6M [61], HumanEva [62], MPI-INF-3DHP [63]) to support the development of 3D human pose estimation algorithms [64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86]. These algorithms estimate 3D positions from RGB images and as such, they represent a promising step in the direction of viable markerless motion capture for humans.

Significant progress has been made in 3D human pose estimation by virtue of 2D human pose estimation. A strong use case for the latter has emerged due to the fact that motion capture datasets lack diversity: all the images are collected from a single environment, and motions are recorded for a finite number of subjects performing a finite set of actions or gestures. As a result, 3D human pose estimation algorithms that naively map RGB images to 3D pose by training on motion capture datasets directly do not generalize well to new environments, subjects, or movements. Researchers have instead opted to use 2D human pose estimation models in the design of 3D human pose estimation algorithms because they have greater generalizability from training on diverse image datasets. A common approach is to "lift" 2D poses to 3D using a neural network. In the simplest implementation, the input to the neural network is an estimated 2D pose and the output is the 3D pose prediction for the same frame [67]. The 3D prediction is therefore conditioned on the 2D pose information rather than the original RGB image. In other words, 3D poses can be estimated for any image that can be processed by the underlying 2D model. To help resolve the ambiguities associated with lifting 2D poses to 3D, temporal information has been exploited using multi-frame sequences of 2D poses as input to sequential deep learning models (*e.g.*, recurrent neural networks [73, 87], 1D CNNs [75, 78], and transformers [81, 82, 83]). The methods that achieve the highest 3D reconstruction accuracy triangulate 2D poses from multi-view images using epipolar geometry [88, 89, 76, 79]. In any case, it is apparent that 2D human pose estimation is a vital component of 3D human pose estimation algorithms and as such, it is an important facet of markerless motion capture.

1.3 Research Overview

Evidently, 2D human pose estimation is an important precursor to 3D human pose estimation. The development of accurate and efficient 2D human pose estimation algorithms is therefore critical to the overall advancement of markerless motion capture technology, and such is the focus of this thesis. The pivotal role of 2D human pose estimation is underscored by the fact that the accuracy of 2D pose estimates directly influences the accuracy of 3D pose reconstructions [67]. Moreover, 2D human pose estimation models are responsible for the majority of the computational cost accrued in 3D human pose estimation pipelines; the models used to estimate 2D poses from images can be orders of magnitude more computationally expensive than the models used to lift 2D poses to 3D. To illustrate, a state-of-the-art 2D human pose estimation model introduced by Cheng *et al.* [54] (HigherHRNet) runs at approximately 1 frame per second on an NVIDIA TITAN Xp GPU [58]. For comparison, the 2D-to-3D pose lifting model of Pavllo *et al.* [75] runs at approximately 150k frames per second (FPS) on the same GPU.

A practical performance target for any computer vision algorithm is for it to run at least as fast as the input video frame rate (*i.e.*, in “real-time”), which ranges from 24 to 60 FPS for most applications. Seemingly, this is already a challenge in human pose estimation using workstations with powerful GPUs (as evidenced by the speed of HigherHRNet on a TITAN Xp GPU). There is also a desire to make this technology portable and accessible to everyone, and so a more ambitious goal is to have these algorithms run in real-time on mobile devices (*i.e.*, on smartphones and tablets). This poses an even greater challenge given the limited computational resources available on mobile devices compared to GPU workstations and servers. In summary, there remains ample room to improve the computational efficiency of 3D human pose estimation algorithms from a practical standpoint, and if such is the objective, improving the efficiency of the underlying 2D human pose estimation models should be prioritized given their substantial computational demand.

The size and speed of a CNN can be improved without modifying the network architecture using compression techniques such as pruning [90], quantization [91] and Huffman coding [92]. For example, Han *et al.* [93] reduced the storage required by AlexNet by 35×, from 240MB to 6.9MB, without loss of accuracy and observed a speedup of 3× to 4× on CPU and GPU. Importantly, such compression techniques are model-agnostic, meaning they can be applied to any model or architecture. This research instead focuses on the design of CNN architectures that are inherently more efficient.

To this end, two 2D human pose estimation models were developed as a part of this thesis, one for each of the multi-person estimation paradigms (*i.e.*, two-stage and single-stage). In the two-stage paradigm, a novel neuroevolution framework was used in the design of a heatmap-based human pose estimator. A new flexible weight transfer scheme was introduced to accelerate the neuroevolution, where pretrained weights in a parent network are transferred to mutated child networks to improve the weight initialization in future training cycles. The weight transfer method is simpler and has fewer constraints than network morphisms [94] and function-preserving mutations [95], and led to a six-fold reduction in the neuroevolution runtime. While two-stage approaches are generally more accurate, single-stage approaches have simpler implementations and are more efficient for multi-person estimation. Furthermore, recent research has highlighted the computational inefficiencies associated with the heatmap keypoint representation [96, 97]. Motivated by this, new keypoint representations are introduced in this thesis by modeling individual keypoints and entire poses as objects. The pose object representation lends itself well to single-stage human pose estimation, and so a novel heatmap-free single-stage human pose estimation network was engineered around the idea. In the context of CNN design, there is often a trade-off between computational efficiency and accuracy, but the most impactful contributions lead to improvements in both. Notably, both the proposed models achieve state-of-the-art accuracy in their respective categories on widely used human pose estimation benchmarks while being more computationally efficient than previous methods. An outline of the thesis is provided in the next section.

1.4 Thesis Outline

Chapter 2 provides a review of the literature. Section 2.1 describes the relevant datasets and the AP/AR accuracy metrics that are used for the evaluations in the subsequent chapters.

Section 2.2 provides a summary of the recent work in 2D human pose estimation, object detection, and neural architecture search.

Chapter 3 introduces EvoPose2D, a human pose estimation network for two-stage inference that was designed using a novel accelerated neuroevolution method exploiting a simple yet effective weight transfer scheme. This research represents the first application of neuroevolution to human pose estimation (*i.e.*, 2D or 3D), and more generally, the first application of neural architecture search to human pose estimation in which state-of-the-art accuracy has been achieved.

Chapter 4 introduces KAPAO, an entirely new method for single-stage human pose estimation that does away with computationally expensive heatmaps and uses new keypoint representations in which individual keypoints and poses are modeled as objects. Chapters 3 and 4 each include a brief introduction describing the motivation behind the research, a description of the proposed methodology, supporting experiments and analysis of the results, and a final discussion. Section 4.4 discusses the advantages/disadvantages of EvoPose2D and KAPAO and provides potential use cases for each.

Chapter 5 concludes the thesis with a summary of the research and the contributions made to the field of human pose estimation. Finally, recommendations are made and potential areas for future work are discussed.

The figures in this thesis are best viewed in colour. In all tables, the best results are emphasized with **bold** typeface. Unless "3D" is explicitly referenced, any further discussion of human pose estimation henceforth relates to 2D human pose estimation.

Chapter 2

Background

This chapter discusses the relevant background for the research presented in Chapters 3 and 4. An overview of human pose estimation datasets is provided first, placing emphasis on the datasets used for training and validating the models developed in this thesis. A discussion of the related work is then given for the areas of human pose estimation, object detection, and neural architecture search.

2.1 Datasets

Two-dimensional human pose estimation using deep learning was made possible in large part by the release of public image datasets containing annotations for the coordinates of human joints and other anatomical keypoints. Leeds Sports Pose (LSP) [98] and FLIC [99] were two of the first datasets of this type and were formed of images containing a single person. LSP contained 2k images of humans playing sports gathered from the online image repository Flickr, and the images were annotated with 14 keypoints. The FLIC dataset is slightly larger, containing 5k images extracted from popular Hollywood movies, with annotations for 11 upperbody keypoints. LSP was later extended to 10k images [100].

The early success of deep learning-based human pose estimation [40, 41] on LSP and FLIC spurred the release of larger datasets to support the development of human pose estimation models with greater capacity and generalizability. The MPII Human Pose Database [39] is arguably the first large-scale benchmark for 2D human pose estimation. It contains 25k images collected from 410 YouTube videos, including 40k person instances annotated with up to 16 keypoints.

The MPII dataset has been widely popular; however, it focuses primarily on single-person human pose estimation and does not consider person detection, which is an important component of deploying human pose estimation models in the real world. As a result, several large-scale *multi-person* human pose estimation datasets were subsequently released to benchmark end-to-end human pose estimation pipelines, including the detection of potentially numerous people, the estimation of their poses, and in some cases person tracking [101]. Microsoft COCO Keypoints [60], AI Challenger [102], PoseTrack [101], CrowdPose [103], and COCO-WholeBody [104] are examples of multi-person human pose estimation datasets (listed in chronological order of publication). Microsoft COCO Keypoints is the most established and frequently used dataset for benchmarking multi-person human pose estimation algorithms at the time of writing. It introduced robust accuracy metrics to handle the new multi-person estimation protocol. The COCO metrics were subsequently adopted by the more recent datasets. Microsoft COCO, CrowdPose, and PoseTrack were used in the development of the models presented in this thesis and are described in more detail in the following sections.

2.1.1 Microsoft COCO

The Microsoft COCO dataset [60] (COCO) is the one of the most commonly used datasets for benchmarking the performance of object detection, instance segmentation, and multi-person human pose estimation (referred to as *keypoint detection* by the dataset creators) models. COCO contains over 200k labeled images and an additional 123k unlabeled images to support semi-supervised and self-supervised methods. For the object detection and instance segmentation tasks, over 500k object instances comprising 80 object categories were annotated with bounding boxes and segmentation masks. For keypoint detection, over 250k person object instances were additionally labeled with up to 17 keypoints each, including the nose, eyes, ears, shoulders, elbows, wrists, hips, knees, and ankles. Between the training and validation subsets, over 1.7M labeled keypoints are publicly available for model development.

The models developed in this thesis were trained on the `train2017` split containing 118k images and 150k person instances. Validation was performed on the `val2017` split containing 5k images and 6.3k person instances. Additional testing was performed on the `test-dev` split, which contains 41k images. The labels for `test-dev` are not public and image predictions must be uploaded to an evaluation server to obtain the test results.

The accuracy metrics adopted for keypoint detection on COCO are based on common object detection metrics [105]. It is helpful to first describe the object detection metrics to better understand the metrics used for keypoint detection. The main object detection metric is called the average precision (AP) and is based on the intersection over union (IoU), a similarity measure between two bounding boxes. More explicitly, the IoU is the intersection area divided by the union area between a ground-truth ($\mathbf{b}$) and detected ($\hat{\mathbf{b}}$) object bounding boxes:

$$\text{IoU} = \frac{\text{area}(\mathbf{b} \cap \hat{\mathbf{b}})}{\text{area}(\mathbf{b} \cup \hat{\mathbf{b}})} \ . \tag{2.1}$$

A detected object is considered to be a true positive (tp) if its IoU with a ground-truth annotation is greater than some threshold α. If the IoU is less than α, the detected object is considered to be a false positive (fp). A ground-truth object can only have one corresponding true positive detection and any extra detections are considered to be false positives. The number of false negatives (fn) equals the number of ground-truth objects minus the number of true positives. The precision (p) and recall (r) metrics may then computed as follows:

$$p = \frac{tp}{tp + fp} \quad r = \frac{tp}{tp + fn} \ . \tag{2.2}$$

Each detected object has an associated confidence score, and detections that fall below a user-defined confidence threshold may be removed. Naturally, using a lower confidence threshold results in more false positives, lower precision, and higher recall. A precision-recall curve $p(r)$ is generated for a given IoU threshold α by sorting the detections across all the images by confidence score and evaluating the precision and recall over the full range of confidence thresholds (*i.e.*, from 0 to 1) [105]. The average precision at the IoU threshold α (AP$^{\alpha}$) is the area under the precision-recall curve:

$$\text{AP}^{\alpha} = \int_0^1 p(r)\ dr \ . \tag{2.3}$$

In practice, interpolation methods are used over a series of recall bins as $p(r)$ is not guaranteed to be monotonically decreasing. For COCO, 101 recall bins are used.

The primary COCO metric, referred to simply as the average precision (AP), is equal to AP$^{\alpha}$ averaged over ten IoU thresholds: $\alpha \in \{0.50, 0.55, ..., 0.95\}$. For the object detection and instance segmentation tasks, the AP is averaged again over the object categories. This metric has previously been called the mean average precision (mAP); however, COCO makes no distinction between the nomenclature and assumes the mean of the object categories is computed when applicable.

AP$^{.50}$ and AP$^{.75}$ correspond to the AP computed for the individual IoU thresholds of 0.50 and 0.75, respectively. AP^{S} is the AP computed for small objects (area $\leq 32^2$), AP^{M}

is for medium objects ($32^2 <$ area $\leq 96^2$), and AP^L is for large objects (area $> 96^2$), as measured in the original image coordinates in units of pixels. The average recall (AR) is equal to the maximum recall (*i.e.*, when the confidence threshold is set to 0) averaged over the IoU thresholds and object categories. It is computed for a fixed number of detections per image. For object detection and instance segmentation, the number of detections is 100. For keypoint detection, the number of detections is 20.

At the core of the described object detection metrics is the IoU, which effectively represents a measure of similarity between a detected and ground-truth object bounding box. The IoU is also used in the instance segmentation task to generate the same metrics, except the areas are computed using the object segmentation masks as opposed to the bounding boxes. It follows that the same metrics can be adopted for keypoint detection using an analogous similarity measure for human poses. The Object Keypoint Similarity (OKS) was introduced for this purpose. It is a summation of Gaussian-scaled Euclidean distances between the ground-truth and predicted keypoints of a human pose. More explicitly, it is defined as follows:

$$\text{OKS} = \frac{\sum_{i=1}^{K} \exp(-d_i^2/2s^2k_i^2)\delta(\nu_i > 0)}{\sum_{i=1}^{K} \delta(\nu_i > 0)} \tag{2.4}$$

where K is the number of keypoint types (K=17 for COCO), d_i are the Euclidean distances between the ground-truth and predicted keypoints (units of pixels, computed in the original image coordinates), and ν_i are visibility flags of the ground-truth keypoints ($\nu = 0$: not labeled, $\nu = 1$: labeled but not visible, and $\nu = 2$: labeled and visible). The Gaussian standard deviation k_i is specific to the keypoint type and is scaled by the area of the person bounding box s, measured in pixels. For each keypoint type, k_i reflects the standard deviation of human annotators when labeling keypoints of type i and is computed from a set of 5k redundantly annotated images [60]. A perfectly predicted pose has an OKS of 1; however, only 70% of human annotated keypoints fall within a keypoint similarity $\exp(-d_i^2/2s^2k_i^2) > 0.85$ [106, 60].

The OKS directly replaces the functionality of IoU and is used to generate the same AP/AR metrics for the keypoint detection task with the exception of AP^S, which is not considered as no keypoints were labeled for the small person instances.

2.1.2 CrowdPose

CrowdPose [103] is a multi-person human pose estimation dataset containing an abundance of images with heavy occlusion caused by crowded scenes with overlapping people. It is considered a more challenging dataset, especially for two-stage human pose estimation methods that rely on isolating individual people using person detectors. The dataset was designed with the help of a metric called the *Crowd Index*, introduced to assess the crowding level in images and computed using the labeled keypoint and bounding box data [103]. The Crowd Index was evaluated for all the images in COCO [60], MPII [39], and AI Challenger [102], and it was found that these datasets were dominated by uncrowded scenes (Crowd Index ≤ 0.1).

To form the CrowdPose dataset, a total of 30k images were uniformly sampled by Crowd Index. For consistency, all the images were relabeled with 14 keypoints, including the top of the head, neck, shoulders, elbows, wrists, hips, knees, and ankles. The Crowd Indices were recomputed and the dataset was subsampled to 20k high quality images containing approximately 80k person instances. The 20k images were randomly split into 10k training images, 2k validation images, and 8k test images. A protocol frequently used in the literature is to train on the 12k images in the combined `trainval` split and then evaluate on the 8k held out test images. The dataset accuracy metrics are consistent with COCO and include AP, AP^{50}, and AP^{75}. AP^{E}, AP^{M}, and AP^{H} are additionally considered for images with easy (0-0.1), medium (0.1-0.8), and hard (0.8-1) Crowd Index.

2.1.3 PoseTrack

PoseTrack [101] is a large-scale benchmark for multi-person human pose estimation and person tracking in video. The dataset contains 1,356 video sequences and 46k frames annotated with 15-keypoint poses and person identifiers for tracking. In total, the dataset includes 276k annotated person instances. While person tracking is an integral component of deploying human pose estimation models for practical applications, tracking algorithms typically run on top of human pose estimation models and are developed independently [51]. This thesis focuses on the design of human pose estimation models specifically, and therefore person tracking falls outside the scope of this research. Nonetheless, the PoseTrack dataset was used to assess the generalizability of some of the developed models for single-frame multi-person pose estimation, which is one of the three official challenges of the PoseTrack dataset (multi-frame multi-person pose estimation and multi-person pose tracking are the other two). For convenience, the dataset was converted to COCO format and the COCO evaluation toolbox was used to compute the standard AP/AR evaluation metrics. Specifically, the 2018 version of the dataset was used with the v0.45 labels. The training split contains 97k person instances and the validation split contains 45k person instances.

2.2 Related Work

2.2.1 Human Pose Estimation

Two-dimensional human pose estimation is popular topic in computer vision that deals with the autonomous localization of human joints and other anatomical keypoints in RGB images and video [40, 41, 39]. More broadly, it is a form of *keypoint detection*, a family of computer vision algorithms concerned with localizing keypoints regardless of whether or not they are anatomically-based (*e.g.*, Li *et al.* [107] detect keypoints on vehicles). Emerging as one of the most highly researched topics in the literature, keypoint detection plays an important role in several computer vision applications, including 2D and 3D human pose estimation [57, 75], hand pose estimation [108, 109], action recognition [110, 111, 112], object detection [113], multi-person tracking [101, 114], facial and object landmark detection [115, 116, 117, 118, 119], and sports analytics [120, 96]. Of these diverse applications, 2D human pose estimation is one of the most representative keypoint detection tasks and is widely considered a fundamental problem in computer vision. It has several direct downstream applications, including pose-based action recognition [121, 111, 110] and pose-based human tracking [122, 101, 51]. Moreover, as highlighted in the previous chapter, it is the precursor to 3D human pose estimation, which serves as a potential alternative to invasive and expensive marker-based motion capture.

An overview of the various types of 2D human pose estimation models is provided. Following the chronology of the literature, the methodologies can broadly be grouped into earlier single-person models and the more recent multi-person models. Single-person models are stratified further into traditional methods that use part-based pictorial structures versus those that use deep learning. Multi-person approaches are predominantly deep learning-based but can be grouped into two categories based on whether they use two-stage inference (*i.e.*, including a person detector) or single-stage inference. Figure 2.1 provides an overview of the model classifications and references for each. The following sections review the literature for each type.

Part-Based Models (Pre-Deep Learning)

Interest in the idea of using image processing to detect human features dates back to as early as 1973, when Fischler and Elschlager [123] introduced pictorial structures, a deformable part model, to recognize facial attributes in photographs. Coincidentally, the first publication from Dr. Geoffrey E. Hinton, a highly influential scientist in the deep learning community, dealt with the problem of arranging arbitrary sets of rectangles into a

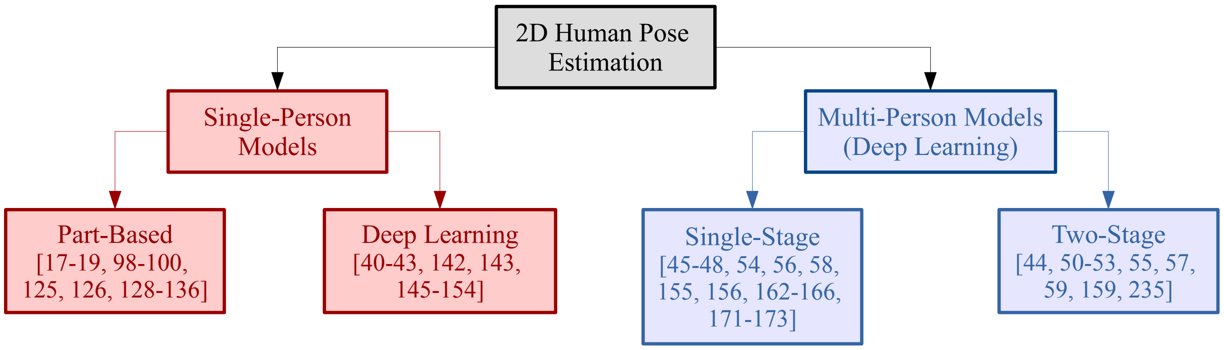

2D Human Pose Estimation
Single-Person Models
Multi-Person Models (Deep Learning)
Part-Based [17-19, 98-100, 125, 126, 128-136]
Deep Learning [40-43, 142, 143, 145-154]
Single-Stage [45-48, 54, 56, 58, 155, 156, 162-166, 171-173]
Two-Stage [44, 50-53, 55, 57, 59, 159, 235]

Figure 2.1: A classification of human pose estimation models and references for each type.

puppet-like figures [124] (1976). Hinton went on to contribute to a landmark 1987 work that popularized the backpropagation algorithm [26]. Motivated by the early work of Fischler and Elschlager, in 2000 Felzenszwalb and Huttenlocher [17] proposed an efficient global matching algorithm for pictorial structures and applied it to predict articulated human poses using a geometric human body model made up of object parts. The primitive object part models consisted of manually defined rectangles with fixed aspect ratios, an average colour, and colour variance. They later refined their method by learning the object model parameters from 10 example training images using maximum likelihood estimation [18].

The vast majority of methods leading up to the deep learning era were based around deformable part models / pictorial structures but used competing machine learning methods for detecting body parts and modeling their connections [125, 126, 127, 128, 129, 130, 131, 132, 100, 133, 134, 135, 136, 19, 99]. To highlight a few, Ronfard *et al.* [126] trained support vector machines (SVMs) [137] to classify image patches based on features extracted using Gaussian filters; Ramanan and Sminchisescu [127, 128] used conditional random fields; Andriluka *et al.* [131] used shape context descriptors to train AdaBoost [138] classifiers; and Pishchulin *et al.* [19] conditioned pictorial structures on poselets [139] based on histograms of oriented gradients [15]. With the introduction of CNNs that automatically learn image features directly from image data, hand-engineered part-based models have since become obsolete. There was also a shift towards regressing keypoints locations as opposed to detecting body parts due to the public release of large keypoint datasets like LSP [98, 100], FLIC [99], and MPII [39].

Single-person Human Pose Estimation using Deep Learning

One of the first uses of deep learning in 2D human pose estimation came in 2014, when Toshev and Svegedy [40] regressed keypoint coordinates directly from RGB images using a 7-layer CNN. Their method, called DeepPose, invoked a cascaded approach that refined keypoint predictions over successive stages. By learning features directly from the image data, DeepPose laid the foundation for a series of CNN-based methods offering superior performance over previous part-based models that relied on hand-crafted part feature descriptors. Its use of iterative keypoint refinement also had a significant influence on future works [42, 45, 43, 50, 53].

In the same year, Tompson *et al.* [41] argued that the direct regression of pose vectors from images was a highly non-linear and difficult to learn mapping. Indeed, the regression of pose vectors requires the use of fully-connected layers and therefore does not exploit the translational equivariance property of convolutions to its full potential [140]. Fully-convolutional networks that output dense representations exploit this property more effectively, which may facilitate the learning of spatial mappings [141]. Tompson et al. therefore introduced the notion of keypoint heatmaps, which model keypoint locations using 2D fields containing Gaussians with small variances centered on the keypoint coordinates (see Figure 1.1 for sample heatmap predictions). They trained their network using a mean squared error (MSE) loss that minimized the error between the predicted and target heatmaps and then predicted keypoint locations using the arguments of the predicted heatmaps maxima. Numerous researchers immediately took inspiration from the heatmap representation: Carreira *et al.* [142] refined heatmap predictions with iterative error feedback; Hu and Ramanan [143] regressed heatmaps using a hierarchical rectified Gaussian model; Rafi *et al.* [144] regressed similar binary belief maps; and Bulat and Tzimiropoulos [145] regressed heatmaps using a CNN cascade consisting of two connected subnetworks. Another contribution of Tompson *et al.* was the fusion of feature maps learned from different input resolutions [146]. The use of heatmaps and multi-scale fusion proved to be effective techniques for human pose estimation and continue to be used in many state-of-the-art models [50, 52, 53].

In 2016, two seminal single-person methods using iterative heatmap refinement in a multi-stage fashion with intermediate supervision were introduced. Wei *et al.* [42] introduced Convolutional Pose Machines, which used numerous identical convolutional stages that operated directly on heatmaps from previous stages. At each stage, the spatial context of the heatmaps provided disambiguating cues for the next stage. Shortly thereafter, Newell *et al.* [43] made use of repeated top-down, bottom-up processing in "hourglass" convolutional modules. This style of CNN architecture was popularized by U-net [147], a semantic segmentation model for biomedical imaging, in the year prior. However, the hourglass modules built on the success of the recently published ResNet [27] using numerous skip connections and residual sub-modules, and were stacked with intermediate supervision to iteratively refine heatmaps. The Stacked Hourglass network garnered a good deal of attention and several works extended it directly or used the hourglass module as a building block in new approaches [148, 149, 150, 151, 152, 153, 154].

Quantitative comparisons on the MPII dataset give perspective to the rapid advancement attributed to deep learning and CNNs: one of the last pictorial structures models, that of Pischulin *et al.* [19], provided an accuracy of 44.1 PCKh (percentage of correct keypoints detected within 50% of the head segment length); the first deep learning heatmap-based method of Tompson *et al.* [41] provided an accuracy 76.9 PCKh; the Stacked Hourglass method of Newell *et al.* [43] provided an accuracy of 90.9 PCKh. This represents a relative accuracy improvement of over 100% in the span of three years.

Two-Stage Multi-Person Human Pose Estimation

As previously mentioned in Section 2.1, single-person human pose estimation has largely been succeeded by multi-person human pose estimation, which integrates person detection and is more representative of how human pose estimation models are deployed in practice. Methods for multi-person human pose estimation generally fall into two categories: single-stage methods (*e.g.*, [45, 46, 47, 155, 54, 56, 156, 58]) and two-stage methods (*e.g.*, [42, 43, 50, 51, 52, 57, 59]). Figure 2.2 provides an overview of the two approaches. Two-stage methods emerged as a natural extension of single-person human pose estimation, where off-the-shelf person detectors (*e.g.*, Faster R-CNN [157], YOLOv3 [158], *etc.*) were used in the first stage to detect the people in the image, and single-person human pose estimators were run over the detections in the second stage. Two-stage methods are commonly referred to as being *top-down* for this reason. The model developed in Chapter 3 is an example of a two-stage multi-person human pose estimation model.

Since two-stage human pose estimation methods use preexisting person detectors, recent research has focused on improving the accuracy and efficiency of the single-person human pose estimators used in the second stage. In one of the first top-down approaches, Papandreou *et al.* [159] used Faster R-CNN [157] to detect person bounding boxes and developed a human pose estimator to predict disk-shaped heatmaps around each keypoint and offset fields that point towards the exact keypoint position within each disk, aggregating them in a weighted voting process. They also introduced OKS-based non-maximum suppression (NMS) to remove redundant pose detections during post-processing. A pose-based

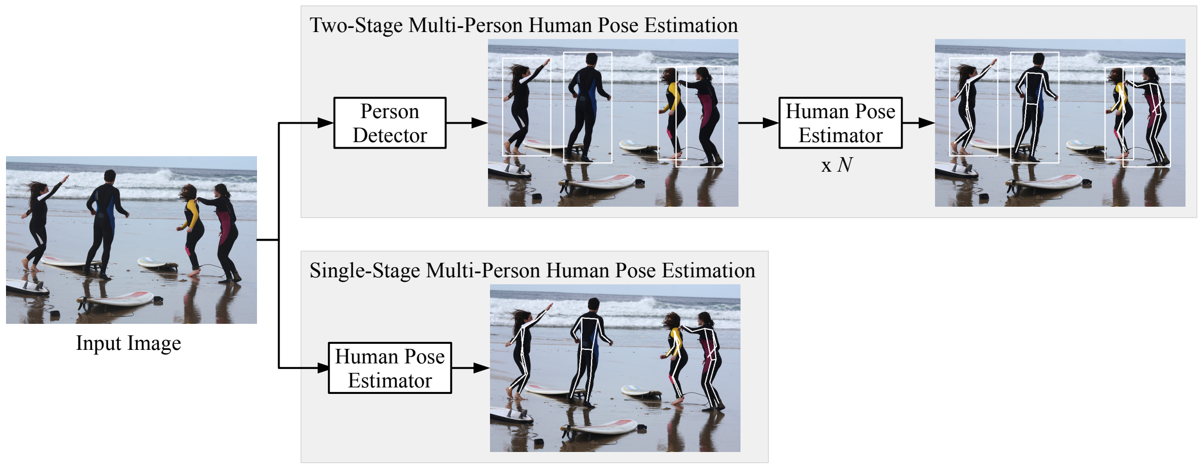

Two-Stage Multi-Person Human Pose Estimation
Person Detector
Human Pose Estimator
x N
Input Image
Single-Stage Multi-Person Human Pose Estimation
Human Pose Estimator

Figure 2.2: High-level comparison of two-stage and single-stage multi-person human pose estimation approaches. Two-stage approaches first detect the people in the image using a person detector and then run a single-person human pose estimator N times, where N is the number of detected people. Single-stage approaches predict N poses following a single forward pass through the human pose estimation network. The input image was sourced from the CrowdPose dataset [103]. The detections and poses were generated using the human pose estimation model developed in Chapter 4.

NMS algorithm was also proposed in Regional Multi-Person Pose Estimation (RMPE) [44] around the same time.

Remarking the computational inefficiencies associated with stacking repeated modules in the hourglass network, Chen *et al.* [50] introduced the Cascaded Pyramid Network (CPN), a holistically designed network (*i.e.*, without identical module stacking) including a feature pyramid network [160] with a ResNet-50 [27] backbone. Intermediate supervision was applied at each level of the feature pyramid to progressively refine the heatmap predictions. The CPN included an adjacent network that further refined the heatmaps produced by the feature pyramid and efficiently combined them across multiple scales. As a result, the CPN outperformed an 8-stage hourglass network at less than a third of the computational cost [50]. The CPN placed first in the 2017 COCO Keypoints Challenge.

A year later, Xiao *et al.* [51] introduced a remarkably simple human pose estimation model that added three transpose convolutions to ResNet [27]. The authors reported small performance gains over CPN while using a single-branch architecture and no intermediate supervision. However, the use of more accurate person detections raises questions regarding the fairness of the comparisons. Nonetheless, there is no arguing that the SimpleBaseline

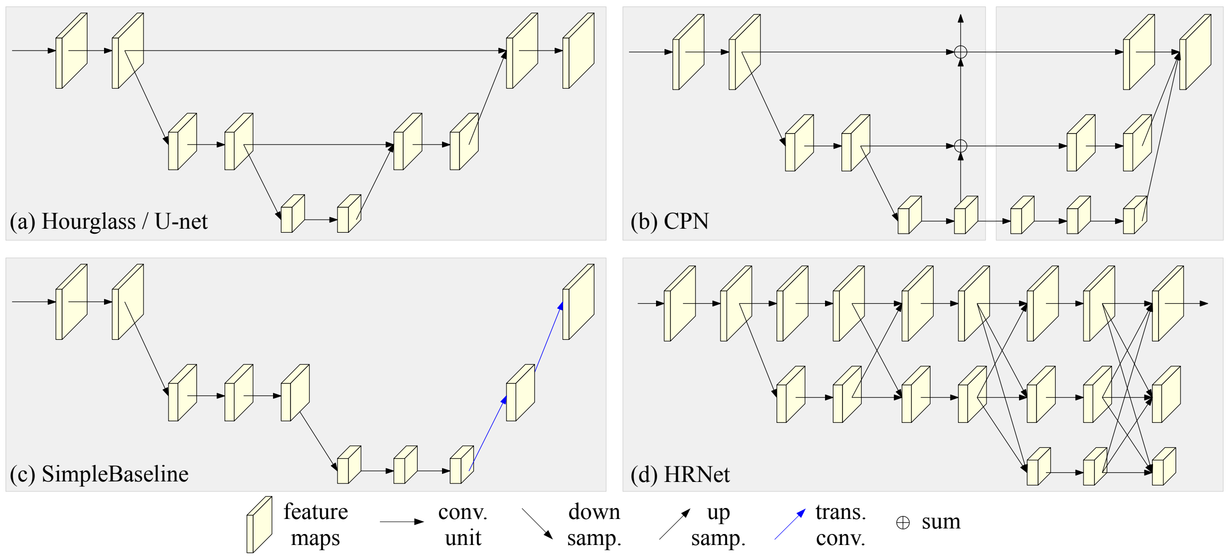


Figure 2.3: Common CNN topologies used in the second stage of the two-stage multi-person human pose estimation pipeline. (a) Hourglass / U-net [147] module used in the Stacked Hourglass Network [43]. (b) Cascaded Pyramid Network (CPN) [50], including GlobalNet feature pyramid (left) and adjacent RefineNet (right). (c) SimpleBaseline [51] single-branch architecture. (d) HRNet [52] multi-branch architecture. Figure adapted from [52].

model of Xiao *et al.* [51] has faster inference speed than the CPN due to its architectural simplicity. While no speed comparisons were made in the original paper, it is observed that SimpleBaseline runs 6.4× faster on a TITAN Xp GPU[1].

Sun *et al.* [52] later observed that most existing methods must recover high-resolution features from low-resolution feature embeddings. In another holistically designed network called HRNet, they showed that high-resolution features can be maintained throughout the whole network and as a result, they reported better accuracy using fewer floating point operations (FLOPs) compared to SimpleBaseline. Further comparisons presented in Chapter 3 contradict these findings. Using the TensorFlow Profiler API, it is found that HRNet-W48 actually uses more FLOPs than SimpleBaseline (ResNet-152). Furthermore, HRNet-W48 runs significantly slower on a TITAN Xp GPU due to its complex multi-branch architecture that combines feature maps across multiple scales (see Table 3.5 for

[1]SimpleBaseline ran at 47 FPS versus 7.3 FPS for the CPN. Excludes person detection. Averaged over 1k inferences using a batch size of 1, horizontal flipping, ResNet-50 backbone, and an input size of 384x288. TensorFlow 2.3 was used for the SimpleBaseline implementation and TensorFlow 1.15 was used for the CPN implementation.

details). Figure 2.3 compares the various CNN topologies that are used in human pose estimation, including the Hourglass, CPN, SimpleBaseline, and HRNet models.

Intrigued by the fact that multi-stage pose estimation networks (*e.g.*, Convolutional Pose Machines [42] and Stacked Hourglass [43]) seemingly underperform on the COCO benchmark despite showing good performance on MPII, Li *et al.* [53] revisited the multi-stage design with the multi-stage pose network (MSPN). Repurposing the CPN feature pyramid as a single stage in a multi-stage design, they achieved state-of-the-art accuracy on COCO. However, their architecture was computationally inefficient as a result of stacking several large modules together. Two-stage top-down methods generally perform worse in crowded scenes due the ambiguity caused by overlapping person detections. To combat this issue, Khirodkar *et al.* [59] introduced the Multi-Instance Pose Network (MIPNet), which allowed for predicting multiple pose instances from a single bounding box. As a result, MIPNet performed better than HRNet on datasets with greater occlusion, including CrowdPose [103] and OCHuman [161].

Single-Stage Multi-Person Human Pose Estimation

In contrast to two-stage methods, single-stage methods predict the poses of every person in an image following a single forward pass through the human pose estimation network (see Figure 2.1 for high-level differences between single-stage and two-stage methods). Single-stage methods are less accurate than their two-stage counterparts, but usually perform better in crowded scenes [103] and are often preferred because of their simplicity and efficiency, which becomes particularly favourable as the number of people in the image increases. Single-stage approaches vary more in their design compared to two-stage approaches. For instance, they may:

(i) detect all the keypoints in an image and perform a *bottom-up* grouping into human poses [162, 163, 164, 45, 46, 54, 155];

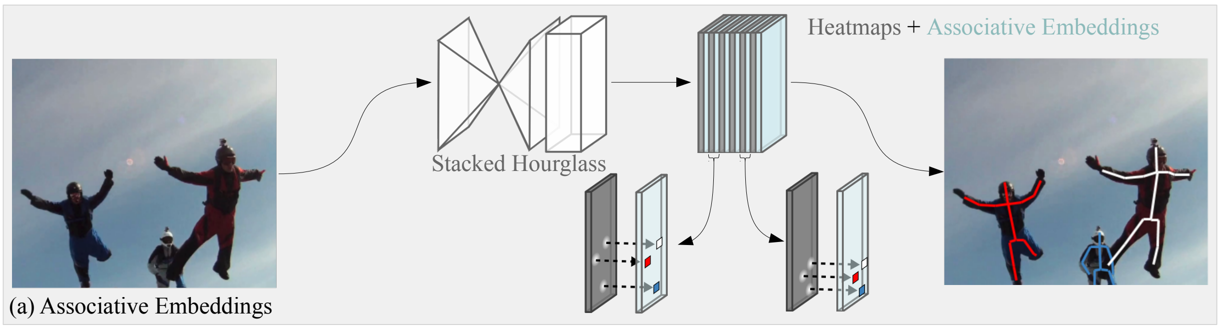
(ii) unify person detection and keypoint estimation, which is similar to the model developed in Chapter 4, Mask R-CNN [47], Point-Set Anchors [165], and FCPose [166], all of which extend object detectors to perform human pose estimation; or

(iii) use alternative keypoint/pose representations (*e.g.*, predicting root keypoints and relative keypoint displacements [48, 156, 56])

Widely regarded as the first single-stage bottom-up approach to multi-person human pose estimation, Pishchulin *et al.* [162] proposed DeepCut to partition a set of body part (keypoint) hypotheses generated with CNN-based part detectors. The formulation was cast as an integer linear program that operated over a fully-connected graph of pairwise probabilities. However, the minimum cost multicut problem is known to have NP-hard complexity [167], and the solution required hours of processing [45]. The algorithm was improved a year later with DeeperCut [163], which used stronger part detectors, image-conditioned pairwise terms, and a new optimization method to reduce the runtime by orders of magnitude. Still, the processing time per image was on the order of minutes [45]. Iqbal and Gall [164] used a similar integer linear programming approach to DeepCut while using Convolutional Pose Machines to detect keypoints.

In a more efficient bottom-up scheme, Cao *et al.* [45] extended Convolutional Pose Machines by additionally predicting "part affinity fields" that encoded the orientations of body segments. Colloquially known as OpenPose [168], their method detected all the keypoints in an image using heatmaps and used the jointly learned part affinity fields to perform a greedy bottom-up parsing over a set of bipartite matchings to predict human poses. The required processing time was orders of magnitude less than DeeperCut, and they achieved real-time multi-person inference on a GPU for the first time. OpenPose also placed first in the inaugural COCO Keypoints Challenge in 2016. Around the same time, Newell *et al.* [46] extended the Stacked Hourglass Network in another bottom-up approach that jointly learned keypoint heatmaps and associative embedding tags that were used to group the detected keypoints into poses. The person tags were cleverly learned without explicit labeling using a novel two-piece grouping loss that "pushed" tag values in distinct poses away from each other and "pulled" tag values within a single pose together. Figure 2.4 illustrates the intuition behind the use of associative embedding tags and part affinity fields for bottom-up human pose estimation.

Kreiss *et al.* [155] extended the concept of part affinity fields to estimate composite fields called Part Intensity Fields and Part Association Fields (PifPaf). With the incorporation of a scale dependent Laplacian L1 loss [170], their approach outperformed previous methods on low resolution images. Cheng *et al.* [54] repurposed HRNet for bottom-up human pose estimation by adding a transpose convolution to double output heatmap resolution (HigherHRNet) and using associative embeddings for keypoint grouping. They also implemented multi-resolution training to address the scale variation problem. In doing so, HigherHRNet achieved new state-of-the-art accuracy for a single-stage model on COCO. Jin *et al.* [171] proposed an alternative method for directly learning bottom-up keypoint grouping. Their differentiable Hierarchical Graph Grouping (HGG) took keypoint candidates as graph nodes and clustered them using a multi-layer graph neural network model that was trained end-to-end. Luo *et al.* [172] used HigherHRNet as a base and proposed scale and weight adaptive heatmap regression (SWAHR), which scaled the ground-truth



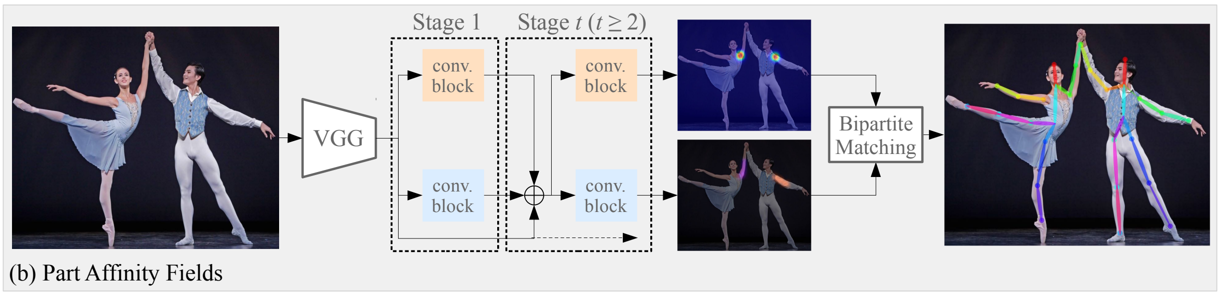


Figure 2.4: Two influential methods for the bottom-up parsing of keypoints into poses: (a) Associative Embeddings [46] and (b) Part Affinity Fields [45]. VGG is a CNN feature extractor [169]. Figure adapted from [46, 45].

heatmap Gaussian standard deviations based on the person scale and introduced a novel loss to help balance the foreground/background loss weighting. Their judicious modifications provided significant accuracy gains over HigherHRNet and comparable performance to many two-stage methods. Again using HigherHRNet as a base, Brasó *et al.* [173] proposed CenterGroup to match keypoints to person centers using a fully differentiable attention mechanism that was trained end-to-end together with the keypoint detector.

Straying away from bottom-up keypoint grouping, He *et al.* [47] used the Mask-RCNN instance segmentation model for human pose estimation by predicting keypoints using one-hot masks. Papandreou *et al.* [48] predicted keypoint heatmaps along with relative displacements from parent joints in the kinematic tree structure. Nie *et al.* [156] introduced the single-stage multi-person pose machine (SPM), which incorporated a root keypoint for each person and encoded the relative positions of the remaining keypoints via displacement maps. Wei *et al.* [165] proposed Point-Set Anchors, which adapted the RetinaNet [174] object detector using pose anchors instead of bounding box anchors. Mao *et al.* [166] adapted a different object detector (FCOS [175]) with FCPose, using dynamic filters [176] to process person detections and provide a favourable accuracy-speed trade-off. Using an

HRNet backbone, Geng *et al.* [56] predicted person center heatmaps and $2K$ offset maps representing offset vectors for the K keypoints of a pose candidate centered on each pixel. They also disentangled the keypoint regression (DEKR) using separate regression heads and adaptive convolutions.

2.2.2 Object Detection

Object detection involves detecting instances of physical objects in images and classifying them. Detections are made by estimating rectangular bounding boxes that enclose object extents. Based on the review in the previous section, it is apparent that there is much overlap between the tasks of object detection and human pose estimation. After all, body segments, keypoints, and poses may be thought of as objects even if they do not conform to the intuitive notion of a physical object. Earlier part-based models for human pose estimation adopted object detectors directly for body part detection. Several more modern approaches extend object detectors for keypoint detection. Notably, McNally *et al.* [96] model keypoints as objects directly in the development of an automatic scoring system for steel-tip darts. Conversely, keypoint detectors have been repurposed for object detection. Multiple groups of researchers have proposed to model objects as points by regressing bounding box centers and corners using heatmaps [177, 113, 178]. Coincidentally, two of these methods [113, 178] were published at the same time and share the same name: CenterNet. One version of CenterNet was applied to multi-person pose estimation [113]. Clearly, there is significant crossover between the two streams of research and object detection methods should therefore be explored as a means for advancing human pose estimation.

Before the deep learning era, traditional object detection methods scanned image regions using multi-scale sliding windows and extracted HOG [15], SIFT [16], or Haar-like features [179, 180]. SVMs [137] and AdaBoost [138] were commonly used to classify the extracted features into target object classes. Deformable part models were used to create more complex and precise feature representations [181]. In line with other research areas in image recognition, deep learning approaches leveraging CNNs have overshadowed traditional object detection methods. Analogous to human pose estimation, deep learning-based object detection methods are categorized into two groups based on whether they use one or two inference stages. Figure 2.5 provides an overview of the different architectural components in single-stage and two-stage CNN-based detectors. A modern detection architecture contains a backbone CNN feature extractor (*e.g.*, VGG [169], ResNet [27], MobileNetV2 [182], *etc.*), a "neck" that amalgamates feature maps from various stages of the backbone (*e.g.*, feature pyramid networks (FPN) [160], Path Aggregation Network

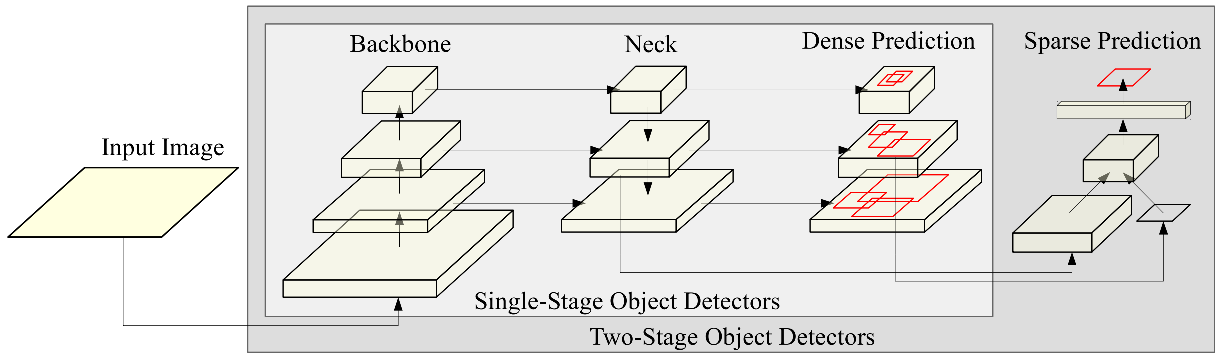


Figure 2.5: The various architectural components in modern CNN-based object detectors. Figure inspiration taken from [186].

(PAN) [183], BiFPN [184], NAS-FPN [185]), and a network head for dense or sparse prediction. Single-stage detectors with dense detection heads provide bounding box and class predictions for all locations in multi-scale output grids. Conversely, two-stage extract a sparse set of region proposals and classify them in a separate stage.

Two-stage Object Detectors

Popularized by the R-CNN family of networks (R-CNN [187], Fast R-CNN [188], Faster R-CNN [157], R-FCN [189], Mask R-CNN [47], Cascade R-CNN [190], and Libra R-CNN [191]), two-stage object detectors generate a sparse set of candidate object locations (*i.e.*, region proposals) in the first stage and classify each candidate into a foreground object class or background in the second stage. The original R-CNN detection framework [187] first used selective search [192] to extract around 2k class-independent region proposals per image. AlexNet [22] was then used to produce 4096-dimensional feature vectors for each region, which were classified using class-specific linear SVMs. An additional bounding box regressor was used to improve localization performance. Despite providing a relative mAP improvement of 30% on the PASCAL VOC 2012 benchmark [193], R-CNN possessed several drawbacks: the feature extractor input size was fixed to 227×227 due to the use of fully-connected layers, each component required separate training, and processing 2k region proposals was time consuming (13 s/image on a GPU). He *et al.* [194] proposed to speed up R-CNN using shared computation. They computed a feature map for the entire input image and then classified each region proposal by extracting fixed-length feature vectors from the shared feature map using a spatial pyramid pooling (SPP) layer based on spatial pyramid matching [195]. SPP-net accelerated the R-CNN framework 10 to 100×

at test time, but still suffered from having to train its components separately.

Girshick [188] unified the training of the feature extractor and classifier with Fast R-CNN. Similar to SPP-net, fixed-length feature vectors were extracted from each region proposal using a region of interest (RoI) pooling layer (a special case of the SPP layer that has only one pyramid level). Each feature vector was then fed into a sequence of fully-connected layers before branching into two output layers: one for producing softmax probabilities for the object categories and the other for encoding refined bounding box positions. The entire procedure (excluding region proposal generation) was optimized end-to-end using a multi-task loss. As a result, Fast R-CNN was 10$\times$ faster than SPP-net at test time and was also more accurate.

Finally, Ren *et al.* [157] integrated region proposals into the convolutional pipeline using a Region Proposal Network (RPN). The fully-convolutional RPN operated on features produced by the detection backbone and densely predicted object bounds and objectness scores at each position using template anchor boxes. The top-300 scoring region proposals were then classified in a similar manner as in Fast R-CNN. Due to sharing of convolutional layers, the Fast R-CNN and RPN were optimized end-to-end in an alternating fashion. The resulting Faster R-CNN network was an order of magnitude faster than Fast R-CNN, which used selective search to generate the region proposals. Improvements to Faster R-CNN were inevitably proposed: fully-convolutional classification eliminated costly fully-connected layers (R-FCN [189]); feature pyramid networks (FPN [160]) made better use of multi-scale feature maps; Cascade R-CNN [190] implemented a sequence of detectors trained with increasing IoU thresholds; and Libra R-CNN [191] alleviated training imbalance at the sample, feature, and objective level. The use of dense prediction and anchor boxes in the RPN greatly influenced imminent single-stage approaches.

Single-Stage Object Detectors

The more recent single-stage detectors unify the two-stage detection process by simultaneously classifying objects and regressing their locations over a dense grid. Popular single-stage detectors include the "You Only Look Once" (YOLO) family of detectors [196, 197, 158, 186, 198, 199], the Single Shot MultiBox Detector (SSD) [200], and RetinaNet [174].

The basic idea behind YOLO is that feature maps produced by the backbone are converted directly to a dense output grid. Each output grid cell contains information regarding a candidate detection, including the confidence that an object exists (objectness), the class probabilities, and the bounding box estimate. A grid cell is responsible for predicting a target object if the center of the target object lies inside the grid cell (*i.e.*, when the grid is overlaid on the input image). In the original YOLO network [196], two 7×7 output output grids were used for redundancy. During training, the grid whose cell had the highest current IoU with the target object was assigned to be responsible for making the prediction. This led to specialization amongst the output grids, where each grid became better at predicting certain sizes, aspect ratios, or classes. Due to its architectural simplicity, YOLO provided an admirable accuracy-speed trade-off. Using the VGG backbone, YOLO ran 3× faster than Faster R-CNN on GPU (21 FPS versus 7 FPS) but was less accurate by 6.8 mAP on the PASCAL VOC 2007 test set.

Inspired the anchor boxes used in the RPN, Liu *et al.* [200] incorporated anchor boxes into the Single-Shot MultiBox Detector (SSD). In a similar manner to YOLO, SSD generated scores for the presence of objects in each grid cell, but instead of predicting bounding boxes directly, it predicted adjustments to the anchor boxes to better match the object shape. Additionally, SSD used multi-scale output grids stemming from multiple feature maps with different resolutions to naturally handle objects of various sizes. Remarkably, SSD had comparable inference speed to YOLO and was more accurate than Faster R-CNN. YOLOv2 [197] later adopted the anchor box strategy, using k-means clustering to find optimal anchor box sizes. Other strategies, including batch normalization and multi-scale training, led to further accuracy improvements. While YOLOv2 performed better than SSD on the PASCAL VOC metric (IoU = 0.5), it was 7.2 AP less accurate than SSD on COCO (IoU = 0.5:0.95), which was an indication that it had good recall but less precise bounding box localization. Meanwhile, the two-stage Faster R-CNN with FPN was 7.4 AP more accurate than SSD on COCO [160].

Single-stage object detectors were initially considered to be a justifiable trade-off: they provided lower accuracy but better computational efficiency compared to two-stage detectors. Lin *et al.* [174] discovered that the extreme foreground-background class imbalance destabilized the training of single-stage detectors and was responsible for their inferior accuracy. To remedy the issue, they proposed the Focal Loss, which added a simple scaling factor to the standard cross entropy criterion. Using the Focal Loss to train a new dense detector called RetinaNet, they surpassed the accuracy of all existing two-stage detectors while matching the speed of previous single-stage detectors. Recent iterations of the YOLO detector have also surpassed the accuracy of two-stage detectors, interestingly without using the Focal Loss [158, 186, 198]. It was hypothesized that YOLO may already be robust to the problem the Focal Loss addresses due to the use of separate objectness and class predictions [158].

Like single-stage object detection methods, single-stage human pose estimation methods have the potential to be much faster than their two-stage counterparts. However, current state-of-the-art single-stage human pose estimators are heavily burdened by the processing of large heatmaps [54, 56, 172, 173]. As a result, their inference speed leaves much to be desired.

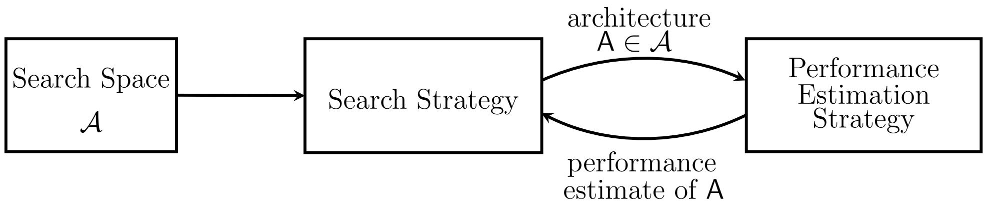
2.2.3 Neural Architecture Search

Although deep learning models are typically developed (trained and validated) on workstations or servers with specialized hardware such as GPUs and Tensor Processing Units (TPUs), they are frequently deployed on mobile devices for portable applications. The computational resources of mobile devices are generally much less than workstations and servers and so large deep learning models, such as those used in human pose estimation, quickly overtax the limited storage, battery power, and processors on mobile devices. As a result, a considerable amount of engineering has revolved around reducing the computational burden of running deep learning models on mobile devices, including the development of model reduction techniques such as network quantization [91] and pruning [201]. While these methods have facilitated mobile deployment, they have not solved the problem completely. Researchers continuously seek to design network architectures that are inherently more efficient, in an effort to push inference on mobile devices towards real-time.

The design principles of efficient deep learning models are centered around minimizing the size of the network and its computational complexity. The former relates to the number of model parameters and the latter to the number of floating point operations (FLOPs), sometimes referred to as multiply-accumulate operations (MACs). The number of parameters and operations may be reduced without significantly hindering accuracy by making judicial modifications to the network architecture; this is the premise of human-principled neural network design. For example, the use of pointwise convolutions and the advent of depthwise separable convolutions [202, 203] have vastly improved the efficiency of CNNs [204, 205, 206, 207, 182].

In parallel to these human-principled advances, there has been a growing interest in the use of machine-driven methods to design efficient architectures by optimizing accuracy within predefined architectural search spaces. This process is known as neural architecture search (NAS) [208, 209, 210, 211]. NAS methods eliminate human bias and permit the automated exploration of diverse network architectures that often transcend human intuition and provide greater accuracy using less computation. Moreover, networks designed using NAS often have fewer parameters [212], which reduces the need for expensive main memory access on embedded hardware designed with small memory caches [182].

The implementation of NAS algorithms is three-pronged: they require the design of a



Search Space
$\mathcal{A}$
Search Strategy
architecture
$\mathsf{A} \in \mathcal{A}$
Performance Estimation Strategy
performance estimate of A

Figure 2.6: Overview of neural architecture search (adapted from [210]). A search strategy samples an architecture A from search space $\mathcal{A}$. The architecture is evaluated using a performance estimation strategy, and the search strategy uses that information to sample better architectures.

search space, implementing a search strategy (*i.e.*, an optimization method), and use of a performance estimation strategy [210] (see Figure 2.6). Brief summaries of the literature relating to each of these components are provided in the remainder of this section. Since its inception, most NAS implementations have addressed the automated search of architectures for the task of image recognition and are consequently concerned with CNNs.

Search Spaces

The search space design is a critical component of the NAS methodology because the search space encompasses the set of feasible solutions that may be sampled by the search strategy. Neural architecture search spaces are broadly grouped into two categories: global and cell-based search spaces [210, 211].

Since CNN architectures are a form of directed acyclic graph, a sensible global search space is that of chain-structured neural networks [210, 211]. A chain-structured neural network is encoded as a single-branch comprising a sequence of layers, where layer L_i receives input from layer L_{i-1} and its output serves as the input to layer L_{i+1}. The network may then be parameterized by the number of layers, the type of operation used in each layer (*e.g.*, convolution, pooling, or more advanced variants thereof), and the hyperparameters associated with each operation (*e.g.*, kernel size, number of filters, stride, *etc.*). Each additional layer increases the size of the search space exponentially.

Cell-based search spaces are built on the observation that most effective human-designed CNN architectures include repetitions of fixed convolutional blocks, or *cells*. These cells are connected within a predefined macroarchitectural template to form larger architectures. Besides dramatically reducing the size of the search space, searching for cells has the added benefit of being scalable and generalizable across datasets and tasks, as the cells can be flexibly stacked together to generate networks ranging in size and depth. Zoph *et al.* [213]

popularized the cell-based search space with NASNet, which included *normal* and *reduction* cells. A key insight in their approach was that the cell topologies learned on CIFAR-10 [214] (a small-scale image classification dataset) effectively transferred to ImageNet [23] (a large-scale image classification dataset). State-of-the-art accuracy was achieved on both datasets using a smaller and more computationally efficient network than previous human-designed networks.

Performance Estimation Strategies

To guide the search, a performance value must be assigned to a sampled architecture. A straightforward way of assigning a performance value is to train the network from scratch and then evaluate its accuracy on a held-out validation set. However, training many networks from scratch is extremely computationally intensive and can require thousands of GPU days [208, 213, 215, 216][2]. Instinctively, this has led to a body of research focused on strategies for speeding up performance estimation. Proxy metrics have frequently been used to estimate performance based on lower fidelities of the actual performance. Examples include training for fewer epochs [213], on lower resolution images [217], on a subset of the data [218], or using fewer filters and/or cells [213, 216]. Performance proxies generally underestimate performance, which isn't a problem provided that the ranking of the sampled architectures is consistent with the ranking of their full-scale counterparts. Alternatively, full-scale performance may be estimated in the early stages of training using learning curve extrapolation. Domhan *et al.* [219] used a weighted combination of 11 parametric learning curve models to terminate poorly performing architectures and speed up a hyperparameter optimization. Others have leveraged architectural meta-data and partial learning curves to predict which architectures were the most promising [220]. Taking this idea a step further, surrogate models have been trained to predict performance based on architectural meta-data alone [221, 222]. Recently, state-of-the-art accuracy in image classification was achieved by searching on lower resolution images and then using compound scaling to appropriately scale the depth, width, and resolution of the discovered network in a synergistic manner [223].

Search Strategies

Numerous search strategies have been used, of which reinforcement learning and evolutionary algorithms have been the most prominent [211].

[2]22,400 GPU days for [208], 2,000 GPU days for [213], 2,600 GPU days for [215], and 3,150 GPU days for [216].

Reinforcement learning (RL) [224] is a learning algorithm where an agent interacts with an environment through sequential decision making with the objective of maximizing its future return. The agent learns to improve its decision-making through multiple interactions with the environment. As such, RL is well-suited for NAS [209, 208, 225, 213], where the agent (search algorithm) makes decisions to modify the state (neural architecture) so as to maximize the return (accuracy on the validation set).

A drawback to the RL search strategy is the excessive computational demand; some of the first implementations required thousands of GPU days [208, 213]. To address this issue, Pham *et al.* [226] used an RL-based controller to search for an optimal subgraph within a larger supergraph. The supergraph embodied the entire search space and weights were shared between the sampled architectures, thus reducing the amount of training required. Methods based around this idea are collectively referred to as one-shot architecture search[3]. As an alternative to using an RL-based controller, Liu *et al.* [228] sampled subgraphs in a differentiable manner using learnable parameters that weighted the sampled edges.

A popular alternative to reinforcement learning is *neuroevolution*, a form of neural architecture search that harnesses evolutionary algorithms [215, 218, 229, 230, 95, 231, 216]. Evolutionary algorithms represent a broad class of global optimization algorithms that evolve a population of individuals using biologically inspired mechanisms, namely selection, recombination, and mutation. After the population is initialized, an optimization step (*i.e.*, one generation) proceeds as follows: parents are selected for reproduction, offspring are created by applying recombination and/or mutation operators, and the fitness of each offspring is evaluated. The selection algorithm favors individuals with higher fitness; the recombination algorithm mixes the genes of the parents to create superior offspring; and the mutation algorithm adds diversity to the gene pool. In turn, the fitness is maximized over several generations. In the context of NAS, the population is a set of neural architectures. Often in practice, only mutation is used [211]. Examples of mutations include adding or removing a layer or skip connections [95], or altering a layer type or its hyperparameters [218]. The fitness is typically an estimate of the accuracy on the validation set. The underlying principles of all neuroevolution methods are more or less the same. Where the differences lie is in the way the neural architecture is encoded, and choices regarding the population initialization and size, selection algorithms, and mutations.

To speed up the search process, evolutionary algorithms often build on the principle of network morphisms [94] to transfer the weights of the parent networks to the mutated children. In essence, network morphisms mutate the network architecture to add new operations such that the network output remains unchanged (*i.e.*, network morphisms

[3]Not to be confused with one-shot learning [227].

are "function-preserving"). Due to the function-preserving constraint, network morphisms require intricate implementations, but ultimately enable the child networks to be trained for just a few epochs by "fine-tuning" the network weights to accommodate the mutation. Elsken *et al.* [230] and Wistuba [95] demonstrated the effectiveness of network morphisms and weight transfer by generating state-of-the-art CNNs on CIFAR-10 in just 1 and 0.5 GPU days, respectively.

The important distinctions between neuroevolution methods that leverage weight *transfer* and one-shot approaches that leverage weight *sharing* [226, 228] are briefly discussed. In one-shot methods that use weight sharing, the search is performed over a single training run of the supergraph. Subgraphs are selected, evaluated using the supergraph weights, and then ranked. The best performing subgraph is finally trained from scratch. One-shot methods are based around the hypothesis that the ranking of the candidate subgraphs correlates with their true ranking following final training. However, Yu *et al.* [232] observed the correlation to be weak, and ultimately found that ENAS [226] and DARTS [228] perform no better than random search. Moreover, some one-shot methods require the entire supergraph to be kept in memory, which limits the size of the search space. These issues are not a concern in neuroevolution, where the candidate architectures are trained separately and thus do not share weights.

Due to a lack of fair comparisons, it remains unclear as to which search strategy is best. Many of the available experiments differ significantly in terms of search space, search durations, and training methodologies. In one instance, Real *et al.* [216] conducted a controlled comparison of evolution and reinforcement learning approaches and found that evolution can obtain better results faster with the same hardware, especially at the earlier stages of the search. This finding is especially relevant when fewer computational resources are available. Moreover, in a recent benchmarking of NAS algorithms, neuroevolution methods were among the top performing algorithms and consistently outperformed reinforcement learning and random search [233].

2.3 Discussion

The abundant use of heatmaps in human pose estimation is noteworthy, if not staggering. Their early adoption and observed utility may have induced a methodological bias to which subsequent researchers, perhaps unknowingly, have been subjected. Regardless of the apparent heatmap prowess, alternative methods for modeling keypoint locations should at least be considered and explored. Supporting this notion, recent research has elucidated some of the limitations of heatmaps, including the inherent issue of quantization error and the excessive computation required to generate and post-process large heatmap fields. In a recent heatmap-free approach to human pose estimation, Li *et al.* [97] proposed to disentangle the horizontal and vertical keypoint coordinates such that each coordinate was represented using a one-hot encoded vector. This representation saved computation and permitted an expansion of the output resolution, thereby reducing the effects of quantization error and eliminating the need for refinement post-processing. In another recent work, researchers introduced the residual log-likelihood (RLE), a novel loss function for direct keypoint regression based on normalizing flows [234], and matched the accuracy of state-of-the-art heatmap-based methods [235]. Direct keypoint regression has also been attempted using transformers [236].

The similarities between keypoint detection and object detection has led to alternative heatmap-free keypoint detection approaches that extend object detectors [47, 165, 166]. Outside the realm of human pose estimation, Xu *et al.* [117] regressed anchor templates of facial keypoints and aggregated them to achieve state-of-the-art accuracy in facial alignment. In sports analytics, McNally *et al.* [96] encountered the issue of overlapping heatmap signals in the development of an automatic scoring system for darts, and instead chose to model keypoints as objects using small square bounding boxes. Inspirations from object detection may prove to be useful in human pose estimation as well.

Despite continuous efforts devoted to hand-designing human pose estimation networks that provide superior performance and computational efficiency, to date there are but a few peer-reviewed works [237, 238] (and some non-peer-reviewed [239, 240]) that use NAS in the design of human pose estimation networks. Notably, none have managed to improve upon the state-of-the-art in terms of accuracy. This presents a compelling research opportunity. Considering the wide range of potential applications for human pose estimation and motion capture on smartphones, the dearth of NAS research in human pose estimation is even more compelling. The ability of NAS to generate efficient networks yielding state-of-the-art accuracy has been demonstrated repeatedly for the task of image classification [212, 223]. Several engineering advancements, such as the use of one-shot models and network morphisms, have made NAS possible on single-GPU workstations by reducing the search time to less than a single GPU day [210, 95, 226]. With regards to network morphisms, an opportunity is presented to develop a less restrictive weight transfer scheme that is not constrained by complete function-preservation.

The computational demands of NAS will continue to diminish as the field matures, which will inevitably lead to increased use in more complex visual perception tasks, including human pose estimation. As an illustration, NAS was recently used to design state-of-the-art semantic segmentation and object detection models [241, 185]. The extension of NAS to human pose estimation is thus very encouraging and highly anticipated.

Chapter 3

EvoPose2D: A Two-Stage Human Pose Estimation Network Designed using Neuroevolution Accelerated with Weight Transfer

Neural architecture search has proven to be highly effective in the design of efficient convolutional neural networks that are better suited for mobile deployment than hand-designed networks. Hypothesizing that neural architecture search holds great potential for human pose estimation, this chapter explores the application of neuroevolution, a form of neural architecture search inspired by biological evolution, to the design of a two-stage human pose estimation model. Additionally, a flexible weight transfer scheme is proposed to accelerate the neuroevolution. In experiments, the neuroevolution produces a network design that is more efficient and more accurate than previous hand-designed human pose estimation networks. In fact, the generated networks process images at higher resolutions using less computation than previous hand-designed networks at lower resolutions, providing an opportunity to push the boundaries of human pose estimation. The base network designed via neuroevolution, referred to as EvoPose2D-S, achieves comparable accuracy to SimpleBaseline [51] while being 50% faster and 12.7x smaller in terms of file size. When scaled appropriately, EvoPose2D-L achieves higher AP than HRNet-W48 [52] on the Microsoft COCO Keypoints benchmark while being 4.3x smaller and having comparable inference speed. The source code is publicly available at https://github.com/wmcnally/evopose2d.

3.1 Introduction

As reviewed in Chapter 2, the use of deep learning [20], and specifically deep CNNs [21], has been prevalent in human pose estimation [40, 41, 43, 45, 50, 51, 52]. The most accurate human pose estimation methods use a two-stage, top-down pipeline, where an off-the-shelf person detector is first used to detect human instances in an image, and a heatmap-based single-person human pose estimation network is run over the detections to obtain keypoint predictions for each person [50, 51, 52]. The research presented in this chapter focuses on the latter stage of this two-stage pipeline; however, the presented methodology is applicable to the design of single-stage human pose estimation methods (*e.g.*, [45, 54, 58]) as well. Specifically, preprocessed person detections produced by a Faster R-CNN object detector [157] are used in the first stage, and a new heatmap-based single-person human pose estimator is developed for the second stage. At the time of development, HRNet [52], a CNN that makes extensive use of multi-branch and multi-scale feature mixing, represented the state of the art in two-stage human pose estimation, providing the highest accuracy on the Microsoft COCO Keypoints dataset.

The study presented in this chapter explores the application of neuroevolution [242], a form of neural architecture search inspired by biological evolution, to human pose estimation for the first time. Despite the widespread success of neural architecture search in many areas of computer vision [223, 243, 241, 185, 244, 245, 246], the design of human pose estimation models has primarily been led by human designers as opposed to machine-driven methods. In a few cases, other neural architecture search methods such as reinforcement learning [240, 237] and differentiable architecture search [239, 238] have been leveraged in the design of human pose estimation networks, albeit with limited success. While these machine-designed networks exhibited superior computational efficiency as a result of having fewer parameters and mathematical operations, none managed to surpass the best performing hand-designed networks in terms of accuracy. Furthermore, independent studies have identified neuroevolution as one of the most effective search strategies, surpassing reinforcement learning and random search in its ability to find better architectures using less computation [233, 216].

Network morphisms [94] and function-preserving mutations [95] are techniques that reduce the computational cost of neuroevolution. In essence, these methods iteratively mutate networks and perform weight transfer in such a way that the function of the network is completely preserved upon mutation. In other words, the output of the mutated network is identical to that of the parent network. Ergo, the mutated child networks need only be trained for a relatively small number of steps compared to when training from a randomly initialized state. As a result, these techniques are capable of reducing the search time to within a practical time frame – a matter of GPU days. However, function-preserving mutations can be challenging to implement and also restricting (*e.g.*, the complexity of the network cannot be reduced [95]). Motivated to address these drawbacks, a new flexible weight transfer scheme is proposed that (1) is less restrictive, (2) has a simple implementation, and (3) is effective in accelerating neuroevolution. The new weight transfer scheme is exploited in conjunction with large-batch training on high-bandwidth TPUs to run fast neuroevolutions within a search space tailored towards human pose estimation. The neuroevolution produces a base human pose network that has a relatively simple design, provides state-of-the-art accuracy when scaled, and uses fewer floating-point operations and parameters than the best performing networks in the literature (see Figure 3.1). The research contributions of this study are summarized below:

- A new weight transfer scheme is proposed to accelerate a neuroevolution, and neuroevolution is used in the design of a human pose estimation model for the first time. In contrast to previous neuroevolution methods that exploit weight transfer, the proposed weight transfer method is not constrained by complete function preservation [95, 94]. Despite relaxing this constraint, supporting experiments indicate that the level of functional preservation afforded by the proposed weight transfer scheme is sufficient to provide fitness convergence, thereby simplifying neuroevolution and making it more flexible.

- Empirical evidence is presented to support the use of large-batch training (*i.e.*, batch size of 2048) in conjunction with the Adam optimizer [25] to accelerate the training of human pose networks with no loss in accuracy. The benefits of large-batch training are reaped in subsequent neuroevolution experiments by maximizing training throughput on high-bandwidth TPUs.

- A search space conducive to human pose estimation is proposed and the above contributions are leveraged to run a fast (∼1 day using eight v2-8 TPUs) full-scale neuroevolution of a population of human pose networks. A computationally efficient human pose estimation model is produced that achieves state-of-the-art accuracy on the most widely used benchmark dataset when scaled appropriately.

As a final preface to this study, it is noted that it is often difficult to make fair comparisons with previous human pose estimation methods due to discrepancies in training algorithms and the use of model-agnostic enhancements. Examples include the use of different learning rate schedules [52, 53], more data augmentation [53, 247], loss functions that target more challenging keypoints [50], specialized post-processing steps [248, 249], or

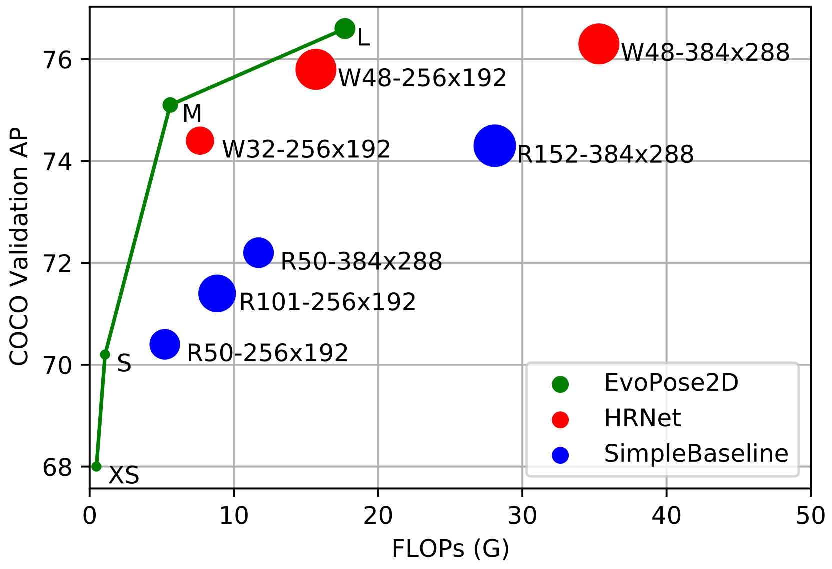


Figure 3.1: Accuracy vs. FLOPs: A comparison of the accuracy, size, and computational cost of EvoPose2D, SimpleBaseline [51], and HRNet[52] at different scales. The circle size is proportional to the network file size. EvoPose2D-S provides comparable accuracy to SimpleBaseline (ResNet-50), is 12.7x smaller, and uses 4.9x fewer FLOPs. At full-scale, EvoPose2D-L obtains state-of-the-art accuracy using 2.0x fewer FLOPs and is 4.3x smaller compared to HRNet-W48. In contrast to SimpleBaseline and HRNet, EvoPose2D does not make use of model-agnostic enhancements such as ImageNet pretraining and half-body augmentation [52].

more accurate person detectors [53, 249]. These discrepancies in training algorithms can potentially account for reported differences in accuracy. To fairly compare with state-of-the-art methods, SimpleBaseline [51] and HRNet [52] were re-implemented in this work and all networks were trained under the same settings using the same software and hardware.

3.2 Neuroevolution Acceleration via Weight Transfer

Suppose that a pretrained *parent* neural network is represented by the function $\mathcal{P}\left(\mathbf{x};\theta^{(\mathcal{P})}\right)$, where $\mathbf{x}$ is the input to the network and $\theta^{(\mathcal{P})}$ are its learned parameters. The foundation of the proposed neuroevolution framework lies in the process by which the unknown parameters $\theta^{(\mathcal{C})}$ in a mutated child network $\mathcal{C}$ are inherited from $\theta^{(\mathcal{P})}$ such that $\mathcal{C}\left(\mathbf{x};\theta^{(\mathcal{C})}\right) \approx \mathcal{P}\left(\mathbf{x};\theta^{(\mathcal{P})}\right)$. That is, the output, or *function* of the mutated child network is similar to the parent, but not necessarily equal. To enable fast neural architecture search, the degree to which the parent's function is preserved must be sufficient to allow $\theta^{(\mathcal{C})}$ to be trained to convergence in a small fraction of the number of steps normally required when training from a randomly initialized state.

To formalize the proposed weight transfer in the context of 2D convolution, let $W^{(l)} \in \mathbb{R}^{k_{p1} \times k_{p2} \times i_p \times o_p}$ denote the weights used by layer l of the parent network, and $V^{(l)} \in \mathbb{R}^{k_{c1} \times k_{c2} \times i_c \times o_c}$ the weights of the corresponding layer in the mutated child network, where k is the kernel size, i is the number of input channels, and o is the number of output channels. For the sake of brevity, consider the special case when $k_{p1} = k_{p2} = k_p$, $k_{c1} = k_{c2} = k_c$, and $o_p = o_c$, but note that the following definition is easily extended to when $k_{p1} \neq k_{p2}$, $k_{c1} \neq k_{c2}$, or $o_p \neq o_c$. The inherited weights V_W are given by:

$$V_W^{(l)} = \begin{cases} W^{(l)}_{p:p+k_c,\, p:p+k_c,\, :i_c,\, :} & (i_c < i_p) \wedge (k_c < k_p) \\ W^{(l)}_{p:p+k_c,\, p:p+k_c,\, :,\, :} & (i_c \geq i_p) \wedge (k_c < k_p) \\ W^{(l)}_{:,\, :,\, :i_c,\, :} & (i_c < i_p) \wedge (k_c \geq k_p) \\ W^{(l)} & (i_c \geq i_p) \wedge (k_c \geq k_p) \end{cases} \tag{3.1}$$

where $p = \frac{1}{2}(k_p - k_c)$. $V_W^{(l)}$ is transferred to $V^{(l)}$ and the remaining non-inherited weights in $V^{(l)}$ are randomly initialized. An illustration of the weight transfer between two convolutional layers is shown in Figure 3.2. In principle, the proposed weight transfer can be used with convolutions of any dimensionality (*e.g.*, 1D, 2D, or 3D convolutions), and is permitted between convolutional operators with different kernel size, stride, dilation, input channels, and output channels. Thus, it is highly flexible. More generally, it can be applied to any operations with learnable parameters, including batch normalization [250] and fully-connected layers.

In essence, the proposed weight transfer method relaxes the function-preservation constraint imposed in [94, 95]. In practice, it is observed that the proposed weight transfer preserves the majority of the function of deep CNNs following mutation. This permits the application of network mutations in a simple and flexible manner while maintaining good parameter initialization in the mutated network. As a result, the mutated networks can be trained using fewer iterations, thereby accelerating the neuroevolution.

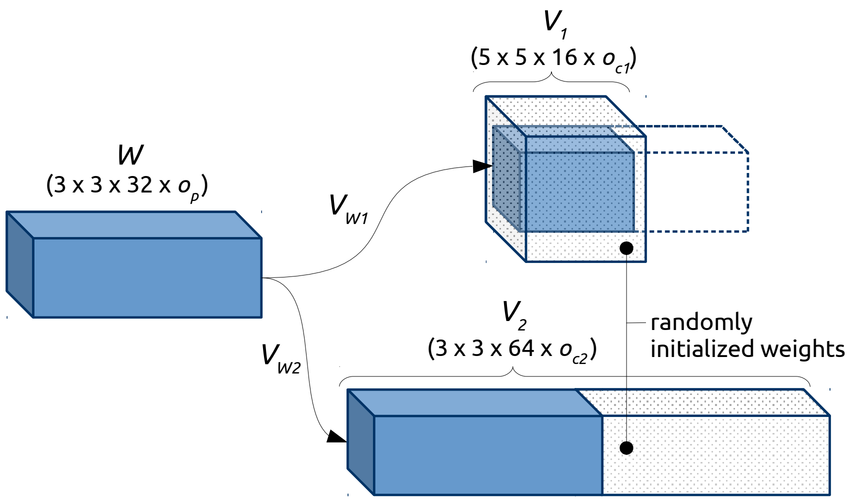


Figure 3.2: Two examples ($W \rightarrow V_1$, $W \rightarrow V_2$) of the weight transfer used in the proposed neuroevolution framework. The output channels of the convolutional filters or omitted for clarity. The trained weights (shaded in blue) in the parent convolutional filter W are transferred, either in part (V_{W1}) or in full (V_{W2}), to the corresponding filters (V_1, V_2) in two separate mutated child networks following Eq. 3.1. The weight transfer extends to all output channels in the same manner as depicted for the input channels.

3.3 Neuroevolution Design

This section includes the engineering details for the neuroevolution implementation that uses weight transfer scheme introduced in the previous section to accelerate the evolution of a population of human pose networks. While this study focuses on the application of human pose estimation, it is noted that the neuroevolution approach is generally applicable to all types of deep networks.

3.3.1 Search Space

Neural architecture search helps moderate human involvement in the design of deep neural networks. However, neural architecture search is by no means fully automatic. To some extent, our role transitions from a network designer to that of a search designer. Decisions regarding the search space are particularly important because the search space encompasses all possible solutions to the optimization problem, and its size correlates with the amount of computation and time required to run an exhaustive search. As such, it is common to

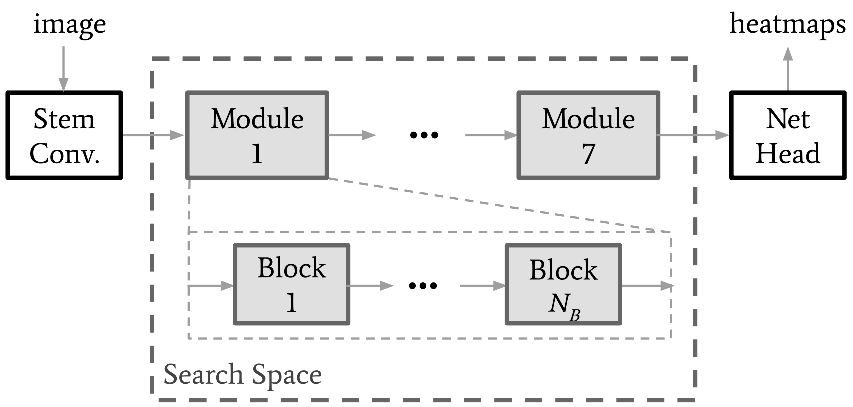


Figure 3.3: Neuroevolution search space diagram.

exploit prior knowledge to reduce the size of the search space and ensure that the sampled architectures are tailored towards the task at hand [213].

Motivated by the simplicity and elegance of the SimpleBaseline architecture [51], an optimal human pose estimation backbone is evolved using a search space inspired by [212, 223]. Specifically, the search space encompasses a single-branch hierarchical structure that includes seven modules stacked in series. Each module is constructed of chain-linked inverted residual blocks [182] that use an expansion ratio of six and squeeze-excitation [251]. The neuroevolution seeks to find the optimal kernel size, the number of inverted residual blocks, and the number of output channels for each module. Considering the newfound importance of spatial resolution in the deeper layers of human pose networks [52], the stride of the last three modules are also included in the search space. To complete the network, an initial convolutional layer with 32 output channels precedes the seven modules, and three transpose convolutions with kernel size of 3x3, stride of 2, and 128 output channels are used to construct the network head. Each transpose convolution doubles the spatial resolution of the feature maps. For example, if the output of Module 7 had a spatial resolution of 8×6, then the size of the output heatmaps would be 64×48. A diagram of the search space is provided in Figure 3.3.

Table 3.1 shows the module configuration for the common ancestor network used in the neuroevolution experiments. The kernel size options used were 3x3 and 5x5. The maximum number of blocks was set to four. The maximum number of output channels were set to the values in the common ancestor network (given in rightmost column of Table 3.1).

Each module was encoded into an integer quartet, representing its number of blocks, kernel size, number of output channels (as a multiple of 8), and stride, respectively. Ar-

Component	Blocks N_B	Kernel Size	Stride	Output Shape
Module 1	1	3	1	$(\frac{h}{2}, \frac{w}{2}, 16)$
Module 2	2	3	2	$(\frac{h}{4}, \frac{w}{4}, 24)$
Module 3	2	5	2	$(\frac{h}{8}, \frac{w}{8}, 40)$
Module 4	3	3	2	$(\frac{h}{16}, \frac{w}{16}, 80)$
Module 5	3	5	1	$(\frac{h}{16}, \frac{w}{16}, 112)$
Module 6	4	5	2	$(\frac{h}{32}, \frac{w}{32}, 192)$
Module 7	1	3	1	$(\frac{h}{32}, \frac{w}{32}, 320)$

Table 3.1: Module configuration for the common ancestor network.

chitectures were sampled using a 7×4 integer array, referred to as the genotype. The mutations used included increasing/decreasing the number of blocks by 1, changing the kernel size, increasing/decreasing the stride by 1, and increasing/decreasing the number of output channels by 8. During the neuroevolution, the genotypes are cached to ensure that no genotype was sampled twice. The mutation function is provided in Algorithm 1. The search space can produce 10^{14} unique architectures.

3.3.2 Fitness

To strike a balance between computational efficiency and accuracy, the multi-objective optimization minimizes a fitness function including the validation loss and the number of network parameters. Given a human pose network represented by the function $\mathcal{N}\left(\mathbf{x}\,;\,\theta^{(\mathcal{N})}\right)$, the loss $\mathcal{L}_i$ for a single RGB input image $\mathbf{I} \in \mathbb{R}^{h\times w\times 3}$ and corresponding target heatmap $\mathbf{S} \in \mathbb{R}^{h'\times w'\times K}$ is given by:

$$\mathcal{L}_i(\mathcal{N}, \mathbf{I}) = \frac{1}{K}\sum_{k=1}^{K} \delta(\nu_k > 0) \left\| \mathcal{N}\left(\mathbf{I} \,|\, \theta^{(\mathcal{N})}\right)_k - \mathbf{S}_k \right\|_2^2 \tag{3.2}$$

where K is the number of keypoints and ν represents the keypoint visibility flag[1]. The target heatmaps $\mathbf{S}$ are generated by centering 2D Gaussians with a standard deviation of $\frac{h'}{32}$ pixels on the ground-truth keypoint coordinates and normalizing to a maximum intensity

[1]More details available at https://cocodataset.org/#keypoints-eval.

Algorithm 1: Neuroevolution mutation

```
Input: parent genotype $g_p$, ancestor genotype $g_a$, genotype cache $G$
Output: mutated child genotype $g_c$
$g_c \leftarrow g_p$
while $g_c \in G$ or $g_c = g_p$ do
    $g_c \leftarrow g_p$
    $i, j \leftarrow$ randint(7), randint(4)
    if $j = 0$ then
        if $g_c[i, j] = 1$ then
            $g_c[i, j]$ += 1 // increase the number of blocks
        else if $g_c[i, j] = 4$ then
            $g_c[i, j]$ −= 1 // decrease the number of blocks
        else if randint(2) $> 0$ then
            $g_c[i, j]$ += 1 // increase the number of blocks
        else
            $g_c[i, j]$ −= 1 // decrease the number of blocks
    else if $j = 1$ then
        $g_c[i, j] \leftarrow \{3, 5\}$[randint(2)] // mutate the kernel size
    else if $j = 2$ then
        $g_c[i, j] \leftarrow$ randint($g_a[i, j] + 1$) // mutate the number of output channels
    else if $j = 3$ and $i \geq 4$ then
        if $g_c[i, j] = 2$ and sum($g_p[:, j] - 1$) $= 4$ then
            $g_c[i, j]$ −= 1 // decrease the stride
        else if $g_c[i, j] = 1$ and sum($g_p[:, j] - 1$) $< 4$ then
            $g_c[i, j]$ += 1 // increase the stride
$G$.append($g_c$)
```

of 255 (see Eq. 1.1). The network loss is computed as:

$$\mathcal{L}(\mathcal{N}) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_i(\mathcal{N}, \mathbf{I}_i) \tag{3.3}$$

where N is the number of image samples in the dataset. Finally, the fitness $\mathcal{J}$ of a network $\mathcal{N}$ is computed on the validation dataset by:

$$\mathcal{J}(\mathcal{N}) = \left(\frac{T}{n(\theta^{(\mathcal{N})})}\right)^{\Gamma} \mathcal{L}(\mathcal{N}) \tag{3.4}$$

where $n(\theta^{(\mathcal{N})})$ is the number of network parameters, T is the target number of parameters, and Γ controls the fitness trade-off between the number of parameters and the validation loss. Minimizing the number of parameters instead of the number of FLOPs permits indirectly minimizing the FLOPs while not penalizing mutations that decrease the stride of the network.

3.3.3 Evolutionary Strategy

The proposed evolutionary strategy proceeds as follows. In "generation 0", the genotype of the common ancestor network is manually defined and the decoded ancestor network is trained from scratch for e_0 epochs. In generation 1, λ child genotypes are generated by mutating the ancestral genotype according to Algorithm 1. The child network architectures are decoded from the mutated child genotypes and all weights in the child networks are randomly initialized. Then, the weight transfer scheme described in Section 3.2 is used to transfer the trained weights from the ancestor to the children. For batch normalization layers, the non-transferred weights are initialized with the means of the parent. When a new block is added as a result of a mutation, the weights from the parent module's last block are copied to the new block in the child. The child networks are then trained for e epochs ($e \ll e_0$). At the end of generation 1, the μ networks with the best fitness from the pool of $(\lambda + 1)$ networks (children + ancestor) become the parents in the next generation. In generation 2 and beyond, the mutation $\rightarrow$ weight transfer $\rightarrow$ training process is repeated and the top-μ networks from the pool of $(\lambda + \mu)$ networks (children + parents) become the parents in the next generation. The evolution continues until manual termination, typically after the fitness has converged.

3.3.4 Large-batch Training

Even with the computational savings afforded by weight transfer, running a full-scale neuroevolution of human pose networks using a standard input size of 256x192 would not be feasible within a practical time-frame using common GPU resources (*e.g.*, an 8-GPU server). To reduce the search time to within a practical range, large batch sizes were exploited when training the human pose networks on TPUs.

It has been shown that training using large batch sizes with stochastic gradient descent causes a degradation in the quality of the model as measured by its ability to generalize to unseen data [252, 253]. The difference in accuracy on training and test sets, sometimes referred to as the generalization gap, can drop by as much as 5% (absolute change in accuracy) as a result of using large batch sizes (> 256) [253]. Recently, Goyal *et al.* [254] implemented measures for mitigating the training difficulties caused by large batch sizes, including linear scaling of the learning rate, and an initial warm-up period where the learning rate was gradually increased. Deep learning methods are often data-dependent and task-dependent, so it is unclear whether the training measures imposed by Goyal *et al.* for image classification apply in the general case. It is also unclear whether the learning rate modifications are applicable to optimizers that use adaptive learning rates. Adam [25] is an example of such an optimizer that is widely used in human pose estimation.

Maximizing training efficiency using large-batch training is critical when the computational demand of training is very high, such as in neural architecture search. Therefore, in an effort to maximize training throughput on TPUs, this study investigates the training behaviour of human pose networks using large batch sizes and the Adam optimizer. Section 3.4.1 provides supporting experiments, where the learning rate was linearly scaled with the batch size and gradually ramped-up during the first few epochs, following [254]. The experimental results demonstrate that the training regimen can be used to train human pose networks up to a batch size of 2048 with no loss in accuracy. The largest batch size previously used to train a human pose network was 256, which required 8 GPUs [53]. Large batch sizes are then exploited in the subsequent neuroevolution experiments.

3.3.5 Compound Scaling

It has recently been shown that scaling a network's input resolution, width (*i.e.*, the number of ouput channels), and depth (*i.e.*, the number of layers) together provides greater benefits than scaling either of these dimensions individually [223]. Motivated by this finding, the base network designed using neuroevolution is scaled to different input resolutions using the following depth (c_d) and width (c_w) coefficients:

$$c_d = \alpha^{\phi} \quad c_w = \beta^{\phi} \quad \phi = \frac{\log r - \log r_s}{\log \gamma} \tag{3.5}$$

where r_s is the search resolution, r is desired resolution, and α, β, γ are scaling parameters. The scaling parameters presented in [223] ($\alpha = 1.2$, $\beta = 1.1$, $\gamma = 1.15$) are used for convenience, but it is hypothesized that better results could be obtained if these parameters were tuned.

3.4 Experiments

This section first presents the results from large-batch training experiments (Section 3.4.1) that examine the training behaviour of human pose networks using large batch sizes and the Adam optimizer on TPU hardware. These preliminary experiments support the subsequent neuroevolution experiments (Section 3.4.2), which use a large batch size to reduce the search time.

3.4.1 Large-batch Training of Human Pose Networks on TPUs

For the large-batch training experiments, the SimpleBaseline model of Xiao *et al.* [51] was re-implemented and trained on the Microsoft COCO Keypoints dataset. The SimpleBaseline network stacks three transpose convolutions with 256 channels each and kernel size of 3x3 on top of a ResNet-50 backbone, which was pretrained on ImageNet [23]. The experiments were run at an input resolution of 256×192, yielding output heatmap predictions of size 64×48. According to the TensorFlow profiler used, the model has 34.1M parameters and 5.21G FLOPs.

Implementation Details

The following experimental setup was used to obtain the results for all models trained on COCO in this chapter. Additional implementation details for neuroevolution and PoseTrack training are provided in the next section.

TensorFlow 2.3 and the `tf.keras` API were used for implementation. The COCO Keypoints dataset was first converted to TFRecords for TPU compatibility (1024 examples per shard). The TFRecord dataset contained the serialized examples including the raw images, target keypoint locations, and the person bounding boxes. The labeled bounding boxes were used during training (including validation loss computation), and the preprocessed bounding box detections were used during testing. More details on the person detections are provided under the **Testing** heading. The dataset was stored in a Google Cloud Storage Bucket where it was accessed remotely by the TPU host CPU over the cloud network. Thus, all preprocessing (*i.e.*, including target heatmap generation, image transformations, and data augmentation) was performed on the TPU's host CPU. A single-device v3-8 TPU (8 TPU cores, 16GB of high-bandwidth memory per core) was used for training, validation, and testing.

Preprocessing. The RGB input images were first normalized to a range of [0, 1] then centered and scaled by the ImageNet pixel means and standard deviations. The images were then transformed and cropped to the input size of the network using the bounding boxes. During training, random horizontal flipping, scaling, and rotation were used for data augmentation. The exact data augmentation is provided in the source code.

Training. The networks were trained for 200 epochs using `bfloat16` floating-point format, which consumes half the memory compared to the commonly used float32 data type. The loss represented in Eq. (3.3) was minimized using the Adam optimizer [25] with a cosine-decay learning rate schedule [255] and L2 regularization with 1e$-$5 weight decay. The base learning rate l_r was set to 2.5e$-$4 and was scaled to $\frac{n}{32}l_r$, where n is the global batch size. Additionally, a warm-up period was implemented by linearly increasing the learning rate from l_r to $\frac{n}{32}l_r$ over the first five epochs. The validation loss was computed after every epoch using the ground-truth bounding boxes.

Testing. The common two-stage, top-down pipeline was used during testing [50, 51, 52]. The person detections were generated using Faster R-CNN [157] and were the same as those used in [51, 52]. The detected bounding boxes provide an AP of 56.4 on COCO `val2017` and 60.9 on COCO `test-dev` [51] for the person object category. The standard testing protocol was followed: the predicted heatmaps from the original and horizontally flipped images were averaged and the keypoint predictions were obtained after applying a quarter offset in the direction from the highest response to the second highest response.

Large-batch Training Results

The batch size was doubled from an initial size of 256 until the memory of the v3-8 TPU was exceeded. The maximum batch size attained was 2048. The loss curves for the corresponding training runs are shown in Figure 3.4. While the final training loss increased marginally with the batch size, the validation losses converged in the later stages of training, signifying that the networks provide similar accuracy. The AP values provided in Table 3.2 confirm that the networks can be trained using batch sizes up to 2048 with no

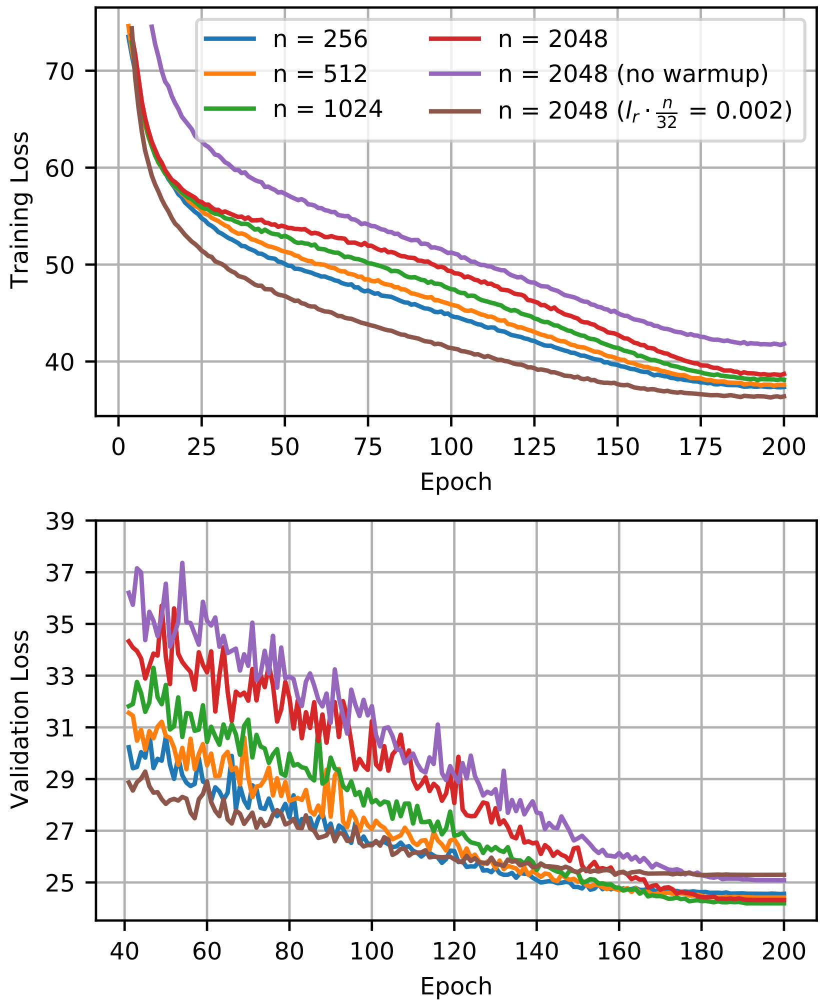


Figure 3.4: Training (top) and validation (bottom) losses during training of SimpleBaseline (ResNet-50) [51] on a v3-8 Cloud TPU using various large batch sizes and learning rate schedules.

loss in accuracy. It is hypothesized that the increase of 0.6 AP over the original SimpleBaseline implementation (AP of 70.4) was due to training for longer (200 epochs versus 140). Additionally, the importance of warm-up and learning rate scaling is highlighted in the bottom section of Table 3.2; when training at the maximum batch size, removing warm-up resulted in a loss of 1.3 AP, and removing learning rate scaling resulted in a loss of 0.7 AP.

While preprocessing the data online using the TPU host CPU provides flexibility for

Batch size (n)	Warm-up	Scale l_r	Training Time (hrs)	AP
256	Y	Y	7.20	71.0
512	Y	Y	5.42	71.0
1024	Y	Y	5.25	71.2
2048	Y	Y	5.32	71.0
2048	N	Y	5.35	69.7
2048	Y	N	5.33	70.3

Table 3.2: Training times and final AP for large-batch training of SimpleBaseline on Cloud TPU. The original implementation reports an AP of 70.4 [51]. The bottom two rows highlight the importance of warm-up and scaling the learning rate when using large batch sizes.

training using different input sizes and data augmentation, it ultimately caused a bottleneck in the input pipeline. This is evidenced by the training times in Table 3.2, which decreased after increasing the batch size to 512, but leveled-off at around 5.3 hours using batch sizes of 512 or greater. The training time could be reduced substantially if preprocessing and augmentation were included in the TFRecord dataset, or if the TPU host CPU had greater processing capabilities. It is also noted that training these models for 140 epochs instead of 200, as in the original implementation [51], reduces the training time to 3.7 hours. Bypassing validation after every epoch reduces the training time further. For comparison, training a model of similar size on eight NVIDIA TITAN Xp GPUs takes approximately 1.5 days [50].

3.4.2 Neuroevolution Experiments

The neuroevolution described in Section 3.3 was run under various settings on an 8-CPU, 40 GB memory virtual machine that called on eight v2-8 Cloud TPUs to train several generations of a human pose network population. COCO `train2017` and `val2017` were used for network training and fitness evaluation, respectively. The input resolution used was 256x192, and the target number of parameters T was set to 5M. Other settings, including Γ, λ, and μ, are provided in the legend of Figure 3.5 (top). ImageNet pretraining was exploited in the neuroevolution by seeding the common ancestor network using the same inverted residual blocks as used in EfficientNet-B0 [223] (module configuration provided in Table 3.1). The ancestor network was trained for 30 epochs, and the proposed weight transfer scheme was used to quickly fine-tune the mutated child networks over just

5 epochs. A batch size of 512 was used to provide near-optimal training efficiency (as per the results in the previous section) and prevent memory exhaustion mid-search (limited to v2-8 TPU nodes that have half the memory compared to v3-8 nodes). No learning rate warm-up was used during neuroevolution, and the only data augmentation used was horizontal flipping. All other training details are the same as in Section 3.4.1.

Figure 3.5 (top) shows the convergence of fitness for three independent neuroevolutions E1, E2, and E3, which had runtimes of 1.5, 0.8 and 1.1 days, respectively. The gap between the fitness (solid line) and validation loss (dashed line) was larger in E2 and E3 compared to E1, indicating that smaller networks were favored more as a result of decreasing Γ. After increasing the number of children from 32 in E2 to 64 in E3, it became apparent that using fewer children may provide faster convergence, but may also cause the fitness to converge prematurely to a local minimum. Figure 3.5 (bottom) plots the validation loss against the number of parameters for all the sampled genotypes. The prominent Pareto frontier near the bottom-left of the figure provides confidence that the search space was thoroughly explored.

To explicitly demonstrate the benefit of the proposed weight transfer scheme, E3 was run without weight transfer using the same training schedule. As shown in Figure 3.5 (top), the fitness did not decrease below that of the ancestor network. It stands that the child networks would need to be trained at least as long as the ancestor network (30 epochs in this case) to achieve the same level of convergence without using the proposed weight transfer scheme. As a result, the neuroevolution runtime would increase six-fold.

The network with the lowest fitness from neuroevolution E3 was selected as the baseline network, and is referred to as EvoPose2D-S. Its architectural details are provided in Table 3.3. Notably, the overall stride of EvoPose2D-S is less than what is typically seen in hand-designed human pose networks. The lowest spatial resolution observed in the network is $\frac{1}{16}$ the input size, compared to $\frac{1}{32}$ in SimpleBaseline [51] and HRNet [52]. The output heatmaps are twice as large as a result. Another observation is that the optimal number of output channels in the first five modules were found to be equal to the upper bounds, so it is possible that a better architecture could be obtained if these limits were increased.

The baseline network EvoPose2D-S was scaled to various levels of computational expense. A lighter version (EvoPose2D-XS) was created by increasing the stride in Module 6, which halved the number of FLOPs. Using the compound scaling method described in Section 3.3, EvoPose2D-S was scaled to an input resolution of 384x288 (EvoPose2D-M), which is the highest input resolution used in top-down human pose estimation. In an effort to push the boundaries of human pose estimation, the input resolution was scaled to 512x384 with EvoPose2D-L. Even at this high spatial resolution, EvoPose2D-L has roughly

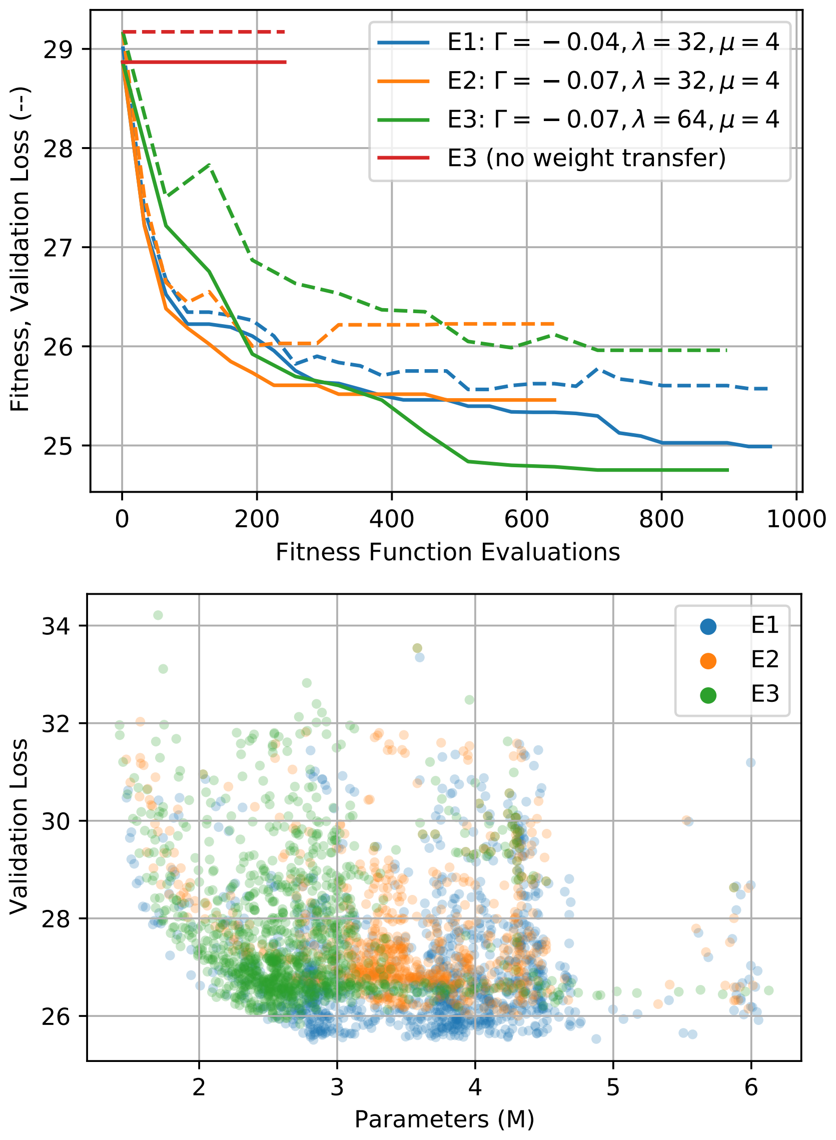


Figure 3.5: Top: Tracking the network with the best fitness in three independent neuroevolutions. The dashed line represents the validation loss of the network with the lowest fitness. Γ: fitness coefficient controlling trade-off between validation loss and number of parameters. λ: number of children. μ: number of parents. Bottom: Validation loss versus the number of network parameters for all sampled networks.

Component	Blocks	Kernel Size	Stride	Output Shape
Stem Conv	-	3	2	$(\frac{h}{2}, \frac{w}{2}, 32)$
Module 1	1	3	1	$(\frac{h}{2}, \frac{w}{2}, 16)$
Module 2	3	3	2	$(\frac{h}{4}, \frac{w}{4}, 24)$
Module 3	2	5	2	$(\frac{h}{8}, \frac{w}{8}, 40)$
Module 4	4	3	2	$(\frac{h}{16}, \frac{w}{16}, 80)$
Module 5	2	5	1	$(\frac{h}{16}, \frac{w}{16}, 112)$
Module 6	4	5	1	$(\frac{h}{16}, \frac{w}{16}, 128)$
Module 7	2	3	1	$(\frac{h}{16}, \frac{w}{16}, 80)$
Head Conv 1	-	3	2	$(\frac{h}{8}, \frac{w}{8}, 128)$
Head Conv 2	-	3	2	$(\frac{h}{4}, \frac{w}{4}, 128)$
Head Conv 3	-	3	2	$(\frac{h}{2}, \frac{w}{2}, 128)$
Final Conv	-	1	1	$(\frac{h}{2}, \frac{w}{2}, K)$

Table 3.3: The architecture of our base 2D human pose network, EvoPose2D-S, designed via neuroevolution. With $h = 256$, $w = 192$, and $K = 17$, EvoPose2D-S contains 2.53M parameters and 1.07G FLOPs.

Model	Input Size	ϕ	c_d	c_w
EvoPose2D-M	384×288	2.90	1.70	1.32
EvoPose2D-L	512×384	4.96	2.47	1.60

Table 3.4: Scaling coefficients for EvoPose2D-M and L. See Eq. 3.5 for details.

half number of FLOPs compared to the largest version of HRNet. The scaling parameters were computed for EvoPose2D-M and L following Eq. 3.5 and are provided in Table 3.4; c_d scales the number of blocks in each module, rounded to the nearest integer, and c_w scales the number of output channels used in each block, rounded to the nearest multiple of eight.

3.4.3 Comparisons with the State of the Art

Following the neuroevolution experiments, the family of EvoPose2D networks were trained from scratch (*i.e.*, without ImageNet pretraining) on COCO as per the implementation described in Section 3.4.1. The accuracy of EvoPose2D is compared to state-of-the-art methods on COCO and PoseTrack.

Microsoft COCO Keypoints

To directly compare EvoPose2D with the best methods in the literature, SimpleBaseline ResNet-50 (SB-R50) and HRNet-W32 were re-implemented following Section 3.4.1. In the re-implementation of HRNet, a strided transpose pointwise convolution was used in place of a pointwise convolution followed by nearest-neighbour upsampling. This modification was required to make the model TPU-compatible, and did not change the number of parameters or FLOPs. The accuracy of the re-implementation is verified against the original in Table 3.5.

Comparing EvoPose2D-S with the SB-R50 re-implementation without ImageNet pretraining, it was found that EvoPose2D-S provides comparable accuracy on COCO `val2017` but is 50% faster and 12.7x smaller (see Table 3.5). EvoPose2D-S is also compared to a baseline that stacks the EvoPose2D network head on top of EfficientNet-B0 [223], and it is found that while EvoPose2D-S is 20% slower due to its decreased stride, its AP is 1.8 points higher and it is 2.2x smaller. Compared to the HRNet-W32 (256x192) re-implementation, EvoPose2D-M is more accurate by 1.5 AP while being 23% faster and 3.9x smaller.

Despite not using ImageNet pretraining, EvoPose2D-L achieves state-of-the-art AP on COCO `val2017`[2] (with and without PoseFix [248]) while being 4.3x smaller than HRNet-W48. Since EvoPose2D was designed using the COCO validation data, it is especially important to perform evaluation on COCO `test-dev`. In Table 3.5, it is shown that EvoPose2D-L also achieves state-of-the-art accuracy on COCO `test-dev`, again without ImageNet pretraining.

Since the development of EvoPose2D, two hand-designed top-down models have managed to match the accuracy of EvoPose2D-L on COCO `test-dev` using the same experimental protocol (*i.e.*, using no external data, the same person detections, and no model-agnostic post-processing enhancements). The Multi-Instance Pose Network (MIPNet) [59] addressed the problem of overlapping bounding boxes and occlusion by allowing the human pose estimator to predict multiple poses from a single bounding box. The other method was RLE [235], a heatmap-free method that was briefly discussed in Section 2.3.

[2]Higher AP has been reported using HRNet with model-agnostic improvements, including a better person detector and unbiased data processing [249].

Method	PT	Input Size	Params (M)	FLOPs (G)	Size (MB)	FPS (GPU)	AP	AR
COCO `val2017`								
CPN [50]	Y	256×192	27.0	6.20	–	–	68.6	–
SB-R50 [51]	Y	256×192	34.0	$5.21^{\dagger}$	$137^{\dagger}$	$67.7^{\dagger}$	70.4	76.3
SB (R-101) [51]	Y	256×192	53.0	$8.84^{\dagger}$	$214^{\dagger}$	$45.1^{\dagger}$	71.4	77.1
SB (R-152) [51]	Y	256×192	68.6	$12.5^{\dagger}$	$277^{\dagger}$	$34.4^{\dagger}$	72.0	77.8
HRNet-W32 [52]	N	256×192	28.5	$7.65^{\dagger}$	$119^{\dagger}$	$29.0^{\dagger}$	73.4	78.9
HRNet-W32 [52]	Y	256×192	28.5	$7.65^{\dagger}$	$119^{\dagger}$	$29.0^{\dagger}$	74.4	79.8
HRNet-W48 [52]	Y	256×192	63.6	$15.7^{\dagger}$	$259^{\dagger}$	$21.7^{\dagger}$	75.1	80.4
MSPN [53]	Y	256×192	120	19.9	–	–	75.9	–
SB (R-152) [51]	Y	384×288	68.6	$28.1^{\dagger}$	$277^{\dagger}$	$24.9^{\dagger}$	74.3	79.7
HRNet-W32 [52]	Y	384×288	28.5	$16.0^{\dagger}$	$119^{\dagger}$	$22.7^{\dagger}$	75.8	81.0
HRNet-W48 [52]	Y	384×288	63.6	$35.3^{\dagger}$	$259^{\dagger}$	$16.2^{\dagger}$	76.3	81.2
HRNet-W48* [248]	Y	384×288	63.6	$35.3^{\dagger}$	$259^{\dagger}$	$16.2^{\dagger}$	77.3	82.0
SB-R50	N	256×192	34.1	5.21	137	67.7	70.6	77.3
HRNet-W32	N	256×192	28.6	7.65	119	29.0	73.6	80.0
EfficientNet-B0‡	N	256×192	5.82	0.60	23.9	123	68.4	75.2
EvoPose2D-XS	N	256×192	**2.53**	**0.47**	**10.8**	**136**	68.0	75.0
EvoPose2D-S	N	256×192	**2.53**	1.07	**10.8**	102	70.2	76.9
EvoPose2D-M	N	384×288	7.34	5.59	30.7	35.8	75.1	81.0
EvoPose2D-L	N	512×384	14.7	17.7	60.6	15.9	76.6	82.3
EvoPose2D-L*	N	512×384	14.7	17.7	60.6	15.9	**77.5**	**82.5**
COCO `test-dev`								
CPN [50]	Y	384×288	-	-	-	-	72.1	78.5
SB (R-152) [51]	Y	384×288	68.6	$28.1^{\dagger}$	$277^{\dagger}$	$24.9^{\dagger}$	73.7	79.0
HRNet-W48 [52]	Y	384×288	63.6	$35.3^{\dagger}$	$259^{\dagger}$	**$16.2^{\dagger}$**	75.5	80.5
HRNet-W48* [248]	Y	384×288	63.6	$35.3^{\dagger}$	$259^{\dagger}$	**$16.2^{\dagger}$**	76.7	81.5
EvoPose2D-L	N	512×384	**14.7**	**17.7**	**60.6**	15.9	75.7	81.7
EvoPose2D-L*	N	512×384	**14.7**	**17.7**	**60.6**	15.9	**76.8**	**81.7**
PoseTrack `val2018`								
SB-R50	Y	256×192	34.1	5.21	137	67.7	54.3	59.9
EvoPose2D-S	Y	256×192	**2.53**	**1.07**	**10.8**	**102**	**55.1**	**60.6**
HRNet-W32	Y	256×192	28.6	7.65	119	29.0	58.7	**64.3**
EvoPose2D-M	Y	384×288	**7.34**	**5.59**	**30.7**	**35.8**	**60.4**	**64.3**

Table 3.5: Comparison of EvoPose2D against state-of-the-art methods on COCO and PoseTrack datasets. The `pycocotools` package was used for evaluation. For COCO, the models in the bottom sections were implemented as per Section 3.4.1. For PoseTrack, all models were implemented as described in Section 3.4.2 (see **PoseTrack** heading). PT: ImageNet pretraining for COCO, and COCO pretraining for PoseTrack. *: including PoseFix post-processing [248]. ‡: EvoPose2D network head stacked on top of EfficientNet-B0. †: recalculated for consistency. Network file size based on `float32` models. Frames per second (FPS) averaged over 1k forward passes on a TITAN Xp GPU using a batch size of 1.

PoseTrack

For evaluation on PoseTrack, all networks were initialized with the weights pretrained on COCO and the networks were fine-tuned on PoseTrack `train2018` (97k person instances). The final layers of the networks were modified to accommodate the different number of keypoints (K=15). All training details are consistent with Section 3.4.1, except the fine-tuning process was run for 20 epochs and early-stopping was used. The models were validated on `val2018` (45k person instances) using the ground-truth bounding boxes. As shown in Table 3.5, the relative performance of EvoPose2D compared to the state of the art is consistent with the COCO dataset: EvoPose2D-S and EvoPose2D-M provide higher accuracy than SB-R50 and HRNet-W32, respectively, despite having fewer parameters and FLOPs, and faster inference speed.

3.5 Discussion

In this chapter, a simple yet effective weight transfer scheme was proposed and was used to accelerate a neuroevolution of computationally efficient human pose networks for use in the two-stage, top-down human pose estimation paradigm. The search space design drew inspiration from the simplicity of the single-branch human pose estimation architecture introduced by Xiao *et al.* [51] and the computational benefits of inverted residual convolutional modules and squeeze-excitation [182, 251, 212, 223]. Using a multi-objective fitness function to balance the number of network parameters with pose estimation accuracy, the neuroevolution produced a baseline network (EvoPose2D-S) that has a favorable accuracy-speed trade-off compared to existing models and is better suited for mobile deployment due to its fast inference speed and small file size. When scaled, EvoPose2D achieved new state-of-the-art accuracy on the predominant multi-person human pose estimation benchmark dataset: Microsoft COCO Keypoints.

This research represents the first application of neuroevolution to human pose estimation, and more generally, the first application of neural architecture search to human pose estimation where state-of-the-art accuracy has been achieved. The utility of neural architecture search as a general network design tool is thus further supported by this research. However, the importance of the search space design choices should not be overlooked. Although the search space used was capable of producing 10^{14} unique network architectures, prior knowledge of efficient hand-designed networks greatly influenced the search space design and ultimately limited its expressiveness, especially as it relates to the topologies (*i.e.*, macroarchitectures) of the sampled networks. For instance, HRNet makes liberal use multi-branch and multi-scale feature aggregation [43, 50, 52], but these macroarchitectural traits cannot be manifested by the search space used in this study. In some sense, the EvoPose2D search space gives rise to a search for an optimal microarchitecture within a template macroarchitecture. As such, the proposed neuroevolution could be viewed as an extensive hyperparameter search. It may be contended, however, that all neural architecture searches are generalizations of hyperparameter searches. Importantly, the proposed search space led to the discovery of a state-of-the-art human pose network and the utility of the proposed weight transfer scheme in accelerating the search was effectively demonstrated.

The human bias induced when selecting a macroarchitectural template in neural architecture search is arguably unavoidable. Moreover, it would be naive not to exploit prior knowledge to narrow the search space and promote the sampling of architectures that are specialized for the task at hand. EvoPose2D is representative of how prior knowledge can be used effectively in neural architecture search. However, a dilemma is presented: the architectural optimization is always subject to the constraints imposed on the search space, and softening the search space constraints is a costly endeavour. Further, there is evidence to suggest that large search spaces are not only more expensive, but detrimental to the effectiveness of the search strategy [256, 232, 257]. These search space design challenges underscore the merits of human intuition and human-principled design. Humans are able to think outside the box when it comes to new network designs and methodologies for solving a problem; NAS algorithms will always be constrained by the search space. For example, in the heatmap-based search space of EvoPose2D, there is no potential for the neuroevolution to produce a more efficient representation for modeling keypoint locations. As pointed out in Section 2.3, heatmaps are costly to generate and post-process, yet their use is generally accepted as standard practice.

Perhaps the most consequential insight brought to light by this study is the importance of directly measuring inference speed as opposed to using FLOPs as a proxy for the same. Often, researchers seek to create more efficient network architectures by reducing the number of FLOPs (*e.g.*, FLOPs are often included in NAS optimization criteria [223, 258, 256]). Moreover, there are countless examples in the literature where researchers make comparisons and draw conclusions about computational efficiency based on FLOPs alone. In their defense, such comparisons are supported by studies like that of Canziani *et al.* [259], who analyzed some of the first CNNs used for ImageNet classification and concluded that the number of operations was a reliable estimate of inference time. In the present study, it was found that EvoPose2D-L and HRNet-48 have comparable inference speeds on a GPU despite EvoPose2D-L having half the number of FLOPs, which shows that the relationship between FLOPs and inference time is not so straightforward. In hindsight, it was found that recent architectural innovations such as skip connections [27] and squeeze-excitation [251] improve accuracy using fewer FLOPs but ultimately create inference-time bottlenecks because their hardware implementations are less efficient compared to standard convolution operations [260]. The choice of inference hardware is also a factor. In the previously cited study of Canziani *et al.*, an NVIDIA Jetson TX1 was used for the evaluations. The TX1 is a low-power embedded visual computing system with limited memory (4 GB). As a result, memory-intensive networks (*e.g.*, VGG [169]) run slower on the TX1. To illustrate, VGG19 (20 GFLOPs) was the slowest model tested and had an inference time of ∼155 ms per image using a batch size of 16. For comparison, ResNet-50 (3.9 GFLOPs) had an inference time of ∼55 ms. When sufficient memory bandwidth is available, however, such as on a v3-8 TPU, VGG19 runs faster than ResNet-50 (1.4 ms versus 2.5 ms per image using a batch size of 256) [261]. In fact, on TPUs VGG19 even runs faster than "efficiently designed" CNNs such as MobileNetV2 (0.30 GFLOPs) and EfficientNetB0 (0.39 GFLOPs) [261]. These contradicting results, in addition to the results presented in this chapter, highlight the importance of measuring inference speed directly and on target hardware, if possible.

Although neural architecture search is a powerful tool for optimizing CNN architectures, unavoidable search space constraints limit the discovery of novel approaches and different ways of conceptualizing the problem at hand. In the next chapter, human intuition is used to develop novel heatmap-free keypoint representations that model keypoint locations more efficiently than heatmaps. A human-principled network design approach is taken to develop a single-stage human pose estimation model that exploits the proposed keypoint representations. The inference speed and accuracy of the proposed model are directly compared to several state-of-the-art single-stage methods on the same hardware.

Chapter 4

KAPAO: Modeling Keypoints and Poses as Objects for Single-Stage Human Pose Estimation

In keypoint estimation tasks such as human pose estimation, heatmap-based regression has been the dominant approach despite possessing notable drawbacks: heatmaps intrinsically suffer from quantization error and require excessive computation to generate and post-process. Motivated to find a more efficient solution, this chapter presents a new heatmap-free keypoint estimation method in which individual keypoints and sets of spatially related keypoints (*i.e.*, poses) are modeled as objects within a dense single-stage anchor-based detection framework. Hence, the method was named KAPAO, for **K**eypoints **A**nd **P**oses **A**s **O**bjects. KAPAO was applied to the problem of single-stage multi-person human pose estimation by simultaneously detecting human pose objects and keypoint objects and fusing the detections to exploit the strengths of both object representations. In experiments, it was found that KAPAO is significantly faster and more accurate than previous single-stage human pose estimation methods, which suffer greatly from heatmap post-processing. Moreover, the accuracy-speed trade-off is especially favourable in the practical setting when not using test-time augmentation (TTA). The largest model, KAPAO-L, achieves an AP of 70.6 on the Microsoft COCO Keypoints validation set without test-time augmentation while being 3.1× faster than the next best single-stage model (CenterGroup [173]), whose accuracy is 1.5 AP less using the same test settings (*i.e.*, without TTA). Furthermore, KAPAO excels in the presence of heavy occlusion. On the CrowdPose test set, KAPAO-L achieves competitive accuracy for a single-stage method with an AP of 68.9. The source code is publicly available at `https://github.com/wmcnally/kapao`.

4.1 Introduction

As reviewed in Chapter 2, the most common method for estimating keypoint locations involves generating target fields referred to as heatmaps by centering 2D Gaussians with small variances on the target keypoint coordinates. Deep convolutional neural networks are then used to regress the target heatmaps on the input images, and keypoint predictions are made via the arguments of the maxima of the predicted heatmaps [41]. If a peak in the heatmap surpasses a predefined confidence threshold, then a keypoint is detected. The vast majority of human pose estimation models, including the one developed in the previous chapter, model and predict keypoints this way.

While strong empirical results have positioned heatmap regression as the *de facto* standard method for detecting and localizing keypoints, there are several known drawbacks. First, these methods suffer from quantization error, where the precision of a keypoint prediction is inherently limited by the spatial resolution of the heatmap. Larger heatmaps are therefore advantageous, but require additional upsampling operations and costly processing at higher resolutions [57, 54, 56, 172]. Even when large heatmaps are used, special post-processing steps are required to refine keypoint predictions, which can slow down inference [43, 50, 54, 56, 172]. Second, when two keypoints of the same type (*i.e.*, class) appear in close proximity to one another, the overlapping heatmap signals may be mistaken for a single keypoint. Indeed, this is a common failure case [45]. As discussed in Section 2.3, researchers have started investigating alternative, heatmap-free keypoint detection methods because of these drawbacks [96, 97, 236, 235, 117].

In this chapter, a new heatmap-free keypoint detection method is introduced and is applied to single-stage multi-person human pose estimation. The proposed method builds on recent research showing how keypoints can be modeled as objects within a dense anchor-based detection framework by representing keypoints at the center of small square *keypoint bounding boxes* [96][1]. In preliminary experimentation with human pose estimation, it was found that this keypoint detection approach works well for human keypoints that are characterized by local image features (*e.g.*, the eyes), but the same approach is less effective at detecting human keypoints that require a more global understanding (*e.g.*, the hips). To this end, a new *pose object* representation is introduced to help detect sets of keypoints that are spatially related. In the overall approach, keypoint objects and pose objects are

[1]I was the lead author of this paper. In this work, we encountered the issue of overlapping heatmap signals in the development of an automatic scoring system (DeepDarts) for steel-tip darts. To address this issue, we introduced the concept of modeling keypoints as objects using small square bounding boxes. This keypoint representation proved to be highly effective and served as the inspiration for KAPAO. The source code for DeepDarts is publicly available at https://github.com/wmcnally/deep-darts.

detected simultaneously and the detections are fused using a simple matching algorithm to exploit the benefits of both object representations. By virtue of detecting pose objects, person detection and keypoint estimation are effectively unified, leading to a highly efficient single-stage approach to multi-person human pose estimation.

KAPAO uses a recent implementation of the "You Only Look Once" (YOLO) dense detection network [198, 199] as its backbone. It includes a highly efficient network design that has been iteratively improved over several years of research and engineering. Because of its efficient network design, and the fact that it is not burdened by generating large and costly heatmaps that are subject to quantization error, KAPAO compares favourably against recent single-stage human pose estimation models in terms of accuracy *and* inference speed, especially when not using test-time augmentation (TTA), which is more representative of how these models are deployed in practice. As shown in Figure 4.1, KAPAO achieves an AP of 70.6 on the Microsoft COCO Keypoints validation set without TTA with an average latency of 54.4 ms (forward pass + post-processing time). Compared to the state-of-the-art single-stage model HigherHRNet + SWAHR [172], KAPAO is 5.1× faster and 3.3 AP more accurate when not using TTA. Compared to CenterGroup [173], which is faster and more accurate than HigherHRNet + SWAHR when not using TTA, KAPAO is still 3.1× faster and 1.5 AP more accurate. Moreover, pose objects are highly generalizable and can even be detected in depth images. In Section 4.3.6, several inference video demos are provided, including one where KAPAO detects pose objects in depth images with high confidence without having been trained on such images. The research contributions of this work are summarized below:

- A new *pose object* representation is proposed that extends the conventional object representation by additionally including a set of keypoints associated with the object. Supporting experiments demonstrate how to learn the pose object representation using a multi-task loss.

- A new approach to single-stage multi-person human pose estimation is developed by simultaneously detecting *keypoint objects* and *human pose objects* and fusing the detections to exploit the strengths of both object representations and improve the accuracy of the predicted poses. Compared to previous state-of-the-art methods, which all use heatmaps, the proposed heatmap-free method is significantly faster and more accurate on the Microsoft COCO Keypoints benchmark when not using test-time augmentation.

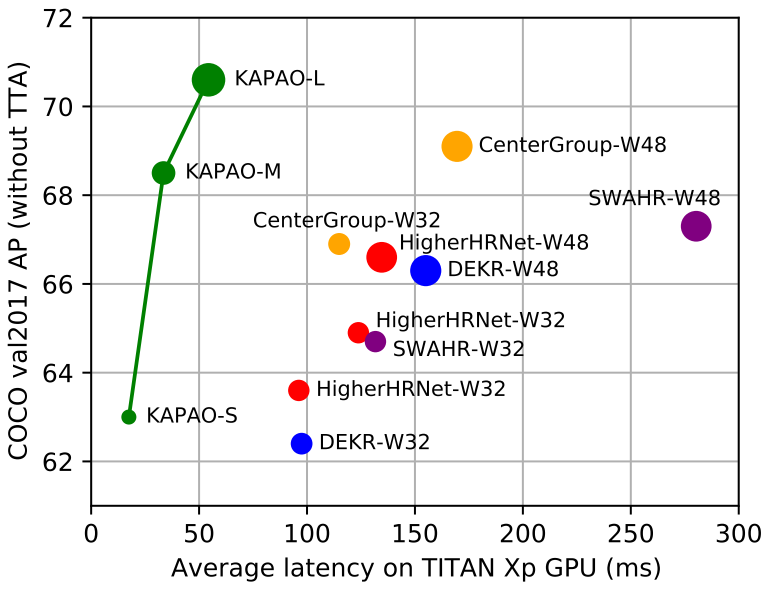


Figure 4.1: Accuracy vs. Inference Speed: KAPAO compared to state-of-the-art single-stage multi-person human pose estimation methods DEKR [56], HigherHRNet [54], HigherHRNet + SWAHR [172], and CenterGroup [173] without test-time augmentation (TTA). The raw data are provided in Table 4.2. The circle size is proportional to the number of model parameters.

4.2 Keypoints and Poses as Objects

In the proposed approach to multi-person human pose estimation, a dense detection network is trained to simultaneously predict a set of keypoint objects $\{\hat{\mathcal{O}}^k \in \hat{\mathbf{O}}^k\}$ and a set of pose objects $\{\hat{\mathcal{O}}^p \in \hat{\mathbf{O}}^p\}$, collectively $\hat{\mathbf{O}} = \hat{\mathbf{O}}^k \bigcup \hat{\mathbf{O}}^p$. The concept behind each object type is briefly introduced, along with the relevant notation. All units are assumed to be in pixels unless stated otherwise.

A keypoint object $\mathcal{O}^k$ is an adaptation of the conventional object representation in which the coordinates of a keypoint are represented at the center (b_x, b_y) of a small bounding box $\mathbf{b}$ with equal width b_w and height b_h: $\mathbf{b} = (b_x, b_y, b_w, b_h)$. The hyperparameter b_s controls the keypoint bounding box size (*i.e.*, $b_s = b_w = b_h$). There are K classes of keypoint objects, one for each keypoint type in the labeled dataset [96].

Generally speaking, a pose object $\mathcal{O}^p$ is considered to be an extension of the conventional object representation that additionally includes a set of keypoints associated with the object. While pose objects should be useful in related tasks such as facial and object landmark detection [117, 262], they are applied herein to human pose estimation via detection of *human pose objects*, comprising a bounding box of class “person,” and a set of keypoints $\mathbf{z} = \{(x_k, y_k)\}_{k=1}^{K}$ that coincide with anatomical landmarks.

Both object representations possess unique advantages. Keypoint objects are specialized for the detection of individual keypoints that are characterized by strong local features. Examples of such keypoints that are common in human pose estimation include the eyes, ears and nose. However, keypoint objects carry no information regarding the concept of a person or pose. If used on their own for multi-person human pose estimation, a bottom-up grouping method would be required to parse the detected keypoints into human poses. In contrast, pose objects are better suited for localizing keypoints with weak local features as they enable the network to learn the spatial relationships within a set of keypoints. Moreover, they can be leveraged for multi-person human pose estimation directly without the need for bottom-up keypoint grouping.

Recognizing that keypoint objects exist in a subspace of a pose objects, the KAPAO network is designed to simultaneously detect both object types with minimal computational overhead using a single shared network head. During inference, the more precise keypoint object detections are fused with the human pose detections using a simple tolerance-based matching algorithm that improves the accuracy of the human pose predictions without sacrificing any significant amount of inference speed. The essence of the KAPAO algorithm is illustrated in Figure 4.2. The following sections provide details on the network architecture, the loss function used to train the network, and inference.

4.2.1 Architectural Details

A diagram of the KAPAO pipeline is provided in Figure 4.3. It uses a deep convolutional neural network $\mathcal{N}$ to map an RGB input image $\mathbf{I} \in \mathbb{R}^{h\times w\times 3}$ to a set of four output grids $\hat{\mathbf{G}} = \{\hat{\mathcal{G}}^s \mid s \in \{8, 16, 32, 64\}\}$ containing the object predictions $\hat{\mathbf{O}}$, where $\hat{\mathcal{G}}^s \in \mathbb{R}^{\frac{h}{s}\times\frac{w}{s}\times N_a\times N_o}$:

$$\mathcal{N}(\mathbf{I}) = \hat{\mathbf{G}}. \tag{4.1}$$

N_a is the number of anchor channels and N_o is the number of output channels for each object. $\mathcal{N}$ is a YOLO-style feature extractor that makes extensive use of Cross-Stage-Partial (CSP) bottlenecks [263] within a feature pyramid [160] macroarchitecture. To provide flexibility for different speed requirements, three sizes of KAPAO models were trained (*i.e.*, KAPAO-S, KAPAO-M, and KAPAO-L) by scaling the number of layers and channels in $\mathcal{N}$. Relative to KAPAO-L, the number of layers and channels in KAPAO-M

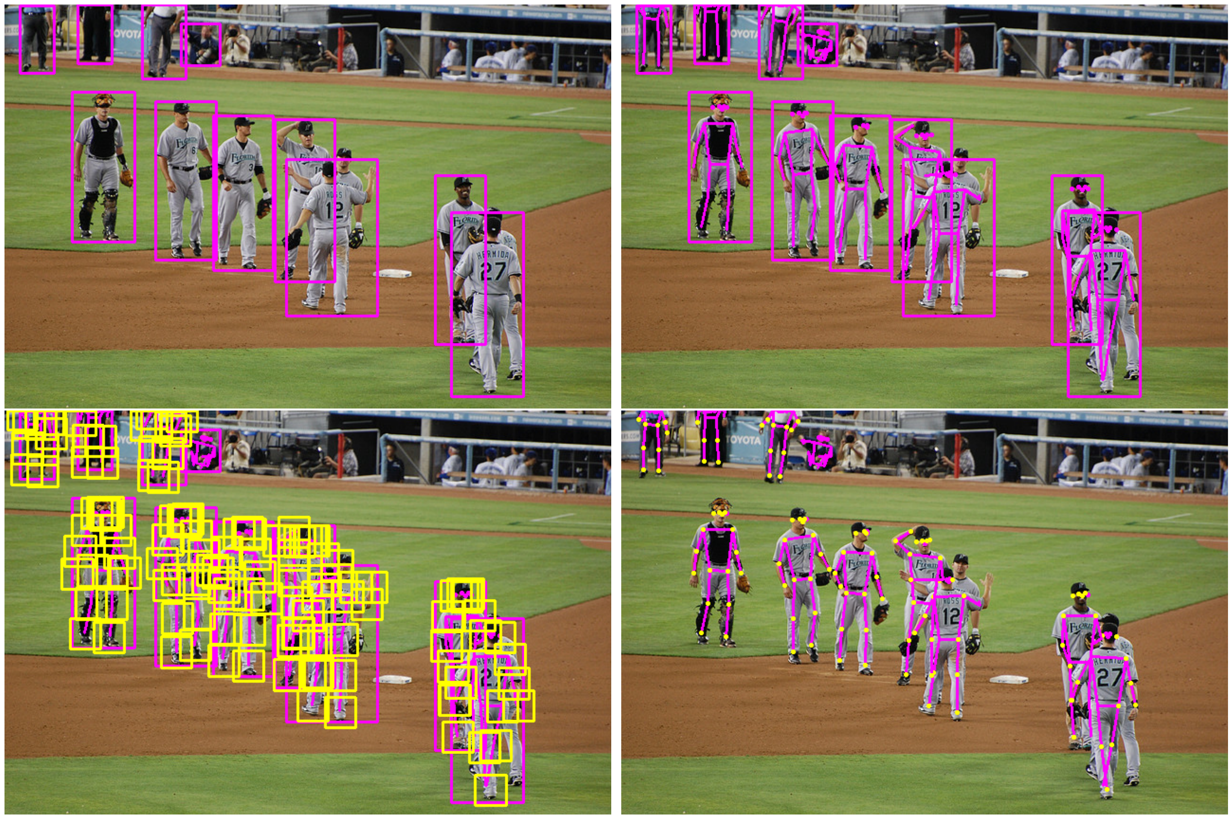


Figure 4.2: Illustrating the essence of modeling keypoints and poses as objects for single-stage multi-person human pose estimation. Top-left: conventional object detection. Top-right: pose object detection (extending the conventional object to include a set of keypoints). Bottom-left: keypoint object detections overlaid in yellow. Bottom-right: result of the fusing keypoint objects with the pose objects (fused keypoints represented in yellow).

were scaled by 2/3 and 3/4, respectively. Similarly, the number of layers and channels in KAPAO-S were scaled by 1/3 and 1/2, respectively [198, 199].

Due to the nature of strided convolutions, the features in an output grid cell $\hat{\mathcal{G}}^s_{i,j}$ are conditioned on the image patch $\mathbf{I}_p = \mathbf{I}_{si:s(i+1),sj:s(j+1)}$. Therefore, if the center of a target object (b_x, b_y) is situated in $\mathbf{I}_p$, the output grid cell $\hat{\mathcal{G}}^s_{i,j}$ is responsible for detecting it. The receptive field of an output grid increases with s, so smaller output grids are better suited for detecting larger objects. To give full play to this behaviour, the anchor boxes are also configured such that larger anchor boxes are used with the smaller grids. More details on the anchor box sizes are provided in Section 4.3.

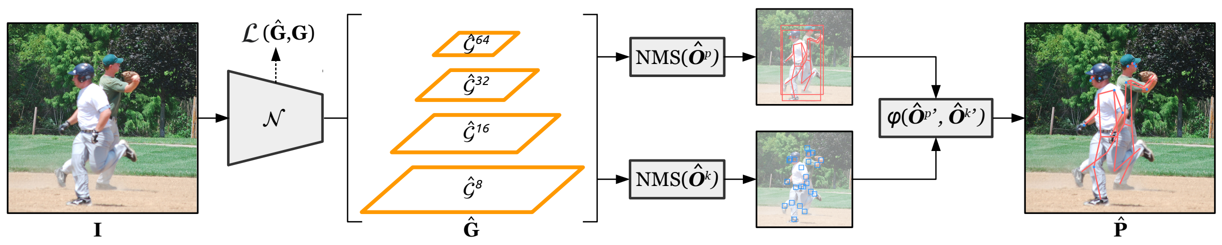

$\mathcal{L}(\hat{\mathbf{G}},\mathbf{G})$
$\mathcal{N}$
$\hat{\mathcal{G}}^{64}$
$\hat{\mathcal{G}}^{32}$
$\hat{\mathcal{G}}^{16}$
$\hat{\mathcal{G}}^{8}$
NMS($\hat{\mathbf{O}}^p$)
NMS($\hat{\mathbf{O}}^k$)
$\varphi(\hat{\mathbf{O}}^{p\prime}, \hat{\mathbf{O}}^{k\prime})$
$\mathbf{I}$
$\hat{\mathbf{G}}$
$\hat{\mathbf{P}}$

Figure 4.3: KAPAO uses a dense detection network $\mathcal{N}$ trained using the multi-task loss $\mathcal{L}$ to map an RGB image $\mathbf{I}$ to a set of output grids $\hat{\mathbf{G}}$ containing the predicted pose objects $\hat{\mathbf{O}}^p$ and keypoint objects $\hat{\mathbf{O}}^k$. Non-maximum suppression (NMS) is used to obtain candidate detections $\hat{\mathbf{O}}^{p\prime}$ and $\hat{\mathbf{O}}^{k\prime}$, which are fused together using a matching algorithm φ to obtain the final human pose predictions $\hat{\mathbf{P}}$. The N_a and N_o dimensions in $\hat{\mathbf{G}}$ are not shown for clarity.

The output grid cells $\hat{\mathcal{G}}^s_{i,j}$ contain N_a anchor channels corresponding to anchor boxes $\mathbf{A}^s = \{(A_{w_a}, A_{h_a})\}_{a=1}^{N_a}$. A target object $\mathcal{O}$ is assigned to an anchor channel via tolerance-based matching of the object and anchor box sizes. This provides redundancy such that the grid cells $\hat{\mathcal{G}}^s_{i,j}$ can detect multiple objects and enables specialization for different object sizes and shapes. Additional detection redundancy is provided by also allowing the neighbouring grid cells $\hat{\mathcal{G}}^s_{i\pm1,j}$ and $\hat{\mathcal{G}}^s_{i,j\pm1}$ to detect an object in $\mathbf{I}_p$ [198, 199].

The N_o output channels of $\hat{\mathcal{G}}^s_{i,j,a}$ contain the properties of a predicted object $\hat{\mathcal{O}}$, including the objectness $\hat{p}_o$ (the probability that an object exists), the intermediate bounding boxes $\hat{\mathbf{t}}' = (\hat{t}'_x, \hat{t}'_y, \hat{t}'_w, \hat{t}'_h)$, the object class scores $\hat{\mathbf{c}} = (\hat{c}_1, ..., \hat{c}_{K+1})$, and the intermediate keypoints $\hat{\mathbf{v}}' = \{(\hat{v}'_{xk}, \hat{v}'_{yk})\}_{k=1}^{K}$ for the human pose objects. Hence, $N_o = 3K + 6$.

Following [199, 198], an object's intermediate bounding box $\hat{\mathbf{t}}$ is predicted in the grid coordinates and relative to the grid cell origin (i, j) using:

$$\hat{t}_x = 2\sigma(\hat{t}'_x) - 0.5 \qquad \hat{t}_y = 2\sigma(\hat{t}'_y) - 0.5 \tag{4.2}$$

$$\hat{t}_w = \frac{A_w}{s}(2\sigma(\hat{t}'_w))^2 \qquad \hat{t}_h = \frac{A_h}{s}(2\sigma(\hat{t}'_h))^2. \tag{4.3}$$

This detection strategy was extended in this work to the keypoints of a pose object. A pose object's intermediate keypoints $\hat{\mathbf{v}}$ are predicted in the grid coordinates and relative to the grid cell origin (i, j) using:

$$\hat{v}_{xk} = \frac{A_w}{s}(4\sigma(\hat{v}'_{xk}) - 2) \qquad \hat{v}_{yk} = \frac{A_h}{s}(4\sigma(\hat{v}'_{yk}) - 2). \tag{4.4}$$

The sigmoid function σ facilitates learning by constraining the ranges of the object properties (*e.g.*, $\hat{v}_{xk}$ and $\hat{v}_{yk}$ are constrained to $\pm 2\frac{A_w}{s}$ and $\pm 2\frac{A_h}{s}$, respectively). To learn $\hat{\mathbf{t}}$ and $\hat{\mathbf{v}}$, losses are applied in the grid space. Sample targets $\mathbf{t}$ and $\mathbf{v}$ are shown in Figure 4.4.

4.2.2 Loss Function

A target set of grids $\mathbf{G}$ is constructed and a multi-task loss $\mathcal{L}(\hat{\mathbf{G}}, \mathbf{G})$ is applied to learn the objectness $\hat{p}_o$ ($\mathcal{L}_{obj}$), the intermediate bounding boxes $\hat{\mathbf{t}}$ ($\mathcal{L}_{box}$), the class scores $\hat{\mathbf{c}}$ ($\mathcal{L}_{cls}$), and the intermediate pose object keypoints $\hat{\mathbf{v}}$ ($\mathcal{L}_{kps}$). The loss components are computed for a single image as follows:

$$\mathcal{L}_{obj} = \sum_s \frac{\omega_s}{n(G^s)} \sum_{G^s} \mathrm{BCE}(\hat{p}_o, p_o \cdot \mathrm{IoU}(\hat{\mathbf{t}}, \mathbf{t})) \tag{4.5}$$

$$\mathcal{L}_{box} = \sum_s \frac{1}{n(\mathcal{O} \in G^s)} \sum_{\mathcal{O} \in G^s} 1 - \mathrm{IoU}(\hat{\mathbf{t}}, \mathbf{t}) \tag{4.6}$$

$$\mathcal{L}_{cls} = \sum_s \frac{1}{n(\mathcal{O} \in G^s)} \sum_{\mathcal{O} \in G^s} \mathrm{BCE}(\hat{c}, c) \tag{4.7}$$

$$\mathcal{L}_{kps} = \sum_s \frac{1}{n(\mathcal{O}^p \in G^s)} \sum_{\mathcal{O}^p \in G^s} \sum_{k=1}^{K} \delta(\nu_k > 0) ||\hat{\mathbf{v}}_k - \mathbf{v}_k||_2 \tag{4.8}$$

where ω_s is the grid weighting, BCE is the binary cross-entropy, IoU is the complete intersection over union (CIoU) [264], and ν_k are the visibility flags of the target keypoints $\mathbf{v}_k$. When $\mathcal{G}^s_{i,j,a}$ represents a target object $\mathcal{O}$, the target objectness $p_o = 1$ is multiplied by the IoU score to promote specialization amongst the anchor channel predictions [196]. When $\mathcal{G}^s_{i,j,a}$ is not a target object, $p_o = 0$. In practice, the losses are applied over a batch of images using batched grids. The total loss $\mathcal{L}$ is the weighted summation of the loss components scaled by the batch size N_b:

$$\mathcal{L} = N_b(\lambda_{obj}\mathcal{L}_{obj} + \lambda_{box}\mathcal{L}_{box} + \lambda_{cls}\mathcal{L}_{cls} + \lambda_{kps}\mathcal{L}_{kps}). \tag{4.9}$$

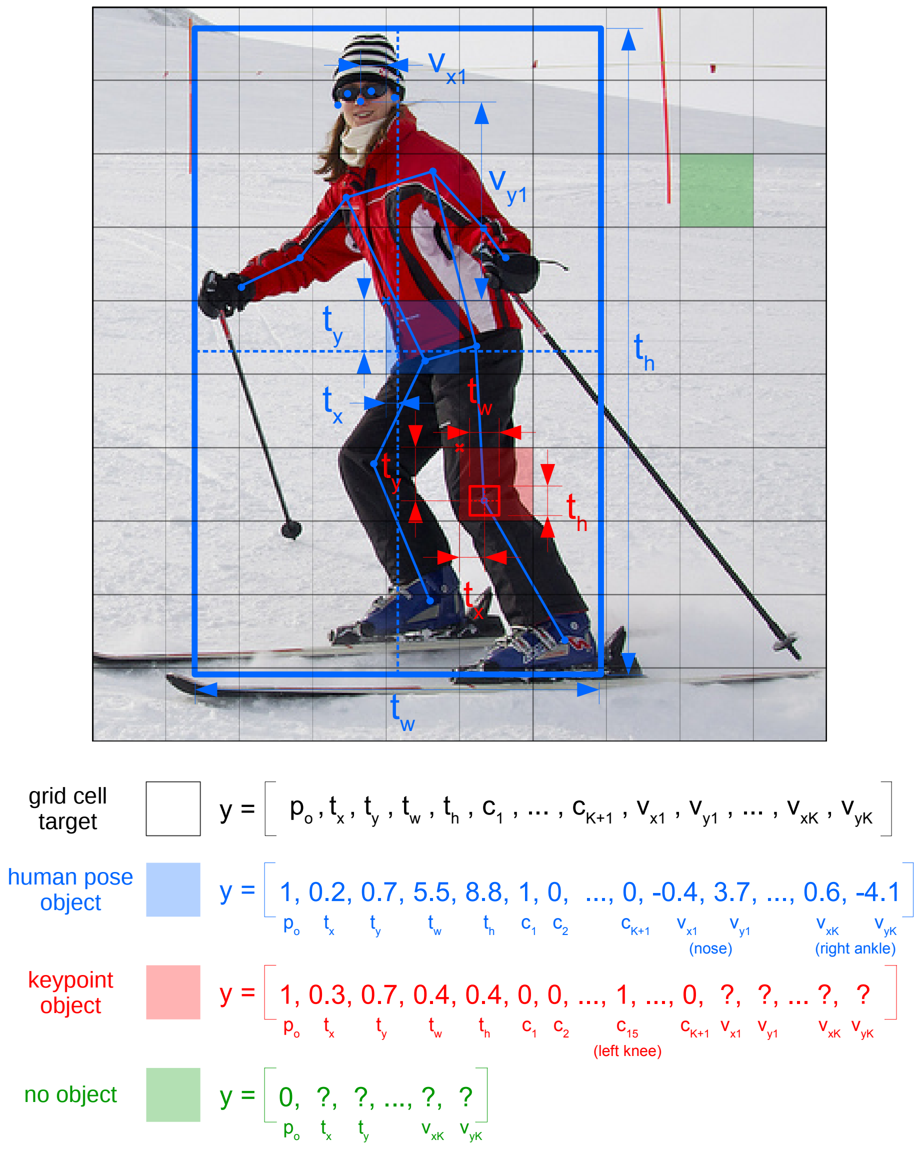

v_x1
v_y1
t_y
t_x
t_w
t_h
t_y
t_x
t_w
t_h
grid cell target
y = [p_o , t_x , t_y , t_w , t_h , c_1 , ... , c_K+1 , v_x1 , v_y1 , ... , v_xK , v_yK]
human pose object
y = [1, 0.2, 0.7, 5.5, 8.8, 1, 0, ..., 0, -0.4, 3.7, ..., 0.6, -4.1]
p_o t_x t_y t_w t_h c_1 c_2 c_K+1 v_x1 v_y1 v_xK v_yK
(nose)
(right ankle)
keypoint object
y = [1, 0.3, 0.7, 0.4, 0.4, 0, 0, ..., 1, ..., 0, ?, ?, ... ?, ?]
p_o t_x t_y t_w t_h c_1 c_2 c_15 c_K+1 v_x1 v_y1 v_xK v_yK
(left knee)
no object
y = [0, ?, ?, ..., ?, ?]
p_o t_x t_y v_xK v_yK

Figure 4.4: Sample targets used to train KAPAO, including a human pose object (blue), keypoint object (red), and no object (green). The "?" values are not used in the loss computation. In this example, a 10x10 output grid is overlaid on the input image. A single anchor box channel is assumed for clarity. The center of the pose object bounding box lies in grid cell $\mathcal{G}_{5,4}$, so that cell is responsible for its prediction. Sample target values for the N_o output channels of three grid cells are shown below the image.

4.2.3 Inference

The predicted intermediate bounding boxes $\hat{\mathbf{t}}$ and intermediate keypoints $\hat{\mathbf{v}}$ are mapped back to the original image coordinates using the following transformation:

$$\hat{\mathbf{b}} = s(\hat{\mathbf{t}} + [i, j, 0, 0]) \qquad \hat{\mathbf{z}}_k = s(\hat{\mathbf{v}}_k + [i, j]). \tag{4.10}$$

$\hat{\mathcal{G}}^s_{i,j,a}$ represents a positive pose object detection $\hat{\mathcal{O}}^p$ if its confidence $\hat{p}_o \cdot \max(\hat{\mathbf{c}})$ is greater than a threshold τ_{cp} and $\arg\max(\hat{\mathbf{c}}) = 1$. Similarly, $\hat{\mathcal{G}}^s_{i,j,a}$ represents a positive keypoint object detection $\hat{\mathcal{O}}^k$ if $\hat{p}_o \cdot \max(\hat{\mathbf{c}}) > \tau_{ck}$ and $\arg\max(\hat{\mathbf{c}}) > 1$, where the keypoint object class is $\arg\max(\hat{\mathbf{c}}) - 1$.

To remove redundant detections and obtain the candidate pose objects $\hat{\mathbf{O}}^{p\prime}$ and the candidate keypoint objects $\hat{\mathbf{O}}^{k\prime}$, the sets of positive pose object detections $\hat{\mathbf{O}}^p$ and positive keypoint object detections $\hat{\mathbf{O}}^p$ are filtered using non-maximum suppression (NMS) applied to the pose object bounding boxes with the IoU thresholds τ_{bp} and τ_{bk}[2]:

$$\hat{\mathbf{O}}^{p\prime} = \text{NMS}(\hat{\mathbf{O}}^p, \tau_{bp}) \qquad \hat{\mathbf{O}}^{k\prime} = \text{NMS}(\hat{\mathbf{O}}^k, \tau_{bk}). \tag{4.11}$$

Finally, the human pose predictions $\hat{\mathbf{P}} = \{\hat{\mathbf{P}}_i \in \mathbb{R}^{K\times 3}\}$ for $i \in \{1...n(\hat{\mathbf{O}}^{p\prime})\}$ are obtained by fusing the candidate keypoint objects with the candidate pose objects using a distance tolerance τ_{fd}. To promote correct matches of keypoint objects to poses, the keypoint objects are only fused to pose objects with confidence $\hat{p}_o \cdot \max(\hat{\mathbf{c}}) > \tau_{fc}$:

$$\hat{\mathbf{P}} = \varphi(\hat{\mathbf{O}}^{p\prime}, \hat{\mathbf{O}}^{k\prime}, \tau_{fd}, \tau_{fc}). \tag{4.12}$$

The keypoint object fusion function φ is defined in Algorithm 2, where the following notation is used to index an object's properties: $\hat{x} = \hat{\mathcal{O}}_x$ (*e.g.*, a pose object's keypoints $\hat{\mathbf{z}}$ are referenced as $\hat{\mathcal{O}}^p_{\mathbf{z}}$).

4.2.4 Limitations

A limitation of KAPAO is that pose objects do not include individual keypoint confidences, so the human pose predictions typically contain a sparse set of keypoint confidences $\hat{\mathbf{P}}_i[:, 3]$ populated by the fused keypoint objects (see Algorithm 2 for details). If desired, a complete set of keypoint confidences can be induced by only using keypoint objects, which is realized

[2] τ_{ck} and τ_{bk} are scalar thresholds and are used for all keypoint classes.

Algorithm 2: Keypoint object fusion (φ)

```
Input: $\hat{\mathbf{O}}^{p\prime}$, $\hat{\mathbf{O}}^{k\prime}$, $\tau_{fd}$, $\tau_{fc}$
Output: $\hat{\mathbf{P}}$
if $n(\hat{\mathbf{O}}^{p\prime}) > 0$ then
    $\hat{\mathbf{P}} \leftarrow \{0_{K\times 3} \mid \_ \in \{1...n(\hat{\mathbf{O}}^{p\prime})\}\}$ // initialize poses
    $\zeta \leftarrow \{0 \mid \_ \in \{1...n(\hat{\mathbf{O}}^{p\prime})\}\}$ // initialize pose confidences
    for $(i, \hat{\mathcal{O}}^p) \in$ enumerate$(\hat{\mathbf{O}}^{p\prime})$ do
        $\zeta_i = \hat{\mathcal{O}}^p_{p_o} \cdot \max(\hat{\mathcal{O}}^p_{\mathbf{c}})$
        for $k \in \{1...K\}$ do
            $\hat{\mathbf{P}}_i[k] \leftarrow (\hat{\mathcal{O}}^p_{x_k}, \hat{\mathcal{O}}^p_{y_k}, 0)$
    $\hat{\mathbf{P}}^* \leftarrow \{\hat{\mathbf{P}}_i \in \hat{\mathbf{P}} \mid \zeta_i > \tau_{fc}\}$
    if $n(\hat{\mathbf{P}}^*) > 0 \wedge n(\hat{\mathbf{O}}^{k\prime}) > 0$ then
        for $\hat{\mathcal{O}}^k \in \hat{\mathbf{O}}^{k\prime}$ do
            $k \leftarrow \arg\max(\hat{\mathcal{O}}^k_{\mathbf{c}}) - 1$
            $C_k \leftarrow \hat{\mathcal{O}}^k_{p_o} \max(\hat{\mathcal{O}}^k_{\mathbf{c}})$ // keypoint object confidence
            $\mathbf{d}_i \leftarrow ||\hat{\mathbf{P}}^*_i[k, [1, 2]] - (\hat{\mathcal{O}}^k_{b_x}, \hat{\mathcal{O}}^k_{b_y})||_2$
            $m \leftarrow \arg\min(\mathbf{d})$ // match index
            if $\mathbf{d}_m < \tau_{fd} \wedge \hat{\mathbf{P}}^*_m[k, 3] < C_k$ then
                $\hat{\mathbf{P}}^*_m[k] = (\hat{\mathcal{O}}^k_{b_x}, \hat{\mathcal{O}}^k_{b_y}, C_k)$
else
    $\hat{\mathbf{P}} = \emptyset$ // empty set
```

when $\tau_{ck} \rightarrow 0$ (demonstrated in a video inference demo described in Section 4.3.6). Another limitation is that training requires a considerable amount of time and GPU memory due to the large input size used.

4.3 Experiments

KAPAO was trained and tested on two multi-person human pose estimation datasets: Microsoft COCO Keypoints [60] and CrowdPose [103]. This section provides the experimental details and test results. The accuracy and speed of KAPAO are compared against state-of-the-art methods in Sections 4.3.1 and 4.3.2. An error analysis is provided in Section 4.3.3, qualitative comparisons are made in Section 4.3.4, and certain design characteristics are

Hyperparameter Description	Symbol	Value(s)
output grid scales	s	$\{8, 16, 32, 64\}$
keypoint object bounding box size (px)	b_s	64
input image height, width (px)	h, w	1280, 1280
$\mathcal{G}^8$ anchor boxes (width, height) (px)	$\mathbf{A}^8$	$\{(19, 27), (44, 40), (38, 94)\}$
$\mathcal{G}^{16}$ anchor boxes (width, height) (px)	$\mathbf{A}^{16}$	$\{(96, 68), (86, 152), (180, 137)\}$
$\mathcal{G}^{32}$ anchor boxes (width, height) (px)	$\mathbf{A}^{32}$	$\{(140, 301), (303, 264), (238, 542)\}$
$\mathcal{G}^{64}$ anchor boxes (width, height) (px)	$\mathbf{A}^{64}$	$\{(436, 615), (739, 380), (925, 792)\}$
loss weights for $\mathcal{G}^8$, $\mathcal{G}^{16}$ $\mathcal{G}^{32}$, and $\mathcal{G}^{64}$	ω	$\{4.0, 1.0, 0.25, 0.06\}$
objectness loss weight	λ_{obj}	$0.7 \times (w/640)^2 \times 3/n(s)$
bounding box loss weight	λ_{box}	$0.05 \times 3/n(s)$
class loss weight	λ_{cls}	$0.3 \times (K + 1)/80 \times 3/n(s)$
pose object keypoints loss weight	λ_{kps}	$0.025 \times 3/n(s)$
batch sizes for KAPAO-S, M, and L	N_b	$128, 72, 48$
pose, keypoint obj. conf. thresholds	τ_{cp}, τ_{ck}	0.001, 0.2
pose, keypoint obj. IoU thresholds	τ_{bp}, τ_{bk}	0.65, 0.25
maximum fusion distance (px)	τ_{fd}	50
pose obj. conf. threshold for fusion	τ_{fc}	0.3

Table 4.1: The hyperparameters used in the KAPAO experiments. $n(s)$ is the number of output grids.

empirically analyzed in Section 4.3.5.

PyTorch 1.9 was used for the implementation. For convenience, the KAPAO hyperparameters used to generate the results in this section are provided in Table 4.1 in the order they appeared in the previous section. Other hyperparameters not referenced in the text (*e.g.*, the augmentation settings) are included in the source code. Many of the hyperparameters were inherited from [199], where an evolutionary algorithm was used to search for optimal values for object detection on COCO. Some hyperparameters, such as the keypoint bounding box size b_s and the keypoint loss weight λ_{kps}, were manually tuned using a small grid search. The influence of b_s is studied in Section 4.3.5.

4.3.1 Microsoft COCO Keypoints

Training. KAPAO-S/M/L were all trained for 500 epochs on COCO `train2017` using stochastic gradient descent with Nesterov momentum [265], weight decay, and a learning rate decayed over a single cosine cycle [255] with a 3-epoch warm-up period [254]. The input images were resized and padded to 1280×1280, keeping the original aspect ratio. Data augmentation used during training included mosaic [186], HSV color-space perturbations, horizontal flipping, translations, and scaling. The models were trained on four V100 GPUs with 32 GB memory each using batch sizes of 128, 72, and 48 for KAPAO-S, M, and L, respectively. Validation was performed after every epoch, saving the model weights that provided the highest validation AP.

Testing. The inference parameters (τ_{cp}, τ_{ck}, τ_{bp}, τ_{bk}, τ_{fd}, and τ_{fc}) were manually tuned on the validation set using a coarse grid search to maximize accuracy. However, the results were not overly sensitive to the inference parameter values. For test-time augmentation, the input image was scaled by factors of 0.8, 1, and 1.2, and the unscaled image was horizontally flipped. During post-processing, the multi-scale detections are concatenated before running NMS. When not using test-time augmentation, rectangular input images are used (*i.e.*, 1280 px on the longest side), which marginally reduced the accuracy but increased the inference speed.

Results. Table 4.2 compares the accuracy and latency (sum of the forward pass and post-processing times) of KAPAO with state-of-the-art single-stage methods HigherHRNet [54], HigherHRNet + SWAHR [172], DEKR [56], and CenterGroup [173] on `val2017`. Two test settings were considered: (1) without any test-time augmentation (*i.e.*, using a single forward pass of the network), and (2) with multi-scale and horizontal flipping test-time augmentation (TTA). The setting without TTA is representative of how these models are deployed in practice, whereas TTA is used to maximize accuracy at the cost of inference speed. All latencies were averaged over the 5k images in `val2017` using a batch size of 1 and were measured on the same workstation housing a TITAN Xp GPU. Many new insights are uncovered through evaluation of the forward pass (FP) and post-processing (PP) times required by each method. It is noted that with the exception of CenterGroup, no inference speeds were reported in the original works. Rather, FLOPs were used as an indirect measure of computational efficiency. FLOPs are not only a poor indication of inference speed (see Section 3.5 for discussion), but they are also only computed for the forward pass of the network and so they do not provide an indication of the amount of computation required for post-processing.

The post-processing and refinement of the large heatmaps and associative embeddings

Method	TTA	Input Size(s)	Params (M)	FP (ms)	PP (ms)	Lat. (ms)	AP	AR
HigherHRNet-W32 [54]	N	512	28.6	46.1	50.1	96.2	63.6	69.0
+ SWAHR [172]	N	512	28.6	45.1	86.6	132	64.7	70.3
HigherHRNet-W32 [54]	N	640	28.6	52.4	71.4	124	64.9	70.3
HigherHRNet-W48 [54]	N	640	63.8	75.4	59.2	135	66.6	71.5
+ SWAHR [172]	N	640	63.8	86.3	194	280	67.3	73.0
DEKR-W32 [56]	N	512	29.6	62.6	34.9	97.5	62.4	69.6
DEKR-W48 [56]	N	640	65.7	109	45.8	155	66.3	73.2
CenterGroup-W32 [173]*	N	512	30.3	98.9	16.0	115	66.9	71.6
CenterGroup-W48 [173]*	N	640	65.5	155	14.5	170	69.1	74.0
KAPAO-S	N	1280	**12.6**	**14.7**	**2.80**	**17.5**	63.0	70.2
KAPAO-M	N	1280	35.8	30.7	2.88	33.5	68.5	75.5
KAPAO-L	N	1280	77.0	51.3	3.07	54.4	**70.6**	**77.4**
HigherHRNet-W32 [54]	Y	256, 512, 1024	28.6	365	372	737	69.9	74.3
+ SWAHR [172]	Y	256, 512, 1024	28.6	389	491	880	71.3	75.9
HigherHRNet-W32 [54]	Y	320, 640, 1280	28.6	431	447	878	70.6	75.0
HigherHRNet-W48 [54]	Y	320, 640, 1280	63.8	643	436	1080	72.1	76.1
+ SWAHR [172]	Y	320, 640, 1280	63.8	809	781	1590	73.0	77.6
DEKR-W32 [56]	Y	256, 512, 1024	29.6	552	137	689	70.5	76.2
DEKR-W48 [56]	Y	320, 640, 1280	65.7	1010	157	1170	72.1	77.8
CenterGroup-W32 [173]*	Y	256, 512, 1024	30.3	473	13.8	487	71.9	76.1
CenterGroup-W48 [173]*	Y	320, 640, 1280	65.5	1050	11.8	1060	**73.3**	77.6
KAPAO-S	Y	1024, 1280, 1536	**12.6**	**61.5**	**3.70**	**65.2**	64.4	71.5
KAPAO-M	Y	1024, 1280, 1536	35.8	126	3.67	130	69.9	76.8
KAPAO-L	Y	1024, 1280, 1536	77.0	211	3.70	215	71.6	**78.5**

Table 4.2: Accuracy and speed comparison with state-of-the-art single-stage human pose estimation models on COCO `val2017`, including the forward pass (FP) and post-processing (PP). Latencies (Lat.) averaged over `val2017` using a batch size of 1 on a TITAN Xp GPU. *Uses `mmpose pytorch-lightning` HigherHRNet implementation, which has faster forward pass time than the original (37.9 ms versus 46.1 ms for HigherHRNet-W32).

produced by HigherHRNet-W32 require more processing time than the forward pass of the network itself. Furthermore, the post-processing time of HigherHRNet correlates with the size of the output heatmaps, resulting in very costly multi-scale inference when using TTA. Despite using identical network architectures and inference code, SWAHR seemingly adds significant computational overhead to HigherHRNet during post-processing, especially when using the heavier W48 backbone. Upon further investigation, it was found that using SWAHR results in more detections (*i.e.*, fewer false negatives), which increases the accuracy but has the side-effect of slowing down inference through additional heatmap refinement.

For DEKR, the post-processing time is slightly less than HigherHRNet without TTA and significantly less when using TTA. The reason is two-fold: the heatmaps and offset fields produced by the HRNet backbone in DEKR are half the size compared to HigherHRNet, and only the pose center heatmap requires refinement to obtain pose center candidates while the $2K$ offset fields are indexed at the center heatmap maxima to predict poses. However, DEKR's reduced post-processing time when using TTA is offset by a greater forward pass time, which is suspected to be caused by its architectural complexities, including its use of adaptive convolutions and separate regression heads for each keypoint.

KAPAO has the advantage of not using heatmaps, so the post-processing times of HigherHRNet, HigherHRNet + SWAHR, and DEKR are at least an order of magnitude greater than KAPAO-L when not using TTA. Furthermore, the post-processing time of KAPAO depends less on the input size so it only increases by approximately 1 ms when using TTA. Conversely, HigherHRNet and HigherHRNet + SWAHR generate and refine large heatmaps with multi-scale testing and therefore require more than two orders of magnitude more post-processing time than KAPAO-L when using TTA.

CenterGroup, which uses a HigherHRNet backbone, requires significantly less post-processing time than HigherHRNet and DEKR because it skips heatmap refinement and directly encodes pose center and keypoint heatmaps as embeddings that are fed to an attention-based grouping module that predicts associations among the keypoints and centers. When not using TTA, CenterGroup-W48 provides an improvement of 2.5 AP over HigherHRNet-W48, so the attention-based grouping module is more effective than associative embeddings for learning the groupings of keypoints. However, the grouping module is relatively slow. It is responsible for over 50% of the total forward pass time of CenterGroup, which offsets the reduction in post-processing time. Still, CenterGroup has the best accuracy-speed trade-off of the models discussed thus far. Equipped with an efficient network architecture and near cost-free post-processing, KAPAO-L is 3.1× faster than CenterGroup-W48 and 1.5 AP more accurate when not using TTA.

When using TTA, the AP of KAPAO-L is 1.7 points less than CenterGroup-W48, but

KAPAO-L is 4.9$\times$ faster. KAPAO-L also provides state-of-the-art AR, indicating that it potentially provides more true positive detections and fewer false negatives. Because KAPAO was trained on larger images, there is less benefit from multi-scale testing. Using input sizes greater than 1280 provides less of a benefit as the dataset images are a maximum of 640 pixels on their longest side. In effect, KAPAO reaps the benefits of multi-scale testing without explicitly using it. If larger images were available, it is suspected that KAPAO would benefit more from multi-scale testing where the other methods would not.

In Table 4.3, the accuracy of KAPAO is compared to single-stage and two-stage methods on `test-dev`. The results with the highest AP are reported (*i.e.*, including TTA). KAPAO-L achieves state-of-the-art AR and falls within 1.7 AP of best performing single-stage method HigherHRNet-W48 + SWAHR while being 7.4$\times$ faster. Notably, KAPAO-L is more accurate than the early two-stage methods G-RMI [159] and RMPE [44]. Moreover, KAPAO-L is significantly more accurate than landmark single-stage methods like OpenPose [45, 168], Associative Embeddings [46], and PersonLab [48]. Compared to other single-stage methods that extend object detectors for human pose estimation (Mask R-CNN [47], CenterNet [113], Point-Set Anchors [165], and FCPose [166]), KAPAO-L is considerably more accurate. Among all the single-stage methods, KAPAO-L achieves state-of-the-art AP at an OKS threshold of 0.50, which could be an indication of better detection but less precise keypoint localization.

4.3.2 CrowdPose

KAPAO was trained on the `trainval` split with 12k images and was evaluated on the 8k images in `test`. The same training and inference settings as on COCO were used except the models were trained for 300 epochs instead of 500 and no validation was performed. The final model weights were used for testing. Table 4.4 compares the accuracy of KAPAO against state-of-the-art methods. It was found that KAPAO excels in the presence of occlusion, achieving competitive results across all metrics compared to previous single-stage methods. The proficiency of KAPAO in crowded scenes is clear when analyzing AP^E, AP^M, and AP^H: KAPAO-L and DEKR-W48 [56] perform equally on images with easy Crowd Index (less occlusion), but KAPAO-L is 1.1 AP more accurate for both medium and hard Crowd Indices (more occlusion).

Method	Lat. (ms)	AP	AP$^{.50}$	AP$^{.75}$	AP^{M}	AP^{L}	AR
G-RMI [159]†	-	64.9	85.5	71.3	62.3	70.0	69.7
RMPE [44]†	-	61.8	83.7	69.8	58.6	67.6	-
CPN [50]†	-	72.1	91.4	80.0	68.7	77.2	78.5
SimpleBaseline [51]†	-	73.7	91.9	81.1	70.3	80.0	79.0
HRNet-W48 [52]†	-	75.5	**92.5**	**83.3**	71.9	**81.5**	80.5
EvoPose2D-L [57]†	-	**75.7**	91.9	83.1	72.2	**81.5**	**81.7**
MIPNet [59]†	-	**75.7**	-	-	-	-	-
RLE [235]†	-	**75.7**	92.3	82.9	**72.3**	81.3	-
OpenPose [45, 168]	74*	61.8	84.9	67.5	57.1	68.2	66.5
Mask R-CNN [47]	-	63.1	87.3	68.7	57.8	71.4	-
Associative Embeddings [46]	-	65.5	86.8	72.3	60.6	72.6	70.2
PersonLab [48]	-	68.7	89.0	75.4	64.1	75.5	75.4
SPM [156]	-	66.9	88.5	72.9	62.6	73.1	-
PifPaf [155]	-	66.7	-	-	62.4	72.9	-
HGG [171]	-	67.6	85.1	73.7	62.7	74.6	71.3
CenterNet [113]	-	63.0	86.8	69.6	58.9	70.4	-
Point-Set Anchors [165]	-	68.7	89.9	76.3	64.8	75.3	-
HigherHRNet-W48 [54]	1080	70.5	89.3	77.2	66.6	75.8	74.9
+ SWAHR [172]	1590	**72.0**	90.7	**78.8**	**67.8**	**77.7**	-
FCPose [166]	93*	65.6	87.9	72.6	62.1	72.3	-
DEKR-W48 [56]	1170	71.0	89.2	78.0	67.1	76.9	76.7
CenterGroup-W48 [173]	1060	71.4	90.5	78.1	67.2	77.5	-
KAPAO-S	**65.2**	63.8	88.4	70.4	58.6	71.7	71.2
KAPAO-M	130	68.8	90.5	76.5	64.3	76.0	76.3
KAPAO-L	215	70.3	**91.2**	77.8	66.3	76.8	**77.7**

Table 4.3: Accuracy comparison with two-stage (†) and single-stage methods on COCO `test-dev`. Best results (*i.e.*, including TTA). DEKR results use a model-agnostic rescoring network [56]. Latencies (Lat.) taken from Table 4.2. *Latencies reported in original papers [168, 166] and measured using an NVIDIA GTX 1080Ti GPU.

Method	Lat. (ms)	AP	$AP^{.50}$	$AP^{.75}$	AP^{E}	AP^{M}	AP^{H}
Mask R-CNN [47]	-	57.2	83.5	60.3	69.4	57.9	45.8
AlphaPose [44]†	-	61.0	81.3	**66.0**	**71.2**	**61.4**	**51.1**
SimpleBaseline [51]†	-	60.8	81.4	65.7	71.4	61.2	51.2
SPPE [103]	-	66.0	84.2	71.5	75.5	66.3	57.4
MIPNet [59]†	-	**70.0**	-	-	-	-	-
OpenPose [45]	74.0*	-	-	-	62.7	48.7	32.3
HigherHRNet-W48 [54]	1080	67.6	87.4	72.6	75.8	68.1	58.9
+ SWAHR [172]	1590	**73.8**	**90.5**	**79.9**	**81.2**	**74.7**	**64.7**
DEKR-W48 [56]	1170	68.0	85.5	73.4	76.6	68.8	58.4
CenterGroup-W48 [173]	1060	70.0	88.9	75.7	77.3	70.8	63.2
KAPAO-S	**65.2**	63.8	87.7	69.4	72.1	64.8	53.2
KAPAO-M	130	67.1	88.8	73.4	75.2	68.1	56.9
KAPAO-L	215	68.9	89.4	75.6	76.6	69.9	59.5

Table 4.4: Accuracy comparison with single-stage and two-stage (†) methods on CrowdPose `test`, including TTA. DEKR results use a model-agnostic rescoring network [56]. Latencies (Lat.) taken from Table 4.2. *Latency reported in original paper [168] and measured using NVIDIA GTX 1080Ti GPU on COCO.

4.3.3 Error Analysis

While the AP and AR metrics are robust and perceptually meaningful (*i.e.*, algorithms with higher AP/AR are generally more accurate), they can hide the underlying causes of error and are not sufficient for truly understanding an algorithm's behaviour. For example, the results presented in the previous section showed that KAPAO consistently provides higher AR than previous single-stage methods and higher AP at lower OKS thresholds (*e.g.*, $AP^{.50}$). The exact cause for this result cannot be understood through analysis of the AP/AR metrics alone, but a potential explanation is that KAPAO provides more precise person/pose detection (*i.e.*, more true positives and fewer false positives and false negatives), but less precise keypoint localization. To investigate this further, this section provides a more in-depth analysis of the error using the error taxonomy and analysis code provided by Ronchi and Perona [106].

Ronchi and Perona propose an error taxonomy for multi-person human pose estimation on COCO that includes four error categories: *Background False Positives*, *False Negatives*, *Scoring*, and *Localization*. False positives and false negatives were previously described in Section 2.1.1. Scoring errors are due to sub-optimal confidence score assignment; they occur when two detections are in the proximity of a ground-truth annotation and the one with the highest confidence has the lowest OKS. Localization errors are due to poor keypoint localization within a detected instance; they are further categorized into four types: *Jitter*: small localization error ($0.5 \leq \exp(-d_i^2/2s^2k_i^2) < 0.85$); *Miss*: large localization error ($\exp(-d_i^2/2s^2k_i^2) < 0.5$); *Inversion*: left-right keypoint flipping within an instance; and *Swap*: keypoint swapping between instances. The reader may refer to [106] for a more detailed description of the error classifications.

Figure 4.5 plots the precision-recall curves for HigherHRNet-W48 [54], HigherHRNet-W48 + SWAHR [172], DEKR-W48 [56], CenterGroup-W48 [173], and KAPAO-L at an OKS threshold of 0.85 using the results without TTA. Recalling that AP^{α} is equal to the area under the precision-recall curve generated at $\mathrm{OKS} = \alpha$, the coloured regions in Figure 4.5 reflect the theoretical improvement in $\mathrm{AP}^{.85}$ as a result of sequentially rectifying the aforementioned error types.

KAPAO-L provides the highest original $\mathrm{AP}^{.85}$ (represented by the white region). Furthermore, KAPAO-L is less prone to Swap and Inversion errors, which can be attributed to the pose object representation that models cohesive pose instances and eliminates the need for bottom-up keypoint grouping algorithms that are prone to such errors. This is further supported by DEKR having lower Swap error than the other three methods, which all perform bottom-up keypoint grouping. Interestingly, DEKR has the most room for improvement in assigning optimal confidence scores, which could be the motivation behind using a model-agnostic scoring regression network to improve the AP in the original paper (not included in these results). As previously hypothesized, KAPAO has more Jitter error than some of the other methods as a result of having less precise keypoint localization. Conversely, KAPAO-L provides better detection as shown by less improvement after correcting the false positive and false negative errors. It is speculated that since KAPAO was designed using an object detection network as its backbone, it is more optimized for person/pose object detection and less optimized for keypoint localization compared to the other single-stage human pose estimation methods. Rebalancing KAPAO to focus more on keypoint localization is thus a recommended area for future work.

4.3.4 Qualitative Comparisons

KAPAO-L predictions from COCO `val2017` are qualitatively compared with CenterGroup-W48 without TTA. To systematically review cases where KAPAO-L performs better than

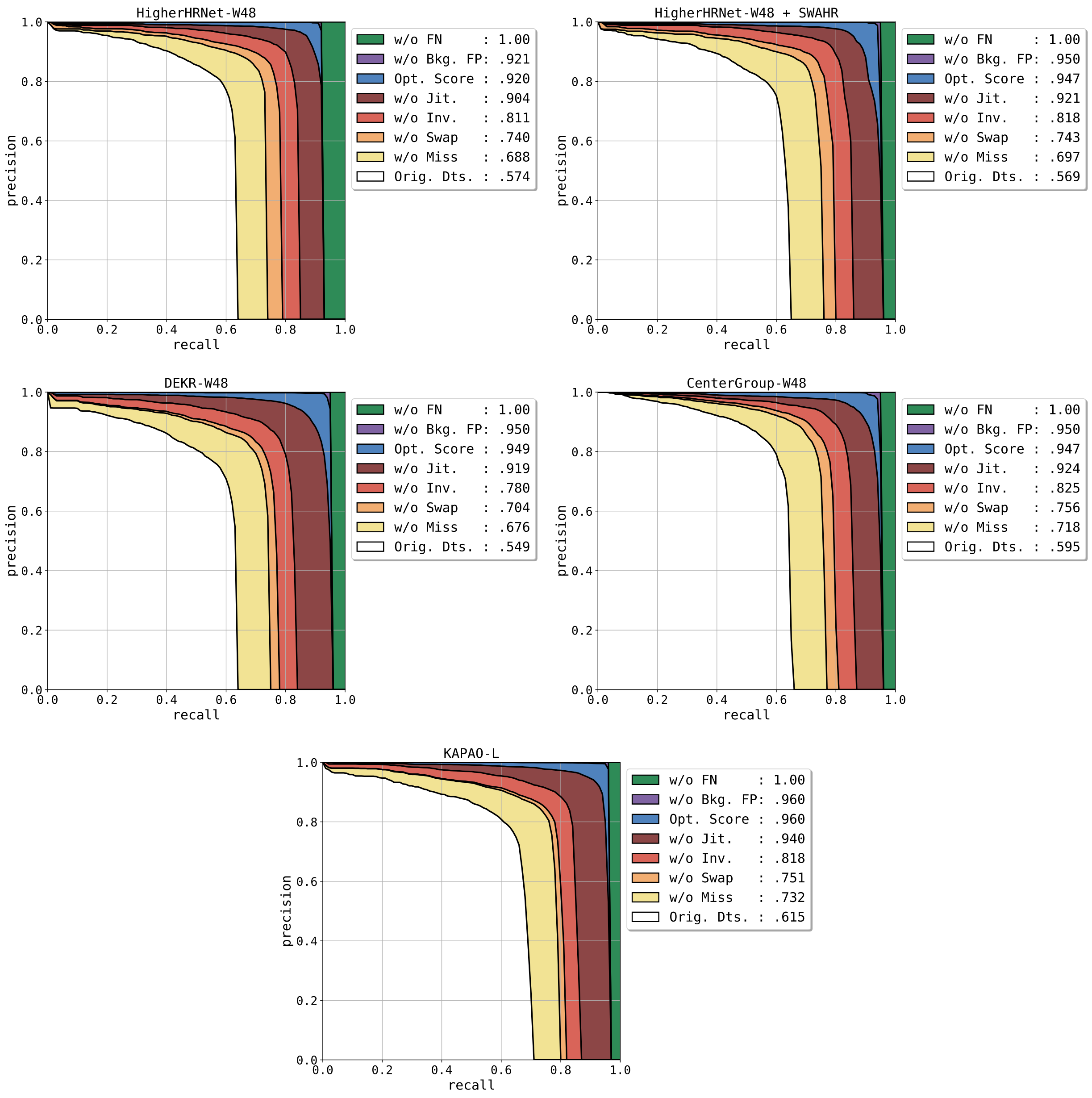


Figure 4.5: Error type analysis on COCO `val2017` for HigherHRNet-W48 [54], HigherHRNet-W48 + SWAHR [172], DEKR-W48 [56], CenterGroup-W48 [173], and KAPAO-L for an OKS threshold of 0.85 (without TTA). Plots generated using the `coco-analyze` toolbox [106]. AP$^{.85}$ values in legends given in decimals as opposed to percent.

CenterGroup-W48, and *vice versa*, the maximum OKS was found for each ground-truth instance (OKS_{max}) using the top-20 scoring pose detections for each model. For each validation image, the difference between the summations of the OKS maxima was computed:

$$\Delta\text{OKS} = \Sigma\text{OKS}_{max}^{KAPAO-L} - \Sigma\text{OKS}_{max}^{CenterGroup-W48} \tag{4.13}$$

Δ OKS is positive for images where KAPAO-L performs better than CenterGroup-W48. Conversely, Δ OKS is negative for images where CenterGroup-W48 performs better than KAPAO-L. To plot partial poses using KAPAO-L, the keypoint object confidence threshold τ_{ck} was lowered to 0.01 to promote the fusion of keypoint objects and increase the frequency of keypoint confidences in the predicted poses $\hat{\mathbf{P}}$ (see Section 4.2.4 for details). The τ_{ck} of 0.01 lowered the AP from 70.4 to 70.1. It also increased the post-processing time from approximately 3 ms to 5 ms per image due to the increased number of keypoint objects that are fused in Algorithm 2. Using these inference settings, KAPAO-L is still 1.0 AP more accurate and 3.0$\times$ faster than CenterGroup-W48 on `val2017`.

It is observed that extreme values of Δ OKS are associated with crowded images (> 20 people) containing a limited number of annotations (< 20 annotations). For these images, ΣOKS_{max} is contingent on whether the ground-truth instances are predicted by the top-20 scoring detections and therefore an element of chance is involved. Figure 4.6 illustrates such a scenario. The top-left image shows the ground-truth pose annotations (white); the top-right image shows the top-20 scoring CenterGroup-W48 detections (orange); the bottom-left image shows the top-20 scoring KAPAO-L detections (green); and the bottom-right image shows the same KAPAO-L detections but only plots the fused keypoint objects (light green). The top-20 KAPAO-L detections contain 8 of the 10 ground-truth instances whereas the top-20 CenterGroup-W48 predictions contain 6. As a result, Δ OKS = 2.06. Because all the COCO keypoint metrics are computed using the 20 top-scoring detections, false negatives are artificially inflated while true positives are artificially deflated. While it is perplexing that the COCO metrics possess an element of randomness, it is conceivable that over many images the randomness averages out and does not favour one model over another. Moreover, the COCO dataset is sparsely populated with crowded images so these rare cases likely have a negligible influence on AP/AR. The implications of only using 20 detections on datasets like CrowdPose may be more severe and worth investigating, however.

To avoid the aforementioned issues with comparing OKS in crowded scenes, the following comparisons consist of images where $\text{OKS}_{max} > 0.5$ for all ground-truth instances using both models. Figure 4.7 shows an example where KAPAO-L performs better than CenterGroup-W48 (Δ OKS = +0.68). The keypoint grouping module of CenterGroup-

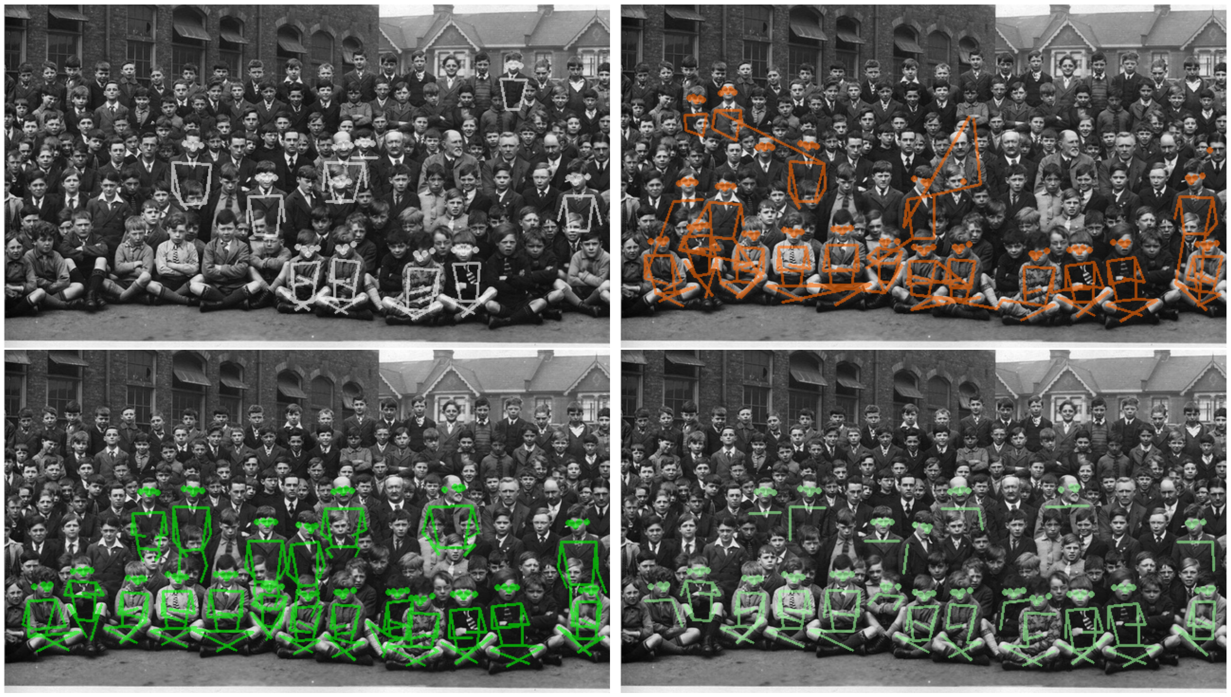


Figure 4.6: Qualitative comparison between KAPAO-L and CenterGroup-W48 (COCO image `24021`, $\Delta\,\mathrm{OKS} = 2.06$). Top-left: ground-truth. Top-right: top-20 scoring CenterGroup-W48 predictions. Bottom-left: top-20 scoring KAPAO-L predictions (all keypoints). Bottom-right: top-20 scoring KAPAO-L predictions (fused keypoint objects only).

W48 severely mixes up the keypoint identities (*swap* error). Swap error is a common failure case for CenterGroup but an uncommon failure case for KAPAO due to its detection of holistic pose objects (quantitative errors provided in Figure 4.5). Figure 4.8 shows the image with the lowest $\Delta\,\mathrm{OKS}$ (-0.66). For three of the ground-truth instances situated near the top of the frame, KAPAO predicts the locations of the nose, eyes, and ears significantly lower than the ground-truth locations, resulting in lower OKS for these poses. These errors are the result of the keypoint object bounding boxes being cut-off by the edge of the frame such that the center of the keypoint object bounding box no longer coincides with the actual keypoint locations. These errors could be rectified with relatively simple alterations to the inference code in future work.

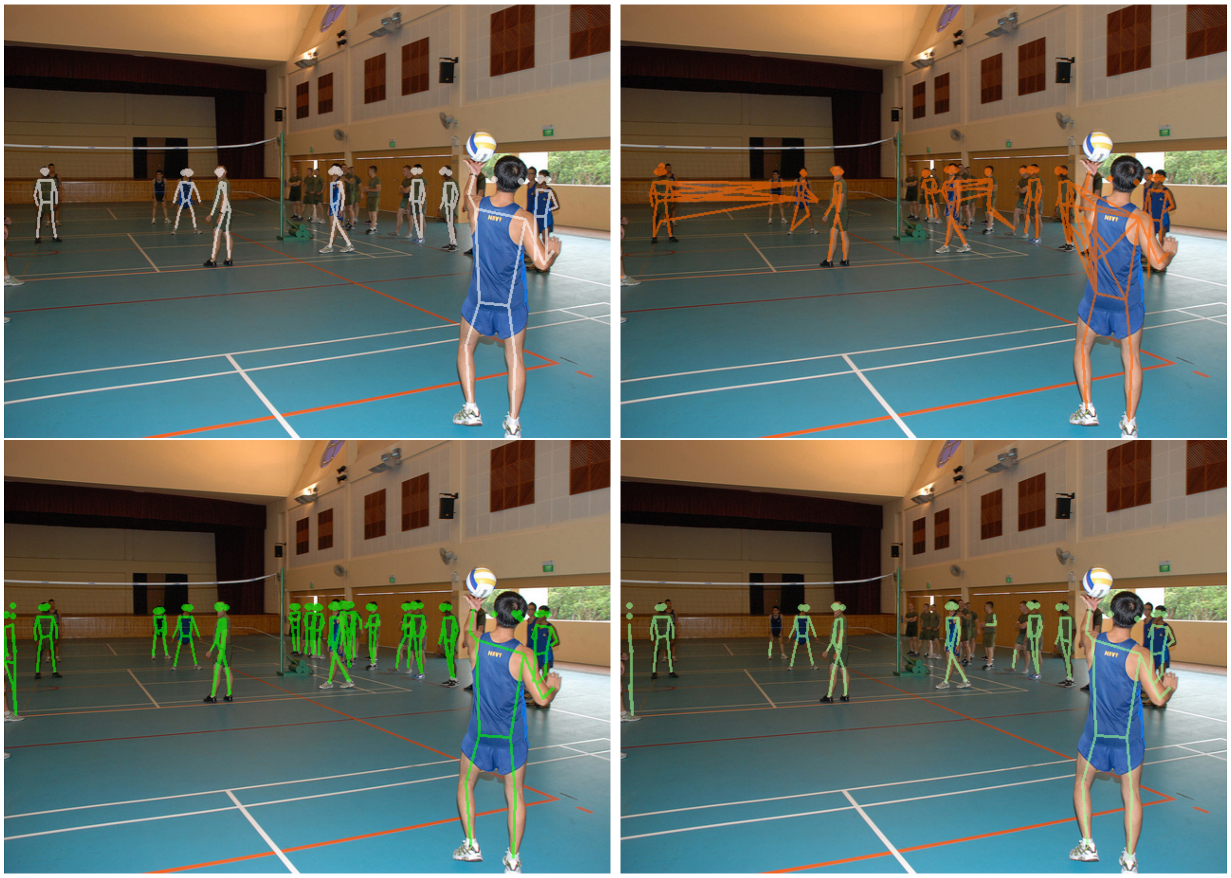


Figure 4.7: Qualitative comparison between KAPAO-L and CenterGroup-W48 (COCO image `49759`, Δ OKS $= +0.68$). Top-left: ground-truth. Top-right: top-20 scoring CenterGroup-W48 predictions. Bottom-left: top-20 scoring KAPAO-L predictions (all keypoints). Bottom-right: top-20 scoring KAPAO-L predictions (fused keypoint objects only).

4.3.5 Ablation Studies

The influence of the keypoint bounding box size b_s, one of KAPAO's important hyper-parameters, was empirically analyzed. Five KAPAO-S models were trained on COCO `train2017` for 50 epochs using normalized keypoint bounding box sizes $b_s/max(w,h) \in \{0.01, 0.025, 0.05, 0.075, 0.1\}$. The validation AP is plotted in Figure 4.9. The results are consistent with the prior work of McNally *et al.* [96]: $b_s/max(w,h) < 2.5\%$ destabilizes training leading to poor accuracy, and optimal $b_s/max(w,h)$ is observed around 5% (used for the experiments in Sections 4.3.1 and 4.3.2). In contrast to McNally *et al.*, the accuracy

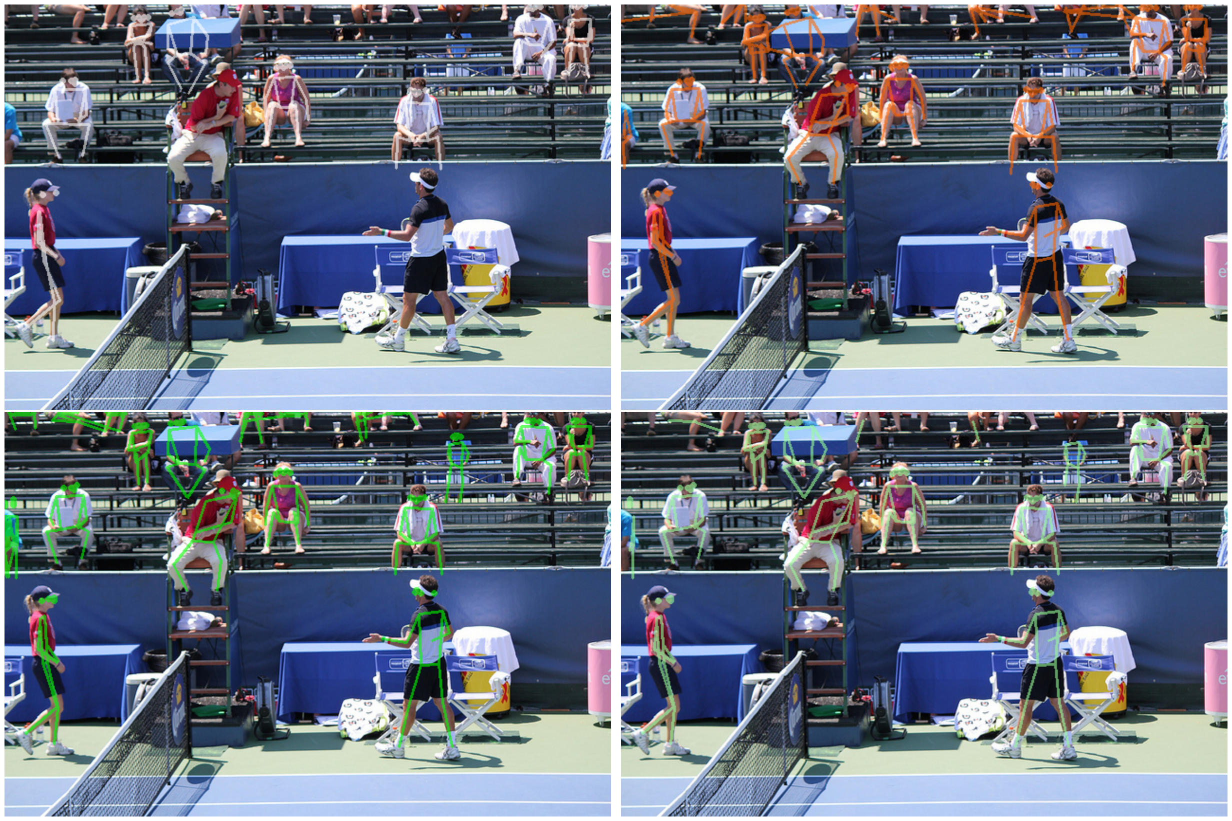


Figure 4.8: Qualitative comparison between KAPAO-L and CenterGroup-W48 (COCO image `326248`, Δ OKS $= -0.66$). Top-left: ground-truth. Top-right: top-20 scoring CenterGroup-W48 predictions. Bottom-left: top-20 scoring KAPAO-L predictions (all keypoints). Bottom-right: top-20 scoring KAPAO-L predictions (fused keypoint objects only).

in this study degrades quickly for $b_s/max(w, h) > 5\%$. It is hypothesized that large b_s in this application interferes with pose object learning.

The accuracy improvements resulting from fusing the keypoint objects with the pose objects are provided in Table 4.5. Keypoint object fusion adds no less than 1.0 AP and over 3.0 AP in some cases. Moreover, keypoint object fusion is fast; the added post-processing time per image is ≤ 1.7 ms on COCO and ≤ 4.5 ms on CrowdPose. Relative to the time required for the forward pass of the network (see Table 4.2), these are small increases.

The fusion of keypoint objects by class is also studied. Figure 4.10 plots the fusion rates for each keypoint type for KAPAO-S with no TTA on COCO `val2017`. The fusion rate is equal to the number of fused keypoint objects divided by the number of keypoints

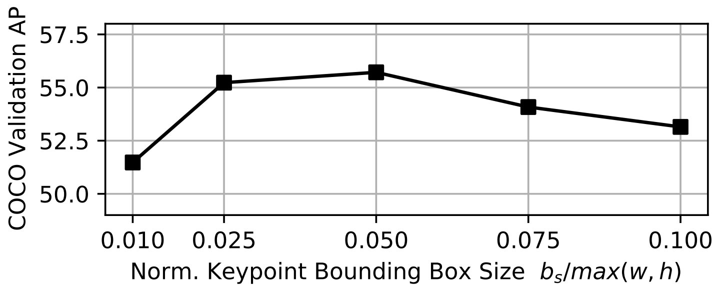


Figure 4.9: The influence of keypoint object bounding box size on learning. Each KAPAO-S model was trained for 50 epochs.

Method	TTA	Δ Lat. (ms) / ΔAP (COCO `val2017`)	Δ Lat. (ms) / ΔAP (CrowdPose `test`)
KAPAO-S	N	+1.2 / +2.4	+3.3 / +2.9
KAPAO-M	N	+1.2 / +1.1	+3.5 / +3.2
KAPAO-L	N	+1.7 / +1.2	+4.2 / +1.0
KAPAO-S	Y	+1.7 / +2.8	+3.9 / +3.2
KAPAO-M	Y	+1.6 / +1.5	+4.4 / +3.5
KAPAO-L	Y	+1.4 / +1.1	+4.5 / +1.0

Table 4.5: Accuracy improvement when fusing keypoint object detections with human pose detections. Latencies averaged over each dataset using a batch size of 1 on a TITAN Xp GPU.

of that type in the dataset. Because the number of human pose predictions is generally greater than the actual number of person instances in the dataset, the fusion rate can be greater than 1. As originally hypothesized, keypoints that are characterized by distinct local image features (*e.g.*, the eyes, ears, and nose) have higher fusion rates as they are detected more precisely as keypoint objects than as pose objects. Conversely, keypoints that require a more global understanding (*e.g.*, the hips) are better detected using pose objects, as evidenced by lower fusion rates.

Finally, the trade-off between accuracy and inference speed was investigated for various input sizes. The AP on COCO `val2017` was computed without TTA for $\max(w, h) \in \{640, 768, 896, 1024, 1152, 1280\}$. Figure 4.11 plots the results for each model. For all three models, reducing the input size to 1152 had a negligible effect on the accuracy but provided a meaningful latency reduction. For KAPAO-M and KAPAO-L, using an input size of 1024 reduced the accuracy marginally but also reduced the latency by ∼30%.

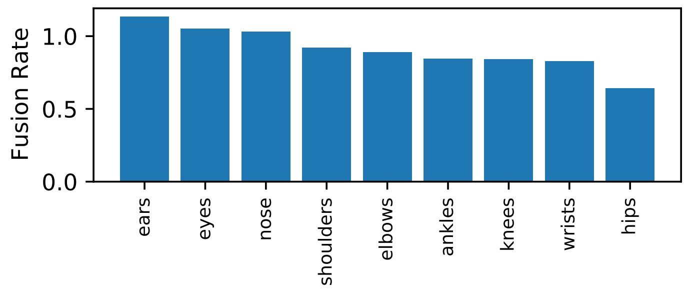


Figure 4.10: Keypoint object fusion rates for each keypoint type. Evaluated on COCO `val2017` using KAPAO-S without TTA.

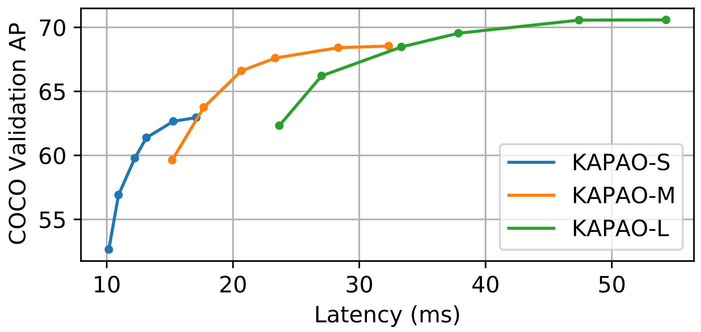


Figure 4.11: Accuracy-speed trade-off for input sizes ranging from 640 to 1280. Evaluated on COCO `val2017` using a batch size of 1 on a TITAN Xp GPU.

4.3.6 Video Inference Demos

The source code includes five video inference demos. The first four demos run inference on RGB video clips sourced from YouTube to demonstrate the practical use of KAPAO under various inference settings. The final demo demonstrates the generalization of KAPAO by running inference on a depth video converted to RGB. All reported inference speeds include *all* processing (*i.e.*, image loading, resizing, inference, graphics plotting, *etc.*).

Shuffle. KAPAO runs fastest on low resolution video with few people in the frame. This demo runs KAPAO-S on a single-person 480p dance video using an input size of 1024. The inference speed is ∼9.5 FPS on a workstation CPU (Intel Core i7-8700K), and ∼65 FPS on the TITAN Xp GPU. A screenshot of the inference is provided in Figure 4.12 (top-left).

Flash Mob. KAPAO-S was run on a 720p flash mob dance video using an input size of

1280. A screenshot of the inference is shown in Figure 4.12 (top-right). On a workstation housing a TITAN Xp GPU, the inference speed was ∼35 FPS.

Red Light, Green Light. This demo runs KAPAO-L on a 480p video clip from the TV show *Squid Game* using an input size of 1024. For this demo, incomplete poses were plotted based on keypoint confidence by setting $\tau_{ck} = 0.01$ (see Section 4.2.4 for details). A screenshot of the inference is provided in Figure 4.12 (bottom-left). The GPU inference speed varied between 15 and 30 FPS depending on the number of people in the frame.

Squash. KAPAO-S was run on a 1080p slow-motion squash video using an input size of 1280. A simple player tracking algorithm was implemented based on the frame-to-frame pose differences. The inference speed was ∼22 FPS on the TITAN Xp GPU. A screenshot is provided in Figure 4.12 (bottom-right).

Depth Videos. Finally, the robustness and generalization capabilities of KAPAO are demonstrated by running inference with KAPAO-S on depth videos obtained from a fencing action recognition dataset [266]. The depth information was converted to RGB format in a 480p video. A screenshot from the inference video, which ran at ∼60 FPS on the TITAN Xp GPU, is displayed in Figure 4.13. Despite the marked difference in appearance between the depth images and the KAPAO training images, human poses were still detected with high confidence. This interesting test result can be attributed to pose object representation learning, where spatial relations between human keypoints are learned using large-scale features and high-level context (*e.g.*, like the edges making up the human shape). This is further supported by the fact that fewer keypoint objects were detected in the depth images.[3]

4.4 Discussion

This chapter introduced KAPAO, a new heatmap-free keypoint estimation method based around modeling keypoints and poses as objects. A novel pose object representation was introduced that could prove to be useful in a variety of keypoint detection tasks where sets of spatially related object keypoints are of interest. It is concluded that KAPAO is effectively applied to the problem of single-stage multi-person human pose estimation by detecting *human* pose objects. Moreover, fusing jointly learned keypoint objects improved the accuracy of the predicted human poses with minimal computational overhead using

[3] Acknowledgement is given to Kevin Zhu for recognizing that KAPAO can detect pose objects in depth images.

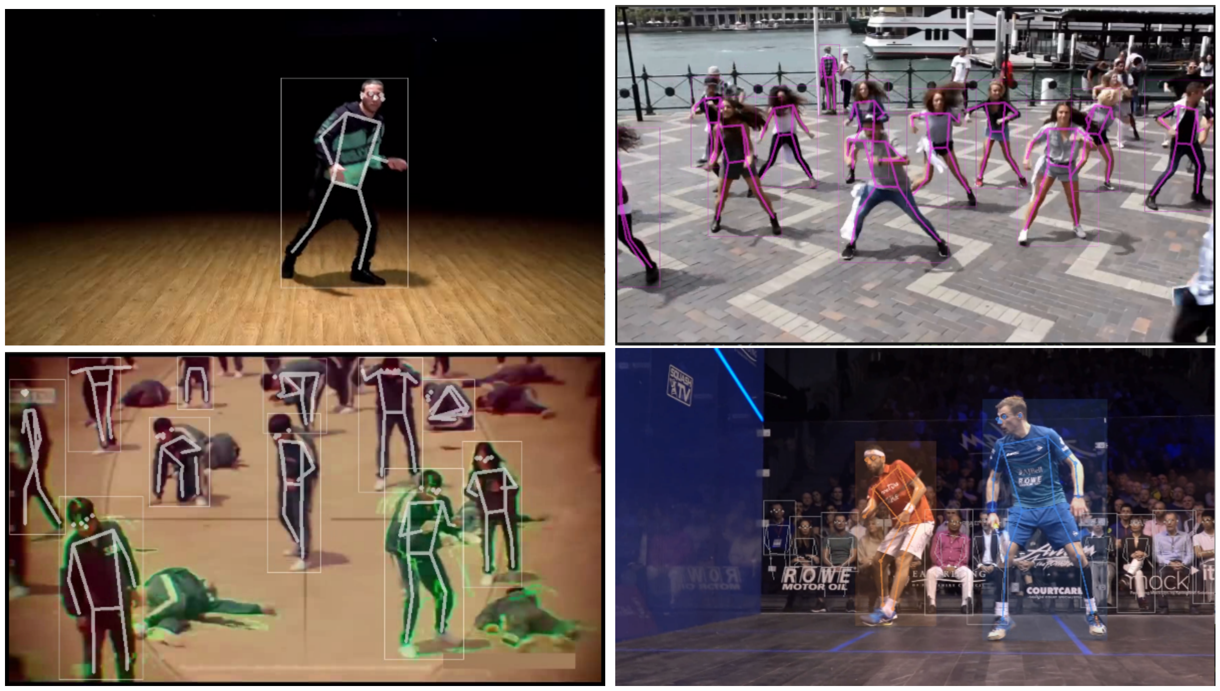


Figure 4.12: KAPAO video inference demo screenshots. Top-left: shuffling demo. Top-right: flash mob demo. Bottom-left: red light, green light demo. Bottom-right: squash demo. All video clips were sourced from YouTube.

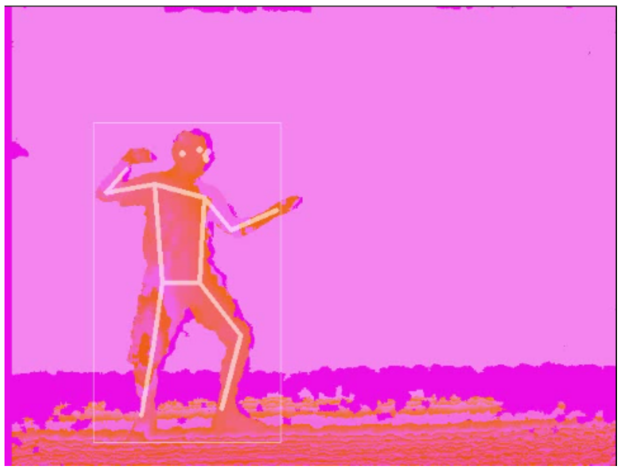


Figure 4.13: Pose objects generalize well and can even be detected in depth video. Shown here is a screenshot of KAPAO-S running inference on a depth video obtained from a fencing action recognition dataset [266].

a simple tolerance-based matching algorithm. As a result, KAPAO is significantly faster than previous single-stage methods, which are impeded by heatmap post-processing and bottom-up keypoint grouping. When not using test-time augmentation, KAPAO is considerably more accurate than the state-of-the-art. Moreover, because KAPAO learns full pose representations via pose objects, it is more robust to occlusion than bottom-up methods that detect individual keypoints. This was evidenced by achieving new state-of-the-art accuracy for a single-stage method on CrowdPose.

The six inference parameters may be viewed by some readers as an inconvenience. However, the parameters were quickly tuned for maximum accuracy using a coarse grid search, the accuracy is not overly sensitive to them, and further tuning is not required for practical usage. It is noted that the inference demos all use the same inference parameters. Moreover, if only pose objects are used, which marginally reduces the accuracy according to Table 4.5, four of the inference parameters become trivial. For comparison, HigherHRNet [54] has three inference parameters and DEKR [56] has seven. Some readers may also contend that the comparisons made in Table 4.2 are not fair comparisons because KAPAO uses a larger input size compared to HigherHRNet and DEKR (1280 versus 640). It is emphasized that in practical applications, accuracy and speed are the critical metrics. As such, fair comparisons are ones that use the same training data and the same hardware to measure speed. HigherHRNet and DEKR were specifically re-implemented on the same hardware as KAPAO to measure their speed and make fair comparisons. In essence, the computational efficiency afforded by KAPAO permits the use of larger input sizes while maintaining superior inference speed. If KAPAO was trained on an input size of 640 (*i.e.*, like HigherHRNet and DEKR), the accuracy would decrease slightly, but the latency would improve dramatically. Specifically, the latency of KAPAO-L would decrease by 50% to 24 ms (comparable to the speed of KAPAO-S). The input size of 1280 was chosen to prioritize accuracy in this trade-off scenario since KAPAO is already fast enough for real-time inference.

Compared to EvoPose2D, KAPAO has a simpler implementation as it does not require pairing with a person detector. The requirement of using a person detector represents a marked source of inefficiency. Exacerbating the issue, to maximize accuracy, two-stage approaches typically use the most accurate person detectors available, which are large CNNs that carry a significant computational burden. Preprocessed detections (provided in [51]) were used to make fair comparisons in the EvoPose2D evaluations; however, the original person detector was cited as being a Faster R-CNN [157] object detection network. It was not disclosed what specific variant of Faster R-CNN was used (*i.e.*, what backbone), but it was found that a Faster R-CNN with a ResNeXt-101 [267] FPN [185] backbone matched the reported AP of the one used in [51]. The `detectron2` API [268] was used to implement this network and it was found to have a forward pass time of 396 ms on the TITAN Xp GPU (measured over COCO `val2017` using a batch size of 1). The AP for the person category was evaluated to be 56.6. This particular person detector is already 7× slower than KAPAO-L (without TTA, see Table 4.2 for details). If a faster person detector was used, it would likely be less accurate, which would degrade the overall accuracy of the two-stage human pose estimation approach. Notably, Faster R-CNN with a regular ResNet-101 FPN backbone provides a more favourable accuracy-speed trade-off (84 ms inference and person AP of 55.7). Still, it is slower than KAPAO-L even before considering the extra computation required by EvoPose2D. Depending on the number of people detected, EvoPose2D may need to be run multiple times (batched inference could be exploited, but would ultimately be limited by GPU memory).

These insights help put the computational advantages of KAPAO and other single-stage methods into perspective. However, in accuracy-critical applications or scenarios with large computational budgets (*e.g.*, when processing is performed offline on multi-GPU workstations or servers), EvoPose2D is still the recommended approach to maximize accuracy. Alternatively, if using cloud GPU instances, KAPAO may be preferred to minimize cloud computing costs. KAPAO may also be preferred in speed-critical applications or when computational resources are limited, such as on edge devices. If a potential application is constrained to a single person centered in the frame, using EvoPose2D without a person detector may be the preferred approach. However, doing so would require retraining EvoPose2D using entire images as input instead of cropped person detections.

Chapter 5

Conclusion

This thesis research presented two novel human pose estimation models, one for each of the multi-person estimation paradigms (*i.e.*, two-stage and single-stage). Motivated by the success of neural architecture search in image recognition, a novel accelerated neuroevolution framework was introduced in Chapter 3 and was leveraged in the machine-aided design of EvoPose2D, a parameter-efficient heatmap-based single-person human pose estimation network. Recognizing the limitations of heatmaps and the drawbacks of two-stage inference, new keypoint representations were introduced in Chapter 4 and an entirely new approach to single-stage human pose estimation was engineered around these representations. At the time of development, both methods improved upon the state of the art in their respective categories on the standard multi-person human pose estimation benchmarks. EvoPose2D is recommended in accuracy-critical applications with large computational budgets, whereas KAPAO is recommended in speed-critical applications or when fewer computational resources are available.

The developed models contribute to the overall advancement of the field of human pose estimation and in turn, make progress towards the highly anticipated use of markerless motion capture in the wild. The following sections summarize the research contributions and give direction relating to potential paths for future work.

5.1 Summary of Contributions

The research contributions contained in this thesis include:

1. A method for accelerating the training of human pose estimation networks using large batch sizes (up to a batch size of 2048) and adaptive learning rate optimizers, including supporting experiments demonstrating no loss of generalization.

2. A simple and flexible weight transfer scheme for accelerating neuroevolution that relaxes the complete function-preservation constraint imposed by previous weight transfer methods.

3. A neural architectural search space for the design of efficient single-person heatmap-based human pose estimation networks, including the required mapping for network encoding / decoding.

4. An accelerated neuroevolution framework, including large-scale neuroevolution experiments supporting the efficacy of the proposed weight transfer scheme.

5. EvoPose2D, heatmap-based human pose estimator for two-stage multi-person estimation that was designed via neuroevolution and achieves state-of-the-art accuracy when scaled.

6. A novel keypoint representation that models individual keypoints as objects using small keypoint bounding boxes.

7. A novel pose object representation that extends the conventional object representation to include a set of keypoints.

8. A method for jointly learning keypoint objects and pose objects using a dense anchor-based detection network and a multi-task loss.

9. KAPAO, a state-of-the-art hand-designed single-stage human pose estimator that simultaneously detects human pose objects and keypoint objects and fuses the detections to exploit the strengths of both object representations.

10. Extensive empirical analysis of KAPAO, including a thorough error analysis and ablation experiments demonstrating the efficacy of keypoint-pose object fusion and the benefits of both object representations.

11. A comprehensive evaluation of the accuracies, errors, and inference speeds of KAPAO and other state-of-the-art single-stage methods HigherHRNet [54], HigherHRNet + SWAHR [172], DEKR [56], and CenterGroup [173] using the same hardware.

5.2 Future Work

There are many potential applications that are centered around the use of this research in the wild (*e.g.*, biomechanical analysis at home, in a gym, on a golf course, *etc.*). It would therefore be desirable to deploy the developed algorithms on mobile devices (*i.e.*, on smartphones and tablets). While such an undertaking is not particularly research-oriented, a significant amount of engineering would be required to successfully deploy the developed models on iOS and Android devices, including converting the models to TFLite or CoreML format, quantizing the models to accelerate inference and reduce storage requirements, and building iOS or Android applications around their desired use. Furthermore, evaluating the inference speeds of the developed models on edge device hardware is of interest, especially considering the influence that hardware selection has on the relative speeds of CNNs (see discussion in Section 4.4). Extending the inference pipeline to perform monocular 3D human pose estimation via pairing with a 2D-to-3D pose lifting network may also be interest. The implementations and datasets used in this thesis focus on frame-by-frame inference, so there is also an opportunity to develop methods for video estimation that produce more temporally consistent results.

Given the distinct computational advantages of the single-stage approach, it is recommended that future research continue in this direction. Specific recommendations for future research therefore relate to the KAPAO model and are discussed below.

Architectural Optimization for KAPAO

KAPAO uses a highly efficient YOLO-style CNN architecture that has been iteratively improved over several years of research and engineering. However, its design improvements have been centered around the task of object detection, not human pose estimation. Object detection on COCO involves detecting 80 unique object classes. The YOLO architecture design has therefore been tailored to accommodate a large number of object classes using numerous detection grids and anchor boxes. As a result, the architecture may be unnecessarily complex for human pose estimation. With KAPAO, there are only two object classes of concern: keypoint objects and pose objects. Furthermore, all keypoint objects have the same bounding box size, which supports the use of fewer anchor boxes. Evidently, there exists an opportunity to optimize the architecture for human pose estimation. It is recommended that the accelerated neuroevolution method proposed in Chapter 3 be used for the architectural optimization using a new search space built around KAPAO's current design. Additionally, a more thorough investigation into the balancing of the multi-task loss weighting is recommended. The current results suggest that the loss favours pose object detection over keypoint localization, as evidenced by higher overall AR and higher AP at lower OKS thresholds.

Pose Objects with Keypoint-wise Confidence

As discussed in Section 4.2.4, one of the main limitations of KAPAO is that the current pose object definition does not include keypoint-wise confidences. Instead, a single confidence, given by $\hat{p}_o \cdot \max(\hat{\mathbf{c}})$, is used to represent the confidence of the entire pose object. This creates an issue when people appear partially in the frame (*i.e.*, around the edges) as there is no way of knowing which keypoints in the pose object actually exist in the frame. A temporary solution involves lowering the keypoint object confidence threshold τ_{ck} to be near zero such that only keypoint objects are used (see inference demos in Section 4.3.6 for more details). While this approach is reasonably effective, it slows down inference and degrades the accuracy as a result of using keypoint objects only. A more prudent solution is recommended. The suggested approach involves extending the pose object representation to include keypoint-wise confidences that can be learned using a binary cross-entropy loss and the ground-truth keypoint visibility flags provided with the human pose estimation dataset. Not only will this solution help with predicting partial poses, but it may also increase the AP by using more authentic pose detection scoring (see Section 4.3.3 for details on the influence of pose detection scoring on AP). The main challenge with this modification is that another loss component would need to be added to the multi-task loss defined in Section 4.2.2 and it would have to be balanced against the existing loss components. In terms of the network architecture, the number of output channels N_o would increase by K, but this alteration would have a negligible effect on the computational complexity.

Jointly Learning Human Segmentation Masks

Multi-task learning, or joint learning, can improve generalization using the domain information contained in the training signals of related tasks as an inductive bias [269]. Its benefits have been demonstrated in recent computer vision applications employing CNNs [270, 271, 272, 273], as well as countless other works. Instance segmentation annotations are readily available in the COCO dataset [60], so there is an opportunity to jointly learn human segmentation masks in addition to human pose. The learning of human segmentation masks may provide an inductive bias that improves learning efficiency and the prediction accuracy of human poses. To jointly learn human segmentation masks using KAPAO, the human pose object could be extended to include an additional set of keypoints corresponding to the edges of the person segmentation. This approach for representing segmentation masks was adopted in Point-Set Anchors [165]; however, multi-task learning was not exploited. The architectural modifications required for the implementation would have a negligible effect on the computational complexity. Similar to learning keypoint-wise confidences for pose objects, another loss component would need to be added to the overall multi-task loss and it would have to be balanced against the existing loss components. An appropriate initial guess for the segmentation loss weight is the same value as used for the weight of the pose object keypoint loss.

Pose-based Non-maximum Suppression

In cases where people appear in severe overlap (*e.g.*, two people hugging, one person standing behind another), their predicted pose object bounding boxes would be overlapping as well. In the current implementation of KAPAO, if the amount of overlap falls above the NMS threshold τ_{bp}, the pose object bounding box with the highest confidence is kept while the other is discarded. On the surface, this functionality appears to be a flaw rather than a feature. However, NMS is required to handle duplicate detections that occur due to the redundancy measures built into KAPAO's architecture (see Section 4.2.1 for details). An opportunity is presented to improve the NMS algorithm using pose object keypoint information. For example, when the bounding boxes of two people are severely overlapping, it is likely that their poses will still be unique, and that information should be exploited to determine whether the pose objects represent unique people or duplicate detections. Just as OKS is used as a substitute similarity measure for IoU in the computation of AP, OKS could also serve as a similarity measure between pose predictions in an advanced pose-based NMS algorithm. Similar techniques have been exploited in previous studies [44, 159, 48, 56].

Keypoint Object Fusion using Self-Attention

The experiments provided in Section 4.3.5 empirically demonstrated that keypoint objects are better suited for detecting keypoints with distinct local features (*e.g.*, eyes, ears, etc.). The keypoint object fusion method defined in Algorithm 2 is based on the simple idea of matching keypoint objects to the nearest pose object keypoint of the same class. The nearest pose object keypoint is overwritten by the keypoint object center if its proximity falls within τ_{fd} pixels. While this distance-based matching algorithm proved to be reliable for increasing the accuracy of the predicted human poses (see improvements reported in Table 4.5), there is much room for improvement. The proposed algorithm can fail when pose object keypoints are inaccurate, or when keypoints from multiple pose objects exist in close proximity to one another. More complex matching algorithms should be considered for future iterations. Specifically, a deep learning-based approach could be used to machine-learn the correct matching directly from the data. The self-attention layers used in Transformers [34] model relations between entities in a global context. Hence, self-attention represents a strong candidate for modeling the relations between keypoint and pose object detections. Moreover, there is a precedence for using self-attention in single-stage human pose estimation. CenterGroup [173] used multi-attention heads to associate keypoints with pose centers in a heatmap-based approach. While the attention-based grouping module provided significant accuracy improvements, it was slower than the CNN used to produce the heatmaps themselves. An opportunity presents itself to develop an efficient attention-based grouping module within the heatmap-free single-stage paradigm.

Two-stage Recurrent Inference using Shared Weights

It is well-documented that two-stage human pose estimation methods are more accurate than single-stage methods. In accuracy-critical scenarios or scenarios where computational cost is less critical (*e.g.*, offline processing on high-powered multi-GPU workstations), two-stage methods reign supreme. Because KAPAO additionally detects person bounding boxes by virtue of pose objects, a unique opportunity presents itself: KAPAO could be configured to run in two-stage mode using recurrent inference and shared weights. In essence, a first pass would be used to obtain the person bounding boxes. The detected bounding boxes would then be cropped from the original image, resized, and fed to KAPAO a second time to predict the final poses. KAPAO could potentially be configured to run this way without further training, providing a distinct advantage over traditional two-stage methods (*i.e.*, only one model needs to be trained instead of two). Additionally, the use of recurrent inference reduces model storage requirements because the weights are shared by each stage. Initial exploration into this idea is provided in the source code.

References

[1] T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Computer Vision and Image Understanding*, vol. 104, no. 2-3, pp. 90–126, 2006.

[2] Z. Wu, Y. Li, and R. J. Radke, "Viewpoint invariant human re-identification in camera networks using pose priors and subject-discriminative features," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 5, pp. 1095–1108, 2014.

[3] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, 2013.

[4] K. Pullen and C. Bregler, "Motion capture assisted animation: Texturing and synthesis," in *PACMCGIT*, 2002.

[5] M.-Y. Wu, P.-W. Ting, Y.-H. Tang, E.-T. Chou, and L.-C. Fu, "Hand pose estimation in object-interaction based on deep learning for virtual reality applications," *Journal of Visual Communication and Image Representation*, vol. 70, p. 102802, 2020.

[6] C. Bregler, "Motion capture technology for entertainment [in the spotlight]," *IEEE Signal Processing Magazine*, vol. 24, no. 6, pp. 160–158, 2007.

[7] A. Pfister, A. M. West, S. Bronner, and J. A. Noah, "Comparative abilities of microsoft kinect and vicon 3d motion capture for gait analysis," *Journal of Medical Engineering & Technology*, vol. 38, no. 5, pp. 274–280, 2014.

[8] Y. Chu, T. C. Sell, and S. M. Lephart, "The relationship between biomechanical variables and driving performance during the golf swing," *Journal of Sports Sciences*, vol. 28, no. 11, pp. 1251–1259, 2010.

[9] M. Fani, H. Neher, D. A. Clausi, A. Wong, and J. S. Zelek, “Hockey action recognition via integrated stacked hourglass network.,” in *CVPRW*, 2017.

[10] Z. Cai, H. Neher, K. Vats, D. A. Clausi, and J. Zelek, “Temporal hockey action recognition via pose and optical flows,” in *CVPRW*, 2019.

[11] K. Yamane and J. Hodgins, “Simultaneous tracking and balancing of humanoid robots for imitating human motion capture data,” in *IROS*, 2009.

[12] M. W. Mehrez, K. Worthmann, J. P. Cenerini, M. Osman, W. W. Melek, and S. Jeon, “Model predictive control without terminal constraints or costs for holonomic mobile robots,” *Robotics and Autonomous Systems*, vol. 127, p. 103468, 2020.

[13] S. Kim and M. A. Nussbaum, “Performance evaluation of a wearable inertial motion capture system for capturing physical exposures during manual material handling tasks,” *Ergonomics*, vol. 56, no. 2, pp. 314–326, 2013.

[14] CNET, “Behind the scenes at an NBA 2K17 motion capture session.” `https://www.youtube.com/watch?v=yyGgKcRbQIU`. Accessed September 2019.

[15] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *CVPR*, 2005.

[16] D. G. Lowe, “Object recognition from local scale-invariant features,” in *CVPR*, 1999.

[17] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient matching of pictorial structures,” in *CVPR*, 2000.

[18] P. F. Felzenszwalb and D. P. Huttenlocher, “Pictorial structures for object recognition,” *International Journal of Computer Vision*, vol. 61, no. 1, pp. 55–79, 2005.

[19] L. Pishchulin, M. Andriluka, P. Gehler, and B. Schiele, “Poselet conditioned pictorial structures,” in *CVPR*, 2013.

[20] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[21] Y. LeCun, Y. Bengio, *et al.*, “Convolutional networks for images, speech, and time series,” *The Handbook of Brain Theory and Neural Networks*, vol. 3361, no. 10, 1995.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *NeurIPS*, 2012.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *CVPR*, 2009.

[24] R. C. Gonzalez, R. E. Woods, and B. R. Masters, “Digital image processing,” 2009.

[25] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *ICLR*, 2015.

[26] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[27] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *CVPR*, 2016.

[28] J. Schmidhuber, U. Meier, and D. Ciresan, “Multi-column deep neural networks for image classification,” in *CVPR*, 2012.

[29] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *ICCV*, 2015.

[30] C. Lu and X. Tang, “Surpassing human-level face verification performance on lfw with gaussianface,” in *AAAI*, 2015.

[31] X. Zhang, H. Luo, X. Fan, W. Xiang, Y. Sun, Q. Xiao, W. Jiang, C. Zhang, and J. Sun, “Alignedreid: Surpassing human-level performance in person re-identification,” *arXiv preprint arXiv:1711.08184*, 2017.

[32] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, “Dermatologist-level classification of skin cancer with deep neural networks,” *Nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[33] A. Buetti-Dinh, V. Galli, S. Bellenberg, O. Ilie, M. Herold, S. Christel, M. Boretska, I. V. Pivkin, P. Wilmes, W. Sand, *et al.*, “Deep neural networks outperform human expert’s capacity in characterizing bioleaching bacterial biofilm composition,” *Biotechnology Reports*, vol. 22, 2019.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *NeurIPS*, 2017.

[35] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *ICLR*, 2021.

[36] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," *ICCV*, 2021.

[37] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," *arXiv preprint arXiv:2201.03545*, 2022.

[38] C. Zheng, W. Wu, T. Yang, S. Zhu, C. Chen, R. Liu, J. Shen, N. Kehtarnavaz, and M. Shah, "Deep learning-based human pose estimation: A survey," *arXiv preprint arXiv:2012.13392*, 2020.

[39] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2d human pose estimation: New benchmark and state of the art analysis," in *CVPR*, 2014.

[40] A. Toshev and C. Szegedy, "DeepPose: Human pose estimation via deep neural networks," in *CVPR*, 2014.

[41] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in *NeurIPS*, 2014.

[42] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *CVPR*, 2016.

[43] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *ECCV*, 2016.

[44] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu, "RMPE: Regional multi-person pose estimation," in *ICCV*, 2017.

[45] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *CVPR*, 2017.

[46] A. Newell, Z. Huang, and J. Deng, "Associative embedding: End-to-end learning for joint detection and grouping," in *NeurIPS*, 2017.

[47] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *ICCV*, 2017.

[48] G. Papandreou, T. Zhu, L.-C. Chen, S. Gidaris, J. Tompson, and K. Murphy, "Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model," in *ECCV*, 2018.

[49] M. Kocabas, S. Karagoz, and E. Akbas, "MultiPoseNet: Fast multi-person pose estimation using pose residual network," in *ECCV*, 2018.

[50] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun, "Cascaded pyramid network for multi-person pose estimation," in *CVPR*, 2018.

[51] B. Xiao, H. Wu, and Y. Wei, "Simple baselines for human pose estimation and tracking," in *ECCV*, 2018.

[52] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *CVPR*, 2019.

[53] W. Li, Z. Wang, B. Yin, Q. Peng, Y. Du, T. Xiao, G. Yu, H. Lu, Y. Wei, and J. Sun, "Rethinking on multi-stage networks for human pose estimation," *arXiv preprint arXiv:1901.00148*, 2019.

[54] B. Cheng, B. Xiao, J. Wang, H. Shi, T. S. Huang, and L. Zhang, "HigherHRNet: Scale-aware representation learning for bottom-up human pose estimation," in *CVPR*, 2020.

[55] F. Zhang, X. Zhu, H. Dai, M. Ye, and C. Zhu, "Distribution-aware coordinate representation for human pose estimation," in *CVPR*, 2020.

[56] Z. Geng, K. Sun, B. Xiao, Z. Zhang, and J. Wang, "Bottom-up human pose estimation via disentangled keypoint regression," in *CVPR*, 2021.

[57] W. McNally, K. Vats, A. Wong, and J. McPhee, "EvoPose2D: Pushing the boundaries of 2d human pose estimation using accelerated neuroevolution with weight transfer," *IEEE Access*, 2021.

[58] W. McNally, K. Vats, A. Wong, and J. McPhee, "Rethinking keypoint representations: Modeling keypoints and poses as objects for multi-person human pose estimation," *arXiv preprint arXiv:2111.08557*, 2021.

[59] R. Khirodkar, V. Chari, A. Agrawal, and A. Tyagi, "Multi-hypothesis pose networks: Rethinking top-down pose estimation," in *ICCV*, 2021.

[60] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *ECCV*, 2014.

[61] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1325–1339, 2013.

[62] L. Sigal, A. O. Balan, and M. J. Black, "Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion," *International Journal of Computer Vision*, vol. 87, no. 1-2, p. 4, 2010.

[63] D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu, and C. Theobalt, "Monocular 3d human pose estimation in the wild using improved CNN supervision," in *3DV*.

[64] S. Li and A. B. Chan, "3d human pose estimation from monocular images with deep convolutional neural network," in *ACCV*, 2014.

[65] X. Zhou, M. Zhu, S. Leonardos, K. G. Derpanis, and K. Daniilidis, "Sparseness meets deepness: 3d human pose estimation from monocular video," in *CVPR*, 2016.

[66] S. Park, J. Hwang, and N. Kwak, "3d human pose estimation using convolutional neural networks with 2d pose information," in *ECCV*, 2016.

[67] J. Martinez, R. Hossain, J. Romero, and J. J. Little, "A simple yet effective baseline for 3d human pose estimation," in *ICCV*, 2017.

[68] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis, "Coarse-to-fine volumetric prediction for single-image 3d human pose," in *CVPR*, 2017.

[69] B. Tekin, P. Márquez-Neila, M. Salzmann, and P. Fua, "Learning to fuse 2d and 3d image cues for monocular body pose estimation," in *ICCV*, 2017.

[70] C.-H. Chen and D. Ramanan, "3d human pose estimation = 2d pose estimation + matching," in *CVPR*, 2017.

[71] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt, "Vnect: Real-time 3d human pose estimation with a single RGB camera," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 1–14, 2017.

[72] X. Zhou, Q. Huang, X. Sun, X. Xue, and Y. Wei, "Towards 3d human pose estimation in the wild: a weakly-supervised approach," in *ICCV*, 2017.

[73] M. R. I. Hossain and J. J. Little, “Exploiting temporal information for 3d human pose estimation,” in *ECCV*, 2018.

[74] W. Yang, W. Ouyang, X. Wang, J. Ren, H. Li, and X. Wang, “3d human pose estimation in the wild by adversarial learning,” in *CVPR*, 2018.

[75] D. Pavllo, C. Feichtenhofer, D. Grangier, and M. Auli, “3d human pose estimation in video with temporal convolutions and semi-supervised training,” in *CVPR*, 2019.

[76] H. Qiu, C. Wang, J. Wang, N. Wang, and W. Zeng, “Cross view fusion for 3d human pose estimation,” in *ICCV*, 2019.

[77] Y. Cheng, B. Yang, B. Wang, and R. T. Tan, “3d human pose estimation using spatio-temporal networks with explicit occlusion training,” in *AAAI*, 2020.

[78] R. Liu, J. Shen, H. Wang, C. Chen, S.-c. Cheung, and V. Asari, “Attention mechanism exploits temporal contexts: Real-time 3d human pose reconstruction,” in *CVPR*, 2020.

[79] Y. He, R. Yan, K. Fragkiadaki, and S.-I. Yu, “Epipolar transformers,” in *CVPR*, 2020.

[80] F. Huang, A. Zeng, M. Liu, Q. Lai, and Q. Xu, “DeepFuse: An IMU-aware network for real-time 3d human pose estimation from multi-view image,” in *WACV*, 2020.

[81] A. Llopart, “Liftformer: 3d human pose estimation using attention models,” *arXiv preprint arXiv:2009.00348*, 2020.

[82] W. Li, H. Liu, R. Ding, M. Liu, P. Wang, and W. Yang, “Exploiting temporal contexts with strided transformer for 3d human pose estimation,” *arXiv preprint arXiv:2103.14304*, 2021.

[83] C. Zheng, S. Zhu, M. Mendieta, T. Yang, C. Chen, and Z. Ding, “3d human pose estimation with spatial and temporal transformers,” *arXiv preprint arXiv:2103.10455*, 2021.

[84] K. Gong, J. Zhang, and J. Feng, “Poseaug: A differentiable pose augmentation framework for 3d human pose estimation,” in *CVPR*, 2021.

[85] A. Bouazizi, U. Kressel, and V. Belagiannis, “Learning temporal 3d human pose estimation with pseudo-labels,” in *AVSS*, 2021.

[86] N. D. Reddy, L. Guigues, L. Pishchulin, J. Eledath, and S. G. Narasimhan, "Tesse-track: End-to-end learnable multi-person articulated 3d pose tracking," in *CVPR*, 2021.

[87] K. Lee, I. Lee, and S. Lee, "Propagating LSTM: 3d pose estimation based on joint interdependency," in *ECCV*, 2018.

[88] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2 ed., 2004.

[89] K. Iskakov, E. Burkov, V. Lempitsky, and Y. Malkov, "Learnable triangulation of human pose," in *ICCV*, 2019.

[90] R. Reed, "Pruning algorithms-a survey," *IEEE Transactions on Neural Networks*, vol. 4, no. 5, pp. 740–747, 1993.

[91] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless cnns with low-precision weights," in *ICLR*, 2017.

[92] J. Van Leeuwen, "On the construction of huffman trees," in *ICALP*, 1976.

[93] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[94] T. Wei, C. Wang, Y. Rui, and C. W. Chen, "Network morphism," in *ICML*, 2016.

[95] M. Wistuba, "Deep learning architecture search by neuro-cell-based evolution with function-preserving mutations," in *ECML PKDD*, 2018.

[96] W. McNally, P. Walters, K. Vats, A. Wong, and J. McPhee, "DeepDarts: Modeling keypoints as objects for automatic scorekeeping in darts using a single camera," in *CVPRW*, 2021.

[97] Y. Li, S. Yang, S. Zhang, Z. Wang, W. Yang, S.-T. Xia, and E. Zhou, "Is 2d heatmap representation even necessary for human pose estimation?," *arXiv preprint arXiv:2107.03332*, 2021.

[98] S. Johnson and M. Everingham, "Clustered pose and nonlinear appearance models for human pose estimation.," in *BMVC*, 2010.

[99] B. Sapp and B. Taskar, "Modec: Multimodal decomposable models for human pose estimation," in *CVPR*, 2013.

[100] S. Johnson and M. Everingham, "Learning effective human pose estimation from inaccurate annotation," in *CVPR*, 2011.

[101] M. Andriluka, U. Iqbal, E. Insafutdinov, L. Pishchulin, A. Milan, J. Gall, and B. Schiele, "Posetrack: A benchmark for human pose estimation and tracking," in *CVPR*, 2018.

[102] J. Wu, H. Zheng, B. Zhao, Y. Li, B. Yan, R. Liang, W. Wang, S. Zhou, G. Lin, Y. Fu, *et al.*, "Ai challenger: A large-scale dataset for going deeper in image understanding," *arXiv preprint arXiv:1711.06475*, 2017.

[103] J. Li, C. Wang, H. Zhu, Y. Mao, H.-S. Fang, and C. Lu, "Crowdpose: Efficient crowded scenes pose estimation and a new benchmark," in *CVPR*, 2019.

[104] S. Jin, L. Xu, J. Xu, C. Wang, W. Liu, C. Qian, W. Ouyang, and P. Luo, "Whole-body human pose estimation in the wild," in *ECCV*, 2020.

[105] R. Padilla, W. L. Passos, T. L. Dias, S. L. Netto, and E. A. da Silva, "A comparative analysis of object detection metrics with a companion open-source toolkit," *Electronics*, vol. 10, no. 3, p. 279, 2021.

[106] M. R. Ronchi and P. Perona, "Benchmarking and error diagnosis in multi-instance pose estimation," in *ICCV*, 2017.

[107] P. Li, H. Zhao, P. Liu, and F. Cao, "RTM3d: Real-time monocular 3d detection from object keypoints for autonomous driving," in *ECCV*, 2020.

[108] U. Iqbal, P. Molchanov, T. Breuel Juergen Gall, and J. Kautz, "Hand pose estimation via latent 2.5 d heatmap regression," in *ECCV*, 2018.

[109] W. Huang, P. Ren, J. Wang, Q. Qi, and H. Sun, "Awr: Adaptive weighting regression for 3d hand pose estimation," in *AAAI*, 2020.

[110] W. McNally, A. Wong, and J. McPhee, "Action recognition using deep convolutional neural networks and compressed spatio-temporal pose encodings," *Journal of Computational Vision and Imaging Systems*, vol. 4, no. 1, pp. 3–3, 2018.

[111] W. McNally, A. Wong, and J. McPhee, "STAR-Net: Action recognition using spatio-temporal activation reprojection," in *CRV*, 2019.

[112] K. Gavrilyuk, R. Sanford, M. Javan, and C. G. Snoek, "Actor-transformers for group activity recognition," in *CVPR*, 2020.

[113] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019.

[114] Y. Raaj, H. Idrees, G. Hidalgo, and Y. Sheikh, "Efficient online multi-person 2d pose tracking with recurrent spatio-temporal affinity fields," in *CVPR*, 2019.

[115] X. Dong, Y. Yan, W. Ouyang, and Y. Yang, "Style aggregated network for facial landmark detection," in *CVPR*, 2018.

[116] X. Wang, L. Bo, and L. Fuxin, "Adaptive wing loss for robust face alignment via heatmap regression," in *ICCV*, 2019.

[117] Z. Xu, B. Li, Y. Yuan, and M. Geng, "AnchorFace: An anchor-based facial landmark detector across large poses," in *AAAI*, 2021.

[118] S. Suwajanakorn, N. Snavely, J. Tompson, and M. Norouzi, "Discovery of latent 3d keypoints via end-to-end geometric reasoning," in *NeurIPS*, 2018.

[119] T. Jakab, A. Gupta, H. Bilen, and A. Vedaldi, "Unsupervised learning of object landmarks through conditional image generation," in *NeurIPS*, 2018.

[120] R. Voeikov, N. Falaleev, and R. Baikulov, "Ttnet: Real-time temporal and spatial video analysis of table tennis," in *CVPRW*, 2020.

[121] G. Chéron, I. Laptev, and C. Schmid, "P-CNN: Pose-based cnn features for action recognition," in *ICCV*, 2015.

[122] E. Insafutdinov, M. Andriluka, L. Pishchulin, S. Tang, E. Levinkov, B. Andres, and B. Schiele, "ArtTrack: Articulated multi-person tracking in the wild," in *CVPR*, 2017.

[123] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *IEEE Transactions on Computers*, no. 1, pp. 67–92, 1973.

[124] G. Hinton, "Using relaxation to find a puppet," in *Proceedings of the 2nd Summer Conference on Artificial Intelligence and Simulation of Behaviour*, pp. 148–157, 1976.

[125] S. Ioffe and D. Forsyth, "Human tracking with mixtures of trees," in *ICCV*, 2001.

[126] R. Ronfard, C. Schmid, and B. Triggs, “Learning to parse pictures of people,” in *ECCV*, 2002.

[127] D. Ramanan, “Learning to parse images of articulated bodies,” in *NeurIPS*, 2006.

[128] D. Ramanan and C. Sminchisescu, “Training deformable models for localization,” in *CVPR*, 2006.

[129] V. Ferrari, M. Marin-Jimenez, and A. Zisserman, “Progressive search space reduction for human pose estimation,” in *CVPR*, 2008.

[130] M. Eichner, V. Ferrari, and S. Zurich, “Better appearance models for pictorial structures.,” in *BMVC*, 2009.

[131] M. Andriluka, S. Roth, and B. Schiele, “Pictorial structures revisited: People detection and articulated pose estimation,” in *CVPR*, 2009.

[132] Y. Yang and D. Ramanan, “Articulated pose estimation with flexible mixtures-of-parts,” in *CVPR*, 2011.

[133] Y. Wang, D. Tran, and Z. Liao, “Learning hierarchical poselets for human parsing,” in *CVPR*, 2011.

[134] K. Duan, D. Batra, and D. J. Crandall, “A multi-layer composite model for human pose estimation.,” in *BMVC*, 2012.

[135] L. Pishchulin, A. Jain, M. Andriluka, T. Thormählen, and B. Schiele, “Articulated people detection and pose estimation: Reshaping the future,” in *CVPR*, 2012.

[136] Y. Tian, C. L. Zitnick, and S. G. Narasimhan, “Exploring the spatial hierarchy of mixture models for human pose estimation,” in *ECCV*, 2012.

[137] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[138] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[139] L. Bourdev and J. Malik, “Poselets: Body part detectors trained using 3d human pose annotations,” in *ICCV*, 2009.

[140] M. Weiler, F. A. Hamprecht, and M. Storath, “Learning steerable filters for rotation equivariant cnns,” in *CVPR*, 2018.

[141] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *CVPR*, 2015.

[142] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, “Human pose estimation with iterative error feedback,” in *CVPR*, 2016.

[143] P. Hu and D. Ramanan, “Bottom-up and top-down reasoning with hierarchical rectified gaussians,” in *CVPR*, 2016.

[144] U. Rafi, B. Leibe, J. Gall, and I. Kostrikov, “An efficient convolutional network for human pose estimation.,” in *BMVC*, 2016.

[145] A. Bulat and G. Tzimiropoulos, “Human pose estimation via convolutional part heatmap regression,” in *ECCV*, 2016.

[146] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, “Efficient object localization using convolutional networks,” in *CVPR*, 2015.

[147] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *MICCAI*, 2015.

[148] X. Chu, W. Yang, W. Ouyang, C. Ma, A. L. Yuille, and X. Wang, “Multi-context attention for human pose estimation,” in *CVPR*, 2017.

[149] W. Yang, S. Li, W. Ouyang, H. Li, and X. Wang, “Learning feature pyramids for human pose estimation,” in *ICCV*, 2017.

[150] G. Ning, Z. Zhang, and Z. He, “Knowledge-guided deep fractal neural networks for human pose estimation,” *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1246–1259, 2017.

[151] Y. Chen, C. Shen, X.-S. Wei, L. Liu, and J. Yang, “Adversarial posenet: A structure-aware convolutional network for human pose estimation,” in *ICCV*, 2017.

[152] C.-J. Chou, J.-T. Chien, and H.-T. Chen, “Self adversarial training for human pose estimation,” in *APSIPA ASC*, 2018.

[153] L. Ke, M.-C. Chang, H. Qi, and S. Lyu, “Multi-scale structure-aware network for human pose estimation,” in *ECCV*, 2018.

[154] Z. Tang, X. Peng, S. Geng, L. Wu, S. Zhang, and D. Metaxas, "Quantized densely connected u-nets for efficient landmark localization," in *ECCV*, 2018.

[155] S. Kreiss, L. Bertoni, and A. Alahi, "Pifpaf: Composite fields for human pose estimation," in *CVPR*, 2019.

[156] X. Nie, J. Feng, J. Zhang, and S. Yan, "Single-stage multi-person pose machines," in *ICCV*, 2019.

[157] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," 2015.

[158] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[159] G. Papandreou, T. Zhu, N. Kanazawa, A. Toshev, J. Tompson, C. Bregler, and K. Murphy, "Towards accurate multi-person pose estimation in the wild," in *CVPR*, 2017.

[160] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *CVPR*, 2017.

[161] S.-H. Zhang, R. Li, X. Dong, P. Rosin, Z. Cai, X. Han, D. Yang, H. Huang, and S.-M. Hu, "Pose2seg: Detection free human instance segmentation," in *CVPR*, 2019.

[162] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, and B. Schiele, "DeepCut: Joint subset partition and labeling for multi person pose estimation," in *CVPR*, 2016.

[163] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "DeeperCut: A deeper, stronger, and faster multi-person pose estimation model," in *ECCV*, 2016.

[164] U. Iqbal and J. Gall, "Multi-person pose estimation with local joint-to-person associations," in *ECCV*, 2016.

[165] F. Wei, X. Sun, H. Li, J. Wang, and S. Lin, "Point-set anchors for object detection, instance segmentation and pose estimation," in *ECCV*, 2020.

[166] W. Mao, Z. Tian, X. Wang, and C. Shen, "Fcpose: Fully convolutional multi-person pose estimation with dynamic instance-aware convolutions," in *CVPR*, 2021.

[167] E. D. Demaine, D. Emanuel, A. Fiat, and N. Immorlica, "Correlation clustering in general weighted graphs," *Theoretical Computer Science*, vol. 361, no. 2-3, pp. 172–187, 2006.

[168] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "Openpose: real-time multi-person 2d pose estimation using part affinity fields," *arXiv preprint arXiv:1812.08008*, 2018.

[169] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[170] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?," in *NeurIPS*, 2017.

[171] S. Jin, W. Liu, E. Xie, W. Wang, C. Qian, W. Ouyang, and P. Luo, "Differentiable hierarchical graph grouping for multi-person pose estimation," in *ECCV*, 2020.

[172] Z. Luo, Z. Wang, Y. Huang, L. Wang, T. Tan, and E. Zhou, "Rethinking the heatmap regression for bottom-up human pose estimation," in *CVPR*, 2021.

[173] G. Brasó, N. Kister, and L. Leal-Taixé, "The center of attention: Center-keypoint grouping via attention for multi-person pose estimation," in *ICCV*, 2021.

[174] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *ICCV*, 2017.

[175] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: Fully convolutional one-stage object detection," in *ICCV*, 2019.

[176] X. Jia, B. De Brabandere, T. Tuytelaars, and L. V. Gool, "Dynamic filter networks," *NeurIPS*, 2016.

[177] H. Law and J. Deng, "CornerNet: Detecting objects as paired keypoints," in *ECCV*, 2018.

[178] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "CenterNet: Keypoint triplets for object detection," in *ICCV*, 2019.

[179] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *CVPR*, 2001.

[180] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *ICIP*, 2002.

[181] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009.

[182] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018.

[183] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *CVPR*, 2018.

[184] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *CVPR*, 2020.

[185] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "NAS-FPN: Learning scalable feature pyramid architecture for object detection," in *CVPR*, 2019.

[186] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[187] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.

[188] R. Girshick, "Fast R-CNN," in *ICCV*, 2015.

[189] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," *NeurIPS*, 2016.

[190] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving into high quality object detection," in *CVPR*, 2018.

[191] J. Pang, K. Chen, J. Shi, H. Feng, W. Ouyang, and D. Lin, "Libra R-CNN: Towards balanced learning for object detection," in *CVPR*, 2019.

[192] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[193] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, June 2010.

[194] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[195] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *CVPR*, 2006.

[196] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016.

[197] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *CVPR*, 2017.

[198] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Scaled-YOLOv4: Scaling cross stage partial network," *arXiv preprint arXiv:2011.08036*, 2020.

[199] G. J. et. al., "ultralytics/yolov5: v5.0," Apr. 2021.

[200] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *ECCV*, 2016.

[201] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *CVPR*, 2019.

[202] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *CVPR*, 2017.

[203] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[204] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[205] A. Gholami, K. Kwon, B. Wu, Z. Tai, X. Yue, P. Jin, S. Zhao, and K. Keutzer, "Squeezenext: Hardware-aware neural network design," in *CVPRW*, 2018.

[206] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *CVPR*, 2018.

[207] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *ECCV*, 2018.

[208] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *ICLR*, 2017.

[209] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *ICLR*, 2017.

[210] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *arXiv preprint arXiv:1808.05377*, 2018.

[211] M. Wistuba, A. Rawat, and T. Pedapati, "A survey on neural architecture search," *arXiv preprint arXiv:1905.01392*, 2019.

[212] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *CVPR*, 2019.

[213] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *CVPR*, 2018.

[214] A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, University of Toronto, 2009.

[215] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *ICML*, 2017.

[216] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *AAAI*, 2019.

[217] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "A downsampled variant of imagenet as an alternative to the cifar datasets," *arXiv preprint arXiv:1707.08819*, 2017.

[218] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *GECCO*, 2017.

[219] T. Domhan, J. T. Springenberg, and F. Hutter, "Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves," in *IJCAI*, 2015.

[220] B. Baker, O. Gupta, R. Raskar, and N. Naik, "Accelerating neural architecture search using performance prediction," *arXiv preprint arXiv:1705.10823*, 2017.

[221] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *ECCV*, 2018.

[222] M. Wistuba and T. Pedapati, "Inductive transfer for neural architecture optimization," *arXiv preprint arXiv:1903.03536*, 2019.

[223] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *ICML*, 2019.

[224] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[225] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *CVPR*, 2018.

[226] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," in *PLMR*, 2018.

[227] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[228] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *ICLR*, 2019.

[229] L. Xie and A. Yuille, "Genetic CNN," in *ICCV*, 2017.

[230] T. Elsken, J.-H. Metzen, and F. Hutter, "Simple and efficient architecture search for convolutional neural networks," *arXiv preprint arXiv:1711.04528*, 2017.

[231] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," in *ICLR*, 2019.

[232] K. Yu, C. Sciuto, M. Jaggi, C. Musat, and M. Salzmann, "Evaluating the search phase of neural architecture search," in *ICLR*, 2020.

[233] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "NAS-Bench-101: Towards reproducible neural architecture search," in *ICML*, 2019.

[234] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *ICML*, 2015.

[235] J. Li, S. Bian, A. Zeng, C. Wang, B. Pang, W. Liu, and C. Lu, "Human pose regression with residual log-likelihood estimation," in *ICCV*, 2021.

[236] K. Li, S. Wang, X. Zhang, Y. Xu, W. Xu, and Z. Tu, "Pose recognition with cascade transformers," in *CVPR*, 2021.

[237] H. Zhang, L. Wang, S. Jun, N. Imamura, Y. Fujii, and H. Kobashi, "CPNAS: Cascaded pyramid network via neural architecture search for multi-person pose estimation," in *CVPRW*, 2020.

[238] W. Zhang, J. Fang, X. Wang, and W. Liu, "Efficientpose: Efficient human pose estimation with neural architecture search," *Computational Visual Media*, vol. 7, no. 3, pp. 335–347, 2021.

[239] S. Yang, W. Yang, and Z. Cui, "Pose neural fabrics search," *arXiv preprint arXiv:1909.07068*, 2019.

[240] X. Gong, W. Chen, Y. Jiang, Y. Yuan, X. Liu, Q. Zhang, Y. Li, and Z. Wang, "AutoPose: Searching multi-scale branch aggregation for pose estimation," *arXiv preprint arXiv:2008.07018*, 2020.

[241] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, "Auto-DeepLab: Hierarchical neural architecture search for semantic image segmentation," in *CVPR*, 2019.

[242] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, "Designing neural networks through neuroevolution," *Nature Machine Intelligence*, vol. 1, no. 1, pp. 24–35, 2019.

[243] L.-C. Chen, M. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens, "Searching for efficient multi-scale architectures for dense image prediction," in *NeurIPS*, 2018.

[244] V. Nekrasov, H. Chen, C. Shen, and I. Reid, "Fast neural architecture search of compact semantic segmentation models via auxiliary cells," in *CVPR*, 2019.

[245] Z. Zhu, C. Liu, D. Yang, A. Yuille, and D. Xu, "V-NAS: Neural architecture search for volumetric medical image segmentation," in *3DV*, 2019.

[246] B. Yan, H. Peng, K. Wu, D. Wang, J. Fu, and H. Lu, "LightTrack: Finding lightweight neural networks for object tracking via one-shot architecture search," in *CVPR*, 2021.

[247] Y. Bin, X. Cao, X. Chen, Y. Ge, Y. Tai, C. Wang, J. Li, F. Huang, C. Gao, and N. Sang, "Adversarial semantic data augmentation for human pose estimation," in *ECCV*, 2020.

[248] G. Moon, J. Y. Chang, and K. M. Lee, “Posefix: Model-agnostic general human pose refinement network,” in *CVPR*, 2019.

[249] J. Huang, Z. Zhu, F. Guo, and G. Huang, “The devil is in the details: Delving into unbiased data processing for human pose estimation,” in *CVPR*, 2020.

[250] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.

[251] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *CVPR*, 2018.

[252] E. Hoffer, I. Hubara, and D. Soudry, “Train longer, generalize better: closing the generalization gap in large batch training of neural networks,” in *NeurIPS*, 2017.

[253] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, “On large-batch training for deep learning: Generalization gap and sharp minima,” in *ICLR*, 2017.

[254] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, “Accurate, large minibatch sgd: Training imagenet in 1 hour,” *arXiv preprint arXiv:1706.02677*, 2017.

[255] I. Loshchilov and F. Hutter, “SGDR: Stochastic gradient descent with warm restarts,” in *ICLR*, 2017.

[256] Y. Ci, C. Lin, M. Sun, B. Chen, H. Zhang, and W. Ouyang, “Evolving search space for neural architecture search,” in *ICCV*, 2021.

[257] N. Wang, S. XIANG, C. Pan, *et al.*, “You only search once: Single shot neural architecture search via direct sparse optimization,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[258] A. Wan, X. Dai, P. Zhang, Z. He, Y. Tian, S. Xie, B. Wu, M. Yu, T. Xu, K. Chen, *et al.*, “Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions,” in *CVPR*, 2020.

[259] A. Canziani, A. Paszke, and E. Culurciello, “An analysis of deep neural network models for practical applications,” *arXiv preprint arXiv:1605.07678*, 2016.

[260] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, “RepVGG: Making VGG-style convnets great again,” in *CVPR*, 2021.

[261] B. Laschowski, W. McNally, A. Wong, and J. McPhee, “Environment classification for robotic leg prostheses and exoskeletons using deep convolutional neural networks,” *Frontiers in Neurorobotics*, vol. 15, p. 730965, 2022.

[262] S. Jeon, D. Min, S. Kim, and K. Sohn, “Joint learning of semantic alignment and object landmark detection,” in *ICCV*, 2019.

[263] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, “Cspnet: A new backbone that can enhance learning capability of cnn,” in *CVPR*, 2020.

[264] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, “Distance-IoU loss: Faster and better learning for bounding box regression,” in *AAAI*, 2020.

[265] Y. Nesterov, “A method for solving the convex programming problem with convergence rate o(1/k2),” *Proceedings of the USSR Academy of Sciences*, vol. 269, pp. 543–547, 1983.

[266] F. Malawski and B. Kwolek, “Classification of basic footwork in fencing using accelerometer,” in *Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, 2016.

[267] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in *CVPR*, 2017.

[268] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, “Detectron2.” https://github.com/facebookresearch/detectron2, 2019.

[269] R. Caruana, “Multitask learning,” *Machine Learning*, vol. 28, no. 1, pp. 41–75, 1997.

[270] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, “Cross-stitch networks for multitask learning,” in *CVPR*, 2016.

[271] I. Kokkinos, “Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory,” in *CVPR*, 2017.

[272] D. C. Luvizon, D. Picard, and H. Tabia, “2d/3d pose estimation and action recognition using multitask deep learning,” in *CVPR*, 2018.

[273] T. D. Le, D. T. Huynh, and H. V. Pham, “Efficient human-robot interaction using deep learning with Mask R-CNN: detection, recognition, tracking and segmentation,” in *ICARCV*, 2018.

[274] S. Yang, Z. Quan, M. Nie, and W. Yang, “Transpose: Keypoint localization via transformer,” in *ICCV*, 2021.

[275] M. Dantone, J. Gall, C. Leistner, and L. Van Gool, “Human pose estimation using body parts dependent joint regressors,” in *CVPR*, 2013.