3M-POSE: MULTI-RESOLUTION, MULTI-PATH AND MULTI-OUTPUT NEURAL ARCHITECTURE SEARCH FOR BOTTOM-UP POSE PREDICTION

A Thesis

by

DUC HOANG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2021

Major Subject: Computer Science

ABSTRACT

Human pose estimation is a challenging computer vision task and often hinges on carefully handcrafted architectures.

This paper aims to be the first to apply Neural Architectural Search (NAS) to automatically design a bottom-up, one-stage human pose estimation model with significantly lower computational costs and smaller model size than existing bottom-up approaches. Our framework dubbed 3M-Pose co-searches and co-trains with the novel building block of Early Escape Layers (EELs), producing native modular architectures that are optimized to support dynamic inference for even lower average computational cost. To flexibly explore the fine-grained spectrum between the performance and computational budget, we propose Dynamic Ensemble Gumbel Softmax (Dyn-EGS), a novel approach to sample micro and macro search spaces by allowing varying numbers of operators and inputs to be individually selected for each cell. We additionally enforce a computational constraint with a student-teacher guidance to avoid the trivial search collapse caused by the pursuit of lightweight models. Experiments demonstrate 3M-Pose to find models of drastically superior speed and efficiency compared to existing works, reducing computational costs by up to 93% and parameter size by up to 75% at the cost of minor loss in performance.

ACKNOWLEDGMENTS

I would like to express gratitude to those who has helped me as I worked on this thesis. This work could not have been possible without the guidance of my advisor, Dr. Tracy Hammond, thank you for giving me the opportunity, and for providing with constant supports when I needed it.

Thank you to my committee member, Dr. Zhangyang Wang for your advice has been utmost critical.

Finally, I would like to thanks the members of Sketch Recognition, especially Samantha Ray, for providing me a place of encouragement, and always being available to help.

TABLE OF CONTENTS

Page

ABSTRACT ii

ACKNOWLEDGMENTS iii

TABLE OF CONTENTS iv

LIST OF FIGURES vi

LIST OF TABLES vii

1. INTRODUCTION 1

  1.1 Human Pose Estimation 1
    1.1.1 Top-Down Pose Estimation 1
    1.1.2 Bottom-Up Pose Estimation 1
  1.2 Neural Architectural Search 2
  1.3 Proposed Work 3
  1.4 Summary 5

2. PRIOR AND RELATED WORK 7

  2.1 Recent Methods in top-down pose estimation 7
    2.1.1 Traditional Approaches 7
    2.1.2 NAS Approaches 7
  2.2 Recent Methods in bottom-up pose estimation 8
  2.3 Recent Methods in Neural Architectural Search 9
  2.4 Recent Methods in Dynamic Inference 10

3. METHODOLOGY 11

  3.1 3M-Pose 11
    3.1.1 Pose Neural Fabric Search 11
    3.1.2 Fuse-Split block 11
    3.1.3 Early Escape Layers (EELS) 12
    3.1.4 Dynamic Inference 13
  3.2 Search Spaces 14
    3.2.1 Gumbel-Softmax trick 14

3.2.2 Micro Search Space ... 15
3.2.3 Macro Search Space ... 16
3.3 Dynamic Ensemble Gumbel Softmax (Dyn-EGS) ... 16
3.4 Supervision ... 17

4. EXPERIMENTS ... 20

4.1 COCO Keypoint Detection ... 20
4.2 Ablation Study ... 23

5. CONCLUSION, LIMITATION, AND DISCUSSION ... 25

REFERENCES ... 27

APPENDIX A. ... 32

A.1 Resulting Architecture ... 32

LIST OF FIGURES

FIGURE Page

1.1 An illustration of 3M-Pose. The framework is based on PNFS [1] with cells densely connected in a fabric-like grid. Each EEL has what we call a fuse-split stage, where features are aggregated, outputted, and then finally redistributed back into the the features stream. See Figure 3.2, and Section 3.1.2 for details. Note that the intermediate output is also used as the skip features to the next EEL or to the last layer. 4

3.1 An illustration of of our micro search space. This figure includes our input, output, and hidden stages. 12

3.2 An illustration of our fuse-split stage with two scale factors. Note that input features into FUSE are forwarded to SPLIT, this behavior is not shown in Figure 1.1 for simplicity. 14

3.3 An illustration of different architecture sampling methods: a) densely connected cells, b) *argmax* discrete architecture, c) DARTS-EGS [2] with $r = 2$, d) Our Dynamic Ensemble Gumbel-Softmax. Note that while the cells in (a), (b), and (c) have a static architecture, our cells' architectures can vary from cell to cell. 16

3.4 We demonstrate our model's performance on obtained data from publicly available COCO dataset [3]. The colored skeletal structure is drawn using predicted keypoints from 3M-Pose. 19

A.1 An illustration of the searched architecture at the macro level. We simplify the Fuse-Split block as a dashed line for better clarity. Each cell is marked with a number that will correspond to table A.1. 32

LIST OF TABLES

TABLE Page

4.1 Comparison with hand-crafted bottom-up and top-down NAS methods on COCO2017 validation set. GFLOPs are calculated at single scale. Note that we are using static-inference on the last layer to report our results. (*) the GFLOP per person detected bounding boxes and not image sample. [1] indicates using refinement. 20

4.2 Ablation study of 3M-Pose's components on COCO2017 validation dataset. tea+flop: GFLOP constraint and student-teacher loss. FS, fuse-split module 20

4.3 Ablation study comparing performance, and computational cost at different layers, versus using dynamic inference 21

A.1 Our searched architecture break down by cell's position corresponding to figure A.1. $O^0_{s,l}$ is the operator between I and $h^1_{s,l}$, $O^1_{s,l}$ is between I and $h^2_{s,l}$, and $O^2_{s,l}$ is between $h^1_{s,l}$ and $h^2_{s,l}$. SE stands for Squeeze and Excite. CSI stands for Conv-Sigmoid * Identity. SepConv is Depth-wise Separable Convolution. 33

1. INTRODUCTION

1.1 Human Pose Estimation

Human Pose Estimation (HPE) is one of the fundamental tasks in computer vision that serves as the foundation for many subfields such as activity recognition, medical AI research, and interactive entertainment. Higher accuracy in HPE enables a better understanding of human behaviors, human actions, and human anatomies. Current works in the field over-emphasize improving performance without consideration for the size of the models. HPE applications often need run on resource-constrained platforms such as mobile and handheld devices, and need to recognize from image streams with low latency or even in nearly real time.

1.1.1 Top-Down Pose Estimation

High performing HPE DNNs are all 2-stage top-down (TD) approaches which rely on two networks: one to detect people in bounding box, the other to estimate human pose from that detection. Some of these algorithms can reach well over 35 billion floating point operations (FLOP) per person [4]. It is common to find DNN algorithms with over 60 million parameters [4]. Accordingly, the energy cost of TD is astronomical and there is a disconnect between their prohibitive energy cost and the energy capability of many resource-constrained IoT devices. Therefore to close the gap between power and performance in pose detection, there is a need for an efficiency-focused ultra-lightweight HPE framework with comparable performance to SOTA.

1.1.2 Bottom-Up Pose Estimation

The bottom-up (BU) or one-stage approaches [5, 6, 7, 8, 9, 10, 11] utilize a single network for both keypoint detection and identity grouping. This type of approach is ideal for real-time estimation. However, while BU's computational cost is not proportional to the number of people, it can be expensive (405.5 GFLOP [5]) and the network itself can be large (277.8M parameters [10]). Scale invariance is a unique problem for BU and poses two challenges in predicting keypoints for small persons. One is dealing with scale variation by improving prediction while not harming prediction of large persons. The other is to generate quality heatmaps for precise localization of keypoints of small persons. There are various solutions for scale invariance, e.g., using feature pyramids, direct image upsampling, and outputting outputs of different scales. Solving this specific challenge while remaining conscious of efficiency is a daunting task to do manually.

In addition, since BU expenses the same amount of GFLOPs for five persons or for one, there is efficiency to be gained if there is a way to let easy samples out early without having to traverse the entire model. As such, there is a need for models that support dynamic inference to avoid those unnecessary computational costs.

1.2 Neural Architectural Search

Neural Architectural Search (NAS) [12, 13, 1, 14, 15, 16, 17, 18, 19, 2, 20, 21, 22, 23, 2, 24] is capable of generating task-specific model comparable to those that are hand-crafted. There have been but a few pose NAS works [1, 12] for TD and none for BU. But existence of prior works in TD show that it is possible to extend NAS beyond simple classification and onto complex dense prediction tasks. However, as seen later in Table 4.3, a naive implementation of NAS in BU resulted in a design with unimpressive results. A large search space and limited sampling capability mean that NAS is often ineffective without additional guidance and constraints to provide support.

Furthermore, when generating the final architecture after search, we find the choice between generating either a dense [1] or sparse architecture [12] to be limiting. A densely connected architecture is computationally inefficient and physically impossible to fit on a

GPU in our case. On the other hand, a sparse network is efficient, but it does not have enough parameters to represent the data effectively. Recent work by [2], proposed DART-EGS, a differentiable method to sample *r* number operators in a differentiable manner. However, DART-EGS is static in nature, and thus cannot react to either performance or computational needs during the architecture search.

1.3 Proposed Work

To this end, we propose 3M-Pose, a *multi-resolution*, *multi-paths*, and *multi-output* NAS for BU pose prediction to generate an efficient-orientated BU HPE framework.

Multi-resolution Scale variation is the main challenge we face designing BU approach. We use Pose Neural Fabrics Search (PNFS) [1] as the base of our framework to facilitate multi-resolution macro and micro search spaces, which we denote as the fabric and cell respectively (see Figure 1.1).

Multi-path We propose Dynamic Ensemble Gumbel-Softmax (Dyn-EGS), a new method for architecture sampling to facilitate multi-paths searching (see Figure 3.3). It controls the network's density for both the micro and macro search spaces. Dyn-EGS dynamically determines the number of operations per edge and the number of inputs per cell to optimize between performance and computational budget. By allowing cells to have heterogeneous number of operators and inputs, 3M-Pose is able to generate high performing architecture with a small computational footprint.

Multi-output: Motivated to gain additional efficiency with mulitple-outputs, we implement EELs along the fabric depth. During train and search, the features from these layers are fed through our Fuse-Split module (FS) (see Figure 3.2), to become an additional output along with the final layer's. Since we co-train and co-search with EELs, they are sub-models that can work in concert or used independently for improved computational costs without extra effort. To be used in concert, we design a gating mechanism

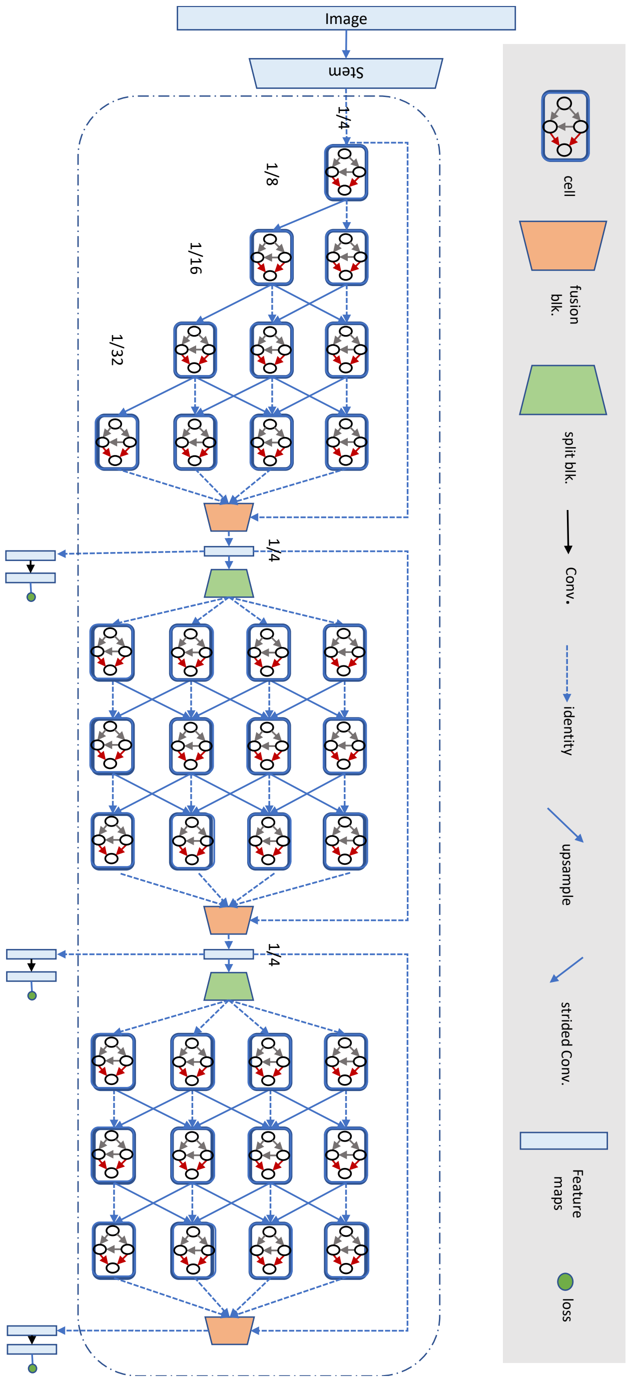

Figure 1.1: An illustration of 3M-Pose. The framework is based on PNFS [1] with cells densely connected in a fabric-like grid. Each EEL has what we call a fuse-split stage, where features are aggregated, outputted, and then finally redistributed back into the the features stream. See Figure 3.2, and Section 3.1.2 for details. Note that the intermediate output is also used as the skip features to the next EEL or to the last layer.

to measure our confidence in the output at each EEL and let the network stop early if it reaches above a threshold.

Supervision: To better guide our searching process, we supervised our search with MSE for heatmap loss, Associative Embedding Loss [10] for identities, and two other architecture supervision losses, a FLOP constraint and a student-teacher loss. We pair FLOP constraint with a student-teacher loss to avoid trivial architectural optimizations that satisfy the computational condition by disabling all expensive high-functioning functions at great expense to performance. Inspired by [15], we investigate an end-to-end version of the proposed TG-SAGE loss for faster search time. Instead of training and ranking premature architectures for final selection, we opt for a one-shot searching approach by continuously apply TG loss throughout the searching process. We use Higher-HRNet [7] as our teacher network.

1.4 Summary

We summarize our contributions as follows:

- **Framework:** We are the first to propose a NAS framework on a BU pose prediction task. We co-search and co-train our architecture with EELs for dynamic inference, therefore reducing overall computational cost without any significant degradation to model's performance.

- **Enabling Technique:** We propose Dyn-EGS a novel approach to sample micro and macro search spaces by allowing varying numbers of operators and inputs to be individually selected for each given cell. Furthermore, we employed a student-teacher loss that continuously guides 3M-Pose using any mature teacher network.

- **Performance:** Our discovered architecture is more efficient than existing bottom-up solutions with respect to parameter size and computational costs. Experiments demonstrate that 3M-Pose finds models with drastically superior speed and efficiency than existing works: 3M-Pose reduces computational cost by up to 93% and parameter size by 75% with the price of only a minor loss in performance.

2. PRIOR AND RELATED WORK

2.1 Recent Methods in top-down pose estimation

By the number of published literature, top-down or 2 stages approach is the more popular skew within the pose estimation research community. Since most top-down papers focused mainly on keypoint detection, the required hardware needed to do research is low, thus attracting more research. Recent works including [25, 26, 12, 1, 27, 28, 29] contains many hand-crafted approaches, and some Neural Architectural Search approach as well.

2.1.1 Traditional Approaches

Traditional approaches or hand-crafted approaches use human samples generated from an image by popular object detection models such as [30, 31, 32].

Typically, works such as [25, 27, 28] make use of a deep encoder network to generate feature-rich but spatially small tensors from a input image, before passing the encoded feature-rich tensor through a series of deconvolution layers to recover the desired information. The common chosen encoder networks are usually Resnet [33], Mobilenet[34], or Shufflenet [35]. However, physical capcity and the problem of vansihing gradients limit how deep simple feed-forward networks, such as those mentioned earlier, can be.

Works by [26], uses multiple parallel branches, with cross-resolution feature fusion, to take advantage of high resolution feature representation to achieve the current SOTA performance without the need of a very deep backbone.

2.1.2 NAS Approaches

Beside hand-crafted approaches, we are seeing more Neural Architectural Search approaches making their way into the pose estimation domain. Several recent works [1, 12] have used NAS to create their TD architecture.

[1] proposes Pose Neural Fabric Search framework, a derivative of Convolutional Neural Fabric [36], to estimate keypoints spatial location using keypoint vector space representation, and assumed prior knowledge. Their approach is similar to DARTs [16], but instead of training model's weights and model's architectural parameters on two separated training sets, they unified their search and train stages, thus saving time. However, because of this unique approach, their resulting network is very dense, since they cannot prune their search space.

[12] proposes a hybrid optimization scheme. They use DARTS [16] to optimization the mirco search spaces, and reinforcement learning [23] to optimize the macro search spaces.

2.2 Recent Methods in bottom-up pose estimation

Bottom-up or 1 stage approach is considered rare. Having to solve not only the keypoint estimation problem, but also the identity problems, researching into bottom-up approach can be prohibitively costly. All recent bottom-up works, [5, 6, 7, 8, 9, 10, 11], are hand-crafted approaches. All of them generate Gaussian heatmaps to localize the keypoints spatial locations, but differ greatly in how they approach keypoints identity groupings.

HigherHRNet [7] uses HRNet [4] and a devolution module to predict heatmaps at different resolutions. HigherHRNet groups by associative embedding, which assign keypoints with a identity vector or tag and groups keypoints by their l_2 tag distances. Unlike other approaches, [2] uses multiple parallel branches to develop high dimension representational matrix, therefore avoiding a need for a very deep backbone.

OpenPose [11] uses a two-branch multi-stage network: one to predict heatmaps, the other for grouping. OpenPose groups keypoints by using Part Affinity Fields, which learn a 2D vector field to link pairs of keypoints. Unlike HigherHRNet which unifed both keypoints estimation and identity grouping into a "single" stream , OpenPose opted to specialized individual branches for specific purpose. The main disadvantage from this approach is the inability for the network to share information between parallel branches, thus muting its potential.

[10] proposes the method of associative embedding to predict heatmaps and group keypoints. However, unlike other literature listed in this section, [10] focused mainly on the development of associative embedding for keypoint estimation and segmentation. To facilitate their experimentation, they uses Stack hour glass [37], which is a series of auto-encoder, as the backbone for their framework.

PersonLab [5] uses dilated ResNet[33] to predict and group keypoints directly. This approach is the most straight forward methods, however its simplicity does not yield very high quality results.

PifPaf [8] proposes a novel model to detect human poses in crowded images by using a Part Intensity Field (PIF) to localize parts and Part Association Field (PAF) to associate body parts with each other. Unlike other grouping method which require extensive post-processing steps to group identity correctly, PifPaf's approach allows for a simple greedy keypoints association thus making it quite ideal for mobile application.

2.3 Recent Methods in Neural Architectural Search

Neural Architectural Search is an active research topic in the field of computer vision and machine learning [12, 13, 1, 14, 15, 16, 17, 18, 19, 2, 20, 21, 22, 23, 2, 24]. Capable of designing networks that can surpass those handcrafted by human experts, NAS plays an important role in reducing labor costs associated with traditional approaches.

Reinforcement Learning NAS [23, 21, 12, 13] uses REINFORCE algorithm to transform non-gradient base metrics to reward signals that encourage unseen agents to design a network that maximize these reward signals.

Differentiable searching NAS [16, 14, 1] is a gradient base search approach which directly optimize the architecture against some loss metrics. Our framework 3M-Pose, falls into differentiable searching algorithm category. Despite the recent popularity of NAS, dense prediction tasks, e.g., human pose prediction, remain a challenge. To our knowledge, only a few works have applied NAS to human pose prediction and both use top-down approaches [12, 27].

2.4 Recent Methods in Dynamic Inference

Dynamic inference or adaptive inference [38, 13, 39, 40] aims to develop dedicated strategies to dynamically skip some computations during the inference phase for easy inputs. As a result, the model does not use unnecessary resources on simple inputs. Many works will build escape points into the architecture to achieve this outcome

Multi-Scale DenseNet (MSDNet) [38] manually designs an architecture to support anytime classification with classification blocks strategically placed throughout the depth of the network. However, because of the blatant placement of escape layer at every cross-section, this approach's worst case scenario is very costly.

[39, 40] divides the CNN architecture into multiple stages and exits inference on easy inputs. Unlike previous approaches which place escape layer at every junctions, these approaches incorporate some intelligent design thus minimizing the impact of the worst case scenario.

S2DNAS [13] uses NAS to generate all possible model configurations for dynamic inferencing from a seeded CNN architecture and then searches for the most optimal configuration from those options. This approach generates optimal escape layer placement using Neural Architectural Search but cannot generate a model on its own.

3. METHODOLOGY

In this section, we will go over the implementation of the framework and its supporting modules. We illustrate 3M-Pose framework in Figure 1.1. We will firstly discuss the overview of the proposed 3M-Pose framework and then describe its components in detail.

3.1 3M-Pose

3.1.1 Pose Neural Fabric Search

We borrow the general fabric layout as employed by [1] to facilitate the micro and macro search spaces. Each micro search space or cells are arranged in a grid or fabric, starting with highest resolution on the top row to the smallest resolution at the bottom row. Every subsequent row is $\frac{1}{2}$ the resolution of the previous row. Each cell contains h hidden layers (we chose h to be 2), and takes up to three inputs from its predecessors (see Figure 3.1).

We get our stem network from HRNet [4] which consists of two 3 x 3 strided convolution, followed by 4 residual units, and ends with one 1 x 1 convolution to reduce the resolution to $\frac{1}{4}$ the scale and feature maps' width to *C*. Our fabric is ten columns deep. The feature maps' width for each row are *C*, *2C*, *4C*, and *8C*, with *C* being 32. Following [10], we predicted the scalar tag for every keypoint.

3.1.2 Fuse-Split block

Unlike [1], we empirically found better performance by aggregating features across scales, instead of only using the features of the highest resolution.

To facilitate feature aggregation, we pass the set of inputs through a 1 x 1 convolution and up-sample them to $\frac{1}{4}$ resolution in a parallel fashion before we concatenate them together. Next, we feed the resulting tensors, which are *4C* wide, through another 1 x 1

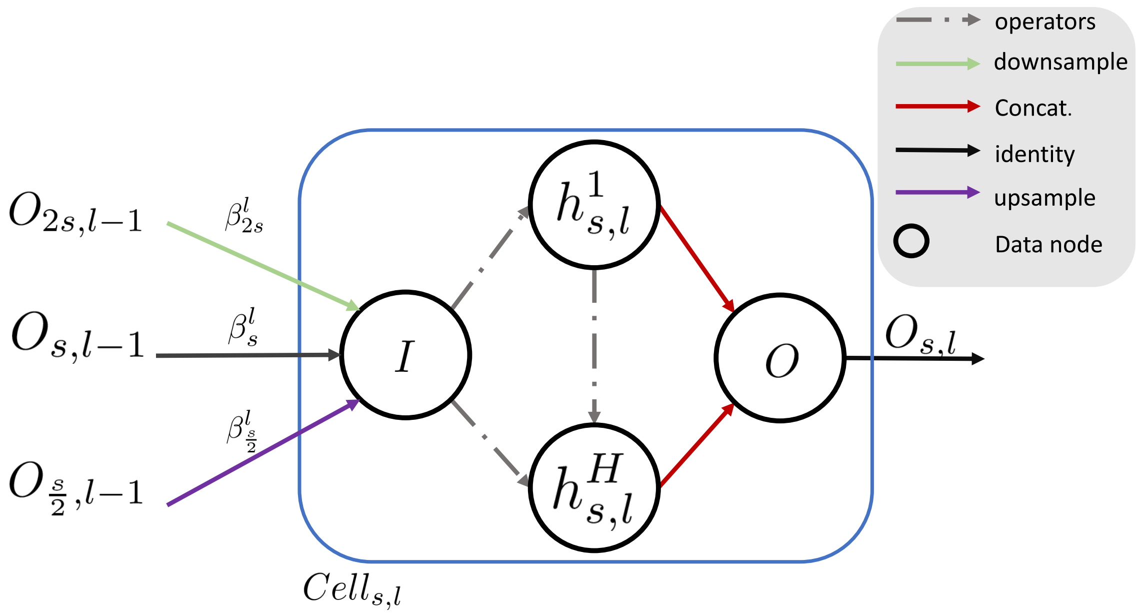

Figure 3.1: An illustration of of our micro search space. This figure includes our input, output, and hidden stages.

convolution, reducing the width back to C, before being activated by Squeeze-and-Excite block [41]. We repeat the previous step between skip-features and the fused features to obtain the final features.

The final features derived from fusion are forwarded to three places: the next Fuse block, the Split block, or the final stage to obtain the final output. For the EELs, we must keep feature continuity. We hence use the Split block to separate the final features using a 1 x 1 stride convolution to obtain the correct scale and width for each respective row before adding them with the input features of the Fuse block. Detail of our implementation can be view in Figure 3.2.

3.1.3 Early Escape Layers (EELS)

We designate layers along with the fabric depth as our EELs. At each EEL, we place a Fuse-Split module to facilitate feature fusion and dispersion. We chose the 4th and 7th layers, respectively. EELs provide additional points of supervision during training, and we found this approach improve our final layer's performance. Since 3M-Pose co-searches network architectures with consideration for EELs, the resulting architecture achieves better performance than those without.

3.1.4 Dynamic Inference

We recognize that not all inputs are equally hard. We improve efficiency by outputting easy inputs early. The challenge is constructing a reliable confidence score that can detect good outputs among the EELs. Since our model predicts the Gaussian distributed heatmap provided as ground truth, we can measure relative "goodness" of outputs by examining the predicted heatmaps's skew and kurtosis. Given a distribution curve, skew is given as:

$$\frac{moment_3}{(moment_2)^{\frac{3}{2}}} \tag{3.1}$$

while kurtosis is given as:

$$\frac{moment_4}{(moment_2)^2} \tag{3.2}$$

and moment is defined as:

$$moment_k = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^k \tag{3.3}$$

Our confidence equation is given as:

$$conf = f(c_0 * (\frac{c_1 * skew + c_2 * kurtosis}{2})) - 0.5 \tag{3.4}$$

Where $f(.)$ is the sigmoid activation function, N is the total number of detected keypoints, skew and kurtosis fall somewhere between 1 and -1, and $\{c_0, c_1, c_2\}$ are hyperparameters that can be tuned to extract better performance. Furthermore, if $N = 0$, we automatically

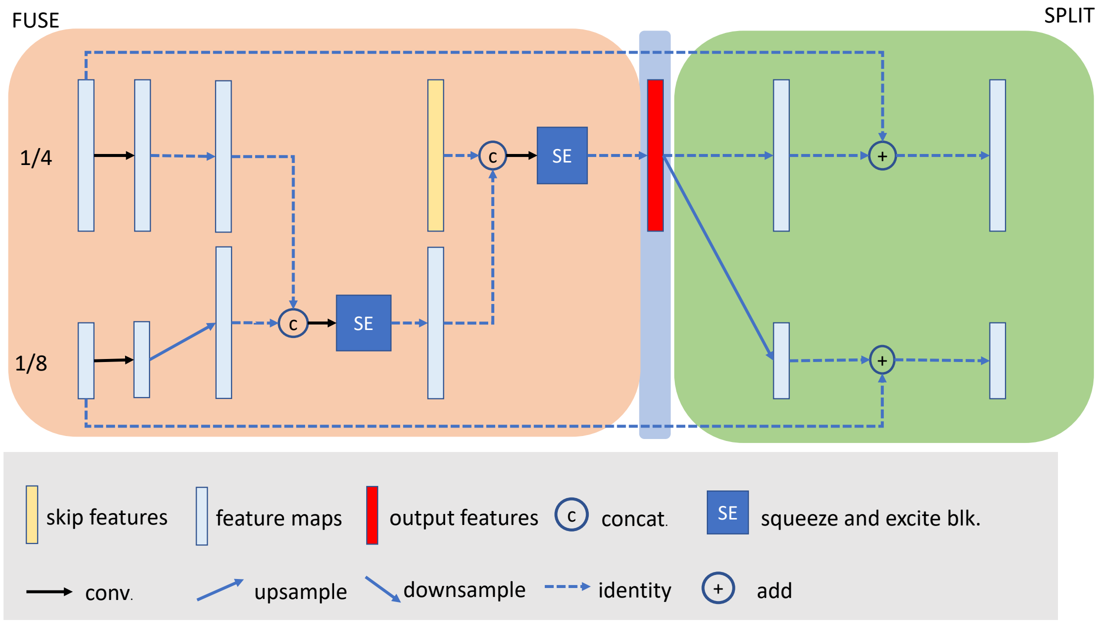

Figure 3.2: An illustration of our fuse-split stage with two scale factors. Note that input features into FUSE are forwarded to SPLIT, this behavior is not shown in Figure 1.1 for simplicity.

stop inferencing. Each EEL has its own set of (c_0, c_1, c_2). In our implementation we set the fourth layer's to be $(1, 1, 1)$ with a confidence threshold of 1, and the seventh layer to be $(2, 1, 1)$ with a confidence threshold of 0.6.

3.2 Search Spaces

In this section we denote h for hidden layer, H for number of hidden layers, s for scale, l for layer, α for the micro search space, β for the macro search space, and m for the density search space.

3.2.1 Gumbel-Softmax trick

We utilize DART as our primary method for discovering optimal architecture. Following, the common approach [16], we use Gumbel-softmax trick [42] to enhance the network ability to sample diverse range of architecture while avoiding initialization bias; the trick is given as:

$$y_i = \frac{e^{\frac{z_i+g_i}{\tau}}}{\sum_{j=1}^{k} e^{\frac{z_j+g_j}{\tau}}} \tag{3.5}$$

where g_k is the Gumbel noise and τ is the temperature parameter. We apply Gumbel-Softmax instead of Soft-max on our architectural parameter during the searching duration starting with $\tau = 10$ and linearly decrease it to a minimum of 0.001.

3.2.2 Micro Search Space

for any $cell_{s,l}$ uses $\alpha_{s,l}$ to determine which operator from the set of operators, $\mathcal{O}$, to be used. This behaviour is expressed as:

$$h_{s,l}^{H} = \begin{cases} \sum_{k=1}^{H-1} h_{s,l}^{k} + \sum_{k=1}^{|\mathcal{O}|} \alpha_{s,l}^{H,k} O_{s,l}^{H,k}(h_{s,l}^{H-1}), & H > 0 \\ \overline{I}_{s,l}, & H = 0 \end{cases} \tag{3.6}$$

where $\overline{I}_{s,l}$ is the input to the cell. O $\in \mathcal{O}$, and $\mathcal{O}$ is a set of 9 different operators, which are:

- Identity
- Conv (3x3)
- Avg pool (3x3)
- Max pool (3x3)
- Residule Block (3x3)
- Depth Wise Separable Convolution (3x3)
- Conv-Sigmoid * Identity
- Zoom Convolution (3x3) [43]
- Squeeze-and-Excite [41]

Finally, we obtain the $cell_{s,l}$'s output, $O_{s,l}$, by concatenating all features, $\{h_{s,l}^{1}, ..., h_{s,l}^{H}\}$, and passing the concatenated features through a 1 x 1 convolution.

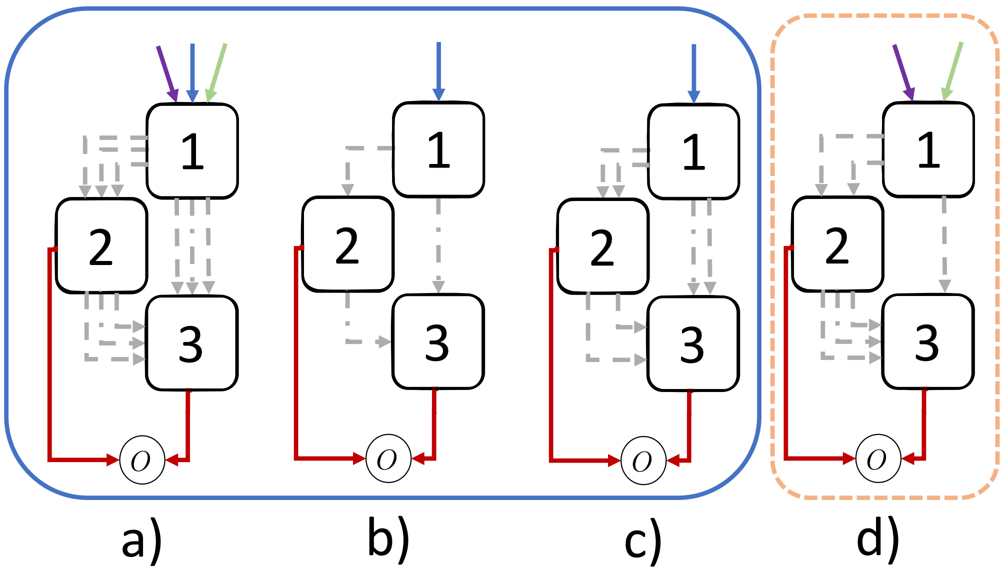

Figure 3.3: An illustration of different architecture sampling methods: a) densely connected cells, b) *argmax* discrete architecture, c) DARTS-EGS [2] with $r = 2$, d) Our Dynamic Ensemble Gumbel-Softmax. Note that while the cells in (a), (b), and (c) have a static architecture, our cells' architectures can vary from cell to cell.

3.2.3 Macro Search Space

The input, $\overline{I}_{s,l}$, for any $cell_{s,l}$ is a combination of inputs from its predecessors at different scales. The resulting fusion is given the following expression:

$$\overline{I}_{s,l} = \beta^0_{s,l} O_{\frac{s}{2},l-1} + \beta^1_{s,l} O_{s,l-1} + \beta^2_{s,l} O_{2s,l-1} \tag{3.7}$$

3.3 Dynamic Ensemble Gumbel Softmax (Dyn-EGS).

Once we completed the searching process, we have to derive the final architecture from the supernet. Most works will constrain the selected architecture to be either dense or sparse, but making this choice places unnecessary limitations on the final architecture. Both types of architecture have their associated problems. A densely connected architecture is computationally inefficient and physically impossible to fit on a GPU in our case. On the other hand, a sparse network is computationally efficient but does not have enough parameters to represent the data effectively. Recent work by [2] proposed DART-EGS, a differentiable method to sample multiple operators in an end-to-end manner. However, DART-EGS uses a fixed sampling rate *r* which forces every cell to sample *r* times. DART-EGS' static nature means that it cannot react to either performance or computational needs during the architecture search.

To add the ability to adapt the number of parameters in the current representation based on the available budget, we improve on this idea and propose Dyn-EGS. We represent Dyn-EGS as m in our architecture parameters. To independently control the density for micro and macro searching spaces, we further differentiate $\{m_\alpha, m_\beta\}$, respectively. m_α has a dimension of $|cells|$ x H x R_α. Similarly, m_β has a dimension of $|cells|$ x 3 x R_β. Here, $|cells|$ are the total number of cells, H the total number of hidden layers, and $\{R_\alpha, R_\beta\}$ the possible number of sampling states for the respective search spaces.

During search, we apply the one-shot Gumbel-Softmax trick, θ, to *m*. Let *i* be one of the positive indexes from $\theta(m_\alpha)$ which represent a sampling state for $cell_{s,l}$ at hidden layer h; then the resulting operators for this particular cell, at this particular hidden layer are $\overline{\alpha}^h_{s,l} = \sum_{k=1}^{i+1} \theta(\alpha^h_{s,l})$ (note that $\overline{\alpha}$ is discrete while α is continuous). To derive the final architecture, we replace θ with *argmax* to find *i* from $\{m_\alpha, m_\beta\}$, and select the top-i operators and or paths from $\{\alpha, \beta\}$, see Figure 3.3.

3.4 Supervision

During the searching process, we use ModelLoss+ $\left(\frac{GFLOPS}{T}\right)^{\mathbf{w}}$+TeacherLoss as the optimization goal, where *ModelLoss* is the summation of Mean-Square-Error loss for heatmaps and Associative Embedding loss [10] for identities.

FLOP constraint We target GFLOPs rather than latency since we are not targeting any specific hardware. Our value for *T*, 47.8, is reported by Higher-HRNet [7], and *w* is 0.15. We observe that aggressive pruning in the early searching stage can produce efficient but ineffective architectures. To rectify this phenomenon, we employ a simple step-case loss scaling from 0.0001, increasing by a power of 10 every three epochs to a ceiling of 1.

Student-teacher loss. 3M-Pose uses a teacher model to guide the search process to avoid trivial optimization caused by aggressive pruning from our FLOP constraint such as the "latency collapse" as revealed in [43]. Inspired from [15], we use the Representational Dissimilarity Matrix (RDM) [44] to quantify the activation patterns of the representational space in response to a set of inputs for any particular activated tensors. RDM is calculated by computing the pair-wise distances between each pair of activation vectors using a distance measure; for our case, we use simple Euclidean distance. Let *X* be the set of RDM from the student network, and *Y* be the set of RDM from the teacher network. For the student network, RDM is calculated for every cell's output; for the teacher network, RDM is calculated for only a set of handpicked outputs. We calculate the student-teacher similarity by $Sim_i = max(corr(Y_i, X_j))$ for $j \in \{1, 2, ..., |X|\}$ and $i \in \{1, 2, ..., |Y|\}$. We use Pearson correlation to measure the similarity and then construct the student-teacher loss by taking $1 - mean(Sim)$.

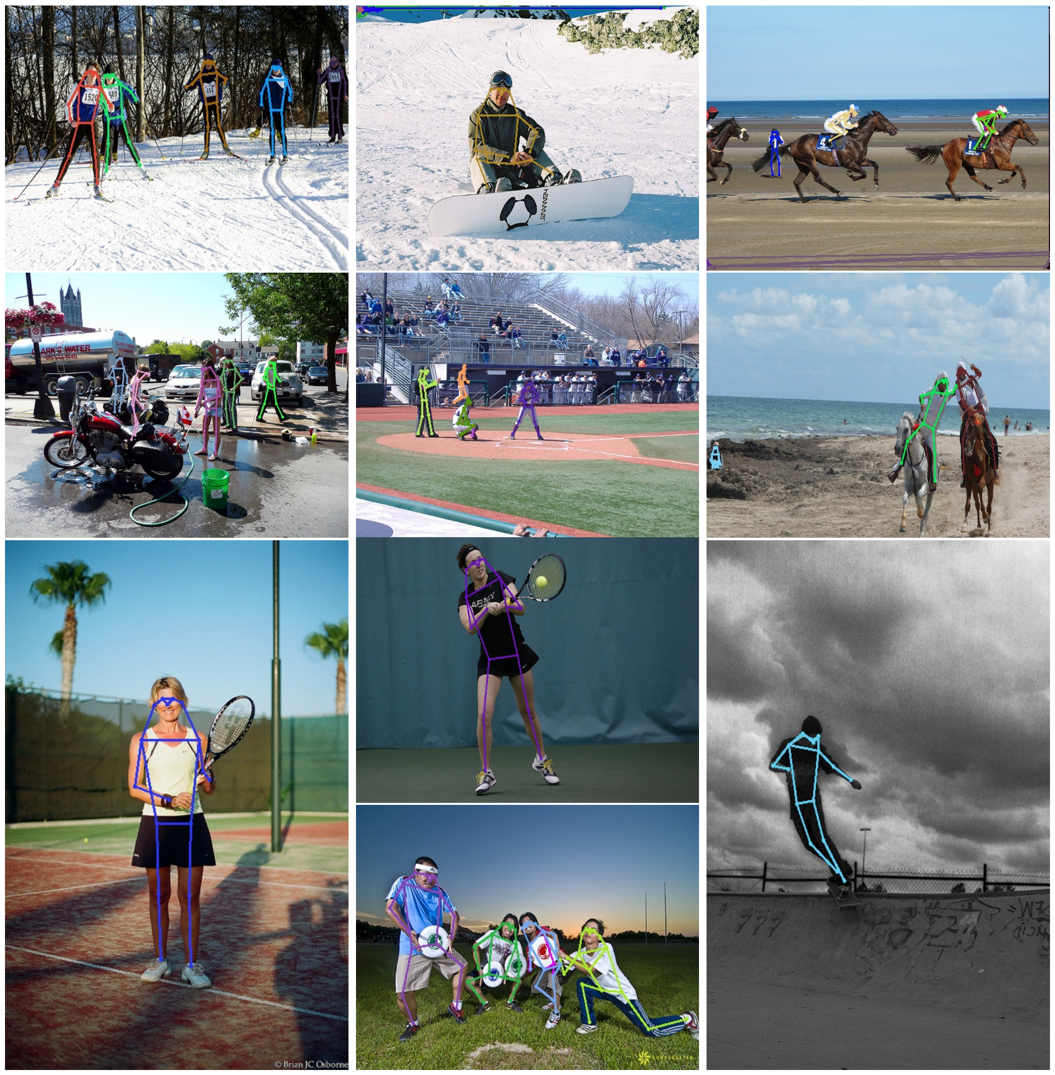

Figure 3.4: We demonstrate our model's performance on obtained data from publicly available COCO dataset [3]. The colored skeletal structure is drawn using predicted keypoints from 3M-Pose.

4. EXPERIMENTS

Table 4.1: Comparison with hand-crafted bottom-up and top-down NAS methods on COCO2017 validation set. GFLOPs are calculated at single scale. Note that we are using static-inference on the last layer to report our results. (*) the GFLOP per person detected bounding boxes and not image sample. [1] indicates using refinement.

Method	Mode	Backbone	Input size	#Params(M)	GFLOPs	AP	AP^{50}	AP^{75}	AP^{M}	AP^{L}
Manual approaches										
OpenPose [11] [1]	BU	-	-	-	-	61.0	84.9	67.5	56.3	63.3
Hourglass [10]	BU	Hourglass	512	277.8	206.9	56.6	81.8	61.8	49.8	67.0
Personlab [5]	BU	ResNet-152	1401	68.7	405.5	66.5	86.2	71.9	62.3	73.7
PifPaf [8]	BU	ResNet-152	241x321	62.09	75.38	67.4	-	-	-	-
HigherHRNet [7]	BU	HRNET-w32	512	28.6	47.9	67.1	87.5	72.8	61.5	76.1
NAS approaches										
AutoPose [12]	TD	-	256	-	10.65*	73.6	90.6	80.1	69.8	79.7
PNFS [1]	TD	-	256	27.5	11.4*	73.0	-	-	-	-
3M-Pose (Ours)	BU	4r10d32w	512	17.79	16.63	58.3	82.4	64.1	53.9	67.6
3M-Pose (Ours)	BU	4r10d32w	640	17.85	25.98	62.5	84.0	68.4	57.6	69.8

4.1 COCO Keypoint Detection

Dataset. The COCO dataset [3] contains 250,000 samples with one or more people; each person instance is labeled with 17 keypoints. The dataset is split into train/validation/test subsets with 57k, 5k, and 20k images respectively. Following [16], we perform our architecture search by training on the train set and checking performance on the validation

Table 4.2: Ablation study of 3M-Pose's components on COCO2017 validation dataset. tea+flop: GFLOP constraint and student-teacher loss. FS, fuse-split module

Dyn-EGS	tea+flop	EELs	FS	AP	GFLOP
✗	✗	✗	✗	31	31
✓	✓	✗	✗	41.5	20.6
✓	✓	✓	✗	50.1	24.5
✓	✓	✓	✓	58.3	16.8

Table 4.3: Ablation study comparing performance, and computational cost at different layers, versus using dynamic inference

Input	layer	Gflop	AP	AP(M)	AP(L)
512	4	10.14	22.1	25.0	17.9
	7	12.38	54.3	49.1	62.3
	10	16.63	58.3	53.9	67.6
	Dynamic	13.82	57.9	52.2	66.9
640	4	15.85	24.5	27.1	22.0
	7	19.34	59.0	54.4	65.5
	10	25.98	62.5	57.6	69.8
	Dynamic	20.16	61.7	56.5	69.2

dataset. During evaluation, we train exclusively on the train dataset and evaluate on the test dataset. We reports our results on the validation set for our ablation and dynamic inference studies and on the test set for when comparing with other SOTA methods.

Following [10] and [7], we perform random scaling in the range of $[0.75, 1.25]$, random rotation in the range of $[-30^o, 30^o]$, random translation in the range of $[-40, 40]$, and random flip on all our image data before cropping the images to 512x512. We use the same preprocessing steps during both the search and train stages.

Searching. Following [16], we optimize our architecture parameters $\{\alpha, \beta, m\}$ on the validation set and the model's parameters, ω, on the training set. We use the Adam optimizer [45] for both processes, which we will denote as op_{arch} and op_{model} respectively. op_{model} has a base learning rate of $1\text{e}{-3}$ which remains constant throughout. On the other hand, op_{arch} has a base learning of $5\text{e}{-3}$ which drops to $2.5\text{e}{-3}$ and $1.25\text{e}{-3}$ at the $20th$ and $25th$ epochs respectively. We search for a total of 35 epochs.

We balance between heatmap loss, grouping loss, teacher guidance loss, and GFLOP loss. To give more weight to the heatmap loss and teacher guidance losses, we set the weights to 1, $1\text{e}{-3}$, 1, and $1\text{e}{-3}$, respectively, when combining these metrics.

We aim for tiny but capable BU architectures. We limit Dyn-EGS possible sampling states by setting R_α, our operator's sampling limit, and R_β, our path sampling limit, to be 2 and 3, respectively. While it is possible to extract more performance with higher sampling, we find the trade-of between the gain in performance and the increase the inefficiency unappealing for our paper's goal.

Our fabric is four rows wide and ten layers deep with 34 cells, each with two hidden stages. We select the fourth and seventh layers to be EELs. We set *C*, the channel-width, to be 32. We found the discovered architecture is transferable across different fabrics with different input sizes and channel-width, as long as the numbers of rows, columns, and hidden stages remain constant. We report our architecture in the supplementary section.

Training. From the derived architecture, we first prune unnecessary operators and paths from the micro and macro search spaces. We use an Adam optimizer with a base learning rate of $1ee{-}3$ that drops to $1e{-}4$ and $1e{-}5$ at the *200th* and *260th* epochs, respectively. We train the model for 300 epochs. We follow [7] and balance heat map losses with grouping loss with weights of 1 and $1e{-}3$.

Testing. Following testing procedures by [7] and [10], we resize the short side of image to 512 while keeping the aspect ratio. We use flip testing on all experiments. We report accuracy and GFLOPs from both static and dynamic inference in Table 4.3. In static inference, we only report the output of the last layer of the fabric.

Results on COCO2017 validation. Table 4.1 summarizes the results on COCO2017 validation dataset on hand-crafted bottom-up approaches and NAS top-down approaches. The results shown that 3M-Pose is small and efficient in terms of the model's total numbers of parameters and GFLOPs. We outperform OpenPose [11] and Hourglass [10] in accuracy, while being 87.5% more computationally efficient compared to [10]. We acknowledge our performance short-coming comparing to Personlab [5] and HigherHRNet [7]. We see a reduction of 93%, and 45.76% in computational cost and 75% and 63% fewer parameters in exchange for only 4% degradation in performance, an acceptable tradeoff. In comparison to other NAS's pose estimation works, 3M-Pose is smaller with respect to the number of parameters. This benefit is compounded by the fact that 3M-Pose is a single-model approach while the other works use two models. Furthermore, our model has comparable performance with respect to GFLOPs and better performance on images containing two or more people. Top-down computational cost is proportional to the number of people in the image while bottom-up approaches' cost is independent of that factor.

4.2 Ablation Study

Effectiveness of our proposed modules: Table 4.2 compares the baseline to our proposed modules to demonstrate their effectiveness in guiding both search and train processes. The models in this table is evaluate on COCO2017 validation dataset.

a) *Baseline model*: We search and train the baseline architecture using none of the proposed features, relying solely on heatmap and grouping loss to guide both processes. It achieves an AP = 31$\%$ while costing 31 GFLOP per sample.

b) *Dyn-EGS, student-teacher loss and GFLOP constraint*: By adding Dyn-EGS to sample complex operators and paths alongside with GFLOP constraint and teacher guidance, 3M-Pose achieves a AP = 41.5, a significant $+10.5$ gain in performance with a -10.44 reduction in computational costs. The result indicates Dyn-EGS's ability to design effective micro and macro sampling strategies under computational constraint, and student-teacher loss's ability to counteracts trivial search collapse causes by excessive pruning.

c) *Early Escape Layers*: Next, we co-search and co-train 3M-Pose with EELs. Note that without a fuse-block module, we only use the feature of the largest dimension as intermediary outputs, and we do not forward these features to the next EEL or final layer. We again saw a significant improvement in performance, $+8.6$, which indicate the important of supervising intermediate outputs in both search and train process.

d) *Fuse-Split module*: Finally, we add fuse-split after every EELs to facilitate feature fusion, dispersion, and skip connection. This time, the GFLOP constraint accounts for the entire model, searchable and non-searchable modules alike. This final configuration achieves an AP of 58.3, an $+8.2$ increase from the previous model and $+27.3$ total increase from baseline. This version also has the lowest computational costs of only 16.8 GFLOP per sample.

Evaluation with Dynamic Inference. Table 4.3 compares the effectiveness of dynamic inference against static inference at different output layers in 3M-Pose. Again we use COCO2017 validation set to evaluate our results.

We see a 16.8% reduction in computational cost for our smaller model with a 0.6% degradation in performance. 38.7% of the total outputs are from the fourth layer, while 8% is from the seventh layer, with the rest, 53.26%, attributed to the tenth layer. Likewise, the larger model sees a 22% reduction in GFLOP with a 1.28% degradation in performance. The outputs are distributed among the fourth, seventh, and tenth layers at 49%, 11.8%, and 38.9%, respectively. Despite the poor performance, the fourth layer is an excellent human detector and can reject most of the negative samples within the validation dataset. This result shows our simple metric's effectiveness to select the correct output layer for a given input.

5. CONCLUSION, LIMITATION, AND DISCUSSION

We introduce 3M-Pose as the first BU human pose prediction framework via NAS. Our predecessors [12, 1] made respectable initial attempts for NAS in HPE, but their discovered models were computationally heavy yet still produced inferior and uncompetitive performance w.r.t. handcrafted SOTAs. By contrast, we are delighted to see that 3M-Pose has made a much more meaningful step towards practically useful NAS in HPE by finding an ultra-lightweight BU network that achieves comparable results, revealing the appeal of NAS to HPE practitioners. Our work is aligned with the recent trend in "refocusing" the NAS goal to generating mobile friendly and tiny neural networks. For example, TinySR [46] utilized NAS to find extremely compact super-resolution networks, with slightly compromised yet still strong performance compared to SOTA methods. We consider this efficiency-oriented direction to be a blue ocean for NAS.

Applying NAS to HPE, or complicated vision tasks that involve dense or point set predictions in general, is more challenging than applying it to image classification. This difference is owing to the the enormous design variations caused by the task complexity. For HPE, a large degree of ambiguity for identity grouping and the challenge of identifying keypoints at different scales have placed daunting barriers for any manual or automated design. Recall the initial difficulty of AutoML, where it can only design neural networks with comparable results to human experts, and these results were constrained to small academic datasets such as CIFAR-10 and the Penn Treebank [47]. Likewise, there are many aspects in which 3M-Pose can improve.

In order to make 3M-Pose more competitive, there are many techniques we can apply to improve its performance. Some are directly inspired by the handcrafted models, e.g., [7]. For example, we can increase the channel width, e.g., from 32 to 48 which has been shown by [7] to increase overall accuracy by 2%, leverage our efficiency with inputs at higher resolution, or make 3M-Pose scale-aware by using a deconvolutional module to output multiple different resolutions which has shown to improve accuracy by 1.4% [7]. Since 3M-Pose is modular in design, we can easily scale the model up to enjoy similar or even better performance gains, though perhaps sacrificing a bit efficiency. We further note that the scaling up can be an integrated part of NAS by holistically searching the expansion ratio, the scaling factor, the number of channels, and the depth of the overall network. For example, EfficientNet [48] uses NAS to search a network's compound scaling ratio to achieve a favorable accuracy-resource tradeoff on ImageNet. We aim to adopt their multi-dimensional scaling into the future versions of 3M-Pose too.

REFERENCES

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

[2] J. Chang, X. Zhang, Y. Guo, G. Meng, S. Xiang, and C. Pan, “Differentiable architecture search with ensemble gumbel-softmax,” *arXiv preprint arXiv:1905.01786*, 2019.

[3] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: common objects in context,” *CoRR*, vol. abs/1405.0312, 2014.

[4] K. Sun, B. Xiao, D. Liu, and J. Wang, “Deep high-resolution representation learning for human pose estimation,” *CoRR*, vol. abs/1902.09212, 2019.

[5] 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.

[6] K. Sun, Z. Geng, D. Meng, B. Xiao, D. Liu, Z. Zhang, and J. Wang, “Bottom-up human pose estimation by ranking heatmap-guided adaptive keypoint estimates,” *arXiv preprint arXiv:2006.15480*, 2020.

[7] 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,” *arXiv preprint arXiv:1908.10357*, 2019.

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

[9] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *ECCV*, 2016.

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

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

[12] 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.

[13] Z. Yuan, B. Wu, Z. Liang, S. Zhao, W. Bi, and G. Sun, "S2dnas: Transforming static cnn model for dynamic inference via neural architecture search," *arXiv preprint arXiv:1911.07033*, 2019.

[14] S. Saxena and J. Verbeek, "Convolutional neural fabrics," in *NeurIPS*, 2016.

[15] P. Bashivan, M. Tensen, and J. J. DiCarlo, "Teacher guided architecture search," in *CVPR*, 2019.

[16] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[17] J. Mei, Y. Li, X. Lian, X. Jin, L. Yang, A. Yuille, and J. Yang, "Atomnas: Fine-grained end-to-end neural architecture search," *arXiv preprint arXiv:1912.09640*, 2019.

[18] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," *arXiv preprint arXiv:1908.09791*, 2019.

[19] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *CVPR*, 2019.

[20] 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.

[21] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[22] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *CVPR*, pp. 8697–8710, 2018.

[23] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," *arXiv preprint arXiv:1611.02167*, 2016.

[24] H. Cai, J. Yang, W. Zhang, S. Han, and Y. Yu, "Path-level network transformation for efficient architecture search," *arXiv preprint arXiv:1806.02639*, 2018.

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

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

[27] X. Sun, B. Xiao, F. Wei, S. Liang, and Y. Wei, "Integral human pose regression," in *ECCV*, 2018.

[28] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun, "Cascaded pyramid network for multi-person pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7103–7112, 2018.

[29] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, *et al.*, "Deep high-resolution representation learning for visual recognition," *TPAMI*, 2020.

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

[31] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NeurIPS*, 2015.

[32] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[34] 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," *CoRR*, vol. abs/1704.04861, 2017.

[35] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," *CoRR*, vol. abs/1707.01083, 2017.

[36] S. Saxena and J. Verbeek, "Convolutional neural fabrics," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, (Red Hook, NY, USA), p. 4060–4068, Curran Associates Inc., 2016.

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

[38] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger, "Multi-scale dense networks for resource efficient image classification," *arXiv preprint arXiv:1703.09844*, 2017.

[39] K. Berestizshevsky and G. Even, "Dynamically sacrificing accuracy for reduced computation: Cascaded inference based on softmax confidence," in *ICANN*, 2019.

[40] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *ICPR*, 2016.

[41] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," 2019.

[42] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," 2017.

[43] W. Chen, X. Gong, X. Liu, Q. Zhang, Y. Li, and Z. Wang, "Fasterseg: Searching for faster real-time semantic segmentation," *arXiv preprint arXiv:1912.10917*, 2019.

[44] N. Kriegeskorte, M. Mur, and P. A. Bandettini, "Representational similarity analysis-connecting the branches of systems neuroscience," *Frontiers in systems neuroscience*, vol. 2, p. 4, 2008.

[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[46] R. Lee, Ł. Dudziak, M. Abdelfattah, S. I. Venieris, H. Kim, H. Wen, and N. D. Lane, "Journey towards tiny perceptual super-resolution," in *European Conference on Computer Vision*, pp. 85–102, Springer, 2020.

[47] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *CoRR*, vol. abs/1611.01578, 2016.

[48] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *CoRR*, vol. abs/1905.11946, 2019.

APPENDIX A

A.1 Resulting Architecture

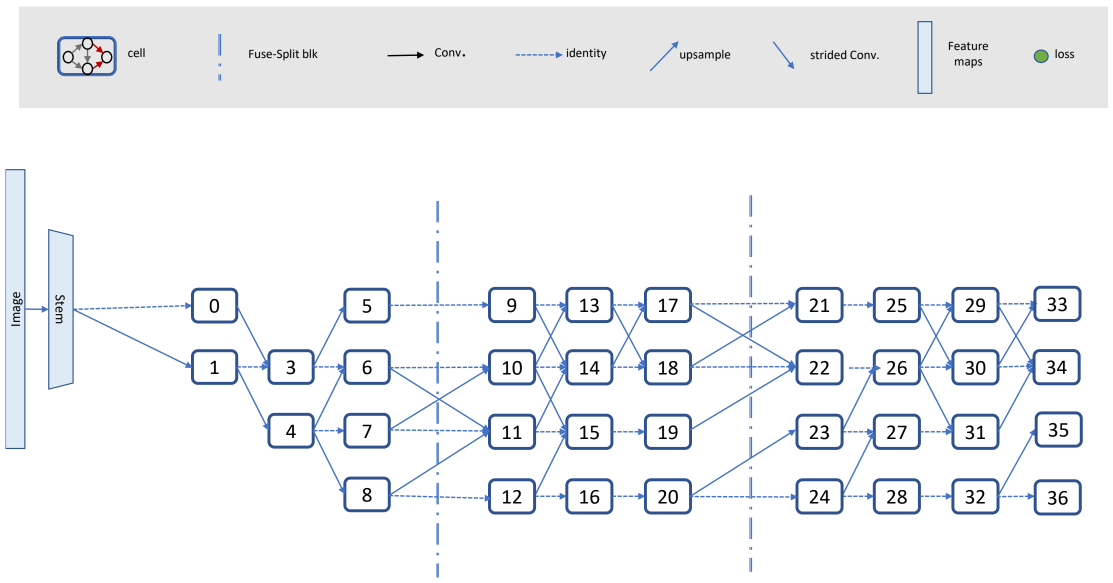

Figure A.1: An illustration of the searched architecture at the macro level. We simplify the Fuse-Split block as a dashed line for better clarity. Each cell is marked with a number that will correspond to table A.1.

Table A.1: Our searched architecture break down by cell's position corresponding to figure A.1. $O^0_{s,l}$ is the operator between I and $h^1_{s,l}$, $O^1_{s,l}$ is between I and $h^2_{s,l}$, and $O^2_{s,l}$ is between $h^1_{s,l}$ and $h^2_{s,l}$. SE stands for Squeeze and Excite. CSI stands for Conv-Sigmoid * Identity. SepConv is Depth-wise Separable Convolution.

Cell No.	Inputs	$O^0_{s,l}$	$O^1_{s,l}$	$O^2_{s,l}$
0	parallel	Max + Avg	Max + Avg	Identity
1	above	Identity	Identity + CSI	Identity + CSI
2	-	-	-	-
3	parallel + above + below	Identity + Residual	Identity + SepConv	Identity + Residual
4	above	Identity + Avg	Max + Avg	Identity + Avg
5	below	CSI + SE	CSI + SE	Residual + SepConv
6	parallel + below	Identity + Residual	Identity + SepConv	Identity + SE
7	parallel	Identity + CSI	Identity + CSI	Identity + CSI
8	above	Max	Max	Avg
9	parallel	Identity + Avg	Identity + SepConv	Avg + SE
10	parallel + below	Identity	Identity + ZoomConv	Residual + ZoomConv
11	parallel + above + below	Residual + ZoomConv	Identity + Residual	Avg + Residual
12	parallel	Avg	Avg	Conv
13	parallel + below	Identity + SE	Identity + SE	Avg + SE
14	parallel + above + below	CSI	Identity + Residual	ZoomConv + SepConv
15	parallel + above + below	Avg + CSI	Avg + CSI	Avg
16	parallel	Avg + CSI	CSI + SepConv	Residual + Conv
17	parallel + below	Identity + CSI	CSI + Conv	SE + SepConv
18	parallel + above	SE + ZoomConv	Identity + SE	ZoomConv + SepConv
19	parallel	CSI + ZoomConv	CSI	Residual + SepConv
20	parallel	Conv + SepConv	Identity + ZoomConv	Conv + ZoomConv
21	parallel + below	SE	SE + SepConv	Max
22	parallel + above + below	Max + Avg	Max + Avg	Max + SE
23	below	ZoomConv + SepConv	ZoomConv + SepConv	Max + Avg
24	parallel	Max + Avg	Identity + Avg	Max + Conv
25	parallel	SE + SepConv	Identity + Residual	Max + Avg
26	parallel + below	Max + Avg	Max + Avg	Max + Avg
27	parallel + below	Residual + SepConv	ZoomConv + SepConv	Max + Avg
28	parallel	Identity + CSI	Conv + SepConv	Max + Conv
29	parallel + below	Conv	SepConv	Max
30	parallel + above	Max + Avg	Max + Avg	Max + SE
31	parallel + above	Avg + Residual	Conv + ZoomConv	Max + ZoomConv
32	parallel	CSI + ZoomConv	Avg + ZoomConv	Conv + ZoomConv
33	parallel + below	Residual	SepConv	SepConv
34	parallel + above + below	ZoomConv	Identity	SepConv
35	below	CSI	SepConv	SepConv
36	parallel	CSI + ZoomConv	ZoomConv + SepConv	Avg + Residual