MTL-NAS: Task-Agnostic Neural Architecture Search towards General-Purpose Multi-Task Learning

Yuan Gao[1*], Haoping Bai[2*†], Zequn Jie[1], Jiayi Ma[3], Kui Jia[4], and Wei Liu[1]
[1] Tencent AI Lab [2] Carnegie Mellon University
[3] Wuhan University [4] South China University of Technology
{ethan.y.gao, bhpfelix, zequn.nus, jyma2010}@gmail.com, kuijia@scut.edu.cn, wl2223@columbia.edu

Abstract

*We propose to incorporate neural architecture search (NAS) into general-purpose multi-task learning (GP-MTL). Existing NAS methods typically define different search spaces according to different tasks. In order to adapt to different task combinations (*i.e.*, task sets), we disentangle the GP-MTL networks into single-task backbones (optionally encode the task priors), and a hierarchical and layer-wise features sharing/fusing scheme across them. This enables us to design a novel and general* ***task-agnostic search space****, which inserts cross-task edges (*i.e.*, feature fusion connections) into fixed single-task network backbones.*

Moreover, we also propose a novel single-shot gradient-based search algorithm that closes the performance gap between the searched architectures and the final evaluation architecture. This is realized with a minimum entropy regularization on the architecture weights during the search phase, which makes the architecture weights converge to near-discrete values and therefore achieves a single model. As a result, our searched model can be directly used for evaluation without (re-)training from scratch.

We perform extensive experiments using different single-task backbones on various task sets, demonstrating the promising performance obtained by exploiting the hierarchical and layerwise features, as well as the desirable generalizability to different i) task sets and ii) single-task backbones. The code of our paper is available at https://github.com/bhpfelix/MTLNAS.

1. Introduction

Recent years have witnessed the great success of deep neural networks. It integrates hierarchical feature extraction and optimization in an automatic and end-to-end manner [24, 26, 30, 57]. Although deep learning algorithms relieve researchers from feature engineering, they still need carefully designed neural architectures. More recently, Neural Architecture Search (NAS) has received increasing attentions in automating the design of deep neural architectures [71]. NAS methods have produced highly competitive architectures on various computer vision tasks, such as image classification [5, 8, 9, 25, 35, 43, 59, 71], object detection [7, 14, 66], and semantic segmentation [33, 44].

Another paradigm to boost the performance of the deep neural networks is multi-task learning (MTL) [29, 51]. Multi-task learning has achieved a success in many applications by learning multiple related tasks simultaneously. This success mainly owes to two key factors, *i.e.*, different tasks produce multiple supervision signals that, i) impose additional regularization among different tasks, and ii) produce implicit data augmentation on labels [13, 51]. Traditional MTL methods share a single convolutional feature extractor among all tasks, and keep a separate head for each task to produce task-specific predictions. It implicitly assumes that all the learned hierarchical and layerwise features are identical, and are able to perform well on all the tasks. Recent researches show that such an assumption does not always hold [42], *i.e.*, improper feature sharing may introduce negative transfer of some tasks, thus leading to the degradation of the performance [51].

It is natural to consider incorporating NAS into general-purpose MTL (GP-MTL) to pursue better architectures. To benefit GP-MTL, the NAS algorithm has to adapt to different tasks (or different combinations of tasks, *i.e.*, task sets, in MTL). However, the existing NAS by design prohibits to do so as the search spaces are typically different across different tasks. Fundamentally, those differences in the search spaces reflect different task priors (*e.g.*, see how the prior of semantic segmentation is encoded in Auto-Deeplab [33]). But when learning on multiple tasks, it is extremely difficult to encode the priors for multiple tasks into the search space, especially when the tasks are loosely related.

We tackle this difficulty by disentangling the task prior from the feature sharing/fusing scheme in GP-MTL. More

* Equal contributions with a random order.
† Work performed at Tencent AI Lab.

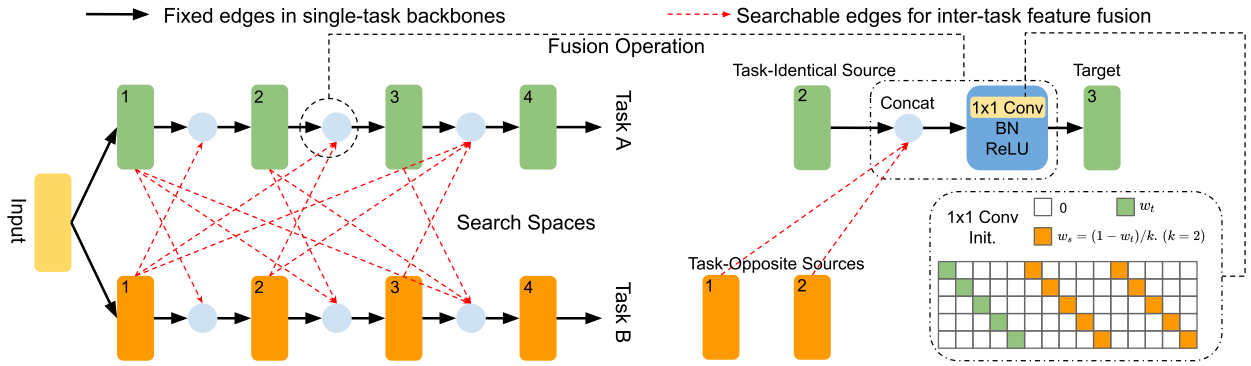

Figure 1. The problem formulation of the proposed general-purpose MTL-NAS. We disentangle the GP-MTL networks into fixed task-specific single-task backbones and general feature fusion schemes between them. This allows us to define a general task-agnostic search space compatible with any task combinations, as shown in the leftmost subfigure. The right-top subfigure illustrates the inter-task fusion operation, which is motivated by and extends from the NDDR-CNN [13]. We show the initialization of the fusion operation in the right-bottom subfigure. As we are inserting new edges between the fixed and well-trained single-task network backbones, we wish to make a minimal impact on the original output at each layer at initialization (*i.e.*, initializing with a large w_t) (best viewed in color).

specifically, we formulate the GP-MTL paradigm to consist of i) multiple single-task backbone networks (optionally encode the task priors), and ii) the general, hierarchical and layerwise feature sharing/fusing scheme across different backbones (please see the leftmost subfigure of Fig. 1). This formulation enables us to design a general **task-agnostic search space** for GP-MTL[1]. Specifically, we start with multiple fixed single-task network branches, representing each intermediate layer as a node and the associated feature fusion operations as an edge. The problem thus becomes to seek the optimal edges between the inter-task node pairs, where *the search space is unified for any task set*.

Moreover, we also propose a novel single-shot gradient-based search algorithm that closes the performance gap between search and evaluation. Specifically, it is noticed that the validation performance obtained in the search phase cannot always generalize to evaluation. Fundamentally, it is because that the architecture mixture optimized during the search phase often fails to converge to a discrete architecture, leading to a drop in performance when the final architecture is derived [64]. We address this issue by reducing the uncertainty of the architecture via entropy minimization during the search phase, resulting in a single model for evaluation directly without the need of (re-)training from scratch, which significantly reduces the performance gap existing in the popular single-shot gradient-based NAS algorithms [35, 64].

Note that we focus on GP-MTL in this paper. The proposed method can adapt to different task combinations to generate different inter-task architectures. We fix the single-task backbone branches and search good inter-task edges for hierarchical and layerwise feature fusion/embedding. We also note that the task priors of specific backbone networks (*e.g.*, large convolution kernel in semantic segmentation) can be exploited to further improve the performances on those tasks. But learning the task-specific backbone architecture itself is beyond our scope of GP-MTL. Instead, we design a task-agnostic search space to learn the feature sharing/fusing architecture. We also verify the consistent performance improvement by applying the proposed method to different backbone architectures against the GP-MTL methods [13, 42]. In summary, our contribution lies in both the search space and the search algorithm:

- **Search Space:** We define a novel task-agnostic search space that enables us to exploit the hierarchical and layerwise feature fusion scheme for GP-MTL, decoupled from the task priors within fixed single-task network backbones. This also makes our method different from the current NAS paradigm, where we are searching/inserting novel inter-task edges into fixed and well-trained network backbones.

- **Search Algorithm:** We propose a novel single-shot gradient-based search algorithm. It alleviates the performance gap between the search phase and the evaluation phase. We achieve this by imposing minimum entropy regularization on the architecture weights. This enables the architecture mixture to converge to a single model/architecture, which will be directly available for evaluation without the need of architecture pruning or re-training from scratch.

2. Related Works

Neural Architecture Search. Recently, many neural architecture search (NAS) methods have emerged to jointly learn the weights and the network architectures [4, 6, 12, 14, 22, 31, 34, 41, 44, 45, 71]. This is realized by various search algorithms including Bayesian optimization [3], reinforcement learning [18, 58, 72], evolutionary algorithms [49, 63], network transformation [11, 17], and gradient descent [1, 33, 61, 70]. A lot of works employ a single-shot

[1]Note that only the search space is task-agnostic, the searched architectures are flexible and can be different for different task combinations.

learning strategy (*i.e.*, sharing the model weights and sampling different architectures), which drastically reduces the search time [2, 19, 47, 53]. Our work is built on the single-shot gradient-based search methods, where we propose a novel search space and a novel search algorithm tailored to multi-task architecture learning. Our search algorithm is directly related to two popular algorithms in this category, *i.e.*, DARTS [35] and SNAS [64]. We analyze the objective bias in DARTS and the sampling variance in SNAS, and propose a unified solution, *i.e.*, entropy minimization, to alleviate both the issues. More recently, MTL and multi-modal learning have been exploited using NAS [32, 45]. Our method is different from both of them, *i.e.*, we disentangle the problem into task-specific backbones and general inter-task feature fusion connections, which enables us to design a unified search space to realize *general-purpose* MTL-NAS.

Multi-Task Learning. Multi-task learning has been integrated with deep neural networks to boost performance by learning multiple tasks simultaneously. A great success has been witnessed in various areas such as detection [15, 16, 23, 50, 54, 55], depth prediction and semantic segmentation (also surface normal prediction) [10, 65], human-related tasks [21, 48, 60, 62, 68], *etc*. Our work leverages NAS to move towards GP-MTL, which (or whose search space) is compatible with any task combinations [27, 29, 36–38, 40, 42, 52, 67]. Our method (or more specifically, our search space design) is mainly inspired by recent researches including cross-stitch networks [42] and NDDR-CNN [13], which enables to treat GP-MTL as an inter-task feature fusion scheme decoupled from the task-specific backbone networks. Our method extends [13, 42] by allowing fusion of an arbitrary number of source features. Moreover, rather than heuristically inserting the fusion operation into selected layers in [13, 42], our method can automatically learn better fusion positions.

3. Problem Formulation

We introduce our problem formulation in this section. Firstly, we show how to dissect the GP-MTL problem by disentangling the task-specific backbone networks and general inter-task feature fusion in Sect. 3.1. In Sect. 3.2, we formally present our task-agnostic search space, which is unified for different task combinations. Finally, we detail our choice of feature fusion operation in Sect. 3.3.

3.1. Task-Specific Backbone Networks and General Inter-Task Feature Fusion

Arguably, the main difficulty to integrate NAS into GP-MTL is the difference in the designed search spaces for different tasks. This is because the search space by definition should reflect the inductive bias of the associated task. The situation is even more rigorous for GP-MTL as we have exponentially more task combinations (*i.e.*, task sets).

Our strategy is to disentangle multi-task architecture into a (general) shared structure involving feature fusion connections and the task-specific sub-networks, and optimize the shared structure with NAS. Recent researches in GP-MTL [13, 42] inspire us to formulate the single-task backbones as the task-specific parts, while focusing on designing the feature fusion scheme with a general *task-agnostic* search space which is independent of different task combinations. We illustrate the fixed single-task backbone networks and the learnable inter-task fusion edges in the leftmost subfigure of Fig. 1.

3.2. Search Space

We formally describe the task-agnostic search space for GP-MTL based on the above discussion. We consider the same GP-MTL scenario as described by in [42], where two tasks A and B share the same input. Our goal is to construct a multi-task network by learning good inter-task feature fusion edges on two well-trained single-task networks.

We aim to search a direct-acyclic graph (DAG) by adding directed edges at every intermediate layer of each fixed single-task network (please see the leftmost subfigure of Fig. 1). Every directed edge (*i.e.*, computations) points from the *source* features to the *target* features. We have two types of source features in the GP-MTL framework. Considering learning on Task A, we denote the source features from the same task A as the *task-identical source* features. The source features from the other task(s) are *task-opposite source* features, which provide complementary features from the opposite task. We fix the *task-identical source* edge and search over the complete set of possible *task-opposite source* edges.

Formally, we aim to calculate the optimal fused feature at the j^{th} *target* layer O_j, exploiting the *task-identical source* features F_j^{TI} and the completed set of the candidate *task-opposite source* features $\mathcal{S}^{\text{TO}}$. The construction of $\mathcal{S}^{\text{TO}}$ determines the extent of our search space. In order to avoid creating cycles in the resulting multi-task architecture, we limit the indices of the candidate *task-opposite source* features to be not larger than j. We denote such a limited candidate *task-opposite source* feature set as $\mathcal{S}_j^{\text{TO}} = [F_0^{\text{TO}}, \cdots, F_j^{\text{TO}}]$. Therefore, our search space associated with the *task-identical source* feature F_j^{TI} can be characterized by a tuple $(\mathcal{S}_j^{\text{TO}}, \mathcal{C})$, where $\mathcal{C}$ is the fusion operations on F_j^{TI} and $\mathcal{S}_j^{\text{TO}}$. Finally, the optimal fused feature O_j is:

$$O_j = \mathcal{C}(F_j^{\text{TI}}, \mathcal{S}_j^{\text{TO}}) = G\Big(H\big([F_j^{\text{TI}}, z_{0j}\mathtt{R}(F_0^{\text{TO}}), ..., z_{jj}\mathtt{R}(F_j^{\text{TO}})]\big)\Big), \tag{1}$$

where $\mathcal{C} = \{G, H\}$ with G being the nonlinear activation and H being the feature transformations. $\mathtt{R}$ is the spatial resizing operation (*e.g.*, bilinear interpolation) to enable concatenation. Each z_{ij} is a binary indicator denoting if there is

$$O_j = \texttt{ReLU}\Big(\texttt{BN}\big([w_{\texttt{TI}}\mathbb{I}, \overbrace{w_{\texttt{TO}}\mathbb{I}, ..., w_{\texttt{TO}}\mathbb{I}}^{j \text{ elements}}][F_j^{\texttt{TI}}, z_{0j}\texttt{R}(F_0^{\texttt{TO}}), ..., z_{jj}\texttt{R}(F_j^{\texttt{TO}})]^\top\big)\Big) = \texttt{ReLU}\Big(\texttt{BN}\big([w_{\texttt{TI}}F_j^t + w_{\texttt{TO}}\sum_{i=1}^{j} z_{jj}\texttt{R}(F_j^{\texttt{TO}})]\big)\Big) \tag{3}$$

an edge from the i-th task-opposite source node to the j-th target node, which will be optimized by the NAS algorithm.

For two tasks with n layers for each, this general search space can produce $2^{n(n+1)}$ candidate fusion architectures, including the state-of-the-art NDDR-CNN [13] and cross-stitch networks [42] as the special cases.

3.3. Inter-Task Feature Fusion Operation

We follow NDDR-CNN [13] to design our feature transformation $H = \{\texttt{1x1 Conv}(\cdot)\}$ and our nonlinear activation $G = \{\texttt{ReLU}\big(\texttt{BN}(\cdot)\big)\}$. Note that our fusion operation in Eq. (1) generalizes that of NDDR-CNN as we can take an *arbitrary* number of input features, which enables the convergence to a *heterogeneous and asymmetric* architecture[2]. Our fusion operation in Eq. (1) becomes the following Eq. (2), which is also shown in the right-top subfigure of Fig. 1:

$$O_j = \texttt{ReLU}\Big(\texttt{BN}\big(\texttt{1x1Conv}[F_j^{\texttt{TI}}, z_{0j}\texttt{R}(F_0^{\texttt{TO}}), ..., z_{jj}\texttt{R}(F_j^{\texttt{TO}})]\big)\Big). \tag{2}$$

We also note that the initialization of H (*i.e.*, the 1×1 convolution) is important, as we are inserting novel edges into fixed and well-trained single-task backbones. Therefore, we should avoid severe changes of the original single-task output at each layer. Formally, we show the initialization for the operation in Eq. (3).

In Eq. (3), $\mathbb{I}$ is an identity matrix which enables us to focus on only initializing the block-diagonal elements in the 1×1 convolution, $w_{\texttt{TI}}$ and $w_{\texttt{TO}}$ are the initialized weights for the *task-identical source* and the *task-opposite source* features, respectively. We empirically set $w_{\texttt{TI}} + jw_{\texttt{TO}} = 1$ and initialize a large $w_{\texttt{TI}}$ similar to [13, 42]. The initialization is illustrated in the right-bottom subfigure of Fig. 1.

4. Search Algorithm

In this section, we present our single-shot gradient-based search algorithm, which optimizes the model weights and the architecture weights over the meta-network (*i.e.*, the network includes all legal connections defined by the search space) by gradient descent. Our method is able to alleviate the performance gap between the searched and the evaluation architectures, where the performance gap was caused by the inconsistency between the searched mixture architectures and the derived single evaluation architectures in previous single-shot gradient-based search algorithms.

[2]Note that it is easy to incorporate more candidate operations into H and G. We have tested including summation into H but witnessed a negligible improvement. Therefore, we fix the design of H and G for simplicity.

Fundamentally, the undesirable inconsistency in the searched and evaluation architectures is introduced by the **continuous relaxation** and the **discretization** procedures of the single-shot gradient-based search algorithms. In order to better understand this, we first discuss the *continuous relaxation* and the *discretization* in Sect. 4.1. Based on that, in Sect. 4.2, we analyze the objective bias caused by the inconsistency between the deterministic continuous relaxation and deterministic discretization. We note that stochastic SNAS [64] was proposed to solve the objective bias, but it may introduce a large sampling variance, leading to an unstable evaluation performance. Finally, we propose the minimum entropy regularization to alleviate both the issues and present our optimization procedure in Sect. 4.3.

4.1. Continuous Relaxation and Discretization

Typical single-shot gradient-based methods usually contain two stages, *i.e.*, i) continuous relaxation and ii) discretization. The continuous relaxation enables the gradients to be calculated and back-propagated to search the architecture, as the original objective of the NAS methods is discrete and non-differentiable. As the search phase often converges to a mixture model (with many architecture weights between 0 and 1), we thus need to derive a single child network for evaluation by discretization.

We denote the connectivity of the network $\boldsymbol{Z}$ as a set of random variables Z_{ij}, where Z_{ij} is sampled from some discrete distribution parameterized by the *architecture weights* α_{ij}, *i.e.*, $\boldsymbol{Z} = \{Z_{ij} \sim \texttt{DisDist}(\alpha_{ij}) \mid \forall(i,j) \text{ in the search space}\}$, where i, j refer to the source node (*i.e.*, the input position) and the target node (*i.e.*, the output position to perform the associated operations), respectively. Here, the discrete sampling distribution can be `Categorical` or `Bernoulli`, depending on whether there are multiple or only one candidate operations to search. We use a Bernoulli distribution to present our problem as there is only the NDDR feature fusion operation in our search space (see Sect. 3.3 and Eq. (2)), but note that the proposed search algorithm is general and can also be used with multiple candidate operations.

We denote the multivariate sampling distribution for all the fusion connections as $p_{\boldsymbol{\alpha}}(\cdot)$, where $\boldsymbol{\alpha} = \{\alpha_{ij} \mid \forall(i,j) \text{ in the search space}\}$. The architecture search objective is [46, 64]:

$$\min_{\boldsymbol{\alpha}} \mathbb{E}_{\boldsymbol{Z}\sim p_{\boldsymbol{\alpha}}(\boldsymbol{Z})}[\mathcal{L}_\theta(\boldsymbol{Z})], \tag{3}$$

where θ is a set of CNN weights and $\mathcal{L}_\theta(\boldsymbol{Z})$ is the loss func-

tion of the sampled (discrete) architecture $\boldsymbol{Z}$.

In order to optimize $\boldsymbol{\alpha}$ with gradient based methods, one solution is to relax the discrete sampling procedure $\boldsymbol{Z} \sim p_{\boldsymbol{\alpha}}(\boldsymbol{Z})$ to be continuous:

Deterministic Relaxation. The deterministic approaches such as DARTS directly maintain and optimize a mixture of architectures. For Bernoulli random variable $Z_{ij} \sim \texttt{Ber}(\alpha_{ij})$, it directly uses the distribution mean instead of the discrete samples, *i.e.*, $Z_{ij} = \alpha_{ij}$. Therefore, the relaxed objective of Eq. (3) and fusion operation of Eq. (1) become:

$$\min_{\boldsymbol{\alpha}} \mathcal{L}_\theta(\boldsymbol{\alpha}), \tag{4}$$

$$O_j = G\Big(H\big([F_j^{\text{TI}}, \alpha_{0j}\texttt{R}(F_0^{\text{TO}}), ..., \alpha_{jj}\texttt{R}(F_j^{\text{TO}})]\big)\Big). \tag{5}$$

Stochastic Relaxation. SNAS [64] uses the reparameterization trick with the `Concrete` distribution [39] to sample architectures during search while allowing gradient to back-propagate through the sampling procedure. The reparameterized multivariate `Bernoulli` is:

$$\boldsymbol{X} = \frac{1}{1+\exp(-(\log(\boldsymbol{\alpha}) + \boldsymbol{L})/\tau)} \sim q_{\boldsymbol{\alpha}}(\boldsymbol{X}), \tag{6}$$

where $\boldsymbol{X}$ is the continuous relaxation of $\boldsymbol{Z}$, *i.e.*, each entry in $\boldsymbol{X}$ takes a continuous value in $[0, 1]$, and $\boldsymbol{L} = (L_1, L_2, \cdots, L_n)$ with $L_i \sim \texttt{Logistic}(0, 1)$. As the temperature τ approaches 0, each element in $\boldsymbol{X}$ smoothly approaches a discrete `Binomial` random variable.

As a result, the relaxed objective of Eq. (3) and the fusion operation of Eq. (1) become:

$$\min_{\boldsymbol{\alpha}} \mathbb{E}_{\boldsymbol{X}\sim q_{\boldsymbol{\alpha}}(\boldsymbol{X})}[\mathcal{L}_\theta(\boldsymbol{X})], \tag{7}$$

$$O_j = G\Big(H\big([F_j^{\text{TI}}, x_{0j}\texttt{R}(F_0^{\text{TO}}), ..., x_{jj}\texttt{R}(F_j^{\text{TO}})]\big)\Big), \tag{8}$$

where x_{ij} is the sampled connectivity from source node i to target node j, *i.e.*, $\boldsymbol{X} = \{x_{ij} \mid \forall(i, j) \text{ in the search space}\}$.

The search optimizations on both the relaxed objectives *converge to a mixture model with* $\alpha_{ij} \in [0, 1]$, which require discretizations to derive a single child model for evaluation:

Deterministic Discretization. This is the discretization method used in DARTS, which simply keeps the connection with the highest architecture weight. For our binary case:

$$\texttt{Ind}(\alpha_{ij}) = \begin{cases} 1, & \text{if } \alpha_{ij} > 0.5, \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

where $\texttt{Ind}(\cdot)$ is an indicator function.

Therefore, the final discretizated child network is:

$$O_j = G\Big(H\big([F_j^{\text{TI}}, \texttt{Ind}(\alpha_{0j})\texttt{R}(F_0^{\text{TO}}), ..., \texttt{Ind}(\alpha_{jj})\texttt{R}(F_j^{\text{TO}})]\big)\Big). \tag{10}$$

Stochastic Discretization. The discretization of SNAS has already taken place during the search optimization Eq. (7). After the search converges, SNAS samples a child architecture of each x_{ij} according to Eq. (6) with the converged $\boldsymbol{\alpha}$ and $\tau = 0$, resulting in the same form as Eq. (8).

4.2. Objective Bias and Sampling Variance

Objective Bias of the Deterministic Method. The inconsistency between Eqs. (5) and (10) introduces an objective bias between the relaxed parent and the discretized child:

$$\Big|\mathcal{L}_\theta(\boldsymbol{\alpha}) - \mathcal{L}_\theta(\texttt{Ind}(\boldsymbol{\alpha}))\Big| \geq 0, \tag{11}$$

where $\mathcal{L}_\theta(\boldsymbol{\alpha})$ is the objective for the search optimization, and $\mathcal{L}_\theta(\texttt{Ind}(\boldsymbol{\alpha}))$ is the *true* objective we aim to minimize with the actual *evaluation architecture* $\texttt{Ind}(\boldsymbol{\alpha})$.

Remark 1 *Due to the complex and architecture-dependent nature of* $\mathcal{L}_\theta$*, it is difficult to deduce all cases where* $\mathcal{L}_\theta(\boldsymbol{\alpha}) = \mathcal{L}_\theta(\texttt{Ind}(\boldsymbol{\alpha}))$*. Instead, with a well-defined* $\mathcal{L}_\theta$*,* i.e.*,* $x = y \implies \mathcal{L}_\theta(x) = \mathcal{L}_\theta(y)$ *and the local Lipschitz continuity near* $\texttt{Ind}(\boldsymbol{\alpha})$*, we can simply force* $\boldsymbol{\alpha}$ *close to* 0 *or* 1*, therefore achieving* $\boldsymbol{\alpha} = \texttt{Ind}(\boldsymbol{\alpha})$ *and ultimately* $\mathcal{L}_\theta(\boldsymbol{\alpha}) = \mathcal{L}_\theta(\texttt{Ind}(\boldsymbol{\alpha}))$*. We achieve this by applying minimum entropy regularization on* $\boldsymbol{\alpha}$*.*

Sampling Variance of the Stochastic Method. There is no objective bias in the stochastic method when $\tau = 0$, as the training and evaluation objectives align to Eq. (8)[3]. However, there does exist a (probably large) variance when sampling the child variance after convergence.

As shown in the `Concrete` distribution [39], when the temperature τ is annealed to 0, x_{ij} is sampled to 1 with probability α_{ij}, *i.e.*, $\lim_{\tau\to 0} P(x_{ij} = 1) = \alpha_{ij}$. Note that α_{ij} is not well regularized during the search optimization. When α_{ij} converges to 0.5, the sampling of x_{ij} randomly picks 0 or 1, which leads to an unstable evaluation. Our empirical results in Fig. 2 show that this can happen in practice.

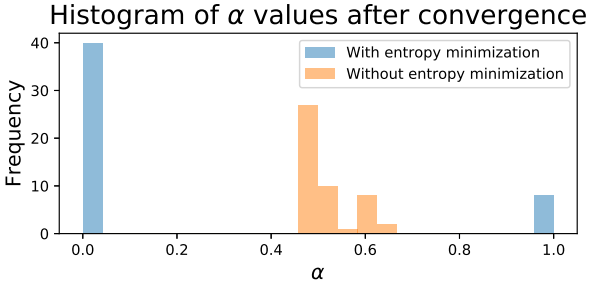

Figure 2. The histogram of converged α from *stochastic* continuous relaxation with and without *minimum entropy regularization*. This shows that the minimum entropy regularization efficiently regularizes the distribution of α's, therefore leading to a smaller variance in sampling $\boldsymbol{X}$ by Eq. (6). We draw 25 bins uniformly from 0 to 1, where each bin represents an interval with 0.04 (best viewed in color).

Remark 2 *It is desirable to have explicit regularization on* α_{ij} *to avoid it from converging around* 0.5*, so as to reduce the sampling variance. Interestingly, such a motivation also*

[3]Note that the bias exists during the search phase when τ is not 0 [39].

leads us to minimize the uncertainty of α_{ij}. We implement this also by minimum entropy regularization on α_{ij}.

In summary, the proposed minimum entropy regularization alleviates both the objective bias of the deterministic methods and the sampling variance of the stochastic methods, which makes the search optimization converge to a single model that is ready for evaluation.

4.3. Losses and Optimization

After imposing the minimum entropy regularization, the full losses of our problem for both deterministic and stochastic versions are shown in Eqs. (12) and (13):

$$\mathcal{L}_\theta(\boldsymbol{\alpha}) = \mathcal{L}_\theta^A(\boldsymbol{\alpha}) + \lambda \mathcal{L}_\theta^B(\boldsymbol{\alpha}) + \frac{\gamma}{n}\sum_{i,j}\mathcal{H}(\alpha_{ij}), \quad (12)$$

$$\mathcal{L}_\theta(\boldsymbol{X}) = \mathcal{L}_\theta^A(\boldsymbol{X}) + \lambda \mathcal{L}_\theta^B(\boldsymbol{X}) + \frac{\gamma}{n}\sum_{i,j}\mathcal{H}(\alpha_{ij}), \quad (13)$$

where $\mathcal{H}(\alpha_{ij}) = -\alpha_{ij}\log\alpha_{ij} - (1-\alpha_{ij})\log(1-\alpha_{ij})$ is the entropy of α_{ij}, γ is the regularization weight, and $\mathcal{L}_\theta^A$ and $\mathcal{L}_\theta^B$ are the losses for tasks A and B. $\boldsymbol{X}$ for the stochastic method is sampled from $q_{\boldsymbol{\alpha}}(\boldsymbol{X})$ in Eq. (6).

Our optimization for $\boldsymbol{\alpha}$ and θ iterates the following steps:

1. Sample two distinct batches of training data $\mathcal{X}_1 = \{x_i\}_{i=1}^n$ with label $\mathcal{Y}_1 = \{y_i\}_{i=1}^n$ and $\mathcal{X}_2 = \{x_j\}_{j=1}^n$ with label $\mathcal{Y}_2 = \{y_j\}_{j=1}^n$, where $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$.
2. Compute the network output $\mathcal{O}_1$ of $\mathcal{X}_1$ with either deterministic Eq. (5) or stochastic Eq. (8) given current $\boldsymbol{\alpha}$ and θ, and then update θ by $\nabla_\theta \mathcal{L}(\mathcal{O}_1, \mathcal{Y}_1)$.
3. Compute the network output $\mathcal{O}_2$ of $\mathcal{X}_2$ with either deterministic Eq. (5) or stochastic Eq. (8) given current $\boldsymbol{\alpha}$ and θ, and then update $\boldsymbol{\alpha}$ by $\nabla_{\boldsymbol{\alpha}} \mathcal{L}(\mathcal{O}_2, \mathcal{Y}_2)$.

Note that once the above iterations converge, the architecture, along with the model weights, can be used directly for evaluation, without the need of (re-)training the model weights from scratch. This is because that the proposed minimal entropy regularization enables the search optimization to converge to a single network. The overall procedure of our method is shown in Fig. 3.

Connections to DARTS and SNAS. Our method alleviates the objective bias in DARTS and the sampling variance in SNAS. This is achieved by the unified, simple, and effective (verified in the ablation section) minimum entropy regularization on the architecture weight α_{ij}.

Our analysis also enables to re-incorporate the continuous relaxation methods and the discretization in a novel way, *e.g.*, *deterministic* continuous relaxation plus *stochastic* discretization, or *stochastic* continuous relaxation plus *deterministic* discretization[4].

[4]Though we show in the supplementary material that the different combinations make an insignificant performance difference, we hope that the analysis in this section sheds light on inventing novel NAS methods.

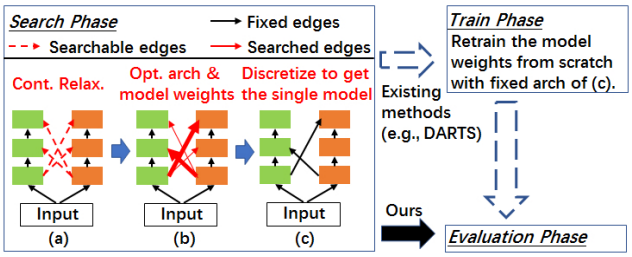

Figure 3. The overall procedure of the proposed search algorithm. The line width of the searched edges (solid red lines) indicate the converged architecture weights. *Our method incorporates minimal entropy regularization on the architecture weights during the search phase, which makes the search optimization (b) converge closely to (c), therefore avoiding the train phase and achieving a better performance with less training time*. On the contrary, without architecture weights entropy minimization (*e.g.*, DARTS), (b) and (c) can be very different, which requires retraining the model weights of (c) from scratch (best viewed in color).

5. Experiments

In this section, we would like to investigate i) the **performance**, *i.e.*, how the proposed MTL-NAS performs against the state-of-the-art *general-purpose* multi-task networks, and ii) the **generalizability**, *i.e.*, how the proposed MTL-NAS generalizes to various *datasets*, *network backbones*, and more importantly, various *task sets*.

To validate the performance, we evaluate the proposed method against the state-of-the-art **NDDR-CNN** [13] and **cross-stitch network** [42]. Additionally, we also provide various additional baselines for better evaluation:
Single-task Baseline: it uses the single-task backbones.
Multi-task Baselines: it is the most intuitive multi-task network, which shares all the layers and splits at the last one.
MTL-NAS (SuperNet): it is the supernet of the proposed method before the NAS pruning, where all the intermediate layers from different tasks are connected, as long as they are connectable in the search space. This can also be regarded as a generalized version of the state-of-the-art NDDR-CNN. We leave its results in the supplementary materials.
NDDR-CNN (SuperNet): this is a direct extension of NDDR-CNN analogous to MTL-NAS (SuperNet). The difference from MTL-NAS (SuperNet) is that we only allow interlinks between layers from the same CNN level. We leave its results in the supplementary materials.

We also have extensive configurations to demonstrate the generalizability of our method:
Datasets: the NYU v2 [56] and the Taskonomy [69].
Network Backbones: VGG-16 [57] and ResNet-50 [24].
Task Sets: *pixel labeling tasks* including semantic segmentation and surface normal estimation, and *image level tasks* including object classification and scene classification.

In the following, we first give the implementation details, and then present our results based on different task sets.

5.1. Implementation Details

For the VGG-16 backbone [57], we consider the features from all the convolutional layers. For ResNet-50 [24], we consider the features produced by each bottleneck block. We also constrain the search space due to the hardware limitation, *i.e.*, we require the searchable *task-opposite source* features (*i.e.*, the source features from the opposite task from the target) to satisfy the following rules *w.r.t* the fixed *task-identical source* feature (*i.e.*, the source features from the same task of the target): i) within the same stage, ii) from the same or earlier convolutional layers, and iii) no further by 3 layers apart. This produces search spaces with 2^{24} architectures for VGG-16, and 2^{37} for ResNet-50.

We perform 20000 training iterations for the VGG-16 backbone and 30000 iterations for the ResNet-50 backbone on the NYU v2 dataset [56]. On the Taxonomy dataset [69] (3x larger than ImageNet), we train the network for 90000 steps. We learn the model weights θ via an SGD optimizer with momentum 0.9, weight decay 0.00025, learning rate 0.001 for VGG-16 backbone and 0.00025 for ResNet-50 backbone. We use a poly learning rate decay with a power of 0.9. We optimize architecture weights $\boldsymbol{\alpha}$ via an Adam optimizer [28] with initial learning rate 0.003 and weight decay 0.001. We set $\gamma = 10$ for entropy minimization.

MTL-NAS is proxyless *w.r.t.* both the *datasets* and the *architectures*, *i.e.*, we directly search for the final architecture on the target (large) dataset. Our search time is only 12-13 and 27-28 GPU hours, respectively, for the NYUv2 experiments and the extremely large Taskonomy experiments, with the VGG-16 backbones on a single Nvidia Titan RTX GPU. The searched architectures and model weights can be used directly for evaluation without retraining from scratch.

5.2. Semantic Segmentation and Surface Normal Estimation

We use the NYU v2 dataset [56] for semantic segmentation and surface normal estimation. The groundtruth for 40 classes semantic labeling is from [20] and that for surface normal is precomputed from the depth map [10].

Per-pixel losses are used for both tasks. For semantic segmentation, we use the softmax cross-entropy loss, and calculate the pixel accuracy (PAcc), the mean intersection of union (mIoU) as the evaluation metrics. While for surface normal estimation, we train the network using the cosine loss (indicating the angle difference), and evaluate using Mean and Median angle distances of all the pixels, as well as the percentage of pixels that are within the angles of 11°, 22.5°, and 30° to the ground-truth.

We perform this task on both VGG-16 and ResNet-50 network backbones, which are shown in Tables 1 and 2, respectively. The results show that our method outperforms state-of-the-art methods, demonstrating the effectiveness of the proposed method on semantic segmentation and surface normal estimation with different network backbones.

	Surface Normal Prediction					**Semantic Seg.**	
	Err (↓)		**Within** t° **(%)** (↑)			**(%)** (↑)	
	Mean	Med.	11.25	22.5	30	mIoU	PAcc
Single	15.6	12.3	46.4	75.5	86.5	33.5	64.1
Multiple	15.2	11.7	48.4	76.2	87.0	33.4	64.2
C.-S.	15.2	11.7	48.6	76.0	86.5	34.8	65.0
NDDR	13.9	10.2	53.5	79.5	88.8	36.2	66.4
MTL-NAS	**12.6**	**8.9**	**59.1**	**83.3**	**91.2**	**37.6**	**67.9**

Table 1. Semantic segmentation and surface normal prediction on the NYU v2 dataset using the **VGG-16** network. C.-S. represents the the cross-stitch network. ↑/↓ shows the higher/lower the better.

	Surface Normal Prediction					**Semantic Seg.**	
	Err (↓)		**Within** t° **(%)** (↑)			**(%)** (↑)	
	Mean	Med.	11.25	22.5	30	mIoU	PAcc
Single	16.2	13.6	41.6	74.1	86.5	34.5	65.5
Multiple	16.6	14.2	39.2	73.8	86.5	34.8	65.1
C.-S.	16.6	14.3	39.1	73.7	86.5	34.8	65.7
NDDR	16.4	**12.8**	42.6	73.3	**86.6**	36.7	66.7
MTL-NAS	**16.2**	**12.8**	**44.8**	**73.9**	85.7	**38.6**	**68.6**

Table 2. Semantic segmentation and surface normal prediction on the NYU v2 dataset using the **ResNet-50** network. C.-S. represents the the cross-stitch network. ↑/↓ shows the higher/lower the better.

	Object		**Scene**	
	RecRate (%) (↑)		**RecRate (%)** (↑)	
	Top 1	Top 5	Top 1	Top 5
Single	33.8	63.0	37.8	70.5
Multiple	34.1	66.1	37.8	71.2
Cross-Stitch	33.2	65.2	34.0	70.3
NDDR	32.1	57.7	37.9	71.8
MTL-NAS	**34.8**	**67.0**	**38.2**	**72.5**

Table 3. Object classification and scene classification on the Taskonomy dataset using the VGG-16 network. ↑/↓ shows the higher/lower the better.

5.3. Object Classification and Scene Classification

We evaluate object classification and scene classification tasks on the extremely large Taskonomy dataset [69] (3x larger than ImageNet). We use the tiny split with data collected from 40 buildings. For both object and scene classifications, we use the ℓ_2 distance between model prediction and the soft class probabilities distilled from pretrained networks as the loss. We report the Top-1 and Top-5 recognition rates for both tasks.

The results are shown in Table 3, which exhibit promising performance of our method on a different task set.

6. Ablation Analysis

In this section, we investigate the different choices of the building blocks of the proposed MTL-NAS by ablation analysis. Specifically, we are especially interested in the following questions: i) How does the proposed search algorithm perform *w.r.t.* the baseline methods DARTS [35] and

SNAS [64]? ii) How to initialize the novel inter-task layers into the fixed and well-trained backbones? iii) How to set the learning rate for the novel inter-task layers?

We answer these questions in the following. We put the learning rate analysis in the supplementary materials, which shows that the training of the novel architectures is not sensitive to learning rates. We also include the illustrations of the learned architectures into the supplementary material, demonstrating that the learned architectures are *heterogeneous and asymmetric*, which are arguably very difficult to be discovered by human experts.

We perform all the ablation analyses using the VGG-16 network [57] for semantic segmentation and surface normal estimation on the NYU v2 dataset [56].

6.1. Search Algorithm

In this section, we take a closer investigation on the search algorithm, specifically, the **continuous relaxation**, the **discretization**, and most importantly, the **entropy minimization**. We validate the *entropy minimization associating* with the *deterministic* continuous relaxation and *deterministic* discretization (analogous to DATRS w/o retraining), as well as the *stochastic* continuous relaxation and *stochastic* discretization (analogous to SNAS). We also provide the *random search* baseline for comparison.

The experimental results are shown in Table 4, where both DARTS (w/o retraining, denoted as D in the table) and SNAS (denoted as S in the table) fail in our problem without minimal entropy regularization. We do not report the performance of SNAS due to the large sampling variance (see also Fig. 2). We also perform the stochastic method with minimal entropy regularization for 10 runs and witness negligible performance variance (see also Fig. 2).

Moreover, it is interesting to witness that, after imposing the minimal entropy constraints, the deterministic and stochastic methods produces similar performances. We also perform different combinations of the continuous relaxation and discretization in the supplementary material, *i.e.*, *deterministic* continuous relaxation and *stochastic* discretization, as well as the *stochastic* continuous relaxation and *deterministic* discretization. Those configurations also produce similar results (similar to DARTS plus MinEntropy and SNAS plus MinEntropy), which suggests the potential of our method to unify the popular DARTS and SNAS.

6.2. Weight Initialization for the Novel Layers

We are interested in investigating the initialized weights for the novel inter-task feature fusion layers. As we are inserting novel architectures into the fixed and well-trained single-task backbones, intuitively, we should make minimal changes of the original single-task output at each layer.

The ablation analysis is performed on different initializations of w_t, which is defined in Eq. (3) and illustrated in the bottom right subfigure of Fig. 1. The results shown in Table 5 align our intuition, where initializing w_t with a larger value, *e.g.*, 0.9 or 1.0, produces the best performance.

			Surface Normal Prediction					Semantic Seg.	
			Err (↓)		Within t° (%) (↑)			(%) (↑)	
D	S	E	Mean	Med.	11.25	22.5	30	mIoU	PAcc
Random Search			14.1	10.1	53.9	79.2	88.2	35.3	66.1
✓			15.9	12.7	45.3	74.0	85.9	19.1	46.1
	✓		-	-	-	-	-	-	-
✓		✓	12.7	**8.9**	58.9	83.1	90.9	**37.7**	**67.9**
	✓	✓	**12.6**	**8.9**	**59.1**	**83.3**	**91.2**	37.6	**67.9**

Table 4. Effects of continuous relaxation, discretization, and entropy minimization. D denotes the **deterministic** method (*i.e.*, DARTS without retraining), S means the **stochastic** method (*i.e.*, SNAS), and E represents **minimum entropy regularization**. We do not report the results of the stochastic method as it produces too large sampling variances (see also the corresponding histogram of the converged architecture weights in Fig. 2). ↑/↓ shows the higher/lower the better.

	Surface Normal Prediction					Semantic Seg.	
	Err (↓)		Within t° (%) (↑)			(%) (↑)	
Init $w_{\mathtt{TI}}$	Mean	Med.	11.25	22.5	30	mIoU	PAcc
Random	16.7	12.8	45.4	71.8	82.9	30.3	61.9
0	16.9	12.9	45.1	71.3	82.4	30.1	61.8
0.1	16.6	12.6	45.8	72.2	83.1	31.4	63.1
0.2	14.4	10.6	52.2	78.2	87.8	33.8	65.2
0.5	13.5	9.8	55.2	80.6	89.4	36.7	67.4
0.8	12.9	9.2	57.6	82.6	90.7	37.0	67.6
0.9	**12.6**	9.0	58.8	**83.3**	**91.3**	37.2	67.4
1.0	**12.6**	**8.9**	**59.1**	**83.3**	91.2	**37.6**	**67.9**

Table 5. Effects of different initializations of the 1x1 convolution in the fusing operation. w is defined in Eq. (3), also in the bottom right subfigure of Fig. 1. ↑/↓ shows the higher/lower the better.

7. Conclusion

In this paper, we employed NAS for general-purpose multi-task learning (GP-MTL). We first disentangled GP-MTL into the task-specific backbone and inter-task feature fusion connections. We then focused on searching for a good inter-task feature fusion strategy within a task agnostic search space. We also proposed a novel search algorithm that is able to close the performance gap between search and evaluation. Our search algorithm also generalizes the popular single-shot gradient-based methods such as DARTS and SNAS. We conducted detailed ablation analysis to validate the effect of each proposed component. The extensive experiments demonstrate the promising performance and the desirable generalizability (to various datasets, task sets, and single-task backbones) of the proposed method.

Acknowledgments. This work was partially supported by NSFC 61773295 and 61771201, NSF of Hubei Province 2019CFA037, and Guangdong R&D key project of China 2019B010155001.

References

[1] Youhei Akimoto, Shinichi Shirakawa, Nozomu Yoshinari, Kento Uchida, Shota Saito, and Kouhei Nishida. Adaptive stochastic natural gradient method for one-shot neural architecture search. In *ICML*, 2019. 2

[2] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *ICML*, 2018. 3

[3] J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *ICML*, 2013. 2

[4] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *ICLR*, 2019. 2

[5] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *ICCV*, 2019. 1

[6] Yukang Chen, Gaofeng Meng, Qian Zhang, Shiming Xiang, Chang Huang, Lisen Mu, and Xinggang Wang. Renas: Reinforced evolutionary neural architecture search. In *CVPR*, 2019. 2

[7] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Chunhong Pan, and Jian Sun. Detnas: Neural architecture search on object detection. *arXiv preprint arXiv:1903.10979*, 2019. 1

[8] Xuanyi Dong and Yi Yang. One-shot neural architecture search via self-evaluated template network. In *ICCV*, 2019. 1

[9] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *CVPR*, 2019. 1

[10] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *CVPR*, 2015. 3, 7

[11] Thomas Elsken, Jan-Hendrik Metzen, and Frank Hutter. Simple and efficient architecture search for convolutional neural networks. *arXiv preprint arXiv:1711.04528*, 2017. 2

[12] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*, 2018. 2

[13] Yuan Gao, Jiayi Ma, Mingbo Zhao, Wei Liu, and Alan L Yuille. Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In *CVPR*, 2019. 1, 2, 3, 4, 6

[14] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*, 2019. 1, 2

[15] Ross Girshick. Fast R-CNN. In *ICCV*, 2015. 3

[16] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 3

[17] Ariel Gordon, Elad Eban, Ofir Nachum, Bo Chen, Hao Wu, Tien-Ju Yang, and Edward Choi. Morphnet: Fast and simple resource-constrained structure learning of deep networks. In *CVPR*, 2018. 2

[18] Minghao Guo, Zhao Zhong, Wei Wu, Dahua Lin, and Junjie Yan. Irlas: Inverse reinforcement learning for architecture search. In *CVPR*, 2019. 2

[19] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. *arXiv preprint arXiv:1904.00420*, 2019. 3

[20] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual organization and recognition of indoor scenes from RGB-D images. In *CVPR*, 2013. 7

[21] Hu Han, Anil K Jain, Shiguang Shan, and Xilin Chen. Heterogeneous face attribute estimation: A deep multi-task learning approach. *TPAMI*, 2018. 3

[22] Chaoyang He, Haishan Ye, Li Shen, and Tong Zhang. Milenas: Efficient neural architecture search via mixed-level reformulation. In *CVPR*, 2020. 2

[23] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 3

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 6, 7

[25] Hanzhang Hu, John Langford, Rich Caruana, Saurajit Mukherjee, Eric Horvitz, and Debadeepta Dey. Efficient forward architecture search. In *NeurIPS*, 2019. 1

[26] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017. 1

[27] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018. 3

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7

[29] Iasonas Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *CVPR*, 2017. 1, 3

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[31] Xin Li, Yiming Zhou, Zheng Pan, and Jiashi Feng. Partial order pruning: For best speed/accuracy trade-off in neural architecture search. In *CVPR*, 2019. 2

[32] Jason Liang, Elliot Meyerson, and Risto Miikkulainen. Evolutionary architecture search for deep multitask networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018. 3

[33] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L. Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *CVPR*, 2019. 1, 2

[34] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *ECCV*, 2018. 2

[35] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *ICLR*, 2019. 1, 2, 3, 7

[36] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *CVPR*, 2019. 3

[37] Mingsheng Long and Jianmin Wang. Learning multiple tasks with deep relationship networks. In *NIPS*, 2017. 3

[38] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *CVPR*, 2017. 3

[39] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*, 2016. 5

[40] Kevis-Kokitsi Maninis, Ilija Radosavovic, and Iasonas Kokkinos. Attentive single-tasking of multiple tasks. In *CVPR*, 2019. 3

[41] Jieru Mei, Yingwei Li, Xiaochen Lian, Xiaojie Jin, Linjie Yang, Alan Yuille, and Jianchao Yang. AtomNAS: Fine-grained end-to-end neural architecture search. In *ICLR*, 2020. 2

[42] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *CVPR*, 2016. 1, 2, 3, 4, 6

[43] Niv Nayman, Asaf Noy, Tal Ridnik, Itamar Friedman, Rong Jin, and Lihi Zelnik-Manor. Xnas: Neural architecture search with expert advice. In *NeurIPS*, 2019. 1

[44] Vladimir Nekrasov, Hao Chen, Chunhua Shen, and Ian Reid. Fast neural architecture search of compact semantic segmentation models via auxiliary cells. In *CVPR*, 2019. 1, 2

[45] Juan-Manuel Perez-Rua, Valentin Vielzeuf, Stephane Pateux, Moez Baccouche, and Frederic Jurie. Mfas: Multimodal fusion architecture search. In *CVPR*, 2019. 2, 3

[46] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *ICML*, 2018. 4

[47] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018. 3

[48] Rajeev Ranjan, Vishal M Patel, and Rama Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *TPAMI*, 2017. 3

[49] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *AAAI*, 2019. 2

[50] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 3

[51] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017. 1

[52] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Latent multi-task architecture learning. In *AAAI*, 2019. 3

[53] Shreyas Saxena and Jakob Verbeek. Convolutional neural fabrics. In *NIPS*, 2016. 3

[54] Abhinav Shrivastava and Abhinav Gupta. Contextual Priming and Feedback for Faster R-CNN. In *ECCV*, 2016. 3

[55] Abhinav Shrivastava, Rahul Sukthankar, Jitendra Malik, and Abhinav Gupta. Beyond Skip Connections: Top-Down Modulation for Object Detection. *arXiv:1612.06851*, 2016. 3

[56] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 6, 7, 8

[57] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 6, 7, 8

[58] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, 2019. 2

[59] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 1

[60] Ludovic Trottier, Philippe Giguère, and Brahim Chaib-draa. Multi-task learning by deep collaboration and application in facial landmark detection. *arXiv preprint arXiv:1711.00111*, 2017. 3

[61] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *CVPR*, 2019. 2

[62] Fangting Xia, Peng Wang, Xianjie Chen, and Alan Yuille. Joint multi-person pose estimation and semantic part segmentation. In *CVPR*, 2017. 3

[63] Lingxi Xie and Alan Yuille. Genetic cnn. In *CVPR*, 2017. 2

[64] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: stochastic neural architecture search. In *ICLR*, 2019. 2, 3, 4, 5, 8

[65] Dan Xu, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In *CVPR*, 2018. 3

[66] Hang Xu, Lewei Yao, Wei Zhang, Xiaodan Liang, and Zhenguo Li. Auto-fpn: Automatic network architecture adaptation for object detection beyond classification. In *ICCV*, 2019. 1

[67] Yongxin Yang and Timothy Hospedales. Deep multi-task representation learning: A tensor factorisation approach. In *ICLR*, 2017. 3

[68] Junho Yim, Heechul Jung, ByungIn Yoo, Changkyu Choi, Dusik Park, and Junmo Kim. Rotating your face using multi-task deep neural network. In *CVPR*, 2015. 3

[69] Amir R. Zamir, Alexander Sax, William B. Shen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, 2018. 6, 7

[70] Yiheng Zhang, Zhaofan Qiu, Jingen Liu, Ting Yao, Dong Liu, and Tao Mei. Customizable architecture search for semantic segmentation. In *CVPR*, 2019. 2

[71] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017. 1, 2

[72] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018. 2