

C2FNAS: Coarse-to-Fine Neural Architecture Search for 3D Medical Image Segmentation

Qihang Yu^{1*} Dong Yang² Holger Roth²
Yutong Bai¹ Yixiao Zhang^{1*} Alan L. Yuille¹ Daguang Xu²
¹ The Johns Hopkins University ² NVIDIA

Abstract

3D convolution neural networks (CNN) have been proved very successful in parsing organs or tumours in 3D medical images, but it remains sophisticated and time-consuming to choose or design proper 3D networks given different task contexts. Recently, Neural Architecture Search (NAS) is proposed to solve this problem by searching for the best network architecture automatically. However, the inconsistency between search stage and deployment stage often exists in NAS algorithms due to memory constraints and large search space, which could become more serious when applying NAS to some memory and time-consuming tasks, such as 3D medical image segmentation. In this paper, we propose a **coarse-to-fine neural architecture search (C2FNAS)** to automatically search a 3D segmentation network from scratch without inconsistency on network size or input size. Specifically, we divide the search procedure into two stages: 1) the coarse stage, where we search the macro-level topology of the network, i.e. how each convolution module is connected to other modules; 2) the fine stage, where we search at micro-level for operations in each cell based on previous searched macro-level topology. The coarse-to-fine manner divides the search procedure into two consecutive stages and meanwhile resolves the inconsistency. We evaluate our method on 10 public datasets from Medical Segmentation Decathlon (MSD) challenge, and achieve state-of-the-art performance with the network searched using one dataset, which demonstrates the effectiveness and generalization of our searched models.

1. Introduction

Medical image segmentation is an important prerequisite of computer-aided diagnosis (CAD) which has been applied in a wide range of clinical applications. With the emerging of deep learning, great achievements have been made in this area. However, it remains very difficult

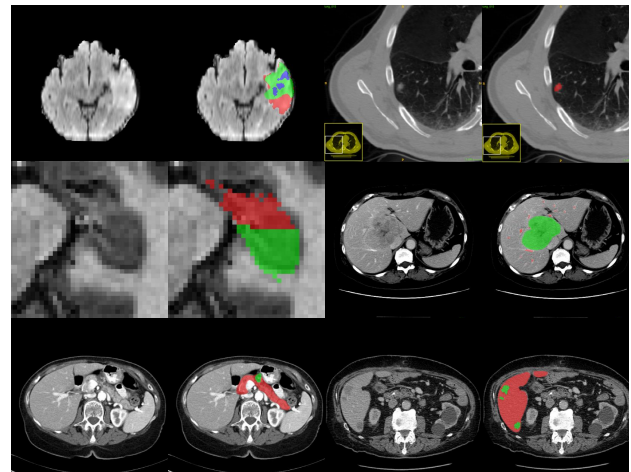


Figure 1. Image and mask examples from MSD tasks (from left to right and top to bottom): brain tumours, lung tumours, hippocampus, hepatic vessel and tumours, pancreas tumours, and liver tumours, respectively. The abnormalities, texture variance, and anisotropic properties make it very challenging to achieve satisfying segmentation performance. Red, green, and blue correspond to labels 1, 2 and 3, respectively, of each dataset.

to get satisfying segmentation for some challenging structures, which could be extremely small with respect to the whole volume, or vary a lot in terms of location, shape, and appearance. Besides, abnormalities, which results in a huge change in texture, and anisotropic property (different voxel spacing) make the segmentation tasks even harder. Some examples are showed in Fig 1.

Meanwhile, manually designing a high-performance 3D segmentation network requires adequate expertise. Most researchers are building upon existing 3D networks, such as 3D U-Net [8] and V-Net [19], with moderate modifications. In some case, an individual network is designed and only works well for certain task. To leverage this problem, Neural Architecture Search (NAS) technique is proposed in [42], which aims at automatically discovering better neural network architectures than human-designed ones in terms of performance, parameters amount, or computation cost. Starting from NASNet [43], many novel search spaces

*Work done during an internship at NVIDIA.

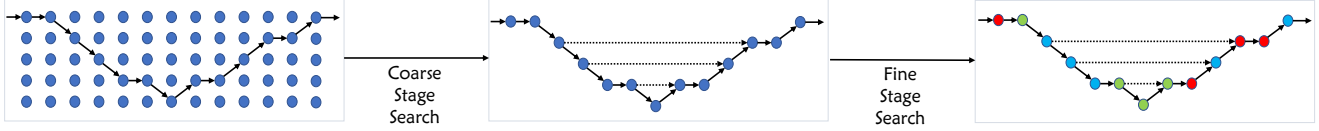


Figure 2. An illustration of proposed C2FNAS. Each path from the left-most node to the right-most node is a candidate architecture. Each color represents one category of operations, *e.g.* depthwise conv, dilated conv, or 2D/3D/P3D conv which are more common in medical image area. The dotted line indicates skip-connections from encoder to decoder. The macro-level topology is determined by coarse stage search, while the micro-level operations are further selected in fine stage search.

and search methods have been proposed [2, 9, 15, 16, 24]. However, only a few works apply NAS on medical image segmentation [13, 33, 40], and they only achieve a comparable performance versus those manually designed networks.

Inspired by the successful handcrafted architectures such as ResNet [11] and MobileNet [28], many NAS works focus on searching for network building blocks. However, such works usually search in a shallow network while deploying with a deeper one. An inconsistency exists in network size between the search stage and deployment stage [6]. [3] and [10] avoided this problem through activating only one path at each iteration, and [6] proposed to progressively reduce search space and enlarge the network in order to reduce the performance gap.

Nevertheless, when the network topology is involved in the search space, things become more complex because no inconsistency is allowed in network size. [15] incorporated the network topology into search space and relieved the memory tensity instead with a sacrifice on batch size and crop size. However, on memory-costly tasks such as 3D medical image segmentation, the memory scarcity cannot be solved by lowering the batch size or cropping size, since they are already very small compared to those of 2D tasks. Reducing them to a smaller number would lead to a much worse performance and even failure on convergence.

To avoid the inconsistency on network size or input size between search stage and deployment stage, we propose a coarse-to-fine neural architecture search scheme for 3D medical image segmentation (see Fig. 2). In detail, we divide the search procedure into coarse stage and fine stage. In the coarse stage, the search is in a small search space with limited network topologies, therefore searching in a train-from-scratch manner is affordable for each network. Moreover, to reduce the search space and make the search procedure more efficient, we constrain the search space under inspirations from successful medical segmentation network designs: (1) U-shape encoder-decoder structure; (2) Skip-connections between the down-sampling paths and the up-sampling paths. The search space is largely reduced with these two priors. Afterwards, we apply a topology-similarity based evolutionary algorithm considering the search space properties, which makes the searching procedure focused on the promising architecture topologies. In the fine stage, the aim is to find the best operations in-

side each cell. Motivated by [40], we let the network itself choose the operation among 2D, 3D and pseudo-3D (P3D), so that it can capture features from different viewpoints. Since the topology is already determined by coarse stage, we mitigate the memory pressure in single-path one-shot NAS manner [10].

For validation, we apply the proposed method on ten segmentation tasks from MSD challenge [30] and achieve state-of-the-art performance. The network is searched using the pancreas dataset which is one of the largest dataset among the 10 tasks. Our result on this proxy dataset surpasses the previous state-of-the-art by a large margin of 1% on pancreas and 2% on pancreas tumours. Then, we apply the same model and training/testing hyper-parameters across the other tasks, demonstrating the robustness and transfer-ability of the searched network.

Our contributions can be summarized into 3 folds: (1) we search a 3D segmentation network from scratch in a coarse-to-fine manner without sacrifice on network size or input size; (2) we design the specific search space and search method for each stage based on medical image segmentation priors; (3) our model achieves state-of-the-art performance on 10 datasets from MSD challenge and shows great robustness and transfer-ability.

2. Related Work

2.1. Medical Image Segmentation

Deep learning based methods have achieved great success in natural image recognition [11], detection [25], and segmentation [5], and they also have been dominating medical image segmentation tasks in recent years. Since U-Net was first introduced in biomedical image segmentation [26], several modifications have been proposed. [8] extended the 2D U-Net to a 3D version. Later, V-Net [19] is proposed to incorporate residual blocks and soft dice loss. [21] introduced attention module to reinforce the U-Net model. Researchers also tried to investigate other possible architectures despite U-Net. For example, [27, 38, 39] cut 3D volumes into 2D slices and handle them with 2D segmentation network. [17] designed a hybrid network by using ResNet50 as 2D encoder and appending 3D decoders afterwards. In [35], 2D predictions are fused by a 3D network to obtain a better prediction with contextual information.

However, until now, U-Net based architectures are still the most powerful models in this area. Recently, [12] introduced nnU-Net and won the first place in Medical Segmentation Decathlon (MSD) Challenge [30]. They ensemble 2D U-Net, 3D U-Net, and cascaded 3D U-Net. The network is able to dynamically adapt itself to any given segmentation task by analysing the data attributes and adjusting hyper-parameters accordingly. The optimal results are achieved with different combinations of the aforementioned networks given various tasks.

2.2. Neural Architecture Search

Neural Architecture Search (NAS) aims at automatically discovering better neural network architectures than human-designed ones. At the beginning stage, most NAS algorithms are based on either reinforcement learning (RL) [1, 42, 43] or evolutionary algorithm (EA) [24, 36]. In RL based methods, a controller is responsible for generating new architectures to train and evaluate, and the controller itself is trained with the architecture accuracy on validation set as reward. In EA based methods, architectures are mutated to produce better off-springs, which are also evaluated by accuracy on validation set. Since parameter sharing scheme was proposed in [23], more search methods were proposed, such as differentiable NAS approaches [16] and one-shot NAS approaches [2], which reduced the search cost to several GPU days or even several GPU hours.

Besides the successes NAS has achieved in natural image recognition, researchers also tried to extend it to other areas such as segmentation [15], detection [9], and attention mechanism [14]. Moreover, there are also some works applying NAS to medical image segmentation area. [40] designed a search space consisting of 2D, 3D, and pseudo-3D (P3D) operations, and let the network itself choose between these operations at each layer. [20, 37] use the policy gradient algorithm for automatically tuning the hyper-parameters and data augmentations. In [13, 33], the cell structure is explored with a pre-defined 3D U-Net topology.

3. Method

3.1. Inconsistency Problem

Early works of NAS [1, 24, 36, 42, 43] typically use a controller based on EA or RL to select network candidates from search space; then the selected architecture is trained and evaluated. Such methods need to train numerous models from scratch and thus lead to an expensive search cost. Recent works [2, 16] propose a differentiable search method that reduces the search cost significantly, where each network is treated as a sub-network of a super-network. However, a critical problem is that the super-network cannot fit into the memory. For these methods, a trade-off is made by sacrificing the network size at search stage and building

a deeper one at deployment, which results in an inconsistency problem. [3] proposed to activate single path of the super-network at each iteration to reduce the memory cost, and [6] proposed to progressively increase the network size with a reduced approximate search space. However, these methods also face problems when the network topology is included in search. For instance, the progressive manner cannot deal with the network topology. As for single-path methods, since there exist illegal paths in network topology, some layers are naturally trained more times compared to others, which results in a serious fairness problem [7].

A straightforward way to solve the issue is to train each candidate from scratch respectively, yet the search cost is too expensive considering the magnitude of search space, which may contain millions of candidates or more. Auto-DeepLab [15] introduces network topology into search space and sacrifices the input size instead of network size at training stage, where it uses a much smaller batch size and crop size. However, it introduces a new inconsistency at input size to solve the old one at network size. Besides, for memory-costly tasks such as 3D medical image segmentation, sacrificing input size is infeasible. The already small input size needs to be reduced to unreasonably smaller to fit the model in memory, which usually leads to an unstable training problem in terms of convergence, and the method only yields a random architecture finally.

3.2. Coarse-to-fine Neural Architecture Search

In order to resolve the inconsistency in network size and input size, and combine NAS with medical image segmentation, we develop a coarse-to-fine neural architecture search method for automatically designing 3D segmentation networks. Without loss of generality, the architecture search space \mathcal{A} consists of topology search space \mathcal{S} , which is represented by a directed acyclic graph (DAG), and cell operation space \mathcal{C} , which is represented by the color of each node in the DAG. Each network candidate is a sub-graph $s \in \mathcal{S}$ with color scheme $c \in \mathcal{C}$ and weights w , denoted as $\mathcal{N}(s, c, w)$.

Therefore, the search space \mathcal{A} is divided into two parts: a small search space of topology \mathcal{S} , and a huge search space of operation \mathcal{C} :

$$\mathcal{A} = \mathcal{S} \times \mathcal{C}. \quad (1)$$

The topology search space is usually small and it is affordable to handle the inconsistency by training each candidate from scratch. For instance, the topology search space \mathcal{S} only has up to 2.9×10^4 candidates for a network with 12 cells [15]. The operation search space \mathcal{C} can have millions of candidates, but since topology s is given, techniques in NAS for recognition, *e.g.* activating only one path at each iteration, are incorporated naturally to solve the memory limitation. Therefore, by regarding neural architecture search

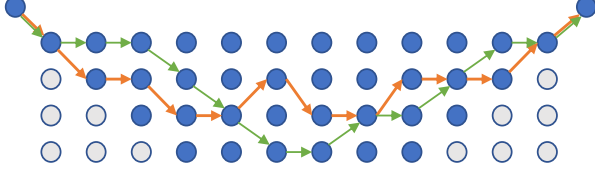


Figure 3. An example of how introduced priors help reduce search space. The grey nodes are eliminated entirely from the graph. Besides, many illegal paths have been pruned off as well. An example of illegal path and legal path is shown as the orange line path and green line path separately.

from scratch as a process of constructing a colored DAG, we divide the search procedure into two stages: (1) Coarse stage: search at macro-level for the network topology and (2) Fine stage: search for the best way to color each node, *i.e.* finding the most suitable operation configuration.

We start with defining macro-level and micro-level. Each network consists of multiple cells, which are composed of several convolutional layers. On macro level, by defining how every cell is connected to each other, the network topology is uniquely determined. Once the topology is determined, we need to define which operation each node represents. On micro-level, we assign an operation to each node, which represents the operation inside the cell, such as standard convolution or dilated convolution.

With this two-stage procedure, we first construct a DAG representing network topology, then assign operations to each cell by coloring the corresponding node in the graph. Therefore, a network is constructed from scratch in a coarse-to-fine manner. By separating the macro-level and micro-level, we relieve the memory pressure and thus resolve the inconsistency problem between search stage and deployment stage.

3.3. Coarse Stage: Macro-level Search

In this stage, we mainly focus on searching the topology of the network. A default operation is assigned to each cell, specifically standard 3D convolution in this paper, and the cell is used as the basic unit to construct the network.

Due to memory constraint and fairness problem, training a super-network and evaluating candidates with a weight-sharing method is infeasible, which means each network needs to be trained from scratch. The search on macro-level is formulated into a bi-level optimization with weight optimization and topology optimization:

$$w_s = \arg \min_w \mathcal{L}_{\text{train}}(\mathcal{N}(s, c_0, w)), \quad (2)$$

$$s^* = \arg \max_{s \in \mathcal{S}} \text{Acc}_{\text{val}}(\mathcal{N}(s, c_0, w_s)), \quad (3)$$

where s represents current topology and c_0 denotes a default coloring scheme, *e.g.* standard 3D convolution every-

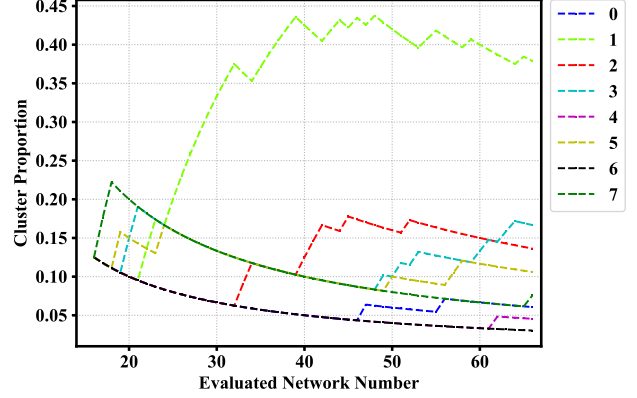


Figure 4. Proportion of clusters sampled during searching at coarse stage. This figure illustrates effectiveness of the proposed evolutionary searching algorithm. Different clusters are in different colors. The x-axis label “Evaluated Network Number” means the total number of networks trained and evaluated, while the y-axis label “Cluster Proportion” is the proportion of number of networks belonging to a specific cluster to the total number of evaluated networks. It is shown that the algorithm gradually focuses on the most promising cluster 1, making the search procedure more efficient.

where, and $\mathcal{L}_{\text{train}}$ is the loss function used at the training stage, and Acc_{val} the accuracy on validation set.

It is extremely time-consuming, especially considering that 3D networks have heavier computation requirements compared with 2D models. Thus, it is necessary to reduce the search space to make the search procedure more focused and efficient.

We revisit the successful medical image segmentation networks, and we find they all share something in common: (1) a U-shape encoder-decoder topology and (2) skip-connections between the down-sampling paths and the up-sampling paths. We incorporate these priors into our method and prune the search space accordingly. An illustration of how the priors help prune search space is shown in Fig. 3. Therefore, the search space \mathcal{S} is pruned to \mathcal{S}' and the topology optimization becomes:

$$\mathcal{S}' = \text{PriorPrune}(\mathcal{S}), \quad (4)$$

$$s^* = \arg \max_{s \in \mathcal{S}'} \text{Acc}_{\text{val}}(\mathcal{N}(s, c_0, w_s)). \quad (5)$$

To further improve the search efficiency, we propose an evolutionary algorithm based on topology similarity to make use of macro-level properties. The idea is that with an assumption of continuous relaxation of topology search space, two similar networks should also share a similar performance. Specifically, we represent each network topology with a code, and we define the network similarity as the euclidean distance between two codes. Smaller the distance is, more similar two networks are. Based on the distance measurement, we classify all network candidates into several clusters with K-means algorithm [18] based on their

Algorithm 1 Topology Similarity based Evolution

```
1:  $population \leftarrow$  all topologies
2:  $\mathcal{P} = \{p_1, p_2, \dots, p_k\} \leftarrow \text{Cluster}(population)$ 
3:  $history \mathcal{H} \leftarrow \emptyset$ 
4: set of trained models  $\mathcal{M} = \{m_1, m_2, \dots, m_k\} \leftarrow \{\emptyset\}^k$ 
5: for  $i = 1$  to  $k$  do
6:    $model.topology \leftarrow \text{RandomSample}(p_i)$ 
7:    $model.accuracy \leftarrow \text{TrainEval}(model.topology)$ 
8:   add  $model$  to  $\mathcal{H}$  and  $m_i$ 
9: while  $|\mathcal{H}| \leq l$  do
10:  while  $\text{HasIdleGPU}()$  do
11:    model for compare  $\mathcal{D} \leftarrow \emptyset$ 
12:    for  $i = 1$  to  $k$  do
13:      add  $\text{RandomSample}(m_i)$  to  $\mathcal{D}$ 
14:      rank  $\mathcal{P}$  based on corresponding accuracy in  $\mathcal{D}$ 
15:       $model.topology \leftarrow \text{SampleUntrained}(p_{rank1})$ 
16:       $model.accuracy \leftarrow \text{TrainEval}(model.topology)$ 
17:      add model to  $\mathcal{H}$  and  $m_{rank1}$ 
18: return highest-accuracy model in  $\mathcal{H}$ 
```

encoded codes. The evolution procedure is prompted in the unit of cluster. In details, when producing next generation, we random sample some networks from each cluster, and rank the clusters by comparing performance of these networks. The higher rank a cluster is, the higher proportion of next generation will come from this cluster. As shown in Fig. 4, the topology proposed by our algorithm gradually falls into the most promising cluster, demonstrating the effectiveness of it. To better make use of computation resources, we further implement this EA algorithm in an asynchronous manner as shown in Algorithm 1.

3.4. Fine Stage: Micro-level Search

After the topology of the network is determined, we further search the model at a fine-grained level by replacing the operations inside each cell. Each cell is a small fully convolutional module, which takes 1 or 2 input tensors and outputs 1 tensors. Since the topology is pre-determined in coarse stage, cell i is simply represented by its operations O_i , which is a subset of the possible operation set \mathcal{O} . Our cell structure is much simpler compared with [15], this is because there is a trade-off between the cell complexity and cell numbers. Given the tense memory requirement of 3D models, we prefer more cells instead of a more complex cell structure.

The set of possible operations, \mathcal{O} , consisting of the following 3 choices: (1) $3 \times 3 \times 3$ 3D convolution; (2) $3 \times 3 \times 1$ followed by $1 \times 1 \times 3$ P3D convolution; (3) $3 \times 3 \times 1$ 2D convolution;

Considering the magnitude of fine stage search space, training each candidate from scratch is infeasible. Therefore, to address the problem of memory limitation while making search efficient, we adopt single-path one-shot NAS

with uniformly sampling [10] as our search method. In details, we construct a super-network where each candidate is a sub-network of it, and then at each iteration of the training procedure, a candidate is uniformly sampled from the super-network and trained and updated. After the training procedure ends, we do random search for final operation configuration. That is to say, at searching stage, we random sample K candidates, and each candidate is initialized with the weights from trained super-network. All these candidates are ranked by validation performance, and the one with the highest accuracy is finally picked.

Therefore, optimization of fine stage is in single-path one-shot NAS manner with uniformly sampling, which is formulated as:

$$w = \arg \min_w \mathbb{E}_{c \in \mathcal{C}} [\mathcal{L}_{\text{train}}(\mathcal{S}(s^*, c, w))], \quad (6)$$

$$c^* = \arg \max_c \text{Acc}_{\text{val}}(\mathcal{S}(s^*, c, w)), \quad (7)$$

where \mathcal{C} is the search space of fine stage, *i.e.* all possibles combinations of operations.

After the coarse stage is finished, the topology s^* is obtained. And the operation configuration c^* comes from the fine stage. Therefore, the final network architecture $\mathcal{N}(s^*, c^*, w)$ is constructed.

4. Experiments

In this section, we firstly introduce our implementation details of C2FNAS, and then report our found architecture (searched on MSD Pancreas dataset) with semantic segmentation results on all 10 MSD datasets [30], which is a public comprehensive benchmark for general-purpose algorithmic validation and testing covering a large span of challenges, such as small data, unbalanced labels, large-ranging object scales, multi-class labels, and multi-modal imaging, *etc.* It contains 10 segmentation datasets, *i.e.* Brain Tumours, Cardiac, Liver Tumours, Hippocampus, Prostate, Lung Tumours, Pancreas Tumours, Hepatic Vessels, Spleen, Colon Cancer.

4.1. Implementation Details

Coarse Stage Search. At coarse stage search, the network has 12 cells at total, where 3 of them are down-sampling cells and 3 up-sampling cells, so that the model size is moderate. With the priors introduced in Section 3, the search space is largely reduced from 2.9×10^4 to 9.24×10^2 .

For network architecture, we define one stem module at the beginning of the network, and another one at the end. The beginning module consists of two 3D $3 \times 3 \times 3$ convolution layers, and strides are 1, 2 respectively. The end module consists of two 3D $3 \times 3 \times 3$ convolution layers, and a trilinear up-sampling layer between the two layers. Each

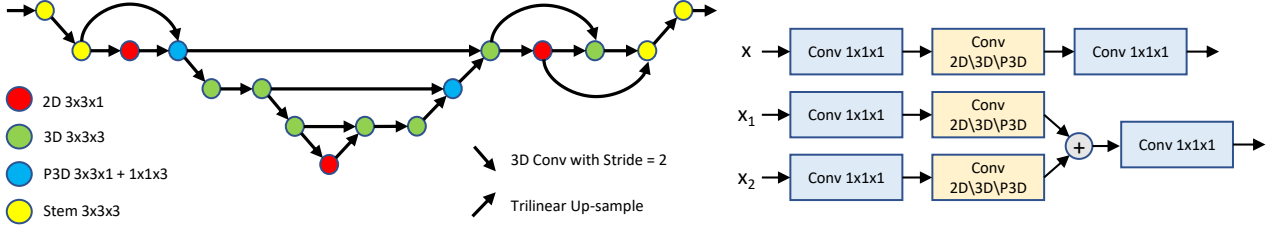


Figure 5. **Left:** The final architecture of C2FNAS-Panc. Red, green, and blue denote cell with 2D, 3D, P3D operations separately. **Right:** The structure of cell with single input and two inputs.

cell takes the output of its previous cell as input, and it will also take another input if it satisfies (1) it has a previous-previous cell at the same feature resolution level, or (2) it is the first cell after an up-sampling. In situation (1), the cell takes its previous-previous cell’s output as additional input. And in situation (2), it takes the output of last cell before the corresponding down-sampling as another input, which serves as the skip-connection from encoder part to decoder part. A convolution with kernel size $1 \times 1 \times 1$ serves as pre-processing for the input. The two inputs go through convolution separately and get summed afterwards, then a $1 \times 1 \times 1$ convolution is applied to the output. The filter number starts with 32, and it is doubled after a down-sampling layer and halved after an up-sampling layer. All down-sampling operations are implemented by a $3 \times 3 \times 3$ 3D convolution with stride 2, and up-sampling by a trilinear interpolation with scale factor 2 followed by a $1 \times 1 \times 1$ convolution. Besides, in coarse stage we also set the operations in all cells to standard 3D convolution with kernel size of $3 \times 3 \times 3$.

For evolutionary algorithm part, we firstly represent each network topology with a code, which is a list of numbers and the length is the same as cell numbers. The number starts at 0 and increases one after a down-sampling and decreases one after an up-sampling. We use K-means algorithm to classify all candidates into 8 clusters based on the Euclidean metric of corresponding codes. At the beginning, two networks are randomly sampled from each cluster. Afterwards, whenever there is an idle GPU, one trained network is sampled from each cluster, and the cluster which the best network belongs to is picked and a new network is sampled from that cluster for training. Meanwhile, the algorithm also random samples a cluster with the probability 0.2 to add randomness and avoid local minimum. After 50 networks are evaluated, the algorithm terminates and returns the best network topology it has found.

We conduct the coarse stage search on the MSD Pancreas Tumours dataset, which contains 282 3D volumes for training and 139 for testing. The dataset is labeled with both pancreatic tumours and normal pancreas region. We divide the training data into 5 folds sequentially, where the first 4 folds for training and last fold for validation purpose. To

address the anisotropic problem, we re-sample all cases to an isotropic resolution with voxel distance 1.0 mm for each axis as data pre-processing.

At training stage, we use batch size of 8 with 8 GPUs, and patch size of $[96, 96, 96]$, where two patches are randomly cropped from each volume at each iteration. All patches are randomly rotated by $[0^\circ, 90^\circ, 180^\circ, 270^\circ]$ and flipped as data augmentation. We use SGD optimizer with learning rate of 0.02, momentum of 0.9, and weight decay of 0.00004. Besides, there is a multi-step learning rate schedule which decay the learning rate at iterations $[8000, 16000]$ with a factor 0.5. We use 1000 iterations for warm-up stage, where the learning rate increases linearly from 0.0025 to 0.02, and 20000 iterations for training. The loss function is the summation of Dice Loss and Cross-Entropy Loss, and we adopt Instance Normalization [32] and ReLU activation function. We also use Horovod [29] to speed up the multi-GPU training procedure.

At validation stage, we test the network in a sliding window manner, where the stride = 16 for all axes. Dice-Sørensen coefficient (DSC) metric is used to measure the performance, which is formulated as $DSC(\mathcal{Y}, \mathcal{Z}) = \frac{2 \times |\mathcal{Y} \cap \mathcal{Z}|}{|\mathcal{Y}| + |\mathcal{Z}|}$, where \mathcal{Y} and \mathcal{Z} denote for the prediction and ground-truth voxels set for a foreground class. The DSC has a range of $[0, 1]$ with 1 implying a perfect prediction.

Fine Stage Search. In the fine stage search, we mainly choose the operations from [2D, 3D, P3D] for each cell. This search space can be large as 5.3×10^5 . Since the search space is numerous, we adopt a single-path one-shot NAS method based on super-network, which is trained by uniformly sampling.

The data pre-processing, data split, and training/validation setting are exactly the same as what we use in coarse stage, except that we double the number of iterations to ensure the super-network convergence. At each iteration, a random path is chosen for training. After the super-network training is finished, we random sample 2000 candidates from the search space, and use the super-network weight to initialize these candidates. Since the validation process takes a very long time due to the

Task	Brain				Liver			Pancreas			Prostate		
Class	1	2	3	Avg	1	2	Avg	1	2	Avg	1	2	Avg
CerebriuDIKU [22]	69.52	43.11	66.74	59.79	94.27	57.25	75.76	71.23	24.98	48.11	69.11	86.34	77.73
Lupin	66.15	41.63	64.15	57.31	94.79	61.40	78.10	75.99	21.24	48.62	72.73	87.62	80.18
NVDLMED [34]	67.52	45.00	68.01	60.18	95.06	71.40	83.23	78.42	38.48	58.45	69.36	86.66	78.01
K.A.V.athlon	66.63	46.62	67.46	60.24	94.74	61.65	78.20	74.97	43.20	59.09	73.42	87.80	80.61
nnU-Net [12]	67.71	47.73	68.16	61.20	95.24	73.71	84.48	79.53	52.27	65.90	75.81	89.59	82.70
C2FNAS-Panc	67.62	48.56	69.09	61.76	94.91	71.63	83.27	80.59	52.87	66.73	73.11	87.43	80.27
C2FNAS-Panc*	67.62	48.60	69.72	61.98	94.98	72.89	83.94	80.76	54.41	67.59	74.88	88.75	81.82

Task	Lung	Heart	Hippocampus			HepaticVessel			Spleen	Colon	Avg (Task)	Avg (Class)
Class	1	1	1	2	Avg	1	2	Avg	1	1		
CerebriuDIKU [22]	58.71	89.47	89.68	88.31	89.00	59.00	38.00	48.50	95.00	28.00	67.01	66.40
Lupin	54.61	91.86	89.66	88.26	88.96	60.00	47.00	53.50	94.00	9.00	65.61	65.89
NVDLMED [34]	52.15	92.46	87.97	86.71	87.34	63.00	64.00	63.50	96.00	56.00	72.73	71.66
K.A.V.athlon	60.56	91.72	89.83	88.52	89.18	62.00	63.00	62.50	97.00	36.00	71.51	70.89
nnU-Net [12]	69.20	92.77	90.37	88.95	89.66	63.00	69.00	66.00	96.00	56.00	76.39	75.00
C2FNAS-Panc	69.47	92.13	86.87	85.44	86.16	63.78	69.41	66.60	96.60	55.68	75.87	74.42
C2FNAS-Panc*	70.44	92.49	89.37	87.96	88.67	64.30	71.00	67.65	96.28	58.90	76.97	75.49

Table 1. Comparison with state-of-the-art methods on MSD challenge test set (number from MSD leaderboard) measured by **Dice-Sørensen coefficient (DSC)**. * denotes the 5-fold model ensemble. The numbers of tasks hepatic vessel, spleen, and colon from other teams are rounded. We also report the average on tasks and on targets respectively for an overall comparison across all tasks/targets.

Model	Params (M)	FLOPs (G)
3D U-Net [8]	19.07	825.30
V-Net [19]	45.59	301.88
VoxResNet [4]	6.92	173.02
ResDSN [41]	10.03	188.37
Attention U-Net [21]	103.88	1162.75
C2FNAS-Panc	17.02	150.78

Table 2. Comparison of parameters and FLOPs with other 3D networks. The FLOPs are calculated based on input size $96 \times 96 \times 96$.

Task	Lung	Pancreas		
Class	1	1	2	Avg
C2FNAS-C-Lung	71.74	80.26	52.51	66.39
C2FNAS-C-Panc	69.05	80.39	53.32	66.86
C2FNAS-F-Panc	69.77	80.37	56.36	68.37

Table 3. Comparison with different stages and different proxy datasets on 5-fold cross-validation.

sliding window method, we increase the stride to 48 at all axes to speed up the search stage.

The coarse search stage takes 5 days with 64 NVIDIA V100 GPUs with 16GB memory. In fine stage, the super-network training costs 10 hours with 8 GPUs, and the searching procedure, where 2000 candidates are evaluated on validation set, takes 1 day with 8 GPUs. The large search cost is mainly because training and evaluating a 3D model itself is very time-consuming.

Deployment Stage. The final network architecture based on the topology searched in coarse stage and operations searched in fine stage is shown in Fig. 5. We keep the training setting same when deploying this network architecture, which means no inconsistency exists in our method.

We use the same training setting mentioned in coarse stage, and the iteration is 40000 and multi-step decay at iterations [16000, 32000]. The model is trained based on same settings from scratch for each dataset, except that Prostate dataset has a very small size on Z (Axial) axis, and Hippocampus dataset has a very small shape around only 50 for each axis. Therefore we change the patch size to $128 \times 128 \times 32$ and stride = [16, 16, 4] for Prostate, and up-sample all data to shape $96 \times 96 \times 96$ for Hippocampus.

4.2. Segmentation Results

We report our test set results of all 10 tasks from MSD challenge and compare with other state-of-the-art methods.

Our test set results are summarized in Table 1. We notice that other methods apply multi-model ensemble to reinforce the performance, *e.g.* nnU-Net ensembles 5 or 10 models based on 5-fold cross-validation with one or two models, NVDLMED and CerebriuDIKU ensemble models trained from different viewpoints. Therefore, besides single-model result, we also report results with a 5-fold cross-validation model ensemble, which means 5 models are trained in a 5-fold cross-validation setting, and final test results are fused with results from these 5 models with a majority voting.

Our model shows superior performance than state-of-the-art methods on most tasks, especially the challenging ones, while enjoying a lighter model size comparing to most popular 3D models (see Table 2). We also has a higher performance in terms of average on task/class. It is noticeable that the previous state-of-the-art nnU-Net uses various kinds of data augmentation and test-time augmentation to boost the performance, while we only adopt simple data augmentation of rotation and flip, and no test-time augmentation is applied. Small datasets such as Heart and Hippocampus rely more on augmentation while a powerful architecture is easy to get over-fitting, which illustrates why our performance on these datasets does not outperform the competitors. Besides, nnU-Net uses different networks and hyper-parameters for each task, while we use the same model and hyper-parameters for all task, showing that our model is not only more powerful but also much more robust and generalizable. Some visualization comparisons are available in Fig. 6.

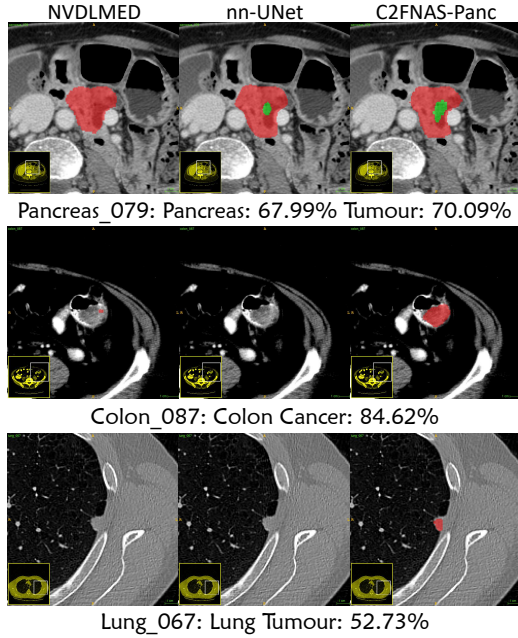


Figure 6. The visualization comparison between state-of-the-art methods (1st and 2nd teams) and C2FNAS-Panc on MSD test sets. We visualize one case from each of the three most challenging tasks: pancreas and pancreas tumours, colon cancer, and lung tumours. Red denotes abnormal pancreas, colon cancer, and lung tumours respectively, and green denotes pancreas tumours. Case id and dice score of C2FNAS-Panc is at the bottom.

5. Ablation Study

5.1. Coarse Stage versus Fine Stage

To verify the improvement of this two-stage design, we compare the performance of network from coarse stage and network from fine stage. The “C2FNAS-C-Panc” indicates the coarse stage network searched on pancreas dataset, where the topology is searched and all operations are in standard 3D manner, while “C2FNAS-F-Panc” is the fine stage network, where the operation configuration is searched. We compare their performance on pancreas and lung dataset with a 5-fold cross-validation. The result is shown in table 3. It is noticeable that the fine stage search not only improves the performance on target dataset (pancreas) but also increases the model generality, thus obtains a better performance on other datasets (lung).

5.2. Search on Different Datasets

Our model is searched on MSD Pancreas dataset, which contains 282 cases, and it is one of the largest dataset in MSD challenge. To verify the data number effect on our method, we also search a model topology on MSD Lung dataset, which contains 64 cases, as ablation study. The search method and hyper-parameters are same as what we use on pancreas dataset. The result is summarized in Ta-

Task	Lung	Pancreas			Hippocampus		
Class	1	1	2	Avg	1	2	Avg
0.25	72.32	79.24	40.02	59.63	80.29	79.81	80.05
0.50	73.89	80.51	46.34	63.43	80.74	80.84	80.79
0.75	76.15	81.40	47.50	64.45	80.88	81.72	81.30
1.00	74.26	80.74	49.94	65.34	81.82	82.10	81.96
1.25	76.94	81.45	48.03	64.74	82.13	82.24	82.19
1.50	75.37	81.40	48.87	65.14	81.02	81.39	81.21
1.75	75.98	81.85	49.03	65.44	81.52	81.31	81.42
2.00	77.75	82.18	50.61	66.40	82.57	82.34	82.46

Table 4. Influence of model scaling, the number in first column indicates the scale factor applied to model C2FNAS-Panc. The results are based on single fold of validation set and the final searched model on pancreas dataset.

ble 3. The “C2FNAS-C-Lung” is the topology on lung dataset, while “C2FNAS-C-Panc” is the topology on pancreas dataset. Topology on lung dataset performs better on lung task, while topology on pancreas dataset performs better on pancreas task. However, it is noticeable that both topologies show good performance on another dataset, demonstrating that our method works well even on a smaller dataset and the models are of great generality.

5.3. Incorporate Model Scaling as Third Stage

Inspired by EfficientNet [31], we add model scaling into the search space as the third search stage. In this ablation study, we only study for scaling of filter numbers for simplicity, but a compound scaling including patch size and cell numbers is feasible. Following [31], we adopt grid search for a channel number multiplier ranging from 0.25 to 2.0 with a step of 0.25. We report the results based on single fold validation set on pancreas and lung dataset respectively, which are summarized in Table 4. It shows that model scaling can increase the model capacity and lead to a better performance. Nevertheless, scaling up the model also results in a much higher model parameters and FLOPs. Considering the large extra computation cost and to keep the model in a moderate size, we do not include model scaling into our main experiment. Yet we report it in ablation study as a potential and promising way to reinforce C2FNAS and achieve even higher performance.

6. Conclusions

In this paper, we propose to use coarse-to-fine neural architecture search to automatically design a transferable 3D segmentation network for 3D medical image segmentation, where the existing NAS methods cannot work well due to the memory-consuming property in 3D segmentation. Besides, our method, with the consistent model and hyper-parameters for all tasks, outperforms MSD champion nnU-Net, a series of well-modified and/or ensembled 2D and 3D U-Net. We do not incorporate any attention module or pyramid module, which means this is a much more powerful 3D backbone model than current popular network architectures.

References

- [1] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *ICLR*, 2017. [2](#), [3](#)
- [2] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Smash: one-shot model architecture search through hypernetworks. *ICLR*, 2018. [2](#), [3](#)
- [3] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *ICLR*, 2019. [2](#), [3](#)
- [4] Hao Chen, Qi Dou, Lequan Yu, Jing Qin, and Pheng-Ann Heng. Voxresnet: Deep voxelwise residual networks for brain segmentation from 3d mr images. *NeuroImage*, 170:446–455, 2018. [7](#)
- [5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *PAMI*, 40(4):834–848, 2018. [2](#)
- [6] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. *ICCV*, 2019. [2](#), [3](#)
- [7] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. *arXiv preprint arXiv:1907.01845*, 2019. [3](#)
- [8] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3D u-net: learning dense volumetric segmentation from sparse annotation. In *MICCAI*, 2016. [1](#), [2](#), [7](#)
- [9] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*, pages 7036–7045, 2019. [2](#), [3](#)
- [10] 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. [2](#), [5](#)
- [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. [2](#)
- [12] Fabian Isensee, Jens Petersen, Andre Klein, David Zimmerer, Paul F Jaeger, Simon Kohl, Jakob Wasserthal, Gregor Koehler, Tobias Norajitra, Sebastian Wirkert, et al. nnu-net: Self-adapting framework for u-net-based medical image segmentation. *arXiv preprint arXiv:1809.10486*, 2018. [3](#), [7](#)
- [13] Sungwoong Kim, Ildoo Kim, Sungbin Lim, Woonhyuk Baek, Chiheon Kim, Hyungjoo Cho, Boogeon Yoon, and Taesup Kim. Scalable neural architecture search for 3d medical image segmentation. *arXiv preprint arXiv:1906.05956*, 2019. [2](#), [3](#)
- [14] Yingwei Li, Xiaojie Jin, Jieru Mei, Xiaochen Lian, Linjie Yang, Cihang Xie, Qihang Yu, Yuyin Zhou, Song Bai, and Alan Yuille. Autonl: Neural architecture search for lightweight non-local networks in mobile vision. In *CVPR*, 2020. [3](#)
- [15] 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*, pages 82–92, 2019. [2](#), [3](#), [5](#)
- [16] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *ICLR*, 2019. [2](#), [3](#)
- [17] Siqi Liu, Daguang Xu, S Kevin Zhou, Olivier Pauly, Sasa Grbic, Thomas Mertelmeier, Julia Wicklein, Anna Jerebko, Weidong Cai, and Dorin Comaniciu. 3d anisotropic hybrid network: Transferring convolutional features from 2d images to 3d anisotropic volumes. In *MICCAI*, pages 851–858. Springer, 2018. [2](#)
- [18] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. [4](#)
- [19] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*, pages 565–571. IEEE, 2016. [1](#), [2](#), [7](#)
- [20] Aliasghar Mortazi and Ulas Bagci. Automatically designing cnn architectures for medical image segmentation. In *MLMI*, pages 98–106. Springer, 2018. [3](#)
- [21] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. Attention u-net: Learning where to look for the pancreas. *MIDL*, 2018. [2](#), [7](#)
- [22] Mathias Perslev, Erik Bjørnager Dam, Akshay Pai, and Christian Igel. One network to segment them all: A general, lightweight system for accurate 3d medical image segmentation. In *MICCAI*, pages 30–38. Springer, 2019. [7](#)
- [23] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *ICML*, 2018. [3](#)
- [24] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI*, volume 33, pages 4780–4789, 2019. [2](#), [3](#)
- [25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015. [2](#)
- [26] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015. [2](#)
- [27] Holger R Roth, Le Lu, Amal Farag, Hoo-Chang Shin, Jiamin Liu, Evrim B Turkbey, and Ronald M Summers. Deeporgan: Multi-level deep convolutional networks for automated pancreas segmentation. In *MICCAI*, 2015. [2](#)
- [28] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018. [2](#)
- [29] Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in tensorflow. *arXiv preprint arXiv:1802.05799*, 2018. [6](#)
- [30] Amber L Simpson, Michela Antonelli, Spyridon Bakas, Michel Bilello, Keyvan Farahani, Bram van Ginneken, Annette Kopp-Schneider, Bennett A Landman, Geert Litjens, Bjoern Menze, et al. A large annotated medical image dataset for the development and evaluation of segmentation algorithms. *arXiv preprint arXiv:1902.09063*, 2019. [2](#), [3](#), [5](#)

- [31] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *ICML*, 2019. 8
- [32] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 6
- [33] Yu Weng, Tianbao Zhou, Yujie Li, and Xiaoyu Qiu. Nas-unet: Neural architecture search for medical image segmentation. *IEEE Access*, 7:44247–44257, 2019. 2, 3
- [34] Yingda Xia, Fengze Liu, Dong Yang, Jinzheng Cai, Lequan Yu, Zhuotun Zhu, Daguang Xu, Alan Yuille, and Holger Roth. 3d semi-supervised learning with uncertainty-aware multi-view co-training. *WACV*, 2020. 7
- [35] Yingda Xia, Lingxi Xie, Fengze Liu, Zhuotun Zhu, Elliot K Fishman, and Alan L Yuille. Bridging the gap between 2d and 3d organ segmentation with volumetric fusion net. In *MICCAI*, pages 445–453. Springer, 2018. 2
- [36] Lingxi Xie and Alan Yuille. Genetic cnn. In *ICCV*, pages 1379–1388, 2017. 3
- [37] Dong Yang, Holger Roth, Ziyue Xu, Fausto Milletari, Ling Zhang, and Daguang Xu. Searching learning strategy with reinforcement learning for 3d medical image segmentation. In *MICCAI*, pages 3–11. Springer, 2019. 3
- [38] Qihang Yu, Lingxi Xie, Yan Wang, Yuyin Zhou, Elliot K Fishman, and Alan L Yuille. Recurrent saliency transformation network: Incorporating multi-stage visual cues for small organ segmentation. In *CVPR*, pages 8280–8289, 2018. 2
- [39] Yuyin Zhou, Lingxi Xie, Wei Shen, Yan Wang, Elliot K Fishman, and Alan L Yuille. A fixed-point model for pancreas segmentation in abdominal ct scans. In *MICCAI*, pages 693–701. Springer, 2017. 2
- [40] Zhuotun Zhu, Chenxi Liu, Dong Yang, Alan Yuille, and Daguang Xu. V-nas: Neural architecture search for volumetric medical image segmentation. *3DV*, 2019. 2, 3
- [41] Zhuotun Zhu, Yingda Xia, Wei Shen, Elliot K. Fishman, and Alan L. Yuille. A 3d coarse-to-fine framework for volumetric medical image segmentation. In *3DV*, 2018. 7
- [42] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *ICLR*, 2017. 1, 3
- [43] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, pages 8697–8710, 2018. 1, 3