DiNTS: Differentiable Neural Network Topology Search for 3D Medical Image Segmentation

Yufan He[1] Dong Yang[2] Holger Roth[2] Can Zhao[2] Daguang Xu[2]
[1]Johns Hopkins University [2]NVIDIA

Abstract

Recently, neural architecture search (NAS) has been applied to automatically search high-performance networks for medical image segmentation. The NAS search space usually contains a network topology level (controlling connections among cells with different spatial scales) and a cell level (operations within each cell). Existing methods either require long searching time for large-scale 3D image datasets, or are limited to pre-defined topologies (such as U-shaped or single-path) . In this work, we focus on three important aspects of NAS in 3D medical image segmentation: flexible multi-path network topology, high search efficiency, and budgeted GPU memory usage. A novel differentiable search framework is proposed to support fast gradient-based search within a highly flexible network topology search space. The discretization of the searched optimal continuous model in differentiable scheme may produce a sub-optimal final discrete model (discretization gap). Therefore, we propose a topology loss to alleviate this problem. In addition, the GPU memory usage for the searched 3D model is limited with budget constraints during search. Our ***Di****fferentiable* ***N****etwork* ***T****opology* ***S****earch scheme (****DiNTS****) is evaluated on the Medical Segmentation Decathlon (MSD) challenge, which contains ten challenging segmentation tasks. Our method achieves the state-of-the-art performance and the top ranking on the MSD challenge leaderboard.*

1. Introduction

Automated medical image segmentation is essential for many clinical applications like finding new biomarkers and monitoring disease progression. The recent developments in deep neural network architectures have achieved great performance improvements in image segmentation. Manually designed networks, like U-Net [34], have been widely used in different tasks. However, the diversity of medical image segmentation tasks could be extremely high since the image characteristics & appearances can be completely distinct for different modalities and the presentation of diseases can vary considerably. This makes the direct application of even a successful network like U-Net [34] to a new task less likely to be optimal.

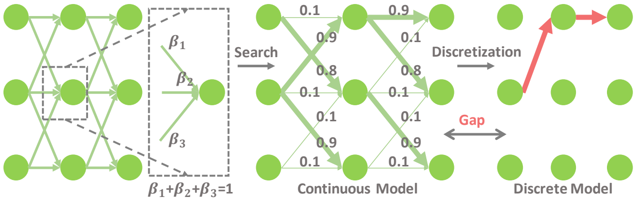


Figure 1. Limitations of existing differentiable topology search formulation. E.g. in Auto-DeepLab [21], each edge in the topology search space is given a probability β. The probabilities of input edges to a node sum to one, which means only one input edge for each node would be selected. A single-path discrete model (red path) is extracted from the continuous searched model. This can result in a large "discretization gap" between the feature flow of the searched continuous model and the final discrete model.

The neural architecture search (NAS) algorithms [49] have been proposed to automatically discover the optimal architectures within a search space. The NAS search space for segmentation usually contains two levels: network topology level and cell level. The network topology controls the connections among cells and decides the flow of the feature maps across different spatial scales. The cell level decides the specific operations on the feature maps. A more flexible search space has more potential to contain better performing architectures.

In terms of the search methods in finding the optimal architecture from the search space, evolutionary or reinforcement learning-based [49, 33] algorithms are usually time consuming. C2FNAS [45] takes 333 GPU days to search one 3D segmentation network using the evolutionary-based methods, which is too computationally expensive for common use cases. Differentiable architecture search [23] is much more efficient and Auto-DeepLab [21] is the first work to apply differentiable search for segmentation network topology. However, Auto-DeepLab's differentiable formulation limits the searched network topology. As shown in Fig. 1, this formulation assumes that only one input edge would be kept for each node. Its final searched model only has a single path from input to output which limits its complexity. Our first goal is to propose a new differentiable scheme to support more complex topologies in order to find novel architectures with better performance.

Meanwhile, the differentiable architecture search suffers from the "discretization gap" problem [4, 38]. The discretization of the searched optimal continuous model may produce a sub-optimal discrete final architecture and cause a large performance gap. As shown in Fig. 1, the gap comes from two sides: 1) the searched continuous model is not binary, thus some operations/edges with small but non-zero probabilities are discarded during the discretization step; 2) the discretization algorithm has topology constraints (e.g. single-path), thus edges causing infeasible topology are not allowed even if they have large probabilities in the continuous model. Alleviating the first problem by encouraging a binarized model during search has been explored [5, 38, 27]. However, alleviating the second problem requires the search to be aware of the discretization algorithm and topology constraints. In this paper, we propose a topology loss in search stage and a topology guaranteed discretization algorithm to mitigate this problem.

In medical image analysis, especially for some longitudinal analysis tasks, high input image resolution and large patch size are usually desired to capture miniscule longitudinal changes. Thus, large GPU memory usage is a major challenge for training with large high resolution 3D images. Most NAS algorithms with computational constraints focus on latency [1, 3, 18, 36] for real-time applications. However, real-time inference often is not a major concern compared to the problem caused by huge GPU memory usage in 3D medical image analysis. In this paper, we propose additional GPU memory constraints in the search stage to limit the GPU usage needed for retraining the searched model.

We validate our method on the Medical Segmentation Decathlon (MSD) dataset [37] which contains 10 representative 3D medical segmentation tasks covering different anatomies and imaging modalities. We achieve state-of-the-art results while only takes 5.8 GPU days (recent C2FNAS [45] takes 333 GPU days on the same dataset). Our contributions can be summarized as:

- We propose a novel **Di**fferentiable **N**etwork **T**opology **S**earch scheme **DiNTS**, which supports more flexible topologies and joint two-level search.
- We propose a topology guaranteed discretization algorithm and a discretization aware topology loss for the search stage to minimize the discretization gap.
- We develop a memory usage aware search method which is able to search 3D networks with different GPU memory requirements.
- We achieve the new state-of-the-art results and top ranking in the MSD challenge leaderboard while only taking 1.7% of the search time compared to the NAS-based C2FNAS [45].

2. Related Work

2.1. Medical Image Segmentation

Medical image segmentation faces some unique challenges like lacking manual labels and vast memory usage for processing 3D high resolution images. Compared to networks used in natural images like DeepLab [2] and PSPNet [46], 2D/3D UNet [34, 6] is better at preserving fine details and memory friendly when applied to 3D images. VNet [26] improves 3D UNet with residual blocks. UNet++ [47] uses dense blocks [13] to redesign skip connections. H-DenseUNet [17] combines 2D and 3D UNet to save memory. nnUNet [14] ensembles 2D, 3D, and cascaded 3D UNet and achieves state-of-the-art results on a variety of medical image segmentation benchmarks.

2.2. Neural Architecture Search

Neural architecture search (NAS) focuses on designing network automatically. The work in NAS can be categorized into three dimensions: search space, search method and performance estimation [8]. The search space defines what architecture can be searched, which can be further divided into network topology level and cell level. For image classification, [23, 50, 22, 33, 31, 11] focus on searching optimal cells and apply a pre-defined network topology while [9, 42] perform search on the network topology. In segmentation, Auto-DeepLab [21] uses a highly flexible search space while FasterSeg [3] proposes a low latency two level search space. Both perform a joint two-level search. In medical image segmentation, NAS-UNet [40], V-NAS [48] and Kim et al [15] search cells and apply it to a U-Net-like topology. C2FNAS [45] searches 3D network topology in a U-shaped space and then searches the operation for each cell. MS-NAS [44] applies PC-Darts [43] and Auto-DeepLab's formulation to 2D medical images.

Search method and performance estimation focus on finding the optimal architecture from the search space. Evolutionary and reinforcement learning has been used in [49, 33] but those methods require extremely long search time. Differentiable methods [23, 21] relax the discrete architecture into continuous representations and allow direct gradient based search. This is magnitudes faster and has been applied in various NAS works [23, 21, 43, 48, 44]. However, converting the continuous representation back to the discrete architecture causes the "discretization gap". To solve this problem, FairDARTS [5] and Tian et al [38] proposed zero-one loss and entropy loss respectively to push the continuous representation close to binary. Some works [27, 12]

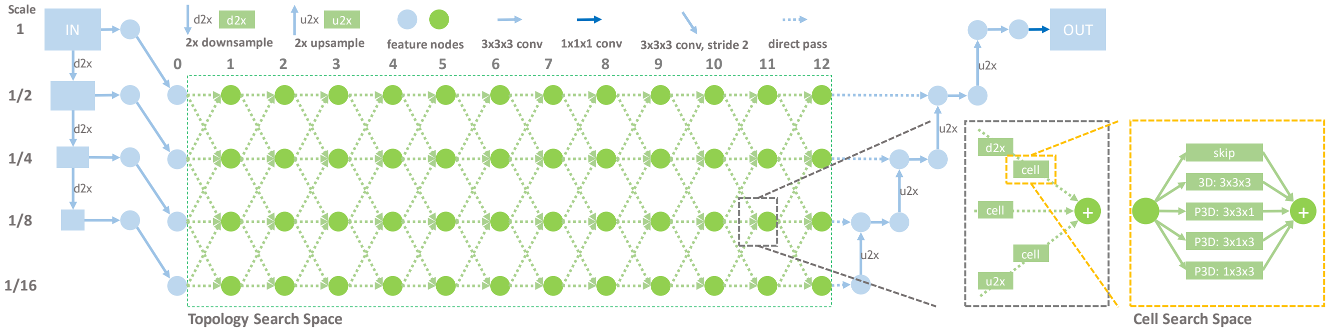


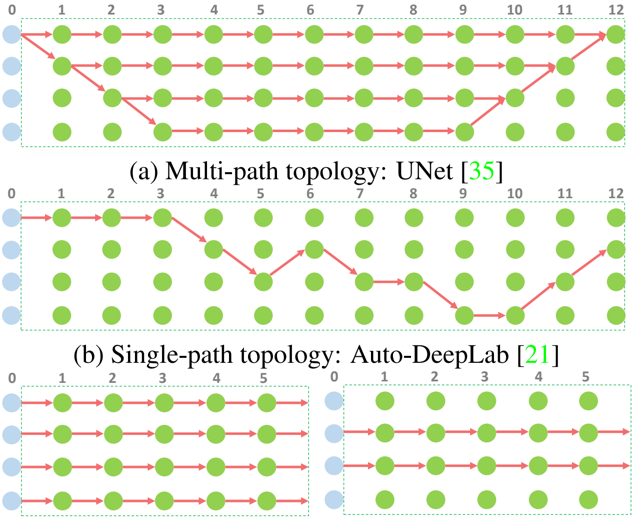
Figure 2. Our search space contains L=12 layers. The blue edges are the stem containing pre-defined operations. The cell operations are defined on the edges while the nodes are feature maps. Edges in the topology search space that are selected for features to flow from input to output form a candidate network topology. Each edge in the search space includes a cell which contains O=5 operations to select from. A downsample/upsample edge also contains a 2× downsample/upsample operation.

use temperature annealing to achieve the same goal. Another problem of the differentiable method is the large memory usage during search stage. PC-DARTS [43] uses partial channel connections to reduce memory, while Auto-DeepLab [21] reduces the filter number at search stage. It's a common practice to retrain the searched model while increasing the filter number, batch size, or patch size to gain better performance. But for 3D medical image segmentation, the change of retraining scheme (e.g. transferring to a new task which requires larger input size) can still cause out-of-memory problem. Most NAS work has been focused on searching architecture with latency constraints [1, 3, 18, 36], while only a few considered memory as a constraint. Mem-NAS [24] uses a growing and trimming framework to constrain the inference GPU memory but does not allow integration in a differentiable scheme.

3. Method

3.1. Network Topology Search Space

Inspired by Auto-Deeplab [21] and [19], we propose a search space with fully connected edges between adjacent resolutions (2× higher, 2× lower or the same) from adjacent layers as shown in Fig. 2. A stack of multi-resolution images are generated by down-sampling the input image by 1/2, 1/4, 1/8 along each axis. Together with the original image, we use four $3 \times 3 \times 3$ 3D convolutions with stride 2 to generate multi-resolution features (layer 0 in Fig. 2) to the following search space. The search space has L layers and each layer consists of feature nodes (green nodes) from D=4 resolutions and E=3D-2 candidate input edges (dashed green edges). Each edge contains a cell operation, and a upsample/downsample operation (factor 2) is used before the cell if the edge is an upsample/downsample edge. A feature node is the summation of the output features from each input edge. Compared to Auto-DeepLab [21], our search space supports searching for input image scales



(a) Multi-path topology: UNet [35]

(b) Single-path topology: Auto-DeepLab [21]

(c) Multi-resolution input [20] and Input selection

Figure 3. Our search space covers a variety of topologies (single-path, multi-path) and can select input resolutions.

and complex multi-path topologies, as shown in Fig. 3. As for multi-path topology, MS-NAS [44] discretizes and combines multiple single-path models searched from Auto-DeepLab's framework, but the search is still unaware of the discretization thus causing the gap. [19] also supports multi-path topology, but [19] is more about feature routing in a "fully connected" network, not a NAS method.

3.2. Cell Level Search Space

We define a cell search space to be a set of basic operations where the input and output feature maps have the same spatial resolution. The cell search space in DARTS [23] and Auto-Deeplab [21] contains multiple blocks and the connections among those blocks can also be searched. However, the searched cells are repeated over all the cells in the network topology level. Similar to C2FNAS [45], our algorithm searches the operation of each cell independently, with one operation selected from the following:

(1) skip connection (2) 3x3x3 3D convolution
(3) P3D 3x3x1: 3x3x1 followed by 1x1x3 convolution
(4) P3D 3x1x3: 3x1x3 followed by 1x3x1 convolution
(5) P3D 1x3x3: 1x3x3 followed by 3x1x1 convolution

P3D represents pseudo 3D [32] and has been used in V-NAS [48]. A cell also includes ReLU activation and Instance Normalization [39] which are used before and after those operations respectively (except for skip connection). The cell operations do not include multi-scale feature aggregation operations like atrous convolution and pooling. The feature spatial changes are performed by the upsample/downsample operations in the edges searched from the topology level.

3.3. Continuous Relaxation and Discretization

3.3.1 Preliminaries

We briefly recap the relaxation in DARTS [23]. NAS tries to select one from N candidate operations $O_1, O_2, \cdots, O_N$ for each computational node. Each operation O_i is paired with a trainable parameter α_i where $\sum_{i=1}^{N} \alpha_i = 1$, $\alpha_i \geq 0$, and the output feature $x_{out} = \sum_{i=1}^{N} \alpha_i O_i(x_{in})$, where x_{in} is the input feature. Thus, the discrete operation is relaxed by the continuous representation α which can be optimized using gradient descent. After optimization, O_i with larger α_i is more important and will be selected. However, a small α_j (as long as $\alpha_j \neq 0$) can still make a significant difference on x_{out} and following layers. Therefore, directly discarding non-zero operations will lead to the discretization gap.

Auto-DeepLab [21] extends this idea to edge selection in network topology level. As illustrated in Fig. 1, every edge is paired with a trainable parameter β ($0 \geq \beta \geq 1$), and parameters paired with edges that pointed to the same feature node sum to one. This is based on an assumption that "one input edge for each node" because the input edges to a node are competing with each other. After discretization, a single path is kept while other edges, even with a large β, are discarded. This means the feature flow in the searched continuous model has a significant gap with the feature flow in the final discrete model. The single-path topology limitation comes from the previous assumption for topology level relaxation while the gap comes from the unawareness of the discretization in the search stage, such that edges with large probabilities can be discarded due to topology.

3.3.2 Sequential Model with Super Feature Node

We propose a network topology relaxation framework which converts the multi-scale search space into a sequential space using "Super Feature Node". For a search space with L layers and D resolution levels, these D feature nodes in the same layer i are combined as a super feature node s_i and features flow sequentially from these L super nodes as shown in Fig. 4. There are E=3D-2 candidate input edges to each super node and the topology search is to select an optimal set of input edges for each super node. We define a connection pattern as a set of selected edges and there are $M = 2^E - 1$ feasible candidate connection patterns. The j-th connection pattern cp_j is an indication vector of length E, where $cp_j(e) = 1$, if e-th edge is selected in j-th pattern.

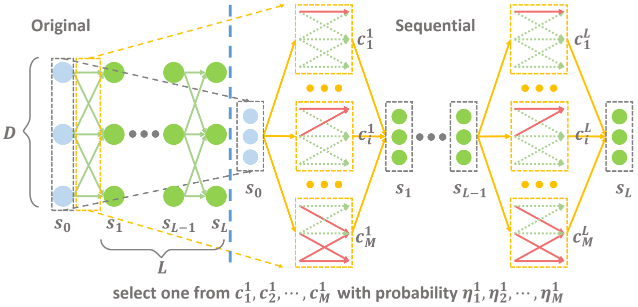

Original
Sequential
D
s_0
s_1
s_{L-1}
s_L
L
c_1^1
c_i^1
c_M^1
c_1^L
c_i^L
c_M^L
select one from $c_1^1, c_2^1, \cdots, c_M^1$ with probability $\eta_1^1, \eta_2^1, \cdots, \eta_M^1$

Figure 4. The feature nodes at the same layer i are combined as a super node s_i. A set of selected edges (e.g. red edges in a dashed yellow block) that connects s_{i-1} and s_i is a "connection". For E edges, there are $M = 2^E - 1$ connection patterns. Topology search becomes selecting one connection pattern to connect adjacent super nodes sequentially.

We define the input connection operation to s_i with connection pattern cp_j as c_j^i. cp_j defines c_j^i's topology while c_j^i also includes cell operations on the selected edges in cp_j. c_j^i, c_k^{i+1} means the input/output connection patterns for s_i are cp_j, cp_k respectively. Under this formulation, the topology search becomes selecting an input connection pattern for each super node and the competition is among all M connection patterns, not among edges. We associate a variable η_j^i to the connection operation c_j^i for every s_i and every pattern j. Denote the input features at layer 0 as s_0, we have a sequential feature flow equation:

$$s_i = \sum_{j=1}^{M} (\eta_j^i * c_j^i(s_{i-1})) \quad i = 1 \cdots, L \tag{1}$$
$$\sum_{j=1}^{M} \eta_j^i = 1, \eta_j \geq 0 \quad \forall i, j$$

$$\eta_j^i = \frac{\prod_{e=1}^{E} (1 - p_e^i)^{1-cp_j(e)} (p_e^i)^{cp_j(e)}}{\sum_{j=1}^{M} \prod_{e=1}^{E} (1 - p_e^i)^{1-cp_j(e)} (p_e^i)^{cp_j(e)}} \tag{2}$$
$$0 \leq p_e^i \leq 1 \quad \forall i, e$$

However, M is growing exponentially with D. To reduce the architecture parameters, we parameterize η_j^i with a set of edge probability parameters p_e^i, e=1, $\cdots$, E in Eq. 2.

For a search space with L=12 layers and D=4, the network topology parameter number is reduced from $M \times L = 1023 \times 12$ to $E \times L = 10 \times 12$. Under this formulation, the probability η of connections are highly correlated. If an input edge e to s_i has low probability, all the candidate patterns to s_i with e selected will have lower probabilities.

For cell operation relaxation, we use the method in Sec. 3.3.1. Each cell on the input edge e to s_i has its own cell architecture parameters $\alpha_1^{i,e}, \alpha_2^{i,e}, \cdots, \alpha_N^{i,e}$ and will be optimized. Notice the c_j^i in Eq. 1. contains the cell operations defined on the selected edges, and it contains relaxed cell architecture parameters α. Thus we can perform gradient based search for topology and cell levels jointly.

3.3.3 Discretization with Topology Constraints

After training, the final discrete architecture is derived from the optimized continuous architecture representation η (derived from p_e^i) and α. η_i^j represents the probability of using input connection pattern cp_j^i for super node s_i. Since the network topology search space is converted into a sequential space, the easiest way for topology discretization is to select cp_j with the maximum η_j^i. However, the topology may not be feasible. We define topology infeasibility as:

> *"a feature node has an input edge but no output edge or has an output edge but no input edge".*

The gray feature nodes in Fig. 5 indicate infeasible topology. Therefore, we cannot select cp_j and cp_k as s_i's input/output connection patterns even if they have the largest probabilities. For every connection pattern cp_j, we generate a feasible set $\mathcal{F}(j)$. If a super node with input pattern j and output pattern k is feasible (all feature nodes of the super node are topologically feasible), then $k \in \mathcal{F}(j)$. Denote the array of selected input connection pattern indexes for these L super nodes as I, and the topology discretization can be performed by sampling I from its distribution $p(I)$ using maximum likelihood (minimize negative log likelihood):

$$p(I) = \begin{cases} \prod_{i=1}^{L} \eta_i^{I(i)}, & \forall i: \quad I(i+1) \in \mathcal{F}(I(i)) \\ 0, & \text{else.} \end{cases} \tag{3}$$

$$I = \underset{I}{\text{argmin}} \sum_{i=1}^{L} \text{-log}(\eta_i^{I(i)}), \forall i: I(i+1) \in \mathcal{F}(I(i)) \tag{4}$$

We build a directed graph $\mathcal{G}$ using η and $\mathcal{F}$ as illustrated in Fig. 5. The nodes (yellow blocks) of $\mathcal{G}$ are connection operations and the input edge cost to a node c_j^i in $\mathcal{G}$ is $-\log(\eta_j^i)$. The path with minimum cost from the source to the sink nodes (green nodes with gray contour) corresponds to Eq. 4, and we obtained the optimal I using Dijkstra algorithm [7]. For cell operations on the selected edges from I, we simply use the operation with the largest α.

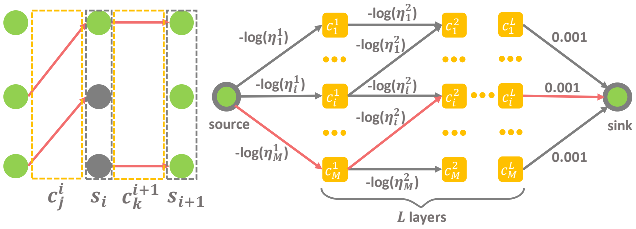


Figure 5. *Left:* The gray feature nodes are topologically infeasible, thus connection pattern index k is not in j's feasible set, $k \notin \mathcal{F}(j)$. *Right:* A directed graph $\mathcal{G}$ which contains $L \times M$+2 nodes. A node c_j^i (yellow block) is connected with c_k^{i+1} and c_m^{i-1} if $j \in \mathcal{F}(m)$ and $k \in \mathcal{F}(j)$. The cost of edges directed to c_j^i is -logη_j^i. The source connects to all first layer nodes and all L-th layer nodes connect to the sink (edge cost is a constant value). Those L nodes on the shortest path from source to sink (red path) in $\mathcal{G}$ represent the optimal feasible connection operations (final architecture).

3.4. Bridging the Discretization Gap

To minimize the gap between the continuous representation and the final discretized architecture, we add entropy losses to encourage binarization of α and η:

$$\begin{aligned} \mathcal{L}_\alpha &= \frac{-1}{L*E*N} \sum_{i=1}^{L} \sum_{e=1}^{E} \sum_{n=1}^{N} \alpha_n^{i,e} * log(\alpha_n^{i,e}) \\ \mathcal{L}_\eta &= \frac{-1}{L*M} \sum_{i=1}^{L} \sum_{j=1}^{M} \eta_j^i * log(\eta_j^i) \end{aligned} \tag{5}$$

However, even if the architecture parameters α and η are almost binarized, there may still be a large gap due to the topology constraints in the discretization algorithm. Recall the definition of topology feasibility in Sec. 3.3.3: an activated feature node (node with at least one input edge) must have an output edge while an in-activated feature node cannot have an output edge. Each super node has D feature nodes, thus there are $2^D - 1$ node activation pattern. We define $\mathcal{A}$ as the set of all node activation patterns. Each element $a \in \mathcal{A}$ is a indication function of length D, where $a(i) = 1$ if the i-th node of the super-node is activated. We further define two sets $\mathcal{F}_{in}(a)$ and $\mathcal{F}_{out}(a)$ representing all feasible input and output connection pattern indexes for a super node with node activation a as shown in Fig. 6. We propose the following topology loss:

$$p_{in}^i(a) = \sum_{j \in F_{in}(a)} \eta_j^i, \quad p_{out}^i(a) = \sum_{j \in F_{out}(a)} \eta_j^{i+1} \tag{6}$$

$$\begin{aligned} \mathcal{L}_{tp} = -\sum_{i=1}^{L-1} \sum_{a \in \mathcal{A}} (\quad & p_{in}^i(a) \log(p_{out}^i(a)) \quad + \\ & (1 - p_{in}^i(a)) \log(1 - p_{out}^i(a)) \quad) \end{aligned} \tag{7}$$

$p_{in}^i(a)$ is the probability that the activation pattern for s_i is a, and $p_{out}^i(a)$ is the probability that s_i with pattern a

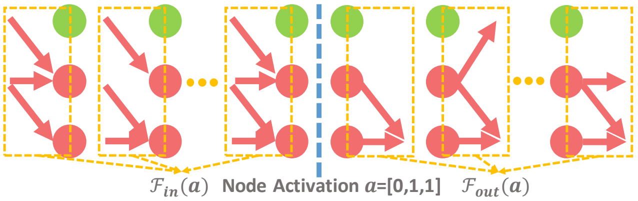


Figure 6. The connection patterns in $\mathcal{F}_{in}(a)$ activates pattern a, and all feasible output connection patterns are in $\mathcal{F}_{out}(a)$. $a = [0, 1, 1]$ means the last two nodes of the super-node are activated.

is feasible. By minimizing $\mathcal{L}_{tp}$, the search stage is aware of the topology constraints and encourages all super nodes to be topologically feasible, thus reduce the gap caused by topology constraints in the discretization step.

3.5. Memory Budget Constraints

The searched model is usually retrained under different training settings like patch size, filter number, or tasks. Auto-DeepLab [21] used 4× larger image patch and 6× more filters in retraining compared to the search stage. But this can cause out of memory problem for 3D images in retraining, thus we consider memory budget in architecture search. A cell's expected memory usage is estimated by $M^{i,e} = \sum_{n=1}^{N} \alpha_n^{i,e} M_n$. M_n is the memory usage of operation O_n (estimated by tensor size [10]) defined in Sec. 3.2. The expected memory usage M_e of the searched model is:

$$M_e = \sum_{i=1}^{L}\sum_{j=1}^{M} \eta_j^i * (\sum_{e=1}^{E} M^{i,e} * cp_j(e)) \tag{8}$$

Similar to [19], we consider the budget as the percentage σ of the maximum memory usage M_a, of which all α and η equal to one.

$$M_a = \sum_{i=1}^{L}\sum_{j=1}^{M} *(\sum_{e=1}^{E}(\sum_{n=1}^{N} M_n) * cp_j(e)) \tag{9}$$

$$\mathcal{L}_m = |M_e/M_a - \sigma|_1 \tag{10}$$

3.6. Optimization

We adopt the same optimization strategy as in DARTS [23] and Auto-DeepLab [21]. We partition the training set into *train1* and *train2*, and optimize the network weights w (e.g. convolution kernels) using $\mathcal{L}_{seg}$ on *train1* and network architecture weights α and p_e using $\mathcal{L}_{arch}$ on *train2* alternately. The loss $\mathcal{L}_{seg}$ for w is the evenly sum of dice and cross-entropy loss [45] in segmentation, while

$$\mathcal{L}_{arch} = \mathcal{L}_{seg} + t/t_{all} * (\mathcal{L}_\alpha + \mathcal{L}_\eta + \lambda * \mathcal{L}_{tp} + \mathcal{L}_m) \tag{11}$$

t and t_{all} are the current and total iterations for architecture optimization such that the searching is focusing more on $\mathcal{L}_{seg}$ at the starting point. We empirically scale $\mathcal{L}_{tp}$ to the same range with other losses by setting λ=0.001.

4. Experiments

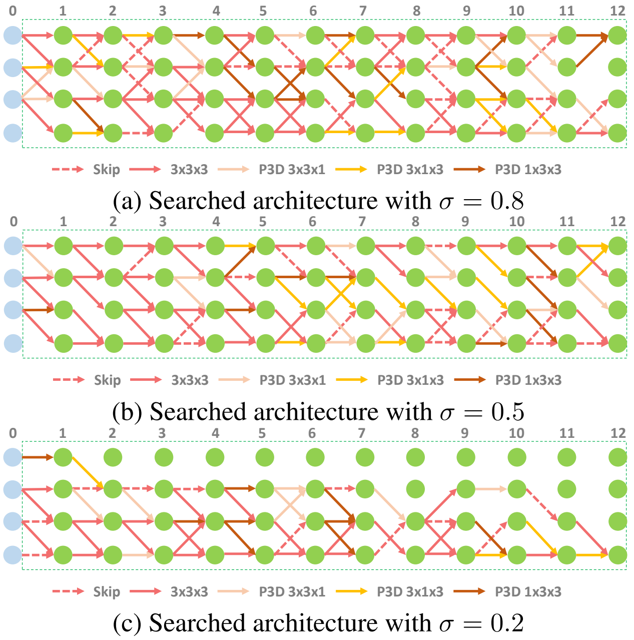
We conduct experiments on the MSD dataset [37] which is a comprehensive benchmark for medical image segmentation. It contains ten segmentation tasks covering different anatomies of interest, modalities and imaging sources (institutions) and is representative for real clinical problems. Recent C2FNAS [45] reaches state-of-the-art results on MSD dataset using NAS based methods. We follow its experiment settings by searching on the MSD Pancreas dataset and deploying the searched model on all 10 MSD tasks for better comparison. All images are resampled to have a $1.0 \times 1.0 \times 1.0\ mm^3$ voxel resolution.

4.1. Implementation Details

Our search space has L=12 layers and D=4 resolution levels as shown in Fig. 2. The stem cell at scale 1 has 16 filters and we double the filter number when decreasing the spatial size by half in each axis. The search is conducted on Pancreas dataset following the same 5 fold data split (4 for training and last 1 for validation) as C2FNAS [45]. We use SGD optimizer with momentum 0.9, weight decay of 4e-5 for network weights w. We train w for the first one thousand (1k) warm-up and following 10k iterations without updating architecture. The architecture weights α, p_e are initialized with Gaussian $\mathcal{N}(1, 0.01), \mathcal{N}(0, 0.01)$ respectively before softmax and sigmoid. In the following 10k iterations, we jointly optimize w with SGD and α, p_e with Adam optimizer [16] (learning rate 0.008, weight decay 0). The learning rate of SGD linearly increases from 0.025 to 0.2 in the first 1k warm-up iterations, and decays with factor 0.5 at the following [8k, 16k] iterations. The search is conducted on 8 GPUs with batch size 8 (each GPU with one 96×96×96 patch). The patches are randomly augmented with 2D rotation by [90, 180, 270] degrees in the x-y plane and flip in all three axis. The total training iterations, SGD learning rate scheduler and data pre-processing and augmentation are the same with C2FNAS [45]. After searching, the discretized model is randomly initialized and retrained with doubled filter number and doubled batch size to match C2FNAS [45]'s setting. We use the SGD optimizer with 1k warm-up and 40k training iterations and decay the learning rate by a factor of 0.5 at [8k, 16k, 24k, 32k] iterations after warm-up. The learning rate scheduler is the same with search stage in the warm-up and the first 20k iterations. The latter 20k iterations are for better convergence and match the 40k total retraining iterations used in C2FNAS [45]. The same data augmentation as C2FNAS (also the same as the search stage) is used for the Pancreas dataset for better comparison. To test the generalizability of the searched model, we retrain the model on all of the rest nine tasks. Some tasks in the MSD dataset contain very few training data so we use additional basic 2D data augmentations of random rotation, scaling and gamma correction for all nine tasks. We use

Table 1. Comparison of FLOPs, Parameters and Retraining GPU memory usage and the 5-Fold cross validation Dice-Sørensen score of our searched architectures on Pancreas dataset

Model	FLOPs (G)	Params. (M)	Memory (MB)	DSC1	DSC2	Avg.
3D UNet [6] (nn-UNet)	658	18	9176	-	-	-
Attention UNet [28]	1163	104	13465	-	-	-
C2FNAS [45]	151	17	5730	-	-	-
DiNTS (σ=0.2)	146	163	5787	77.94	48.07	63.00
DiNTS (σ=0.5)	308	147	10110	**80.20**	52.25	66.23
DiNTS (σ=0.8)	334	152	13018	80.06	**52.53**	**66.29**



(a) Searched architecture with $\sigma = 0.8$

(b) Searched architecture with $\sigma = 0.5$

(c) Searched architecture with $\sigma = 0.2$

Figure 7. Searched architectures (not including the stem in Fig. 2) on Pancreas dataset with varying memory constraints.

patch size $96 \times 96 \times 96$ and stride $16 \times 16 \times 16$ for all ten tasks except Prostate and Hippocampus. Prostate data has very few slices (less than 40) in the z-axis, so we use patch size $96 \times 96 \times 32$ and stride $16 \times 16 \times 4$. Hippocampus data size is too small (around $36 \times 50 \times 35$) and we use patch size $32 \times 32 \times 32$ and stride $4 \times 4 \times 4$. Post-processing with largest connected component is also applied.

4.2. Pancreas Dataset Search Results

The search takes 5.8 GPU days while C2FNAS takes 333 GPU days on the same dataset (both using 8 16GB V100 GPU). We vary the memory constraints $\sigma = [0.2, 0.5, 0.8]$ and show the search results in Fig. 7. The searched models have highly flexible topology which are searched jointly with the cell level. The 5-fold cross-validation results on Pancreas are shown in Table 1. By increasing σ, the searched model is more "dense in connection" and can achieve better performance while requiring more GPU memory (estimated using PyTorch [29] functions in training described in Sec. 4.1). The marginal performance drop by decreasing $\sigma = 0.8$ to $\sigma = 0.5$ shows that we can reduce memory usage without losing too much accuracy. Although techniques like mixed-precision training [25] can be used to further reduce memory usage, our memory aware search tries to solve this problem from NAS perspective. Compared to nnUNet [14] (represented by 3D UNet because it ensembles 2D/3D/cascaded-3D U-Net differently for each task) and C2FNAS in Table 1, our searched models have no advantage in FLOPs and Parameters which are important in mobile settings. We argue that for medical image analysis, light model and low latency are less a focus than better GPU memory usage and accuracy. Our DiNTS can optimize the usage of the available GPU and achieve better performance.

4.3. Segmentation Results on MSD

The searched model with $\sigma = 0.8$ from Pancreas is used for retraining and testing on all ten tasks of MSD dataset. Similar to the model ensemble used in nnUNet [14] and C2FNAS [45], we use a 5 fold cross validation for each task and ensemble the results using majority voting. The largest connected component post-processing in nnUNet [14] is also applied. The Dice-Sørensen (DSC) and Normalised Surface Distance (NSD) as used in the MSD challenge are reported for the test set in Table 2. nnUNet [14] uses extensive data augmentation, different hyper-parameters like patch size, batch size for each task and ensembles networks with different architectures. It focuses on hyper-parameter selection based on hand-crafted rules and is the champion of multiple medical segmentation challenges including MSD. Our method and C2FNAS [45] focus on architecture search and use consistent hyper-parameters and basic augmentations for all ten tasks. We achieved better results than C2FNAS [45] in all tasks with similar hyper-parameters while only takes 1.7% searching time. Comparing to nn-UNet [14], we achieve much better performance on challenging datasets like Pancrease, Brain and Colon, while worse on smaller datasets like Heart (10 test cases), Prostate (16 test cases) and Spleen (20 test cases). Task-specific hyper-parameters, test-time augmentation, extensive data augmentation and ensemble more models as used in nn-UNet [14] might be more effective on those small datasets than our unified DiNTS searched architecture. Overall, we achieved the best average results and top ranking in the MSD challenge leaderboard, showing that a non-UNet based topology can achieve superior performance in medical imaging.

4.4. Ablation Study

4.4.1 Search on Different Datasets

The models in Sec. 4.2 and Sec. 4.3 are searched from the Pancreas dataset (282 CT 3D training images). To test the generalizability of DiNTS, we perform the same search as in Sec. 4.1 on Brain (484 MRI data), Liver (131 CT data)

	Brain							
Metric	DSC1	DSC2	DSC3	Avg.	NSD1	NSD2	NSD3	Avg.
CerebriuDIKU [30]	**69.52**	43.11	66.74	59.79	88.25	68.98	88.90	82.04
NVDLMED [41]	67.52	45.00	68.01	60.18	86.99	69.77	89.82	82.19
Kim et al [15]	67.40	45.75	68.26	60.47	86.65	72.03	90.28	82.99
nnUNet [14]	68.04	46.81	68.46	61.10	87.51	72.47	90.78	83.59
C2FNAS [45]	67.62	48.60	69.72	61.98	87.61	72.87	91.16	83.88
DiNTS	69.28	**48.65**	**69.75**	**62.56**	**89.33**	**73.16**	**91.69**	**84.73**

	Heart		Liver					
Metric	DSC1	NSD1	DSC1	DSC2	Avg.	NSD1	NSD2	Avg.
CerebriuDIKU [30]	89.47	90.63	94.27	57.25	75.76	96.68	72.60	84.64
NVDLMED [41]	92.46	95.57	95.06	71.40	83.23	98.26	87.16	92.71
Kim et al [15]	93.11	96.44	94.25	72.96	83.605	96.76	88.58	92.67
nnUNet [14]	**93.30**	**96.74**	**95.75**	**75.97**	**85.86**	98.55	90.65	94.60
C2FNAS [45]	92.49	95.81	94.98	72.89	83.94	98.38	89.15	93.77
DiNTS	92.99	96.35	95.35	74.62	84.99	**98.69**	**91.02**	**94.86**

	Lung		Hippocampus					
Metric	DSC1	NSD1	DSC1	DSC2	Avg.	NSD1	NSD2	Avg.
CerebriuDIKU [30]	58.71	56.10	89.68	88.31	89.00	97.42	97.42	97.42
NVDLMED [41]	52.15	50.23	87.97	86.71	87.34	96.07	96.59	96.33
Kim et al [15]	63.10	62.51	90.11	88.72	89.42	97.77	**97.73**	**97.75**
nnUNet [14]	73.97	76.02	**90.23**	**88.69**	**89.46**	**97.79**	97.53	97.66
C2FNAS [45]	70.44	72.22	89.37	87.96	88.67	97.27	97.35	97.31
DiNTS	**74.75**	**77.02**	89.91	88.41	89.16	97.76	97.56	97.66

	Spleen		Prostate					
Metric	DSC1	NSD1	DSC1	DSC2	Avg.	NSD1	NSD2	Avg.
CerebriuDIKU [30]	95.00	98.00	69.11	86.34	77.73	94.72	97.90	96.31
NVDLMED [41]	96.01	99.72	69.36	86.66	78.01	92.96	97.45	95.21
Kim et al [15]	91.92	94.83	72.64	89.02	80.83	95.05	98.03	96.54
nnUNet [14]	**97.43**	**99.89**	**76.59**	**89.62**	**83.11**	**96.27**	**98.85**	**97.56**
C2FNAS [45]	96.28	97.66	74.88	88.75	81.82	98.79	95.12	96.96
DiNTS	96.98	99.83	75.37	89.25	82.31	95.96	98.82	97.39

	Colon		Hepatic Vessels					
Metric	DSC1	NSD1	DSC1	DSC2	Avg.	NSD1	NSD2	Avg.
CerebriuDIKU [30]	28.00	43.00	59.00	38.00	48.50	79.00	44.00	61.50
NVDLMED [41]	55.63	66.47	61.74	61.37	61.56	81.61	68.82	75.22
Kim et al [15]	49.32	62.21	62.34	68.63	65.485	83.22	78.43	80.825
nnUNet [14]	58.33	68.43	**66.46**	**71.78**	**69.12**	**84.43**	80.72	**82.58**
C2FNAS [45]	58.90	**72.56**	64.30	71.00	67.65	83.78	80.66	82.22
DiNTS	**59.21**	70.34	64.50	71.76	68.13	83.98	**81.03**	82.51

	Pancreas						**Overall**	
Metric	DSC1	DSC2	Avg.	NSD1	NSD2	Avg.	DSC	NSD
CerebriuDIKU [30]	71.23	24.98	48.11	91.57	46.43	69.00	67.01	77.86
NVDLMED [41]	77.97	44.49	61.23	94.43	63.45	78.94	72.78	83.26
Kim et al [15]	80.61	51.75	66.18	95.83	73.09	84.46	74.34	85.12
nnUNet [14]	**81.64**	52.78	67.21	96.14	71.47	83.81	77.89	88.09
C2FNAS [45]	80.76	54.41	67.59	96.16	75.58	85.87	76.97	87.83
DiNTS	81.02	**55.35**	**68.19**	**96.26**	**75.90**	**86.08**	**77.93**	**88.68**

Table 2. Dice-Sørensen score (DSC) and Normalised Surface Distance (NSD) results on the MSD test dataset (numbers from MSD challenge live leaderboard).

Test Dataset	Brain		Liver		Lung	
Search Dataset	Brain	Pancreas	Liver	Pancreas	Lung	Pancreas
DSC1	**80.20**	79.68	**94.15**	94.12	**69.30**	68.90
DSC2	**61.09**	60.67	**58.74**	57.86	-	-
DSC3	**77.63**	77.48	-	-	-	-
Avg.	**72.97**	72.61	**76.44**	75.99	**69.30**	68.90

Table 3. Dice-Sørensen score (DSC) of 5-fold cross validation on Brain, Liver and Lung datasets of architectures searched from Pancreas, Brain, Liver and Lung datasets with $\sigma = 0.8$.

and Lung (64 CT data) covering big, medium and small datasets. The results are shown in Table. 3 and demonstrate the good generalizability of our DiNTS.

4.4.2 Necessity of Topology Loss

As illustrated in Sec. 1, the discretization algorithm discards topologically infeasible edges (even with large probabilities), which causes a gap between feature flow in the optimized continuous model (Eq. 1) and the discrete model. Our topology loss encourages connections with large probabilities to be feasible, thus will not be discarded and causing the gap. We denote C_{max} as the topology decoded by selecting connection j with largest η_j^i for each layer i (can be infeasible). C_{top} is the topology decoded by our discretization algorithm. C_{max}, C_{top} are the indication matrices of size $[L, E]$ representing whether an edge is selected, and $G = \sum_{i=1}^{L} \sum_{e=1}^{E} |C_{max}(i,e) - C_{top}(i,e)|$. Larger G represents larger gap between the feature flow before and after discretization. Fig. 8 shows the change of G during search with/without topology loss under different memory constraints. With topology loss, the gap between C_{max} and C_{top} is reduced, and it's more crucial for smaller σ where the searched architecture is more sparse and more likely to have topology infeasibility.

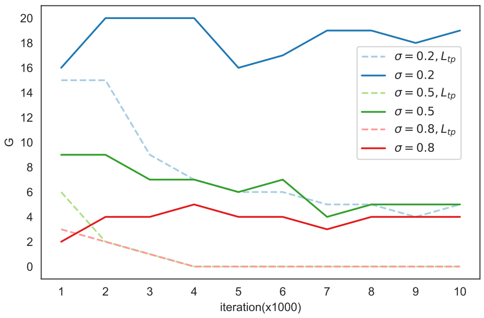


Figure 8. The indication G of discretization gap during architecture search with different memory constraints σ. With topology loss (dashed line), G is decreased compared to no topology loss (solid line), showing the importance of topology loss.

5. Conclusions

In this paper, we present a novel differentiable network topology search framework (DiNTS) for 3D medical image segmentation. By converting the feature nodes with varying spatial resolution into super nodes, we are able to focus on connection patterns rather than individual edges, which enables more flexible network topologies and a discretization aware search framework. Medical image segmentation challenges have been dominated by U-Net based architectures [14], even NAS-based C2FNAS is searched within a U-shaped space. DiNTS's topology search space is highly flexible and achieves the best performance on the benchmark MSD challenge using non-UNet architectures, while only taking 1.7% search time compared to C2FNAS. Since directly converting Auto-DeepLab [21] to the 3D version will have memory issues, we cannot fairly compare with it. For future work, we will test our proposed algorithm on 2D natural image segmentation benchmarks and explore more complex cells.

References

[1] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *ICLR*, 2019.

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, pages 801–818, 2018.

[3] Wuyang Chen, Xinyu Gong, Xianming Liu, Qian Zhang, Yuan Li, and Zhangyang Wang. Fasterseg: Searching for faster real-time semantic segmentation. *ICLR*, 2020.

[4] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1294–1303, 2019.

[5] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair darts: Eliminating unfair advantages in differentiable architecture search. *arXiv preprint arXiv:1911.12126*, 2019.

[6] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *MICCAI*, pages 424–432. Springer, 2016.

[7] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[8] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *JMLR*, 2019.

[9] Jiemin Fang, Yuzhu Sun, Qian Zhang, Yuan Li, Wenyu Liu, and Xinggang Wang. Densely connected search space for more flexible neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10628–10637, 2020.

[10] Yanjie Gao, Yu Liu, Hongyu Zhang, Zhengxian Li, Yonghao Zhu, Haoxiang Lin, and Mao Yang. Estimating gpu memory consumption of deep learning models. Technical report, Microsoft, May 2020.

[11] Yu-Chao Gu, Yun Liu, Yi Yang, Yu-Huan Wu, Shao-Ping Lu, and Ming-Ming Cheng. Dots: Decoupling operation and topology in differentiable architecture search. *arXiv preprint arXiv:2010.00969*, 2020.

[12] Shoukang Hu, Sirui Xie, Hehui Zheng, Chunxiao Liu, Jianping Shi, Xunying Liu, and Dahua Lin. Dsnas: Direct neural architecture search without parameter retraining. In *CVPR*, pages 12084–12092, 2020.

[13] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, pages 4700–4708, 2017.

[14] Fabian Isensee, Paul F Jäger, Simon AA Kohl, Jens Petersen, and Klaus H Maier-Hein. Automated design of deep learning methods for biomedical image segmentation. *arXiv preprint arXiv:1904.08128*, 2019.

[15] 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. In *MICCAI*, pages 220–228. Springer, 2019.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[17] Xiaomeng Li, Hao Chen, Xiaojuan Qi, Qi Dou, Chi-Wing Fu, and Pheng-Ann Heng. H-denseunet: hybrid densely connected unet for liver and tumor segmentation from ct volumes. *TMI*, 37(12):2663–2674, 2018.

[18] Xin Li, Yiming Zhou, Zheng Pan, and Jiashi Feng. Partial order pruning: for best speed/accuracy trade-off in neural architecture search. In *CVPR*, pages 9145–9153, 2019.

[19] Yanwei Li, Lin Song, Yukang Chen, Zeming Li, Xiangyu Zhang, Xingang Wang, and Jian Sun. Learning dynamic routing for semantic segmentation. In *CVPR*, pages 8553–8562, 2020.

[20] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, July 2017.

[21] 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.

[22] 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*, pages 19–34, 2018.

[23] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *ICLR*, 2019.

[24] Peiye Liu, Bo Wu, Huadong Ma, and Mingoo Seok. Memnas: Memory-efficient neural architecture search with grow-trim learning. In *CVPR*, pages 2108–2116, 2020.

[25] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *ICLR*, 2018.

[26] 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.

[27] Niv Nayman, Asaf Noy, Tal Ridnik, Itamar Friedman, Rong Jin, and Lihi Zelnik. Xnas: Neural architecture search with expert advice. In *NeurIPS*, pages 1977–1987, 2019.

[28] 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.

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8026–8037, 2019.

[30] 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.

[31] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *ICML*, 2018.

[32] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, pages 5533–5541, 2017.

[33] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

[34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015.

[35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015.

[36] Albert Shaw, Daniel Hunter, Forrest Landola, and Sammy Sidhu. Squeezenas: Fast neural architecture search for faster semantic segmentation. In *ICCV Workshops*, Oct 2019.

[37] 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.

[38] Yunjie Tian, Chang Liu, Lingxi Xie, Jianbin Jiao, and Qixiang Ye. Discretization-aware architecture search. *arXiv preprint arXiv:2007.03154*, 2020.

[39] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv:1607.08022*, 2016.

[40] Yu Weng, Tianbao Zhou, Yujie Li, and Xiaoyu Qiu. Nas-unet: Neural architecture search for medical image segmentation. *IEEE Access*, 7:44247–44257, 2019.

[41] 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. In *WACV*, pages 3646–3655, 2020.

[42] Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. Exploring randomly wired neural networks for image recognition. In *ICCV*, pages 1284–1293, 2019.

[43] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient differentiable architecture search. *ICLR*, 2020.

[44] Xingang Yan, Weiwen Jiang, Yiyu Shi, and Cheng Zhuo. Ms-nas: Multi-scale neural architecture search for medical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 388–397. Springer, 2020.

[45] Qihang Yu, Dong Yang, Holger Roth, Yutong Bai, Yixiao Zhang, Alan L Yuille, and Daguang Xu. C2FNAS: Coarse-to-Fine Neural Architecture Search for 3D Medical Image Segmentation. In *CVPR*, pages 4126–4135, 2020.

[46] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, pages 2881–2890, 2017.

[47] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: Redesigning skip connections to exploit multiscale features in image segmentation. *TMI*, 39(6):1856–1867, 2019.

[48] Zhuotun Zhu, Chenxi Liu, Dong Yang, Alan Yuille, and Daguang Xu. V-nas: Neural architecture search for volumetric medical image segmentation. In *3DV*, pages 240–248. IEEE, 2019.

[49] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *ICLR*, 2017.

[50] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, pages 8697–8710, 2018.