T-AutoML: Automated Machine Learning for Lesion Segmentation using Transformers in 3D Medical Imaging

Dong Yang Andriy Myronenko Xiaosong Wang Ziyue Xu Holger R. Roth Daguang Xu NVIDIA

Abstract

Lesion segmentation in medical imaging has been an important topic in clinical research. Researchers have proposed various detection and segmentation algorithms to address this task. Recently, deep learning-based approaches have significantly improved the performance over conventional methods. However, most state-of-the-art deep learning methods require the manual design of multiple network components and training strategies. In this paper, we propose a new automated machine learning algorithm, T-AutoML, which not only searches for the best neural architecture, but also finds the best combination of hyper-parameters and data augmentation strategies simultaneously. The proposed method utilizes the modern transformer model, which is introduced to adapt to the dynamic length of the search space embedding and can significantly improve the ability of the search. We validate T-AutoML on several large-scale public lesion segmentation data-sets and achieve state-of-the-art performance.

1. Introduction

Nowadays, given the technological advances in algorithmic design (such as deep learning) and hardware platforms (such as GPU), medical image analysis has become a key step in disease understanding, clinical diagnosis, and treatment planning. The sizes, shapes, and appearances of lesions in medical images can vary greatly across different anatomical structures (shown in Fig. 1). These semantic features are closely related to the severity of the disease. At the same time, variations in the imaging patterns related to pathologies, scanning protocols, and medical devices introduce a computational challenge for automated algorithms [1]. Therefore, automated lesion segmentation is one of the most challenging tasks in automated analysis of medical images, such as 3D CT, 3D MRI, histopathology, etc. When facing with such challenges, most machine/deep learning models struggle to provide comprehensive solutions.

In recent years, deep convolutional neural networks have been widely applied for medical image analysis. For instance, one of the most influential works in medical image segmentation using deep neural networks is U-Net [2, 3], which has been found to be more effective and efficient than previous non-learning based methods. Furthermore, automated machine learning (AutoML) with neural networks has been explored for various applications, such as image recognition, semantic segmentation, object detection, natural image generation, etc. One of the most popular tasks in AutoML is neural architecture search (NAS) [4], which aims to design neural network architectures automatically without much human heuristics or assumptions. Besides the model weights, the model structure itself becomes fitted for the task after searching, and is even transferable to different applications [5]. Additional constraints, such as latency or parameter quantity of models, can be added as searching objectives to fit the models into different computing platforms. Several works [6] have achieved state-of-the-art (SOTA) performance with NAS in large-scale image recognition (e.g., on ImageNet [7]). Other works applied similar techniques to optimize other components (e.g., augmentations, loss functions) or hyper-parameters in conventional deep learning solutions. Although the recent development of AutoML has shown promising results in reducing the workload of algorithm design and improving model performance, most AutoML methods only optimize few com-
ponents of the framework, e.g., network or transform, and continue to rely on the human choice of other components, which can lead to sub-optimal solutions.

In this paper, we propose a new method to automatically estimate “almost” all components of a deep learning solution for lesion segmentation in 3D medical images. At first, a new search space for segmentation networks is introduced to enable flexible connection of global network structure beyond the U-shape design (encoder-decoder based models), which is commonly used in medical image segmentation [8]. Then, candidates of deep learning configurations (neural architecture, augmentations, hyper-parameters) are encoded to a 1D vector as the abstract representation. Next, a binary relation predictor is trained with the representative vectors of configurations and their corresponding validating metrics. More specifically, the predictor distinguishes between two input vectors to see if one could lead to better performance than the other. Given such predictors, all configurations of deep learning solutions can be sorted with a direct comparison. Finally, the search configurations are generated by sampling candidates from the candidate pool and picked by their predicted performance in the searched task of the lesion segmentation. Furthermore, the searched configurations can be transferred to similar tasks in different datasets and achieve reasonable performance.

The contributions of our proposed method can be summarized as follows.

1. We propose a new search space for lesion segmentation in 3D volumes, which has much more flexibility compared to the traditional U-shape architecture in medical imaging, and our architecture searching provides insights about the importance and fitness of various dense connections inside segmentation networks;

2. Our method reaches the state-of-the-art performance of lesion segmentation on two public datasets;

3. Both the search process and the configuration deployment of the proposed method is designed to be computationally efficient and effective;

4. To the best of our knowledge, this is the first time that a comprehensive AutoML (including both NAS and optimization of other related training hyper-parameters) is applied to medical image segmentation leveraging the capacity of transformer modules.

2. Related Work

Lesion segmentation: Automated lesion segmentation in medical imaging has been studied for decades. Some early studies involved handcrafted rules such as convergence index filter [9] and multi-scale hessian-based blob measurements [10]. Later, researchers proposed to adopt machine learning models to tackle this challenging task. The standard pipeline is to extract a series of handcrafted features, then distinguish lesion from the background by utilizing various classifiers, including the k-nearest neighbor [11], random forest [12, 13], and support vector machine [14]. The models can further leverage the advantage from ensembling with sufficient interpretation capacity. However, the reported model is usually very sensitive to imaging appearance with numerous false positives [13]. Recently, deep neural networks have been introduced to localize and segment lesions from multiple modalities, including MRI [15], ultrasound [16], and CT [17, 18], given several publicly available annotated datasets. The majority of these techniques take a network specifically designed for one previous task, and then do the training by pre-processing the data of the current task to accommodate the network design.

Efficient ConvNet models for medical imaging: Researchers have adopted the design concepts of U-Net (“U”-shape convolutional encoder-decoder architecture with skip connections), and further extended it into novel and effective neural network architectures [19, 20, 21, 22, 23, 24, 25, 26, 27]. Nowadays, 2D/3D U-Net is considered as the common baseline model for most segmentation tasks. Often state-of-the-art performance in several tasks is reached via using variants of U-Net with model ensembling [28]. However, one potential downside of U-Net is that once the U-shape models are trained properly, their prediction relies more on local context instead of the global structure because of the multi-level skip connections. Then densely connected U-Net (e.g., U-Net++) was proposed to further improve the performance and generality of the segmentation models [29]. It gives us a hint that dense connection is helpful, but we have very limited intuition as to which connection plays a more important role and which connection can be pruned. Recently, researchers used neural architecture search to answer such questions to improve the performance of 3D segmentation networks further.

Automated machine/deep learning: Neural architecture search (NAS) is commonly used to design neural networks automatically with limited human heuristics to meet different user requirements (e.g., light-weight models or minimal computation) [30]. Zoph and Le firstly introduced a novel framework to conduct neural architecture search using reinforcement learning for large-scale image recognition [4, 5, 31]. Specifically, they formulated the NAS problem as searching convolutional layers for repeated modules inside the entire network, which is capable of reducing the searching space and cost to a great extent since the same module structure is applied for several modules in the entire architecture. Their searched networks (after being trained) are able to achieve state-of-the-art performance on the ImageNet dataset [32]. Others followed a similar idea to adopt reinforcement learning (RL) to tune the hyper-parameters.
of training settings or to find optimal data augmentation solutions [33, 34]. Using newly searched neural networks, researchers have improved the performance of various computer vision applications (semantic segmentation, object detection, design of loss function, etc.) in some challenging datasets [35, 36, 37, 38, 39]. Meanwhile, researchers have explored other concepts of NAS, i.e., “super-net” and one-shot NAS [40, 41, 42, 43, 44, 45, 46, 47, 48]. Another trend in NAS is predictor-based methods [49, 50, 51]. The predictors do not have performance gaps of searched architectures between searching and re-training. However, they require many trained models with validation accuracy as the ground truth to train a robust predictor model.

Recently, the related research has been widely developed in automated deep learning (AutoDL) to optimize almost every aspect of deep learning, including network architecture design [4, 52, 43, 6, 53, 54], data augmentation strategies [33, 34, 54], and loss functions [55], etc. With those automated components of deep learning, several computer vision applications successfully perform much better in various metrics (accuracy, latency, model size, etc.). However, previous works mainly focus on optimizing few components of a deep learning framework only. Our proposed framework aims to simultaneously search for the optimal combination of deep learning components, introducing greater flexibility to AutoDL.

3. T-AutoML

The success of deep learning comes from several aspects. One major advantage of deep learning is that neural networks are end-to-end trainable without the need for feature engineering. However, there is a new need for designing the best network architecture, which often determines the upper-bound performance of deep learning models. A good architecture enables effective gradient back-propagation during training and feature learning. To further promote the performance, data augmentation during training is utilized to increase model robustness and mitigate the gap between the domains of training, validation, and testing datasets. Last but not least, hyper-parameters in model training are also critical for fast convergence and decent accuracy.

Our proposed transformer-based AutoML (T-AutoML) method (see Sec. 3.3) trains a predictor to compare performance between different training configurations (neural architecture, data augmentation, and hyper-parameters). T-AutoML leveraging the advanced capacities of the transformer modules [56] to handle the input sequence with dynamic sequence, which is suitable for neural architecture space with a various number of blocks/layers. The proposed method is capable of covering (almost) all components in a conventional deep learning solution. The neural network architecture, data augmentation, and other related hyper-parameters can be fit into the method with proper encoding strategies. And the combination of encoding is the reference for the predictor to determine the optimal architecture and training configurations for the target tasks.

3.1. SpineNet Searching Space

U-shape neural networks for segmentation are very popular in modern deep learning since they are good at image-to-image translation [21, 2, 3, 19, 28]. The encoder reduces the feature maps gradually, and meanwhile increases the filter numbers. The decoder reverts back the spatial resolution gradually to match the input shape. Also, the skip connections between the encoder and decoder in the U-shape have been validated to be effective in capturing the low-level image details for the fine segmentation boundaries [57]. Although such a design is simple and straightforward, it produces promising results in many segmentation applications, and has become the go-to network in the field of medical image segmentation. However, there is no evidence that if the “encoder, decoder, and skip connection” design patterns are optimal for the task at hand.

Hence, we propose a new search space of neural architectures to further study the network connections for image segmentation. Our method is inspired by SpineNet [58], which was proposed for object detection in natural image processing. Under the framework, the feature maps at different spatial levels of the network can be connected with each other arbitrarily. At the same time, the order of operations to increase or decrease the feature maps’ spatial size can be arranged randomly. Thus, the search space not only contains U-shape networks or densely connected networks [29] but also covers other network topologies with asymmetric structure (see Fig. 2). Therefore, the search space is much larger than previous ones in the segmentation-related NAS literature [59, 60, 61, 62, 63, 64].

Like other methods in NAS, the search space consists of several blocks with different operations. In our method, the candidates of different blocks are 3D residual blocks, 3D bottleneck blocks, and 3D axial-attention blocks [65]. The residual and bottleneck blocks are effective in avoiding vanishing gradient [66]. The axial-attention block was originally proposed to resolve the issue of weak long-range dependency in the 2D plane for segmentation tasks [65]. Our axial-attention is the extension from 2D to two 3D versions. The first one is conducted axis-wise attention along $X$, $Y$, $Z$ axes sequentially. And the second one is conducted along $Y$, $X$, $Z$ axes sequentially. Since the axial plane ($X$-$Y$ plane) is commonly imaged in higher resolution and is more informative with low-level image details compared to $X$-$Z$ plane and $Y$-$Z$ planes, we let the axial attention block process the $X$-$Y$ plane first, and then conduct $Z$ axis-wise attention.

In practice, we construct the architecture with $N$ blocks...
one-by-one. Whenever introducing a new block \( c \) to the architecture, its category and spatial resolution level of the block needs to be determined first. From the 3rd block, the block \( c_i \) would collect feature maps from two of all precedent blocks \( c_j \) and \( c_k \), and combine them to a single feature map (\( c_2 \) would only receive the feature map from \( c_1 \). \( i, j, k \) are not necessarily adjacent). In order to combine layers from different spatial resolutions and match the resolution of the current block, necessary up-sampling and down-sampling are applied to the feature maps, respectively. We change the spatial resolution of combined feature maps to the target spatial resolution following the fashion from [58]. Then the combined feature maps would be converted to \( c_i \)'s spatial resolution level with necessary up-sampling/down-sampling. Lastly, the \( \mathcal{N} \)'-th block is the one before the final activation layer (softmax layer to generate multi-class probability maps). The connection between previous blocks and the current one can be arbitrary. In order to further reduce GPU memory consumption during training and meanwhile speed-up the training process, we have one stem layer to down-sample input volume by half of its original size using \( 3 \times 3 \times 3 \) convolution following [43]. At the end of the architecture, another up-sampling layer (linear interpolation) is used to restore the feature maps back to the original volume size.

### 3.2. Encoding

To represent architecture and other training configurations and simplify the computation of the next step, we encode architecture and training configuration in the search space jointly to form a “large” one-dimensional vector \( \mathcal{V} \). \( \mathcal{V} \) encodes both numerical values and non-numerical values (e.g., choices of optimizers/losses/data augmentation).

The architecture is encoded as a one-dimensional vector \( \mathcal{A} \) with dynamic length. For each block, we use five integer indices to represent its current block ID, choice of operations, spatial resolution level, and two IDs of predecessor blocks, respectively. The IDs of predecessor for the 1st block is \((-1, -1)\), for the 2nd block is \((0, -1)\).

During training, we apply \( n = 5 \) augmentation methods in a sequence. Therefore, we have \( n \) place holders for \( m \) augmentation candidates. For each place holder, we use indices \((0 \text{ to } m - 1)\) to indicate the choice of augmentation method. Thus, the 1D vector for augmentation has length \( n \).

Meanwhile, we encode the options of different optimizers and loss functions using integer indices as well. The other related hyper-parameters (e.g., learning rate) can be further optimized as long as they can be formulated into continuous or discrete values. After encoding all necessary components in the search space, we concatenate all 1D vectors into one large vector \( v \).

**Search space:** The search space is designed to cover most components in a typical deep learning framework. For data augmentation, the candidates are random flipping (along \( X, Y, Z \) axes respectively), random rotation (90 degrees) in \( X-Y \) planes, random zooming, random Gaussian noise, random intensity shift, random intensity scale shift. By default, we set the probability of activation (of each augmentation) to 0.15 according to the recommendations in [67]. The candidate learning rates are \([0.01, 0.005, 0.001, 0.0005, 0.00001] \), and the candidate learning rate schedulers include constant and polynomial.
We formulate the predictor as a binary classifier instead of an accuracy regressor. Once the predictor is trained, it can be used to rank unseen configurations with a sorting algorithm. For each configuration, we compare it with all other sampled configurations. It would be indexed with how many configurations have worse validation accuracy than it. And sorting can be simply conducted based on such indices. The comparison between \( v_i \) and \( v_j \) is light-weighted using CPU or GPU. Thus, the sorting would be finished in seconds for hundreds of randomly sampled candidates.

The advantage of this predictor design is that it requires less overall training time. Firstly, various configurations can be compared with fewer training iterations, since the absolute values of predicted accuracy are not always necessary. Especially when transferring the predictor to another dataset, the predicted accuracy becomes much less informative. The actual ranking between configuration \( v_i \) and \( v_j \) is more informative to a new task without any search/training experience, compared to the predicted accuracy with the previous predictor. Second, achieving a stable predictor (accuracy regressor) requires at least hundreds of ground truth (GT) data points, which also increases the overall time cost [49]. In our method, the binary relation predictor requires much fewer GT points in order to learn a similar amount of parameters of the predictors. For instance, training 20 jobs can create only 20 GT points for an accuracy-based predictor. However, the same amount of trained jobs/instances can create \( 20 \times 20 = 400 \) GT points for the relation-based predictor. Therefore, such a relation can be estimated with less training time/iterations.

### 4. Experiments

#### 4.1. Datasets

LiTS 2017: The LiTS challenge is about developing automatic segmentation algorithms to segment liver lesions in contrast-enhanced abdominal CT scans [71]. It has been publicly available since 2017, and more than 3000 participants in the challenges made their submissions over the years. CT volumes and ground truth labels are provided by multiple clinical sites around the world. The dataset contains 201 3D abdominal CT volumes with ground truth labels of liver and lesion segmentation. 131 of them are used for model training, and 70 volumes are used for testing on a public server. The ground truth labels of the test set are not visible to the participants. The spacing of 3D CT volumes varies from \( 0.598 \times 0.598 \times 0.450 \) mm\(^3\) to \( 0.977 \times 0.977 \times 6.0 \) mm\(^3\), and the range of volume shapes is from \( 512 \times 512 \times 42 \) voxel\(^3\) to \( 512 \times 512 \times 1026 \) voxel\(^3\). Large variance also exists in the imaging quality, contrast, and field-of-view (FOV) (shown in Fig. 1). The dataset has been largely adopted for performance evaluation of 3D segmentation methods by many es-
established works. In practice, the image intensity is clipped within an abdominal window of [-125, 225] Hounsfield units (HU) and pre-processed using standard normalization. The dataset is re-sampled into the voxel spacing $(1.0, s_y, s_z) \text{ mm}$, $(1.0, s_y) \text{ mm}$, $1.0 \text{ mm}$ based on each case’s spacing $s$. Such re-sampling strategies are proved to be capable of preserving some small lesions in the high-resolution CT images.

**Medical Segmentation Decathlon:** The medical decathlon challenge (MSD) hosts several tasks of 3D medical image segmentation [72]. We choose to take 06 (lung lesion/tumor segmentation in 3D CT images) to test the transferability of our searched configurations from previous LiTS datasets. The dataset contains 63 volumes for training and validation, and 32 volumes for testing. The spacing of 3D CT volumes varies from $0.598 \times 0.598 \times 0.625 \text{ mm}^3$ to $0.977 \times 0.977 \times 2.5 \text{ mm}^3$, and the range of volume shapes is from $512 \times 512 \times 84$ to $512 \times 512 \times 947$. The image intensity is clipped in a chest window of [-1000, 1000] HU and pre-processed using linear mapping to [0, 1]. The dataset is re-sampled into the isotropic voxel spacing $1.0 \text{ mm}$.

**Implementation:** The task for searching is the liver and lesion segmentation on LiTS dataset. The segmentation model takes a 1-channel input and outputs 3-class probability maps (background, liver, and lesion, respectively) with the same shape as the input. The task 06 of MSD is utilized to verify the transferability of our searched deep learning configurations from LiTS. The task is formulated as a binary segmentation problem. The last layer of search network is modified to take a 1-channel input and output 2-class probability maps (background, lung lesion).

To train and validate the predictor, we uniformly sample 100 configuration candidates from the search space. 75 of them are used for predictor training, and the remaining 25 are used for validation. Then, we train each candidate configuration for 10,000 iterations, and validate the segmentation model every 1,000 iterations. The best validation Dice score is referred to as the GT accuracy for that configuration. Once all the GT points are generated, the predictor model would be trained for 10,000 iterations with Adam optimizer, learning rate 0.001, and batch size 32. At last, we select the optimal configuration from the predictor with 200 candidates (100 existing ones and another 100 unseen random samples) to be our final model training solution. We use single GPU training jobs/instances for the configuration search. Each job takes about 3 hours for both training and validation, and the total searching time is around 300 GPU hours for the task. But our search process can be fully parallel, and thus searching would be done within two days using an 8-GPU server. Our searching efficiency is decent considering the huge search space. For example, C2FNAS used thousands of GPU hours to search for neural architectures (3D segmentation network) only [63]. The predictor model takes few minutes to train.

Our searched neural network is shown in Fig. 2. It has the least amount of parameters (16.96M), compared with other popular network models (19.07M for 3D U-Net [3], 22.60M for 3D U-Net++ [29], 17.02M for C2FNAS [63]). We reuse the configuration for Tab. 1 from the search method to determine all necessary components. During training, the input of the network are patches with size $128 \times 128 \times 128$, and the ratio between foreground and background patches is 1 : 1. The batch size is 4 (2 patches from 2 volumes) per GPU. To achieve better and robust segmentation performance, we linearly extend the number of total training iterations to 40,000 with the same learning rate scheduler as the searched one. The validation is conducted per 1,000 iterations to select the best model checkpoint. The validation accuracy is measured with the Dice score. The model inference uses a sliding-window scheme, and the overlapped region of adjacent windows is 80% of the window size. Similar to [28, 67], we conducted 5-fold cross-validation for all tasks, resulting in 5 segmentation models after training. The final prediction of test data is the ensemble result of the probability maps from the 5 models. Our proposed method is implemented with PyTorch and trained on two NVIDIA V100 GPUs with 16 GB memory.

<table>
<thead>
<tr>
<th>component</th>
<th>searched result</th>
</tr>
</thead>
<tbody>
<tr>
<td>data augmentation</td>
<td>random flip, random gaussian noise, random intensity shift, random zoom, random intensity scale shift</td>
</tr>
<tr>
<td>learning rate (LR)</td>
<td>0.0001</td>
</tr>
<tr>
<td>LR scheduler</td>
<td>constant</td>
</tr>
<tr>
<td>loss function</td>
<td>Dice loss (no squared prediction) + CE</td>
</tr>
<tr>
<td>optimizer</td>
<td>Adam</td>
</tr>
</tbody>
</table>

Table 1. The final searched training configuration for liver and lesion segmentation using LiTS dataset.

**4.2. Results**

**Evaluation:** We evaluate the segmentation accuracy in terms of the Dice score per case for LiTS challenge, and the Dice score and the normalized surface distance (NSD) for the lung task of the MSD challenge. Tab. 2, Tab. 3 and Fig. 4 results demonstrate the superior performance of our method compared to the other SOTA methods.

For the LiTS dataset, our method performs better for both lesion and liver segmentation. Tab. 2 shows that our method was able to capture not only the lesion regions,
but also the fine boundary of the large organs. Unlike other works, our method does not use any advanced model training features, such as sophisticated boundary loss functions [73] or deep supervision [28]. Our method benefits from fundamental components of neural network models and the training process. For the lung task of the MSD challenge, our method performs better than other SOTA methods with substantial improvements in terms of both the Dice score and the NSD. In the evaluation system of MSD, the higher the NSD value is, the better the segmentation quality is provided by the models. By inspecting the MSD leaderboard statistics of other methods for failed cases, we observe that all previous methods tend to make false negative predictions in a few specific test volumes (because the min metric values are close to 0). Whereas our method succeeds in such cases by capturing lesion regions well. Moreover, our method has clear advantage in 25 percentile and standard deviation of overall metric values, which shows our proposed method has both accurate and robust performance in this task. The fact that we migrated the found solution from LiTS to MSD lung task without any additional search, validates the transferability of our method.

### Table 2. LiTS challenge test-set performance evaluation for lesion and liver segmentations in terms of the average Dice score per case. The metrics of our method are copied from the LiTS leaderboard, and the metrics of the other methods are copied from their respective publications and the leaderboard.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean Dice Score</th>
<th>Std Dice Score</th>
<th>Min Dice Score</th>
<th>25% Dice Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA-UNet [74]</td>
<td>0.5950</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AH-Net [75]</td>
<td>0.6340</td>
<td>0.9630</td>
<td>0.7895</td>
<td></td>
</tr>
<tr>
<td>FCN [76]</td>
<td>0.6610</td>
<td>0.9510</td>
<td>0.8060</td>
<td></td>
</tr>
<tr>
<td>3D-DenseUNet [77]</td>
<td>0.6960</td>
<td>0.9620</td>
<td>0.8290</td>
<td></td>
</tr>
<tr>
<td>H-DenseUNet [20]</td>
<td>0.7220</td>
<td>0.9610</td>
<td>0.8415</td>
<td></td>
</tr>
<tr>
<td>LW-HCN [78]</td>
<td>0.7300</td>
<td>0.9650</td>
<td>0.8475</td>
<td></td>
</tr>
<tr>
<td>U3-Net [79]</td>
<td>0.7360</td>
<td>0.9638</td>
<td>0.8504</td>
<td></td>
</tr>
<tr>
<td>Cascade U-ResNets [80]</td>
<td>0.7520</td>
<td>0.9490</td>
<td>0.8505</td>
<td></td>
</tr>
<tr>
<td>VolumetricAttention [81]</td>
<td>0.7410</td>
<td>0.9610</td>
<td>0.8510</td>
<td></td>
</tr>
<tr>
<td>MA-Net [82]</td>
<td>0.7490</td>
<td>0.9600</td>
<td>0.8545</td>
<td></td>
</tr>
<tr>
<td>DistanceMetric [73]</td>
<td>0.7640</td>
<td>0.9650</td>
<td>0.8645</td>
<td></td>
</tr>
<tr>
<td>nnU-Net [67]</td>
<td>0.7630</td>
<td>0.9670</td>
<td>0.8650</td>
<td></td>
</tr>
<tr>
<td>T-AutoML (ours)</td>
<td>0.7650</td>
<td>0.9670</td>
<td>0.8660</td>
<td></td>
</tr>
</tbody>
</table>

**Comparison with nnU-Net:** nnU-Net has been validated as the state-of-the-art method for several medical image segmentation tasks [28, 67]. It leverages 3 different types of 2D/3D U-Net architectures for prediction with ensembling. It requires training 15 models (3 networks for 5-fold cross validation) in parallel to determine which models are used to create an ensemble and generate the final prediction. Although we also conducted 5-fold cross validation using the searched architecture and training configuration, only one fixed neural architecture needs to be trained on each fold for a new task. This is because once the searching on a single task is accomplished, it can be applied to other similar tasks directly. Moreover, nnU-Net requires deep supervision during model training, which is unnecessary for our method. Therefore, the marginal effort of applying our searched model and training configuration to a new task is considerably less than nnU-Net. At the same time, our method achieved better performance and the overall segmentation quality.

### Table 3. MSD challenge performance evaluation for lung tumor segmentation in terms of the Dice score and the normalized surface distance (NSD, higher is better). The metrics of our method are copied from the MSD leaderboard, and the metrics of other methods are copied from their respective publications and the leaderboard. *The results of U-Net++ are from the same task, but using a much larger training dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Dice</th>
<th>NSD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
<td>std</td>
</tr>
<tr>
<td>MPUnet [83]</td>
<td>0.5900</td>
<td>-</td>
</tr>
<tr>
<td>C2FNAS [63]</td>
<td>0.7044</td>
<td>0.2099</td>
</tr>
<tr>
<td>nnU-Net [67]</td>
<td>0.7397</td>
<td>0.2164</td>
</tr>
<tr>
<td>T-AutoML (ours)</td>
<td>0.7533</td>
<td>0.1574</td>
</tr>
</tbody>
</table>

**Comparison with nnU-Net:** nnU-Net is recognized as the state-of-the-art method for several medical image segmentation tasks. It leverages 3 different types of 2D/3D U-Net architectures for prediction with ensembling. It requires training 15 models (3 networks for 5-fold cross validation) in parallel to determine which models are used to create an ensemble and generate the final prediction. Although we also conducted 5-fold cross validation using the searched architecture and training configuration, only one fixed neural architecture needs to be trained on each fold for a new task. This is because once the searching on a single task is accomplished, it can be applied to other similar tasks directly. Moreover, nnU-Net requires deep supervision during model training, which is unnecessary for our method. Therefore, the marginal effort of applying our searched model and training configuration to a new task is considerably less than nnU-Net. At the same time, our method achieved better performance and the overall segmentation quality.

### 4.3. Ablation Studies and Discussion

We conduct ablation studies to motivate our predictor choice (see Fig. 5). For fair comparisons, we utilize three different types of predictors to rank data points in the validation set. The accuracy-based predictor generates accuracy for each validation data point, and the relation and transformer based relation predictor generates ranks after sorting. The regular relation predictor is based on a multi-layer perceptron (MLP). From Fig. 5, we see the accuracy predictor does not produce a positive correlation ($\rho = -0.322$) between ground truth ranks and predicted ranks (based on linear regression). But both relation-based predictors can rank them in a positively correlated fashion. Especially the relation-based predictor captures the top ranking points (left bottom corner of Fig. 5). And the transformer-based predictor has more than 20% improvement over the MLP-based predictor in terms of the corre-
Figure 4. The first row shows the superimposed display of the liver and lesion segmentation in CT and their 3D rendering results. The second row shows the superimposed display of lung lesion segmentation in chest CT and its 3D rendering.

Figure 5. Comparison of different predictors for ranking prediction in validation set with 25 uniformly sampled configurations.

Based on the results of LiTS, apart from lesion segmentation, our method also works very well on organ segmentation. However, search on organ segmentation may not be able to generate good architecture or training configurations for lesion segmentation (even in the same dataset). This is because organ body segmentation is much simpler than lesion segmentation (especially for healthy cases). And organ segmentation models typically converge much faster than lesion segmentation models. Most of the segmentation networks with simple settings would achieve decent performance, even without data augmentation. The liver segmentation model would converge within 2,000 iterations via a constant learning rate 0.001, and its validation accuracy achieves a Dice score above 0.950. However, using the same dataset, the lesion segmentation model would take more iterations (around 10,000) to converge and data augmentation is necessary. Because of the relative simplicity of liver segmentation, searched configuration did not fully exploit the potentials of models and training recipes. Therefore, searching on liver segmentation would result in an AutoML predictor with low discriminatory capacity among candidates. Hence, we suspect that challenging tasks (such as lesion segmentation) are more helpful for AutoML search in order to transfer the searched results to other applications.

5. Conclusion

In this paper, we proposed a novel AutoML method to optimize deep learning configurations for lesion segmentation in 3D medical images leveraging the advanced capacity of transformer modules. The proposed method predicts the relation between different training configurations and neural networks. We introduced a new search space for the neural architecture through modifications of SpineNet [58]. Meanwhile, we created a predictor-based AutoML algorithm, which is computationally efficient and effective. It can cover “almost” all components in conventional deep learning frameworks. Our experiments showed that, compared to other existing methods in the literature, our method can achieve the most advanced performance in large-scale lesion segmentation datasets. Furthermore, the proposed methods have been proven to be effectively transferable to different datasets. Future work could investigate how to further reduce searching and training time given certain computational budgets.
References


[42] Yingda Xia, Fengze Liu, Dong Yang, Jinzheng Cai, Lequan Yu, Zhuotun Zhu, Daguang Xu, Alan Yuille, and Holger
Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Xinyu Gong, Shiyu Chang, Yifan Jiang, and Zhangyang Lukasz Dudziak, Thomas C. P. Chau, Mohamed S. Abdelfat ...


