CAKES: Channel-wise Automatic KErnel Shrinking for Efficient 3D Networks

Qihang Yu, Yingwei Li, Jieru Mei, Yuyin Zhou, Alan Yuille
Johns Hopkins University
{yucornetto, meijieru, zhouyuyiner, alan.l.yuille}@gmail.com, yingwei.li@jhu.edu

Abstract

3D Convolution Neural Networks (CNNs) have been widely applied to 3D scene understanding, such as video analysis and volumetric image recognition. However, 3D networks can easily lead to over-parameterization which incurs expensive computation cost. In this paper, we propose *Channel-wise Automatic KErnel Shrinking (CAKES)*, to enable efficient 3D learning by shrinking standard 3D convolutions into a set of economic operations (*e.g.*, 1D, 2D convolutions). Unlike previous methods, CAKES performs channel-wise kernel shrinkage, which enjoys the following benefits: 1) enabling operations deployed in every layer to be heterogeneous, so that they can extract diverse and complementary information to benefit the learning process; and 2) allowing for an efficient and flexible replacement design, which can be generalized to both spatial-temporal and volumetric data. Further, we propose a new search space based on CAKES, so that the replacement configuration can be determined automatically for simplifying 3D networks. CAKES shows superior performance to other methods with similar model size, and it also achieves comparable performance to state-of-the-art with much fewer parameters and computational costs on tasks including 3D medical imaging segmentation and video action recognition. Codes and models are available at https://github.com/yucornetto/CAKES.

Introduction

3D learning has attracted more and more research attention with the recent advance of deep neural networks. However, 3D convolution layers typically result in expensive computation and suffer from convergence problems due to overfitting issues and the lack of pre-trained weights (Carreira and Zisserman 2017; Tajbakhsh et al. 2016).

To resolve the redundancy in 3D convolutions, many efforts have been investigated to design efficient alternatives. For instance, Qiu, Yao, and Mei (2017) and Tran et al. (2018) propose to factorize the 3D kernel and replace the 3D convolution with Pseudo-3D (P3D) and (2+1)D convolution, where 2D and 1D convolution layers are applied in a structured manner. Xie et al. (2018) suggest that replacing 3D convolutions with low-cost 2D convolutions at the bottom of the network significantly improves recognition efficiency.

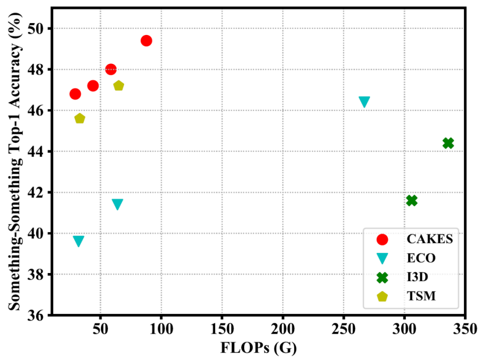

Figure 1: CAKES shows better accuracy-cost trade-off to its counterparts.

Despite their effectiveness for spatial-temporal information extraction, there are several limitations of existing alternatives to 3D convolutions. Firstly, these methods (*e.g.*, P3D) are specifically tailored to video datasets, where data can be explicitly separated into time and space. However, for volumetric data such as CT/MRI where all three dimensions should be treated equally, conventional spatial-temporal operators can lead to biased information extraction. Moreover, existing operations are still insufficient even for spatial-temporal data since they may exhibit certain levels of redundancy either along the temporal or the spatial dimension, as empirically suggested in Xie et al. (2018). Secondly, existing replacements are manually designed. Consequently, this process can be time-consuming and may lead to sub-optimal results.

To address these issues, we introduce *Channel-wise Automatic KErnel Shrinking (CAKES)*, as a general efficient alternatives to existing 3D operations. Specifically, the proposed method simplifies conventional 3D operations by adopting a combination of diverse and economic operations (*e.g.*, 1D, 2D convolutions), where these different operators can extract complementary information to be utilized in the same layer. Our approach is not tailored to any specific type of input (*e.g.*, videos), but can be generalized to different types of data and backbone architectures to achieve a fine-grained and efficient replacement.

As a proof test, our CAKES with a naive manual setting already exhibits superior performances compared with existing 3D replacements (Table 1 & 3). However, the manual selection of the set of replacing operators as well as their positioning requires trial-and-error. To further improve the performance and the model efficiency, we introduce a new search space consisting of computationally-efficient candidate operators, to facilitate the search for the optimal replacement configuration given a backbone architecture. With our search space design, the proposed CAKES is feasible to obtain a good architecture in several GPU days.

The proposed algorithm delivers high-performance and efficient models. As shown in Fig. 1, evaluated on both 3D medical image segmentation and video action recognition tasks, our method achieves a better accuracy-cost trade-off. Compared with its 3D baseline, CAKES not only shows superior performance but also effectively reduces the model size (56.80% less on medical and 19.35% less on video) and computational cost (53.76% less on medical and 19.01% less on video) significantly. The proposed method surpasses their 2D/3D/P3D counterparts significantly.

Our contributions can be summarized into three folds:

(1) We propose a more generic, efficient and flexible alternative to 3D convolution by shrinking 3D kernels into heterogeneous yet complementary efficient counterparts at a fine-grained level.

(2) We automate the replacement configuration for simplifying 3D networks by customizing a search space based on CAKES and combining it with neural architecture search.

(3) By applying CAKES to different 3D models, we achieve comparable results to state-of-the-art while being much more efficient on both volumetric medical data and temporal-spatial video data.

Related Work

Efficient 3D Convolutional Neural Networks

Despite the great advances of 3D CNNs (Carreira and Zisserman 2017; Çiçek et al. 2016; Tran et al. 2015; Zhou et al. 2019a), existing 3D networks usually require heavy computational budget. Besides, 3D CNNs also suffer from unstable training due to lack of pre-trained weights (Carreira and Zisserman 2017; Liu et al. 2017; Tajbakhsh et al. 2016). These facts have motivated researchers to find efficient alternatives to 3D convolutions. For example, it is suggested in Luo and Yuille (2019); Tran et al. (2019) to apply group convolution (Krizhevsky, Sutskever, and Hinton 2012) and depthwise convolution (Chollet 2017) to 3D networks to obtain resource-efficient models. Another type of approach suggests replacing each 3D convolution layer with a structured combination of 2D and 1D convolution layers to achieve better performance while being more efficient. For instance, Qiu, Yao, and Mei (2017) and Tran et al. (2018) propose to use a 2D spatial convolution layer followed by a 1D temporal convolution layer to replace a standard 3D convolution layer. Besides, Xie et al. (2018) demonstrate that 3D convolutions are not needed everywhere and some of them can be replaced by 2D counterparts. Similar attempts also occur in the medical imaging area (Liu et al. 2018). For example, Gonda et al. (2018) try to replace consecutive 3D convolution layers through consecutive 2D convolution layers followed by a 1D convolution layer.

Our method differs from these methods by the following folds: (1) Instead of applying homogeneous operations to all channels, CAKES allows assigning complementary heterogeneous operations at channel-wise, which leads to a more flexible design and potentially better trade-off between accuracy and efficiency (Tan and Le 2019); and (2) We enable the automatic optimization of the replacement configuration instead of manual design through a new search space.

Neural Architecture Search

Neural Architecture Search (NAS) aims at automatically discovering better network architectures than human-designed ones. It has been proved successful not only for 2D natural image recognition (Zoph and Le 2017; Yu et al. 2020a), but also for other tasks such as segmentation (Liu et al. 2019) and detection (Ghiasi, Lin, and Le 2019). Besides the success on natural images, there are also some trials on other data formats such as videos (Ryoo et al. 2019) and 3D medical images (Yu et al. 2020b; Zhu et al. 2019). Earlier NAS algorithms are based on either reinforcement learning (Baker et al. 2017; Zoph and Le 2017; Zoph et al. 2018) or evolutionary algorithm (Real et al. 2019; Xie and Yuille 2017). However, these methods often require training each network candidate from scratch, therefore the intensive computational costs hamper its usage especially with limited computational budget. Since Pham et al. (2018) first proposed parameter sharing scheme, more and more search methods such as differentiable NAS approaches (Chen et al. 2019; Liu, Simonyan, and Yang 2019; Xu et al. 2020; Dong and Yang 2019) and one-shot NAS approaches (Brock et al. 2018; Guo et al. 2019; Stamoulis et al. 2019; Li et al. 2020) began to investigate how to effectively reduce the search cost to several GPU days or even several GPU hours.

Moreover, Gordon et al. (2018); Mei et al. (2020) successfully connect network pruning with NAS and design more efficient search methods. Some methods (Tan and Le 2019; Mei et al. 2020; Stamoulis et al. 2019) also incorporate the kernel size into the search space. Nevertheless, most of them only consider simple cases with choices among 3×3, 5×5, *etc.*, while we consider much more diverse and general kernel deployment across different channels in 3D settings.

Method

Revisit Variants of 3D Convolution

We first revisit 3D convolutions and existing alternatives. Without loss of generality, let $\mathbf{X}$ of size $C_i \times D_i \times H_i \times W_i$ denotes the input tensor, where C_i stands for the input channel number, and D_i, H_i, W_i represent the spatial depth (or temporal length), the spatial height, and the spatial width, respectively. The weights of the corresponding 3D kernel are denoted as $\mathbf{W}^{C_o \times C_i \times k_d \times k_h \times k_w}$, where C_o is the output channel number and $k_d \times k_h \times k_w$ denote the kernel size. For simplicity, we consider each output channel individually in formulation. Therefore, the output tensor $\mathbf{Y}$ of shape $C_o \times D_o \times H_o \times W_o$ can be derived as following:

$$\mathbf{Y}_c^{D_o \times H_o \times W_o} = \mathbf{X}^{C_i \times D_i \times H_i \times W_i} \oplus \mathbf{W}_c^{C_i \times k_d \times k_h \times k_w}, \quad (1)$$

where $\oplus$ denotes convolution, c is the output channel index, *i.e.*, $1 \leq c \leq C_o$.

The computation overhead of 3D convolutions can be significantly heavier than their 2D counterparts. Consequently, the expensive computation and over-parameterization induced by 3D deep networks impede the scalability of network capacity. Recently, there are many works seeking to alleviate the high demand of 3D convolutions. One common strategy is to decouple the spatial and temporal components (Qiu, Yao, and Mei 2017; Tran et al. 2018). The underlying assumption here is that the spatial and temporal kernels are orthogonal to each other, and therefore can effectively extract complementary information from different dimensions. Another option is to discard 3D convolutions and simply use 2D operations instead (Xie et al. 2018). Mathematically speaking, these replacements can be written as:

$$\mathbf{W}_c^{C_i \times k_d \times k_h \times k_w} \leftarrow \{\mathbf{W}_c^{C_i \times 1 \times k_h \times k_w}, \mathbf{W}_c^{C_i \times k_d \times 1 \times 1}\} \quad (2)$$

$$\mathbf{W}_c^{C_i \times k_d \times k_h \times k_w} \leftarrow \{\mathbf{W}_c^{C_i \times 1 \times k_h \times k_w}\}, \quad (3)$$

where $\leftarrow$ indicates the replacement operation. Similar ideas also occur in 3D medical image analysis, where the images are volumetric data. For instance, it is shown in Liu et al. (2017) that using 2D convolutions in encoder and replacing 3D convolutions with Pseudo-3D (P3D) operations in decoder not only largely reduce the computation overhead but also improve the performance over the traditional 3D networks.

Though these methods have furthered the model efficiency compared with standard 3D convolutions, there are several limitations yet to be tackled. On the one hand, as shown in Eqn. (2), decomposing the kernels into orthogonal 2D and 1D components is designed for a specific data type (*i.e.*, spatial-temporal), which may not well generalize to other types such as volumetric data. On the other hand, directly replacing 3D kernels with 2D operators (Eqn. (3)) cannot effectively capture information along the third dimension.

To address these issues, we propose *Channel-wise Automatic KErnel Shrinking (CAKES)*, as a general alternative to 3D convolutions. The core idea is to shrink standard 3D kernels into a set of cheaper 1D, 2D, and 3D components. To ensure the flexibility of our design and avoid the tricky manual configuration, we further make the shrinkage channel-specific, thus heterogeneous kernels can extract complementary information as a 3D kernel does. We additionally introduce a brand-new search space so that the replacement configuration can be optimized automatically.

Kernel Shrinking as Path-level Selection

Let's consider the case for single output channel, and abbreviate $\mathbf{W}_c^{C_i \times k_d \times k_h \times k_w}$ to $\mathbf{W}_c^{k_d \times k_h \times k_w}$ for simplicity. We aim to find the optimal sub-kernel $\mathbf{W}_c^{k_d^{'} \times k_h^{'} \times k_w^{'}}$ ($1 \leq k_d^{'} \leq k_d$,$1 \leq k_h^{'} \leq k_h$,$1 \leq k_w^{'} \leq k_w$) as the substitute for 3D kernel $\mathbf{W}_c^{k_d \times k_h \times k_w}$. Therefore, the original 3D kernels can be effectively reduced to smaller sub-kernels, leading to a more efficient model.

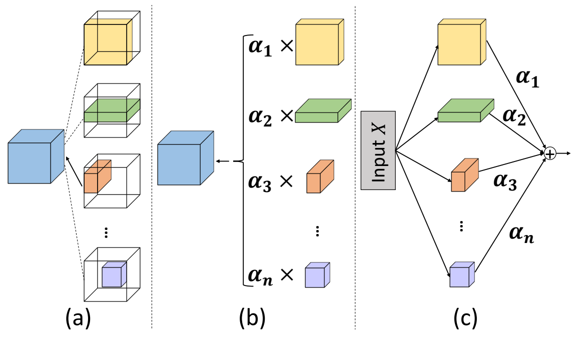

Figure 2: (a) Various sub-kernels of the same 3D kernel. (b) Representation of 3D kernel as weighted summation of sub-kernels. (c) Path-level selection.

As shown in Fig. 2(a), even only considering different kernel sizes, there are $k_d \times k_h \times k_w$ sub-kernel options for a 3D kernel, which makes it impractical to find the optimal sub-kernel via manual designs. Therefore, we provide a new perspective—to formulate this problem as path-level selection (Liu, Simonyan, and Yang 2019), *i.e.*, to encode sub-kernels into a multi-path super-network and select the optimal path among them (Fig. 2(c)). Then this problem can be solved in a differentiable manner.

We first represent a general replacement to 3D kernel as follows (Fig. 2(b)):

$$\mathbf{W}_c^{k_d \times k_h \times k_w} \leftarrow \{\alpha_i \mathbf{W}_c^{k_d^i \times k_h^i \times k_w^i}\}_i, \quad (4)$$

where α_i is the weight of i-th sub-kernel $\mathbf{W}_c^{k_d^i \times k_h^i \times k_w^i}$, $1 \leq k_d^i \leq k_d$, $1 \leq k_h^i \leq k_h$, $1 \leq k_w^i \leq k_w$. With this formulation, the problem of finding the optimal sub-kernel of $\mathbf{W}_c^{k_d \times k_h \times k_w}$ can be approximated as finding the optimal setting of $\{\alpha_i\}$ and then keeping the sub-kernel with maximum α_i. Due to the linearity of convolution, Eqn. (1) can then be derived as below:

$$\mathbf{X} \oplus \mathbf{W}_c^{k_d \times k_h \times k_w} \leftarrow \sum_i \alpha_i (\mathbf{X} \oplus \mathbf{W}_c^{k_d^i \times k_h^i \times k_w^i}). \quad (5)$$

To solve for the path weights $\{\alpha_i\}$, we reformulate Eqn. (5) as an over-parameterized multi-path super-network, where each candidate path consists of a sub-kernel (Fig. 2(c)). By relaxing the selection space, *i.e.*, relaxing the conditions on α to be continuous, Eqn. (5) can be then formulated as a differential NAS problem and optimized via gradient descent (Liu, Simonyan, and Yang 2019).

Channel-wise Shrinkage

While previous replacements (Liu et al. 2017; Qiu, Yao, and Mei 2017; Tran et al. 2018) consist of homogeneous operations in the same layer, we argue that a more efficient replacement requires customized operations at each channel. As shown in Fig. 3, kernel shrinking in a channel-wise fashion can generate heterogeneous operations which extract diverse and complementary information within the same layer, and thereby yields a fine-grained and more efficient replacement (Fig. 3(d)) than prior methods which use layer-wise replacements (Fig. 3(a) & (b) & (c)).

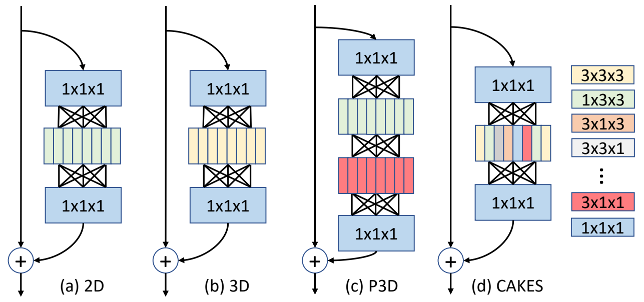

Figure 3: An illustrative example of comparison between different types of convolution in a residual block (He et al. 2016). (a) 2D Convolution. (b) 3D Convolution. (c) P3D Convolution. (d) the proposed CAKES. In our case, starting from a 3D convolution, the 3D operation at each channel is replaced with an efficient sub-kernel.

Contrary to previous layer-wise replacement, our core idea is to replace 3D kernel at each channel individually, thus the target is to find the optimal sub-kernel $\mathbf{W}_c^{k_d^c \times k_h^c \times k_w^c}$ as the substitute for the c-th output channel 3D kernel $\mathbf{W}_c^{k_d \times k_h \times k_w}$:

$$\mathbf{W}_c^{k_d \times k_h \times k_w} \leftarrow \{\mathbf{W}_c^{k_d^c \times k_h^c \times k_w^c}\}, \tag{6}$$

where the optimal size of the sub-kernel ($k_d^c \times k_h^c \times k_w^c$) is subjected to $1 \leq k_d^c \leq k_d$, $1 \leq k_h^c \leq k_h$, $1 \leq k_w^c \leq k_w$. Hence the computation incurred by Eqn. (1) can be largely reduced by our replacement as above.

With our channel-wise replacement design, the original 3D kernels are substituted by a series of diverse and cheaper operations at different channels as following (recall that C_o is the output channel number):

$$\mathbf{W} \leftarrow \{\mathbf{W}_1^{k_d^1 \times k_h^1 \times k_w^1}, \mathbf{W}_2^{k_d^2 \times k_h^2 \times k_w^2}, \dots, \mathbf{W}_{C_o}^{k_d^{C_o} \times k_h^{C_o} \times k_w^{C_o}}\}. \tag{7}$$

Benefited from channel-wise shrinkage, our method provides a more general and flexible design for replacing 3D convolution than previous approaches (Eqn. (2) and Eqn. (3)), where it can also be easily reduced to arbitrary alternatives ($e.g.$, 2D, P3D) by integrating these operations into the set of candidate sub-kernels. An illustration example can be found in Fig 3.

Search for an Efficient Replacement

As aforementioned, given the tremendous feasible choices, it is impractical to manually find the optimal replacement for a 3D kernel through a trial-and-error process. Especially, it becomes even more intractable as the replacement procedure is conducted in a channel-wise manner. Therefore, we propose a new search space for efficient 3D networks and automate the process of learning an efficient replacement to fully exploit the redundancies in 3D convolution operations. By formulating kernel shrinkage as a path-level selection problem, we first construct a super-network where every candidate sub-kernel is encapsulated into a separate trainable branch (Fig. 2(c)) at each channel. Once the path weights are learned in a differentiable manner, the optimal path (sub-kernel) can be determined.

Search Space. A well-designed search space is crucial for NAS algorithms (Yang, Esperança, and Carlucci 2020). Here we aim to answer the following questions: *Should the 3D convolution kernel be kept or replaced per channel? If replaced, which operation should be deployed instead?*

To address these questions, for each channel, we define a set $\mathcal{S}$, which contains all candidates of sub-kernels (replacement) given a 3D kernel $\mathbf{W}^{k_d \times k_h \times k_w}$:

$$\begin{aligned} \mathcal{S} = \{\mathbf{W}^{k_{d_1} \times k_{h_1} \times k_{w_1}}, \mathbf{W}^{k_{d_2} \times k_{h_2} \times k_{w_2}}, \dots, \mathbf{W}^{k_{d_n} \times k_{h_n} \times k_{w_n}}\} \\ \mathbf{W}_c^{k_d^c \times k_h^c \times k_w^c} = \text{Choose}(\mathcal{S}). \end{aligned} \tag{8}$$

As the original 3D convolution kernel can be considered sub-kernel of itself, *i.e.*, $\mathbf{W}^{k_d \times k_h \times k_w} \in \mathcal{S}$, it can be kept in the final configuration. The final optimal operation $\mathbf{W}_c$ is chosen among $\mathcal{S}$.

Another critical problem for NAS is how to reduce the search cost. To make the search cost affordable, we adopt a differentiable NAS paradigm where the model structure is discovered in a single-pass super-network training. Drawing inspirations from previous NAS methods, we directly use the scaling parameters in the normalization layers as the path weights α of the multi-path super-network (Eqn. (5)) (Gordon et al. 2018; Mei et al. 2020). And our goal is then equivalent to finding the optimal sub-network architecture based on the learned path weights. To achieve this goal, we introduce two different search manners which aim at either maximizing the performance or optimizing the computation cost of the sub-network as a search priority, named as performance-priority and cost-priority search, respectively.

Performance-Priority Search. The search aims to maximize the performance by finding the optimal sub-kernels given the backbone architecture. During the search procedure, following Bender et al. (2018); Brock et al. (2018), we randomly pick an operation for each channel at each iteration. This not only allows for memory saving by activating and updating one path per iteration but also propels the weights of the paths in the super-network training to be decoupled. After the super-network is trained, the operation with the largest path weight will be picked as the final choice for the given output channel:

$$\begin{aligned} \mathbf{W}_c^{k_d \times k_h \times k_w} \leftarrow \{\mathbf{W}^{k_{d_{i^*}} \times k_{h_{i^*}} \times k_{w_{i^*}}}\}, \\ \text{where } i^* = \text{argmax}_{i \in \{1 \cdots n\}}(\alpha_i). \end{aligned} \tag{9}$$

Cost-Priority Search. Performance-priority may neglect the possible negative effects on the computation cost. In order to obtain more compact models, we also introduce a "cost-priority" search method. Inspired by Mei et al. (2020), we search the model in a pruning manner with penalty on expensive operations. The outputs of each sub-kernels are concatenated and aggregated by the following $1 \times 1 \times 1$ convolution. To make the searched architecture more compact, we introduce a "cost-aware" penalty term—A lasso term on α which is used as the penalty loss to push many path weights to near-zero values. Therefore, the total training loss $\mathcal{L}$ can be written as:

$$\mathcal{L} = \mathcal{E} + \lambda \sum_i \beta_i |\alpha_i|, \tag{10}$$

Methods	Params (M)	FLOPs (G)	Pancreas DSC (%)	Tumor DSC (%)	Average DSC (%)
2D	11.29	97.77	79.16	43.02	61.09
3D	22.50	188.48	80.34	47.57	63.96
P3D	13.16	112.88	**80.36**	45.27	62.82
$\mathrm{CAKES}^{\mathrm{M}}_{\mathrm{1D}}$	7.56	67.53	79.77	42.73	61.25
$\mathrm{CAKES}^{\mathrm{M}}_{\mathrm{2D}}$	11.29	97.77	80.09	46.17	63.13
$\mathrm{CAKES}^{\mathrm{M}}_{\mathrm{1,2,3D}}$	11.41	99.17	79.82	45.27	62.55
$\mathrm{CAKES}^{\mathrm{P}}_{\mathrm{1D}}$	**7.56**	**67.53**	80.32	45.57	62.95
$\mathrm{CAKES}^{\mathrm{P}}_{\mathrm{2D}}$	11.29	97.77	80.05	48.51	64.28
$\mathrm{CAKES}^{\mathrm{P}}_{\mathrm{1,2,3D}}$	11.26	99.68	80.12	**48.72**	**64.42**
$\mathrm{CAKES}^{\mathrm{C}}_{\mathrm{1,2,3D}}$	9.72	87.16	80.34	47.95	64.15

Table 1: Comparison among different operations and configurations. The subscripts of 1D, 2D, and 3D indicate the dimensions of the operations being used. The superscripts "M", "P", "C" represent "Manual", "Performance-Priority", and "Cost-Priority" respectively.

where β_i is a "cost-aware" term to balance the penalty term, which is proportional to the parameters or FLOPs cost of the sub-kernel. In Table 1, we also empirically show that this term can lead to a more efficient architecture. The introduction of β_i aims at giving more penalty to "expensive" operations and leading to a more efficient replacement. λ is the coefficient of the penalty term, and $\mathcal{E}$ is the conventional training loss ($e.g.$, cross-entropy loss combined with the regularization term such as weight decay).

Experiments

3D Medical Image Segmentation

Dataset. We evaluate the proposed method on two public datasets: 1) Pancreas Tumours dataset from the Medical Segmentation Decathlon Challenge (MSD) (Simpson et al. 2019), which contains 282 cases with both pancreatic tumours and normal pancreas annotations; and 2) NIH Pancreas Segmentation dataset (Roth et al. 2015), consisting of 82 abdominal CT volumes. **For the MSD dataset**, we use 226 cases for training and evaluate the segmentation performance on the rest 56 cases. The resolution along the axial axis of this dataset is extremely low and the number of slices can be as small as 37. For data preprocessing, all images are resampled to an isotropic 1.0 mm^3 resolution. **For the NIH dataset**, the resolution of each scan is $512 \times 512 \times L$, where $L \in [181, 466]$ is the number of slices along the axial axis and the voxel spacing ranges from 0.5 mm to 1.0 mm. We test the model in a 4-fold cross-validation manner following previous methods (Zhou et al. 2017, 2019b).

Implementation Details. For all experiments, C2FNAS (Yu et al. 2020b) is used as the backbone architecture. When replacing the operations, we keep the stem (the first two and the last two convolution layers) as the same. For 3D medical image, for simplicity, we choose a set of most representative sub-kernels as $\mathcal{S}$. The operations set contains conv1D ($1 \times 1 \times 3$, $1 \times 3 \times 1$, $3 \times 1 \times 1$), conv2D ($1 \times 3 \times 3$, $3 \times 1 \times 3$, $3 \times 3 \times 1$) from different directions, and conv3D ($3 \times 3 \times 3$). For every 3D kernel at each output channel, a sub-kernel from $\mathcal{S}$ will be chosen as the replacement. **For manual settings**, we assign all candidates operations uniformly across the output channels. **For NAS settings**, we include both "performance-priority" and "cost-priority" search for performance comparison.

Method	Params (M)	Average DSC	Max DSC	Min DSC
C2F (Zhou et al. 2017)	268.56	82.37%	90.85%	62.43%
RSTN (Yu et al. 2018)	268.56	84.50%	91.02%	62.81%
C2F ResDSN (Zhu et al. 2018)	20.06	84.59%	91.45%	69.62%
V-NAS (Zhu et al. 2019)	29.74	85.15%	91.18%	70.37%
$\mathrm{CAKES}^{\mathrm{C}}_{\mathrm{1,2,3D}}$	**9.27**	84.85%	91.61%	59.32%
$\mathrm{CAKES}^{\mathrm{P}}_{\mathrm{1,2,3D}}$	11.26	**85.28%**	**91.98%**	**72.78%**

Table 2: Comparison with prior arts on the NIH dataset.

Training stage. For the MSD dataset, we use random crop with patch size of $96 \times 96 \times 96$, random rotation (0°, 90°, 180°, and 270°) and flip in all three axes as data augmentation. The batch size is 8 with 4 GPUs. We use SGD optimizer with learning rate starting from 0.01 with polynomial decay of power of 0.9, momentum of 0.9, and weight decay of 0.00004. The training lasts for 40k iters. The loss function is the summation of dice loss (Milletari, Navab, and Ahmadi 2016) and cross-entropy loss. For NIH dataset, the patch size is set as $96 \times 96 \times 64$, following the settings in Zhu et al. (2019). The found architecture will be trained **from scratch** to ensure fair comparison. Both the super-network and the found architecture are trained under the same settings as aforementioned. For search stage with "cost-priority" setting, a lasso term with coefficient $\lambda = 1.0 \times 10^{-4}$ is applied to the path weights. And it is further re-weighted by $\beta = \{\frac{9}{13}, \frac{3}{13}, \frac{1}{13}\}$ for 3D, 2D, 1D operations respectively, which is their ratio of the parameters. After the training process, the operation with the largest α is chosen as the final replacement for 3D operation for each channel.

Testing stage. We test the network in a sliding-window manner, where the patch size is $96 \times 96 \times 96$ and stride is $32 \times 32 \times 32$ for the MSD dataset and patch size is $96 \times 96 \times 64$ and stride is $20 \times 20 \times 20$ for NIH dataset. The result is measured with Dice-Sørensen coefficient (DSC) metric, 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 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.

Manual Settings vs. Auto Settings. As observed from Table 1, even under manual settings, CAKES is already much more efficient with slightly inferior performance ($e.g.$, from 3D to manual $\mathrm{CAKES}^{\mathrm{M}}_{\mathrm{2D}}$, parameters drop from 22.50M to 11.29M, and FLOPs drop from 188.48G to 97.77G, with performance gap of $< 1.0\%$). Besides, $\mathrm{CAKES}^{\mathrm{M}}_{\mathrm{2D}}$ outperforms its counterpart with standard convolution 2D layers by more than 2.0% with the same model size, which indicates the benefits of our design. In addition, with the proposed search space and method, CAKES can further reduce the performance gap and even surpasses the original 3D model with much fewer parameters and computations, *e.g.*, model size is reduced from 22.50M (3D) to 11.26M ($\text{CAKES}^{\text{P}}_{1,2,3\text{D}}$), and FLOPs drop from 188.48G (3D) to 99.68G ($\text{CAKES}^{\text{P}}_{1,2,3\text{D}}$), with a performance improvement of 0.46%. Compared with P3D, $\text{CAKES}^{\text{P}}_{1,2,3\text{D}}$ also yields superior performance (+1.60%) with a more compact model (11.26M vs. 13.16M), which further indicates the effectiveness of the proposed method.

Influence of the Search Space. From Table 1, we can see that using different search space, CAKES consistently outperforms its counterparts with standard 1D/2D/3D convolutions. Out of different search spaces, we find that $\text{CAKES}^{\text{P}}_{1\text{D}}$ (7.56M params and 67.53G FLOPs) offers the most efficient model with a comparable performance, while $\text{CAKES}^{\text{P}}_{2\text{D}}$ (11.29M params and 97.77G FLOPs) can already surpass the 3D baseline (22.50M params and 188.48G FLOPs) with half parameters and computation cost. After we enlarge the search space, $\text{CAKES}^{\text{P/C}}_{1,2,3\text{D}}$ obtains a configuration with even higher performance/efficiency (last 2 rows of Table 1).

Generalization to different backbone architectures. We also test our method on different backbone architectures. Applying $\text{CAKES}^{\text{C}}_{1,2,3\text{D}}$ to another strong model 3D ResDSN (Zhu et al. 2018; Li et al. 2019), our method consistently leads to a more efficient model with much fewer parameters (10.03M to 4.63M) and FLOPs (192.07G to 98.12G) with comparable performance (61.96% to 61.65%).

NIH Results. We compare CAKES with state-of-the-art methods in Table 2, where it can be observed that the proposed method leads to a much more compact model size compared to other models. For instance, our model size is more than 25$\times$ smaller than that of Zhou et al. (2017) and Yu et al. (2018). It is well worth noting that our model performed in a single-stage fashion already outperforms many state-of-the-art methods conducted in a two-stage coarse-to-fine manner (Zhou et al. 2017; Yu et al. 2018; Zhu et al. 2018) on the NIH pancreas dataset with much fewer model parameters and FLOPS. It is also noteworthy to mention that the applied architecture is searched from another dataset (MSD), where images are collected under different protocols and have different resolutions. This result indicates the generalization of our searched model. By directly applying the architecture searched on the MSD dataset, our method also outperforms Zhu et al. (2019) which was directly searched on the NIH dataset with less than half model size.

Action Recognition in Videos

Dataset. Something-Something V1 (Goyal et al. 2017) is a large scale action recognition dataset which requires comprehensive temporal modeling. There are totally about 110k videos for 174 classes with diverse objects, backgrounds, and viewpoints.

Implementation Details. We adopt ResNet50 (He et al. 2016) with pre-trained weight on ImageNet (Krizhevsky, Sutskever, and Hinton 2012) as our backbone. The 3D convolution weights are initialized by repeating 2D kernel by 3 times along the temporal dimension following (Carreira and Zisserman 2017), while 1D convolution weights are initialized by averaging the 2D kernel on spatial dimensions and then repeat by 3 times along temporal axis. For the temporal dimension, we use the sparse sampling method as in (Wang et al. 2016). For spatial dimension, the short side of the input frames are resized to 256 and then cropped to 224 $\times$ 224.

Model	Params (M)	FLOPs (G)	top1	top5
C2D	23.9	33.0	17.2	43.1
P3D	27.6	37.9	44.8	74.6
C3D	46.5	62.6	46.8	75.3
$\text{CAKES}^{\text{M}}_{1,2\text{D}}$	**20.1**	**28.0**	46.2	75.2
$\text{CAKES}^{\text{M}}_{2,3\text{D}}$	35.2	47.7	46.8	76.0
$\text{CAKES}^{\text{P}}_{1,2\text{D}}$	20.9	29.1	47.1	75.9
$\text{CAKES}^{\text{P}}_{2,3\text{D}}$	37.5	50.7	**47.4**	**76.1**
$\text{CAKES}^{\text{P}}_{1,2,3\text{D}}$	33.5	43.9	47.2	75.7
$\text{CAKES}^{\text{C}}_{1,2\text{D}}$	20.5	29.3	46.8	76.0
$\text{CAKES}^{\text{C}}_{2,3\text{D}}$	35.7	41.4	46.9	75.6
$\text{CAKES}^{\text{C}}_{1,2,3\text{D}}$	35.0	38.7	46.9	75.5

Table 3: Comparison among operations and configurations for ResNet50 backbone in terms of parameter number, computation amount (FLOPs), and performance on Something-Something V1 dataset.

Training Stage. We use random cropping and flipping as data augmentation. We train the network with a batch size of 96 on 8 GPUs with SGD optimizer. The learning rate starts from 0.04 for the first 50 epochs and decays by a factor of 10 for every 10 epochs afterwards. The total training epochs are 70. We also set dropout ratio to 0.3 following (Wang and Gupta 2018). The training settings remain the same for both final network and search stage, except that when searching with "performance-priority" we double the training epochs to ensure convergence, and with "cost-priority", we use a lasso term with $\lambda = 1.0 \times 10^{-4}$ and $\beta = \{\frac{9}{13}, \frac{3}{13}, \frac{1}{13}\}$ for 3D, 2D, 1D operations respectively.

Testing Stage. we sample the middle frame in each segment and perform center crop for each frame. We report the results of **single crop**, unless otherwise specified.

Ablation Study. We study the impacts of both different operations set and manual/auto configurations. The results are summarized in Table 3. Considering the spatial-temporal property of video data, we study the following different operations set: (1) Spatial 2D convolution and temporal 1D convolution; (2) Spatial 2D convolution and 3D convolution; (3) Spatial 2D, temporal 1D, and 3D convolutions.

Operation Set with 1D & 2D Sub-kernels. As shown in Table 3, $\text{CAKES}^{\text{C}}_{1,2\text{D}}$ surpass the 2D baseline by a large margin (+29.6%) , while the model size reduces by 14.23%. This suggests that TSN (Wang et al. 2016) may lack the ability to capture temporal information, therefore replacing some of the 2D operations to temporal 1D operations can significantly increase the performance and reduce the model

Method	Backbone Architecture	#Frame	FLOPs	#Param.	top1	top5
TSN (Wang et al. 2016)	ResNet-50	8	33G	24.3M	19.7	46.6
TRN-2stream (Zhou et al. 2018)	BNInception	8+8	-	36.6M	42.0	-
ECO (Zolfaghari, Singh, and Brox 2018)	BNIncep+3D Res18	8	32G	47.5M	39.6	-
ECO (Zolfaghari, Singh, and Brox 2018)	BNIncep+3D Res18	16	64G	47.5M	41.4	-
$\mathrm{ECO}_{En}Lite$ (Zolfaghari, Singh, and Brox 2018)	BNIncep+3D Res18	92	267G	150M	46.4	-
I3D (Carreira and Zisserman 2017)	3D ResNet-50	32×2clip	153G×2	28.0M	41.6	72.2
NL I3D (Wang et al. 2018)	3D ResNet-50	32×2clip	168G×2	35.3M	44.4	76.0
NL I3D+GCN (Wang and Gupta 2018)	3D ResNet-50+GCN	32×2clip	303G×2	62.2M	46.1	76.8
TSM (Lin, Gan, and Han 2019)	ResNet-50	8	33G	24.3M	45.6	74.2
TSM (Lin, Gan, and Han 2019)	ResNet-50	16	65G	24.3M	47.2	77.1
S3D (Xie et al. 2018)	BNInception	64	66.38G	-	47.3	78.1
S3D-G (Xie et al. 2018)	BNInception	64	71.38G	-	48.2	**78.7**
$\mathrm{CAKES}^{\mathrm{C}}_{1,2\mathrm{D}}$	ResNet-50	8	**29.3G**	**20.5M**	46.8	76.0
$\mathrm{CAKES}^{\mathrm{P}}_{2,3\mathrm{D}}$	ResNet-50	8	50.7G	37.5M	**47.4**	76.1
$\mathrm{CAKES}^{\mathrm{P}}_{1,2,3\mathrm{D}}$	ResNet-50	8	43.9G	33.5M	47.2	75.7
$\mathrm{CAKES}^{\mathrm{C}}_{1,2\mathrm{D}}$	ResNet-50	16	**58.6G**	**20.5M**	48.0	78.0
$\mathrm{CAKES}^{\mathrm{P}}_{2,3\mathrm{D}}$	ResNet-50	16	101.4G	37.5M	48.6	78.6
$\mathrm{CAKES}^{\mathrm{P}}_{1,2,3\mathrm{D}}$	ResNet-50	16	87.8G	33.5M	**49.4**	78.4

Table 4: Comparing CAKES against other methods on Something-Something V1 dataset. We mainly consider the methods that adopt convolutions in fully-connected manner and only take RGB as input for fair comparison.

size. Besides, it also surpasses P3D, where each 2D convolution is followed by a temporal 1D convolution, with a significant advantage on both performance (+2.0%) and model cost (25.72% fewer params and 53.19% fewer FLOPs), indicating CAKES makes better use of redundancies in the networks than P3D. Therefore, CAKES using operation set containing 1D and 2D sub-kernels can be an ideal design when looking for efficient video understanding networks.

Operation Set with 2D & 3D Sub-kernels. We aim to see how CAKES balances the trade-off between performance and model cost. From Table 3, $\mathrm{CAKES}^{\mathrm{C}}_{2,3\mathrm{D}}$ yields a much more compact model (-23.23%/33.87% params/FLOPs) with a comparable performance to C3D. Under "performance-priority" setting, $\mathrm{CAKES}^{\mathrm{P}}_{2,3\mathrm{D}}$ searches a slightly larger model , yet its performance boosts significantly to 47.4%.

Operation Set with 1D & 2D & 3D Sub-kernels. Compared to $\mathrm{CAKES}^{\mathrm{C}}_{2,3\mathrm{D}}$, $\mathrm{CAKES}^{\mathrm{C}}_{1,2,3\mathrm{D}}$ shows a similar performance with much fewer FLOPs (38.7G vs. 41.4G). Besides, under the "performance-priority" setting, $\mathrm{CAKES}^{\mathrm{P}}_{1,2,3\mathrm{D}}$ produces a comparable performance to $\mathrm{CAKES}^{\mathrm{P}}_{2,3\mathrm{D}}$ with less computation cost (43.9G vs. 50.7G). This result indicates that with a more general search space (*e.g.*, 1D, 2D, and 3D), the proposed CAKES can find more flexible designs, which lead to better performance/efficiency.

Results. A comparison with other state-of-the-art methods is shown in Table 4. We report the model performance under both 8-frame and 16-frame settings. Compared with other state-of-the-art methods, $\mathrm{CAKES}^{\mathrm{P}}_{2\mathrm{D},3\mathrm{D}}$ sampling only 8 frames can already outperform most current methods. With smaller parameters and FLOPs, $\mathrm{CAKES}^{\mathrm{P}}_{2\mathrm{D},3\mathrm{D}}$ surpasses those complex models such as non-local networks (Wang et al. 2018) with graph convolution (Wang and Gupta 2018). Comparing $\mathrm{CAKES}^{\mathrm{C}}_{1,2\mathrm{D}}$ to other efficient video understanding framework such as ECO (Zolfaghari, Singh, and Brox 2018) and TSM (Lin, Gan, and Han 2019), our model is not only more light-weight (58.6G vs. 64G/65G), but also delivers a better performance (48.0% vs. 41.4%/47.2%). And our best model $\mathrm{CAKES}^{\mathrm{P}}_{1,2,3\mathrm{D}}$ achieves a new state-of-the-art performance of 49.4% top-1 accuracy with a moderate model size. An interesting finding is that although $\mathrm{CAKES}^{\mathrm{P}}_{1,2,3\mathrm{D}}$ shows similar performances to $\mathrm{CAKES}^{\mathrm{P}}_{2,3\mathrm{D}}$ with 8-frame inputs, it achieves a much higher accuracy when it comes to the 16-frame scenario, which demonstrates that with a more general search space, $\mathrm{CAKES}^{\mathrm{P}}_{1,2,3\mathrm{D}}$ shows stronger transferability than other counterparts.

Conclusions

As an important solution to various 3D vision applications, 3D networks still suffer from over-parameterization and heavy computations. How to design efficient alternatives to 3D operations remains an open problem. In this paper, we propose Channel-wise Automatic KErnel Shrinking (CAKES), where standard 3D convolution kernels are shrunk into efficient sub-kernels at channel-level, to obtain efficient 3D models. Besides, by formulating kernel shrinkage as a path-level selection problem, our method can automatically explore the redundancies in 3D convolutions and optimize the replacement configuration. By applying on different backbone models, the proposed CAKES significantly outperforms previous 2D/3D/P3D and other state-of-the-art methods on both 3D medical image segmentation and action recognition from videos.

Ethics Statement

In this paper, we present a new operation as an efficient alternative to 3D convolution and a search space to automate the process of simplifying 3D networks. The findings described in this paper can potentially help reduce the computation burden especially when dealing with 3D data such as CT scan or video. For the research community, our finding sheds light on a new direction to design efficient 3D networks. We expect the methodology to be investigated further in the future to better understand and make full use of the redundancy in 3D networks. To the society, some applications can be greatly benefited from CAKES. For example, in medical applications where artificial intelligence is expected to help doctor to make diagnosis and treatment planning, model efficiency can be crucial for model deployment in real-world clinical flows. And it can also assist deploy the algorithm (*e.g.* on-device video recognition) to mobile devices which do not have strong computation ability.

However, we also note that there is a long-lasting debate on the impacts of AI on human world. As a method improving the fundamental ability of deep learning, our work also advances the development of AI, which means there could be both beneficial and harmful influences depending on the users.

References

Baker, B.; Gupta, O.; Naik, N.; and Raskar, R. 2017. Designing neural network architectures using reinforcement learning. *ICLR* .

Bender, G.; Kindermans, P.-J.; Zoph, B.; Vasudevan, V.; and Le, Q. 2018. Understanding and simplifying one-shot architecture search. In *ICML*, 550–559.

Brock, A.; Lim, T.; Ritchie, J. M.; and Weston, N. 2018. SMASH: one-shot model architecture search through hypernetworks. *ICLR* .

Carreira, J.; and Zisserman, A. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 6299–6308.

Chen, X.; Xie, L.; Wu, J.; and Tian, Q. 2019. Progressive Differentiable Architecture Search: Bridging the Depth Gap between Search and Evaluation. *ICCV* .

Chollet, F. 2017. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 1251–1258.

Çiçek, Ö.; Abdulkadir, A.; Lienkamp, S. S.; Brox, T.; and Ronneberger, O. 2016. 3D U-Net: learning dense volumetric segmentation from sparse annotation. In *MICCAI*.

Dong, X.; and Yang, Y. 2019. Searching for a robust neural architecture in four gpu hours. In *CVPR*, 1761–1770.

Ghiasi, G.; Lin, T.-Y.; and Le, Q. V. 2019. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*, 7036–7045.

Gonda, F.; Wei, D.; Parag, T.; and Pfister, H. 2018. Parallel separable 3D convolution for video and volumetric data understanding. *BMVC* .

Gordon, A.; Eban, E.; Nachum, O.; Chen, B.; Wu, H.; Yang, T.-J.; and Choi, E. 2018. Morphnet: Fast & simple resource-constrained structure learning of deep networks. In *CVPR*, 1586–1595.

Goyal, R.; Kahou, S. E.; Michalski, V.; Materzynska, J.; Westphal, S.; Kim, H.; Haenel, V.; Fruend, I.; Yianilos, P.; Mueller-Freitag, M.; et al. 2017. The" Something Something" Video Database for Learning and Evaluating Visual Common Sense. In *ICCV*, volume 1, 5.

Guo, Z.; Zhang, X.; Mu, H.; Heng, W.; Liu, Z.; Wei, Y.; and Sun, J. 2019. Single path one-shot neural architecture search with uniform sampling. *arXiv preprint arXiv:1904.00420* .

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 1097–1105.

Li, Y.; Jin, X.; Mei, J.; Lian, X.; Yang, L.; Xie, C.; Yu, Q.; Zhou, Y.; Bai, S.; and Yuille, A. L. 2020. Neural Architecture Search for Lightweight Non-Local Networks. In *CVPR*, 10297–10306.

Li, Y.; Zhu, Z.; Zhou, Y.; Xia, Y.; Shen, W.; Fishman, E. K.; and Yuille, A. L. 2019. Volumetric Medical Image Segmentation: A 3D Deep Coarse-to-Fine Framework and Its Adversarial Examples. In *Deep Learning and Convolutional Neural Networks for Medical Imaging and Clinical Informatics*, 69–91. Springer.

Lin, J.; Gan, C.; and Han, S. 2019. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, 7083–7093.

Liu, C.; Chen, L.-C.; Schroff, F.; Adam, H.; Hua, W.; Yuille, A. L.; and Fei-Fei, L. 2019. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *CVPR*, 82–92.

Liu, H.; Simonyan, K.; and Yang, Y. 2019. Darts: Differentiable architecture search. *ICLR* .

Liu, S.; Xu, D.; Zhou, S. K.; Mertelmeier, T.; Wicklein, J.; Jerebko, A.; Grbic, S.; Pauly, O.; Cai, W.; and Comaniciu, D. 2017. 3D Anisotropic Hybrid Network: Transferring Convolutional Features from 2D Images to 3D Anisotropic Volumes. *MICCAI* .

Liu, S.; Xu, D.; Zhou, S. K.; Pauly, O.; Grbic, S.; Mertelmeier, T.; Wicklein, J.; Jerebko, A.; Cai, W.; and Comaniciu, D. 2018. 3d anisotropic hybrid network: Transferring convolutional features from 2d images to 3d anisotropic volumes. In *MICCAI*, 851–858. Springer.

Luo, C.; and Yuille, A. 2019. Grouped Spatial-Temporal Aggretation for Efficient Action Recognition. In *ICCV*.

Mei, J.; Li, Y.; Lian, X.; Jin, X.; Yang, L.; Yuille, A.; and Yang, J. 2020. AtomNAS: Fine-Grained End-to-End Neural Architecture Search. *ICLR* .

Milletari, F.; Navab, N.; and Ahmadi, S.-A. 2016. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*, 565–571. IEEE.

Pham, H.; Guan, M. Y.; Zoph, B.; Le, Q. V.; and Dean, J. 2018. Efficient neural architecture search via parameter sharing. *ICML* .

Qiu, Z.; Yao, T.; and Mei, T. 2017. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, 5533–5541.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2019. Regularized evolution for image classifier architecture search. In *AAAI*, volume 33, 4780–4789.

Roth, H. R.; Lu, L.; Farag, A.; Shin, H.-C.; Liu, J.; Turkbey, E. B.; and Summers, R. M. 2015. Deeporgan: Multi-level deep convolutional networks for automated pancreas segmentation. In *MICCAI*.

Ryoo, M. S.; Piergiovanni, A.; Tan, M.; and Angelova, A. 2019. Assemblenet: Searching for multi-stream neural connectivity in video architectures. *arXiv preprint arXiv:1905.13209* .

Simpson, A. L.; Antonelli, M.; Bakas, S.; Bilello, M.; Farahani, K.; van Ginneken, B.; Kopp-Schneider, A.; Landman, B. A.; Litjens, G.; Menze, B.; et al. 2019. A large annotated medical image dataset for the development and evaluation of segmentation algorithms. *arXiv:1902.09063* .

Stamoulis, D.; Ding, R.; Wang, D.; Lymberopoulos, D.; Priyantha, B.; Liu, J.; and Marculescu, D. 2019. Single-path nas: Designing hardware-efficient convnets in less than 4 hours. *ECML PKDD* .

Tajbakhsh, N.; Shin, J. Y.; Gurudu, S. R.; Hurst, R. T.; Kendall, C. B.; Gotway, M. B.; and Liang, J. 2016. Convolutional neural networks for medical image analysis: Full training or fine tuning? *TMI* 35(5): 1299–1312.

Tan, M.; and Le, Q. V. 2019. Mixconv: Mixed depthwise convolutional kernels. *BMVC* .

Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; and Paluri, M. 2015. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 4489–4497.

Tran, D.; Wang, H.; Torresani, L.; and Feiszli, M. 2019. Video classification with channel-separated convolutional networks. In *ICCV*, 5552–5561.

Tran, D.; Wang, H.; Torresani, L.; Ray, J.; LeCun, Y.; and Paluri, M. 2018. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 6450–6459.

Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; and Van Gool, L. 2016. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 20–36. Springer.

Wang, X.; Girshick, R.; Gupta, A.; and He, K. 2018. Non-local neural networks. In *CVPR*, 7794–7803.

Wang, X.; and Gupta, A. 2018. Videos as space-time region graphs. In *ECCV*, 399–417.

Xie, L.; and Yuille, A. 2017. Genetic cnn. In *ICCV*, 1379–1388.

Xie, S.; Sun, C.; Huang, J.; Tu, Z.; and Murphy, K. 2018. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *ECCV*, 305–321.

Xu, Y.; Xie, L.; Zhang, X.; Chen, X.; Qi, G.-J.; Tian, Q.; and Xiong, H. 2020. Pc-darts: Partial channel connections for memory-efficient differentiable architecture search. *ICLR* .

Yang, A.; Esperança, P. M.; and Carlucci, F. M. 2020. NAS evaluation is frustratingly hard. *ICLR* .

Yu, J.; Jin, P.; Liu, H.; Bender, G.; Kindermans, P.-J.; Tan, M.; Huang, T.; Song, X.; Pang, R.; and Le, Q. 2020a. Bignas: Scaling up neural architecture search with big single-stage models. *ECCV* .

Yu, Q.; Xie, L.; Wang, Y.; Zhou, Y.; Fishman, E. K.; and Yuille, A. L. 2018. Recurrent saliency transformation network: Incorporating multi-stage visual cues for small organ segmentation. In *CVPR*, 8280–8289.

Yu, Q.; Yang, D.; Roth, H.; Bai, Y.; Zhang, Y.; Yuille, A. L.; and Xu, D. 2020b. C2fnas: Coarse-to-fine neural architecture search for 3d medical image segmentation. In *CVPR*, 4126–4135.

Zhou, B.; Andonian, A.; Oliva, A.; and Torralba, A. 2018. Temporal relational reasoning in videos. In *ECCV*, 803–818.

Zhou, Y.; Li, Y.; Zhang, Z.; Wang, Y.; Wang, A.; Fishman, E. K.; Yuille, A. L.; and Park, S. 2019a. Hyper-Pairing Network for Multi-phase Pancreatic Ductal Adenocarcinoma Segmentation. In *MICCAI*, 155–163. Springer.

Zhou, Y.; Xie, L.; Shen, W.; Wang, Y.; Fishman, E. K.; and Yuille, A. L. 2017. A fixed-point model for pancreas segmentation in abdominal CT scans. In *MICCAI*, 693–701. Springer.

Zhou, Y.; Yu, Q.; Wang, Y.; Xie, L.; Shen, W.; Fishman, E. K.; and Yuille, A. L. 2019b. 2D-Based Coarse-to-Fine Approaches for Small Target Segmentation in Abdominal CT Scans. In *Deep Learning and Convolutional Neural Networks for Medical Imaging and Clinical Informatics*, 43–67. Springer.

Zhu, Z.; Liu, C.; Yang, D.; Yuille, A.; and Xu, D. 2019. V-NAS: Neural Architecture Search for Volumetric Medical Image Segmentation. *3DV* .

Zhu, Z.; Xia, Y.; Shen, W.; Fishman, E. K.; and Yuille, A. L. 2018. A 3d coarse-to-fine framework for automatic pancreas segmentation. *3DV* .

Zolfaghari, M.; Singh, K.; and Brox, T. 2018. Eco: Efficient convolutional network for online video understanding. In *ECCV*, 695–712.

Zoph, B.; and Le, Q. V. 2017. Neural architecture search with reinforcement learning. *ICLR* .

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2018. Learning transferable architectures for scalable image recognition. In *CVPR*, 8697–8710.