UniNet: Unified Architecture Search with Convolution, Transformer, and MLP

Jihao Liu[1,2], Xin Huang[1], Guanglu Song[2], Hongsheng Li[1]✉, and Yu Liu[2]✉

[1] CUHK, MMLab
[2] SenseTime Research

Abstract. Recently, transformer and multi-layer perceptron (MLP) architectures have achieved impressive results on various vision tasks. However, how to effectively combine those operators to form high-performance hybrid visual architectures still remains a challenge. In this work, we study the learnable combination of convolution, transformer, and MLP by proposing a novel unified architecture search approach. Our approach contains two key designs to achieve the search for high-performance networks. First, we model the very different searchable operators in a unified form, and thus enable the operators to be characterized with the same set of configuration parameters. In this way, the overall search space size is significantly reduced, and the total search cost becomes affordable. Second, we propose context-aware downsampling modules (DSMs) to mitigate the gap between the different types of operators. Our proposed DSMs are able to better adapt features from different types of operators, which is important for identifying high-performance hybrid architectures. Finally, we integrate configurable operators and DSMs into a unified search space and search with a Reinforcement Learning-based search algorithm to fully explore the optimal combination of the operators. To this end, we search a baseline network and scale it up to obtain a family of models, named UniNets, which achieve much better accuracy and efficiency than previous ConvNets and Transformers. In particular, our UniNet-B5 achieves 84.9% top-1 accuracy on ImageNet, outperforming EfficientNet-B7 and BoTNet-T7 with 44% and 55% fewer FLOPs respectively. By pretraining on the ImageNet-21K, our UniNet-B6 achieves 87.4%, outperforming Swin-L with 51% fewer FLOPs and 41% fewer parameters. Code is available at https://github.com/Sense-X/UniNet.

Keywords: Deep learning architectures, neural architecture search

1 Introduction

Convolutional Neural Networks (CNNs) dominate the learning of visual representations and show effectiveness on various visual tasks, including image classification, object detection, semantic segmentation, etc. Recently, convolution-free backbones show impressive performances on image classification [7]. Vision Transformer (ViT) [8] demonstrates that pure transformer architecture that is mainly built on multi-head self-attentions (MSAs) can attain state-of-the-art performance when trained on large-scale datasets (e.g., ImageNet-21K, JFT-300M). MLP-Mixer [32] introduced a pure multi-layer perceptron (MLP) architecture that can almost match ViT's performance without using the time-consuming attention mechanism. The main operators in those networks perform differently in terms of efficiency and data utilization. On the one hand, convolutions in CNNs are locally connected and their weights are input-independent, which makes it effective at extracting low-level representations and efficient under the low-data regime. On the other hand, MSAs in the transformer capture long-range dependency, and the attention weights are dynamically dependent on the input representations. Hence, it is more data and computation demanding. The token-mixing in MLP-Mixer performs like a depthwise convolution of a full receptive field with parameter sharing, which is also data demanding. It is an important topic to study how to combine them effectively to form high-performance hybrid visual architectures, which, however, remains a challenge.

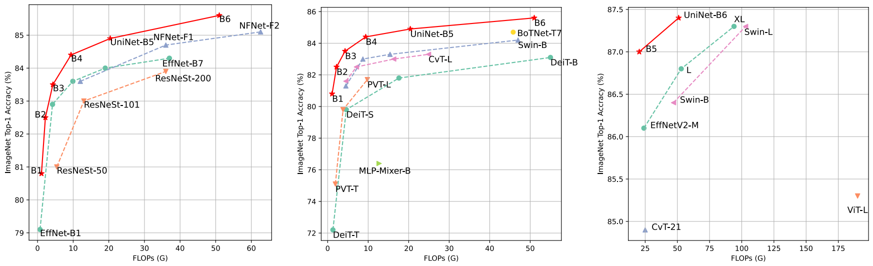
There were recent papers on attempting to manually combine the different types of operators to form hybrid visual networks. In ViT [8], a hybrid architecture using ResNet and transformer is also studied and improves upon pure transformers for smaller model sizes. Besides, many other works [6,5,43,42,13,11,9] also explored the combination of convolution and transformer to form hybrid architectures to improve data or computation efficiency. Furthermore, the combination of convolution and MLP is studied in [18], and the combination of gated MLP and MSA is studied in [19]. Those approaches focus on combining two distinct operators and can achieve satisfactory performances to some extent. However, a unified view and a systematical study are missed in prior arts.

We identify two key challenges when building high-performance hybrid architectures: (1) The operators can be implemented with various styles, and it is infeasible to manually explore all possible implementations and combinations. Although we can automate the exploration with Neural Architecture Search (NAS) techniques, the search space should be properly designed so that the search cost is affordable. (2) Each operator has its own characteristics, and simply combining them together does not lead to optimal results. We conduct a simple pilot study on directly stacking different operators to form hybrid networks. As shown in Table 1, however, the straightforward stacking of different operators achieves even worse performance than the vanilla ViT.

Table 1: ImageNet top-1 accuracy of different operator combinations. T and M refer to transformer block and MLP-Mixer block respectively. Different block numbers are chosen so that their computations are comparable.

Model	Configuration	#Params (M)	#FLOPs (G)	Top-1 Acc.
ViT	12 T	22	4.6	**78.0**
MLP-Mixer	18 M	23	4.7	76.8
ViT-MLP	7 T + 7 M	22	4.5	76.5
MLP-ViT	7 M + 7 T	22	4.5	77.8

In this paper, we study the learnable combination of convolution, transformer, and MLP by proposing a novel unified architecture search approach. Our approach has two key designs to address the challenges mentioned above. First, we model distinct operators in a unified form, and use the same set of searchable



(a) ConvNet

(b) Transformer/Hybrid

(c) ImageNet21K Transfer

Fig. 1: **ImageNet top-1 accuracy vs. FLOPs.** Our UniNet-B5 achieve 84.9% with ImageNet-1K dataset, outperforming EfficientNet-B7 and BoTNet-T7 with 44% and 55% fewer FLOPs, respectively. Our UniNet-B6 achieve 87.4% on ImageNet-1K with ImageNet-21K pre-training, outperforming EfficientNetV2-XL with 46% fewer FLOPs.

configuration parameters (i.e., *OP type*, *expansion*, *channels*, etc) to characterize each of the different operators. The unified design enables us to greatly reduce the overall search space, and as a result, the total search cost becomes affordable. Besides, we propose context-aware downsampling modules (DSMs) to harmonize the combination of different operators. The proposed DSMs can be instantiated into three types, i.e., Local-DSM (L-DSM), Local-Global-DSM (LG-DSM), and Global-DSM (G-DSM), aiming to better adapt the representations from one operator to another. Based on these designs, we build a unified search space consisting of a large family of different general operators (GOPs), DSMs, and network size, and jointly optimize model accuracy and FLOPs for identifying high-performance hybrid networks. We illustrate the search space and the backbone in Figure 2.

The discovered network, named UniNet, exhibits strong performance and efficiency improvements over common ConvNets, Transformers, or hybrid architectures on various visual benchmarks. Our experiments show that UniNet has the following characteristics: (1) placing convolutions in the shallow layers and transformers in the deep layers, (2) allocating a similar amount of FLOPs for both convolutions and transformers, and (3) inserting L-DSM to downsample for convolutions and LG-DSM for transformers. Our analysis shows that the conclusion is consistent among the top-5 models.

To go even further, we build a family of high-performance UniNet models by scaling up the searched baseline network, which achieves better accuracy and efficiency in both small and large model sizes. In particular, our UniNet-B5 achieves comparable accuracy (+0.1%) to EfficientNet-B7 while requires much less computation cost (-44%) (Figure 1 (a)). By pretraining on large-scale ImageNet-21K, our UniNet-B6 achieves 87.4% accuracy, outperforming Swin-L with fewer FLOPs (-51%) and parameters (-41%) (Figure 1 (c)).

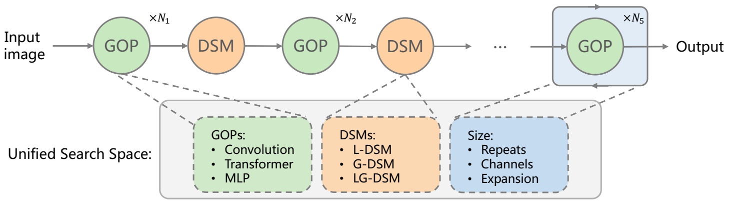


Fig. 2: **Unified Architecture Search.** We jointly search different types of operators as well as downsampling modules (DSM) and network size in a unified search space. We construct UniNet architecture in a multi-stage fashion. Between two successive stages, one of the DSMs is inserted to change the spatial dimension or channels.

2 Related Works

Convolution, Transformer, and MLP. A host of ConvNets have been proposed to push forward the state-of-the-art computer vision approaches such as [14,28,30]. Despite the numerous CNN models, their basic operators, convolution, are the same. Recently, [8] proposed a pure transformer-based image classification model ViT, which achieves impressive performance on the ImageNet benchmark. DeiT [34] shows that well-trained ViT can obtain a better performance-speed trade-off than ConvNets. PVT [41] and Swin [21] propose multi-stage vision transformers, which can be easily transferred to other downstream tasks. On the other hand, recent papers are attempting to use only MLP as the building block. MLP-Mixer [32], ResMLP [33], and ViP [15] show that pure MLP architectures can also achieve near state-of-the-art performance.

Combination of different operators. Another line of work tries to combine different operators to form new networks. CvT [42] propose to incorporate self-attention and convolution by generating `Q`, `K`, and `V` in self-attention with convolution. ConViT [6] tries to unify convolution and self-attention with gated positional self-attention and is more sample-efficient than self-attention. Many other works [5,13,11,9] also explored the combination of convolution and transformer to form hybrid architectures to improve the data or computation efficiency. Besides, ConvMLP [18] studied the combination of convolution and MLP, and gMLP [19] studied the combination of gated MLP and multi-head self-attentions (MSA). Instead of requiring manual exploration of the hybrid architectures, we propose a unified architecture search approach to automatically search for high-performance hybrid architecture.

3 Method

3.1 Unified Architecture Search

As discussed in previous works [6], an appropriate combination of convolution and transformer operators can lead to performance improvements. However, the previous approaches [42,43] only adopt convolution in self-attention or feed-forward network (FFN) sub-layers and stack them repeatedly. Their approaches did not fully explore the combinations to take advantage of their different characteristics.

Prior arts [40,44] show that the downsampling module plays an important role in visual tasks. Most previous approaches adopt hand-crafted downsampling operations, i.e., strided convolution, max-pooling, or avg-pooling, to downsample the feature map based on only the local context. However, these operations are specifically designed for ConvNets, and might not be suitable to the transformer or MLP based architectures, which capture representation globally.

In this paper, we investigate the learnable combination of convolution, transformer, and MLP[3], trying to assemble them to create high-performance hybrid visual network architectures. For better transmitting features across different operator blocks, we proposed context-aware downsampling modules. We jointly search the operators, downsampling modules, and network size in a unified search space. In contrast, previous Neural Architecture Search (NAS) works achieved state-of-the-art performances mainly via searching the network sizes. We show that the searched hybrid architecture by our unified architecture search approach can achieve very promising performance.

In the remaining parts of the section, we firstly present how to properly define different operators into a unified search space and search them jointly. We then present the challenge of incorporating downsampling modules with different operators and present our proposed context-aware downsampling module. Finally, we will introduce our UniNet architectures and NAS pipeline.

3.2 Modeling Convolution, Transformer, MLP with a Unified Searchable Form

Recently, transformer and MLP based architectures are able to achieve comparable performance to convolution networks on different visual tasks. To achieve better performance, it is intuitive to assemble all the types of operators to build high-performance hybrid networks. Actually, a few works [42,43,6] have been studied to empirically combine convolution and self-attention. However, manually searching network architectures is quite time-consuming and cannot ensure optimal performances with different computational budgets.

We introduce a unified search space that contains General Operators (GOPs, including convolution, transformer, and MLP), and then search for the optimal combination of those operators jointly. Compared with prior arts, we propose a unified form to characterize different operators. Specifically, we use the inverted residual [24] to model a general block, which first expands the input channel `c` to a larger size `ec`, and then projects the `ec` channels back to `c` for residual connection. The `e` is defined as the expansion ratio, which is usually a small

[3] Here, MLP refers to a MLP-style sub-layer that captures spatial representations [32,33,15], instead of pure 1×1 convolution.

integer number, e.g., 4. The general operation block is therefore modeled as

$$\mathtt{y} = \mathtt{x} + \mathtt{Operation}(\mathtt{x}), \tag{1}$$

where Operation can be convolution, MLP, or transformer, and $\mathtt{x}, \mathtt{y}$ represent input and output features, respectively. For convolution, we place the convolution operation inside the bottleneck [24], which can be expressed as

$$\mathtt{Operation}(\mathtt{x}) = \mathtt{Proj}_{\mathtt{ec}\to\mathtt{c}}(\mathtt{Conv}(\mathtt{Proj}_{\mathtt{c}\to\mathtt{ec}}(\mathtt{x}))). \tag{2}$$

The Conv operation can be either regular convolution or depth-wise convolution (DWConv) [3], and the Proj represents a linear projection. For self-attention in transformer and token-mixing in MLP, the computation cost on the large bottleneck feature map is quite huge. Following previous works [8,32], we separate them from the bottleneck for computation efficiency, and the Proj is implemented inside the FFN [37] sub-layer. Each transformer block has a query-key-value self-attention sub-layer and an FFN sub-layer, and the token-mixing in the MLP block is implemented by transpose-FFN-transpose as that in [32],

$$\mathtt{y} = \mathtt{y}' + \mathtt{FFN}(\mathtt{y}'), \tag{3}$$

$$\mathtt{y}' = \mathtt{x} + \mathtt{SA}(\mathtt{x}) \text{ or } \mathtt{x} + \mathtt{MLP}(\mathtt{x}), \tag{4}$$

$$\mathtt{FFN}(\mathtt{y}') = \mathtt{Proj}_{\mathtt{ec}\to\mathtt{c}}(\mathtt{Proj}_{\mathtt{c}\to\mathtt{ec}}(\mathtt{y}')), \tag{5}$$

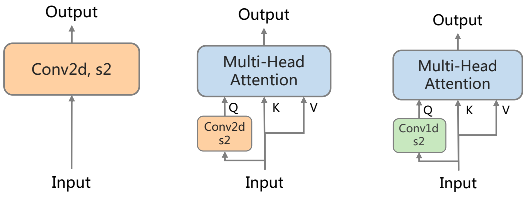
where SA can be either vanilla self-attention or local self-attention LSA, and MLP refers to the token-mixing operation.

There are two main advantages of representing the different types of operators in a unified search space: (1) We can characterize each operator with the same set of configuration parameters (i.e., *OP type*, *expansion*, *channels*, etc). As a result, the overall search space is greatly reduced, and the total search cost becomes affordable. (2) With the unified form, the comparison between operators is fairer, which is important for NAS [29] to identify the optimal hybrid architecture.

3.3 Context-Aware Downsampling Modules

As discussed in Section 3.1, the downsampling module (DSM) plays an important role in visual tasks. In addition to hand-crafted DSM (i.e., max-pooling or avg-pooling), a few works [23,10,40] tried to preserve more information via downsampling with the learnable or dynamic kernel. Most of the approaches utilized downsampling based on local context, which suits conventional ConvNets well. However, in our unified search space, operators with different receptive fields can be assembled unrestrictedly to form a hybrid architecture, where the local context might be destroyed and therefore the previous downsampling operations might not be suitable.

In this paper, we propose context-aware DSM, which is instanced with Local-DSM (L-DSM), Local-Global-DSM (LG-DSM), and Global-DSM (G-DSM). The main difference between those DSMs is the considered context when performing



(a) L-DSM (b) LG-DSM (c) G-DSM

Fig. 3: Structures of the context-aware downsampling modules. The three DSMs are described in Section 3.3. Shortcuts are omitted for better visualization.

downsampling. For L-DSM, only local context is involved, which fits ConvNets well as shown in previous works [41,21]. For G-DSM, only global context is used for downsampling, which may fit other operators, e.g., transformers. The LG-DSM combines the characteristics of L-DSM and G-DSM. It uses both local and global context for downsampling. Our intuition is that one of the largest dissimilarities of different operators is the receptive field. Transformer and MLP naturally have global receptive filed, while convolution has local receptive field, e.g., 3 × 3. When combining those operators, there is no single optimal DSM that satisfies all scenarios.

The proposed DSMs are visualized in Figure 3. To downsample based on global cues, we utilize the self-attention mechanism to capture global context, which is missed by the prior art. For G-DSM, we use `Conv1D` with stride 2 to downsample the query and use the downsampled query features to aggregate key features with downsampled output resolution. Note that, there is no local context preserved after downsampling of G-DSM. For LG-DSM, we first reshape the flattened token sequences back to the spatial grid and apply `Conv2D` with stride 2 to downsample the query, and then flatten the query back to calculate the attention weights.

Compared with previous works, which mainly try to improve ConvNets, our proposed DSMs are not designed for a specific architecture. Our motivation is that different DSMs might be suitable for different operators. For example, the optimal DSM might be L-DSM for ConvNets, but G-DSM for transformers. As thousands of operator combinations would be trained in our NAS process, it is unfeasible to decide which DSM to use by hand. To obtain the optimal architecture, we jointly search DSMs with other operators. In our searched optimal architecture, L-DSM is indeed used between operators with the local receptive field while LG-DSM is favored by operators with a global receptive field. The results validate the effectiveness of our proposed context-aware DSMs.

3.4 UniNet Architecture

As shown in recent studies, combining different operators [42,43] can bring performance improvements. Most previous approaches only repeatedly stack the same operator in the whole architecture and search only different channels in different stages. These approaches do not allow large architecture diversity in each block, which we show is crucial for achieving high accuracy for hybrid architectures.

On the contrary, in our UniNet, the operators are not fixed but searched from the unified search space. We construct our UniNet architecture in a multi-stage fashion, which can be easily transferred to downstream tasks. Between two successive stages, one of our proposed DSMs is inserted to reduce the spatial dimension. We jointly search the GOP and DSM for all stages. The GOP could be different for different stages but repeated multiple times in one stage, which can greatly reduce the search space size as pointed out before [29]. The overall architecture and unified search space are illustrated in Figure 2.

Thanks to our unified form of GOPs, the network size of each stage can be configured with the repeat number `r`, channel size `c`, and expansion ratio `e`. To obtain better computation-accuracy trade-off, we jointly search the network size with the GOP and DSM. For GOP, we search for convolution, transformer, MLP, and their promising variants, i.e., {`SA`, `LSA`, `Conv`, `DWConv`, `MLP`}, as defined in Section 3.2; for `e`, we search from {2, 3, 4, 5, 6}. The kernel size for convolution operation is fixed to 3×3. The head dimension in self-attention is fixed to 32. We start the architecture search with an initial architecture, whose network size is determined based on a reference architecture, e.g., EfficientNetV2 [31]. The initial channels and repeats are set according to the reference architecture. For `c` and `r`, we search from the sets {0.5, 0.75, 1.0, 1.25, 1.5} and {-2, -1, 0, 1, 2}, respectively. Channels are set to be divisible by 32 for self-attention. Suppose we partition the network into K stages, and each stage has a sub-search space of size S. Then the total search space is S^K. In our implementation, K is set to 5 and S equals 1,875. As a result, our search space size is about 2×10^{16} and covers a large set of operators with quite different characteristics.

3.5 Search Algorithm

We use Reinforcement Learning (RL)-based search algorithm to search for high-performance hybrid architecture in our unified search space by jointly optimizing the model accuracy and FLOPs. Concretely, we follow previous work [20,29] and map an architecture in the unified search space to a list of tokens, which are determined by a sequence of actions generated by a Recurrent Neural Network (RNN). The RNN is optimized with the PPO algorithm [25] by maximizing the expected reward. In our implementation, we simultaneously optimize accuracy and the theoretical computation cost (FLOPs). To handle the multi-objective optimization problem, we use a weighted product customized as [29] to approximate Pareto optimal. For a sampled architecture m, the reward is formulated as $r(m) = a(m) \times (\frac{t}{f(m)})^{\alpha}$, where function $a(m)$ and $f(m)$ return the accuracy and the FLOPs of m, t is the target FLOPs, and α is a weight factor that balances the accuracy and computation cost. We include more details of the RL algorithm in the supplementary materials.

During the search process, thousands of combinations of GOPs and DSMs are trained on a proxy task with the same setting, which gives us a fair comparison between those combinations. When the search is over, the top-5 architectures with the highest reward are trained with full epochs, and the top-performing one is kept for model scaling and transferring to other downstream tasks.

4 Experimental Setup and Implementation

To find the optimal architecture in our search space, we directly search on the large-scale dataset, ImageNet-1K. We reserve 50k images from the training set as a validation set. We employ a proxy task setting in the search phase. For each sampled architecture, we train it for 5 epochs and calculate the reward of the architecture with its FLOPs and the accuracy on the validation set. We set the target FLOPs t and weight factor α in the reward function to 550M and 0.07 respectively [30]. During the search process, totally 2K models are trained on the proxy task. After that, we fully train the top-5 architectures on ImageNet-1K and preserve the top-performing one for model scaling and transferring to other downstream tasks.

For full training on the ImageNet-1K dataset, we follow the popular training recipe in DeiT [34]. We employ AdamW optimizer [17] with an initial learning rate of 0.001 and weight decay of 0.05 to train UniNet. The total batch size is set to 1024. We totally train for 300 epochs with a cosine learning rate decay and 5 epochs of linear warm-up. We follow the augmentation strategy in DeiT [34] and apply small augmentation for small models and heavy augmentation for large models as introduced in [35,27]. For training efficiency, UniNet-B5 and UniNet-B6 are trained with 224×224 input size and then finetuned on the large resolution. We also pre-train UniNet on a larger ImageNet-21K dataset, which contains 14.2 million images and 21K classes, to further test UniNet. We pretrain for 90 epochs with AdamW optimizer. We then finetune on ImageNet-1K for 30 epochs and compare the top-1 accuracy on ImageNet-1K with other approaches. We list the details of training and finetuning hyper-parameters in the supplementary materials.

Besides, we also transfer UniNet to downstream tasks, e.g., object detection and instance segmentation on COCO and semantic segmentation on ADE20K. For COCO training, we use the various detection frameworks and train UniNet with the widely-used 1x (12 epochs) and 3x (36 epochs) schedules. For ADE20K training, we use the UperNet framework and train with the same setting as [21]. The training details are listed in the supplementary materials.

5 Main Results

In this section, we firstly present our searched UniNet architecture. We then show the performance of the scaled UniNets on classification, object detection, and semantic segmentation.

Table 2: UniNet-B0 architecture. GOP and DSM represent General Operators and downsampling module respectively. DWConv and SA are described in Section 3.2.

Stage	Operator		Network Size			FLOPs(M)
	GOP	DSM	e	c	r	
0	DWConv	L-DSM	4	48	2	68
1	DWConv	L-DSM	6	80	4	135
2	DWConv	L-DSM	3	128	4	42
3	SA	LG-DSM	2	128	4	63
4	SA	LG-DSM	5	256	8	187

Table 3: Performance of Top-5 models after fully training. D and A are short for DWConv and SA respectively.

Rank	Configuration	Top-1 Acc.
0	**DDDAA**	**79.1**
1	DDDAA	78.7
2	DDDAD	77.9
3	DDDAA	78.6
4	DDDAA	78.4

5.1 UniNet Model Family

Table 2 shows our searched UniNet-B0 architecture. Our searched architecture has the following characteristics: (1) Placing convolution in the shallow layers and transformers with SA in the deep layers. While the previous work [8] shows that the early-stage transformer blocks learn to gather local representations, our searched architecture directly applies convolution at early stages, which is more efficient. We further compare the top-5 searched models in Table 3, and find the conclusion is close to consistent. The exception is the 3rd model, which uses DWConv at the last stage, but with inferior performance. (2) Allocating a similar amount of computations for both convolutions and transformers. Shown in Table 2, the DWConv stages consume 245M FLOPs, and SA stages consume 250M FLOPs. While the operator combination has been studied in prior arts, the computation allocating for different operators is neglected. Our work shed some light on this question by jointly searching the network size in our unified search space. (3) Inserting L-DSM to downsample for convolutions and LG-DSM for transformers. Our search results show that the widely-used downsampling module is sub-optimal for hybrid architectures. We also notice that the MLP operator has not been chosen in the searched UniNet. We empirically find that the MLP-style operation breaks the spatial structure which is important for visual tasks [16], leading to inferior performance when combined with other operators. We add the visualization in the supplementary materials.

To go even further, we build a family of high-performance UniNet models by scaling up the searched UniNet-B0. We utilize the compound scaling [30] to scale depth, width, and resolution simultaneously. Note that the resolution is scaled with a smaller coefficient compared to EfficientNet [30] for training and memory efficiency. We list the details of UniNet-B1 to UniNet-B6 in the supplementary materials. While most previous transformer-based architectures outperform convolution-based architectures in large model sizes but underperform in small model sizes, UniNet achieves consistently better accuracy and efficiency across B0 to B6.

Table 4: UniNet performance on ImageNet. All UniNet models are trained on the ImageNet-1K dataset with 1.28M images. C, T, and H denote convolution, transformer, and hybrid architecture respectively.

Model	Family	Input Size	#FLOPs (G)	#Params (M)	Top-1 Acc.
EffNet-B0 [30]	C	224	0.39	5.3	77.1
EffNetV2-B0 [31]	C	240	0.7	7.4	78.7
DeiT-Tiny [34]	T	224	1.3	5.7	72.2
PVT-Tiny [41]	T	224	1.9	13.2	75.1
ConViT-Ti+ [6]	H	224	2	10	76.7
UniNet-B0	H	160	0.56	11.5	79.1
EffNet-B2 [30]	C	260	1	9.2	80.1
EffNetV2-B1 [31]	C	260	1.2	8.1	79.8
RegNetY-4G [22]	C	224	4	20.6	81.9
DeiT-Small [34]	T	224	4.3	22	79.8
PVT-Small [41]	T	224	3.8	24.5	79.8
UniNet-B1	H	224	1.1	11.5	80.8
EffNet-B3 [30]	C	300	1.8	12	81.6
EffNetV2-B3 [31]	C	300	3	14	82.1
Swin-T [21]	T	224	4.5	29	81.3
CoAtNet-0 [5]	H	224	4.2	25	81.6
UniNet-B2	H	256	2.2	16.2	82.5
EffNet-B4 [30]	C	380	4.2	19	82.9
NFNet-F0 [1]	C	256	12.4	71.5	83.6
Swin-B [21]	T	224	15.4	88	83.5
ConViT-B+ [6]	H	224	30	152	82.5
CoAtNet-1 [5]	H	224	8.4	42	83.3
CvT-21 [42]	H	384	24.9	32	83.3
UniNet-B3	H	288	4.3	24	83.5
EffNet-B7 [30]	C	600	37	66	84.3
EffNetV2-M [31]	C	480	24	54	85.1
NFNet-F2 [1]	C	352	62.6	193.8	85.1
BoTNet-T7 [26]	T	384	45.8	75.1	84.7
CoAtNet-1 [5]	H	384	27.4	42	85.1
UniNet-B4	H	320	9.4	43.8	84.4
UniNet-B5	H	384	20.4	72.9	84.9
UniNet-B6	H	448	51	117	85.6

Table 5: Performance on ImageNet with ImageNet-21K pre-train. All models are pre-trained on ImageNet-21K and finetuned on ImageNet-1K.

Model	Family	Input Size	#FLOPs (G)	#Params (M)	Top-1 Acc.
EffNetV2-M [31]	C	480	24	55	86.1
ViT-L/16 [8]	T	384	190.7	304	85.3
HaloNet-H4 [36]	T	384	-	85	85.6
Swin-B [21]	T	384	47.1	88	86.4
CvT-21 [42]	H	384	25	32	84.9
UniNet-B5	H	384	20.4	72.9	87
EffNetV2-L [31]	C	480	53	121	86.8
EffNetV2-XL [31]	C	512	94	208	87.3
Swin-L [21]	T	384	103.9	197	87.3
CoAtNet-2 [5]	H	384	49.8	75	87.1
CoAtNet-2 [5]	H	512	96.7	75	87.3
UniNet-B6	H	448	51	117	87.4

Table 6: Comparison with previous efficient architectures. UniNet is trained with knowledge distillation for a more fair comparison.

Model	Family	#FLOPs (M)	Top-1 Acc.
AttentiveNAS [39]	C	491	80.1
AlphaNet [38]	C	491	80.3
FBNetv3 [4]	C	557	80.5
OFA [2]	C	595	80.0
LeViT [12]	H	658	80.0
UniNet-B0	H	555	80.8

5.2 ImageNet Classification Performance

ImageNet-1K. Table 4 presents the performance comparison of our searched UniNet with previous proposed architectures. Our searched UniNet has better accuracy and computation efficiency than previous ConvNets, Transformers, or hybrid architectures.

As shown in Table 4, under mobile setting, our UniNet-B0 achieves 79.1% top-1 accuracy with 555M FLOPs, outperforming EfficientNetV2-B0 [31] with less FLOPs. In the middle FLOPs setting, our UniNet-B3 achieves 83.5% top-1 accuracy with 4.3G FLOPs, which outperforms the pure convolution-based EfficientNet-B4, pure transformer-based Swin-B, and hybrid architecture CvT-21. For larger models, our UniNet-B5 achieves 84.9% with 20G FLOPs, outperforming EfficientNet-B7 and BoTNet-T7 with 44% and 55% fewer FLOPs, respectively. Figure 1 (a, b) further visualizes the comparison of UniNet with other architectures in terms of accuracy and FLOPs.

Table 7: Object detection, instance segmentation, and semantic segmentation performance on the COCO val2017 and ADE20K val set. All UniNet models are pre-trained on the ImageNet-1K dataset.

Backbone	#Params (M)	#FLOPs (G)	Mask R-CNN 1x		Mask R-CNN 3x		UperNet
	Det/Seg	Det/Seg	AP@box	AP@mask	AP@box	AP@mask	mIoU (%)
ResNet18 [14]	31/ -	207/885	34.0	31.2	36.9	33.6	-
ResNet50 [14]	44/ -	260/951	38.0	34.4	41.0	37.1	-
PVT-Tiny [41]	33/ -	208/945	36.7	35.1	39.8	37.4	-
UniNet-B1	28/38	211/877	40.5	37.5	44.4	40.1	42.7
ResNet101 [14]	63/86	336/1029	40.4	36.4	42.8	38.5	44.9
PVT-Small [41]	44/ -	245/1039	40.4	37.8	43.0	39.9	-
Swin-T [21]	48/60	267/945	43.7	39.8	46.0	41.6	44.5
UniNet-B3	42/51	270/940	45.2	41.1	47.9	42.9	48.5

Table 8: Performance on the COCO val2017 with various detection frameworks. The AP@box is reported.

Framework	Cascade-Mask-R-CNN	ATSS	Sparse-R-CNN	Mask-R-CNN
ResNet50 [14]	46.3	43.5	44.5	41.0
Swin-T [21]	50.5	47.2	47.9	46.0
UniNet-B3	51.3	49.8	48.9	47.9

We further compare UniNet-B0 to previous searched efficient architectures in Table 6. Note that for a more fair comparison, we train UniNet-B0 with knowledge distillation. The details of distillation are listed in the supplementary materials. Shown in Table 6, UniNet-B0 achieves 80.8% accuracy with 555M FLOPs, outperforming other efficient convolution-based or hybrid architectures.

ImageNet-21K. Table 5 presents the performance comparison of UniNet and other architectures with ImageNet-21K pretrain. Notably, UniNet-B5 obtains 87% top-1 accuracy, which outperforms Swin-L with 4× less computation. UniNet-B6 achieves 87.4% top-1 accuracy, which outperforms CoAtNet-2 [5] with 47% less computation. We further visualize the comparison in Figure 1 (c).

5.3 Object Detection and Semantic Segmentation Performance

For object detection and semantic segmentation, we pick UniNet-B1 and UniNet-B3 and use them as the backbone networks for detection and segmentation frameworks. We compare our UniNet with other convolution or transformer-based architectures. For COCO object detection, we use various detection frameworks and compare the performance under 1× and 3× schedules. For ADE20K semantic segmentation we use the UperNet framework and report mIoU (%) for different architectures under the same training setting.

As shown in Table 7, our searched UniNet consistently outperforms convolution-based ResNet [14] and transformer-based PVT [41] or Swin-Transformer [21]. UniNet-B1 achieves 40.5 AP@box, which is 3.8% better than PVT-Tiny but with 15% fewer parameters. UniNet-B3 achieves 45.2 AP@box with 1× schedule and 47.9 AP@box with 3× schedule, which is 1.5% and 1.9% better than Swin-T, respectively. We further test various detection framework and show the results in Table 8, and find that UniNet achieves consistently better performance among others. For ADE20K semantic segmentation, we achieve 48.5% mIoU with 51M parameters. Compared with transformer-based Swin-T, our UniNet outperforms 4.0% mIoU with a similar parameter size. Besides, compared with convolution-based ResNet101, we achieve 3.6% higher mIoU with 41% fewer parameters. All the results show the effectiveness of our searched UniNet.

6 Ablative Studies and Analysis

In this section, we study the impact of joint search of General Operators and discuss the importance of context-aware downsampling modules (DSMs).

6.1 Single Operator vs. General Operators

Previous works [29,30] mostly focus on the network size search, which uses a single operator, convolution, as the main feature extractor. In comparison, we jointly search the combination of different General Operators (GOPs), i.e., convolution, transformer, MLP, and their promising variants. To verify the importance of GOPs, we keep only one type of operator in the search space and re-run the search experiments under the same settings. After the search, we fully train the top-5 architectures with the highest reward on ImageNet-1K and report the best performance.

Table 9: Performance on ImageNet with different search settings. One type of operator is kept for comparison with the hybrid UniNet.

Model	#FLOPs (G)	#Params (M)	Top-1 Acc.
UniNet-B0	0.56	11.5	**79.1**
Convolution-Only	0.59	11.0	77.7
Transformer-Only	1.2	11.2	78.2
MLP-Only	0.95	11.4	76.8

As shown in Table 9, our searched hybrid architecture consistently achieves better accuracy compared to single-operator-based architectures. The result verifies the effectiveness of our unified architecture search of GOPs, which can take advantage of the characteristics of different operators.

6.2 Fixed vs. Context-Aware downsampling

When combining different operators into a unified network, the traditional downsampling module, such as strided-conv or pooling, could be sub-optimal. To verify the effectiveness of our proposed context-aware DSMs, we replace the DSMs of our search UniNet with one fixed DSM and compare their performance under the same training setting.

As shown in Table 10, our searched UniNet consistently outperforms its variants that use a single-fixed DSM in all stages. Although we see that using G-DSM

Table 10: Performance on ImageNet of UniNet with different DSMs. Note that the traditional strided-conv downsampling module is shown in row 2.

Model	#FLOPs (G)	#Params (M)	Top-1 Acc.
UniNet	0.56	11.5	**79.1**
w/ L-DSM	0.54	11.3	78.5
w/ G-DSM	0.77	12.7	76.8
w/ LG-DSM	0.72	14.1	78.9

Table 11: Performance comparison on ImageNet of different backbones when equipped with our proposed DSMs.

Model	#FLOPs (G)	#Params (M)	Top-1 Acc.
PVT-Tiny [41]	1.9	13.2	75.1
w/ LG-DSM	3.1	17.3	78.6
w/ L→LG-DSM	2.0	14.3	77.5
Swin-T [21]	4.5	29.0	81.2
w/ LG-DSM	6.4	33.4	81.9
w/ L→LG-DSM	4.7	30.0	81.6

or LG-DSM in all stages brings more computation and parameters, the performance does not become better. The result emphasizes the importance of our joint search of GOPs and DSMs.

Besides, we transfer our proposed DSMs to other popular transformer-based architectures, Swin-Transformer [21] and PVT [41]. Both Swin and PVT have 4 stages. We compare 2 settings: 1) using LG-DSM for 4 stages, as both PVT and Swin are pure transformer architectures 2) using L-DSM for the first two stages while LG-DSM for the latter two stages, which requires less computation. As shown in Table 11, our proposed LG-DSM improves PVT-Tiny and Swin-T for 3.5% and 0.7%, respectively. Using L-DSM in the first two stages has a similar computation compared with the baseline, which improves PVT-Tiny and Swin-T for 2.4% and 0.4%, respectively. To note that, PVT uses a strided-conv for downsampling. As discussed in Section 3.3, it is harmful to the main operator in PVT, which has a global receptive field. On the contrary, our proposed DSMs are able to downsample based on both local and global context, and can greatly improve the performance.

7 Conclusion

In this paper, we propose a novel unified architecture search approach to jointly search the combination of convolution, transformer, and MLP. We empirically identify that the widely-used downsampling modules become the performance bottlenecks when the operators are combined. To further improve the performance, we propose context-aware downsampling modules and jointly search them with all operators. We scale the search baseline network up and obtain a family of models, named UniNet, which achieve much better accuracy and efficiency than previous ConvNets and Transformers.

Acknowledgement Hongsheng Li is also a Principal Investigator of Centre for Perceptual and Interactive Intelligence Limited (CPII). This work is supported in part by CPII, in part by the General Research Fund through the Research Grants Council of Hong Kong under Grants (Nos. 14204021, 14207319), in part by CUHK Strategic Fund.

References

1. Brock, A., De, S., Smith, S.L., Simonyan, K.: High-performance large-scale image recognition without normalization. arXiv preprint arXiv:2102.06171 (2021)
2. Cai, H., Gan, C., Wang, T., Zhang, Z., Han, S.: Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791 (2019)
3. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1251–1258 (2017)
4. Dai, X., Wan, A., Zhang, P., Wu, B., He, Z., Wei, Z., Chen, K., Tian, Y., Yu, M., Vajda, P., et al.: Fbnetv3: Joint architecture-recipe search using neural acquisition function (2020)
5. Dai, Z., Liu, H., Le, Q.V., Tan, M.: Coatnet: Marrying convolution and attention for all data sizes. arXiv preprint arXiv:2106.04803 (2021)
6. d'Ascoli, S., Touvron, H., Leavitt, M., Morcos, A., Biroli, G., Sagun, L.: Convit: Improving vision transformers with soft convolutional inductive biases. arXiv preprint arXiv:2103.10697 (2021)
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
8. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
9. Gao, P., Lu, J., Li, H., Mottaghi, R., Kembhavi, A.: Container: Context aggregation network. arXiv preprint arXiv:2106.01401 (2021)
10. Gao, Z., Wang, L., Wu, G.: Lip: Local importance-based pooling. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3355–3364 (2019)
11. Gong, C., Wang, D., Li, M., Chen, X., Yan, Z., Tian, Y., Chandra, V., et al.: Nasvit: Neural architecture search for efficient vision transformers with gradient conflict aware supernet training. In: International Conference on Learning Representations (2021)
12. Graham, B., El-Nouby, A., Touvron, H., Stock, P., Joulin, A., Jégou, H., Douze, M.: Levit: a vision transformer in convnet's clothing for faster inference. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12259–12269 (2021)
13. Guo, J., Han, K., Wu, H., Xu, C., Tang, Y., Xu, C., Wang, Y.: Cmt: Convolutional neural networks meet vision transformers. arXiv preprint arXiv:2107.06263 (2021)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
15. Hou, Q., Jiang, Z., Yuan, L., Cheng, M.M., Yan, S., Feng, J.: Vision permutator: A permutable mlp-like architecture for visual recognition. arXiv preprint arXiv:2106.12368 (2021)
16. Islam, M.A., Jia, S., Bruce, N.D.: How much position information do convolutional neural networks encode? arXiv preprint arXiv:2001.08248 (2020)
17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

18. Li, J., Hassani, A., Walton, S., Shi, H.: Convmlp: Hierarchical convolutional mlps for vision. arXiv preprint arXiv:2109.04454 (2021)
19. Liu, H., Dai, Z., So, D., Le, Q.: Pay attention to mlps. Advances in Neural Information Processing Systems **34** (2021)
20. Liu, J., Zhang, M., Sun, Y., Liu, B., Song, G., Liu, Y., Li, H.: Fnas: Uncertainty-aware fast neural architecture search. arXiv preprint arXiv:2105.11694 (2021)
21. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. arXiv preprint arXiv:2103.14030 (2021)
22. Radosavovic, I., Kosaraju, R.P., Girshick, R., He, K., Dollár, P.: Designing network design spaces. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10428–10436 (2020)
23. Saeedan, F., Weber, N., Goesele, M., Roth, S.: Detail-preserving pooling in deep networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9108–9116 (2018)
24. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4510–4520 (2018)
25. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
26. Srinivas, A., Lin, T.Y., Parmar, N., Shlens, J., Abbeel, P., Vaswani, A.: Bottleneck transformers for visual recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16519–16529 (2021)
27. Steiner, A., Kolesnikov, A., Zhai, X., Wightman, R., Uszkoreit, J., Beyer, L.: How to train your vit? data, augmentation, and regularization in vision transformers. arXiv preprint arXiv:2106.10270 (2021)
28. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–9 (2015)
29. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: Mnasnet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2820–2828 (2019)
30. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning. pp. 6105–6114. PMLR (2019)
31. Tan, M., Le, Q.V.: Efficientnetv2: Smaller models and faster training. arXiv preprint arXiv:2104.00298 (2021)
32. Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Keysers, D., Uszkoreit, J., Lucic, M., et al.: Mlp-mixer: An all-mlp architecture for vision. arXiv preprint arXiv:2105.01601 (2021)
33. Touvron, H., Bojanowski, P., Caron, M., Cord, M., El-Nouby, A., Grave, E., Joulin, A., Synnaeve, G., Verbeek, J., Jégou, H.: Resmlp: Feedforward networks for image classification with data-efficient training. arXiv preprint arXiv:2105.03404 (2021)
34. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: International Conference on Machine Learning. pp. 10347–10357. PMLR (2021)
35. Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., Jégou, H.: Going deeper with image transformers. arXiv preprint arXiv:2103.17239 (2021)

36. Vaswani, A., Ramachandran, P., Srinivas, A., Parmar, N., Hechtman, B., Shlens, J.: Scaling local self-attention for parameter efficient visual backbones. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12894–12904 (2021)
37. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
38. Wang, D., Gong, C., Li, M., Liu, Q., Chandra, V.: Alphanet: Improved training of supernets with alpha-divergence. In: International Conference on Machine Learning. pp. 10760–10771. PMLR (2021)
39. Wang, D., Li, M., Gong, C., Chandra, V.: Attentivenas: Improving neural architecture search via attentive sampling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6418–6427 (2021)
40. Wang, J., Chen, K., Xu, R., Liu, Z., Loy, C.C., Lin, D.: Carafe++: Unified content-aware reassembly of features. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021)
41. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. arXiv preprint arXiv:2102.12122 (2021)
42. Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., Zhang, L.: Cvt: Introducing convolutions to vision transformers. arXiv preprint arXiv:2103.15808 (2021)
43. Yuan, K., Guo, S., Liu, Z., Zhou, A., Yu, F., Wu, W.: Incorporating convolution designs into visual transformers. arXiv preprint arXiv:2103.11816 (2021)
44. Zhang, R.: Making convolutional networks shift-invariant again. In: ICML (2019)