TVConv: Efficient Translation Variant Convolution for Layout-aware Visual Processing

Jierun Chen[1], Tianlang He[1], Weipeng Zhuo[1], Li Ma[1], Sangtae Ha[2], S.-H. Gary Chan[1]
[1]The Hong Kong University of Science and Technology, [2]University of Colorado at Boulder
{jcheneh, theaf, wzhuo, lmaag, gchan}@cse.ust.hk, sangtae.ha@colorado.edu

Abstract

As convolution has empowered many smart applications, dynamic convolution further equips it with the ability to adapt to diverse inputs. However, the static and dynamic convolutions are either layout-agnostic or computation-heavy, making it inappropriate for layout-specific applications, e.g., *face recognition and medical image segmentation. We observe that these applications naturally exhibit the characteristics of large intra-image (spatial) variance and small cross-image variance. This observation motivates our efficient translation variant convolution (TVConv) for layout-aware visual processing. Technically, TVConv is composed of affinity maps and a weight-generating block. While affinity maps depict pixel-paired relationships gracefully, the weight-generating block can be explicitly over-parameterized for better training while maintaining efficient inference. Although conceptually simple, TVConv significantly improves the efficiency of the convolution and can be readily plugged into various network architectures. Extensive experiments on face recognition show that TVConv reduces the computational cost by up to* $3.1\times$ *and improves the corresponding throughput by* $2.3\times$ *while maintaining a high accuracy compared to the depthwise convolution. Moreover, for the same computation cost, we boost the mean accuracy by up to 4.21%. We also conduct experiments on the optic disc/cup segmentation task and obtain better generalization performance, which helps mitigate the critical data scarcity issue. Code is available at* https://github.com/JierunChen/TVConv.

1. Introduction

With the breakthrough of deep neural networks, we are witnessing a flourishing growth in AI-powered applications and services. While a performance gain usually comes with an overhead of model size and computation, more interest has been devoted to the lightweight and computation-efficient network design, which can unleash the potential for on-device inference for user experience and privacy.

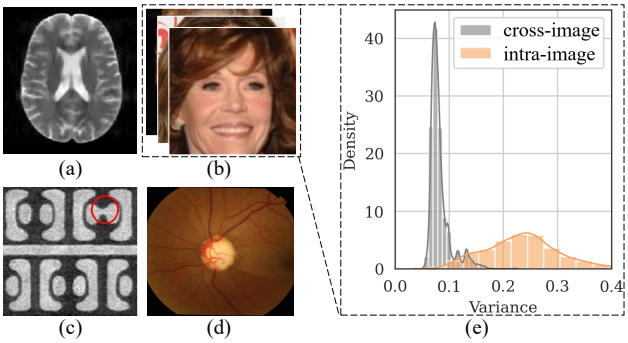


Figure 1. Various applications with specific layout: (a) brain MRI analysis, (b) face recognition, (c) industrial product defects detection and (d) optic disc/cup segmentation. Each application exhibits large intra-image variance and small cross-image variance, as shown in subfigure (e), where the statistic is calculated from the intermediate VGG [35] feature maps by feeding the LFW face verification dataset [15].

Despite various neural network architectures [11, 25, 32, 39] being proposed for efficiency, their fundamental operators remain more or less the same, *e.g.*, vanilla convolution (conv) and depthwise conv. These operators share one key characteristic, *translation equivariance* [51], *i.e.*, filters are shared spatially in a sliding window manner. Though it saves parameters for a lightweight model, it deprives the model of being adaptive to different positions in an image. Therefore, it has to exhaustively learn many filters for feature matching [49], which results in a massive waste of computation for many tasks with a specific layout.

For layout-specific tasks, as in Fig. 1, the inputs demonstrate a regional statistic with large intra-image (spatial) variance and small cross-image variance. For example, when we use face ID to securely and conveniently unlock our mobile phones, our hair generally appears on the upper region, vertically followed by our forehead, eyes, nose, mouth, jaw, etc. Similar tasks include, but are not limited to, talking head generation, industrial product defects detection, and medical image processing.

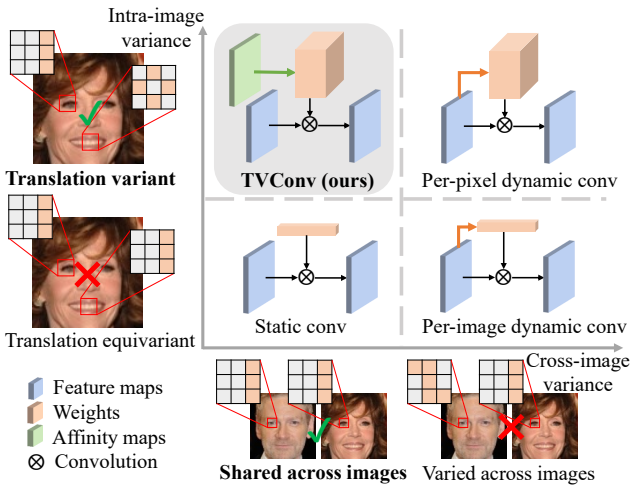


Figure 2. Comparison among conv variants. Different from the static conv, per-image dynamic conv and per-pixel dynamic conv, our TVConv is translation variant and shared across images, superbly fitting into the layout-specific applications.

To account for the large intra-image variance, many works propose per-pixel dynamic conv [3, 20, 22, 37, 45, 49, 59], which extend the per-image dynamic conv [6, 19, 24, 53, 57] to a spatial domain. They apply the local feature dependant filters by assembling multiple templates for each position. However, it can easily suffer from prohibitively large memory footprints and substantial computation overhead. Besides, it omits the nature of small cross-image variance for layout-specific tasks, causing redundant computation of pixel-wise filters for a series of the inputs. Therefore, the challenge remains to exploit an efficient operator to better serve such applications.

In this paper, we propose an efficient fundamental operator called Translation Variant Convolution (TVConv) for layout-aware visual processing. In contrast to existing conv variants, TVConv elegantly supports the applications with a specific layout by its *translation variance* and the property of being *shared across images*, as in Fig. 2. Technically, we first formulate compact and learnable affinity maps to distinguish various local features. The affinity maps implicitly capture the semantic relationships among different regions, similar to the attention mechanism [42, 50], but without bothering to derive a heavy affinity matrix. We then feed the affinity maps into a weight-generating block to yield the weights, which are then applied as filters on the input. Unlike the dynamic conv, whose weight-generating block is constrained by the tradeoff between the adaptivity and the computation overhead, TVConv can be freely over-parameterized to strengthen the spatial adaptivity without slowing down the inference speed. The affinity maps are fixed once they are trained. Thus the weight generating process can be performed only once during initialization. Then it produces and caches the weights in memory, allowing them to be fetched efficiently for subsequent inferences. Through extensive experiments on face recognition, TVConv is shown to strike a better tradeoff between accuracy and computation complexity. Simply replacing the prevalent depthwise conv in various architectures (*e.g.* MobileNetV2, ShuffleNetV2), TVConv reduces the theoretical complexity by up to $\mathbf{3.1}\times$ and correspondingly accelerates the throughput by $\mathbf{2.3}\times$ while maintaining a high accuracy. On the other hand, TVConv boosts the mean accuracy by up to 4.21% under an extreme low complexity constraint. Moreover, thanks to the layout awareness, TVConv shows a better generalization ability for optic disc/cup segmentation on unseen datasets, which helps to alleviate the data-scarcity dilemma for medical image analysis.

In short, our contributions include: (1) We rethink the existing convolution variants of their inappropriate properties for layout-specific tasks in terms of the intra-image and cross-image variance; (2) By being translation variant and shared across images, an efficient fundamental operator, TVConv, is proposed for layout-aware visual processing; (3) Extensive experiments show that TVConv clearly improves the efficiency for face recognition and generalize better for medical image processing.

2. Related Work

This section briefly reviews related works considering the macro network design to the micro operators, such as the dynamic conv and the attention mechanism.

Efficient network design. Stimulated by the practical needs of edge intelligence, an increasing interest has been put into the field of efficient network design. MobileNets [13, 32] are built principally on the depthwise separable conv [34], which works by decomposing a standard conv into a depthwise and a pointwise conv. ShuffleNets [25, 56] further shuffle the channels to facilitate the information flow. Later on, supported by the neural architecture search [61], MobileNetV3 [12], MnasNet [38] and EfficientNets [39, 40] employ reinforcement learning to search for efficient networks. Despite various architectures being proposed, one of their fundamental operators remains primarily a depthwise conv. Our TVConv inherits its lightweight merits and takes one step further to be translation variant and layout-aware.

Dynamic conv. Instead of blindly stacking more static convs to increase models' capacity, dynamic conv leads a new wave to apply adaptive filters conditional on the input. Recent works [24, 48, 53, 57] attempt to use per-image dependant filters, either by discrete gating or continuously weighted averaging of multiple templates. Another branch of works [3, 22, 45, 59] further extends the adaptiveness to the spatial dimension by using per-pixel

dynamic conv. A pioneering work [19] demonstrates its effectiveness on video and stereo prediction. Deformable conv [7, 60] augments the filter shape with auxiliary offsets. Some works [26, 52] apply the dynamic conv to real-world image restoration. Our method follows the content-aware paradigm but is more efficient for layout-specific tasks. As the tasks exhibit a property of small cross-class variance, TVConv is designed to be shared across images and exempted from the heavy and redundant filter recalculation.

Attention mechanism. Attention mechanism originates from the field of machine translation [42]. It excels at capturing long-range dependency by summarizing the global context and reweighting responses at each location. Its tremendous success in the language domain has motivated researches to explore the applicability to computer vision, including image generation [30, 55], object detection [14, 47] and semantic segmentation [9, 16]. While some works [29, 50] propose attention module as a versatile and orthogonal supplement to existing conv operator, more recent works [8, 23] aggressively adopt purely attention-powered architectures. Although significant improvements have been achieved, the above-mentioned attention mechanism involves a heavy computation of the affinity matrix that scales quadratically with the resolution of the input. In contrast, the affinity maps of our TVConv preserve the appealing ability to attend the whole image and obtain pixel-paired relationships, but more efficiently.

3. Approach

In this section, we start with a conventional conv and then describe how it grows to our TVConv for layout-aware processing. After detailing the implementation, we elaborate its connections with prior operators.

3.1. Preliminary

The depthwise conv has been widely adopted in state-of-the-art light-weight architectures. Given an input tensor $\mathbf{I} \in \mathbb{R}^{c\times h\times w}$, it computes the output $\mathbf{O} \in \mathbb{R}^{c\times h\times w}$ by channel-wisely convolving a local patch $\mathbf{P}_{i,j} \in \mathbb{R}^{c\times k\times k}$ with the spatially shared weight $W \in \mathbb{R}^{c\times k\times k}$, where (i, j) denotes a spatial location and $\mathbf{P}_{i,j}$ is centered at $\mathbf{I}_{i,j} \in \mathbb{R}^c$ with $k \times k$ size. For conciseness, we omit the bias term and formulate the process as follows:

$$\mathbf{O}_{i,j} = W \otimes \mathbf{P}_{i,j}, \tag{1}$$

where $\otimes$ denotes a channel-wise convolution. As the output $\mathbf{O}$ shares the same number of channels c with the input $\mathbf{I}$, a depthwise conv is normally followed by a pointwise (1×1) conv for channel projection and fusion.

3.2. Design of TVConv

The depthwise conv shares the same weight across the image. This translation equivariance makes it layout-agnostic and inefficient for layout-specific tasks. In contrast, one may attempt to revert this property to be translation variant:

$$\mathbf{O}_{i,j} = \mathbf{W}_{i,j} \otimes \mathbf{P}_{i,j}. \tag{2}$$

This, however, results in a bulky weight tensor $\mathbf{W} \in \mathbb{R}^{c\times k\times k\times h\times w}$. The unaffordable number of parameters impedes efficient training and may be prone to overfitting. Thus, we propose to first decompose $\mathbf{W}$ as:

$$\mathbf{W}' = \mathbf{B}'\mathbf{A}', \tag{3}$$

where $\mathbf{W}' \in \mathbb{R}^{(c\times k\times k)\times(h\times w)}$ is the reshaped version of $\mathbf{W}$, $\mathbf{B}' \in \mathbb{R}^{(c\times k\times k)\times c_A}$ is the basis matrix, and $\mathbf{A}' \in \mathbb{R}^{c_A\times(h\times w)}$ is the coefficient matrix. By doing so, the number of parameters is greatly reduced from $(ckkhw)$ to $(ckkc_A + c_Ahw)$. In general, c_A can be set to a small value, *e.g.*, $c_A = 1$, approximately yielding a $(ckkhw)/(ckk + hw) \approx (ckk)$ times reduction of parameters.

To further strengthen the translation variance, we replace the linear multiplication in Eq. (3) by a non-linear function:

$$\mathbf{W} = \mathcal{B}(\mathbf{A}), \tag{4}$$

where $\mathbf{A} \in \mathbb{R}^{c_A\times h\times w}$ denotes our affinity maps and $\mathcal{B}$ is a non-linear function, instantiated as our weight generating block. For the affinity maps $\mathbf{A}$, they gracefully depict the pixel-paired relationships in a compact size. For the weight-generating block $\mathcal{B}$, it perceives the affinity maps and produces the weights for operation.

Thanks to the conciseness of our design, TVConv can be fast prototyped, as illustrated in Fig. 3. There are two

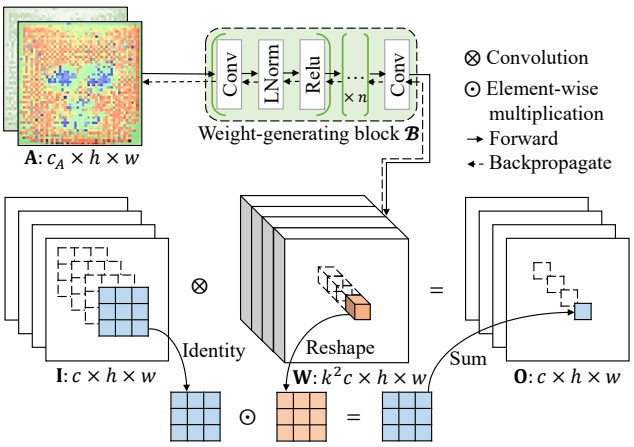


Figure 3. Schematic illustration of our proposed TVConv. The learnable affinity maps $\mathbf{A} \in \mathbb{R}^{c_A\times h\times w}$ are fed into a weight-generating block $\mathcal{B}$ to yield the weight $\mathbf{W} \in \mathbb{R}^{k^2c\times h\times w}$ (reshaped version) for depthwise conv. Each patch within the input feature maps $\mathbf{I} \in \mathbb{R}^{c\times h\times w}$ is then element-wisely multiplied by the corresponding weight patch within $\mathbf{W}$, and followed by a summation to produce each value within the output feature maps $\mathbf{O} \in \mathbb{R}^{c\times h\times w}$.

processes involved: generating and applying the weights. While the latter process simply mirrors the depthwise conv, the weight generation is the focus of our work. The weight generating block $\mathcal{B}$ takes in the affinity maps $\mathbf{A}$ and feeds through a standard conv, a layer normalization, and an activation (*e.g.* Relu). These three layers can be consecutively executed multiple times and finally followed by an output layer. Unlike most other convolutional neural networks which use the batch normalization [17], we adopt the layer normalization [2] within $\mathcal{B}$ because the affinity maps $\mathbf{A}$ are shared for different inputs and the "batch size" for $\mathbf{A}$ can be interpreted as 1. The affinity maps $\mathbf{A}$ can be trained end-to-end by standard back-propagation since all the operations involved are differentiable.

As shown in the work [1], over-parameterization (*e.g.* increase network width) helps the training and achieves better performance. However, it sacrifices the test-time speed. Notably, our TVConv inherits its advantages while avoiding the computational overhead. TVConv allows the weight generating block $\mathcal{B}$ to be over-parameterized up to the memory limit. After training, the affinity maps $\mathbf{A}$ are fixed. Therefore, we can perform the weight generation process only once during initialization and achieve fast inference by readily applying the same weights.

As a versatile replacement of depthwise conv, TVConv can be easily plugged into the bottleneck blocks of various architectures (*e.g.* MobiletNets, ShuffleNets, MnasNet). These networks provide a throttleable hyper-parameter, the width of the network. Thanks to the layout awareness, TVConv can adaptively perceive different regions and allow the width to be greatly narrowed down while maintaining the performance.

3.3. Connection with prior operators

To further distinguish TVConv from prior operators, here we discuss the connections between TVConv with the static/dynamic conv and the self-attention.

Static conv. The diversity within affinity maps $\mathbf{A}$ is the key to our TVConv, which implicitly measures the potential gap between TVConv with the static translation equivariant conv. When the learned affinity maps are filled with a constant value, TVConv degrades to the static depthwise conv, as $\mathbf{W}_{i,j} = W$ for different (i, j). We visualize and discuss the learned affinity maps in Sec. 4.

Dynamic conv. Formally by replacing the affinity maps $\mathbf{A}$ in Eq. (4) with the input $\mathbf{I}$, TVConv would fall into the category of dynamic conv. As a result, the weight generating process has to be repeated for different inputs, and the weight generating block has to be carefully kept compact. Besides, though the weights generated by dynamic conv is translation variant, we clarify that the dynamic conv itself is translation equivariant [49]. The dynamic conv cannot predict the adaptive weights before "seeing" the input. Therefore, dynamic conv is layout-adaptive but not layout-aware.

Self-attention. Our work is also related to the self-attention [42]. Its core idea is to aggregate the global context with adaptive weights. This complicated operator has two important characters: similar inputs in different positions would respond similarly (regardless of the position encoding); response at each position is aware of the global context. Intriguingly, our affinity maps $\mathbf{A}$ resemble these two characters implicitly. Firstly, if two positions within $\mathbf{A}$ appear with similar values, they would produce similar weights for computing. Secondly, once the affinity maps are learned from the data, the pixel-paired relationships are formulated and fixed within the spatial domain, making them mutually context-aware. In short, while preserving the appearing properties of self-attention, our affinity maps are yet much more efficient for layout-specific applications.

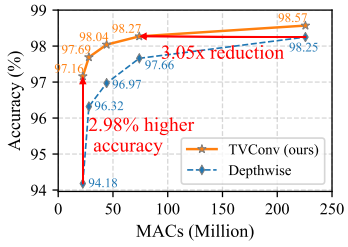
4. Experiments

In this section, we carry out a systematical evaluation of TVConv and report the experimental results. We start with face recognition to validate its efficiency. We then move to the optic disc/cup segmentation task, suggesting a promisingly better generalization by applying the TVConv. We also provide a comprehensive ablation study to consolidate various considerations for implementation.

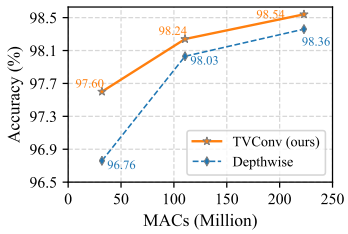
4.1. Face recognition

The focus of this work is to provide an efficient operator for layout-aware processing. Therefore, we replace the depthwise conv in various architectures with TVConv. Instead of building upon some bulky models, we use two popular lightweight architectures, MobileNetV2 [32] and ShuffleNetV2 [25] as baselines (with stride $= 1$ for the first layer, following the work [5]). EfficientNet [39] is not included because it involves a compound scaling factor associated with width, depth, and resolution, while scaling the width factor is more critical for evaluation in this case. For training, we use the publicly available and widely adopted CASIA-WebFace [54] dataset, which consists of 490k images from 10k identities. As it contains many profile view images, we purify it to 329k frontal images by training a simple frontal/profile classifier with the help of the dataset CFP-FP [33]. For validation, as a common practice, we use the face verification datasets LFW [15], CFP-FF [33], AgeDB-30 [27] and CALFW [58]. The inputs are resized to 96x96 with a horizontal flipping augmentation.

All the models are trained with AM-Softmax [43] loss for training stability. The SGD optimizer is used with momentum 0.9 and weight decay 5e-4. The learning rate starts from 0.1 and is divided by 10 at epochs 22, 30, and 35. Total training epochs are 38, with a batch size of 512. We initial-



(a) MobileNetV2 as the architecture



(b) ShuffleNetV2 as the architecture

Figure 4. The accuracy-complexity envelope on the LFW dataset.

Arch× width	MACs (M)	Operator	Accuracy across datasets (%)				
			LFW	CFP-FF	AgeDB-30	CALFW	Mean
MB×0.1	22.47	Depthwise	94.18 ± 0.34	93.01 ± 0.40	79.10 ± 0.44	82.50 ± 0.59	87.20
		TVConv	97.16 ± 0.11	96.49 ± 0.21	84.61 ± 0.20	87.37 ± 0.21	91.41
MB×0.2	28.00	Depthwise	96.32 ± 0.07	95.53 ± 0.26	82.56 ± 0.32	85.62 ± 0.31	90.01
		TVConv	97.69 ± 0.12	97.21 ± 0.26	85.97 ± 0.38	87.96 ± 0.38	92.21
MB×0.3	44.20	Depthwise	96.97 ± 0.27	96.31 ± 0.21	84.72 ± 0.17	86.87 ± 0.50	91.22
		TVConv	98.04 ± 0.16	97.73 ± 0.19	86.99 ± 0.45	88.97 ± 0.58	92.93
MB×0.5	74.03	Depthwise	97.66 ± 0.16	96.98 ± 0.18	85.74 ± 0.46	88.08 ± 0.23	92.11
		TVConv	98.27 ± 0.07	97.96 ± 0.19	87.88 ± 0.12	89.22 ± 0.22	93.33
MB×1.0	225.72	Depthwise	98.25 ± 0.13	97.82 ± 0.11	88.00 ± 0.21	89.41 ± 0.26	93.37
		TVConv	98.57 ± 0.12	98.43 ± 0.09	89.58 ± 0.19	90.29 ± 0.11	94.22
SF×0.5	31.95	Depthwise	96.76 ± 0.08	95.95 ± 0.08	83.87 ± 0.37	86.71 ± 0.44	90.82
		TVConv	97.61 ± 0.22	96.99 ± 0.11	85.86 ± 0.49	88.5 ± 0.35	92.24
SF×1.0	110.53	Depthwise	98.03 ± 0.13	97.43 ± 0.18	86.80 ± 0.43	88.75 ± 0.42	92.75
		TVConv	98.24 ± 0.09	97.83 ± 0.19	87.83 ± 0.78	89.46 ± 0.37	93.34
SF×1.5	222.99	Depthwise	98.36 ± 0.12	97.73 ± 0.15	87.96 ± 0.40	89.43 ± 0.51	93.37
		TVConv	98.54 ± 0.13	98.23 ± 0.08	88.68 ± 0.50	89.91 ± 0.16	93.84

Table 1. TVConv consistently outperforms the depthwise conv across four face verification datasets under varied network width for both MobileNetV2 (MB) and ShuffleNetV2 (SF).

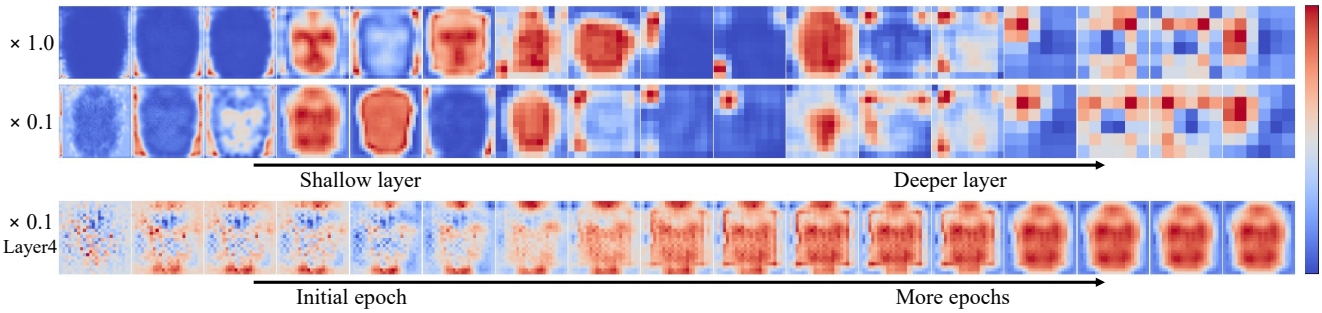
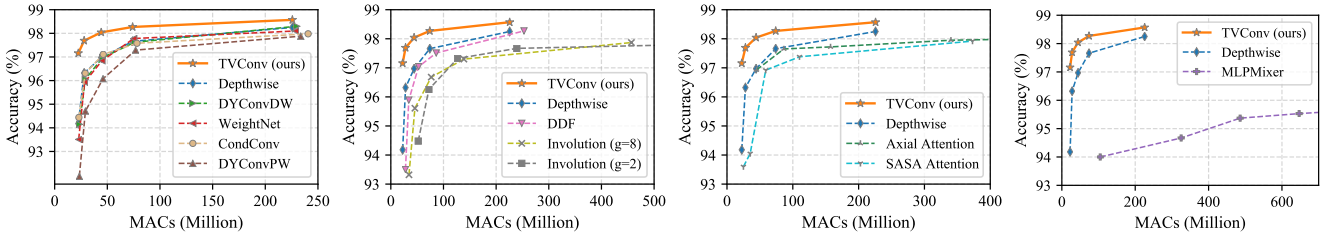


Figure 5. Visualization of our learned affinity maps. The first two rows depict affinity maps in MobileNetV2 x1.0 and x0.1, as the layer goes deeper from left to right. The bottom row shows the convergence process of an affinity map in MobileNetV2 x0.1 layer 4.

ize the affinity maps in TVConv with a constant of 1 (refer to Sec. 4.3 for further examination). Overall, we measure the accuracy of the validation set relative to computational cost in Multiply-Accumulate operations (MACs), throughput (FPS) and peak memory consumption. The accuracy is reported as the mean and standard deviation over 5 runs if not otherwise stated. We set the batch size as 1/2/4/8/16 and the number of threads as 1/2/4, and report the maximum throughput on a 4GB Raspberry Pi 4B for each model. The peak memory is measured with input size [8, 3, 96, 96].

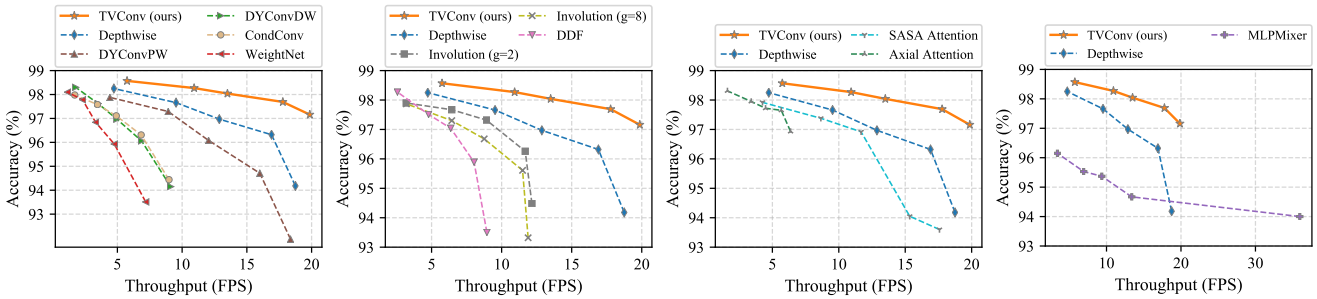
Comparison with depthwise conv. Fig. 4 shows a clear gap between TVConv and the depthwise conv when applied on the MobileNetV2 (MB) and ShuffleNetV2 (SF) with varied network width. In particular, TVConv for MobileNetV2 reduces the complexity by 3.05x while maintaining a high accuracy on the LFW dataset. On the other hand, TVConv boosts the accuracy by 2.98% under an extreme low complexity constraint. More quantitative comparisons can be found in Tab. 1, where a more significant 4.21% mean accuracy boost of applying TVConv on MBx0.1 is achieved. Throughput analysis in Fig. 7 shows a 2.3x improvement of MBx0.5 with TVConv over the original MBx1.0, though TVConv brings a fairly higher peak memory consumption as in shown Fig. 8. Note that the translation variant weights from TVConv reside on memory for query with only the fast READ operation performed.

Comparison with other operators. We compare with the related per-image dynamic conv, per-pixel dynamic conv and self-attention variants applied on the MobileNetV2 as well as the standalone MLPMixer. In particular, we compare with CondConv [53], WeightNet [24], DYConvDW [6], DYConvPW [6] as the per-image dynamic conv variants and with Involution [22], DDF [59] as the per-pixel dynamic conv variants. They are put at every stage of the network, as the way of TVConv being placed. SASA [31] and Axial attention [44] are compared as two efficient self-attention variants, each put at the last three stages of the network for better efficiency. Note that the MLPMixer [41] is out of the conv family. We include it for comparison for its potential layout-aware ability. Results are summarised as three types of trade-offs: the accuracy-complexity trade-



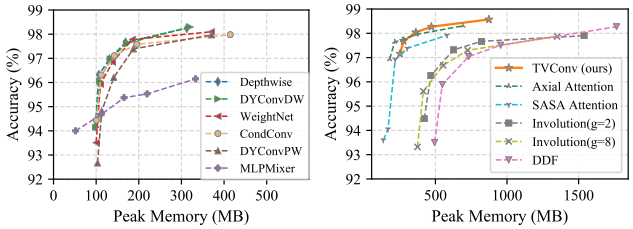
(a) Against **per-image** dynamic conv (b) Against **per-pixel** dynamic conv (c) Against **self-attention** variants (d) Against **MLPMixer**

Figure 6. TVConv strikes a better accuracy-complexity envelope on the LFW dataset compared to four related branches of works.



(a) Against **per-image** dynamic conv (b) Against **per-pixel** dynamic conv (c) Against **self-attention** variants (d) Against **MLPMixer**

Figure 7. TVConv achieves the best accuracy-throughput trade-off on the LFW dataset compared to related works.



(a) Highest peak memory $<$ 500MB (b) Highest peak memory $>$ 500MB

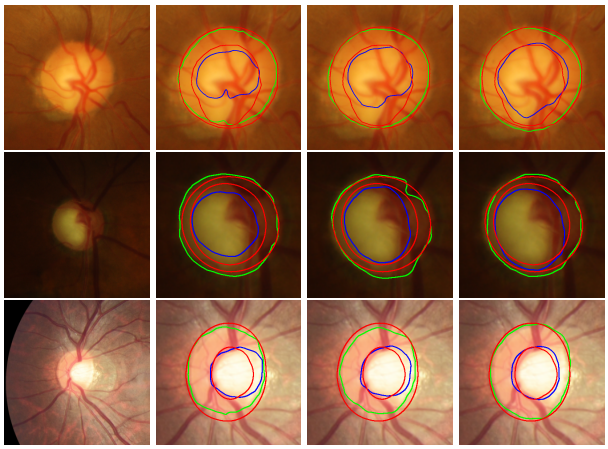
Figure 8. TVConv consumes a medium amount of peak memory.

off (see Fig. 6), accuracy-throughput trade-off (see Fig. 7) and accuracy-peak memory trade-off (see Fig. 8). TVConv consistently outperforms other operators with a better efficiency on accuracy-complexity and accuracy-throughput trade-offs. The gap is remarkable, especially with small network width where the spatial adaptiveness becomes more crucial. Besides, we find that many works of per-image dynamic conv tend to clutter together on the accuracy-complexity trade-off without clear boundaries and perform even inferior to the depthwise conv. This is because the cross-image dynamics are useless here for face recognition with small cross-image variance. For per-pixel dynamic conv, the weight-generating process incurs an expensive computational overhead for marginal accuracy gain. Though the Axial attention greatly improves the accuracy over the depthwise baseline, the computational overhead is still unaffordable. For the MLPMixer, it can easily achieve ultra-fast speed (*e.g.* 36.5 FPS for 94.00% accuracy). However, the accuracy grows slowly as the model size scales up. In contrast, TVConv keeps its high accuracy even at its smallest version (19.8 FPS for 97.16% accuracy). For peak memory usages, TVConv occupies less memory than the per-pixel dynamic conv and self-attention variants but fairly higher than others.

Learned affinity maps visualization. To further investigate the spatial adaptivity of TVConv, we visualize the learned affinity maps. As shown in the bottom row of Fig. 5, the affinity map progressively converges to a face contour. However, such a clear learning process does not occur for each layer. As in the top and middle rows of Fig. 5, the very early layers are mainly kept as a constant map without learning much meaningful layout information. This suggests that the very low-level features tend to be shared across the whole image probably because of the texture similarities and the presence of signal noise. For deeper layers, the learned affinity maps do not appear in the shape of a human face either. This aligns with the paper [35] saying that deeper layers are responsible for more abstract semantic concepts. Interestingly, we observe that for some affinity maps in deep layers, the most "activated" region does not appear at the center but rather tends to appear on the corners. We hypothesize that these corners carry more descriptive features. Because in the more peripheral region, its receptive filed covers a larger blank area, to which the zero-padding is applied. Such zero-padding would help exploit the spatial information, as indicated in the work [21].

4.2. Optic disc/cup segmentation

Due to the layout awareness, we expect this geometrical property to be helpful under a small data regime, particularly for medical image processing. To examine this,



(a) Inputs (b) DeepLabV3+ (c) w/ Depthwise (d) w/ TVConv

Figure 9. Qualitative segmentation comparisons. Column (a) are the inputs while column (b), (c) and (d) show the amplified segmentation results using different approaches. The red contours represent the ground truth while the green and blue lines show the predictions for optic discs and cups, respectively.

Metric	Task	Unseen domain	DeepLabV3+	w/ Depthwise	w/ TVConv
DSC↑	OC	A	79.39 ± 2.83	78.34 ± 2.27	**82.76 ± 1.41**
		B	75.85 ± 1.08	75.28 ± 1.34	**76.21 ± 1.56**
		C	**85.60 ± 0.61**	85.35 ± 0.79	85.52 ± 0.51
		D	81.24 ± 1.70	82.81 ± 0.94	**84.10 ± 0.76**
	OD	A	92.18 ± 0.64	**94.21 ± 0.90**	93.67 ± 1.24
		B	88.66 ± 0.78	89.57 ± 1.64	**90.10 ± 0.69**
		C	90.89 ± 1.46	90.26 ± 1.63	**92.91 ± 0.59**
		D	92.52 ± 0.24	92.96 ± 0.51	**93.25 ± 0.60**
	Mean		85.79	86.10	**87.31**
HD↓	OC	A	43.22 ± 3.86	42.65 ± 3.21	**36.32 ± 2.79**
		B	**30.69 ± 1.02**	31.75 ± 1.68	31.01 ± 1.46
		C	**21.44 ± 0.76**	21.85 ± 1.10	21.71 ± 0.78
		D	21.17 ± 1.56	19.56 ± 1.32	**17.97 ± 1.14**
	OD	A	25.23 ± 1.22	**20.36 ± 2.18**	21.96 ± 3.82
		B	31.63 ± 2.82	29.95 ± 3.21	**27.69 ± 3.01**
		C	24.62 ± 3.53	26.17 ± 3.75	**20.88 ± 1.55**
		D	19.34 ± 0.94	17.75 ± 1.25	**17.05 ± 1.14**
	Mean		27.17	26.26	**24.32**

Table 2. Comparison of domain generalization results on OD/OC segmentation tasks. Ten runs for each setting.

we conduct experiments on the optic disc and cup (OD, OC) segmentation tasks. We employ the fundus images from 4 clinical centers of public datasets [10, 28, 36], which are considered as data from different domains. Following the work [46], we take turns to use data from three domains for training and the remaining domain for evaluation. DeepLabV3+ [4] network is used as a strong baseline. For comparison, we replace the conv in its ASPP module with the inverted residual block [32], which is equipped with either the original depthwise conv or our TVConv as two variants. An Adam optimizer is used to train for 40 epochs. The learning rate starts from 1e-3 and is divided by 5 at epoch 30. Images are resized to 256 x 256 with batch size 16 for training but kept in the original size of 800 x 800 for evaluation. To make the data scarcity even more severe, we do not employ any data augmentation. For evaluation, we use two commonly adopted metrics, dice similarity coefficient (DSC) and Hausdorff distance (HD).

As reported in Tab. 2, with our TVConv, the network surpasses the baseline and its variant (with depthwise conv) on most unseen domains. Notably, comparison with the depthwise conv is necessary to ensure that our performance gain mainly comes from the operator itself instead of the structural change. TVConv improves the baseline by a considerable margin (an average DSC of 1.52%) compared to a slight improvement (an average DSC of 0.31%) brought by the inverted residual blocks. Qualitative comparisons are also provided in Fig. 9. Our TVConv generates more accurate and resilient segmentation results while others tend to produce deviated curves in regions with locally complicated or blurry textures. This better generalization ability is desired and particularly useful in practice. Besides, unlike the sparse prediction tasks (*e.g.*, face recognition), segmentation is in the category of dense prediction, indicating the broad applicability of TVConv in certain degree.

4.3. Ablation study

We conduct experiments to answer the following four questions. Unless otherwise specified, all experiments use a MBx0.2 architecture and are evaluated on the LFW dataset.

How sensitive is TVConv to various affine transformations? For some applications (*e.g.* face recognition for phone unlocking), it generally holds that the input image has a specific and fixed layout pattern. If not, an alignment would be involved and serve as a prepossessing step. Here, we explicitly relax this condition to investigate the sensitivity of TVConv. We apply four affine transforms independently to the training dataset and report the validation results in Fig. 10. We can see that TVConv performs robustly under a reasonable amount of rotation, shearing, or scaling. It performs even more robustly than the depthwise conv against the shearing. But for translation, as expected, the accuracy gain gradually shrinks, and TVConv would degrade to a vanilla depthwise conv under severe translation.

How should we initialize the affinity maps? By default, we initialize the affinity maps $\mathbf{A}$ to be constant maps (*e.g.*, a constant 1). In this way, the training stability can be better guaranteed. Because the weight-generating block $\mathcal{B}$ produces identical weights for different positions at the beginning, and TVConv starts as a plain depthwise conv. This warm-up process enables TVConv to gradually evolve and manifest its adaptivity. Another possible way (see Fig. 11) is to initialize from the data statistics, *e.g.* to calculate the

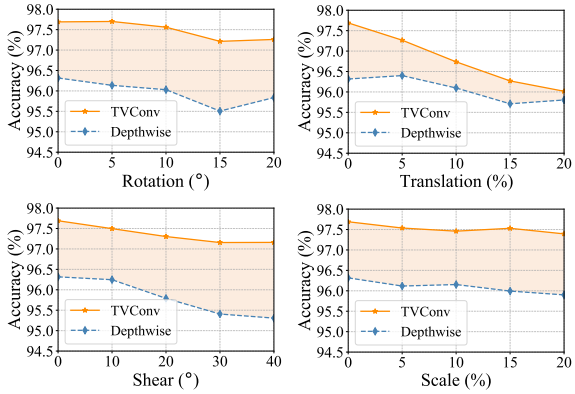


Figure 10. TVConv surpasses the depthwise conv for various transformations, while for translation the gap gradually shrinks.

(a) Mean (b) Std (c) D(mean) (d) D(std) (e) Learned

Figure 11. Statistics calculated from the datasets: (a) mean, (b) std, as well as their down-sampled version (c), (d), respectively. These maps might be used to initialize the affinity maps. However, it may geometrically misalign with learned affinity maps (e).

mean and std maps from the training images, and then down-sample them to a specific size identical to the affinity maps. Though it seems promising, our experiments (see Tab. 3) suggest that it does not provide better performance than simple constant initialization. Probably because in the post layers of a network, the learned affinity maps generally carry some high-level semantic information, which can be visually misaligned with the input contours.

Does the model benefit from a bigger weight-generating block? In line with the work [1], over-parameterization generally brings better performance in our experiments, as in Tab. 4. In particular, we examine the role of different components, including the depth (layers), width (channels) of the weight-generating block $\mathcal{B}$ and the number of channels in affinity maps $\mathbf{A}$. With deeper, wider layers in $\mathcal{B}$ or more channels in $\mathbf{A}$, the model generally performs better. However, the performance slightly drops from the peak when the over-parameterization goes too far. This is because a larger model might require more iterations for training and can suffer from over-fitting.

Where should TVConv be placed in a network? Different patterns appeared in the learned affinity maps (see Fig. 5) motivate us to further investigate the proper position for placing our TVConv. As reported in Tab. 5, applying a single TVConv in post stages is more effective in reducing the error while applying in early stages (*e.g.* S1, S2) might even hurt the performance. Moreover, by stacking more stages with TVConv, the error is monotonically reduced. We place TVConv in all stages as our default setting.

Init.	Error (%) for ×0.1	Error (%) for ×0.5	Error (%) for ×1.0
data statistics	2.92 ± 0.22	1.75 ± 0.18	1.62 ± 0.20
constant 1	**2.84 ± 0.11**	**1.73 ± 0.07**	**1.43 ± 0.12**

Table 3. Initialization from constant maps (*e.g.* constant 1) works better than initialization from the data statistics.

$\mathcal{B}$ #Chans.	Error (%)	$\mathcal{B}$ #Layers	Error (%)	$\mathbf{A}$ #Chans.	Error (%)
128	2.60 ± 0.10	4	2.49 ± 0.16	8	2.49 ± 0.16
64	**2.31 ± 0.12**	**3**	**2.31 ± 0.12**	**4**	**2.31 ± 0.12**
32	2.65 ± 0.21	2	2.44 ± 0.15	2	2.44 ± 0.19
8	2.65 ± 0.12	1	2.48 ± 0.27	1	2.52 ± 0.15

Table 4. Ablation under the varied number of channels and inter layers for weight generating block $\mathcal{B}$, as well as the varied number of channels for affinity maps $\mathbf{A}$. We use hyper-parameters highlighted in bold as our default setting elsewhere in the paper.

Stages	S1	S2	S3	S4	S5	S6	S7
Error(%)	4.01	3.73	3.42	3.43	3.34	3.25	3.39
Stages	Baseline	**S1-S7**	S2-S7	S3-S7	S4-S7	S5-S7	S6-S7
Error(%)	3.68	**2.31**	2.35	2.38	2.51	2.53	2.61

Table 5. Ablation of applying TVConv in different stages. Excluding the input layer, MobileNetV2 has 7 stages. Stages S1-S7 correspond to layers L1, L2-L3, L4-L6, L7-L10, L11-L13, L14-L16 and L17, respectively. Error SD is omitted for simplicity.

5. Conclusion and Future Works

We present a conceptually simple, novel, and efficient fundamental operator TVConv as a versatile replacement of existing convolution variants for layout-aware visual processing. A comprehensive study confirms the promising efficiency as well as the better generalization ability. TVConv can be seamlessly integrated into various neural architectures. Moreover, the learned affinity maps provide additional space for network visualization and interpretation. Nonetheless, TVConv has its own limitations. Since our prototype assumes the affinity maps to be of fixed size for a certain layer, networks with TVConv might fail to process inputs of varying resolutions. This might be addressed by adaptive pooling or sampling techniques. Besides, TVConv may degrade to a plain depthwise convolution when faced with severe layout transformations. One intuitive idea to solve this is to work with a simple Spatial Transformer Network [18], which produces a unified affine transformation matrix to globally transform the affinity maps in different layers. Another promising extension is to work with the dynamic convolution in a complementary way by efficiently producing pixel-wise dynamic bias. These two extensions would broaden the reach of our TVConv to serve more unconstrained scenarios, which are left for future work.

Acknowledgement

This work was supported, in part, by Hong Kong General Research Fund (under grant number 16200120).

References

[1] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019. 4, 8

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 4

[3] Jin Chen, Xijun Wang, Zichao Guo, Xiangyu Zhang, and Jian Sun. Dynamic region-aware convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8064–8073, 2021. 2

[4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 7

[5] Sheng Chen, Yang Liu, Xiang Gao, and Zhen Han. Mobilefacenets: Efficient cnns for accurate real-time face verification on mobile devices. In *Chinese Conference on Biometric Recognition*, pages 428–438. Springer, 2018. 4

[6] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11030–11039, 2020. 2, 5

[7] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017. 3

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020. 3

[9] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019. 3

[10] Francisco Fumero, Silvia Alayón, José L Sanchez, Jose Sigut, and M Gonzalez-Hernandez. Rim-one: An open retinal image database for optic nerve evaluation. In *2011 24th international symposium on computer-based medical systems (CBMS)*, pages 1–6. IEEE, 2011. 7

[11] Tianlang He, Jiajie Tan, Weipeng Zhuo, Maximilian Printz, and S-H Gary Chan. Tackling multipath and biased training data for imu-assisted ble proximity detection. *arXiv preprint arXiv:2201.03817*, 2022. 1

[12] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019. 2

[13] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2

[14] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3588–3597, 2018. 3

[15] Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008. 1, 4

[16] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 603–612, 2019. 3

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 4

[18] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28:2017–2025, 2015. 8

[19] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. *Advances in neural information processing systems*, 29:667–675, 2016. 2, 3

[20] Zi-Rong Jin, Liang-Jian Deng, Tai-Xiang Jiang, and Tian-Jing Zhang. Laconv: Local adaptive convolution for image fusion. *arXiv preprint arXiv:2107.11617*, 2021. 2

[21] Osman Semih Kayhan and Jan C van Gemert. On translation invariance in cnns: Convolutional layers can exploit absolute spatial location. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14274–14285, 2020. 6

[22] Duo Li, Jie Hu, Changhu Wang, Xiangtai Li, Qi She, Lei Zhu, Tong Zhang, and Qifeng Chen. Involution: Inverting the inherence of convolution for visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12321–12330, 2021. 2, 5

[23] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021. 3

[24] Ningning Ma, Xiangyu Zhang, Jiawei Huang, and Jian Sun. Weightnet: Revisiting the design space of weight networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV*, 2020. 2, 5

[25] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018. 1, 2, 4

[26] Ben Mildenhall, Jonathan T Barron, Jiawen Chen, Dillon Sharlet, Ren Ng, and Robert Carroll. Burst denoising with kernel prediction networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2502–2510, 2018. 3

[27] Stylianos Moschoglou, Athanasios Papaioannou, Christos Sagonas, Jiankang Deng, Irene Kotsia, and Stefanos Zafeiriou. Agedb: the first manually collected, in-the-wild age database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 51–59, 2017. 4

[28] José Ignacio Orlando, Huazhu Fu, João Barbosa Breda, Karel van Keer, Deepti R Bathula, Andrés Diaz-Pinto, Ruogu Fang, Pheng-Ann Heng, Jeyoung Kim, JoonHo Lee, et al. Refuge challenge: A unified framework for evaluating automated methods for glaucoma assessment from fundus photographs. *Medical image analysis*, 59:101570, 2020. 7

[29] Jongchan Park, Sanghyun Woo, Joon-Young Lee, and In So Kweon. Bam: Bottleneck attention module. *arXiv preprint arXiv:1807.06514*, 2018. 3

[30] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR, 2018. 3

[31] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. *Advances in Neural Information Processing Systems*, 32, 2019. 5

[32] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 1, 2, 4, 7

[33] Soumyadip Sengupta, Jun-Cheng Chen, Carlos Castillo, Vishal M Patel, Rama Chellappa, and David W Jacobs. Frontal to profile face verification in the wild. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9. IEEE, 2016. 4

[34] Laurent Sifre and Prof Stéphane Mallat. Rigid-motion scattering for image classification author. *English. Supervisor: Prof. Stéphane Mallat. Ph. D. Thesis. Ecole Polytechnique*, 2014. 2

[35] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 6

[36] Jayanthi Sivaswamy, S Krishnadas, Arunava Chakravarty, G Joshi, A Syed Tabish, et al. A comprehensive retinal image dataset for the assessment of glaucoma from the optic nerve head analysis. *JSM Biomedical Imaging Data Papers*, 2(1):1004, 2015. 7

[37] Hang Su, Varun Jampani, Deqing Sun, Orazio Gallo, Erik Learned-Miller, and Jan Kautz. Pixel-adaptive convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11166–11175, 2019. 2

[38] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019. 2

[39] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019. 1, 2, 4

[40] Mingxing Tan and Quoc V Le. Efficientnetv2: Smaller models and faster training. *arXiv preprint arXiv:2104.00298*, 2021. 2

[41] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34, 2021. 5

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 2, 3, 4

[43] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu. Additive margin softmax for face verification. *IEEE Signal Processing Letters*, 25(7):926–930, 2018. 4

[44] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *European Conference on Computer Vision*, pages 108–126. Springer, 2020. 5

[45] Jiaqi Wang, Kai Chen, Rui Xu, Ziwei Liu, Chen Change Loy, and Dahua Lin. Carafe: Content-aware reassembly of features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3007–3016, 2019. 2

[46] Shujun Wang, Lequan Yu, Kang Li, Xin Yang, Chi-Wing Fu, and Pheng-Ann Heng. Dofe: Domain-oriented feature embedding for generalizable fundus image segmentation on unseen datasets. *IEEE Transactions on Medical Imaging*, 39(12):4237–4248, 2020. 7

[47] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 3

[48] Yulong Wang, Xiaolu Zhang, Xiaolin Hu, Bo Zhang, and Hang Su. Dynamic network pruning with interpretable layerwise channel selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6299–6306, 2020. 2

[49] Ze Wang, Zichen Miao, Jun Hu, and Qiang Qiu. Adaptive convolutions with per-pixel dynamic filter atom. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12302–12311, 2021. 1, 2, 4

[50] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 2, 3

[51] Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5028–5037, 2017. 1

[52] Yu-Syuan Xu, Shou-Yao Roy Tseng, Yu Tseng, Hsien-Kai Kuo, and Yi-Min Tsai. Unified dynamic convolutional network for super-resolution with variational degradations. In

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12496–12505, 2020. 3

[53] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. *Advances in Neural Information Processing Systems*, 32:1307–1318, 2019. 2, 5

[54] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014. 4

[55] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019. 3

[56] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018. 2

[57] Yikang Zhang, Jian Zhang, Qiang Wang, and Zhao Zhong. Dynet: Dynamic convolution for accelerating convolutional neural networks. *arXiv preprint arXiv:2004.10694*, 2020. 2

[58] Tianyue Zheng, Weihong Deng, and Jiani Hu. Cross-age lfw: A database for studying cross-age face recognition in unconstrained environments. *arXiv preprint arXiv:1708.08197*, 2017. 4

[59] Jingkai Zhou, Varun Jampani, Zhixiong Pi, Qiong Liu, and Ming-Hsuan Yang. Decoupled dynamic filter networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6647–6656, 2021. 2, 5

[60] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9308–9316, 2019. 3

[61] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 2