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

Jierun Chen\textsuperscript{1}, Tianlang He\textsuperscript{1}, Weipeng Zhuo\textsuperscript{1}, Li Ma\textsuperscript{1}, Sangtae Ha\textsuperscript{2}, S.-H. Gary Chan\textsuperscript{1}
\textsuperscript{1}The Hong Kong University of Science and Technology, \textsuperscript{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.

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.
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 $3.1 \times$ and correspondingly accelerates the throughput by $2.3 \times$ 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 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 lightweight 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 \( \mathbf{W} \in \mathbb{R}^{c \times k \times k} \), 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:

\[
\mathbf{O}_{i,j} = \mathbf{W} \otimes \mathbf{P}_{i,j},
\]

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 \((1 \times 1)\) 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:

\[
\mathbf{O}_{i,j} = \mathbf{W}_{i,j} \otimes \mathbf{P}_{i,j}.
\]

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:

\[
\mathbf{W}' = \mathbf{B}' \mathbf{A}',
\]

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:

\[
\mathbf{W} = \mathbf{B}(\mathbf{A}),
\]

where \( \mathbf{A} \in \mathbb{R}^{c_A \times h \times w} \) denotes our affinity maps and \( \mathbf{B} \) 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 \( \mathbf{B} \), 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 \( \mathbf{B} \) to yield the weight \( \mathbf{W} \in \mathbb{R}^{k^2 \times h \times w} \) (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 \( \mathbf{O} \in \mathbb{R}^{c \times h \times w} \).](image-url)
Dynamic conv. Formally by replacing the affinity maps the learned affinity maps in Sec. 4. The dynamic conv cannot pre-translation equivariant [49]. The dynamic conv cannot pre-generating process has to be repeated for different inputs, and the “batch size” for A can be interpreted as 1. The affinity maps 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 B to be over-parameterized up to the memory limit. After training, the affinity maps 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. Mobilenets, 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 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 $A_{i,j} = W$ for different $(i,j)$. We visualize and discuss the learned affinity maps in Sec. 4.

Dynamic conv. Formally by replacing the affinity maps A in Eq. (4) with the input 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 pre-dict 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 A resemble these two characters implicitly. Firstly, if two positions within 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-FP [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-
Figure 4. The accuracy-complexity envelope on the LFW dataset.

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

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], DYConv-vDW [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, as the way of TVConv being placed. Overall, we measure 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].
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 field 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,
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 × 256 with batch size 16 for training but kept in the original size of 800 × 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 preprocessing 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 $A$ to be constant maps (e.g., a constant 1). In this way, the training stability can be better guaranteed. Because the weight-generating block $B$ 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

<table>
<thead>
<tr>
<th>Metric</th>
<th>Task</th>
<th>Unseen domain</th>
<th>DeepLabV3+ w/ Depthwise</th>
<th>w/ TVConv</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSC↑</td>
<td>OC</td>
<td>A 79.39 ± 2.83</td>
<td>78.54 ± 2.27</td>
<td>82.76 ± 1.41</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>75.85 ± 1.08</td>
<td>75.28 ± 1.34</td>
<td>76.21 ± 1.56</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>85.60 ± 0.61</td>
<td>85.35 ± 0.79</td>
<td>85.52 ± 0.51</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>81.24 ± 1.70</td>
<td>82.81 ± 0.94</td>
<td>84.10 ± 0.76</td>
</tr>
<tr>
<td>DSC↑</td>
<td>OD</td>
<td>A 92.18 ± 0.64</td>
<td>94.21 ± 0.90</td>
<td>93.67 ± 1.24</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>88.66 ± 0.78</td>
<td>89.57 ± 1.64</td>
<td>90.10 ± 0.69</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>90.89 ± 1.46</td>
<td>90.26 ± 1.63</td>
<td>92.91 ± 0.59</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>92.52 ± 0.24</td>
<td>92.96 ± 0.51</td>
<td>93.25 ± 0.60</td>
</tr>
<tr>
<td>Mean</td>
<td></td>
<td>85.79</td>
<td>86.10</td>
<td>87.31</td>
</tr>
<tr>
<td>HD↓</td>
<td>OC</td>
<td>A 25.23 ± 1.22</td>
<td>20.36 ± 2.18</td>
<td>21.96 ± 3.82</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>31.63 ± 2.82</td>
<td>29.95 ± 3.21</td>
<td>27.69 ± 3.01</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>19.34 ± 0.94</td>
<td>17.75 ± 1.25</td>
<td>17.05 ± 1.14</td>
</tr>
<tr>
<td>Mean</td>
<td></td>
<td>27.17</td>
<td>26.26</td>
<td>24.32</td>
</tr>
</tbody>
</table>
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 $B$ and the number of channels in affinity maps $A$. With deeper, wider layers in $B$ or more channels in $A$, 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.

### 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


