Adaptive Convolutions with Per-pixel Dynamic Filter Atom

Ze Wang[1], Zichen Miao[1], Jun Hu[2], and Qiang Qiu[1]
Purdue University[1] Facebook[2]
{zewang, miaoz, qqiu}@purdue.edu junhu2@fb.com

Abstract

Applying feature dependent network weights have been proved to be effective in many fields. However, in practice, restricted by the enormous size of model parameters and memory footprints, scalable and versatile dynamic convolutions with per-pixel adapted filters are yet to be fully explored. In this paper, we address this challenge by decomposing filters, adapted to each spatial position, over dynamic filter atoms generated by a light-weight network from local features. Adaptive receptive fields can be supported by further representing each filter atom over sets of pre-fixed multi-scale bases. As plug-and-play replacements to convolutional layers, the introduced adaptive convolutions with per-pixel dynamic atoms enable explicit modeling of intra-image variance, while avoiding heavy computation, parameters, and memory cost. Our method preserves the appealing properties of conventional convolutions as being translation-equivariant and parametrically efficient. We present experiments to show that, the proposed method delivers comparable or even better performance across tasks, and are particularly effective on handling tasks with significant intra-image variance.

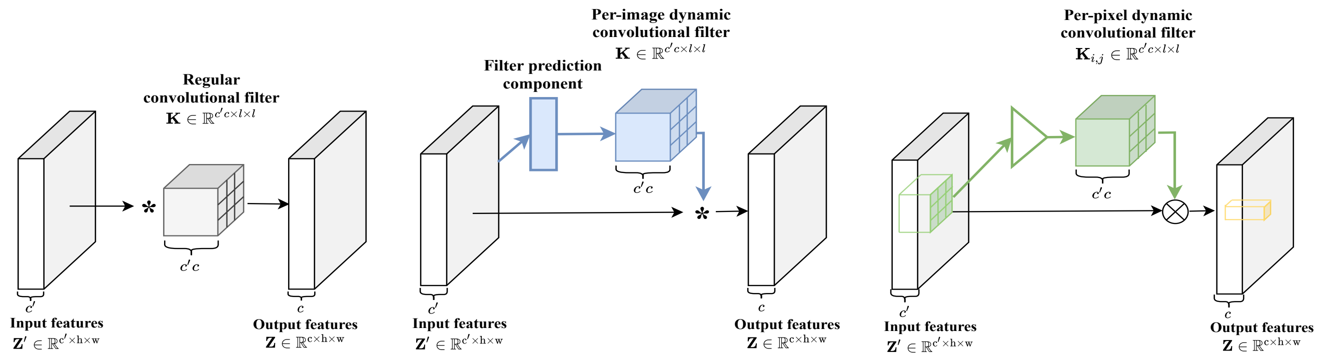
1. Introduction

The idea of data or context dependent network weights have long been studied in the research of neural networks. Many concepts, like fast weight [2, 32, 31] and dynamic plasticity [24, 25] are developed to explicitly model the evolution of model parameters, and have demonstrated improved performance on sequential data. In [2, 32, 31], the parameters in these networks can be divided into two groups: slow weights that are learned through training with gradient descent, and fast weights that are generated on-the-fly depending on both slow weights and observed data.

In recent years, similar idea has been extended to dynamic convolutions. Standard convolutions as in Figure 1a uses shared filters across all samples and all spatial position. Dynamic convolutions, as in Figure 1b, allow convolutional filters to be adapted to data in one-shot (as opposite to evolving through sequential observations). Dynamic filter networks (DFN) [16], conditionally parameterized convolutions (CondConv) [42], dynamic convolutions with attention (DY-CNN) [6] are introduced to allow convolutional filters to be adapted to the current observed input, and explicitly modeling the inter-sample variance among images. While improvements on certain tasks have been shown, the flexibility comes at the cost of extra parameter and computation [6, 42], and sacrificing the translation equivariant property of CNNs [16]. More importantly, it is practically infeasible to extend such methods from per-image adapted filters to per-pixel adapted filters due to the prohibitive memory footprints of applying per-pixel adapted high-dimension filters as we will show in Section 4.4.

In this paper, we enable CNNs with per-pixel adaptive convolutions as illustrated in Figure 1c, at any network layers to better model intra-image variance. We introduce *Adaptive Convolutions with Dynamic Atoms (ACDA)*, a versatile and scalable convolutional layer that allows per-pixel specific filters to be adaptively generated from each local input feature patch across spatial position. To remedy the prohibitively high cost on generating and applying per-pixel adaptive filters in high dimensions, we decompose filters over dynamically generated low-dimensional filter atoms at each spatial location. The adaptive filters can now be reconstructed by linearly combining these per-pixel specific dynamic atoms with cross-location shared compositional coefficients, as illustrated in Figure 2. Most importantly, the decomposition enables a fast two-layer implementation of the adaptive convolutions as shown in Figure 3, which reduces the prohibitive computation and memory footprints of applying per-pixel specific convolutions to a level that matches standard convolutions, allowing our method becomes a versatile replacement to standard convolutions across layers in any CNNs.

To achieve adaptive receptive fields, we further decompose each filter atom over sets of multi-scale pre-fixed atom bases as in Figure 4, for a two-level filter decomposition. Now, instead of directly generating atoms, only per-pixel basis coefficients are required to be generated. The multi-scale atom bases and the generated basis coefficients reconstruct dynamic atoms, and allow the receptive field at each spatial position to be selectively decided from the local features. Meanwhile the adaptive filters are effectively regular-


Regular convolutional filter
$\mathbf{K} \in \mathbb{R}^{c'c \times l \times l}$
$c'c$
c'
Input features
$\mathbf{Z}' \in \mathbb{R}^{c' \times h \times w}$
c
Output features
$\mathbf{Z} \in \mathbb{R}^{c \times h \times w}$
Filter prediction component
Per-image dynamic convolutional filter
$\mathbf{K} \in \mathbb{R}^{c'c \times l \times l}$
$c'c$
c'
Input features
$\mathbf{Z}' \in \mathbb{R}^{c' \times h \times w}$
c
Output features
$\mathbf{Z} \in \mathbb{R}^{c \times h \times w}$
Per-pixel dynamic convolutional filter
$\mathbf{K}_{i,j} \in \mathbb{R}^{c'c \times l \times l}$
$c'c$
c'
Input features
$\mathbf{Z}' \in \mathbb{R}^{c' \times h \times w}$
c
Output features
$\mathbf{Z} \in \mathbb{R}^{c \times h \times w}$

(a) Standard convolutions (b) Per-image adaptive convolutions (c) Per-pixel adaptive convolutions

Figure 1: Convolutional layers. (a) Standard convolutions with filters shared across all samples and spatial position. (b) Per-image adaptive convolutions with per-image specific filters as in [6, 16, 42]. (c) The proposed per-pixel adaptive convolutions dynamically generate filters conditioned on local feature patches, and explicitly model intra-image variance using per-pixel specific filters.

ized by the prefixed atom bases, which accelerates the learning of the large-size adaptive filter generation. Importantly, our approach maintains and even reduces parameters and computation, and preserves appealing properties of CNNs including translation equivariant and weight sharing.

We show empirically that, our approach can work as plug-and-play replacements to standard convolutional layers. We demonstrate the effectiveness of the proposed method using image classification, crowd counting, and real-world image restorations as example tasks that require handling significant intra-image variance.

2. Related Work

Parameter predictions. Instead of directly training parameters for deep networks, parameter predictions are discussed under various motivations. Parameters in Hypernetworks [10] are formulated as the outputs of a network, which achieves parameter compression. BasisGAN [39] adopts bases generators to model the space of parameters and achieve stochastic conditional image generations. Predicting networks weights given samples [27, 8] is widely used for few shot image classification.

Dynamic convolutions. The idea of relaxing the strict weight sharing across spatial position of convolutions has been discussed in works like locally connected networks [3] and dynamic filter networks [16]. And applying locally feature dependent convolutional filters has demonstrated effectiveness on real-world image restorations [4, 26, 41], where degradation models are non-uniformed across spatial position. However, restricted by practical costs, the usages of adaptive convolutions are usually simplified based on certain tasks [34] and assumptions, e.g., the dynamic convolutions are applied only to the image domain [4, 26, 15], or shared across channels in deep features [41], which prevent the methods from being extended to universal applications. Dynamic convolution with attention is introduced in [6] by assembling multiple kernels through attention mechanism. The flexibility is restricted by the fixed number of filters. Deformable convolutions [7, 48] allow adapted kernel shape while fixing the values in kernels, thus are orthogonal to our efforts, and can potentially work together with the proposed framework of adaptive convolutions with dynamic filter atoms.

3. Method

In this section, we first present per-pixel adaptive filter generations with filter decomposition over dynamic atoms, and then show a practical two-layer implementation to address the prohibitive memory footprints and accelerate the speed. We then decompose filter atoms over sets of prefixed multi-scale bases to further enable dynamic receptive fields while maintaining parameter size. We end this section with illustrative toy examples to show the advantages of our method on addressing intra-image variance, and preserving the translation equivariance property.

3.1. Preliminary

We denote scalars, vectors and tensors with lower-case, bold lower-case, and bold upper-case letters, e.g., n, $\mathbf{x}$, $\mathbf{X}$, respectively. We use $\mathbf{Z}' \in \mathbb{R}^{c' \times h \times w}$ and $\mathbf{Z} \in \mathbb{R}^{c \times h \times w}$ to denote the input and output features of a typical convolutional layer with c' input and c output channels, each of which has a spatial resolution of $h \times w$. The corresponding convolutional filter is denoted as $\mathbf{K} \in \mathbb{R}^{c \times c' \times l \times l}$, where l is the kernel size. We use $[i, j]$ to denote the spatial position of a feature map at the i-th row and the j-th column, and $\mathbf{z}_{i,j} = \mathbf{Z}[i, j] \in \mathbb{R}^c$ is the feature vector in $\mathbf{Z}$ at position $[i, j]$. We use $\mathcal{N}^{\delta}_{\mathbf{Z}[i,j]}$ to denote the size-δ neighborhood region of $\mathbf{Z}[i, j]$, i.e., $\mathcal{N}^{\delta}_{\mathbf{Z}[i,j]} = \{\mathbf{Z}[i-u, j-v]\}_{-\delta \le u \le \delta, -\delta \le v \le \delta}$. In the following discussions, without loss of generality, we assume convolutions with a stride of 1 and padding for consistent input and output resolutions.

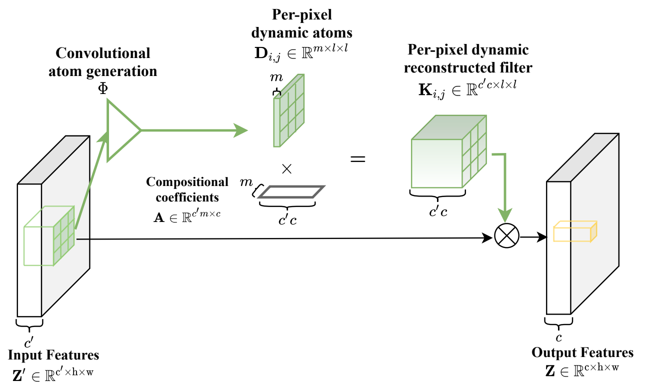

Convolutional atom generation
Φ
Per-pixel dynamic atoms
$\mathbf{D}_{i,j} \in \mathbb{R}^{m\times l\times l}$
m
Per-pixel dynamic reconstructed filter
$\mathbf{K}_{i,j} \in \mathbb{R}^{c'c\times l\times l}$
Compositional coefficients
$\mathbf{A} \in \mathbb{R}^{c'm\times c}$
m
$c'c$
$\times$
$=$
$c'c$
c'
Input Features
$\mathbf{Z}' \in \mathbb{R}^{c'\times h\times w}$
c
Output Features
$\mathbf{Z} \in \mathbb{R}^{c\times h\times w}$

Figure 2: Adaptive convolution with per-pixel dynamic atoms. An atom generation network convolves with the input to generate the per-pixel dynamic filter atoms $\mathbf{D}_{i,j}$, which then multiply with the cross-spatial shared compositional coefficients $\mathbf{A}$ to reconstruct full-size adaptive filters $\mathbf{K}_{i,j}$ across position. The predicted filters are then applied back to local regions centered around each spatial position.

3.2. Adaptive Filter Generation

Conventional convolutional layers, as in Figure 1a, learn filters that are shared both across all spatial position of an input feature map, and among input features from different input images. Formally, typical convolutions in CNNs at a specific position can be expressed as:

$$\begin{aligned}\mathbf{Z}[i,j] &= \mathcal{F}(\mathcal{N}^{\delta}_{\mathbf{Z}[i,j]}; \mathbf{K}) \\ &= \sum_{u=-\delta}^{\delta}\sum_{v=-\delta}^{\delta} \mathbf{K}[u,v]\cdot\mathbf{Z}[i-u, j-v],\end{aligned} \tag{1}$$

where we use $\mathcal{F}$ to denote the convolution operation, and $\delta = \lfloor l/2 \rfloor$. The convolutional filter $\mathbf{K}$ is shared across spatial position of $\mathbf{Z}'$, and remains fixed after training.

We aim at adaptively generating per-pixel convolutional filters conditioned on the corresponding neighborhood regions of input features, as illustrated in Figure 1c, to better handle intra-image variance. Formally, the adaptive convolution is expressed as:

$$\begin{aligned}\mathbf{Z}[i,j] &= \mathcal{F}(\mathcal{N}^{\delta}_{\mathbf{Z}[i,j]}; \mathbf{K}_{i,j}) \\ &= \sum_{u=-\delta}^{\delta}\sum_{v=-\delta}^{\delta} \mathbf{K}_{i,j}[u,v]\cdot\mathbf{Z}[i-u, j-v],\end{aligned} \tag{2}$$

where $\mathbf{K}$ here becomes a collection of $h \times w$ local filters, each of which is denoted as $\mathbf{K}_{i,j}$, and generated as

$$\mathbf{K}_{i,j} = \Phi(\mathcal{N}^{\delta'}_{\mathbf{Z}[i,j]}; \theta). \tag{3}$$

Φ here is the convolutional filter generation network parametrized by θ, which is end-to-end trained. Given a specific spatial position $[i,j]$ in the input features $\mathbf{Z}'$, Φ takes as input the local region centered around $\mathbf{Z}[i,j]$, and predicts the local adaptive filter $\mathbf{K}_{i,j} \in \mathbb{R}^{c\times c'\times l\times l}$. The predicted filter $\mathbf{K}_{i,j}$ is then applied back to a local region centered around $\mathbf{Z}[i,j]$, and outputs the vector at position $[i,j]$ in the output feature $\mathbf{Z}[i,j] \in \mathbb{R}^{c}$. Note that δ here does not necessarily equal to δ', i.e., at each spatial position, the local neighborhood region fed into Φ can be either smaller or larger than the region the generated filter is applied back to.

CNNs nowadays usually have high-dimensional filters, which make direct filter generations infeasible considering the size of parameters and computation. Moreover, since now the convolutional filters are per-pixel specific, *whose gradients need to be stored independently for backward computation*, this will lead to dramatically large memory footprints. For example, with a typical setting of $c' = c = h = w = 100$, and $l = 3$, a single layer will consume 26.8GB memory for storing the per-pixel specific gradients, which is practically prohibitive. This is also the reason why existing per-sample specific adaptive convolutions [16, 42, 6] can hardly be extended to per-pixel adaptations.

Filter atom decomposition. It is shown in [28] that a convolutional filter in a CNN can be decomposed as a linear combination of pre-fixed bases (visualized in Appendix Figure A). We adopt filter decomposition as visualized in Figure 2, where a convolutional filter is decomposed as a linear combination of m dynamic filter atoms $\mathbf{D} \in \mathbb{R}^{m\times l\times l}$ as $\mathbf{K} = \mathbf{AD}$, $\mathbf{A} \in \mathbb{R}^{c\times c'\times m}$ is the composition coefficients. The filter atoms at each spatial position is generated by

$$\mathbf{D}_{i,j} \in \mathbb{R}^{m\times l\times l} = \Phi(\mathcal{N}^{\delta'}_{\mathbf{Z}[i,j]}; \theta). \tag{4}$$

We slightly abuse the notation and use Φ to denote the convolutional atom generation network now. After decomposition, the spatial patterns of the convolutional filters are decided by the filter atoms, which are very low-dimensional comparing to the filters.

Two-layer implementation. Generating dynamic filter atoms of low dimensions significantly reduces the parameters and computation. Moreover, as visualized in Figure 3, the filter decomposition further allows the forward pass in (2) being decomposed into two convolutions, each of which involves multiplications between mild-size tensors only. Specifically, given the generated dynamic atoms, the forward pass is now decomposed into two steps:

- First, in **atom convolution**, the input features with c' channels are convolved spatially only with each of the m generated dynamic filter atoms, and output the intermediate features $\tilde{\mathbf{Z}}$ with $c'm$ channels:

$$\tilde{\mathbf{Z}}[i,j] \in \mathbb{R}^{c'm} = ||\mathcal{F}(\mathcal{N}^{\delta}_{\mathbf{Z}[i,j]}; \mathbf{D}_{i,j}[b])||_{b=\{1,\ldots,m\}}, \tag{5}$$

where $\mathbf{D}_{i,j}[b]$ denotes the b-th generated filter atom at position $[i,j]$, and $||\cdot||_{b=\{1,\ldots m\}}$ here denotes the channel-wise

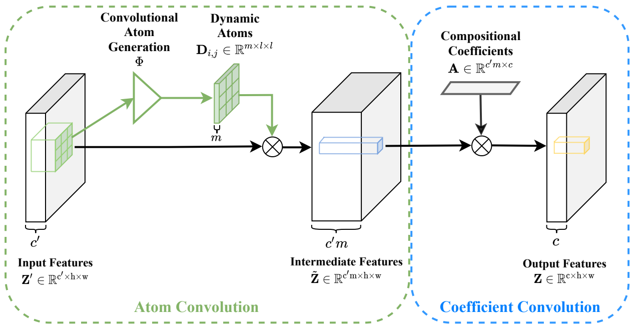

Convolutional Atom Generation Φ
Dynamic Atoms $\mathbf{D}_{i,j} \in \mathbb{R}^{m \times l \times l}$
m
Compositional Coefficients $\mathbf{A} \in \mathbb{R}^{c'm \times c}$
c'
$c'm$
c
Input Features $\mathbf{Z}' \in \mathbb{R}^{c' \times h \times w}$
Intermediate Features $\tilde{\mathbf{Z}} \in \mathbb{R}^{c'm \times h \times w}$
Output Features $\mathbf{Z} \in \mathbb{R}^{c \times h \times w}$
Atom Convolution
Coefficient Convolution

Figure 3: The efficient two-layer implementation of ACDA. The convolutional atom generation network first convolves with the input features to output dynamic atoms $\mathbf{D}_{i,j}$ at each spatial position, which are then convolved spatially with the input features to output the intermediate atom outputs $\tilde{\mathbf{Z}}$. A linear transformation using the compositional coefficients $\mathbf{A}$ is followed for the final outputs.

concatenation of the features.
- Second, in **coefficient convolution**, given the intermediate features $\tilde{\mathbf{Z}}$, the final outputs can be obtained by applying the composition coefficients $\mathbf{A} \in \mathbb{R}^{c \times c'm}$ and linearly combining the intermediate features:

$$\mathbf{Z} \in \mathbb{R}^{c \times h \times w} = \mathbf{A}\tilde{\mathbf{Z}}. \tag{6}$$

In practice, this step can be efficiently implemented by a 1×1 convolutions as $\mathbf{A}$ now is a linear transform shared across spatial position. Since all the involved operations are linear, this *two-layer implementation* in Figure 3 exactly equals to applying reconstructed full-size filters at each local position as in Figure 2, yet it can prevent the prohibitive memory footprints, reduce computation, and accelerate the speed as we will show in Section 4.4.

Multi-scale atom bases for adaptive receptive fields. To the best of our knowledge, adaptive receptive fields in convolutional filters have seldom been discussed in previous works. Selectively deciding receptive fields at each spatial position can potentially benefit tasks with great intra-image scale variance, e.g., crowd counting, where the sizes of the concerned objects in a single image can vary significantly. However, the costs of generating large size dynamic atoms can grow quadratically w.r.t the kernel sizes. To achieve adaptive receptive fields without additional costs, we propose to further decompose the dynamic atoms over multi-scale pre-fixed bases. Specifically, we now decompose filter atoms over S sets of pre-fixed bases, jointly denoted as $\boldsymbol{\psi} = \{\boldsymbol{\psi}_s \in \mathbb{R}^{m' \times l_s \times l_s}\}_s^S$, each set contains m' basis elements at a certain spatial scale $l_s \times l_s$. In practice, we adopt the multi-scale Fourier-Bessel bases as visualized in Appendix Figure B. As discussed in [28], Fourier-Bessel bases can effectively regularize filters and prevent learning

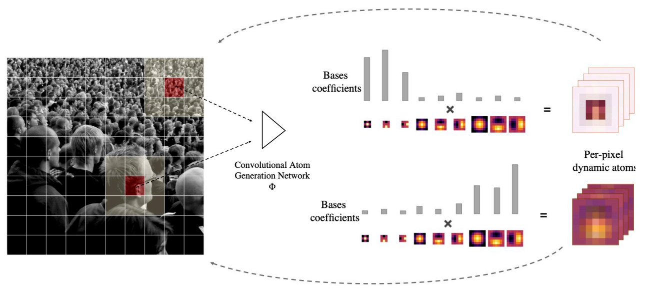


Figure 4: Atom generation with multi-scale Fourier-Bessel bases. The feature at target position (red points) and the neighborhood features (yellow points) are fed into the convolutional atom generation network Φ. At all spatial position, the basis coefficients $\boldsymbol{\alpha} \in \mathbb{R}^{m \times Sm'}$ are generated, which are multiplied with S sets of multi-scale Fourier-Bessel bases for the atoms $\mathbf{D}_{i,j} \in \mathbb{R}^{m \times l \times l}$.

high-frequency noise. As shown in Figure 4, the convolutional atom generation network Φ now outputs the basis coefficients denoted as $\boldsymbol{\alpha}_{i,j} \in \mathbb{R}^{m \times Sm'}$, which are multiplied with the sets of atom bases at different scales to reconstruct the dynamic filter atoms at each spatial position $\mathbf{D}_{i,j} = \boldsymbol{\alpha}_{i,j}\boldsymbol{\psi}$. The effective receptive field of $\mathbf{K}_{i,j}$ is now decided by the predicted basis coefficients $\boldsymbol{\alpha}_{i,j}$, which decide the weights of atom bases $\boldsymbol{\psi}$ at different scales when reconstructing dynamic filter atoms. Given the number of prefixed atom bases m', the atom bases allows filters with both different receptive fields and patterns to be applied across spatial position without increasing the cost of parameters. Meanwhile, since the patterns of the filters at scales are all regularized by $\boldsymbol{\psi}$, we consistently observe that the network can learn fast even with large scale, e.g., 7×7 filters being adaptively generated and applied.

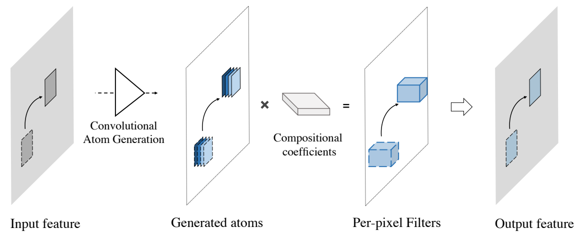
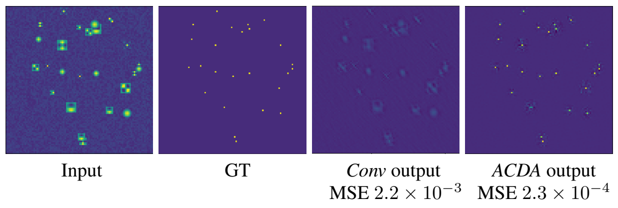


Figure 5: The translation equivariance property of the proposed method. The original and the translated position are denoted by dashed and solid lines, respectively.

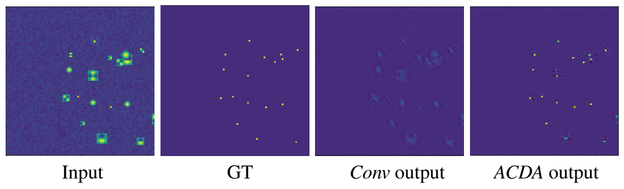
Translation equivariance. In our method, the convolutional atom generation network Φ and the compositional coefficients $\mathbf{A}$ are both shared across spatial position. The *weight sharing* in our method ensures that ACDA preserves the *translation equivariance* property of standard convolutions. As illustrated in Figure 5, a spatial translation to a local feature patch results in an equal shift to the corresponding dynamic atoms output from the convolutional atom generation network. With the shared $\mathbf{A}$, the adaptive filters and therefore the output feature patch are both exactly the spatially-shifted versions of the original spatial position.

3.3. Toy Pattern Detection Examples

To illustrate the advantages of the proposed method on handling intra-image variance and the translation-equivariant property preserved from standard convolutions, we present toy pattern detection examples as visualized in Figure 6. In Figure 6a, we synthesize a single training sample by randomly placing pre-defined multi-scale patterns on a noisy background image. Specifically, we place 25 patterns at three scales, 3×3, 5×5, and 7×7, on a 100×100 noisy map. The goal is to predict the groundtruth binary detection map as shown in Figure 6a, where only the center position of the 25 patterns are marked as 1, and 0 otherwise. We train two single-layer networks, one with a single standard convolution, denoted as *Conv*, and one with a single proposed adaptive convolution layer with dynamic atoms, denoted as *ACDA*. The kernel sizes of both layers are 7×7. Using the introduced synthesized input and groundtruth, both layers are trained with stochastic gradient decent (SGD) and mean square error (MSE) using learning rate 0.01 till converge. As visualized in Figure 6a, single standard convolutional layer is incapable of handling such diverse patterns within the image, and results in a high error rate. On the other hand, we observe that *ACDA* can fast adapt to the diverse patterns, and generate adaptive filters at each position for detection. The accurate detection by *ACDA* is reflected by both low error rate and appealing predicted map.



(a) Toy pattern detection experiment. We train a single standard *Conv* layer and a *ACDA* layer to detect diverse patterns at scales. The ACDA layer clearly outperforms the standard convolutional layer, as the detection error of ACDA is of a magnitude lower than that of the standard convolutional layer.



(b) Translation equivariance. When testing with a shifted input image, the output predictions of both layers are exactly shifted versions of the ones in Figure 6a.

Figure 6: Toy pattern detection experiments.

We then test both layers with a shifted input as shown in Figure 6b. All the patterns in the image are shifted spatially to the bottom-right direction by 20 pixels. As shown in Figure 6b, the outputs of both layers are exactly the shifted versions of those in Figure 6a. This is a clear demonstration that, thanks to the convolutional atom generation and shared coefficients, *ACDA* preserve the appealing translation-equivariance property of standard convolutions. These simple synthesized demonstrations show the flexibility and efficiency of *ACDA*. To further validate the effectiveness, we present real-world experiments in Section 4.

4. Experiments

we present experimental results to fully validate the proposed approach, referred to as ACDA, on various applications. We start with image classification experiments, showing that by plug-and-playing the introduced ACDA into CNNs, comparable and improved performance can be obtained even with reduced channel numbers. The results indicate that the dynamic filters at convolutional layers can better handle image variance, thus alleviate the demand for learning filters with many channels for exhaustive feature matching. We then move to real-world applications on crowd counting and image restorations, both of which involve significant intra-sample variance. We further demonstrate the advantages of ACDA through discussions on computation, memory, and parameters.

In all the experiments, if not otherwise specified, we adopt the same structure of atom generators as described in Appendix Section A.3, with 3 sets of Fourier Bessel bases at scales from 3×3 to 7×7. Note that although only three scales are used in each layer, the low cost allow ACDA layers to be stacked and achieve receptive fields with a very large range. E.g., stacking only two ACDA layers with three scales can achieve effective receptive fields range from 5×5 to 13×13. We provide ablation study in Appendix Section A.6 to validate the selections of hyperparameters.

Table 1: Illustrative image classification performance on CIFAR-10 and CIFAR-100. Top-1 error rates are reported.

Methods	CIFAR-10	CIFAR-100
LeNet [19]	24.74	56.60
LeNet + ACDA	**16.27** (34.2%$\downarrow$)	**49.53** (12.5%$\downarrow$)

4.1. Image Classification

Before diving into large scale experiments with customized networks, we start with a simple illustrative experiment with LeNet [19] and CIFAR. LeNet is a tiny network architecture with only two convolutional layers. We show in Table 1 that, by only replacing the two convolutional layers in LeNet with the proposed adaptive convolutional layers,

Table 2: Image classification performance on ImageNet. We report both parameter size as well as Top-1 and Top-5 error rates.

Methods	Parameters	Top-1	Top-5
ResNet-18	11.69M	30.24	10.92
ResNet-34	21.28M	26.70	8.58
ResNet-50	25.56M	23.85	7.13
MobileNet-V3 small [12]	2.9M	32.6	13.6
CondConv-EfficientNet [42]	13.3M	22.8	-
CondConv-ResNet-50 [42]	-	22.3	-
DY-MobileNetV3 small [6]	4.8M	29.7	11.3
Ad-ResNet-s	3.85M	28.81	9.62
Ad-ResNet-m	9.83M	25.81	7.92
Ad-ResNet-l	18.19M	23.22	6.74

significant accuracy improvements are observed, indicating the extra expressiveness achieved by the proposed ACDA.

We then demonstrate the effective and scalable properties of ACDA by performing experiments on ImageNet. Without heavily tuning the network structure, we construct a simple architecture based upon deep residual networks (ResNet) [11]. We empirically observe that adopting ACDA in shallow layers in a CNN does not significantly influence the network performance, therefore we leave those layers unchanged. In deep layers, we build *dynamic bottleneck blocks* following the bottleneck blocks introduced in [11]. Details on the network configurations are presented in Appendix Section A.4. We construct architectures of adaptive ResNets with different sizes and show in Table 2 that, while significantly reducing the parameters, networks with ACDA can deliver comparable performance as standard CNNs. And comparing to dynamic convolutions [6, 42], ACDA enjoys clear advantages on parameters. Although building compact networks is not our primary focus in this paper, comparisons against state-of-the-art compact network architecture MobileNet-V3 [12] show that ACDA can achieve comparable or even better performance comparing to those heavily tuned architectures.

4.2. Crowd Counting

Crowd counting, aiming at counting the total number of particular objects (typically pedestrians), poses challenges on learning based methods due to the significantly large variance on the object sappearance. We conduct experiments on crowd counting by simply adopting the networks with ACDA trained with ImageNet as feature extractor, and replace the final linear layer with few standard transposed convolutional layers for recovering resolution. We follow the simplest practice and generate groundtruth heatmaps of each object marked by a **fixed-size** Gaussian kernel, and directly train the networks with a mean square error (MSE) loss. Without bells and whistles, networks with ACDA achieve state-of-the-art results on large scale datasets, UCF-QNRF [13] and ShanghaiTech [46] subset A. Both datasets are collected from various sources so that contain significant variance reflected by large diversity of viewing angles, image qualities, and etc.

Table 3: Comparisons on large-scale crowd counting datasets. Numbers of parameters are reported in millions.

Datasets	SHTech-A		UCF-QNRF	
Metrics	MAE	MSE	MAE	MSE
MCNN [47]	110.2	173.2	277	426
Switch-CNN [30]	90.4	135.0	228	445
SCNet [40]	71.9	117.9	-	-
ic-CNN [29]	68.5	116.2	-	-
SANet [5]	67.0	104.5	-	-
CL-CNN [14]	-	-	132	191
PACNN [33]	62.4	102.0	-	-
SFCN [37] (38.60M)	64.8	107.5	102	171
CAN [22] (18.10M)	62.3	100.0	107	183
Wan *et al.* [36]	64.7	97.1	101	176
BL [23] (21.50M)	62.8	101.8	88.7	154.8
CondConv-s [42] (14.8M)	68.44	112.96	117	182
CondConv-l [42] (25.5M)	63.82	104.23	109	179
Baseline (4.78M)	67.73	110.12	124.77	210.05
Ad-ResNet-s (4.87M)	57.88	91.27	99.22	182.13
Ad-ResNet-m (11.87M)	56.04	89.76	96.08	176.87

Following the normal protocol, we report results with mean absolute error (MAE) and mean square error (MSE) in Table 3, and compare the results against various state-of-the-art methods that adopts customized network architectures [47, 30, 40, 5] and loss functions as well as training strategies [29, 14, 33, 22, 23]. The baseline performance is obtained by training a same network architecture as Ad-ResNet with standard convolutional layers only. Networks with ACDA achieve significant improvements over state-of-the-art methods on the ShanghaiTech-A dataset. To further demonstrate the advantages of per-pixel specific adaptive filters with ACDA over per-image specific adaptive filters, we adopt the official models of CondConv [42] and train them on crowd counting by appending a light weight decoder network. The results indicate that, the per-pixel adaptive filters can deliver much better performance comparing to both standard and per-image adaptive filters, with even reduced network scales. We present a visualization of basis coefficient heatmaps in Figure 7 to validate the unique advantages of ACDA on addressing scale variance. It is clearly shown when processing images with significant intra-image scale variance, ACDA can selectively decide the effective receptive fields at each position based on the target object sizes, by adjusting the weights of multi-scale atom bases. Qualitative results are in Figure 8 and Appendix Section B.

4.3. Real-world Image Restorations

In real-world image restorations, the degradation model can be highly non-uniform across spatial position of an im-

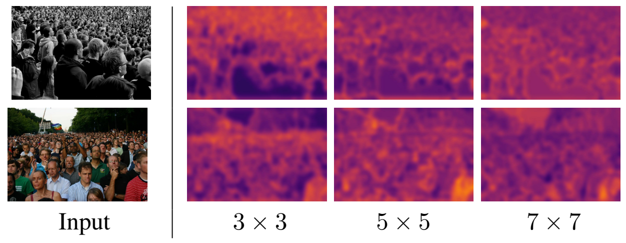


Figure 7: Visualizations of the atom basis coefficients heatmaps (lighter the higher). Our per-pixel adaptive convolutions tend to adopt large kernel sizes, i.e., with 7×7 atom bases, when the objects in the target regions have large spatial sizes, i.e., the closer objects. While 3×3 bases are preferred when targeting on regions with dense objects.

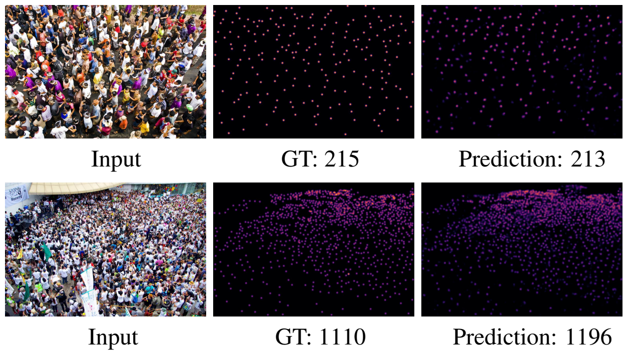


Figure 8: Qualitative results on crowd counting.

Table 4: Comparisons on the RealSR real-world super-resolution dataset. l is the kernel size.

Metrics	PSNR			SSIM		
Scales	×2	×3	×4	×2	×3	×4
RealSR 2019						
Bicubic	32.61	29.34	27.99	0.907	0.841	0.806
VDSR [17]	33.64	30.14	28.63	0.917	0.856	0.821
SRResNet [20]	33.69	30.18	28.67	0.919	0.859	0.824
RCAN [45]	33.87	30.40	28.88	0.922	0.862	0.826
KPN, $l = 5$	33.75	30.26	28.74	0.920	0.860	0.826
KPN, $l = 19$ [26]	33.86	30.39	28.90	0.924	0.864	0.830
LP-KPN, $l = 5$ [4]	33.90	30.42	28.92	0.927	0.868	0.834
ACDA	33.98	30.62	28.97	0.929	0.871	0.937
RealSR Final						
KPN, $l = 5$ [26]	33.41	30.47	28.80	0.913	0.860	0.826
KPN, $l = 19$ [26]	33.45	30.57	28.99	0.914	0.864	0.832
LP-KPN, $l = 5$ [4]	33.49	30.60	29.05	0.917	0.865	0.834
ACDA	33.54	30.73	29.28	0.918	0.868	0.836

age and locally feature dependent. We adopt ACDA to recover real-world degradation in a fully non-linear manner. We perform experiments on real-world SR dataset RealSR [4] and real-world denoising dataset SIDD [1].

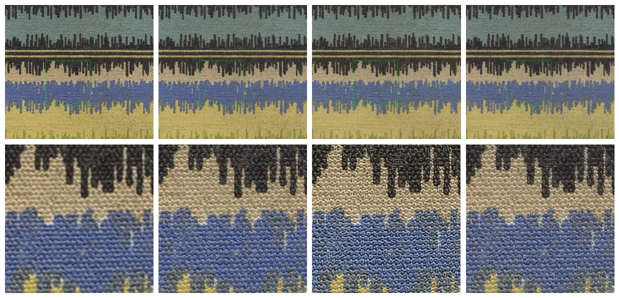


Figure 9: Qualitative comparisons against LP-KPN. LP-KPN suffers from strong artifacts. ACDA produces more faithful results. We believe the adaptive convolutions operated in deep features help better capture image semantics, thus prevent over-sharp results with strong artifacts.

Real-world single-image super resolution. We adopt an extremely simple network architecture modified from the network used in [4]. We use two convolution layers and pixel shuffle downsampling layers to reduce the feature resolution, and 8 consecutive *dynamic bottleneck block*s to process the intermediate features. Finally, two convolution layers with pixel shuffle upsampling layers are followed to restore the feature resolution and output the final predictions. Despite being extremely simple, the adopted network is shown to be effective on handling real-world SR with spatially non-uniform and potentially feature dependent degradation models. We show comparisons on two versions of the RealSR [4] dataset, *RealSR 2019* , and a larger *RealSR final*. We compare ACDA against state-of-the-art methods using standard convolutions [17, 20, 45] and methods that adopt simplified adaptive convolutions [4, 26]. We follow the standard practice in [4] and train the network with random cropped image patches, and simple mean squared error (MSE) as the loss function. The results and comparisons are presented in Table 4. Following [21, 4, 45], we use PSNR and SSIM [38] indices on the Y channel in the YCbCr space as the metrics of performance evaluations. While methods based on simplified adaptive convolutions achieve good results on restoring non-uniformed degradation, they use only combinations of linear corrections. We show that endowing networks with local adaptive filters in a fully non-linear way can further boost performance. The experimental results and our methods shed the light on future real-world SR methods that rely on little assumptions on local degradation. Qualitative results and comparisons are shown in Figure 9 and Appendix Section C.

Real-world image denoising. We then perform real-world image denoising with the SIDD [1] dataset. Similar to real-world SR, real-world denoising aims at restoring high-quality images given noisy inputs with spatially non-uniform non-i.i.d. real-world noise. The noise can be locally feature-dependent, thus adaptive convolutions a naturally good tool for modeling the intra-image variance. We present the quantitative results in Table 5. Qualitative comparisons are presented in Appendix Section D. Baseline performance is obtained by using the same network architecture as ACDA with standard convolutions only. As shown in Table 5, although the dynamic atom generation network introduces additional parameters, the overall size is smaller in ACDA comparing to the baseline thanks to the atom bases decomposition. We present results on ACDA by using both simple MSE training loss (ACDA + MSE) and the state-of-the-art variational denoising framework proposed in VDNet[43] (ACDA + VDNet). ACDA delivers improvements over baseline models with fewer parameters.

Table 5: Comparisons on the RealSR real-world super-resolution dataset.

Metrics	PSNR	SIDD
DnCNN-B [44]	38.41	0.909
CBDNet [9]	38.68	0.901
VDNet [43] with UNet (7.70M)	39.28	0.909
Baseline + MSE (2.28M)	38.74	0.902
ACDA + MSE (1.97M)	38.96	0.905
ACDA + VDNet (1.97M)	39.32	0.912

4.4. Discussions on Efficiency

Computation and parameter. A regular convolution needs $c'hw \cdot c(1+l^2)$ FLOPS, while *ACDA* needs $\underbrace{c'm(1+l^2)}_{\text{atom conv}} + \underbrace{c'mc}_{\text{coefficient}} + \underbrace{c'hwd(1+l_a^2) + dhwSm'(1+l_b^2)}_{\text{atom generation}}$, where $d = 64$, $l_a = 1$, and $l_b = 3$ following Section A.4. For more straightforward comparisons, we present in Table 6 comparisons on parameter size and FLOPs between standard convolution (denoted as Conv) and ACDA. The numbers are obtained by calculating the parameters and computation in a single layer with a typical setting: 256 input and output channels, and 100×100 feature resolution. We report comparisons with three kernel sizes from 3×3 to 7×7. As shown in Table 6, ACDA have clear advantages on both parameter size and computation in all settings. When using large kernel sizes, the advantages become more superior thanks to the atom bases. The comparisons between the numbers of ACDA (conv only) and ACDA (+ atom generation) show that the atom generation only introduces small overhead under the typical settings.

Memory and speed. We then perform comparisons on memory and speed. Ad-ResNet-s, and construct baseline (denoted as Conv) by using standard convolutions only, thus Conv and ACDA in Table 7 have the same architectures. We train the networks on ImageNet with standard settings, and report the training memory consumption and speed of one iteration. The proposed ACDA achieves higher speed without significantly increase the memory footprints comparing to standard convolutions. We further present a comparison by adopting per-pixel dynamic filter networks (DFN [16]), where a light-weight network is used to generation adaptive filters that are directly applied to the feature maps. In practice, training DFN consistently results in out-of-memory (OOM) error, and similar impractical costs are also observed when using CondConv [42] and DY-CNN [6] for per-pixel adaptive filters. The results indicate that, without the proposed two-layer implementation in our ACDA framework, applying pixel-wise adaptive convolutions is prohibitive as it involves the multiplications between very high dimensional tensors. The proposed ACDA successfully addresses this challenge by decomposing the prohibitive multiplication into two mild-size multiplications as quantitatively validated in Table 7. And a faster speed is observed thanks to the reduced computation.

Table 6: Comparisons between standard convolution (Conv) and ACDA on computation (FLOPs) and parameters (Params). All numbers are reported in millions. ACDA (conv only) denotes the two stage convolutions only, and ACDA (+ atom generation) denote the entire process of atom generations and convolutions.

	Kernel size	3×3	5×5	7×7
FLOPs	Conv	5,900.8	16,386.6	32,115.2
	ACDA (conv only)	4,073.0	4,318.7	4,687.4
	ACDA (+ atom generation)	4,311.2	4,557.0	4,925.7
Params	Conv	0.59	1.64	3.12
	ACDA (conv only)	0.39	0.39	0.39
	ACDA (+ atom generation)	0.41	0.43	0.45

Table 7: Comparisons on training memory and time.

Methods	Params	Memory	Time
Conv	4.60M	3.7GB $\times$ 4	0.53s
DFN	2.32M	OOM	-
ACDA (two-layer implementation)	2.28M	4.6GB $\times$ 4	0.46s

5. Conclusion

In this paper, we introduced adaptive convolutions with dynamic filter atoms, plug-and-play replacements to convolution layers to better model intra-image variance. The convolutional filters in ACDA are adaptively generated from local feature. We decomposed adaptive filters over dynamically generated atoms to significantly save in parameter and memory. We further decomposed atoms over multi-scale bases for adaptive receptive fields. We empirically validated our approach on image classification, crowd counting, and real-world image restorations.

6. Acknowledgements

This work was supported by the DARPA TAMI program.

References

[1] Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. A high-quality denoising dataset for smartphone cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1692–1700, 2018. 7

[2] Jimmy Ba, Geoffrey Hinton, Volodymyr Mnih, Joel Z Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. *Advances in neural information processing systems*, 2016. 1

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013. 2

[4] Jianrui Cai, Hui Zeng, Hongwei Yong, Zisheng Cao, and Lei Zhang. Toward real-world single image super-resolution: A new benchmark and a new model. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3086–3095, 2019. 2, 7

[5] Xinkun Cao, Zhipeng Wang, Yanyun Zhao, and Fei Su. Scale aggregation network for accurate and efficient crowd counting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, 2018. 6

[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. 1, 2, 3, 6, 8

[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. 2

[8] Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard E Turner. Meta-learning probabilistic inference for prediction. *arXiv preprint arXiv:1805.09921*, 2018. 2

[9] Shi Guo, Zifei Yan, Kai Zhang, Wangmeng Zuo, and Lei Zhang. Toward convolutional blind denoising of real photographs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1712–1722, 2019. 8

[10] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 2

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6, 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 International Conference on Computer Vision*, pages 1314–1324, 2019. 6

[13] Haroon Idrees, Muhmmad Tayyab, Kishan Athrey, Dong Zhang, Somaya Al-Maadeed, Nasir Rajpoot, and Mubarak Shah. Composition loss for counting, density map estimation and localization in dense crowds. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–546, 2018. 6

[14] Haroon Idrees, Muhmmad Tayyab, Kishan Athrey, Dong Zhang, Somaya Al-Maadeed, Nasir Rajpoot, and Mubarak Shah. Composition loss for counting, density map estimation and localization in dense crowds. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–546, 2018. 6

[15] Xu Jia, Hong Chang, and Tinne Tuytelaars. Super-resolution with deep adaptive image resampling. *arXiv preprint arXiv:1712.06463*, 2017. 2

[16] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *Advances in neural information processing systems*, pages 667–675, 2016. 1, 2, 3, 8

[17] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1646–1654, 2016. 7

[18] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 5

[20] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017. 7

[21] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 136–144, 2017. 7

[22] Weizhe Liu, Mathieu Salzmann, and Pascal Fua. Context-aware crowd counting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5099–5108, 2019. 6

[23] Zhiheng Ma, Xing Wei, Xiaopeng Hong, and Yihong Gong. Bayesian loss for crowd count estimation with point supervision. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6142–6151, 2019. 6

[24] Thomas Miconi, Aditya Rawal, Jeff Clune, and Kenneth O Stanley. Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity. *International Conference on Learning Representations*, 2019. 1

[25] Thomas Miconi, Kenneth Stanley, and Jeff Clune. Differentiable plasticity: training plastic neural networks with backpropagation. In *International Conference on Machine Learning*, pages 3559–3568. PMLR, 2018. 1

[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. 2, 7

[27] Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan L Yuille. Few-shot image recognition by predicting parameters from activations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7229–7238, 2018. 2

[28] Qiang Qiu, Xiuyuan Cheng, Robert Calderbank, and Guillermo Sapiro. DCFNet: Deep neural network with decomposed convolutional filters. *International Conference on Machine Learning*, 2018. 3, 4

[29] Viresh Ranjan, Hieu Le, and Minh Hoai. Iterative crowd counting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 270–285, 2018. 6

[30] Deepak Babu Sam, Shiv Surya, and R Venkatesh Babu. Switching convolutional neural network for crowd counting. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4031–4039. IEEE, 2017. 6

[31] Imanol Schlag, Tsendsuren Munkhdalai, and Jürgen Schmidhuber. Learning associative inference using fast weight memory. In *International Conference on Learning Representations*, 2021. 1

[32] Imanol Schlag and Jürgen Schmidhuber. Gated fast weights for on-the-fly neural program generation. In *NeurIPS Metalearning Workshop*, 2017. 1

[33] Miaojing Shi, Zhaohui Yang, Chao Xu, and Qijun Chen. Revisiting perspective information for efficient crowd counting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7279–7288, 2019. 6

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

[35] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[36] Jia Wan and Antoni Chan. Adaptive density map generation for crowd counting. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1130–1139, 2019. 6

[37] Qi Wang, Junyu Gao, Wei Lin, and Yuan Yuan. Learning from synthetic data for crowd counting in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8198–8207, 2019. 6

[38] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 7

[39] Ze Wang, Xiuyuan Cheng, Guillermo Sapiro, and Qiang Qiu. Stochastic conditional generative networks with basis decomposition. In *ICLR*, 2020. 2

[40] Ze Wang, Zehao Xiao, Kai Xie, Qiang Qiu, Xiantong Zhen, and Xianbin Cao. In defense of single-column networks for crowd counting. *arXiv preprint arXiv:1808.06133*, 2018. 6

[41] 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 Conference on Computer Vision and Pattern Recognition*, pages 12496–12505, 2020. 2

[42] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. *Advances in neural information processing systems*, 2019. 1, 2, 3, 6, 8

[43] Zongsheng Yue, Hongwei Yong, Qian Zhao, Deyu Meng, and Lei Zhang. Variational denoising network: Toward blind noise modeling and removal. In *Advances in neural information processing systems*, pages 1690–1701, 2019. 8

[44] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017. 8

[45] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 286–301, 2018. 7

[46] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 589–597, 2016. 6

[47] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 589–597, 2016. 6

[48] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9308–9316, 2019. 2