ARMA Nets: Expanding Receptive Field for Dense Prediction

Jiahao Su[1] jiahaosu@umd.edu
Shiqi Wang[2] 161170041@smail.nju.edu.cn
Furong Huang[1] furongh@cs.umd.edu
[1]University of Maryland, College Park, MD USA [2]Nanjing University, Nanjing, China

Abstract

Global information is essential for dense prediction problems, whose goal is to compute a discrete or continuous label for each pixel in the images. Traditional convolutional layers in neural networks, initially designed for image classification, are restrictive in these problems since the filter size limits their receptive fields. In this work, we propose to replace any traditional convolutional layer with an autoregressive moving-average (ARMA) layer, a novel module with an adjustable receptive field controlled by the learnable autoregressive coefficients. Compared with traditional convolutional layers, our ARMA layer enables explicit interconnections of the output neurons and learns its receptive field by adapting the autoregressive coefficients of the interconnections. ARMA layer is adjustable to different types of tasks: for tasks where global information is crucial, it is capable of learning relatively large autoregressive coefficients to allow for an output neuron's receptive field covering the entire input; for tasks where only local information is required, it can learn small or near zero autoregressive coefficients and automatically reduces to a traditional convolutional layer. We show both theoretically and empirically that the effective receptive field of networks with ARMA layers (named ARMA networks) expands with larger autoregressive coefficients. We also provably solve the instability problem of learning and prediction in the ARMA layer through a re-parameterization mechanism. Additionally, we demonstrate that ARMA networks substantially improve their baselines on challenging dense prediction tasks, including video prediction and semantic segmentation. Our code is available on `https://github.com/umd-huang-lab/ARMA-Networks`.

1 Introduction

Convolutional layers in neural networks have many successful applications for machine learning tasks. Each output neuron encodes an input region of the network measured by the *effective receptive field* (ERF) [25]. A large ERF that allows for sufficient global information is needed to make accurate predictions; however, a simple stack of convolutional layers does not effectively expand ERF. Convolutional neural networks (CNNs) typically encode global information by adding downsampling (pooling) layers, which coarsely aggregate global information. A fully-connected classification layer subsequently reduces the entire feature map to an output label. Downsampling and fully-connected layers are suitable for image classification tasks where only a single prediction is needed. But they are less effective, due to potential loss of information, in dense prediction tasks such as semantic segmentation and video prediction, where each pixel requests a prediction. Therefore, it is crucial to introduce mechanisms that enlarge ERF without too much information loss.

Naive approaches to expanding ERF, such as deepening the network or enlarging the filter size, drastically increase the model complexity, which results in expensive computation, difficulty in optimization, and susceptibility to overfitting. Advanced architectures have been proposed to expand ERF, including encoder-decoder networks [30], dilated convolutional networks [40, 41], and non-local networks [34]. However, encoder-decoder networks could lose high-frequency information due to the downsampling layers. Dilated convolutional networks could suffer from the gridding effect while the ERF expansion is limited, and non-local networks are expensive in training and inference.

We introduce a novel *autoregressive-moving-average* (ARMA) layer that enables adaptive receptive field by explicit interconnections among its output neurons. Our ARMA layer realizes these interconnections via extra convolutions on output neurons, on top of the convolutions on input neurons as in a traditional convolutional layer. We provably show that an ARMA network can have arbitrarily large ERF, thus capturing global information, with minimal extra parameters at each layer. Consequently, an ARMA network can flexibly enlarge its ERF to leverage global knowledge without reducing spatial resolution. Moreover, the ARMA networks are independent of the architectures above, including encoder-decoder networks, dilated convolutional networks, and non-local networks.

A significant challenge in ARMA networks lies in the complex computations needed in both forward and backward propagations — simple convolution operations are not applicable since the output neurons are influenced by their neighbors and thus interrelated. Another challenge in ARMA networks is instability — the additional interconnections among the output neurons could recursively amplify the outputs and lead them to infinity. We address both challenges in this paper.

Summary of Contributions

- We introduce a novel ARMA layer that is a plug-and-play module substituting convolution layers in neural networks to allow flexible tuning of their ERF, adapting to the task requirements, and improving performance in dense prediction problems.
- We recognize and address the problems of *computation* and *instability* in ARMA layers. **(1)** To reduce computational complexity, we develop FFT-based algorithms for both forward and backward passes; **(2)** To guarantee stable learning and prediction, we propose a *separable ARMA layer* and a re-parameterization mechanism that ensures the layer to operate in a stable region.
- We successfully apply ARMA layers in ConvLSTM network [39] for pixel-level multi-frame video prediction and U-Net model [30] for medical image segmentation. ARMA networks substantially outperform the corresponding baselines on both tasks, suggesting that our proposed ARMA layer is a general and useful building block for dense prediction problems.

2 Related Works

Dilated convolution [15] enlarges the receptive field by upsampling the filter coefficients with zeros. Unlike encoder-decoder structure, dilated convolution preserves the spatial resolution and is thus widely used in dense prediction problems, including semantic segmentation [7, 24, 40], and objection detection [10, 20]. However, dilated convolution by itself creates gridding artifacts if its input contains higher frequency than the upsampling rate [41], and the inconsistency of local information hampers the performance of the dilated convolutional networks [35]. Such artifacts can be alleviated by extra anti-aliasing layer [41], group interacting layer [35] or spatial pyramid pooling [8].

Deformable convolution allows the filter shape (i.e., locations of the incoming pixels) to be learnable [11, 16, 42]. While deformable convolution focuses on adjusting the filter *shape*, our ARMA layer aims to expand the effective filter *size* adaptively.

Non-local attention network [34] inserts non-local attention blocks between the convolutional layers. A non-local attention block computes a weighted sum of all input neurons for each output neuron, similar to attention mechanism [33]. In practice, non-local attention blocks are computationally expensive, thus they are typically inserted in the upper part of the network (with lower resolution). In contrast, our ARMA layers are economical (see section 4) and can be used throughout the network.

Encoder-decoder structured network pairs each downsampling layer with another upsampling layer to maintain the resolution, and introduces skip-connection between the pair to preserve the high-frequency information [24, 30]. Since the shortcut bypasses the downsampling/upsampling layers, the network has a small receptive field for the high-frequency components. A potential solution is to augment upsampling with non-local attention block [27] or ARMA layer (section 6).

Recurrent neural networks over the spatial domain [5, 17, 22, 23, 28, 32] are used to expand the receptive field or learn the affinity between neighboring pixels. However, almost all prior works consider nonlinear recurrent neural networks, where the activation in recursions prohibits an efficient parallel algorithm. Quasi-recurrent neural networks [4] partially address the problem by decoupling the linear operations and parallelizing them using convolutions. In contrast, our proposed ARMA layer is equivalent to a linear recurrent neural network, allowing for efficient evaluation using FFT.

3 ARMA Neural Networks

In this section, we introduce a novel *autoregressive-moving-average* (ARMA) layer and analyze its ability to expand *Effective Receptive Field* (ERF) in neural networks. The analysis is further verified by visualizing the ERF with varying network depth and strength of autoregressive coefficients.

3.1 ARMA Layer

A traditional convolutional layer is essentially a *moving-average* model [3]:

$$\mathcal{Y}_{:,:,t} = \sum_{s=1}^{S} \mathcal{W}_{:,:,t,s} * \mathcal{X}_{:,:,s}, \tag{1}$$

where the *moving-average coefficients* $\mathcal{W} \in \mathbb{R}^{K_m \times K_m \times T \times S}$, is parameterized by a 4th-order kernel (K_m is the filter size, and S, T are input/output channels), **:** denotes all elements from the specified coordinate and $*$ denotes convolution between an input feature and a filter.

As motivated in the introduction, we introduce a novel ARMA layer that enables an adaptive receptive field by introducing explicit interconnections among its output neurons, as illustrated in Figure 1. Our ARMA layer realizes these interconnections by introducing extra convolutions on the outputs, upon the convolutions on the inputs as in a traditional convolutional layer. As a result, in an ARMA layer, each output neuron can be affected by an input pixel faraway through interconnections among the output neurons, thus receives global information. Formally, we define an ARMA layer in Definition 1.

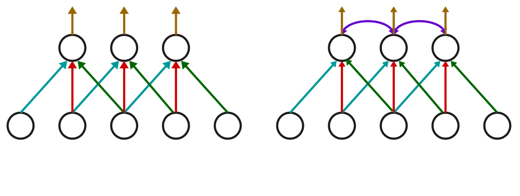
(a) Convolution **(b)** ARMA

Figure 1: The ARMA layer introduces interconnections among output neurons explicitly.

Definition 1 (ARMA Layer). *An ARMA layer is parameterized by a moving-average kernel (coefficients) $\mathcal{W} \in \mathbb{R}^{K_m \times K_m \times S \times T}$ and an autoregressive kernel (coefficients) $\mathcal{A} \in \mathbb{R}^{K_a \times K_a \times T}$. It receives an input $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times S}$ and returns an output $\mathcal{Y} \in \mathbb{R}^{I_1' \times I_2' \times T}$ with an ARMA model:*

$$\mathcal{A}_{:,:,t} * \mathcal{Y}_{:,:,t} = \sum_{s=1}^{S} \mathcal{W}_{:,:,t,s} * \mathcal{X}_{:,:,s}. \tag{2}$$

Remarks: **(1)** The ARMA layer maintains the *shift-invariant* property, since the output interconnections are realized by convolutions. **(2)** The ARMA layer *reduces* to a traditional layer if the autoregressive kernel $\mathcal{A}$ represents an identical mapping. **(3)** The ARMA layer is a *plug-and-play* module that can replace *any* convolutional layer, adding $K_a^2 T$ extra parameters negligible compared to $K_w^2 ST$ parameters in a traditional convolution layer. **(4)** Unlike a traditional layer, computing Equation 2 and its backpropagation are nontrivial, studied in section 4.

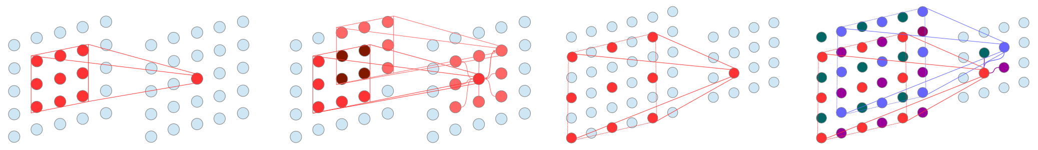
(a) Convolution. **(b)** ARMA. **(c)** Dilated convolution. **(d)** Dilated ARMA.

Figure 2: Diagrams of receptive field. In **(b)**, each output neuron receives its neighbors' receptive field. In **(d)**, ARMA's autoregression fills the gaps created by dilated convolution.

Our ARMA layer can be combined with the methods of dilated convolutional layer, deformable convolutional layer, non-local attention block, as well as encoder-decoder architecture. For instance, a *dilated ARMA layer*, illustrated in Figure 2d, removes the gridding effect caused by dilated convolution — the autoregressive kernel is interpreted as an anti-aliasing filter.

The motivation of introducing the ARMA layer is to enlarge the effective input region for each network output without increasing the filter size or network depth, thus avoiding the difficulties in training larger or deeper models. As illustrated in Figure 2, each output neuron in a traditional convolutional layer (Figure 2a) only receives information from a small input region (the filter size). However, an ARMA layer enlarges the small local region to a larger one (Figure 2b). It enables an output neuron to receive information from a faraway input neuron through the connections to its neighbors. In subsection 3.2, we formally introduce the concept of *effective receptive field* (ERF) to characterize the input region size. Moreover, we will show that an ARMA network can have arbitrarily large ERF with a single extra parameter at each layer in Theorem 3.

3.2 Effective Receptive Field

Effective receptive field (ERF) [25] measures the area of the input region that makes *substantial* contribution to an output neuron. In this section, we analyze the ERF size of an L-layers network with ARMA layers v.s. traditional convolutional layers. Formally, consider an output at location (i_1, i_2), the impact from an input pixel at $(i_1 - p_1, i_2 - p_2)$ (i.e L layers and (p_1, p_2) pixels away) is measured by the amplitude of partial derivative $g(i_1, i_2, p_1, p_2) = \left| \partial \mathcal{Y}^{(L)}_{i_1, i_2, t} / \partial \mathcal{X}^{(1)}_{i_1 - p_1, i_2 - p_2, s} \right|$ (where superscripts index the layers), i.e., how much the output changes as the input pixel is perturbed.

Definition 2 (Effective Receptive Field, ERF). *Consider an L-layers network with an S-channels input $\mathcal{X}^{(1)} \in \mathbb{R}^{I_1 \times I_2 \times S}$ and a T-channels output $\mathcal{Y}^{(L)} \in \mathbb{R}^{I_1 \times I_2 \times T}$, its effective receptive field is defined as the empirical distribution of the gradient maps: ERF$(p_1, p_2) = 1/(I_1 I_2 S T) \cdot \sum_{s,t,i_1,i_2}$ $[g(i_1, i_2, p_1, p_2) / \sum_{j_1, j_2} g(j_1, j_2, p_1, p_2)]$, To measure the size of the ERF, we define its radius $r(\mathit{ERF})$ as the standard deviation of the empirical distribution:*

$$r\left(\mathit{ERF}\right)^2 = \sum_{p_1, p_2} \left(p_1^2 + p_2^2\right) \mathit{ERF}(p_1, p_2) - \left[\sum_{p_1, p_2} \sqrt{p_1^2 + p_2^2}\, \mathit{ERF}(p_1, p_2) \right]^2. \tag{3}$$

Notice that ERF simultaneously depends on the model parameters and a specified input to the network, i.e., ERF is both *model-dependent* and *data-dependent*. Therefore, it is generally intractable to compute the ERF analytically for any practical neural network.

We follow the original paper of ERF [25] to estimate the radius with a simplified linear network. The paper empirically verifies that such an estimation is accurate and can be used to guide filter designs.

Theorem 3 (ERF of a Linear ARMA Network). *Consider an L-layers linear network, where the ℓ^{th} layer computes $y^{(\ell)}_i - a^{(\ell)} y^{(\ell)}_{i-1} = \sum_{p=0}^{K^{(\ell)}-1} [(1 - a^{(\ell)}) / K^{(\ell)}] \cdot y^{(\ell-1)}_{i - d^{(\ell)} p}$ (i.e., the moving-average coefficients are uniform with length $K^{(\ell)}$ and dilation $d^{(\ell)}$, and the autoregressive coefficients $\boldsymbol{a}^{(\ell)} = \{1, -a^{(\ell)}\}$ has length 2). Suppose $0 \leq a^{(\ell)} < 1, \forall \ell \in [L]$, the ERF radius of the network is*

$$r(\mathit{ERF})^2_{\mathit{ARMA}} = \sum_{\ell=1}^{L} \left[\frac{d^{(\ell)2} \left(K^{(\ell)2} - 1\right)}{12} + \frac{a^{(\ell)}}{(1 - a^{(\ell)})^2} \right]. \tag{4}$$

We prove Theorem 3 in Appendix B. If the coefficients for different layers are identical, e.g., $K^{(\ell)} = K, d^{(\ell)} = d, a^{(\ell)} = a$, the radius reduces to $r(\text{ERF})_{\text{ARMA}} = \sqrt{L} \cdot \sqrt{d^2(K^2 - 1)/12 + a/(1 - a)^2}$. Moreover, if $a = 0$ and $d = 1$, the ARMA layers reduce to traditional convolutional layers, and the ERF of the resulted linear CNN has radius $r(\text{ERF})_{\text{CNN}} = \sqrt{L} \cdot \sqrt{(K^2 - 1)/12}$ as shown in [25].

Remarks: **(1) Compared with a (dilated) CNN, an ARMA network can have arbitrarily large ERF with an extra parameter a at each layer.** When the autoregressive coefficient a is large (e.g., $a > 1 - 1/(dK)$), the second term $a/(1 - a)^2$ dominates the radius, and the ERF is substantially larger than that of a CNN. In particular, the radius tends to infinity as a approaches 1. **(2) An ARMA network can adaptively adjust its ERF through learnable parameter a.** As a gets smaller (e.g., $a < 1 - 1/(dK)$), the second term is comparable to or smaller than the first term, and the effect of expanded ERF diminishes. In particular, if $a = 0$, an ARMA network reduces to a CNN.

Visualization of the ERF. We analytically show in Theorem 3 that the ERF radius increases with the network depth and autoregressive coefficients' magnitude. We now verify our analysis by simulating linear ARMA networks with varying depths and autoregressive coefficients' magnitude. As shown in Figure 3, the ERF radius increases as the autoregressive coefficients get larger. When the autoregressive coefficients are zeros, an ARMA network reduces to a traditional convolutional network. The simulation results also indicate that ARMA's ability to expand the ERF increases with the network depth. In conclusion, an ARMA network can have a large ERF even when the network is shallow, and its ability to expand the ERF increases when the network gets deeper.

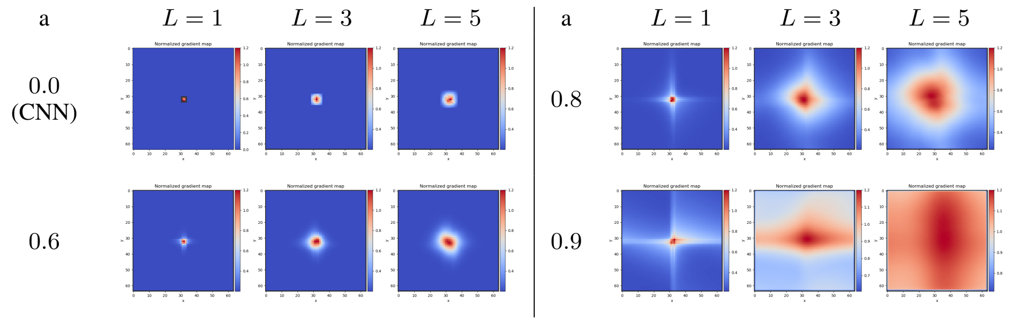

Figure 3: Visualization of ERF in linear ARMA networks under different network depth $L = 1, 3, 5$ and different magnitude of the autoregressive coefficient $a = 0.0, 0.6, 0.8, 0.9$ (See subsection A.1 for detailed experimental setup.)

4 Prediction and Learning of ARMA Layers

In an ARMA layer, each neuron is influenced by its neighbors from all directions (see Figure 2b). As a result, no neuron could be evaluated alone before evaluating any other neighboring neurons. To compute Equation 2, we need to solve a set of linear equations to obtain all values simultaneously. **(1)** However, the standard solver using Gaussian elimination is too expensive to be practical, and therefore we need to seek a more efficient solution. **(2)** Furthermore, the solver for linear equations typically does not support automatic differetiation, and we have to derive the backward equations analytically. **(3)** Finally, we also need to devise an efficient algorithm to compute the backpropagation equations efficiently. In the section, we address these aforementioned problems.

Decomposing an ARMA Layer. We decompose the ARMA layer in Equation 2 into a moving-average (MA) layer and an autoregressive (AR) layer:

$$\text{MA Layer: } \mathcal{T}_{:,:,t} = \sum_{s=1}^{S} \mathcal{W}_{:,:,t,s} * \mathcal{X}_{:,:,s}; \tag{5a}$$

$$\text{AR Layer: } \mathcal{A}_{:,:,t} * \mathcal{Y}_{:,:,t} = \mathcal{T}_{:,:,t}, \tag{5b}$$

where $\mathcal{T} \in \mathbb{R}^{I_1' \times I_2' \times T}$ is the intermediate result.

Difficulty in Computing the AR Layer. While the MA layer in Equation 5a is simply a traditional convolutional layer (Equation 1), it is nontrivial to solve the AR layer in Equation 5b. Naively using Gaussian elimination, the linear equations in the AR layer can be solved in time cubic in dimension $O((I_1^2 + I_2^2)I_1 I_2 T)$, which is too expensive.

Layer	# params.	# FLOPs	$r(ERF)^2$
Conv.	$K_w^2 C^2$	$O(I^2 K_w^2 C^2)$	$O(LK_w^2)$
ARMA	$K_w^2 C^2 + K_a^2 C$	$O(K_w^2 I^2 C^2 + I^2 \log(I)\, C)$	$O\left(LK_w^2 + L\frac{a}{(1-a)^2}\right)$

Table 1: An ARMA layer achieves large gain of the ERF radius through small overhead of extra # of parameters and # of FLOPs. Through a single extra parameter a (thus $K_a = 2$), the ERF radius can be arbitrarily large. For notational simplicity, we assume all heights and widths are equal $I_1 = I_2 = I_1' = I_2' = I$, and the input and output channels are the same $S = T = C$.

Solving the AR Layer. We propose to use the frequency-domain division [21] to solve the *deconvolution* problem in the AR layer. Since the convolution in the spatial domain leads to an element-wise product in the frequency domain, we first transform $\mathcal{A}, \mathcal{T}$ into their frequency representations $\widetilde{\mathcal{A}}, \widetilde{\mathcal{T}}$, with which we compute $\widetilde{\mathcal{Y}}$ (the frequency representation of $\mathcal{Y}$) with the element-wise division. Then, we reconstruct the output $\mathcal{Y}$ by an inverse Fourier transform of $\widetilde{\mathcal{Y}}$.

Computational Overhead. ARMA trades small overhead of extra parameters and computation for a large gain of ERF radius, as shown in Table 1. With *Fast Fourier Transform* (FFT), the FLOPS required by the extra autoregressive layer is $O(\log(\max(I_1, I_2))I_1 I_2 T)$ (see Appendix C for derivations). Importantly, compared with non-local attention block [34], the extra computation introduced in an ARMA layer is smaller; a non-local attention block requires $O(I_1^2 I_2^2 T)$ FLOPS.

Backpropagation. Deriving the backpropagation for an ARMA layer is nontrivial; although the backpropagation rule for the MA layer is conventional, that of the AR layer is not. In Theorem 4, we show that the backpropagation of an AR layer can be computed as two ARMA models.

Theorem 4 (Backpropagation of an AR layer). *Given $\mathcal{A}_{:,:,t} * \mathcal{Y}_{:,:,t} = \mathcal{T}_{:,:,t}$ and the gradient $\partial\mathcal{L}/\partial\mathcal{Y}$, the gradients $\{\partial\mathcal{L}/\partial\mathcal{A}, \partial\mathcal{L}/\partial\mathcal{X}\}$ can be obtained by two ARMA models:*

$$\mathcal{A}^{\top}_{:,:,t} * \frac{\partial\mathcal{L}}{\partial\mathcal{A}_{:,:,t}} = -\mathcal{Y}^{\top}_{:,:,t} * \frac{\partial\mathcal{L}}{\partial\mathcal{Y}_{:,:,t}}; \tag{6a}$$

$$\mathcal{A}^{\top}_{:,:,t} * \frac{\partial\mathcal{L}}{\partial\mathcal{T}_{:,:,t}} = \frac{\partial\mathcal{L}}{\partial\mathcal{Y}_{:,:,t}}. \tag{6b}$$

where $\mathcal{A}^{\top}_{:,:,t}$ and $\mathcal{Y}^{\top}_{:,:,t}$ are the transposed images of $\mathcal{A}_{:,:,t}$ and $\mathcal{Y}_{:,:,t}$ (e.g., $\mathcal{A}^{\top}_{i_1,i_2,t} = \mathcal{A}_{-i_1,-i_2,t}$).

Since the backpropagation is characterized by ARMA models, it can be evaluated efficiently using FFT similar to Equation 5. The proof of Theorem 4, with its FFT evaluation, is given in Appendix C.

5 Stability of ARMA Layers

An ARMA model with arbitrary coefficients is not always stable. For example, the model $y_i - a y_{i-1} = x_i$ is unstable if $|a| > 1$: Consider an input $\boldsymbol{x}$ with $x_0 = 1$ and $x_i = 0, \forall i \neq 0$, the output $\boldsymbol{y}$ will recursively amplify itself as $y_0 = 1, y_1 = a, \cdots, y_i = a^i$ and diverge to infinity.

5.1 Stability Constraints for an ARMA Layer

The key to guaranteeing the stability of an ARMA layer is to constrain its autoregressive coefficients, which prevents the output from repeatedly amplifying itself. To derive the constraints, we propose a special design, *separable ARMA layer* inspired by *separable filters* [21].

Definition 5 (Separable ARMA Layer). *A separable ARMA layer is parameterized by a moving-average kernel $\mathcal{W} \in \mathbb{R}^{K_w \times K_w \times S \times T}$ and $T \times Q$ sets of autoregressive filters $\{(f^{(q)}_{:,t}, g^{(q)}_{:,t})_{q=1}^{Q}\}_{t=1}^{T}$. It takes an input $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times S}$ and returns an output $\mathcal{Y} \in \mathbb{R}^{I'_1 \times I'_2 \times T}$ as*

$$\left(f^{(1)}_{:,t} * \cdots * f^{(Q)}_{:,t}\right) \otimes \left(g^{(1)}_{:,t} * \cdots * g^{(Q)}_{:,t}\right) * \mathcal{Y}_{:,:,t} = \sum_{s=1}^{S} \mathcal{W}_{:,:,t,s} * \mathcal{X}_{:,:,s} \tag{7}$$

where the filters $f^{(q)}_{:,t}, g^{(q)}_{:,t} \in \mathbb{R}^3$ are length-3, and $\otimes$ denotes outer product of two 1D-filters.

Remarks: Each autoregressive filter $\mathcal{A}_{:,:,t}$ is designed to be separable, i.e., $\mathcal{A}_{:,:,t} = F_{:,t} \otimes G_{:,t}$, thus it can be characterized by 1D-filters $F_{:,t}$ and $G_{:,t}$. By the fundamental theorem of algebra [29], any 1D-filter can be represented as a composition of length-3 filters. Therefore, $F_{:,t}$ and $G_{:,t}$ can further be factorized as $F_{:,t} = f^{(1)}_{:,t} * f^{(2)}_{:,t} \cdots * f^{(Q)}_{:,t}$ and $G_{:,t} = g^{(1)}_{:,t} * g^{(2)}_{:,t} \cdots * g^{(Q)}_{:,t}$. In summary, each $\mathcal{A}_{:,:,t}$ is characterized by Q sets of length-3 autoregressive filters $(f^{(q)}_{:,t}, g^{(q)}_{:,t})_{q=1}^{Q}$.

Theorem 6 (Constraints for a Stable Separable ARMA Layer). *A sufficient condition for the separable ARMA layer (Definition 5) to be stable (i.e., output be bounded for any bounded input) is:*

$$\left|f^{(q)}_{-1,t} + f^{(q)}_{1,t}\right| < f^{(q)}_{0,t}, \ \left|g^{(q)}_{-1,t} + g^{(q)}_{1,t}\right| < g^{(q)}_{0,t}, \ \forall q \in [Q], t \in [T]. \tag{8}$$

The proof is deferred to Appendix D, which follows the standard techniques using Z-transform.

5.2 Achieving Stability via Re-parameterization

In principle, the constraints required for stability in an ARMA layer (as in Theorem 6) could be enforced through constraints in optimization. However, a constrained optimization algorithm, such as

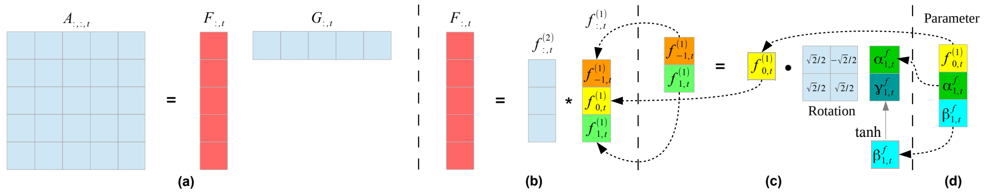

Figure 4: For each channel t, **(a)** the two-dimensional filter $\mathcal{A}_{:,:,t}$ is parameterized through an outer product of two 1D-filters $F_{:,t}$ and $G_{:,t}$; **(b)** $F_{:,t}$ is parameterized through a convolution of $f^{(1)}_{:,t} * \cdots * f^{(Q)}_{:,t}$, and similarly $G_{:,t}$ as a convolution of $g^{(1)}_{:,t} * \cdots * g^{(Q)}_{:,t}$; **(c)** we re-parameterize each constrained $(f^{(q)}_{-1,t}, f^{(q)}_{1,t})$ to unconstrained $(\alpha^f_{q,t}, \beta^f_{q,t})$, and similarly $(g^{(q)}_{-1,t}, g^{(q)}_{1,t})$ to $(\alpha^g_{q,t}, \beta^g_{q,t})$; **(d)** final parameters for unconstrained optimization are $(f^{(q)}_{0,t}, \alpha^f_{q,t}, \beta^f_{q,t}, g^{(q)}_{0,t} \alpha^g_{q,t}, \beta^g_{q,t})_{q=1}^{Q}$.

projected gradient descent [2], is more expensive as it requires an extra projection step. Moreover, it could be more difficult to achieve convergence. In order to avoid the aforementioned challenges, we introduce a *re-parameterization* mechanism to remove constraints needed to guarantee stability.

Theorem 7 (Re-parameterization)**.** *For a separable ARMA layer in Definition 5, if we re-parameterize each tuple* $(f^{(q)}_{-1,t}, f^{(q)}_{1,t}, g^{(q)}_{-1,t}, g^{(q)}_{1,t})$ *as learnable parameters* $(\alpha^f_{q,t}, \beta^f_{q,t}, \alpha^g_{q,t}, \beta^g_{q,t})$*:*

$$\begin{pmatrix} f^{(q)}_{-1,t} & g^{(q)}_{-1,t} \\ f^{(q)}_{1,t} & g^{(q)}_{1,t} \end{pmatrix} = \begin{pmatrix} f^{(q)}_{0,t} & 0 \\ 0 & g^{(q)}_{0,t} \end{pmatrix} \begin{pmatrix} \sqrt{2}/2 & -\sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{pmatrix} \begin{pmatrix} \alpha^f_{q,t} & \alpha^g_{q,t} \\ \tanh(\beta^f_{q,t}) & \tanh(\beta^g_{q,t}) \end{pmatrix} \tag{9}$$

then the layer is stable for ***arbitrary*** $\{(f^{(q)}_{0,t}, \alpha^f_{q,t}, \beta^f_{q,t}, g^{(q)}_{0,t}, \alpha^g_{q,t}, \beta^g_{q,t})_{q=1}^{Q}\}_{t=1}^{T}$ *with no constraints.*

In practice, we set $f^{(q)}_{0,t} = g^{(q)}_{0,t} = 1$ (since the scale can be learned by the moving-average kernel), and only store and optimize over each tuple $(\alpha^f_{q,t}, \beta^f_{q,t}, \alpha^g_{q,t}, \beta^g_{q,t})$. In other words, each autoregressive filter $\mathcal{A}_{:,:,t}$ is constructed from $(\alpha^f_{q,t}, \beta^f_{q,t}, \alpha^g_{q,t}, \beta^g_{q,t})_{q=1}^{Q}$ on the fly during training or inference.

Experimental Demonstration of Re-parameterization. To verify that the re-parameterization mechanism is essential for stable training, we train a VGG-11 network [31] on the CIFAR-10 dataset, where all convolutional layers are replaced by ARMA layers with autoregressive coefficients initialized as zeros. We compare the learning curves using the re-parameterization v.s. not using the re-parameterization in Figure 5. As we can see, the training quickly converges under our proposed re-parameterization mechanism with which the stability of the network is guaranteed. However, without the re-parameterization mechanism, a naive training of the ARMA network never converges and gets NaN error quickly. The experiment thus verifies that the theory in Theorem 7 is effective in guaranteeing stability.

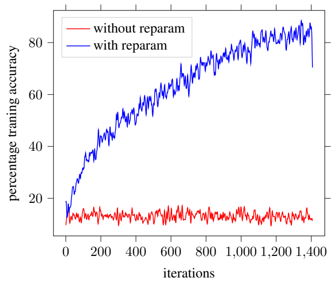

Figure 5: Learning curves with and without re-parameterization on an ARMA network with a VGG-11 backbone on CIFAR-10.

6 Experiments

We apply our ARMA networks on two dense prediction problems – pixel-level video prediction and semantic segmentation to demonstrate the effectiveness of ARMA networks. **(1)** We incorporate our ARMA layers in U-Nets [30, 36] for semantic segmentation, and in the ConvLSTM network [6, 39] for video prediction. **We show that the resulted ARMA U-Net and ARMA-LSTM models uniformly outperform the baselines on both tasks. (2)** We then interpret the varying performance of ARMA networks on different tasks by visualizing the histograms of the learned autoregressive coefficients. We include the detailed setups (datasets, model architectures, training strategies, and evaluation metrics) and visualization in Appendix A for reproducibility purposes.

Semantic Segmentation on Biomedical Medical Images. We evaluate our ARMA U-Net on the lesion segmentation task in ISIC 2018 challenge [38], comparing against a baseline U-Net [30] and non-local U-Net [36] (U-Net augmented with non-local attention blocks).

Table 2: Semantic segmentation on ISIC dataset. For all metrics (ACC, SE, SP, PC, F1 and JS), higher values indicates better performance. The reported numbers are an average of 10 runs with different seeds.

Model	params.	ACC	SE	SP	PC	F1	JS
U-Net [30]	3.453M	0.946 $\pm$ 0.003	0.884 $\pm$ 0.019	**0.977** $\pm$ 0.005	0.857 $\pm$ 0.020	0.842 $\pm$ 0.009	0.754 $\pm$ 0.011
NL U-Net [36]	4.403M	0.945 $\pm$ 0.003	0.877 $\pm$ 0.017	0.973 $\pm$ 0.004	0.844 $\pm$ 0.014	0.831 $\pm$ 0.012	0.741 $\pm$ 0.013
ARMA U-Net	3.455M	0.955 $\pm$ 0.003	0.896 $\pm$ 0.011	0.972 $\pm$ 0.005	**0.873** $\pm$ 0.011	0.861 $\pm$ 0.007	0.780 $\pm$ 0.009
NL ARMA U-Net	4.405M	**0.960** $\pm$ 0.002	**0.909** $\pm$ 0.009	0.968 $\pm$ 0.004	0.870 $\pm$ 0.011	**0.870** $\pm$ 0.006	**0.790** $\pm$ 0.008

ARMA networks outperform both baselines in almost all metrics. As shown in Table 2, our (non-local) ARMA U-Net outperforms both U-Net and non-local U-Net except for specificity (SP). Furthermore, we find that the synergy of non-local attention and ARMA layers achieves the best results among all.

Pixel-level Video Prediction. We evaluate our ARMA-LSTM network on the Moving-MNIST-2 dataset [12] with different moving velocities, comparing against the baseline ConvLSTM network [6, 39] and its augmentation using dilated convolutions and non-local attention blocks [34]. As shown in the visualizations in Appendix A, the dilated ARMA-LSTM does not have gridding artifacts as in dilated Conv-LSTM; that is, *ARMA removes the gridding artifacts.*

Table 3: 10-frames **video prediction** on Moving-MNIST-2 with three different speeds (results averaged over 10 predicted frames). MA and AR denote the size of moving-average and autoregressive kernels respectively, and dil. denotes dilation in the moving-average kernel. Higher PSNR, SSIM values indicate better performance.

Model	MA	AR	dil.	params.	original speed		2X speed		3X speed	
					PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Conv-LSTM (size 3)	3	1	1	0.887M	18.24	0.867	16.62	0.827	15.81	0.810
Conv-LSTM (size 5)	5	1	1	2.462M	19.58	0.901	17.61	0.856	16.99	0.841
Dilated Conv-LSTM	3	2	2	0.887M	19.16	0.893	17.92	0.858	17.48	0.846
Dilated ARMA-LSTM	3	3	2	0.893M	**19.72**	**0.904**	18.05	0.870	17.65	0.855
ARMA-LSTM (size 3)	3	2	1	0.893M	**19.72**	0.899	**18.73**	**0.881**	**18.13**	**0.869**

ARMA networks outperform larger networks: As shown in Table 3, our ARMA networks with kernel sizes 3×3 outperform all baselines under all velocities (at the original speed, our ARMA network requires dilated convolutions to achieve the best performance). Moreover, for videos with a higher moving speed, the advantage is more pronounced as expected due to ARMA's ability to expand the ERF. The ARMA networks improve the best baseline (Conv-LSTM with kernel size 5×5) in PSNR by 6.36% at 2X speed and by 6.70% at 3X speed, with 63.7% fewer parameters.

Table 4: Comparison with non-local attention blocks on **video prediction**. The original networks are the same as in Table 3. Each non-local network additionally inserts two non-local blocks in the corresponding base network.

Model	Original		Non-local	
	PSNR	SSIM	PSNR	SSIM
ConvLSTM (size 3)	18.24	0.867	19.45	0.895
ConvLSTM (size 5)	19.58	**0.901**	19.18	0.891
ARMA-LSTM (size 3)	**19.72**	0.899	**19.62**	**0.897**

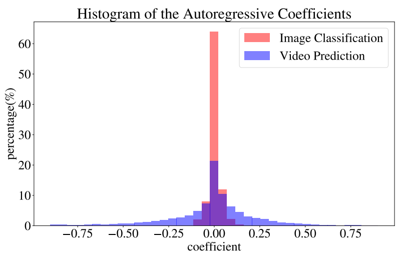

Figure 6: Histogram of the autoregressive coefficients in trained ARMA networks.

ARMA networks outperform non-local blocks: As shown in Table 4, our ARMA-LSTM with kernel sizes 3×3 outperforms the Conv-LSTMs augmented by non-local blocks. However, the non-local mechanism does not always improve the baselines or our models. When both ARMA-LSTM and Conv-LSTM are combined with non-local blocks, our model achieves better performance compared to the non-ARMA baselines.

Interpretation by Autoregressive Coefficients. Figure 6 compares the histograms of the trained autoregressive coefficients between video prediction and image classification to explain why ARMA networks achieve impressive performance in dense prediction, (subsection A.4 demonstrates the ARMA networks' performance in image classifications with baselines VGG and ResNet.)

1. The histograms demonstrate how ARMA networks adaptively learn autoregressive coefficients according to the tasks. As motivated in the introduction, dense prediction such as video prediction requires each layer to have a large receptive field to capture global information.
2. The large autoregressive coefficients in the video prediction model suggest that the overall ERF is significantly expanded. In the image classification model, global information is already aggregated by pooling (downsampling) layers and a fully-connected classification layer. Therefore, the ARMA layers automatically learn nearly zero autoregressive coefficients.

7 Discussion

This paper proposes a novel *ARMA* layer capable of expanding a network's effective receptive field adaptively. Our method is related to techniques in signal processing and machine learning. First, an ARMA layer is equivalent to a multi-channel *impulse response filter* in signal processing [29]. Alternatively, we can interpret the autoregressive layer as a learnable *spectral normalization* [26] following the moving-average layer. Additionally, the ARMA layer is a *linear recurrent neural network*, where the recurrent propagations are over the spatial domain (section 2).

Funding Disclosure

This research is supported by startup fund from Department of Computer Science of University of Maryland, National Science Foundation IIS-1850220 CRII Award 030742- 00001, DOD-DARPA-Defense Advanced Research Projects Agency Guaranteeing AI Robustness against Deception (GARD), and Laboratory for Physical Sciences at University of Maryland. Huang is also supported by Adobe, Capital One and JP Morgan faculty fellowships.

Impact Statement

Our presented ARMA layer is a plug-and-play module that can replace any convolutional layer in neural networks. The module is particularly effective in dense prediction problems, including video prediction, object detection and medical image segmentation.

These improved performance in all these applications could revolutionize people's daily life; it could alarm humans potential risks, relieve workers from repeat laboring, and help experts in making better decisions. For examples, video prediction in autonomous system helps to anticipate future risks and contributes to safe self-driving, and medical image segmentation could provide additional information to doctors and help them to make more reliable decisions on high-stakes tasks.

However, these applications also raise controversies in the society. For examples, a faulty prediction in self-driving car or medical diagnostic system could lead to deadly consequence. Furthermore, object detection could be misused for military purposes. To understand and thus mitigate these potential risks, we suggest researchers in engineering and social sciences to investigate questions such as:

- How to systematically verify the capacity of a machine learning model, such that certain behavior can be prohibited before deployment?
- How to define the responsibility if a machine learning system produces an undesired outcome (e.g. car crash or misdiagnosis)?

References

[1] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179, 2015.

[2] Dimitri P Bertsekas and Athena Scientific. *Convex optimization algorithms*. Athena Scientific Belmont, 2015.

[3] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[4] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576*, 2016.

[5] Wonmin Byeon, Thomas M Breuel, Federico Raue, and Marcus Liwicki. Scene labeling with lstm recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3547–3555, 2015.

[6] Wonmin Byeon, Qin Wang, Rupesh Kumar Srivastava, and Petros Koumoutsakos. Contextvp: Fully context-aware video prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 753–769, 2018.

[7] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[8] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[9] Noel CF Codella, David Gutman, M Emre Celebi, Brian Helba, Michael A Marchetti, Stephen W Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin Mishra, Harald Kittler, et al. Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (isic). In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 168–172. IEEE, 2018.

[10] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[11] 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.

[12] Github repo. `https://github.com/jthsieh/DDPAE-video-prediction/blob/master/data/moving_mnist.py`. [Online; accessed 05-Jan-2020].

[13] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[14] 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.

[15] Matthias Holschneider, Richard Kronland-Martinet, Jean Morlet, and Ph Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets*, pages 286–297. Springer, 1990.

[16] Yunho Jeon and Junmo Kim. Active convolution: Learning the shape of convolution for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4201–4209, 2017.

[17] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*, 2015.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[20] Yanghao Li, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Scale-aware trident networks for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6054–6063, 2019.

[21] Jae S Lim. Two-dimensional signal and image processing. *ph*, 1990.

[22] Sifei Liu, Shalini De Mello, Jinwei Gu, Guangyu Zhong, Ming-Hsuan Yang, and Jan Kautz. Learning affinity via spatial propagation networks. In *Advances in Neural Information Processing Systems*, pages 1520–1530, 2017.

[23] Sifei Liu, Jinshan Pan, and Ming-Hsuan Yang. Learning recursive filters for low-level vision via a hybrid neural network. In *European Conference on Computer Vision*, pages 560–576. Springer, 2016.

[24] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[25] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *Advances in neural information processing systems*, pages 4898–4906, 2016.

[26] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[27] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*, 2018.

[28] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[29] Alan V Oppenheim, John R Buck, and Ronald W Schafer. Discrete-time signal processing. 2014.

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

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

[32] Marijn F Stollenga, Wonmin Byeon, Marcus Liwicki, and Juergen Schmidhuber. Parallel multi-dimensional lstm, with application to fast biomedical volumetric image segmentation. In *Advances in neural information processing systems*, pages 2998–3006, 2015.

[33] 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.

[34] 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.

[35] Zhengyang Wang and Shuiwang Ji. Smoothed dilated convolutions for improved dense prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2486–2495, 2018.

[36] Zhengyang Wang, Na Zou, Dinggang Shen, and Shuiwang Ji. Non-local u-nets for biomedical image segmentation. In *AAAI*, pages 6315–6322, 2020.

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

[38] Webpage. `https://challenge2018.isic-archive.com`. [Online; accessed 29-May-2020].

[39] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.

[40] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[41] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 472–480, 2017.

[42] 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.