ARMA Nets: Expanding Receptive Field for Dense Prediction

Jiahao Su [1] **Shiqi Wang** [2] **Furong Huang** [3]

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, originally designed for image classification, are restrictive in these problems since their receptive fields are limited by the filter size. In this work, we propose autoregressive moving-average (ARMA) layer, a novel module in neural networks to allow explicit dependencies of output neurons, which significantly expands the receptive field with minimal extra parameters. We show experimentally that the effective receptive field of neural networks with ARMA layers expands as autoregressive coefficients become larger. In addition, we demonstrate that neural networks with ARMA layers substantially improve the performance of challenging pixel-level video prediction tasks as our model enlarges the effective receptive field.

1. Introduction

Convolutional layers in neural networks have been successfully applied to a wide range of tasks in computer vision and natural language processing. The convolution of filters and input feature maps in each layer generates output feature maps or responses in which each element encodes some local regions from the input feature maps. The size of the encoded local region is determined by the size of the filters and roughly corresponds to the receptive field of the response. In tasks such as image classifications, small filters and thus small receptive field is usually good enough for accurate classification as most classifications could be accurately achieved from the information extracted in local regions of the images. However, in more challenging dense prediction tasks such as pixel-level video prediction, both global and local information are needed for accurate prediction, requiring larger receptive fields.

Two naive approaches to expand the receptive field are to deepen the network and to enlarge the filters. The former introduces computational and optimization difficulties due to problems of exploding/vanishing gradients, hyper-parameters tuning, and memory explosion. The latter drastically increases the model complexity, making the already expensive training on dense prediction more challenging.

Dilated convolutional models (Yu et al., 2017; Giusti et al., 2013; Li et al., 2014; Sermanet et al., 2013; Papandreou et al., 2015), which upsample convolutional filters by inserting zeros between weights, are proposed to include larger receptive field without extra model parameters in the neural networks (Chen et al., 2017a;b; Hamaguchi et al., 2018). Dilated convolutions do not reduce the spatial resolution of responses, a key difference from downsampling layers such as pooling layers or convolutions with stride larger than one that expand the receptive field but reduce the spatial resolution. However dilated convolutions could result in gridding artifacts (Yu et al., 2017; Wang et al., 2018a; Hamaguchi et al., 2018) where adjacent units in the output are computed from completely separate sets of pixels in the input. It results in inconsistency of local information and hampers the performance of the dilated convolutional neural networks (Wang & Ji, 2018).

We introduce a framework of *Autoregressive Moving-Average* (ARMA) layer to naturally expand the receptive field of the responses. ARMA networks consist of ARMA layers which implement extra convolutions between output neurons in addition to the traditional convolutions between the input and filters. The ARMA layer we introduce effectively correspond to an autoregressive moving-average model on the response; and we recognize traditional convolution as a moving-average model. Our ARMA networks expand the effective receptive field to incorporate global information without reducing spatial resolution by downsampling or introducing substantial amount of parameters.

In this paper, once we introduce the ARMA networks, we design the non-trivial prediction and learning procedures to achieve efficient computation using *fast Fourier transforms*. We recognize ARMA's intrinsic susceptibility to

[1]Department of Electrical and Computer Engineering, University of Maryland, College Park, MD [2]Department of Computer Science, Nanjing University, Nanjing, China [3]Department of Computer Science, University of Maryland, College Park, MD. Correspondence to: Jiahao Su <jiahaosu@terpmail.umd.edu>, Furong Huang <furongh@cs.umd.edu>.

instability during training and test. To guarantee stability, we propose a sophisticated design of ARMA layers, involving separability of autoregressive coefficient (detailed later). Under this sophisticated design of *separable ARMA layers*, a novel re-parameterization mechanism is proposed to guarantee convergence without the computational difficulties of constrained optimization.

Summary of Contributions

- We introduce a novel ARMA layer and ARMA networks, which naturally expand the effective receptive field, improving dense prediction performance.
- We recognize an intrinsic instability problem in ARMA networks. To address the problem, we propose a design of *separable ARMA layers* and a re-parameterization mechanism to guarantee stable learning and prediction.
- We successfully apply ARMA networks to pixel-level video prediction problems. Our proposed models outperform the state-of-the-art convolutional LSTM networks in video prediction. Moreover, we illustrate that dilated ARMA networks, which combines the idea of ARMA layer with dilated convolutions, establish new state-of-the-art performance for pixel-level video prediction.

2. Related Works

Dilated convolution (a.k.a. atrous convolution (Holschneider et al., 1990)) increases the receptive field size by upsampling the filter coefficients. The module is widely used in dense prediction problems, including semantic segmentation (Long et al., 2015; Yu & Koltun, 2015; Chen et al., 2017a), and objection detection (Dai et al., 2016; Li et al., 2019). The naive dilated convolution suffers from gridding artifacts if a feature map contains higher frequency than the upsampling rate (Yu et al., 2017), which can be alleviated by extra anti-aliasing convolutional layer (Yu et al., 2017), group interacting layer (Wang & Ji, 2018) or spatial pyramid pooling (Chen et al., 2017b). Our proposed ARMA layer is perfectly complementary to dilated convolution, and in fact the autoregressive part in ARMA can be interpreted as an anti-aliasing filter for the dilated convolution in the moving-average part. In subsection 6.3, we show an application of *dilated ARMA layer* in video prediction.

Deformable convolution allows the filter shape (i.e. locations of the incoming pixels) to be learnable (Dai et al., 2017; Jeon & Kim, 2017; Zhu et al., 2019), and is therefore capable to adaptively adjust its receptive field. As with dilated convolution, our proposed ARMA layer is complementary to this approach, which replaces the normal convolutions in ARMA layer by deformable convolutions.

Spatial recurrent neural network applies recurrent propagations over the spatial domain (Byeon et al., 2015; Oord et al., 2016; Kalchbrenner et al., 2015; Stollenga et al., 2015; Liu et al., 2016), and learns the affinity between neighboring pixels (Liu et al., 2017). Our proposed ARMA layer can be interpreted as an isotropic linear recurrent neural network. Compared to the aforementioned models, the spatial recurrences in ARMA layer can be efficiently evaluated using fast Fourier transform (FFT) algorithm.

3. ARMA Neural Networks

In this section, we introduce the basic concept of *ARMA layer*, its ability to capture global information and comparison with traditional convolutional layer.

3.1. ARMA Layer

Traditional convolutional layer, parameterized by a fourth-order kernel $\mathcal{W} \in \mathbb{R}^{K_w \times K_w \times S \times T}$, is essentially a *moving-average* model (Box et al., 2015) as

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

where K_w is the size of the filters, S, T are the number of input/output channels, and **:** denotes the aggregation of all elements in the corresponding coordinate. The kernel $\mathcal{W}$ is also called the *moving-average coefficients*.

We introduce a novel *Autoregressive-Moving-Average* layer (ARMA layer), illustrated in Figure 1 and Figure 2. In an ARMA layer, each output neuron can be affected by an input pixel faraway and thus receives global information, through interconnections among the output neurons. These interconnections are realized as a convolution on each output map, allowing efficient evaluation. Formally, we define ARMA layer in Definition 1.

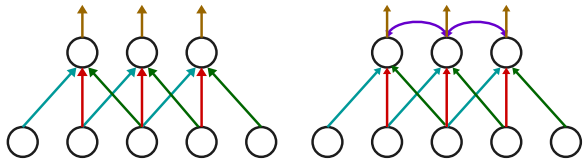
Definition 1 (ARMA layer). *An ARMA layer is parameterized by a moving-average kernel $\mathcal{W} \in \mathbb{R}^{K_w \times K_w \times S \times T}$ and an autoregressive kernel $\mathcal{A} \in \mathbb{R}^{K_a \times K_a \times T}$. The ARMA layer receives an input $\mathcal{X} \in \mathbb{R}^{I_1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

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

where K_a is the size for the autoregressive kernel.

Remarks: **(1)** The ARMA layer takes input and returns output with exactly the same shapes as a traditional convolution layer. Therefore, it can readily replace *any* convolutional layer in neural networks with an extra $K_a^2 T$ parameters [1]. **(2)** Different from traditional convolutional layers, computing Equation 2 and its corresponding backpropagation requires solving linear equations. We study the computational

[1] A traditional convolutional layer requires $K_w^2 ST$ parameters.



(a) Traditional convolution. **(b)** ARMA convolution.

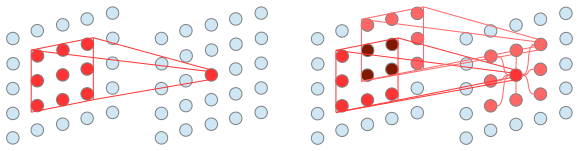
Figure 1. Comparison of ARMA and traditional convolutions. The ARMA layer introduces direct interactions among output neurons (a.k.a. units) explicitly.

problems in section 4. **(3)** Since the output interconnections are realized by convolutions, the ARMA layer maintains the *shift-invariant* property as in traditional convolutional layer.

The motivation of introducing ARMA layer is to enlarge the input region that makes substantial contribution to an output neuron without increasing the filter size, which avoids difficulties in training large models. We now formally introduce the concept of *Effective Receptive Field* (ERF) to characterize the effective input region.

3.2. Effective Receptive Field

Effective receptive field (Luo et al., 2016) denotes roughly the area of the input region that makes *substantial* contribution to an output neuron. Intuitively, as shown in Figure 2, each output neuron in a traditional convolutional layer can only receive information from a small subset of its input map, therefore lacks the capacity of collecting global information. In contrast, each output neuron in an ARMA layer has the potential to receive from an input pixel faraway, due to the information flow through a neighboring neuron.



(a) Traditional convolution. **(b)** ARMA convolution.

Figure 2. Comparison of receptive field in traditional and ARMA convolutions. In an ARMA layer, the centered output neuron has a receptive field larger than the kernel size.

In this section, we analytically compare the ERF size of a network with ARMA layers v.s. traditional convolutional layers. Formally, consider an output neuron at location (i_1, i_2), the impact from an input pixel at $(i_1 - p_1, i_2 - p_2)$ (i.e from (p_1, p_2) pixels away) is measured by the amplitude of partial derivative $\left|\partial \mathcal{Y}^L_{i_1,i_2,t} / \partial \mathcal{X}^0_{i_1-p_1,i_2-p_2,s}\right|$ (where the superscripts index the layers), i.e. how much $\mathcal{Y}^L_{i_1,i_2,t}$ changes as $\mathcal{X}^0_{i_1-p_1,i_2-p_2,s}$ perturbs by a small amount.

Definition 2 (Effective receptive field, ERF). *Consider an L-layers network with S-channels input $\mathcal{X}^0 \in \mathbb{R}^{I_1 \times I_2 \times S}$ and T-channels output $\mathcal{Y}^L \in \mathbb{R}^{I'_1 \times I'_2 \times T}$, its ERF is defined as the empirical distribution of the gradient maps:*

$$ERF(p_1, p_2) = \frac{1}{I'_1 I'_2 S T} \sum_{s,t,i'_1,i'_2} \frac{g(i'_1, i'_2, p_1, p_2)}{\sum_{i'_1,i'_2} g(i'_1, i'_2, p_1, p_2)}$$
$$\text{where } g(i'_1, i'_2, p_1, p_2) = \left| \frac{\partial \mathcal{Y}^L_{i'_1,i'_2,t}}{\partial \mathcal{X}^0_{i'_1-p_1,i'_2-p_2,s}} \right| \tag{3}$$

To measure the size of an ERF, we define its radius as the standard deviation of the empirical distribution:

$$r^2(ERF) = \sum_{p_1,p_2} \left(p_1^2 + p_2^2\right) ERF(p_1, p_2) - \left(\sum_{p_1,p_2} \sqrt{p_1^2 + p_2^2}\, ERF(p_1, p_2) \right)^2 \tag{4}$$

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. However, it is useful to estimate its size approximately by a simplified architecture.

Theorem 3 (ERF of a simplified traditional CNN). *Consider an L layers linear CNN, where each layer computes a moving-average of its input: $y_i = \sum_{p=0}^{K-1} x_{i-p}/K$ (where the coefficients in a kernel of size K are uniform). The radius of ERF for such a network is approximately*

$$r(ERF) = \sqrt{L} \cdot \sqrt{\frac{K^2 - 1}{12}} = O(K\sqrt{L}) \tag{5}$$

The example shows that radius of a CNN grows *linearly* with the kernel size K but *sub-linearly* with the depth L.

Similarity, we can analytically estimate the ERF of ARMA networks with simplified ARMA model at each layer.

Theorem 4 (ERF of an simplified ARMA network). *Consider an linear network with L ARMA layers, each of which computes its output according to $y_i - a\, y_{i-1} = \sum_{p=0}^{K-1} x_{i-p}/K$ (that is, the moving-average coefficients are uniform with size K, and the autoregressive coefficients has length 2 with $a_0 = 1, a_1 = -a$). Suppose $0 \le a < 1$, the radius of ERF for such a network is approximately*

$$r(ERF) = \sqrt{L} \cdot \sqrt{\frac{K^2 - 1}{12} + \frac{a}{(1-a)^2}} \tag{6}$$

Compared with a traditional CNN, an ARMA network can have arbitrarily large effective field with a single extra parameter a at each layer. The proofs for both Theorems 3 and 4 are provided in Appendix A.

ERF comparison: Now we compare the ERF of a CNN in Theorem 3 and the one of an ARMA network in Theorem 4. **(2)** When the autoregressive coefficient a is large (e.g. $a > 1-1/K$), the second term dominates the radius, and the ERF of ARMA network is substantially larger than that of CNN. In particular, the radius tends to infinity as a approaches to 1. **(2)** As a gets smaller (e.g. $a < 1 - 1/K$), 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. These theoretical results are numerically verified in subsection 6.1.

Layer	# of params.	# of FLOPs	r(ERF)^2
Conv.	$K_w^2 ST$	$O(TI_1I_2K_w^2S)$	$O(LK_w^2)$
ARMA	$K_w^2 ST$ $+K_a^2 T$	$O(TI_1I_2K_w^2S$ $+TI_1I_2\log\max(I_1,I_2))$	$O\big(LK_w^2$ $+La/(1-a)^2\big)$

Table 1. Comparison of traditional convolution layer and our proposed ARMA layer. The second terms in ARMA layer is the extra cost/benefit compared to traditional convolutional layer.

4. Prediction and Learning of ARMA Layer

As remarked in subsection 3.1, the forward computation of an ARMA layer in Equation 2 requires *solving a set of linear equations*. **(1)** Naively with Gaussian elimination, the algorithm is too expensive to be practical, and therefore we need to seek for a more efficient solution to Equation 2. **(2)** Furthermore, the process of solving linear equations is not trivially back-propagable, and we have to derive the backward equations analytically. **(3)** Finally, we will also need to devise an efficient algorithm to solve the backpropagation equations efficiently. In the section, we address these aforementioned problems.

4.1. Prediction in the ARMA Layer

The forward pass of a ARMA layer in Equation 2 can be rewritten as a cascade of two steps:

$$\mathcal{T}_{:,:,t} = \sum_{s=1}^{S} \mathcal{W}_{:,:,t,s} * \mathcal{X}_{:,:,s} \tag{7a}$$

$$\mathcal{A}_{:,:,t} * \mathcal{Y}_{:,:,t} = \mathcal{T}_{:,:,t} \tag{7b}$$

where $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times T}$ is an intermediate result. Notice that Equation 7a is simply a traditional convolutional layer, therefore we are only concerned of the computation in Equation 7b, which we name as an *autoregressive layer*.

Using Gaussian elimination, the linear equations in Equation 7b can be solved by in $O((I_1^2 + I_2^2)I_1I_2T)$, which is cubic in the dimensions. Fortunately, the structure of convolutions allow us to accelerate the computation by the *fast Fourier transform* (FFT) algorithm.

Definition 5 (Discrete Fourier Transform, DFT). *Given a third-order tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times C}$, we define its DFT of $\mathcal{X}$ over the spatial coordinates as $\widetilde{\mathcal{T}} \in \mathbb{C}^{I_1 \times I_2 \times C}$.*

$$\widetilde{\mathcal{T}}_{k_1,k_2,c} = \sum_{i_1=0}^{I_1-1} \sum_{i_2=0}^{I_2-1} \mathcal{T}_{i_1,i_2,c}\, \omega_{I_1}^{i_1k_1}\, \omega_{I_2}^{i_2k_2} \tag{8}$$

where $\omega_I = \exp(2\pi/I)$ is the I^{th} root of unity. Given the transformed tensor $\widetilde{\mathcal{T}} \in \mathbb{C}^{I_1 \times I_2 \times C}$, the original tensor $\mathcal{T}$ can be recovered by the inverse DFT (IDFT) as

$$\mathcal{T}_{i_1,i_2,c} = \frac{1}{I_1I_2} \sum_{k_1=0}^{I_1-1} \sum_{k_2=0}^{I_2-1} \mathcal{T}_{i_1,i_2,c} \omega_{I_1}^{-i_1k_1} \omega_{I_2}^{-i_2k_2} \tag{9}$$

The definition above can be extended to convolutional kernels by first zero-padding $\mathcal{A}$ to be $\mathbb{R}^{I_1 \times I_2 \times C}$. With DFT, the *deconvolution* in Equation 7b can be rewritten as

$$\widetilde{\mathcal{A}}_{k_1,k_2,t} \widetilde{\mathcal{Y}}_{k_1,k_2,t} = \widetilde{\mathcal{T}}_{k_1,k_2,t} \tag{10}$$

where $\widetilde{\mathcal{A}}$ and $\widetilde{\mathcal{T}}$ are computed from $\mathcal{A}$ and $\mathcal{T}$ with Equation 8, and $\mathcal{Y}$ is recovered from $\widetilde{\mathcal{Y}}$ by Equation 9. With FFT, each DFT can be evaluated in $O(I_1I_2\log(\max(I_1,I_2)))$, therefore the time complexity of Equation 7b reduces from $O((I_1^2 + I_2^2)I_1I_2T)$ to $O(\log(\max(I_1,I_2))I_1I_2T)$.

4.2. Backpropagation of the ARMA Layer

In order to use ARMA layer in neural networks, we derive the backpropagation of Equation 2. Since Equation 7a is simply traditional convolutional layer, we will only need to derive the backpropagation equations for Equation 7b.

Theorem 6 (Backpropagation of ARMA layer). *Given the forward pass $\mathcal{A}_{:,:,t} * \mathcal{Y}_{:,:,t} = \mathcal{T}_{:,:,t}$ and the gradient $\partial\mathcal{L}/\partial\mathcal{Y}$, the gradients $\partial\mathcal{L}/\partial\mathcal{A}$ and $\partial\mathcal{L}/\partial\mathcal{X}$ can be obtained through the following two ARMA models:*

$$\mathcal{A}^\top_{:,:,t} * \frac{\partial\mathcal{L}}{\partial\mathcal{A}_{:,:,t}} = -\mathcal{Y}^\top_{:,:,t} * \frac{\partial\mathcal{L}}{\partial\mathcal{Y}_{:,:,t}} \tag{11a}$$

$$\mathcal{A}^\top_{:,:,t} * \frac{\partial\mathcal{L}}{\partial\mathcal{T}_{:,:,t}} = \frac{\partial\mathcal{L}}{\partial\mathcal{Y}_{:,:,t}} \tag{11b}$$

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

Similar to the forward pass, both Equation 11a and Equation 11b can be computed efficiently using FFT:

$$\frac{\partial\mathcal{L}}{\partial\widetilde{\mathcal{A}}_{k_1,k_2,t}} = -\frac{\widetilde{\mathcal{Y}}_{k_1,k_2,t}}{\widetilde{\mathcal{A}}_{k_1,k_2,t}} \cdot \frac{\partial\mathcal{L}}{\partial\widetilde{\mathcal{Y}}_{k_1,k_2,t}} \tag{12a}$$

$$\frac{\partial\mathcal{L}}{\partial\widetilde{\mathcal{T}}_{k_1,k_2,t}} = \frac{1}{\widetilde{\mathcal{A}}_{k_1,k_2,t}} \cdot \frac{\partial\mathcal{L}}{\partial\widetilde{\mathcal{Y}}_{k_1,k_2,t}} \tag{12b}$$

Theorem 6 along with Equations (12a) and (12b) are proved in Appendix B.

5. Stability of ARMA Layers

An ARMA model with arbitrary coefficients is not necessarily stable. For example, the model $y_i - cy_{i-1} = x_i$ is unstable if $|c| > 1$, since the output y may recursively amplify itself and diverge to infinity: even when the input x is bounded. Traditionally, the stability is formalized as *Bound-Input, Bound-Output Stability* (BIBO stability) (Oppenheim & Schafer, 2014) defined in Definition 7.

Definition 7 (BIBO stability). *An input x (or an output y) is bounded if $|x_i| < B, \forall i \in \mathbb{Z}$ (or $|y_i| < B, \forall i \in \mathbb{Z}$) for some $B > 0$. A model is BIBO stable if the output y is bounded given any bounded input x, that is*

$$\begin{aligned} &\forall x, (\exists B_1 > 0, |x_i| < B_1, \forall i \in \mathbb{Z}) \\ \implies &(\exists B_2 > 0, |y_i| < B_2, \forall i \in Z) \end{aligned} \tag{13}$$

5.1. Stability Constraints for ARMA layer

The key to guarantee BIBO stability for an ARMA layer is to constrain its autoregressive coefficients $\mathcal{A}$ to prevent the output from repeatedly amplifying the input. However, the analytic form of the constraints are nontrivial for a general ARMA layer. To derive the analytical form of stability constraints, we propose to a special design of the ARMA layer, called *separable ARMA layer* as in Definition 8, inspired from the separable filters (Lim, 1990).

Definition 8 (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 Q sets of length-3 autoregressive kernels $\{(f^q_{:,:,t}, g^q_{:,:,t})\}_{q=1}^{Q}$. The separable ARMA layer receives an input $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times S}$ and returns an output $\mathcal{Y} \in \mathbb{R}^{I_1' \times I_2' \times T}$ according to an ARMA model:*

$$\begin{aligned} &\left(f^1_{:,:,t} * \cdots * f^Q_{:,:,t}\right) \otimes \left(g^1_{:,:,t} * \cdots * g^Q_{:,:,t}\right) * \mathcal{Y}_{:,:,:,t} \\ &\qquad = \sum_{s=1}^{S} \mathcal{W}_{:,:,t,s} * \mathcal{X}_{:,:,:,s}. \end{aligned} \tag{14}$$

where $\otimes$ is outer product of two one-dimensional kernels.

Remarks: The intuition behind the design of separable ARMA layer comes from the separable filters. We design the autoregressive kernel to be a separable filter $\mathcal{A}_{:,:,t} = F_{:,t} \otimes G_{:,t}$. As a result, the original two-dimensional autoregressive filter $\mathcal{A}_{:,:,t}$ is characterized by two one-dimensional filters $F_{:,t}$ and $G_{:,t}$. Furthermore, any one-dimensional filter can be decomposed into a convolution of length-3 filters by the fundamental theorem of algebra (Oppenheim & Schafer, 2014), therefore $F_{:,t}$ and $G_{:,t}$ can be factorized as $F_{:,t} = f^1_{:,t} * f^2_{:,t} \cdots * f^Q_{:,t}$, $G_{:,t} = g^1_{:,t} * g^2_{:,t} \cdots * g^Q_{:,t}$. The parameters of a separable ARMA layer reduce to a moving-average kernel $\mathcal{W} \in \mathbb{R}^{K_w \times K_w \times S \times T}$ and a set of length-3 autoregressive kernels $\{(f^q_{:,:,t}, g^q_{:,:,t})\}_{q=1}^{Q}$.

In Theorem 9, we provide a necessary condition for the proposed separable ARMA layer to be BIBO stable.

Theorem 9 (BIBO Stability of separable ARMA layer). *A necessary condition for BIBO stability of the separable ARMA layer defined in Definition 8 is*

$$\left|f^q_{-1,t} + f^q_{1,t}\right| < f^q_{0,t}, \quad \forall q \in [Q] \tag{15a}$$
$$\left|g^q_{-1,t} + g^q_{1,t}\right| < g^q_{1,t}, \quad \forall q \in [Q] \tag{15b}$$

The proof of this theorem is deferred to Appendix C.

5.2. Achieving Stability via Re-parameterization

In principle, the constraints required for stability in a ARMA layer as in Theorem 9 could be implemented through constrained optimization. However, constrained optimization algorithm, such as projected gradient descent (Bertsekas & Scientific, 2015), 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 guarantee stability in ARMA layer.

Definition 10 (Re-parameterization)**.** *Consider a feasible set $|a_{-1} + a_1| < 1$, we can re-parameterize the variables a_1, a_{-1} to α, β such that (α, β) can take any value in $\mathbb{R}^2$.*

$$a_{-1} = \frac{\sqrt{2}}{2}\tanh(\alpha) - \frac{\sqrt{2}}{2}\beta \tag{16a}$$
$$a_1 = \frac{\sqrt{2}}{2}\tanh(\alpha) + \frac{\sqrt{2}}{2}\beta \tag{16b}$$

The re-parameterization is illustrated in Figure 9.

Theorem 11 (Stability via Re-parameterization)**.** *For a separable ARMA layer defined in Definition 8, if we re-parameterize the coefficients of each length-3 autoregressive kernels $\{(f^q_{:,:,t}, g^q_{:,:,t})\}_{q=1}^{Q}$ as*

$$f^q_{-1,t} = f^q_{0,t}\left(\frac{\sqrt{2}}{2}\tanh(\alpha_{f^q,t}) - \frac{\sqrt{2}}{2}\beta_{f^q,t}\right) \tag{17a}$$
$$f^q_{1,t} = f^q_{0,t}\left(\frac{\sqrt{2}}{2}\tanh(\alpha_{f^q,t}) + \frac{\sqrt{2}}{2}\beta_{f^q,t}\right) \tag{17b}$$
$$g^q_{-1,t} = g^q_{0,t}\left(\frac{\sqrt{2}}{2}\tanh(\alpha_{g^q,t}) - \frac{\sqrt{2}}{2}\beta_{g^q,t}\right) \tag{17c}$$
$$g^q_{1,t} = g^q_{0,t}\left(\frac{\sqrt{2}}{2}\tanh(\alpha_{g^q,t}) + \frac{\sqrt{2}}{2}\beta_{g^q,t}\right) \tag{17d}$$

then the layer is guaranteed to be BIBO stable.

In practice, we store (and optimize over) a pair of parameters α and β for each one dimensional length-3 filter, and construct the autoregressive coefficients bottom-up.

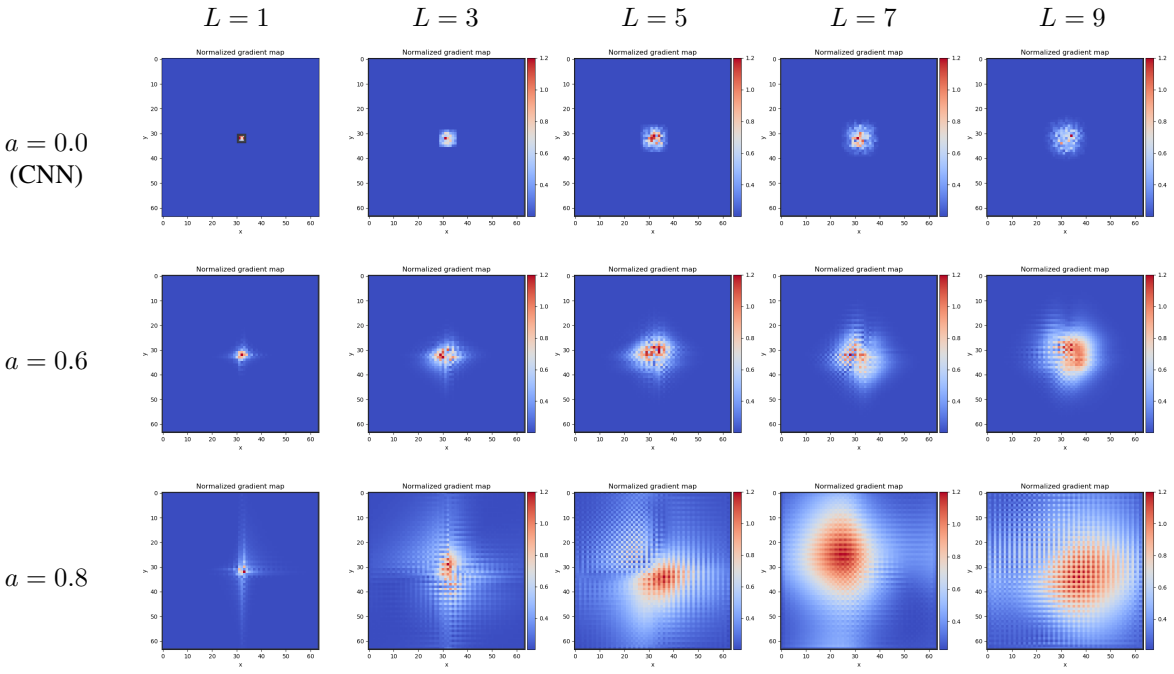


Figure 3. Visualizations of the ERFs. We consider two variables in ARMA networks: depth and the magnitude of the autoregressive coefficients. In the figures, we change the depth from 1, 3, 5, 7 to 9 *horizontally*, and magnitude from 0, 0.6 to 0.8 *vertically*. An ARMA network can have a large ERF even when the network is small, and its ability to expand the ERF increases as the network gets deeper.

6. Experiments

In this section, **(1)** we visualize the changing of effective receptive field (ERF) under varying autoregressive coefficients in ARMA networks, verifying the Theorem 4 that ARMA layers expand the ERF with large autoregressive coefficients. **(2)** We then exhibit the necessity of our proposed re-parameterization mechanism (Theorem 11) in stabilizing the training of ARMA networks. **(3)** With stability guarantee, we apply ARMA networks to a challenging dense-prediction task – pixel-level video prediction, and we show ARMA networks outperform the state-of-the-art Conv-LSTM based models (Xingjian et al., 2015). **(4)** We interpret the varying performance of ARMA networks on different tasks by visualizing the histograms of the learned autoregressive coefficients.

6.1. Visualization of the Effective Receptive Field

We visualize the ERF of ARMA networks and compare it against the one of traditional convolutional networks, under various depths. Shown in Figure 3, as the autoregressive coefficients in ARMA get larger, the size of the ERF increases. Notice that when the autoregressive coefficients are all zeros, an ARMA network reduces to a traditional convolutional network. The results in Figure 3 also indicate that the ERF expands as the networks get deeper, and ARMA's ability to expand the ERF *increases* as the networks get deeper.

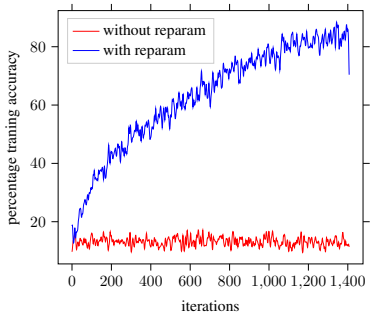
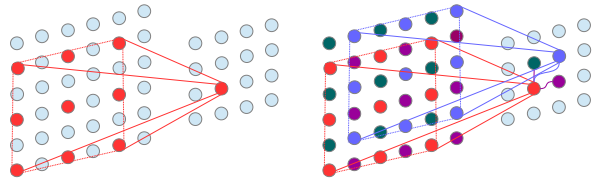


Figure 4. The learning curves with and without our proposed re-parameterization mechanism on an ARMA network with a backbone VGG-11 architecture for CIFAR-10.

6.2. Stabilization of ARMA networks

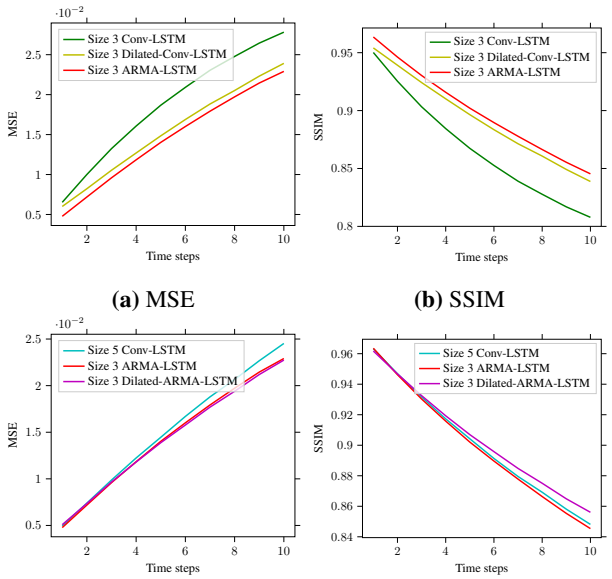
The major challenge for prediction and learning in ARMA network is to constrain its autoregressive coefficients to avoid model instability and divergence in optimization. We compare the learning curves using the re-parameterization v.s. not using the re-parameterization to test our stabilization mechanism in Theorem 11. As shown in Figure 4, the training quickly converges under our proposed re-parameterization mechanism and stability of the network is guaranteed. However without the re-parameterization mechanism, a naive training of ARMA networks diverges quickly, verifying our theory. Therefore, our contribution on stabilization is significant and could potentially be applied

to other networks with constraints.



(a) Dilated convolutional layer. **(b)** Dilated ARMA layer.

Figure 5. Comparison of original dilated convolution and dilated ARMA convolution. The autoregression in ARMA model fills the gaps created by dilated convolution.



(a) MSE **(b)** SSIM

(c) MSE **(d)** SSIM

Figure 6. Performance comparison of our ARMA and our dilated ARMA networks v.s. the state-of-the-art Conv-LSTM, dilated Conv-LSTM baselines for pixel-level video prediction. In the first row, **(a)(b)** illustrate the comparison between baseline Conv-LSTM, baseline dilated Conv-LSTM and our ARMA-LSTM, all with moving average kernel size 3. **(c)(d)** illustrate the comparison between baseline convolutional LSTM with moving average kernel of size 5, our ARMA-LSTM and our dilated ARMA-LSTM with moving average kernel of size 3. Lower MSE values (in 10^{-3}) or higher SSIM values indicate better performance.

6.3. Multi-frame Video Prediction

We evaluate the expressive power of ARMA networks in a challenging dense prediction problem – multi-frames *pixel-level* video prediction on the Moving-MNIST-2 dataset.

Task. The Moving-MNIST-2 dataset is generated by moving two digits of size 28×28 in MNIST dataset within a 64×64 black canvas. These digits are placed at a random initial location, and move with constant velocity in the canvas and bounce when they reach the boundary. Following Wang et al. (2018b), we generate 10,000 videos for training, 3,000 for validation, and 5,000 for test with default parameters in the public generator (Github Repo). All models are trained to predict 10 frames given 10 input frames, and we evaluate their performance based on the metrics of mean square error (MSE), peak signal-noise ratio (PSNR) and structure similarity (SSIM) (Wang et al., 2004).

Model	MA	AR	dil.	params.	MSE	PSNR	SSIM
Conv-LSTM (size 3)	3	1	1	0.887M	18.75	18.24	0.867
Conv-LSTM (size 5)	5	1	1	2.462M	15.23	19.58	0.901
Dilated Conv-LSTM	3	2	1	0.887M	15.47	19.16	0.893
ARMA-LSTM (size 3)	3	2	1	0.893M	14.54	19.72	0.899
Dilated ARMA-LSTM	3	3	2	0.893M	**14.46**	**19.72**	**0.904**

Table 2. Comparison of 10 frames prediction on Moving-MNIST-2 test set, where MA, AR denote the size for moving-average and autoregressive kernels respectively, and dil. is the dilation in the moving-average kernel. Lower MSE values (in 10^{-3}) or higher PSNR, SSIM values indicate better performance. The results are average statistics over 10 predicted frames.

Model Architecture. (1) **Baselines.** We use Conv-LSTM (Xingjian et al., 2015) as the baseline model, and in particular we choose a state-of-the-art 12-layers Conv-LSTM (Byeon et al., 2018) as our backbone architecture. We consider three different convolutions in the baseline networks: (a) traditional convolution with size 3×3, (b) traditional convolution with size 5×5, and (c) 2-dilated convolution with size 3×3. More details of the baseline networks are included in Appendix D.1.

(2) **Our architectures.** Our ARMA networks use the backbone architecture, but replace their convolutional layers with ARMA layers. We set the kernel size for both moving-average and autoregressive coefficient to 3×3 in the ARMA networks. In addition, we propose combining ARMA layer with dilated convolutions, so called dilated-ARMA layer. Figure 5b illustrates how the gridding artifacts in dilated convolutions, where adjacent units in the output are computed from completely disjoint pixels in the input, are addressed by our ARMA model: with the interconnections among neurons in the output layer, the each unit in the output receive information from all pixels in a local region, therefore adjacent units are on longer computed from separate sets of pixels. Both baselines and our ARMA networks are trained by ADAM with learning rate 10^{-3} for 400 epochs, and the detailed learning strategy is described in Appendix D.1.

ARMA outperforms networks with similar size: As shown in Table 2 (and Figure 6), our ARMA network with kernel size 3×3 substantially outperforms two baselines: Conv-LSTM with traditional and 2-dilated convolutions of size 3×3 . The ARMA layers improve the performance of Conv-LSTM with traditional 3×3 convolutions by 3.7% in SSIM, and

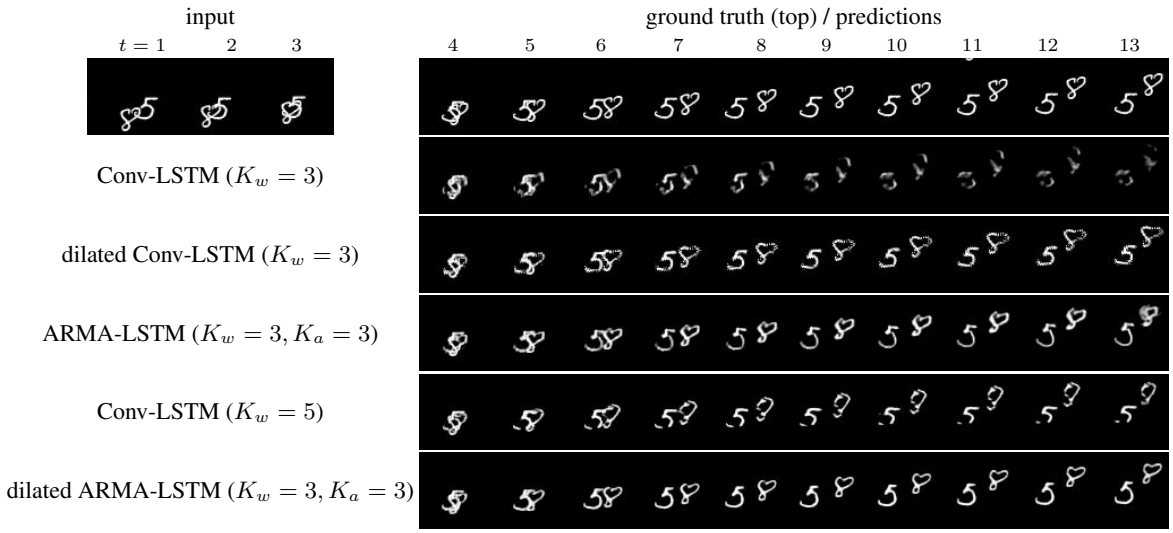


Figure 7. 10 frames prediction on Moving-MNIST-2 given 10 input frames. The first row contains the groudth-truth frames.

also outperformss the dilated convolution (Figure 6(a)(b), and the upper block in Table 2).

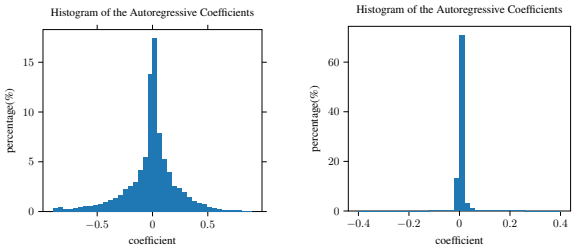
ARMA and dilated ARMA outperform larger networks : ARMA network with filter size 3 achieves comparable performs with the convolutional LSTM model with filter size 5 as shown in Figure 6(c)(d) and lower block in Table 2. In other words, ARMA network uses 63.7% fewer parameters to achieve similar performance as in a convolutional LSTM model. When combined with dilation, our dilated ARMA-LSTM also outperforms the Conv-LSTM with 5×5 convolutions, using 63.7% fewer parameters.

Visualization of video prediction: A visualization of the predictions is in Figure 7. Notice that dilated ARMA-LSTM does not have gridding artifacts as in dilated Conv-LSTM.

6.4. Interpretation of ARMA's Performance

To explain why ARMA networks are able to achieve impressive performance in video prediction, we visualize the the histogram of the autoregressive coefficients after training. In Figure 8, we compare the *histogram* of coefficients of trained ARMA network for **video prediction task** against the coefficients of trained ARMA network for **image classification task** (In Appendix D.2, we demonstrate that ARMA networks achieve comparable or slightly better results in image classification, when ARMA layers are incorporated into benchmark architectures VGG and ResNet).

As motivated at the beginning of the paper, dense prediction such as video prediction requires global information and larger effective receptive field to achieve higher performance, and the large autoregressive coefficients in ARMA network allow expansion of the ERF. In simpler tasks such as image classification, expanded ERF is not required, and an ARMA network automatically learns that, as the autoregressive coefficients learned are much smaller (close to 0) compared to video prediction.



(a) Video Prediction **(b)** Image Classification

Figure 8. Histogram plot of the autoregressive coefficients in a trained ARMA network for video prediction vs for image classification. The sizes of moving-average filter and the autoregressive filters are 3. The x-label range of (b) is half the size of (a).

7. Conclusion

We propose a novel *ARMA* layer to introduce interconnections between output neurons, which naturally expands the effective receptive field of *ARMA networks*. We address the computational problem by deriving efficient FFT algorithms for both forward and backward passes. We devise *separable ARMA layer*, which guarantees the stability of ARMA networks by a re-parameterization mechanism. We apply ARMA networks to pixel-level video prediction problem, and show that our proposed models outperform the state-of-the-art convolutional LSTM networks.

References

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pp. 1171–1179, 2015.

Bertsekas, D. P. and Scientific, A. *Convex optimization algorithms*. Athena Scientific Belmont, 2015.

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

Byeon, W., Breuel, T. M., Raue, F., and Liwicki, M. Scene labeling with lstm recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3547–3555, 2015.

Byeon, W., Wang, Q., Kumar Srivastava, R., and Koumoutsakos, P. Contextvp: Fully context-aware video prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 753–769, 2018.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. 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, 2017a.

Chen, L.-C., Papandreou, G., Schroff, F., and Adam, H. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017b.

Dai, J., Li, Y., He, K., and Sun, J. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pp. 379–387, 2016.

Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., and Wei, Y. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 764–773, 2017.

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

Giusti, A., Cireşan, D. C., Masci, J., Gambardella, L. M., and Schmidhuber, J. Fast image scanning with deep max-pooling convolutional neural networks. In *2013 IEEE International Conference on Image Processing*, pp. 4034–4038. IEEE, 2013.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Hamaguchi, R., Fujita, A., Nemoto, K., Imaizumi, T., and Hikosaka, S. Effective use of dilated convolutions for segmenting small object instances in remote sensing imagery. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pp. 1442–1450. IEEE, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Holschneider, M., Kronland-Martinet, R., Morlet, J., and Tchamitchian, P. A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets*, pp. 286–297. Springer, 1990.

Jeon, Y. and Kim, J. Active convolution: Learning the shape of convolution for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4201–4209, 2017.

Kalchbrenner, N., Danihelka, I., and Graves, A. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*, 2015.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012.

Li, H., Zhao, R., and Wang, X. Highly efficient forward and backward propagation of convolutional neural networks for pixelwise classification. *arXiv preprint arXiv:1412.4526*, 2014.

Li, Y., Chen, Y., Wang, N., and Zhang, Z. Scale-aware trident networks for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6054–6063, 2019.

Lim, J. S. Two-dimensional signal and image processing. *Englewood Cliffs, NJ, Prentice Hall, 1990, 710 p.*, 1990.

Liu, S., Pan, J., and Yang, M.-H. Learning recursive filters for low-level vision via a hybrid neural network. In *European Conference on Computer Vision*, pp. 560–576. Springer, 2016.

Liu, S., De Mello, S., Gu, J., Zhong, G., Yang, M.-H., and Kautz, J. Learning affinity via spatial propagation networks. In *Advances in Neural Information Processing Systems*, pp. 1520–1530, 2017.

Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

Luo, W., Li, Y., Urtasun, R., and Zemel, R. Understanding the effective receptive field in deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 4898–4906, 2016.

Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

Oppenheim, A. V. and Schafer, R. W. *Discrete-time signal processing*. Pearson Education, 2014.

Papandreou, G., Kokkinos, I., and Savalle, P.-A. Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 390–399, 2015.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Stollenga, M. F., Byeon, W., Liwicki, M., and Schmidhuber, J. Parallel multi-dimensional lstm, with application to fast biomedical volumetric image segmentation. In *Advances in neural information processing systems*, pp. 2998–3006, 2015.

Wang, P., Chen, P., Yuan, Y., Liu, D., Huang, Z., Hou, X., and Cottrell, G. Understanding convolution for semantic segmentation. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pp. 1451–1460. IEEE, 2018a.

Wang, Y., Gao, Z., Long, M., Wang, J., and Yu, P. S. Predrnn++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning. *arXiv preprint arXiv:1804.06300*, 2018b.

Wang, Z. and Ji, S. Smoothed dilated convolutions for improved dense prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2486–2495, 2018.

Wang, Z., Bovik, A. C., Sheikh, H. R., Simoncelli, E. P., et al. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Xingjian, S., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pp. 802–810, 2015.

Yu, F. and Koltun, V. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

Yu, F., Koltun, V., and Funkhouser, T. Dilated residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 472–480, 2017.

Zhu, X., Hu, H., Lin, S., and Dai, J. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9308–9316, 2019.

Appendix: ARMA Nets: Expanding Receptive Field for Dense Prediction

A. Analysis of Effective Receptive Field (ERF)

In this section, we prove Theorems 3 and 4 in subsection 3.2. Both proofs are based on the following theorem on ERF of linear convolutional neural network (Luo et al., 2016).

Theorem 12 (ERF of CNN with infinite horizon). *Consider an L-layer linear convolutional neural network (without activation and pooling functions), where each layer computes a moving-average weighted-sum of its inputs $y_i = \sum_{p=-\infty}^{+\infty} w_p x_{i-p}$. Suppose the weights are non-negative and normalized, i.e. $w_p \geq 0, \forall p$ and $\sum_{p=-\infty}^{+\infty} w_p = 1$. Then the ERF converges to a Gaussian density function when L tends to infinity, with a radius of*

$$r^2(\textit{ERF}) = L\left[\sum_{p=-\infty}^{+\infty} p^2 w_p - \left(\sum_{p=-\infty}^{+\infty} p w_p\right)^2\right] \tag{A.1}$$

Proof. In this particular linear convolutional network, the gradient maps can be computed analytically with chain rule

$$g(i,:) = \underbrace{w^\top * w^\top \cdots * w^\top}_{L \text{ terms}}, \quad \forall i \in \mathbb{Z} \tag{A.2}$$

where $w^\top$ denotes the reversed sequence of w. Notice that **(1)** The gradient maps do not depend on the input, i.e. they are data-independent; **(2)** The gradient maps are identical across different locations in the sequence. Consequently, the ERF is equal to any one gradient map above

$$\text{ERF} = \underbrace{w^\top * w^\top \cdots * w^\top}_{L \text{ terms}} \tag{A.3}$$

The remaining part of the proof makes use of *probabilistic method*, which interprets the shifting at each layer as a random variable. Since the weights at each layer are non-negative and normalized, they can be treated as values in a probability mass function. Concretely, we construct L i.i.d. random variables W^l's such that $\mathbb{P}[W^l = p] = w_{-p}, \forall l \in [L]$, where l indexes the layers. Similarly, we introduce a random variable S^L to represent the ERF, i.e. $\mathbb{P}[S^L = p] = \text{ERF}(p)$. As a result, the radius of ERF is equal to standard deviation of S^L, or equivalently $r^2(\text{ERF}) = \mathbb{V}[S^L]$.

Recall that *addition of independent random variables results in convolution among their probability mass functions*, Equation A.3 implies that S^L is an addition of all W^l's:

$$S^L = \sum_{l=1}^{L} W^l \tag{A.4}$$

Since W^l's are i.i.d. random variables, the variance of S^L is equal to the summation of all variances of W^l's, therefore L times of the variance of any W^l:

$$\mathbb{V}[S^L] = \sum_{l=1}^{L} \mathbb{V}[W^l] = L\left[\mathbb{E}[(W^1)^2] - E[W^1]^2\right] \tag{A.5}$$

$$= L\left[\sum_{p=-\infty}^{+\infty} p^2 w_p - \left(\sum_{p=-\infty}^{+\infty} p w_p\right)^2\right] \tag{A.6}$$

which proves the Equation A.1. Furthermore, the *central limit theorem* shows that $(S^L - \mathbb{E}[S^L])/\sqrt{L}$ converges to a Gaussian random variable if L tends to infinity

$$\frac{S^L - \mathbb{E}[S^L]}{\sqrt{L}} = \frac{\sum_{l=1}^{L}(W^l - \mathbb{E}[W^l])}{\sqrt{L}} \xrightarrow{D} \mathcal{N}(0,1) \tag{A.7}$$

that is, the ERF is approximately a Gaussian density function when the number of layers L is large enough. □

The proofs for both theorems also make heavy use of the first two cases of *Faulhaber's formula*:

$$\sum_{p=0}^{K-1} p = \frac{K(K-1)}{2} \tag{A.8a}$$

$$\sum_{p=0}^{K-1} p^2 = \frac{K(K-1)(2K-1)}{6} \tag{A.8b}$$

as well as the first two cases of *polylogarithm function*:

$$\sum_{p=0}^{+\infty} c^p = \frac{1}{1-c} \tag{A.9a}$$

$$\sum_{p=0}^{+\infty} p c^p = \frac{c}{(1-c)^2} \tag{A.9b}$$

where both equations hold when $|c| < 1$.

A.1. ERF of traditional CNNs

Proof. Theorem 3 can be easily obtained by plugging $w_p = 1/K$ for $p \in [K]$ in Equation A.1.

$$r^2(\text{ERF}) = L\left[\sum_{p=0}^{K-1} \frac{p^2}{K} - \left(\sum_{p=0}^{K-1} \frac{p}{K}\right)^2\right] \tag{A.10}$$

$$= L\left[\frac{1}{K} \cdot \frac{K(K-1)(2K-1)}{6} - \left(\frac{1}{K} \cdot \frac{K(K-1)}{2}\right)^2\right] \tag{A.11}$$

$$= \frac{L\left(K^2-1\right)}{12} \tag{A.12}$$

where Equation A.11 uses (A.8a) and (A.8b). Therefore,

$$r(\text{ERF}) = \sqrt{L} \cdot \sqrt{\frac{K^2-1}{12}} = O\left(K\sqrt{L}\right) \tag{A.13}$$

which completes the proof. □

A.2. ERF of ARMA networks

In the part, we provide a detailed proof of Theorem 4. The calculation consists of two major steps: **(1)** we introduce the inverse of convolution and convert the ARMA model to a moving-average model:

$$a * y = w * x \implies y = f * x \tag{A.14}$$

where f is convolution of infinite number of coefficients. **(2)** We plug the coefficients of f into Equation A.1, which yields Equation 6 after some careful calculation.

Definition 13 (Inverse of convolution). *Given a convolution (with coefficients) a, its inverse convolution $\bar{a}$ is defined such that $a * \bar{a} = \bar{a} * a = \delta$ is an identical mapping, i.e.*

$$\sum_{p=-\infty}^{+\infty} a_{i-p}\bar{a}_p = \delta_i = \begin{cases} 1 & i = 0 \\ 0 & i \neq 0 \end{cases} \tag{A.15}$$

Notice that the inverse does not always exist for any convolution a. A necessary and sufficient condition for invertibility of a is that its Fourier transform is non-zero everywhere (Oppenheim & Schafer, 2014).

Proof. Let $f = \bar{a} * w$, we have

$$\begin{aligned} y &= \delta * y = (\bar{a} * a) * y = \bar{a} * (a * y) \\ &= \bar{a} * (w * x) = (\bar{a} * w) * x = f * x \end{aligned} \tag{A.16}$$

Given $a_0 = 1$, $a_1 = -c$ (since a is used as the vector of autoregressive coefficients, we use c for a_1 for the appendix), the inverse of a can be computed as $\bar{a}_p = c^p$ for $p \geq 0$. Let u be a *step* function, defined as

$$u_p = \begin{cases} 1 & p \geq 0 \\ 0 & p < 0 \end{cases} \tag{A.17}$$

Then both w and $\bar{a}$ can be represented in term of u as

$$w_p = \frac{1}{K}(u_p - u_K) \tag{A.18a}$$
$$\bar{a}_p = c^p u_p \tag{A.18b}$$

In order to compute the convolution between $\bar{a}$ and w, we will make use of an useful equality:

$$\{c^p u_p\} * \{u_p\} = \left\{ \frac{1 - c^p}{1 - c} u_p \right\} \tag{A.19}$$

Here we use $\{u_p\}$ to denote a sequence indexed by p (notice that convolution is defined on two sequences instead of two scalars). With this equality, we compute f as follows:

$$f_p = \frac{1}{K}\left\{\{u_p - u_{p-K}\} * \{c^p u_p\}\right\}_p \tag{A.20}$$
$$= \frac{1}{K}\left\{ \frac{1 - c^p}{1 - c} u_p - \frac{1 - c^{p-K}}{1 - c} u_{p-K} \right\}_p \tag{A.21}$$
$$= \begin{cases} 0 & p < 0 \\ \frac{1}{K} \cdot \frac{1 - c^p}{1 - c} & 0 \leq p < K \\ \frac{1}{K} \cdot \frac{c^{p-K} - c^p}{1 - c} & p \geq K \end{cases} \tag{A.22}$$

Notice that f has infinite number of coefficients. Next, we define the *moments* of f as

$$M_i = \sum_{p=-\infty}^{+\infty} p^i f_p, \ \forall i \geq 0 \tag{A.23}$$

with which we rewrite Equation A.1 in Theorem 12 as

$$r^2(\text{ERF}) = L\left[\left(\frac{M_2}{M_0}\right) - \left(\frac{M_1}{M_0}\right)^2\right] \tag{A.24}$$

The remaining parts are to compute the analytic forms of M_0, M_1 and M_2 in terms of K and c respectively.

$$K \cdot M_0 = \sum_{p=0}^{K-1} \frac{1 - c^p}{1 - c} + \sum_{p=K}^{+\infty} \frac{c^{p-K} - c^p}{1 - c} \tag{A.25}$$
$$= \sum_{p=0}^{K-1} \frac{1 - c^p}{1 - c} + \sum_{p=0}^{+\infty} \frac{c^p - c^{p+K}}{1 - c} \tag{A.26}$$
$$= \sum_{p=0}^{K-1} \frac{1}{1 - c} - \sum_{p=K}^{+\infty} \frac{c^p}{1 - c} + \sum_{p=0}^{+\infty} \frac{c^{p+K}}{1 - c} \tag{A.27}$$
$$= \sum_{p=0}^{K-1} \frac{1}{1 - c} = \frac{K}{1 - c} \tag{A.28}$$

Notice that the last two terms in Equation A.27 are equal by a simple change of variable. With $M_0 = 1/(1 - c)$, it is easier to compute M_1/M_0 directly.

$$K \cdot \frac{M_1}{M_0} = \sum_{p=0}^{K-1} p\,(1 - c^p) + \sum_{p=K}^{+\infty} p\left(c^{p-K} - c^p\right) \tag{A.29}$$
$$= \sum_{p=0}^{K-1} p + \sum_{p=K}^{+\infty} p c^{p-K} - \sum_{p=0}^{+\infty} p c^p \tag{A.30}$$
$$= \sum_{p=0}^{K-1} p + \sum_{p=0}^{+\infty} (p + K) c^p - \sum_{p=0}^{+\infty} p c^p \tag{A.31}$$
$$= \sum_{p=0}^{K-1} p + K \sum_{p=0}^{+\infty} c^p \tag{A.32}$$
$$= \frac{K(K-1)}{2} + \frac{K}{1 - c} \tag{A.33}$$

where the last equation uses (A.8a) and (A.9a). Therefore, the normalized first moment takes the form

$$\frac{M_1}{M_0} = \frac{K-1}{2} + \frac{1}{1-c} \tag{A.34}$$

Similarly, we can calculate M_2/M_0.

$$K \cdot \frac{M_2}{M_0} = \sum_{p=K}^{+\infty} p^2 + \sum_{p=K}^{+\infty} p^2 c^{p-K} - \sum_{p=0}^{+\infty} p^2 c^p \tag{A.35}$$

$$= \sum_{p=K}^{+\infty} p^2 + \sum_{p=0}^{+\infty} (p+K)^2 c^p - \sum_{p=0}^{+\infty} p^2 c^p \tag{A.36}$$

$$= \sum_{p=K}^{+\infty} p^2 + 2K \sum_{p=0}^{+\infty} p c^p + K^2 \sum_{p=0}^{+\infty} c^p \tag{A.37}$$

$$= \frac{K(K-1)(2K-1)}{6} + 2K \cdot \frac{c}{(1-c)^2} + K^2 \cdot \frac{1}{1-c} \tag{A.38}$$

where the last equation utilizes (A.8b), (A.9a) and (A.9b) simultaneously. Therefore we have the normalized second moment as

$$\frac{M_2}{M_0} = \frac{(K-1)(2K-1)}{6} + \frac{2c}{(1-c)^2} + \frac{K}{1-c} \tag{A.39}$$

Plugging (A.28), (A.34) and (A.39) into Equation A.24,

$$r^2(\text{ERF}) = L\left[\frac{(K-1)(2K-1)}{6} + \frac{2c}{(1-c)^2} + \frac{K}{1-c}\right] - \left(\frac{K-1}{2} + \frac{1}{1-c}\right)^2 \tag{A.40}$$

$$= L\left[\frac{K^2-1}{12} + \frac{c}{(1-c)^2}\right] \tag{A.41}$$

we complete the proof of Theorem 4. □

B. Backpropagation of ARMA Layers

In this section, we will prove a general theorem for backpropagation in ARMA models. To keep the notations simple, we derive the backpropagation equations for ARMA models with one dimension input/output and one channel. However, the techniques in the proof can be trivially extended to general ARMA models with high-dimensional input/output with multiple channels.

Theorem 14 (Backpropagation in an ARMA model). *Consider an ARMA model $a * y = w * x$, where a and w are moving-average and autoregressive coefficients respectively, the gradients $\partial\mathcal{L}/\partial x$, $\partial\mathcal{L}/\partial w$ and $\partial\mathcal{L}/\partial a$ can be computed from $\partial\mathcal{L}/\partial y$ as follows:*

$$a^\top * \frac{\partial \mathcal{L}}{\partial x} = w^\top * \frac{\partial \mathcal{L}}{\partial y} \tag{B.1a}$$

$$a^\top * \frac{\partial \mathcal{L}}{\partial a} = -y^\top * \frac{\partial \mathcal{L}}{\partial y} \tag{B.1b}$$

$$a^\top * \frac{\partial \mathcal{L}}{\partial w} = x^\top * \frac{\partial \mathcal{L}}{\partial y} \tag{B.1c}$$

where $a^\top$, $w^\top$ and $y^\top$ denote the reversed sequences of a, w and y respectively.

Notice that Theorem 6 is special case of Theorem 14: Equation 11a is proved by Equation B.1b, and Equation 11a is proved by Equation B.1a.

We provide two different proofs of Theorem 14. **(1)** The analysis in our first proof is based on real numbers, and applicable to arbitrary types of convolution. **(2)** If the convolution is *circular* (as in the implementation of this paper), we provide a simpler proof using discrete Fourier transform (therefore complex numbers). The second proof also suggests a FFT-based algorithm to compute the backpropagation equations efficiently.

B.1. Proof in Real Numbers $\mathbb{R}$

Before we prove the theorem, we first prove a useful lemma on the inverse of transposed convolution.

Lemma 15 (Inverse of transposed convolution). *Given a convolution (with coefficients) a, the operations of inversion and transposition are exchangeable,*

$$\overline{a^\top} = \bar{a}^\top \tag{B.2}$$

that is, the inverse of transposed convolution is equal to the transposed inverse convolution.

Proof. The lemma can be easily proved using the definitions of inverse convolution and transposed convolution.

$$\sum_{p=-\infty}^{+\infty} a_p^\top \bar{a}_{i-p}^\top = \sum_{p=-\infty}^{+\infty} a_{-p} \bar{a}_{p-i} = \delta_{-i} = \delta_i \quad \forall i \tag{B.3}$$

which shows the inverse of $a^\top$, i.e. $\overline{a^\top}$, is equal to $\bar{a}^\top$. □

Proof. To begin with, we write the ARMA model $a * y = w * x$ in its summation form:

$$\sum_{q=-\infty}^{+\infty} a_q y_{i-q} = \sum_{p=-\infty}^{+\infty} w_p x_{i-p} \quad \forall i \tag{B.4}$$

Taking derivative w.r.t. a_r on both hand sides, and notice that the right hand side is a constant w.r.t. a_r, we have

$$\frac{\partial \left(\sum_{q=-\infty}^{+\infty} a_q y_q\right)}{\partial a_r} = 0 \quad \forall i, r \tag{B.5}$$

By *implicit function theorem*, the left hand side can be further expanded as

$$\frac{\partial\left(\sum_{q=-\infty}^{+\infty} a_q y_{i-q}\right)}{\partial a_r} = \sum_{q=-\infty}^{+\infty} \frac{\partial\left(a_q y_{i-q}\right)}{\partial a_r} \tag{B.6}$$

$$= \sum_{q\neq r} a_q \frac{\partial y_{i-q}}{\partial a_r} + \left(y_{i-r} + a_r \frac{\partial y_{i-r}}{\partial a_r}\right) \tag{B.7}$$

$$= \sum_{q=-\infty}^{+\infty} a_q \frac{\partial y_{i-q}}{\partial a_r} + y_{i-r} = 0 \quad \forall i, r \tag{B.8}$$

Rearranging the equation above, we have

$$\sum_{q=-\infty}^{+\infty} a_q \frac{\partial y_{i-q}}{\partial a_r} = -y_{i-r} \quad \forall i, r \tag{B.9a}$$

Repeating the procedure twice for the derivatives w.r.t. w_r and x_r, we have two other similar equations:

$$\sum_{q=-\infty}^{+\infty} a_q \frac{\partial y_{i-q}}{\partial w_r} = x_{i-r} \quad \forall i, r \tag{B.9b}$$

$$\sum_{q=-\infty}^{+\infty} a_q \frac{\partial y_{i-q}}{\partial x_r} = a_{i-r} \quad \forall i, r \tag{B.9c}$$

Since the Eqs. (B.9a), (B.9b) and (B.9c) take the same form, we only precede with Eq. (B.9b) and obtain $\partial\mathcal{L}/\partial w$.

Notice that Eq. (B.9b) can be rewritten as

$$\sum_{q=-\infty}^{+\infty} a_{i-q} \frac{\partial y_q}{\partial w_r} = x_{i-r} \quad \forall i, r \tag{B.10}$$

by changing variable q to $i-q$. Since Equation B.10 holds for any i, we further change i to $i-l$ on both hand sides:

$$\sum_{q=-\infty}^{+\infty} a_{i-q-l} \frac{\partial y_q}{\partial w_r} = x_{i-r-l} \quad \forall i, r, l \tag{B.11}$$

Now we convolve both hand sides with $\bar{a}$, the inverse of a. Then $\forall i, r$, we have

$$\sum_{l=-\infty}^{+\infty} \bar{a}_l \left(\sum_{q=-\infty}^{+\infty} a_{i-q-l} \frac{\partial y_q}{\partial w_r}\right) = \sum_{l=-\infty}^{+\infty} \bar{a}_l x_{i-r-l} \tag{B.12}$$

$$\sum_{q=-\infty}^{+\infty} \left(\sum_{l=-\infty}^{+\infty} \bar{a}_l a_{i-q-l}\right) \frac{\partial y_q}{\partial w_r} = \sum_{l=-\infty}^{+\infty} \bar{a}_l x_{i-r-l} \tag{B.13}$$

$$\frac{\partial y_i}{\partial w_r} = \sum_{q=-\infty}^{+\infty} \delta_{i-q} \frac{\partial y_q}{\partial w_r} = \sum_{l=-\infty}^{+\infty} \bar{a}_l x_{i-r-l} \tag{B.14}$$

Subsequently, we apply the chain rule to obtain $\partial\mathcal{L}/\partial w_r$

$$\frac{\partial\mathcal{L}}{\partial w_r} = \sum_{i=-\infty}^{+\infty} \frac{\partial y_i}{\partial w_r} \frac{\partial\mathcal{L}}{\partial y_i} \quad \forall r \tag{B.15}$$
$$= \sum_{i=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} \bar{a}_l x_{i-r-l} \frac{\partial\mathcal{L}}{\partial y_i}$$

Finally, we convolve both hand sides with $a^\top$, the transpose of a, to obtain the ARMA form of backpropagation rule.

$$\sum_{r=-\infty}^{+\infty} a_{j-r}^\top \frac{\partial\mathcal{L}}{\partial w_r}$$

$$= \sum_{r=-\infty}^{+\infty} a_{j-r}^\top \left(\sum_{i=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} \bar{a}_l x_{i-r-l} \frac{\partial\mathcal{L}}{\partial y_i}\right) \tag{B.16}$$

$$= \sum_{r=-\infty}^{+\infty} \sum_{i=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} a_{j-r}^\top \bar{a}_l x_{i-r-l} \frac{\partial\mathcal{L}}{\partial y_i} \tag{B.17}$$

$$= \sum_{r=-\infty}^{+\infty} \sum_{i=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} a_{j-r}^\top \bar{a}_{l-r} x_{i-l} \frac{\partial\mathcal{L}}{\partial y_i} \tag{B.18}$$

$$= \sum_{r=-\infty}^{+\infty} \sum_{i=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} a_{j-r}^\top \bar{a}_{r-l}^\top x_{l-i}^\top \frac{\partial\mathcal{L}}{\partial y_i} \tag{B.19}$$

$$= \sum_{i=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} \left(\sum_{r=-\infty}^{+\infty} a_{j-r}^\top \bar{a}_{r-l}^\top\right) x_{l-i}^\top \frac{\partial\mathcal{L}}{\partial y_i} \tag{B.20}$$

$$= \sum_{i=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} \delta_{j-l} x_{l-i}^\top \frac{\partial\mathcal{L}}{\partial y_i} \tag{B.21}$$

$$= \sum_{i=-\infty}^{+\infty} x_{j-i}^\top \frac{\partial\mathcal{L}}{\partial y_i} \quad \forall j \tag{B.22}$$

where the second last equality uses Lemma 15. Therefore, we prove $a^\top * \frac{\partial\mathcal{L}}{\partial w} = x^\top * \frac{\partial\mathcal{L}}{\partial y}$, i.e. Eq. (B.1c) in the theorem. Eqs. (B.1b) and (B.1a) can be proved similarly. □

B.2. Proof in Complex Numbers $\mathbb{C}$

Proof. If both convolutions in $a * y = w * x$ are circular with period N, the celebrated *convolution theorem* relates the discrete Fourier transform of a, y, w and x with

$$A_l Y_l = W_l X_l \quad \begin{cases} A_l = \sum_{n=0}^{N-1} a_n \omega_N^{nl} \\ Y_l = \sum_{n=0}^{N-1} y_n \omega_N^{nl} \\ W_l = \sum_{n=0}^{N-1} w_n \omega_N^{nl} \\ X_l = \sum_{n=0}^{N-1} x_n \omega_N^{nl} \end{cases} \tag{B.23}$$

where $\omega_N = \exp(-\mathrm{j}2\pi/N)$ is the N-th root of unity. For brevity, we only prove the most difficult equation $a^\top * \frac{\partial \mathcal{L}}{\partial a} = -y^\top * \frac{\partial \mathcal{L}}{\partial y}$ (Eq. (B.1b)) here, and the proofs for the other two equations can be obtained with minor modification.

Taking derivative w.r.t. A_k on both hand sides, we have

$$\begin{cases} A_l \dfrac{\partial Y_l}{\partial A_k} = 0 & l \neq k \\ A_l \dfrac{\partial Y_l}{\partial A_k} + Y_k = 0 & l = k \end{cases} \tag{B.24}$$

Since $A_l \neq 0, \forall l$, the equation can be simplified as

$$\frac{\partial Y_l}{\partial A_k} = \begin{cases} 0 & l \neq k \\ -\dfrac{Y_k}{A_k} & l = k \end{cases} \tag{B.25}$$

Then we apply chain rule to obtain the gradient of A_k

$$\frac{\partial \mathcal{L}}{\partial A_k} = \sum_{l=0}^{N-1} \frac{\partial \mathcal{L}}{\partial Y_l}\frac{\partial Y_l}{\partial A_k} = -\frac{Y_k}{A_k}\frac{\partial \mathcal{L}}{\partial Y_k} \tag{B.26}$$

Again, since $A_k \neq 0, \forall k$, we can simplify the equation as

$$A_k \frac{\partial \mathcal{L}}{\partial A_k} = -Y_k \frac{\partial \mathcal{L}}{\partial Y_k} \tag{B.27}$$

To precede, we apply the chain rule one more time to obtain the derivatives w.r.t. a_n and y_n

$$\frac{\partial \mathcal{L}}{\partial a_n} = \sum_{k=0}^{N-1} \frac{\partial \mathcal{L}}{\partial A_k}\frac{\partial A_k}{\partial a_n} = \sum_{k=0}^{N-1} \frac{\partial \mathcal{L}}{\partial A_k}\omega_N^{kn} \tag{B.28a}$$

$$\frac{\partial \mathcal{L}}{\partial y_n} = \sum_{k=0}^{N-1} \frac{\partial \mathcal{L}}{\partial Y_k}\frac{\partial Y_k}{\partial y_n} = \sum_{k=0}^{N-1} \frac{\partial \mathcal{L}}{\partial Y_k}\omega_N^{kn} \tag{B.28b}$$

With the equations above, the convolution between $a^\top$ and $\partial\mathcal{L}/\partial a$ can be rewritten as

$$\sum_{n=0}^{N-1} a_{i-n}^\top \frac{\partial \mathcal{L}}{\partial a_n} = \sum_{n=0}^{N-1} a_{n-i}\frac{\partial \mathcal{L}}{\partial a_n} \tag{B.29}$$

$$= \sum_{n=0}^{N-1} a_{n-i}\left(\sum_{k=0}^{N-1}\frac{\partial \mathcal{L}}{\partial A_k}\omega_N^{kn}\right) \tag{B.30}$$

$$= \sum_{k=0}^{N-1}\left(\sum_{n=0}^{N-1} a_{n-i}\omega_N^{k(n-i)}\right)\frac{\partial \mathcal{L}}{\partial A_k}\omega_N^{ki} \tag{B.31}$$

$$= \sum_{k=0}^{N-1} A_k \frac{\partial \mathcal{L}}{\partial A_k}\omega_N^{ki} \tag{B.32}$$

With an identical argument, we can rewrite the convolution between $y^\top$ and $\partial\mathcal{L}/\partial y$ as

$$\sum_{n=0}^{N-1} y_{i-n}^\top \frac{\partial \mathcal{L}}{\partial y_n} = \sum_{k=0}^{N-1} Y_k \frac{\partial \mathcal{L}}{\partial Y_k}\omega_N^{ki} \tag{B.33}$$

Recall the relation in Eq. (B.27), we have

$$\sum_{n=0}^{N-1} a_{i-n}^\top \frac{\partial \mathcal{L}}{\partial a_n} = -\sum_{n=0}^{N-1} y_{i-n}^\top \frac{\partial \mathcal{L}}{\partial y_n} \tag{B.34}$$

i.e. $a^\top * \frac{\partial \mathcal{L}}{\partial a} = -y^\top * \frac{\partial \mathcal{L}}{\partial y}$, which completes the proof. □

Notice that Equation B.27 also suggests an efficient algorithm to evaluate the backpropagation equation using FFT.

C. Stability of ARMA Layers

In this section, we will prove the main Theorem 9 in section 5. The section is organized in three subsections: In subsection C.1, we prove a lemma that allows us to reduce the analysis of a complex model to the ones of its submodules; In subsection C.2, we repeatedly use the lemma and reduce to the analysis of an ARMA layer to the one of a length-3 filter; Lastly in subsection C.3, we prove a theorem on the stability of a length-3 filter.

C.1. Algebra of BIBO stability

The following lemma presents that the BIBO stability is preserved under simple algebraic operations of cascade, addition and concatenation. Using this lemma one could decompose a complex model into simpler submodules, which allows for analysis of BIBO stability of the complex model.

Lemma 16 (Preserved BIBO Stability). *BIBO stability is preserved under the operations of cascade, addition and concatenation. Suppose f and g are two BIBO stable models, and consider three compound models:* **(1)** *$h_1 = g \circ f$ is a cascaded model such that $y = h_1(x) = g(f(x))$,* **(2)** *$h_2 = f + g$ is a parallel model such that $y = h_2(x) = f(x) + g(x)$,* **(3)** *$h_3 = f \otimes g$ is a concatenated model such that $y = [y^1, y^2] = h_3([x^1, x^2]) = [f(x^1), g(x^2)]$, h_1, h_2 and h_3 are all BIBO stable.*

Proof. (1) *Cascaded model $h_1 = g \circ f$:* $y = h_1(x) = f(g(x))$. Let $t = h(x)$ denote the intermediate result returned by the model f. Since f is BIBO stable, we have

$$\begin{aligned} &(\exists B_1 > 0, |x_i| < B_1, \forall i \in \mathbb{Z}) \\ \Longrightarrow\ &(\exists B_0 > 0, |t_i| < B_0, \forall i \in Z) \end{aligned} \tag{C.1a}$$

Similarly, since g is BIBO stable, we further have

$$\begin{aligned} &(\exists B_0 > 0, |t_i| < B_0, \forall i \in \mathbb{Z}) \\ \Longrightarrow\ &(\exists B_2 > 0, |y_i| < B_2, \forall i \in \mathbb{Z}) \end{aligned} \tag{C.1b}$$

Combining both Equations (C.1a) and (C.1b), we achieve

$$\begin{aligned} &(\exists B_1 > 0, |t_i| < B_1, \forall i \in \mathbb{Z}) \\ \Longrightarrow\ &(\exists B_2 > 0, |y_i| < B_2, \forall i \in Z) \end{aligned} \tag{C.2}$$

which is the definition of BIBO stability for model h_1.

(2) Parallel model $h_2 = f + g$: $y = h_2(x) = f(x) + g(x)$. Let $u = f(x)$ and $v = g(x)$ be the outputs returned by the models f and g. Since both f and g are BIBO stable, we have the following two relations:

$$\begin{aligned}&(\exists B_1 > 0, |x_i| < B_1, \forall i \in \mathbb{Z})\\ \Longrightarrow\ &(\exists B_{21} > 0, |u_i| < B_{21}, \forall i \in Z)\end{aligned} \tag{C.3a}$$

$$\begin{aligned}&(\exists B_1 > 0, |x_i| < B_1, \forall i \in \mathbb{Z})\\ \Longrightarrow\ &(\exists B_{22} > 0, |v_i| < B_{22}, \forall i \in Z)\end{aligned} \tag{C.3b}$$

Combining both Equations (C.3a) and (C.3b), we have

$$\begin{aligned}&(\exists B_1 > 0, |t_i| < B_0, \forall i \in \mathbb{Z})\\ \Longrightarrow\ &(|y_i| < B_2 = B_{21} + B_{22}, \forall i \in Z)\end{aligned} \tag{C.4}$$

We achieve the definition BIBO stability for model h_2.

(3) Concatenated model $y = f \otimes g$: $y = [y^1, y^2] = h([x^1, x^2]) = [f(x^1), g(x^2)]$*:* Since f and g are both BIBO stable, we have the following two relations:

$$\begin{aligned}&(\exists B_1 > 0, |x_i| < B_1, \forall i \in \mathbb{Z})\\ \Longrightarrow\ &(\exists B_{21} > 0, |y_i^1| < B_{21}, \forall i \in Z)\end{aligned} \tag{C.5a}$$

$$\begin{aligned}&(\exists B_1 > 0, |x_i| < B_1, \forall i \in \mathbb{Z})\\ \Longrightarrow\ &(\exists B_{22} > 0, |y_i^2| < B_{22}, \forall i \in Z)\end{aligned} \tag{C.5b}$$

Again, combining both equations we have

$$\begin{aligned}&(\exists B_1 > 0, |x_i| < B_1, \forall i \in \mathbb{Z})\\ \Longrightarrow\ &(|y_i^2| < B_2 = \max(B_{21}, B_{22}), \forall i \in Z)\end{aligned} \tag{C.6}$$

And we achieve the BIBO stability for model h_3. □

C.2. Reduction of an ARMA layer

In what follows, we repeatedly use Lemma 16 to decompose an ARMA layer into simpler submodules until the stability analysis for the submodule is tractable.

From ARMA model to AR model. In section 4, we show that an ARMA layer can be decomposed into a *cascade* of a *traditional convolutional layer* in Equation 7a and an *autoregressive layer* in Equation 7b. Since the traditional convolutional layer is always BIBO stable (which can be proved by triangular inequality), it is sufficient to guarantee the stability of the autoregressive layer

$$\mathcal{A}_{:,:,t} * \mathcal{Y}_{:,:,t} = \mathcal{T}_{:,:,t} \tag{C.7}$$

From multiple channels to a single channel. Note that the autoregressive layer in Equation C.7 is a *concatenation* of T channels of ARMA models, therefore it is sufficient to guarantee the stability of each channel individually. For simplicity, we drop the subscript t for channels, and our goal reduces to finding a necessary condition for the stability of

$$\sum_{q_1}\sum_{q_2} \mathcal{A}_{q_1,q_2}\mathcal{Y}_{i_1-q_1,i_2-q_2} = \mathcal{T}_{i_1,i_2} \tag{C.8}$$

where $\mathcal{A} \in \mathbb{R}^{K_a \times K_a}$ and $\mathcal{T}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2}$.

From separable 2D-filter to two 1D-filters. We consider the filter $\mathcal{A}$ in Equation C.8 to be separable, i.e.

$$\mathcal{A}_{q_1,q_2} = f_{q_1}\, g_{q_2} \tag{C.9}$$

where the 2D-filter $\mathcal{A}$ is decomposed as an outer product of two 1D-filters f an g. Given the factorization, the model in Equation C.8 can be written as a cascade of two submodules:

$$\sum_{q_1} f_{q_1}\, \mathcal{S}_{i_1-q_1,i_2} = \mathcal{T}_{i_1,i_2} \tag{C.10a}$$

$$\sum_{q_2} g_{q_2}\, \mathcal{Y}_{i_1,i_2-q_2} = \mathcal{S}_{i_1,i_2} \tag{C.10b}$$

where $\mathcal{S} \in \mathbb{R}^{I_1 \times I_2}$ is an intermediate result. Notice that Equation C.10a is a concatenation of I_2 submodules, each of which operates on a column of $\mathcal{T}$. Similarly, Equation C.10b can be decomposed into a concatenation of I_1 submodules, and each submodule operates on a row of $\mathcal{S}$. According to Lemma 16, it is sufficient to guarantee the stability of f and g individually. For simplicity, we denote both f and g as a, and rewrite each submodule in (C.10a) and (C.10b) as

$$a * y = x \quad \Longleftrightarrow \quad \sum_q a_q\, y_{i-q} = x_i \tag{C.11}$$

From general 1D-filter to composition of length-3 filters. By *the fundamental theorem of algebra*, a general filter a can always be decomposed as a composition of shorter filters (Oppenheim & Schafer, 2014). Specifically, suppose $a \in \mathbb{R}^K$ is a length-K filter, it can be factorized into a composition of $Q = (K-1)/2$ filters with length-3:

$$a = a^1 * a^2 \cdots * a^Q \tag{C.12}$$

where each filter $a^q \in \mathbb{R}^3$ has three coefficients. By the decomposition, the model in Equation C.11 is a cascade of Q submodules

$$a^1 * \left(a^2 * \cdots \left(a^Q * y\right)\right) = x \tag{C.13}$$

Therefore, we only need to guarantee the stability for each a^q individually. In the next subsection, we will further drop the superscript q and assume a itself is a length-3 filter.

C.3. Stability of a length-3 1D-filter

Without loss of generality, we assume the filter a is centered at 0 with $a_0 = 1$ (otherwise we can rescale the moving-average coefficients). The model at consideration can be written as

$$a_1 y_{i-1} + y_i + a_{-1} y_{i+1} = x_i \tag{C.14}$$

The analysis of this model follows the standard approach of Z-transform in analysis of BIBO stability (Oppenheim & Schafer, 2014). To begin with, we review the concepts of *Z-transform*, *region of convergence* (ROC) and their relation to the BIBO stability of a linear model.

Definition 17 (Z-transform and ROC). *Given a one-dimensional sequence h, the Z-transform maps the sequence to a complex function on the complex plain $\mathbb{C}$*

$$H(z) = \sum_{i=-\infty}^{+\infty} h_i z^{-i} \tag{C.15}$$

Notice that the infinite series does not necessarily converge for any $z \in \mathbb{C}$, and the transformation exists only if the summation is convergent. The region in the complex plane that the Z-transform exists is known as the region of convergence (ROC) for the sequence h.

Lemma 18 (ROC and BIBO stability). *Consider a linear model $y = h * x$, and let H denote the Z-transform of h, then a necessary and sufficient condition for the model being BIBO stable is that the unit circle belongs to the ROC, i.e. the infinite series*

$$H(e^{\mathrm{j}\omega}) = \sum_{i=-\infty}^{\infty} h_i e^{-\mathrm{j}\omega i} \tag{C.16}$$

converges for any frequency $\omega \in \mathbb{R}$. In other words, the discrete time Fourier transform (DTFT) exists for h.

Lemma 19 (ROC of length-3 AR model). *Consider a length-3 AR model $a * y = x$, i.e. $a_{-1} y_{i-1} + y_i + a_1 y_{i-1} = x_i$, the Z-transform of a is a length-3 complex polynomial $A(z) = a_{-1} z + 1 + a_1 z^{-1}$ with two zeros z_1 and z_2. Then the Z-transform of its inverse convolution $\bar{a}$ is*

$$\bar{A}(z) = \frac{1}{A(z)} = \frac{z}{a_{-1} z^2 + z + a_1} \tag{C.17}$$

with the corresponding ROC $|z_1| < z < |z_2|$ as a ring. Since the model can be written as $y = \bar{a} * x$, it is BIBO stable if $|z_1| < 1 < |z_2|$ according to Lemma 18.

With the lemmas above, we are ready to prove Theorem 9.

Proof. Since the coefficients in a are real numbers, the zeros of $F(z) = zA(z) = a^{-1} z^2 + z + a_1$ has two possible distributions: (1) Both zeros lie on the real axis, i.e. z_1 and z_2 are real numbers; and (2) z_1 and z_2 are complex conjugate to each other, i.e. $z_1^* = z_2$.

The second distribution also implies $|z_1| = |z_2|$. However, Lemma 19 shows that $|z_1| < 1 < |z_2|$ is required for BIBO stability, and therefore the second distribution is not feasible.

If both zeros lie on the real axis, the requirement of $|z_1| < 1 < |z_2|$ is equivalent to $F(1) \cdot F(-1) < 0$, which leads to

$$(a_{-1} + 1 + a_1)(a_{-1} - 1 + a_1) < 0 \tag{C.18}$$

$$(a_{-1} + a_1)^2 - 1 < 0 \implies |a_{-1} + a_1| < 1 \tag{C.19}$$

which completes the proof. □

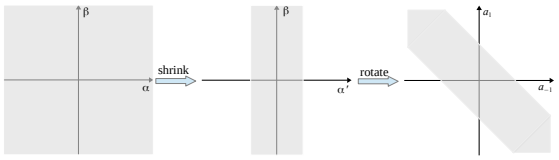


Figure 9. Visualization of the re-parameterization transformation in Equations (16a) and (16b). The ARMA model store the variables in α and β, and construct actual parameters a_{-1} and a_1 during the forward pass.

D. Supplementary Materials for Experiments

D.1. Multi-frame Video Prediction

Details of the Backbone Architecture. The backbone consists of a stack of 12-layers of Conv-LSTM modules, each of which has 32 units (channels). Following Byeon et al. (2018), two skip connections performing concatenation over channels are added between (3, 9) and (6, 12) layers, and a traditional convolutional layer is applied on top of all recurrent layers to compute the predicted frames. Wei Illustration of the backbone architecture in Figure 10.

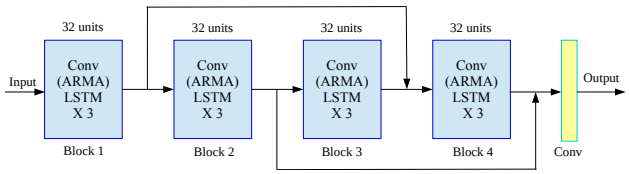


Figure 10. Illustration of the network architecture for the 12-layers model used in the experiments.

Training Strategy. All models are trained with ADAM optimizer (Kingma & Ba, 2014) with $\mathcal{L}_1 + \mathcal{L}_2$ loss for 400 epochs, and the model with lowest validation loss is selected. Gradient clipping with value 3 is used to stabilize the training. Learning rate decay and scheduled sampling (Bengio

et al., 2015) are used to ease training. Scheduled sampling is started once the model does not improve in 20 epochs (in term of validation loss), and the sampling ratio is decreased linearly by 4×10^{-4} each epoch (i.e. the scheduling sampling lasts for 250 epochs). Learning rate decay is further activated if the loss does not drop in 20 epochs, and the rate is decreased exponentially by 0.98 every 5 epochs. All parameters are initialized by Xavier's normalized initializer (Glorot & Bengio, 2010) and states in Conv-LSTMs or ARMA-LSTMs are initialized as zeros.

D.2. Image Classification

In this section, we apply our proposed ARMA networks on CIFAR10 and CIFAR100 image classification tasks. We replace the traditional convolutional layers by ARMA layers in three benchmarking architectures: AlexNet (Krizhevsky et al., 2012), VGG-11 (Simonyan & Zisserman, 2014), and ResNet-18 (He et al., 2016).

Training Strategy. All models are trained using cross-entropy loss and stochastic gradient descent (SGD) with batch size 128, learning rate 0.1, weight decay 0.0005 and momentum 0.9. For CIFAR10, the models are trained for 300 epochs and we divide the learning rate by 2 every 30 epochs. For CIFAR100, the models are trained for 200 epochs and we divide the learning rate by 5 at the 60^{th}, 120^{th}, 160^{th} epochs.

Results. The experimental results are summarized in Table 3. Our results show that ARMA models achieve comparable or slightly better results than the benchmarking architectures for image classification tasks. Replacing traditional convolutional layer with our proposed ARMA layer slightly boosts the performance of VGG-11 and ResNet-18 by 0.2%-0.8% in accuracy. Since image classifications tasks do not require global information as much as the video prediction tasks, the learned autoregressive coefficients are highly concentrated around 0 as shown in Figure 8. Consequently, ARMA networks almost reduce to traditional convolutional neural networks and therefore achieve comparable results.

	AlexNet		VGG-11		ResNet18	
	Conv	ARMA	Conv	ARMA	Conv	ARMA
CIFAR10	0.8677	0.8596	0.9243	0.9273	0.9520	0.9538
CIFAR100	0.5960	0.5780	0.6956	0.7034	0.7405	0.7428

Table 3. Image classification on CIFAR10 and CIFAR100.