3 Method and Mixed-scale Dense Network Overview

In this section, I will introduce the method, i.e. mixed-scale dense neural network that I applied to replace the traditional contrast transfer function method in solving the phase retrieval.

3.1 Problem Definition and Method

When solving inverse problem, contrary to contrast transfer function that solves the mathematical inverse problem based on the original physical model, when applying the neural network, we convert it into a pure image processing problem:

$$F(\mathbf{x}) = f_n(\dots \quad f_3(\quad f_2(\quad f_1(\quad f_0(\mathbf{x}) * \mathbf{w}_1 + \mathbf{b}_1) * \mathbf{w}_2 + \mathbf{b}_2)) * \mathbf{w}_3 + \mathbf{b}_3 \quad \dots) \tag{3.1}$$

where $\mathbf{x}$ corresponds to $I_D(\mathbf{x})$ in section 1. f_0 is normalization of input $\mathbf{x}$, $f_1 \dots f_n$ are non-linear activation function for each layer

We will solve this image processing problem by finding an optimal regression result for minimization of difference between the high-dimensional non-linear composite function $F(\mathbf{x})$ described by deep neural network and expected output vector $\mathbf{T}$:

$$\mathbf{w}, \mathbf{b} = \underset{\mathbf{w},\mathbf{b}}{\operatorname{argmin}} \left[\mathbf{T} - F(\mathbf{x})\right]^2 \tag{3.2}$$

where $\mathbf{T}$ corresponds to $T(\mathbf{x})$ in section 1.

Normally the distance applied to measure such difference in regression problem is mean square error. To solve this regression problem, we use iterative optimizer based on backpropagation of neural network as introduced in last section.

Traditional contrast transfer function and deep neural network solve the inverse problem in the least square sense. However, the contrast transfer function is based on linearization of transmittance function $T(\mathbf{x})$ that is desired output for inverse problem, and deal such least square regression problem in Fourier domain. Since mathematical expression of propagator and transmittance function are given, it is solved using Tikhonov regularization based regression to approximate an optimal result.

While for deep neural network based method, we have no knowledge about direct problem,i.e. neither transmittance function nor Fresnel transform, but the model is trying to learn an optimal mapping rule that essentially transforms high-dimensional vector $\mathbf{x}$ to another high-dimensional vector $F(\mathbf{x})$.

3.2 Dilated Convolution Filter

3.2.1 Receptive Field

The receptive field is defined as the region in the input space that a particular CNN's feature is looking at (i.e. be affected by) and neurons in deeper layer have a greater receptive field. For deep convolutional neural network(e.g.AlexNet[18] or VGG[19]) and fully-convolutional networks(e.g. FCN[20]) with a single path from input to output, it has receptive field:

$$r_n = r_{n-1} + (k-1) \times \prod_{l=1}^{n-1} s_i \tag{3.3}$$

where r_n is receptive field of current layer, r_{n-1} is receptive field of previous layer, s_i is convolution or pooling stride of ith layer, k is convolution kernel size. Here we have identical convolution kernel size throughout all the layers.

While for modern convolutional networks where there may be multiple paths from the input to the output (e.g. ResNet[21] or Inception[22]), it's much more complicated to derive the expression of receptive field which depends on network architecture.

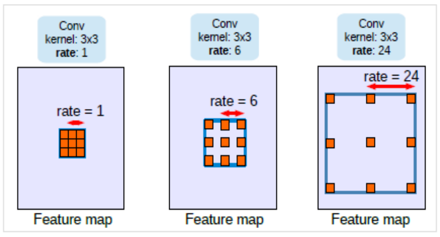
The larger receptive field means feature maps contain more long range information, it is crucial for all classification task, dense prediction and regression. Normally in DCNN(e.g.AlexNet[18] or VGG[19]) for classification task, the receptive field grows as layer becomes deeper and resolution also drops gradually. The fully-convolutional network [20] follows the similar pattern, but replaces last few fully connected layers by convolutional layers to make efficient end-to-end learning and employ deconvolution [23] to learn the upsampling of low resolution feature responses, but deconvolution has upper limitation in resolution recovery. Thus the resolution is sacrificed for growing receptive field. However, dense prediction and regression requires multi scale contextual reasoning in combination with full-resolution output.

3.2.2 Dilated Convolution Filter

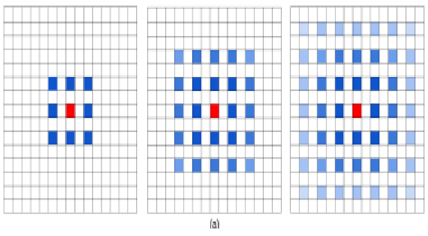
The proposed dilated convolutional filter allows us to explicitly control how densely to compute feature responses and extract them in fully convolutional networks without requiring learning any extra parameters, thus it enlarges receptive field while keeping resolution, eliminating down-sampling. It is generalized version of conventional convolutional filter. Consider two-dimensional signals, for each location i on output $\mathbf{y}$ and a filter w, dilated convolution is applied over input feature map $\mathbf{x}$ [24]:

$$\mathbf{y}[i] = \sum_{k=1}^{K} \mathbf{x}[i + r\dot{k}]w[k] \qquad k_d = k + (k-1)\cdot(r-1) \tag{3.4}$$

where the dilation rate r corresponds to the stride with which we sample the input signal, which is equivalent to convolving the input $\mathbf{x}$ with upsampled filters produced by inserting $r-1$ zeros between two consecutive filter values along each spatial dimension in order to increase image resolution. The standard convolution is a special case for rate $r = 1$, and dilated convolution allows us to adaptively modify filter's receptive field by varying r. $\mathbf{y}$ is the output signal. $w[k]$ denotes the filter of length K. k_d is the size of resulting dilated convolutional filter



(a) 3×3 Sized Dilated convolution kernels with different rates

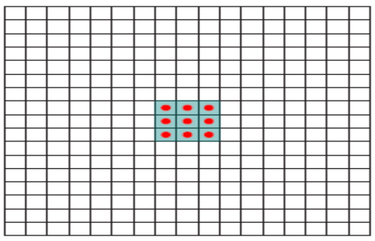


(b) Illustration of gridding problem

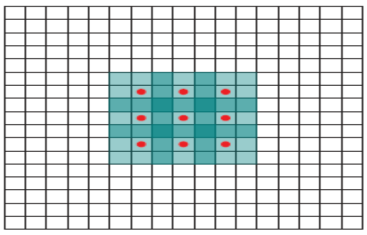
Figure 3.1: Dilated Convolution Kernel and Gridding Effect

Normally we successively apply dilated convolution layer by layer. The gridding artifacts is an inherent problem that exists in the current dilated convolution framework. Adjacent units in the output are computed from completely separate sets of units in the input and thus have totally different actual receptive fields. Since zeros are padded between two pixels in a convolutional kernel, the receptive field of this kernel only covers an area with checkerboard patterns, so only locations with non-zero values are sampled, losing some neighboring information [25]. Because of that, the actual pixels that participate in the computation from the $k_d \times k_d$ region are just $k \times k$ with a gap of $r - 1$ between them as figure (b) above. It mainly happens in the situation where dilated convolution kernels of identical rate are successively applied to layers.

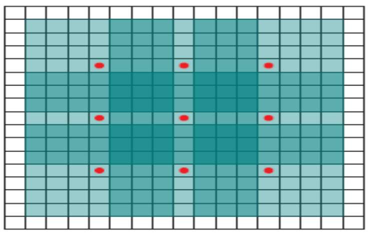
A first specific dilated convolutional network architecture was proposed that aggregates multi-scale contextual information and supports exponential expansion of the receptive field without loss of resolution or coverage[26].



(a) Dilation Rate = 1



(b) Dilation Rate = 2



(c) Dilation Rate = 4

Figure 3.2: Systematic dilation supports exponential expansion of the receptive field without loss of resolution or coverage

The 9 red points in each figure compose the 3×3 sized dilated convolutional kernel. The green region is the receptive field.

The figure (a) corresponds to the dilation rate$= 1$, which is an ordinary convolution operation. Assume it is the first layer, the receptive field size is $r_1 = 1 + (3-1) = 3$, i.e. 3×3. The figure(b) corresponds to the dilation rate$= 2$. Each time it convolves with any 5×5 image patch, only these 9 red points in image patch are taken into account for each single convolution output value. So we can treat its kernel size as 5×5 at this moment. Since it is connected to last layer, its receptive field size is $r_2 = 3 + (5-1) \times 1 = 7$,i.e. 7×7. While for conventional convolution, receptive field is 5×5 at this point. The figure(c) corresponds to the dilation rate$= 4$, similar to (b), we can treat its kernel size as 9×9 at this moment, and its receptive field size is $r_3 = 7 + (9-1) \times 1 \times 1 = 15$, i.e. 15×15. While for conventional convolution, receptive field is 7×7 at this point. The size of receptive field for this architecture follows exponentially increasing trend.

These 3 receptive fields contain all points within a rectangle, this architecture avoids gridding and catch fine-detailed features.

Another specific dilated convolutional network architecture was proposed by [25], namely hybrid dilated convolution. It proposed the architecture to avoid gridding and let the final size of receptive field fully cover the input without any holes or missing edges. Suppose N convolutional layers with kernel size $K \times K$ that have dilation rates of $[r_1, ..., r_i..., r_n]$, but the assignment of dilation rate follows a sawtooth wave-like heuristica: number of layers are grouped together to form the rising edge of the wave that has an increasing dilation rate, and the next group repeats the same pattern. For example, suppose we have 9 layers, the dilation rate for each layer can be sequentially designed as: $[1, 2, 3, 1, 2, 3, 1, 2, 3]$.

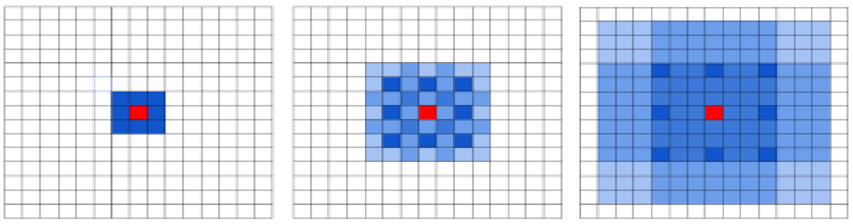


Figure 3.3: Sequential Convolutional layers have dilation rates of $r = 1, 2, 3$ respectively

From the assignment of sequential dilation rates, it follows that the maximum distance between two nonzero values in dilated convolution kernel is:

$$M_i = \max[M_{i+1} - 2r_i, \quad M_{i+1} - 2(M_{i+1} - r_i), \quad r_i] \qquad\qquad M_n = r_n. \tag{3.5}$$

The design goal is to let $M_2 \leq K$. For example, for kernel size $K = 3$, an $r = [1, 2, 5]$ pattern works as $M_2 = 2$; however, an $r = [1, 2, 9]$ pattern does not work as $M_2 = 5$. Therefore, the sequential dilation rates for mixed-scale dense neural network follow this rule and it is $[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, ..., 1, 2, 3, 4, 5...12, ...]$. It means the feature maps in later layers have greater size of receptive field and even cover the entire input image.

3.3 Dense Network and Concatenation Connection

The optimal solution is obtained by iterative minimization of the cost function for the neural network. And the update of neural network parameters is obtained according to the gradient that depends on the expression of the cost function. So gradient carries most important information for training of the neural network, the loss of gradient information will cause irreversible worse effect for model prediction ability.

The works [22, 19] reveal that the network depth is crucial. The better results are obtained with a deeper model. The network model would acquire more generality and better performance in real testing.

As mentioned in last section, gradient vanishing or exploding may happen with deeper layers if sigmoid activation is used. The mixed-scale dense neural network is not designed with intermediate normalization layer, i.e. batch/layer/instance normalization layers to overcome vanishing gradient or exploding. The ReLU activation applied in MSDNet can eliminate it.

.

3.3.1 Highway Network

To overcome this kind of issue, it is suggested to make some improvement on network layer connection architecture by [27]. Suppose a plain feedforward neural network typically consists of L layers where the lth layer is applied with a non-linear affine transformation F on its input x_{l-1} to produce output x_l. A highway network is established by additionally define two non-linear transformations $T_l(\mathbf{x}_{l-1})$ and $C_l(\mathbf{x}_{l-1})$ such that:

$$\mathbf{x}_l = F_l(\mathbf{x}_{l-1}) \cdot T_l(\mathbf{x}_{l-1}) + \mathbf{x}_{l-1} \cdot C_l(\mathbf{x}_{l-1}) \tag{3.6}$$

where T_l refer to the transform gate for lth layer and C_l refer to the carry gate for lth layer, since they express how much of the output is produced by transforming the input and carrying it, respectively [27]. For simplicity, they set $C = 1 - T$[27],giving:

$$\mathbf{x}_l = F_l(\mathbf{x}_{l-1}) \cdot T_l(\mathbf{x}_{l-1}) + \mathbf{x}_{l-1} \cdot (1 - T_l(\mathbf{x}_{l-1})) \tag{3.7}$$

The short connections are established with gating functions. However, these gates are data-dependent and have parameters, When a gated shortcut is closed (approaching zero), the layers in highway networks represent non-residual functions, information may not pass through. Further, highway networks have not demonstrated accuracy gains with extremely increased depth.

3.3.2 Deep Residual Block

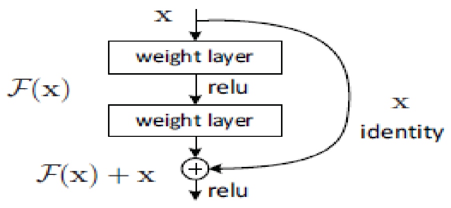
Inspired by the highway network, another building block namely residual learning was proposed. Instead of hoping each few stacked layers directly fit a desired underlying mapping, they explicitly let these layers fit a residual mapping [21]:

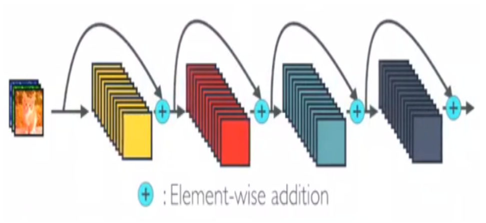
$$\mathbf{x}_l = F_l(\mathbf{x}_{l-1}) + \mathbf{x}_{l-1} \qquad\qquad F_l(\mathbf{x}_{l-1}) := H_l(\mathbf{x}_{l-1}) - \mathbf{x}_{l-1} \tag{3.8}$$

The function $F_l(\mathbf{x}_{l-1})$ represents the residual mapping to be learned. $H_l(\mathbf{x}_{\mathbf{l-1}})$ represents desired underlying mapping.

In this framework, assume that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping. If one assumess that multiple nonlinear layers can asymptotically approximate complicated functions, then it is equivalent to assume that they can asymptotically approximate the residual functions, i.e., $H_l(\mathbf{x}_{\mathbf{l-1}}) - \mathbf{x}_{\mathbf{l-1}}$. So rather than expect stacked layers to approximate $H_l(\mathbf{x}_{\mathbf{l-1}})$, we explicitly let these layers approximate a residual function $F_l(\mathbf{x}_{\mathbf{l-1}}) := H_l(\mathbf{x}_{\mathbf{l-1}}) - \mathbf{x}_{\mathbf{l-1}}$. The original function thus becomes $F_l(\mathbf{x}_{\mathbf{l-1}}) + \mathbf{x}_{\mathbf{l-1}}$. If an identity mapping were optimal, it would be easier to push the residual to zero than to fit an identity mapping by a stack of nonlinear layers. The operation $F_l(\mathbf{x}_{\mathbf{l-1}}) + \mathbf{x}_{\mathbf{l-1}}$ is performed by a shortcut connection and element-wise addition, so the dimensions of $\mathbf{x}_{l-1}$ and $F_l(\mathbf{x}_{l-1})$ must be equal in Eq.3.8. If the added layers can be constructed as identity mappings, a deeper model should have training error no greater than its shallower counter part.

In real cases, the identity mappings seem to be unlikely optimal, but this reformulation may help to precondition the problem. If the optimal function is closer to an identity mapping than to a zero mapping, it should be easier for the solver to find the perturbations with reference to an identity mapping, than to learn the function as a new one.


x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu
x
identity
(a) Residual Learning Block



(b) Element-wise Addition

Figure 3.4: Deep Residual Block

However, the main design purpose of deep residual block is to address the degradation problem for the situation where intermediate normalization [3, 17, 14, 15] and good normalized initialization of input [16] have been performed. As deeper networks is able to start converging, a degradation problem has been exposed: with the network depth increasing, accuracy saturates and then degrades rapidly. Unexpectedly, such degradation is not caused by over-fitting, and adding more layers to a suitably deep model leads to higher training error. An error evolution graph to illustrate the degradation problem is attached in Appendix. But it doesn't mean the residual learning block are unable to overcome the gradient vanishing or exploding problem. As implied by Eqs.2.8 - 2.11, with the help of shortcuts connection of residual learning block, gradient $\frac{\partial C}{\partial w^l_{j,k}}$ or $\frac{\partial C}{\partial b^l_j}$ can be established more directly with less layers involved, so that gradient vanishing or exploding could be overcome as well.

3.3.3 Dense Block and Dense Connectivity

An advantage of ResNets is that the gradient can flow directly through the identity function from later layers to the earlier layers. However, the identity function and the output of F_l are combined by summation, which may impede the gradient information flow in the network. To further improve the gradient information flow between layers and as inspired by both highway network and residual learning block, [28] proposes a different connectivity pattern: an architecture of direct connections from any layer to all subsequent layers is introduced. Consequently, the lth layer receives the feature maps of all preceding layers, $\mathbf{x}_0, ..., \mathbf{x}_{l-1}$, as input:

$$\mathbf{x}_l = F_l([\mathbf{x}_0, ..., \mathbf{x}_{l-1}]) \tag{3.9}$$

where $[\mathbf{x}_0, ..., \mathbf{x}_{l-1}]$ refers to the concatenation of the feature maps produced in layers $0, ..., l-1$. Because of its dense connectivity we refer to this network architecture as densely connected convolutional network. For ease of implementation, we concatenate the multiple inputs of $F_l(\bullet)$ into a single tensor. In dense block, the dimension of input channels and all the feature maps generated should be identical for concatenation reason, this property is in line with residual learning block. If the network is developed for classification purpose, the pooling layer could be inserted into the transition layer between each dense block of densely connected neural network.

The traditional plain convolutional networks with L layers have L connections-one between each layer and its subsequent layer. While the dense connectivity, the network has $\frac{L(L+1)}{2}$ direct connections in each dense block. For each layer, the feature maps of all preceding layers are used as inputs and then its own feature maps will be used as inputs into all subsequent layers in dense block. It means the output of dense block has direct access to all the input channels and feature maps generated, the gradient information flow ($\frac{\partial C}{\partial w^l_{j,k}}$ and $\frac{\partial C}{\partial b^l_j}$) could be much less perturbed by other hidden layers ($w^0_{j,k}, w^1_{j,k}, ..., w^n_{j,k}, b^0_j, b^1_j, ..., b^n_j$) in back-propagation.

If non-linear transformation function F_l for lth layer produces v feature maps, so the lth layer has input $v_0 + v \times (l-1)$ feature maps, where v_0 is the number of input channels. So hyper parameter v refers to the growth rate of the dense block. The very narrow layers of densely connected network in contrast to other existing network architecture benefits from the low growth rate v.

Instead of exploring the representational power of very deep or wide architectures, the dense connectivity structure exploit the potential power of network through feature reuse, creating a condensed model that is easier to train. It alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse and substantially reduce the number of parameters.

However, the limitation of Dense Net is computation cost even though it has narrow size for each layer if we set too many layers for network. Because the gradient information flow is based on multi-path between hidden layers and output.

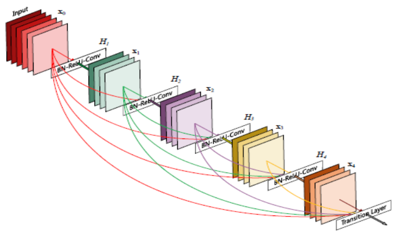


Figure 3.5: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature maps as input.

3.4 Mixed-scale Dense Network

The mixed-scale dense neural network [29] is combining the advantages of dilated convolutional filters and dense connection in optimizing the function $F(\mathbf{x})$ in Eq.3.1.

Since the aim is to solve a regression problem, all the layers are contained in a single dense block. The default growth rate v is set to be 1, each hidden layer generates only 1 feature map. Each feature map is the result of applying a set of operations: perform the channel-specific dilated convolution on each of previously generated feature maps and input channels using 3×3 dilated convolutional kernel, then sum all resulting images pixel by pixel. And then add constant bias to each pixel, finally apply ReLU activation. And each time the dilation rate is selected from the list $[1, 2, 3, ..., 11, 12, 1, 2, 3, ..., 12, 1, 2...]$ in order. As mentioned above, since MSD network has multiple paths from input to output, the expression of size of receptive field computation is hard to derive. However, since of the the way each feature map was generated, the receptive field formed is quite complex, but will not be smaller than before.

With dilated convolutional filters, the long range information in images becomes quickly available in early layers of network, greater receptive field size can be obtained earlier, making it possible to use this information to improve the results of deeper layers. As mentioned in the last part, the consecutively increasing dilation rates applied for MSD network layers avoid the gridding problem.

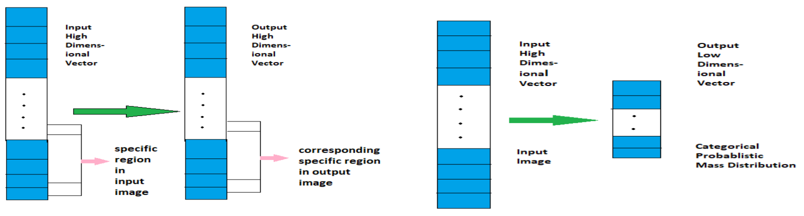
The final output of network is the result of performing point wise convolution to all feature maps, i.e. the linear combination of all feature maps generated and input channels, and then apply activation function. Suppose there are n layers in developed mixed-scale dense network:

$$y = \sigma([x_0, x_1, ..., x_n], W_{final}) \tag{3.10}$$

where y is final output channels of mixed-scale dense neural network, x_0 is input channels, $x_1, ..., x_n$ are feature maps generated by each layer respectively. W_{final} stands for point wise convolution, σ stands for activation function. The weight of each feature map (including input channels) could be learnt according to the receptive field in the generated images that is in line with the desired output. It means feature map or input channels whose receptive field is more in line with desired output would be given more weight in final output formation in processing by point wise convolution.

3.4.1 Analysis on Mixed-scale Dense Network

As mentioned in last section, instead of generally applied min-max normalization that reformulate data distribution to be in range $(0, 1)$, the initialization of mixed-scale dense neural network input is applying the z-score standardization based on channel and batch data information. The mean value of standardized input channels would be around 0, it seems to cause the dead ReLU problem as mentioned in the last section. However, since we use the densely connected structure, the final output of network is very directly accessing the targeted hidden layers, it provides more effective gradient information flow in back-propagation. Thus, some dead neurons could revive for some extent by supervised training. Further, the ReLU can be treated as a kind of information filter for valuable information. For the high dimensional vector processed by ReLU, less valuable information will be stuck in dead region of ReLU, and more valuable information will fall in linear activation region on ReLU, even if it is temporarily dead, and it still has more possibility to revive due to the advantage of dense block.


Input High Dimens-ional Vector
Output High Dimens-ional Vector
specific region in input image
corresponding specific region in output image
Input High Dimes-ional Vector
Output Low Dimens-ional Vector
Input Image
Categorical Probablistic Mass Distribution

(a) (b)

Figure 3.6: Vector Transformation for Regression and Classification respectively