1 Introduction

Convolutional Neural Networks (CNN) are a class of function approximators that have proved to be very successful in computer vision with state-of-the-art applications in image recognition [14, 9], object localization [10, 19], image generation [7], super-resolution [13], etc. Despite their superior performance, CNNs are much less understood by the research community in theory. These models are created using compositions of a large number of convolutional layers, where every layer is composed of a convolution operation with parameterized filters followed by an elementwise nonlinear function (such as $h(x) = \max(0, x)$) followed by optional spatial downsampling, and in general have a huge number of (tens of millions of) parameters. Learning such a model is achieved through minimizing a nonconvex loss function over its huge parameter space, and not a lot of theories exist on what mathematical properties these network functions have. Therefore, it is of interest to design CNN models with more structure in its parameter space, fewer number of parameters, and faster computation time while preserving the predictive performance. In addition, more insights could be gained if provable properties of the designed model could be found under reasonable assumptions.

1.1 Related Works

Broadly speaking, two lines of work are directly relevant and are the inspirations to our research direction. One is in the works of sparse coding and dictionary learning [6], and the other in designing structured CNN models.

There has been work in the field of sparse coding that proposes to view CNN models through the more theoretically developed multi-layer convolutional sparse coding model (ML-CSC)[17]. It was suggested that the forward computation conducted in the convolutional neural network could be viewed as solving an ML-CSC problem to find the sequence of latent representations which generate the observed image data. In this formulation, every convolutional filter is viewed as a dictionary of bases over which the next layer’s representation is a linear combination of. However, the work by Papan et al. does not concern the learning of these convolutional filters; it assumes that they are fixed. On the other hand, work in dictionary learning [3] tries to learn the dictionary as well as the basis coefficients to construct the given image data on a simple one-layer linear model. Rubinstein et al. [20] improves upon [3] to constrain that the dictionary bases to be sparse linear combinations of a set of fixed bases. Having the dictionary basis vectors as sparse combinations over fixed well-known bases ensures more efficient computation than the completely unconstrained version [3] while at the same time provides the adaptability and flexible that is lacking in the sparse coding problem where the dictionary basis is completely fixed.

Additionally, there has already been attempts to bring more structure to the network architecture and parameter space of CNN to allow for faster computation time, smaller number of parameters, and better empirical and theoretical properties. Works like Squeezenet [12], Mobilenet [11] have been suggested to simplify and reduce and the network connectivity patterns while preserving the predictive performance. Deep Neural Network with Decomposed Convolutional Filters (DCFNet) [18], a work most relevant to our research direction, proposes to add more structure in the model’s parameter space by constraining every two-dimensional convolutional filter of the DCFNet to be chosen from a subspace spanned by a fixed set
of bases such as the leading Fourier-Bessel bases [1]. This idea gets inspirations from ideas of both the multilayer convolutional sparse coding (ML-SCS) [17] and double sparsity dictionary learning [20]. Instead of learning a parameter for every variable dimension of each filter, the DCFNet model only needs to learn a smaller set of coefficients for each filter with respect to the fixed basis filters. This approach imposes strong regularity conditions on the smoothness of models’ learned filters, saves the number of parameters, and makes the computation faster, while preserving the classification accuracy for multiple image recognition tasks. In addition, Qiu et al. prove that DCFNet has stability in every layer’s output representations over certain types of variations in the input domain (which are images) when using the Fourier-Bessel bases as the filters.

In this research independent study, we have applied dictionary learning techniques to learning of adaptive bases in structured CNN and developed both a new CNN model and analysis of its data representation property. We propose to allow the convolutional filter basis filters in DCFNet to also be learned instead of being fixed. Learning the basis filters together with its coefficients like in sparse coding [6] allows our model to have better expressiveness than the fixed-bases DCFNet. This makes the new model better at capturing the data distribution and thus achieve better performance. Factoring each convolutional filter as a linear combination of the learnable basis vectors ensures the model will still have a smaller number of parameters and faster computation time like DCFNet. In addition, we don’t allow the basis filters to completely free adapt; instead, the basis filters are constrained to be chosen from a low-rank subspace spanned by a set of fixed pre-basis filters. In this way, we can enforce regularity conditions of the basis filters through a good choice of the pre-bases. This approach allows us to extend the theoretical representation stability results in [18] to our model, which we term as AdDCFNet.

2 Adaptive Decomposed Convolutional Filters

In this section we describe the formulation of our model, its inference procedure, and the parameter and computation advantage it has over standard CNNs. We will follow the notation used by Qiu et al. [18].

2.1 Adaptive bases

In [18], the convolutional filter $W_{\lambda',\lambda}^{(l)}$ between channel $\lambda'$ in layer $l-1$ and channel $\lambda$ in layer $l$ is factorized over a set of basis filters $\{\psi_k\}_{k=1}^{K}$, as

$$W_{\lambda',\lambda}^{(l)}(u) = \sum_{k=1}^{K} (a_{\lambda',\lambda}^{(l)})_k \psi_k(u)$$  \hfill (1)

, where $a_{\lambda',\lambda}^{(l)} \in \mathbb{R}^K$ and $(a_{\lambda',\lambda}^{(l)})_k$ is its $k$-th component. During training, the basis filters $\{\psi_k\}_{k=1}^{K}$ are not adapted and only the coefficients $a_{\lambda',\lambda}^{(l)}_k$ receive gradient updates.

Instead of fixing the basis filters of layer $l$, we propose to learn layer-specific basis filters $\{\phi_k^{(l)}\}_{k=1}^{K_0}$ from the span of a set of fixed pre-basis filters $\{\psi_t^{(l)}\}_{t=1}^{K_0}$ ($K_0 > K$).

Under this formulation, we write each basis filter as a linear combination of the pre-basis filters:

$$\phi_k^{(l)}(u) = \sum_{t=1}^{K_0} C_{t,k}^{(l)} \psi_t^{(l)}(u)$$  \hfill (2)

, where $C^{(l)} \in \mathbb{R}^{K_0 \times K}$ is a matrix whose row-$t$, column-$k$ entry we denote as $C_{t,k}^{(l)}$ and whose $k$-th column we denote as $C_k^{(l)}$.

We then express our convolutional filter as a linear combination of the newly introduced basis filters:

$$W_{\lambda',\lambda}^{(l)}(u) = \sum_{k=1}^{K} (b_{\lambda',\lambda}^{(l)})_k \phi_k^{(l)}(u)$$  \hfill (3)

, where $b_{\lambda',\lambda}^{(l)} \in \mathbb{R}^K$. 


Combining Equation (2) and Equation (3), we have

\[ W_{\lambda', \lambda}^{(l)} = \sum_{t=1}^{K_0} \left( \sum_{k=1}^{K} C_{t,k}^{(l)}(b_{\lambda', \lambda}^{(l)})_k \right) \psi_t' = \sum_{t=1}^{K_0} \left( C_{t}^{(l)}(b_{\lambda', \lambda}^{(l)})_t \psi_t' \right) \]  

(4)

Here \((C_{t}^{(l)}(b_{\lambda', \lambda}^{(l)})_t)\) is the \(t\)-th entry of \(K_0\)-dimensional vector \(C_{t}^{(l)}(b_{\lambda', \lambda}^{(l)}) \). Comparing to the factorization used in DCFNet in Equation (1), we have changed the convolutional filter’s coefficients \(a_{\lambda', \lambda}^{(l)}\) with respect to \(\{\psi_k\}_{k=1}^{K_0}\) to \(C_{t}^{(l)}(b_{\lambda', \lambda}^{(l)})_t\) with respect to \(\{\psi_t'\}_{t=1}^{K_0}\). We notice here that we require \(K < K_0\) since otherwise the matrix \(C_{t}^{(l)}\) can be made full rank and linear combinations of its columns using \(b_{\lambda', \lambda}^{(l)}\) can create any vector in \(\mathbb{R}^{K_0}\), which would makes the AdDCF layer have the same expressiveness as a DCF layer with \(K_0\) bases but with more parameters.

The true advantage of AdDCF of factorizing the bases comes when we use the same number of bases \(K\) as used in a DCF model but allow for more pre-basis filters \((K_0 > K)\). We show later in Section 2.3 that AdDCF retains the same level of parameter and computation reduction as DCF so the comparison is fair in this case. However, AdDCF with \(K\) bases and \(K_0\) pre-bases has more expressiveness than DCF with \(K\) bases.

To see this, for a given \(l\)-th layer of filters \(\{W_{\lambda', \lambda}^{(l)}\}_{\lambda' \in [M_{l-1}], \lambda \in [M_l]}\), with \(W_{\lambda', \lambda}^{(l)}(u) = \sum_{k=1}^{K_0}(a_{\lambda', \lambda}^{(l)})_k \psi_k(u)\) in DCF, we let the first \(K\) pre-basis filters of AdDCF to match those of DCF, i.e. \(\psi_i' = \psi_i\) for \(i \in \{1, ..., K\}\). (The choice for the rest of the pre-bases can be any filters.) Then we take the shared coefficient matrix \(C^{(l)}\) as

\[ C^{(l)} = \begin{bmatrix} I_{K} \\ O \end{bmatrix} \]

, where the first \(K\) rows of \(C^{(l)}\) is the identity matrix \(I_{K}\) while the rest of the rows are all zeroes. By letting \(b_{\lambda', \lambda}^{(l)} = a_{\lambda', \lambda}^{(l)}\) for every \((\lambda', \lambda)\) pair, we have for the AdDCF filter

\[ W_{\lambda', \lambda}^{(l)} = \sum_{t=1}^{K_0} (C_{t}^{(l)}(b_{\lambda', \lambda}^{(l)})_t) \psi_t' = \sum_{t=1}^{K_0} (a_{\lambda', \lambda}^{(l)})_t \psi_t' + \sum_{k=K+1}^{K_0} 0 \cdot \psi_t' = \sum_{k=1}^{K} (a_{\lambda', \lambda}^{(l)})_k \psi_k \]

, which is the same as the corresponding filter in DCF. Therefore, the possible set of representable filters of AdDCF contains the set of representable filters of DCF when the number of bases \(K\) used is the same.

From this we see that AdDCF has more expressiveness than DCF while keeping the same level of parameter and computation reduction. We will see in Section 3 that AdDCF also has provable deformation stability properties as DCF.

2.2 Forward computation

The forward computation of AdDCF takes 3 steps:

1. (Ψ step) The bases \(\phi_{k}^{(l)}\) are constructed from pre-bases \(\{\psi_t'\}\) by convolving the stacked \(\psi_t'\) of shape \(K_0 \times L \times L\) with the coefficients \(C_{t,k}^{(l)}\) organized as a \(K \times K_0 \times 1 \times 1\) kernel.

2. (Φ step) Every channel of the input tensor is convolved with each of the basis \(\phi_{k}^{(l)}, k = 1, ..., K\).

3. (b-step) The resulted tensor output is convolved with the basis coefficients \(\{b_{\lambda', \lambda}^{(l)}\}_{\lambda' = 1}^{M_{l-1}}\) organized as a \(K_{M_{l-1}} \times 1 \times 1\) kernel for every \(\lambda \in 1, ..., M_l\).

2.3 Parameter and computation reduction

Suppose the original \(l\)-th convolutional layer uses filter of size \(L \times L\) and \(M_{l-1} = M'\) and \(M_l = M\). Then in total it uses \(L^2M'M\) number of parameters. In contrast, for AdDCF, we need \(K_0\) coefficients for each of the \(K\) bases, and \(K\) coefficients for every \(b_{\lambda', \lambda}^{(l)}\). Therefore, in total we need \(KM'M + KK_0\) number of parameters for AdDCF.
Since \(K < K_0 < L^2\) and \(K < M'\), we see that \(KK_0/(L^2M'M) < 1/M\), which is much smaller than 1. So using pre-bases in AdDCF does not much change its parameter saving ratio from DCF, which is approximately still \(\frac{K}{M}\).

In terms of number of add/multiply operations, we suppose that the input and output tensor both has spatial dimensions \(W \times W\). Then the original convolutional layer takes approximately \(2M'MW^2L^2\) flops. For AdDCF, the bases construction takes \(2K_0KL^2\) flops since it is run on the prebasis filters which is of size \(L \times L\). Once the bases are computed, the rest of the computation is the same as that of DCF, which takes \(2M'W^2K(L^2 + M)\) flops. So the total number of flops for AdDCF is \(2M'W^2K(L^2 + M) + 2K_0KL^2\). Under the assumption that \(M >> L^2\) used in the analysis of DCF, the added computation \(2K_0KL^2\) is very small compared to \(2M'W^2L^2\), and therefore does not contribute much to the computation ratio. Consequently, the computation cost of AdDCF is of order \(\frac{K}{L^2}\) of the traditional convolutional layers. This is the same order of saving as DCF.

From this section we see that AdDCF has more expressiveness than DCF while keeping the same level of parameter and computation reduction. We will see in the next section that AdDCF also has provable deformation stability properties as DCF.

### 3 Representation Stability

It is shown in [18] that DCFNet with Fourier Bessel bases has stability against input deformation when assumptions (A0), (A1), (A2') are satisfied. To develop the stability theory for AdDCFNet, we will consider the case where the pre-basis filters are chosen from the leading Fourier Bessel Bases. In this case, only assumption (A2') needs to be modified as it is the only assumption that concerns with the property of the model.

In assumption (A2'), \(A_l\) is defined as

\[
A_l := \pi \max_{\lambda} \sum_{\lambda=1}^{M_l-1} ||a_{\lambda}^{(l)}||_{FB}, \sup_{\lambda'} \sum_{\lambda=1}^{M_l-1} ||a_{\lambda}^{(l)}||_{FB}
\]

where

\[
||a_{\lambda}^{(l)}||_{FB} = \sqrt{\sum_{k=1}^{K} \mu_k \left( \left(a_{\lambda}^{(l)}\right)_k \right)^2}
\]

and \(\mu_k\) is the eigenvalue of eigenfunction \(\psi_k\) with respect to the Dirichlet Laplacian on unit Disk.

Assumption (A2') requires that

For all \(l, A_l \leq 1\).

For AdDCF, we have established that \(C^{(l)}b_{\lambda}^{(l)}\) replaces the role of \(a_{\lambda}^{(l)}\) with respect to the pre-basis \({\psi_i}^{K_0}_{i=1}\). Thus we can replace each of the \(a_{\lambda}^{(l)}\) in assumption (A2') with \(C^{(l)}b_{\lambda}^{(l)}\) and obtain the stability result.

However, this formulation does not provide us with enough intuition about the requirement on the shared matrix \(C^{(l)}\) and each of the \(b_{\lambda}^{(l)}\). In addition, there is a scaling ambiguity in the assumption as the assumption only concerns with the matrix product \(C^{(l)}b_{\lambda}^{(l)}\). (Scaling the matrix \(C^{(l)}\) by \(\gamma \in \mathbb{R}\) and simultaneously scaling each \(b_{\lambda}^{(l)}\) by \(\gamma\) would give the same product.)

From this observation, we state two new assumptions to be made that will guarantee the same stability result in Theorem 3.8 in [18].

(A3) For all \(k \in 1, \ldots, K\), \(C^{(l)}b_{\lambda}^{(l)} ||_{FB} \leq 1\)

\[
(A4) \pi \max_{\lambda} \sup_{\lambda'} \sum_{\lambda=1}^{M_l-1} ||b_{\lambda}^{(l)}||_{1}, \sup_{\lambda'} \sum_{\lambda=1}^{M_l-1} ||b_{\lambda}^{(l)}||_{1} \leq 1
\]
Assumption (A3) can be seen as a statement about the basis filters’ $\phi_k^{(l)}$’s 1-norm as was proved in Proposition 3.6 of [18]. Assumption (A4) can be seen as a generalized sparsity constraint of the expansion coefficients of the filters.

To prove (A3) (A4) together can satisfy (A2’), we notice that since $||\cdot||_{FB}$ is a weighted 2-norm, it satisfies triangle inequality:

$$||C^{(l)}b_{\lambda'}^{(l)}||_{FB} = \sum_{k=1}^{K} |(b_{\lambda'}^{(l)})_k| \cdot ||C_k^{(l)}||_{FB} \leq \sum_{k=1}^{K} |(b_{\lambda'}^{(l)})_k| \cdot ||C_{k}^{(l)}||_{FB}$$  \hspace{1cm} (7)

Under assumption (A3), the RHS of Equation 7 is bounded by

$$\sum_{k=1}^{K} |(b_{\lambda'}^{(l)})_k| = ||b_{\lambda'}^{(l)}||_1$$

Combined with assumption (A4), we see that

$$\pi \max \left\{ \sup_{\lambda} \sum_{\lambda' = 1}^{M_l} ||C^{(l)}b_{\lambda'}^{(l)}||_{FB} \cdot \frac{M_l-1}{M_l} \sum_{\lambda = 1}^{M_l} ||C^{(l)}b_{\lambda'}^{(l)}||_{FB} \right\} \leq \pi \max \left\{ \sup_{\lambda} \sum_{\lambda' = 1}^{M_l} ||b_{\lambda'}^{(l)}||_1 , \sup_{\lambda' = 1}^{M_l} \frac{M_l-1}{M_l} \sum_{\lambda = 1}^{M_l} ||b_{\lambda'}^{(l)}||_1 \right\} \leq 1$$ \hspace{1cm} (8)

From this, we see that the equivalent of condition of assumption (A2’) for AdDCF is satisfied and the stability result holds. For completeness, we state the theorem here.

**Theorem 1.** In an AdDCFNet with FB pre-bases, under assumptions (A0), (A1), (A3), (A4),

$$||x^{(L)}[x^{(0)}] - x^{(L)}|D_x x^{(0)}|| \leq (8L|\nabla \tau|_{\infty} + 2 \cdot 2^{-jL} |\tau|_{\infty})||x^{(0)}||$$

In summary, in this section we have extended the results from DCFNet [18] and obtained a theorem for the deformation stability of AdDCFNet.

4 Experiments

We now show some preliminary empirical evaluations of the performance of AdDCFNet in comparison to standard CNN and DCFNet on two tasks in computer vision: the high-level task of image recognition and low-level task of image super-resolution. In all the DCF and AdDCF models used in this section, we choose the bases for DCF and pre-bases for AdDCF to be the leading Fourier Bessel bases.

4.1 Image classification

4.1.1 Dataset description

**MNIST** is a benchmark dataset of gray-scale images of segmented and centered handwritten digits (0-9) [15]. We used 50,000 training examples, 10,000 validation examples, and 10,000 testing examples, where every image is of size $28 \times 28$ pixels.

4.1.2 Architecture details

We compared the classification performance of AdDCF with standard CNN and DCFNet using the base architecture listed in Table 1. For DCF, we let $K = 7$ for both of the two $5 \times 5$ convolutional layers. For DCF, we set $K = 7$ and $K_0 = 11$ for fair comparison.
4.1.3 Performance charts

We train the three models using the same training parameters: 120 epochs of Adam gradient descent with starting learning rate of 0.002 and learning rate decay by a factor of 0.1 at epoch 40 and 80. Since standard CNN has about several times more parameters than DCFNet and AdDCFNet, we apply a weight decay of $10^{-4}$. We list the test accuracies in Table 2. From this, we observe that AdDCF achieves the same level of accuracy as the standard CNN and DCF. More extensive experiments will be done in the future to further evaluate the relative performance of the three models on the task of image classification, using more complicated and deeper models on larger datasets like CIFAR 10, CIFAR 100, VGG-face, ImageNet, etc.

<table>
<thead>
<tr>
<th></th>
<th>CNN</th>
<th>DCFNet</th>
<th>AdDCFNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST test accuracy</td>
<td>99.37%</td>
<td>99.39%</td>
<td>99.43%</td>
</tr>
</tbody>
</table>

Table 2: Accuracy performance

4.2 Image Super-resolution

Image super-resolution aims to recover the high-resolution image from its low-resolution counterpart. We focus on the Single Image Super Resolution (SISR) problem formulation, where the corresponding low-resolution and high-resolution images are given as the input and target pair in a supervised learning setting. Our goal is to learn a neural network from these training examples that can take an input low-resolution image and output the high-resolution image.

4.2.1 Dataset description

We compose our training set using three standard datasets commonly used for image super-resolution: T91[21], General 100[5], the 200 training images from BSD300[16]. For validation, we choose the first 50 images in the new superresolution dataset DIV2K[2]. We evaluate our model’s performance on Set5[4], Set14[22], and the 200 test images from BSD500[16]. Peak signal-to-noise ratio (PSNR) between the ground truth high resolution image and the image produced by the network is used to evaluate the model performance. We focus on the scale $\times 3$ super-resolution in the current experiments.

4.2.2 Architecture details

We choose a simple and light-weight model FSRCNN [5] to conduct the preliminary experiments. The only change from the original model in [5] is that we use ReLU as the activation function instead of Parametric ReLU (PReLU) introduced in [8]. This is to ensure the model would satisfy the assumption made in the theory about nonexpansion of the activation function. We set the hyperparameters used in [5] to $d = 56$, 

Table 1: Image classification network architecture

<table>
<thead>
<tr>
<th>CNN architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv $5 \times 5 \times 1 \times 16$, stride 1, padding 0</td>
</tr>
<tr>
<td>ReLU</td>
</tr>
<tr>
<td>avg pool $2 \times 2$, stride 2</td>
</tr>
<tr>
<td>conv $5 \times 5 \times 16 \times 32$, stride 1, padding 0</td>
</tr>
<tr>
<td>ReLU</td>
</tr>
<tr>
<td>avg pool $2 \times 2$, stride 2</td>
</tr>
<tr>
<td>fc 128</td>
</tr>
<tr>
<td>ReLU</td>
</tr>
<tr>
<td>Dropout 0.5</td>
</tr>
<tr>
<td>fc 10</td>
</tr>
<tr>
<td>softmax</td>
</tr>
</tbody>
</table>
The architecture of the baseline standard convolutional FSRCNN model is shown in Table 3.

<table>
<thead>
<tr>
<th>FSRCNN architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv 5 × 5 × 1 × 56, stride 1, padding 2 - ReLU</td>
</tr>
<tr>
<td>conv 1 × 1 × 56 × 16, stride 1, padding 0 - ReLU</td>
</tr>
<tr>
<td>conv 3 × 3 × 16 × 16, stride 1, padding 1 - ReLU</td>
</tr>
<tr>
<td>conv 3 × 3 × 16 × 16, stride 1, padding 1 - ReLU</td>
</tr>
<tr>
<td>conv 3 × 3 × 16 × 16, stride 1, padding 1 - ReLU</td>
</tr>
<tr>
<td>conv 1 × 1 × 16 × 56, stride 1, padding 0 - ReLU</td>
</tr>
<tr>
<td>deconv 9 × 9 × 56 × 1, stride 3, padding 3</td>
</tr>
</tbody>
</table>

Table 3: Super-resolution network architecture

The input of the network is the Y channel from the YCbCr color decomposition of the low resolution image. The first 7 layers of the network preserves the spatial dimension of the input and the last deconvolutional layer uses a stride size of 3 to produce a high-resolution image with 3 times the width and height of the original input. The PSNR reported later is also evaluated on the Y-channel following the convention used in [5].

For DCFNet and AdDCFNet, we use the same network architecture and only make decisions about the number of bases (and number of pre-bases for AdDCF) used. We replace all the layers that have filters larger than 1 × 1 with DCF/ AdDCF convolutional layers respectively. We choose a consistent number of bases in all layers that have filters of the same shape. Let $K_n$ denote the number of bases and $K_n^0$ denote the number of pre-bases we use for layers with $n × n$ filters. We experiment with two configurations of number of bases choices. For one set, we let $K_5 = 11$, $K_3 = 5$, $K_9 = 13$ and $K_5^0 = 14$, $K_3^0 = 6$, $K_9^0 = 19$. The number of bases chosen is large and should give both model more flexibility in learning the underlying data distribution. We denote the DCFNet and AdDCFNet under these hyperparameter setting $DCFNet_+$ and $AdDCFNet_+$ where + means more bases. For the other set we let $K_5 = 7$, $K_3 = 3$, $K_9 = 7$ while keeping all the $K_n^0$ the same. This is the case where the number of bases used is quite limited, which should make DCFNet underfit. However, we hope to see that the AdDCFNet does not suffer from underfitting as much because its large flexibility of choosing from a large set of pre-bases to form the bases. We denote the DCFNet and AdDCFNet under these parameter setting $DCFNet_-$ and $AdDCFNet_-$ where – means fewer number of bases.

### 4.2.3 Performance Comparison

We use the same training setting for all the models displayed in Table 4. For every epoch, we loop through the entire training set, and for every training image, we randomly crop out a patch of size 72 × 72 and its corresponding low-resolution counterpart of size 24 × 24. We use the mean squared error per pixel over a batch of 16 images as the optimization objective. We use Adam optimizer with learning rate of $10^{-3}$ on the convolutional layers and $10^{-4}$ on the last deconvolutional layer. We train for a total of 40000 epochs and drop the learning rate by a factor of 0.1 after the 20000 epochs. The resulted test PSNR performances are shown in Table 4.

<table>
<thead>
<tr>
<th></th>
<th>Bicubic</th>
<th>$DCFNet_-$</th>
<th>$AdDCFNet_-$</th>
<th>$DCFNet_+$</th>
<th>$AdDCFNet_+$</th>
<th>FSRCNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set5[4]</td>
<td>30.39</td>
<td>31.14</td>
<td>32.54</td>
<td>32.41</td>
<td>32.79</td>
<td>33.23</td>
</tr>
<tr>
<td>Set14[22]</td>
<td>27.54</td>
<td>27.98</td>
<td>29.20</td>
<td>29.06</td>
<td>29.34</td>
<td>29.60</td>
</tr>
<tr>
<td>BSD200[16]</td>
<td>27.26</td>
<td>27.55</td>
<td>28.35</td>
<td>28.28</td>
<td>28.48</td>
<td>28.68</td>
</tr>
</tbody>
</table>

Table 4: PSNR performance

From this, we see that AdDCFNet outperforms DCFNet in both of the parameter settings. This matches our intuition and proof in Section 2 that AdDCFNet has more model expressiveness than DCFNet with the same number of bases. In addition, if we compare the difference between $AdDCFNet_+$ and $AdDCFNet_-$
and the difference between $DCFNet_+$ and $DCFNet_-$, we see that by using fewer bases, the performance drop of DCFNet is much more significant than the performance drop of $AdDCFNet$. In fact, we see that despite using a smaller number of bases, $AdDCFNet_-$ still outperforms $DCFNet_+$. 

### 4.2.4 Super-resolution image visualization

In Figure 1 and 2, we show the high-resolution images produced by bicubic interpolation, our model $AdDCFNet$, and standard FSRCNN together with the original high resolution image. We can easily see the better resolution of AdDCFNet than bicubic interpolation. In addition, although there is some PSNR value difference between standard FSRCNN and $AdDCFNet_+$, visually it is hard to distinguish the quality difference between images produced by these two models.

![Super-resolution images](image-url)

Figure 1: Super-resolution of Set14 image Baboon
4.2.5 Bases and Filter visualization

We visualize the pre-bases, bases, and filters of the first and last layer in AdDCFNet in Figure 3 and Figure 4. We see that for the first layer, the $5 \times 5$ filters become more regular and smooth in AdDCFNet than the standard FSRCNN's filters due to the factorization.

For the last deconvolutional layer, we see that the standard FSRCNN’s $9 \times 9$ filters are also generally smooth, but contain more higher frequency components than the filters learned by AdDCFNet. This difference is likely caused by the low-rank requirement made in AdDCFNet. Further experiments should be done to further investigate the bases and filters properties of the deconvolutional layer.

4.2.6 Denoising in Superresolution

To experimentally verify the representation stability of AdDCFNet, we test its performance when resolving low resolution images that are corrupted by gaussian white noises and salt and pepper noises. We also compare AdDCFNet’s performance with that of FSRCNN and DCFNet.

Specifically, we add mean 0, variance 0.005 gaussian noise (or 2% salt and pepper noise) to the low resolution images of Set5 and see how well the trained models reconstruct the correct high resolution images. The results are shown in Table 5.

We see that there is a significant drop in PSNR across all the models. In general, DCFNet is better at reconstructing the original high resolution image than AdDCFNet with the same number of bases. We do
Figure 3: Bases and filters of the first 5 × 5 convolutional layer

Table 5: Denoising performance

<table>
<thead>
<tr>
<th></th>
<th>DCFNet−</th>
<th>AdDCFNet−</th>
<th>DCFNet+</th>
<th>AdDCFNet+</th>
<th>FSRCNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set5 salt and pepper</td>
<td>24.66</td>
<td>23.66</td>
<td>24.10</td>
<td>23.69</td>
<td>23.26</td>
</tr>
<tr>
<td>Set5 gaussian noise</td>
<td>26.18</td>
<td>24.71</td>
<td>25.21</td>
<td>24.67</td>
<td>24.60</td>
</tr>
</tbody>
</table>
see that AdDCFNet has a slight advantage over standard FSRCNN, although the improvement is not very strong. We also notice that adding more bases from $AdDCFNet_-$ to $AdDCFNet_+$ does not alter much the PSNR performance. This is most likely because the number of pre-bases are kept the same between the two models. From this we hypothethize that $K_0$ has a larger impact on the model’s invariance to noise than $K$ for AdDCFNet. Further analysis needs to be done to understand the stability properties of AdDCFNet.
5 Conclusion and Future Directions

From Table 4 and Table 5 we see that AdDCFNet is more expressive than DCFNet at capturing the data distribution while being (slightly) more stable against input noises than the standard convolutional architecture. More experiments need to be done to ensure proper tuning of different networks’ training parameters and to verify the observed performance differences.

In addition, a future direction to pursue is to use the new assumptions (A3) and (A4) in theory section to develop $l_2$ and $l_1$ soft regularizations of the basis coefficients $C_{k}^{(l)}$ and filter coefficients $b_{\lambda,\lambda'}^{(l)}$. This might help further improve the empirical stability performance of AdDCF models.

References


