

A Frequency Domain Neural Network for Fast Image Super-resolution

Junxuan Li^{1*}

Shaodi You^{1,2}

Antonio Robles-Kelly^{1,2}

¹College of Eng. and Comp. Sci., Australian National University, Canberra, ACT 2601, Australia

²Datat61-CSIRO, Black Mountain Laboratories, Acton, ACT 2601, Australia

Abstract

In this paper, we present a frequency domain neural network for image super-resolution. The network employs the convolution theorem so as to cast convolutions in the spatial domain as products in the frequency domain. Moreover, the non-linearity in deep nets, often achieved by a rectifier unit, is here cast as a convolution in the frequency domain. This not only yields a network which is very computationally efficient at testing but also one whose parameters can all be learnt accordingly. The network can be trained using back propagation and is devoid of complex numbers due to the use of the Hartley transform as an alternative to the Fourier transform. Moreover, the network is potentially applicable to other problems elsewhere in computer vision and image processing which are often cast in the frequency domain. We show results on super-resolution and compare against alternatives elsewhere in the literature. In our experiments, our network is one to two orders of magnitude faster than the alternatives with an imperceptible loss of performance.

1. Introduction

Image super-resolution is a classical problem which has found application in areas such as video processing [7], light field imaging [3] and image reconstruction [9].

Given its importance, super-resolution has attracted ample attention in the image processing and computer vision community. Early approaches to super-resolution are often based upon the rationale that higher-resolution images have a frequency domain representation whose higher-order components are greater than their lower-resolution analogues. Thus, methods such as that in [29] exploited the shift and aliasing properties of the Fourier transform to recover a super-resolved image. Kim *et al.* [15] extended the method in [29] to settings where noise and spatial blurring are present in the input image. In a related development, in [4], super-resolution in the frequency domain is effected using Tikhonov regularization.

Alternative approaches, however, effect super-resolution

by *aggregating* multiple frames with complementary spatial information or by relating the higher-resolved image to the lower resolution one by a sparse linear system. For instance, Baker and Kanade [1] formulated the problem in a regularization setting where the examples are constructed using a pyramid approach. Protter *et al.* [22] used block matching to estimate a motion model and use exemplars to recover super-resolved videos. Yang *et al.* [31] used sparse coding to perform super-resolution by learning a dictionary that can then be used to produce the output image, by linearly combining learned exemplars.

Note that, the idea of super-solution “by example” can be viewed as hinging on the idea of learning functions so as to map a lower-resolution image to a higher-resolved one using exemplar pairs. This is right at the center of the philosophy driving deep convolutional networks, where the net is often considered to learn a non-linear mapping between the input and the output. In fact, Dong *et al.* present in [6] a deep convolutional network for single-image super-resolution which is equivalent to the sparse coding approach in [31]. In a similar development, Kim *et al.* [14] present a deep convolutional network inspired by VGG-net [26]. The network in [14] is comprised of 20 layers so as to exploit the image context across image regions. In [13], a multi-frame deep network for video super-resolution is presented. The network employs motion compensated frames as input and single-image pre-training.

2. Contribution

Here, we present a computationally efficient frequency domain deep network for super-resolution which, at input, takes the frequency domain representation of the lower resolution image and, at output, recovers the residual in the frequency domain. This residual can then be added to the lower resolution input image to obtain the super-resolved image. The network presented here is somewhat related to those above, but there are a number of important differences with respect to other approaches. Its important to note that:

- Following our frequency domain interpretation of deep networks, the convolutions in other networks are

*Corresponding author (email: u5990546@anu.edu.au)

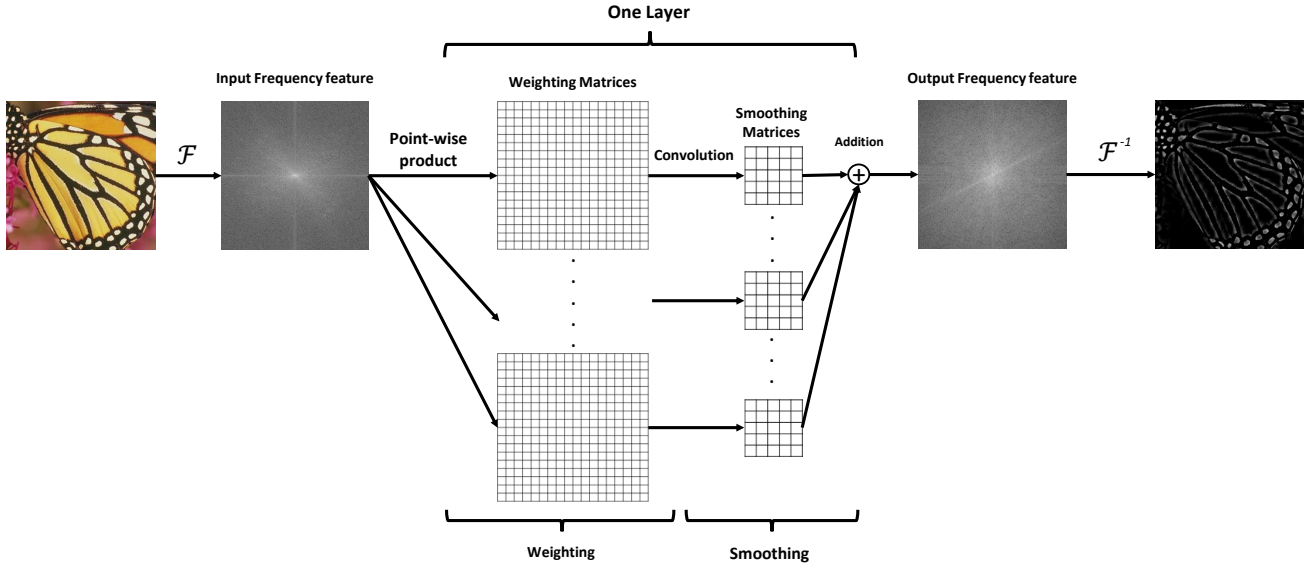


Figure 1. Diagram of our network with a single-layer. The input feature map obtained by transforming the image into the frequency domain is point-wise multiplied with the weighting matrices. After the weighting operation, the result is convolved and added to obtain the output frequency feature map that can then be transformed into the spatial domain to recover the predicted residual. Note that, in our network, the weighting matrices are the same size as the input feature map.

treated here as multiplications. This has the well known advantages of lower computational cost and added computational efficiency.

- Following the frequency domain treatment to the problem as presented here, the non-linearity in the network is given by convolutions in the frequency domain. This contrasts with the work in [23], which employs spectral pooling instead.
- In contrast with deep network architectures elsewhere, where the non-linearity is often attained using an activation function such as a rectified linear unit (ReLU) [10], we can learn the analogue convolutional parameters in the frequency domain in a manner akin to that used by CNNs in the spatial domain.
- We use residual training since its not only known to deal with the vanishing gradients well and often improve convergence, but its also particularly well suited to our net. This is following the notion that the lower resolved image lacks the higher frequencies in high-resolution imagery and, thus, these can be learned by the network based on the residual.
- Finally, we employ the Hartley transform as an alternative to the Fourier transform so as to avoid the need to process imaginary numbers.

3. Spatial and Frequency Domains

3.1. Convolutional Neural Networks

Note that most of the convolutional neural networks nowadays are variants of ImageNet [16]. Moreover, from a signal processing point of view, these networks can be considered to work on the “spatial” image domain¹. In these networks, each layer is comprised of a set of convolutional operations followed by an activation function.

To better understand the relationship between these networks in the spatial domain and ours, which operates in the frequency domain, recall that the two-dimensional discrete convolutions at the i^{th} layer can be expressed as

$$(f * g_j)[u, v] = \sum_{m=-M}^M \sum_{n=-N}^N f[m, n]g_j[u - m, v - n] \quad (1)$$

where f denotes the feature map delivered by the previous layer, *i.e.* that indexed $i - 1$, in the network and g_j is the convolutional kernel of order $(2M + 1) \times (2N + 1)$ in the corresponding layer. In the equation above, we have used u and v as the spatial coordinates (rows and columns) in the image under consideration.

At each layer, this convolutional stage is followed by an activation function which induces the non-linearity in the

¹Here, we adopt the terminology often used in image processing and computer vision when comparing spatial and frequency domain representations. We have done this so as to be consistent with longstanding work on integral transforms such as Fourier, Cosine and Mellin transforms elsewhere in the literature.

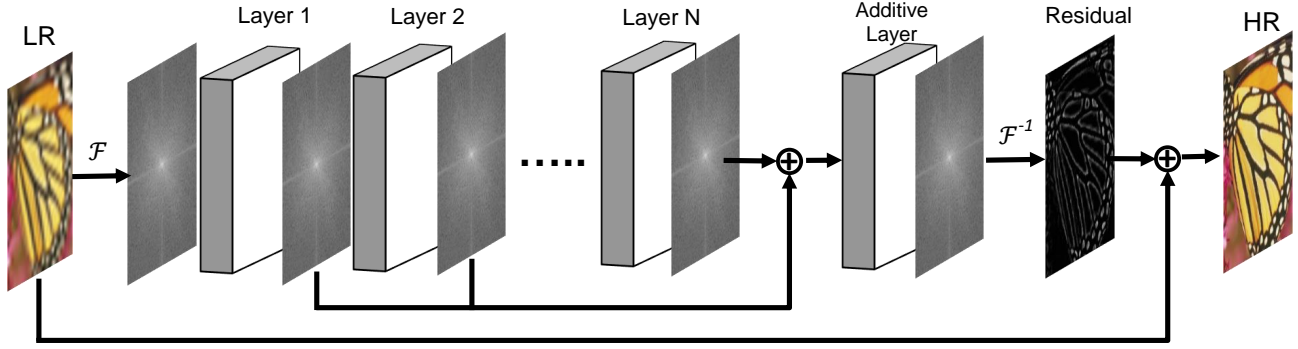


Figure 2. Simplified diagram of our network with multiple layers. Each of these layers accounts for a product-convolution-addition step as presented in Figure 1. Here, the lower-resolved image is transformed into the frequency domain and then, after the final additive layer, the predicted residual is transformed back into the spatial domain. The residual and input images are then added to obtain the higher-resolved output image.

behavior of the network. Nonetheless there are a number of choices of activation function, *i.e.* sigmoid, binary, identity, etc., the most widely used one is ReLU, which can be expressed mathematically using a product with a Heaviside function as follows

$$\begin{aligned} \text{ReLU}[u, v] &= \max(0, (f * g_j)[u, v]) \\ &= (f * g_j)[u, v] \text{HS}((f * g_j)[u, v]) \end{aligned} \quad (2)$$

where $\text{HS}(x)$ is the Heaviside function which yields 1 if $x > 0$, 0.5 if $x = 0$ and 0 if $x < 0$. In the expressions above, and so as to be consistent with Equation 1, we have used $(f * g_j)[u, v]$ as the input to the rectifier unit.

3.2. Fourier Transform

Equations 1 and 2 hint at the use of the convolution theorem of the Fourier transform to obtain an analogue of spatial domain CNNs in the frequency domain. Further, the convolution theorem as widely known for the Fourier transform has similar instantiations for other closely related integral transforms. For now, and for the sake of clarity, we will focus on the Fourier transform. Later on in the paper, we will elaborate further on the use of the Hartley transform as an alternative to the Fourier transform in our implementation.

The convolution theorem states that given two functions in spatial (time) domain, their convolution is given by their point-wise multiplication in the frequency domain. For the sake of consistency with Equations 1 and 2, let f and g_j be the two spatial domain functions under consideration and denote the Fourier transform operator as \mathcal{F} , then we have

$$\mathcal{F}\{f * g_j\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g_j\} \quad (3)$$

where \cdot denotes point-wise product.

Moreover, the converse relation also holds, whereby a product in the spatial domain becomes a convolution in the

frequency domain. *i.e.*

$$\begin{aligned} \mathcal{F}\{(f * g_j) \cdot h_j\} &= \mathcal{F}\{f * g_j\} * \mathcal{F}\{h_j\} \\ &= (\mathcal{F}\{f\} \cdot \mathcal{F}\{g_j\}) * \mathcal{F}\{h_j\} \end{aligned} \quad (4)$$

where we have employed h_j to denote a generic activation function which, in our case can also be learnt. Note that the second line in the equation above follows from substituting Equation 3 into the first line.

4. Network Structure

Note that, in the Equation 4, the term $\mathcal{F}\{g_j\}$ acts as a “weighting” operation in the frequency domain. That is, through the point-wise product operation, it curtails or amplifies the frequency domain components of $\mathcal{F}\{f\}$. These frequency weighting operations take place of the original convolution operation in spatial domain convolutional neural networks such as ImageNet [16]. Similarly, the non-linear operator given by the rectifier units in CNNs is now substituted, in the frequency domain, by a convolution. This can be viewed as a smoothing or regularization operation in the frequency domain.

In Figure 1, we show a single-layer instantiation of our frequency domain network. Note that, at input, the image is transformed into the frequency domain. Once this has been effected, the frequency weighting step takes place, *i.e.* the pointwise multiplication operation, and then a convolution is applied. Once the outputs of all the convolutional outputs are obtained, they are summed together by an additive layer and the inverse of the frequency domain transform is applied so as to obtain the final result. Its worth noting that we have not incorporated a spectral or spatial pooling layer in our network. This is not a major issue in our approach since these pooling layers are often used for classification tasks [16] whereas in other applications, such as super-resolution, pooling is seldom used [6].

4.1. Weighting

As mentioned above, the product $Q = \mathcal{F}\{f\} \cdot \mathcal{F}\{g_j\}$ can be viewed as a frequency weighting operation equivalent to the convolution operation in time domain. As before, consider a feature map f at a given layer in the network and the j^{th} convolutional kernel g_j .

For the layer under consideration, the product Q will take the frequency domain of the feature map $\mathcal{F}\{f\}$ as an input and point-wise multiply it by a wight matrix given by the values of $\mathcal{F}\{g_j\}$. In practice, both, $\mathcal{F}\{f\}$ and $\mathcal{F}\{g_j\}$ can be viewed as matrices which are the same size. This is important since it permits us to pose the problem of performing the forward pass and backpropagation steps in a manner analogous to that used in CNNs operating in the time domain.

To see this more clearly, denote as F^i the input matrix corresponding to $\mathcal{F}\{f\}$ to the i th layer of our frequency domain network. Similarly, let the j^{th} weight matrix corresponding to the coefficients of $\mathcal{F}\{g_j\}$ be W_j . The output of the product of the two matrices is another matrix, which we denote Q and whose entries indexed l, k are given given by

$$Q(l, k) = F^i(l, k)W_j(l, k) + B_j(l, k) \quad (5)$$

where $F^i(l, k)$ and $W_j(l, k)$ are the entries indexed l, k of the matrices F^i and W_j , respectively, and we have introduced the bias matrix B_j with entries $B_j(l, k)$.

Moreover, the Fourier transform of an image, being real and non-negative, is conjugate-symmetric². This is important since, by noting that W_j should be Hermitian, we can reduce the number of learnt weights by half.

4.2. Smoothing

As shown in Figure 1, once the weighting operation is effected, a convolution in the frequency domain, analogous to the rectification operation in the spatial domain is applied. This is inspired upon Equation 4, which hints at the notion that we can express the ReLU as a product between a Heaviside function and its argument. Again, in practice, this can be expressed as follows

$$R_j(l, k) = \sum_{m=-M}^M \sum_{n=-N}^N Q(l-m, k-n)C_j(m, n) \quad (6)$$

Where $Q(l-m, k-n)$ is the corresponding entry of the matrix Q as presented in the previous section and $C_j(m, n)$ are the coefficients of the matrix C containing the values of $\mathcal{F}\{h_j\}$.

From the equation above is straightforward to note that the entries of the matrix R are a linear combination of the

²It can be shown in a straightforward manner that this symmetry property also applies to the Hartley transform.

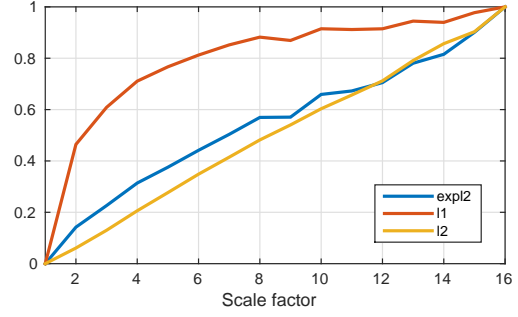


Figure 3. Loss comparison as a function of low-to-high resolution factors. All the loss function values, i.e. $l_1, l_2, \text{Exp-}l_2$, are computed in frequency domain and normalized to unit at their extrema.

values of Q where the matrix C can be viewed as a kernel that can be learnt. Thus, here, we consider the entries $C_j(m, n)$ of C_j as parameters that can be updated at each back-propagation step. This, in turn, allows us to learn both, the weights in W_j as well as the parameters in C_j .

4.3. Additive Layer

Recall that, in applications such as super-resolution, frequency domain approaches aim at recovering or predicting a whole frequency map corresponding to either the enhanced or super-resolved image. As a result, instead of a prediction layer, our network adds the output of all the network feature maps into a single one at output and then applies a final frequency weighting operation. This additive layer can be expressed as follows

$$P = \left(\sum_{i=1}^L \alpha_i S^i \right) \odot W_L \quad (7)$$

where L is the number of layers in the network, \odot denotes the Hadamard (entrywise) product, W_L is the final fre-

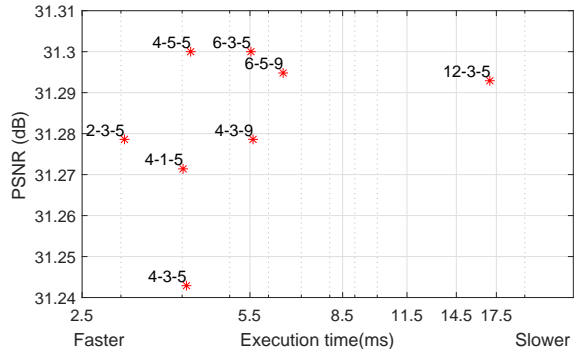


Figure 4. Cross-validation performance of our network as a function of testing time. The text over a point denotes the number of layers L , smoothing matrix size N and the number of weighting matrices per layer K , respectively. All the variants of the network were tested on the Set14 dataset with an upscaling factor of 2.

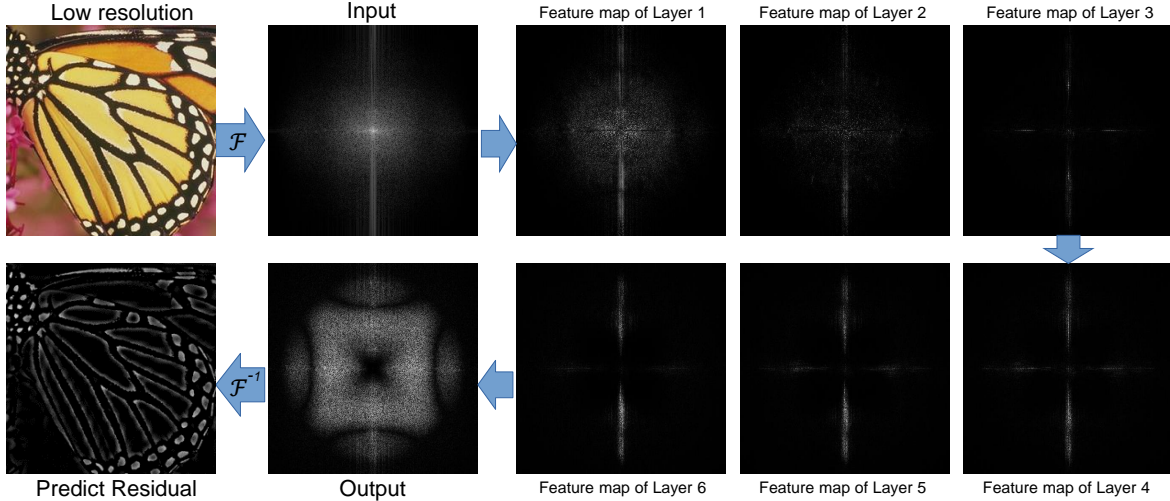


Figure 5. Frequency domain feature maps for each layer in our network. Here we also show the input image and predicted residual in the spatial domain.

quency weighting matrix, \mathbf{P} is the prediction of our network in the frequency domain and α_i is the weight that controls the contribution of the i^{th} layer feature map to the output.

In the equation above, \mathbf{S}^i is given by summing over all the matrices \mathbf{R}_j for each layer i . In our network, we do not require this sum to have independent weights since these can be absorbed, in a straightforward manner, into the matrices \mathbf{C}_j . Thus, for this layer, in practice, we only learn the matrix \mathbf{W}_L .

In Figure 2 we show the simplified diagram of our complete network. Note that the output of each layer is a frequency feature map which is the same size as the input. The output of each layer is then added and weighted by the additive layer. We then compute the spatial domain residual by applying the inverse transform to the frequency domain output of our network and add the predicted residual to the input image so as to compute the super-resolved output.

5. Implementation and Discussion

5.1. Hartley vs Fourier Transform

As mentioned earlier, the implementation of our network makes use of the Hartley transform [11] as an alternative to the Fourier transform. The reasons for this are twofold. Firstly, the Hartley transform is closely related to the Fourier transform and, hence, shares analogue properties. Secondly, the Hartley transform takes at input real numbers and delivers, at output, real numbers. This has the advantage that, by using the Hartley transform, we do not need to cater for complex numbers in our implementation.

Here, we have used the fast Hartley transform introduced by Bracewell [5]. The Hartley transform can be expressed using the real $\Re\{\cdot\}$ and imaginary $\Im\{\cdot\}$ parts of the Fourier

transform as follows

$$\mathcal{H}\{f\} = \Re\{\mathcal{F}\{f\}\} - \Im\{\mathcal{F}\{f\}\} \quad (8)$$

where we have used f for the sake of consistency with previous sections. Moreover, the Hartley transform is an involution, that is, the inverse is given by itself, *i.e.* $f = \mathcal{H}\{\mathcal{H}\{f\}\}$.

It is worth noting that, from Equation 8, its straightforward to show that, since the Fourier transform is linear as are the matrices \mathbf{W}_j , the weighting operation in our network applies, in a straightforward manner to the Hartley transform without any loss of generality. In the case of the Hartley transform, the convolution theorem has the same form as that of the Fourier transform [5], and, hence, we can write Equation 4 as follows

$$\mathcal{H}\{(f * g_j) \cdot h_j\} = (\mathcal{H}\{f\} \cdot \mathcal{H}\{g_j\}) * \mathcal{H}\{h_j\} \quad (9)$$

5.2. Training

For training our network, we have used the mini-batch gradient descent method proposed by LeCun *et al.* [18]. When training our network, the layers at the end, *i.e.* those closer to the final additive layer, tend to have a gradient that is small in magnitude as compared to the first layers. The reason being that, due to the architecture of our network, as shown in Figure 2, the first layers (those closer to the input) will accumulate the gradient contributions of the deeper layers in the network. This is a problem akin to that in [6], which was tackled by the VDSR [14] approach by applying gradient clipping in the back propagation step [20]. As a result, we follow [14] and normalize the gradient at each layer to a fixed range $(-\theta, \theta)$. Thus, the parameters at the layer can only change within a fixed range $(-\gamma\theta, \gamma\theta)$. In

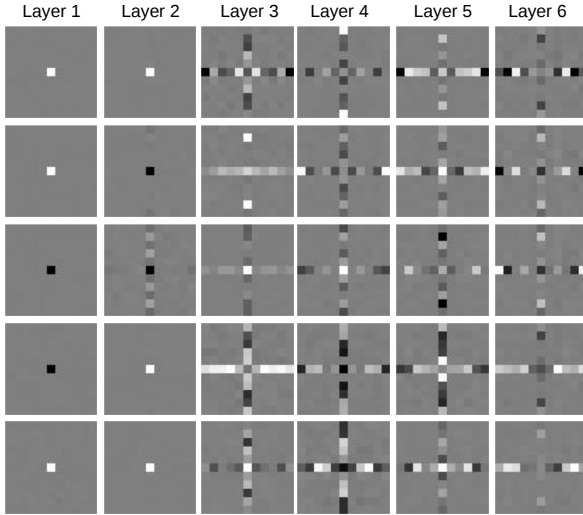


Figure 6. Smoothing matrices across the 6 layers in our network trained on the full dataset with an upscaling factor of 2. In the figure, the columns account for different layers.

our implementation, we set $\theta = 10^3$ and $\gamma = 10^{-5}$ for all layers.

5.3. Residual and Choice of Loss Function

For our loss function, we consider a number of alternatives. These are the L1 (l_1), L2 (l_2) and L2 with exponential decay (Exp- l_2) loss functions. These are defined as follows

$$l_1 \stackrel{\text{def}}{=} \|\mathbf{I} - \mathbf{I}^*\|_1 \quad (10)$$

$$l_2 \stackrel{\text{def}}{=} \|\mathbf{I} - \mathbf{I}^*\|_2^2 \quad (11)$$

$$\text{Exp-}l_2 \stackrel{\text{def}}{=} e^{\beta\omega_x} \|\mathbf{I} - \mathbf{I}^*\|_2^2 \quad (12)$$

where $\|\cdot\|_p$ denotes the p -norm under consideration, β is a hyper-parameter, \mathbf{I} is the matrix corresponding to the ground-truth high resolution image in the frequency domain and \mathbf{I}^* accounts for the frequency domain image recovered by our network. This is yielded by the sum of the prediction \mathbf{P} of our network as given in Equation 7 and the input to our network in the frequency domain, *i.e.* \mathbf{F}^1 , which can be expressed as

$$\mathbf{I}^* = \mathbf{P} + \mathbf{F}^1 \quad (13)$$

It is worth mentioning in passing that, in accordance with the observations made in [14], we also find that the use of residual learning improved the accuracy and converge speed of our network in the frequency domain. In Figure 3, we show a comparison of the three loss functions under consideration. In the figure, we show the loss values, normalized to unity, as a function of the low-resolution image scale factor of \mathbf{I} with respect to \mathbf{I} . In the figure, we have set $\beta = 0.01$. Note that the L2 loss is almost linear with respect to the scale factor. Moreover, in our experiments, we

found that the L2 loss performed the best with respect to both, convergence and speed. As a result, all the experiments shown hereafter employ the L2 loss.

6. Experiments

6.1. Datasets

Recall that DRRN[27] and VSDN[14] use 200 images in Berkeley Segmentation Dataset[19] combined with 91 images from Yang *et al.* [31]. This set of images has also been used for training in other approaches [17, 25]. These methods often use techniques such as data augmentation, *i.e.* application of transformations such as rotation, scaling and flipping transformations to generate novel images so as to complement the ones in the dataset. It is important to note, however, that these rotation, scaling and flipping in the spatial domain become frequency shift and scaling operations. Moreover, the dataset above, comprised of 291 images and their augmentation is, in practice, too small to allow for cross-validation of parameters.

Thus, here we have opted for a two-stage process using 5000 randomly selected images from the Pascal VOC2012 dataset [8] for training and Set5 [2], Set14 [32] and B100 [19] for testing. The first stage corresponds to a cross-validation process of the parameters used in our network employing 800 images out of the 5000 in our dataset for training and Set14 for testing. Also, for cross-validation, we have resized the image to 360×480 and used a scale factor of 2 on the dataset so as to obtain the images that are used as input to our network. After cross-validation, and once the parameters have been selected, the second stage is to proceed to train and test our network on the whole dataset and the three testing sets. In all our experiments, we have taken the color imagery and performed super-resolution on the \mathbf{Y} channel in the \mathbf{YCbCr} space [21].

6.2. Parameter Selection

We have selected, through cross-validation, the number of layers L in the network, the size of the matrices \mathbf{C}_j and the number of weighting matrices K per layer. In all our experiments we have used squared matrices \mathbf{C}_j , *i.e.* $N = M$, and chosen a base line network with $L = 4$, $N = 3$, $K = 5$. We have then progressively increased L , N and K so as to explore the trade-off between timing and performance.

In Figure 4, we show the PSNR as a function of timing for the combinations of L , K and N used in our cross-validation exercise. For the sake of clarity, the time axis is shown in a logarithmic scale. Note that, in general, the networks with 4 or 6 layers seem to deliver the best trade-off between performance and timing. In the figure, $L = 4$, $N = 5$ and $K = 5$ performs the best while $L = 6$, $N = 2$ and $K = 5$ also performs well. Thus, and bearing in mind that a deeper network is expected to perform better in a

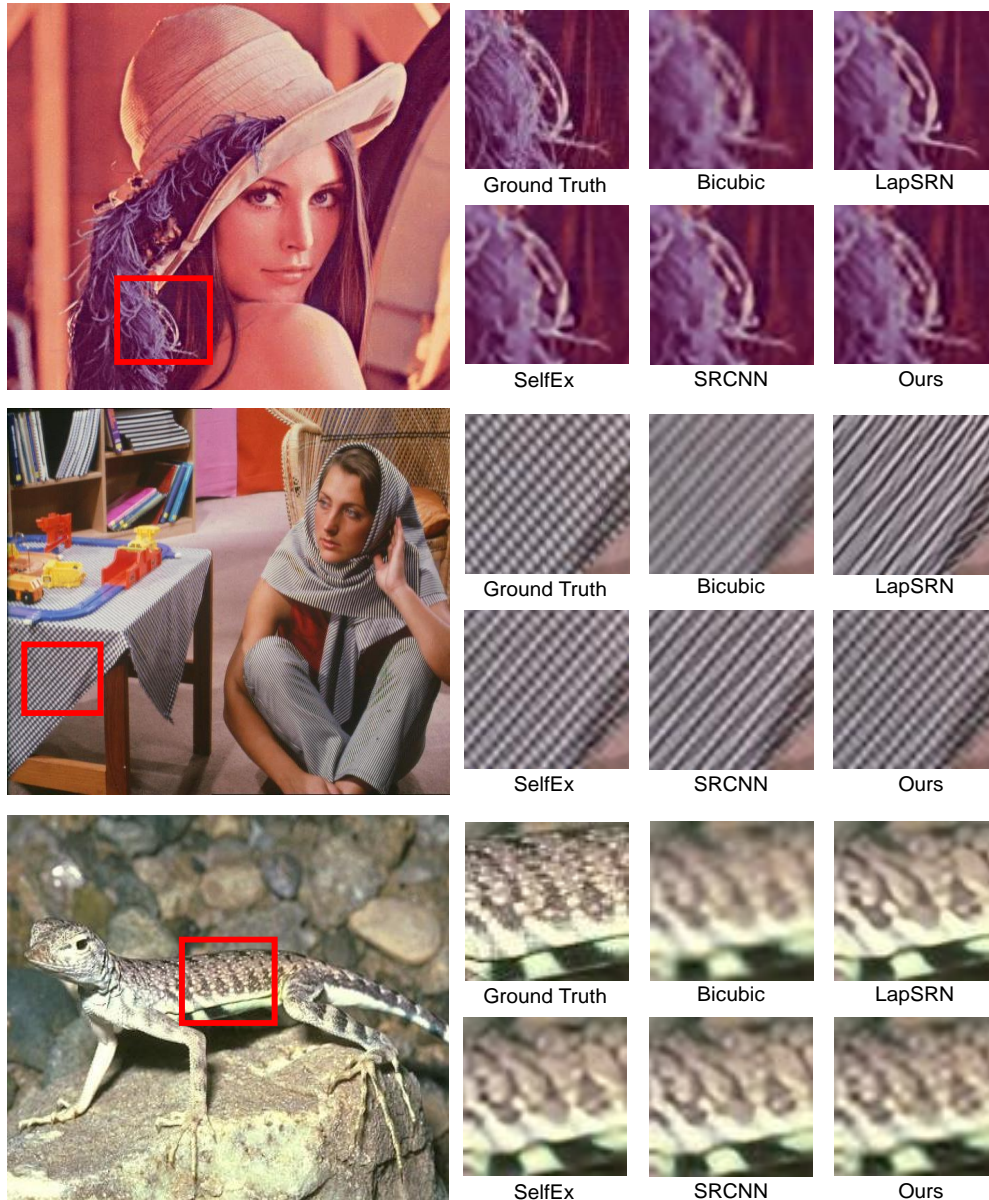


Figure 7. Qualitative comparison for an upscaling factor of 3 on “Lena”, “Barbara” and a sample image from the B100 dataset. Here we show a detail of the results yielded by our method and those delivered by LapSRN [17], SelfEx [12], SRCNN [6].

Dataset	Scale	Bicubic PSNR/SSIM	A+[28] PSNR/SSIM	SRCNN[6] PSNR/SSIM	VDSR[14] PSNR/SSIM	LapSRN[17] PSNR/SSIM	Ours PSNR/SSIM
Set5[2]	2	33.66/0.930	36.54/0.954	36.66/0.954	37.53/0.958	37.25/0.957	35.20/0.943
	3	30.39/0.868	32.58/0.908	32.75/0.909	33.66/0.921	34.06/0.924	31.42/0.883
	4	28.42/0.810	30.28/0.860	30.48/0.862	31.35/0.883	31.33/0.881	29.35/0.827
Set14[32]	2	30.24/0.868	32.28/0.905	32.42/0.906	33.03/0.912	32.96/0.910	31.40/0.895
	3	27.55/0.774	29.13/0.818	29.28/0.820	29.77/0.831	29.97/0.836	28.32/0.802
	4	26.00/0.702	27.32/0.749	27.49/0.750	28.01/0.767	28.06/0.768	26.62/0.727
B100[19]	2	29.56/0.843	31.21/0.886	31.36/0.887	31.90/0.896	31.68/0.892	30.58/0.877
	3	27.21/0.738	28.29/0.783	28.41/0.786	28.82/0.797	28.92/0.802	27.79/0.772
	4	25.96/0.667	26.82/0.708	26.90/0.710	27.29/0.725	27.22/0.724	26.42/0.696

Table 1. Quantitative evaluation of our method as compared to state-of-art super-resolution algorithms. Here we show the average PSNR/SSIM for upscale factors of 2, 3 and 4 for each of the three testing datasets.

larger dataset, for all the experiments shown here onwards, we set $L = 6$, $N = 5$ and $K = 5$.

6.3. Network Behavior

Now, we turn our attention to the behavior of the network in terms of the weighting and smoothing matrices. In Figure 5, we show the feature maps in the frequency domain as yielded by each of the network layers, *i.e.* the matrices \mathbf{F}^i . From the figure, we can appreciate that the feature map for the first layer mainly contains the low frequency information corresponding to the input image. As the layers go deeper, the feature maps become dominated by the high frequency components. This is expected since the aim of prediction in our net is given by the residual, which, in frequency domain, is mainly comprised by higher order frequencies.

In Figure 6, we show all the smoothing matrices in our network after the training has been completed. Surprisingly, note that matrices in each layer all appear to behave slightly different. In the first layer, the matrices are very much a delta in the frequency domain, while as the layer index increases, they develop non-null entries mainly along the central column and row. This follows the intuition that, for the first layer, the convolution would behave as a multiplicative identity removing lower-order frequency components, whereas, for further layers, the main contribution to its output is given by the central rows and columns.

6.4. Results

Finally, we present our results on image super-resolution. To this end, we first show some qualitative results and then provide a quantitative evaluation of our network performance and testing timing.

In Figure 7, we show a detail of the results yielded by our network and a number of alternatives on the “Barbara”, “Lena” images and a sample image from the B100 dataset. In all cases, we have applied an upscale factor of 3 and show, on the left-hand panel the full ground truth image. Note that our method yields results that are quite comparable to the alternatives. Moreover, for the detail of the “Barbara” image, the aliasing does not have a detrimental effect on the results. This contrasts with LapSRN, where the scarf stripes are over enhanced.

In Table 1 we show the performance of our network as compared to the alternatives. Here, we have used the average per-image peak signal-to-noise ratio (PSNR) [24] and the structural similarity index (SSIM) [30]. We have chosen these two image quality metrics due to a couple of reasons. Firstly, these have been used extensively for the evaluation of super-resolution results elsewhere in the literature. Secondly, the PSNR is a signal processing approach based upon the mean-squared error whereas the SSIM is a structural similarity measure. From the table, we can observe

that despite LapSRN is the best performer, our method is often no more than 2 decibels below LapSRN in terms of the PSNR and within a 0.05 difference in the SSIM.

Further, in Figure 8, we show the average per-image testing time, in milliseconds, for the three test datasets under consideration and three upscale factors, *i.e.* 2, 3 and 4. For all testing datasets our network far outperforms the alternatives, being approximately an order of magnitude faster than LapSRN and more than two orders of magnitude faster than SRCNN.

7. Conclusions

In this paper, we have presented a computationally efficient frequency domain neural network for super-resolution. The network can be viewed as a frequency domain analogue of spatial domain CNNs. To our knowledge, this is the first network of its kind, where rectifier units in the spatial domain are substituted by convolutions in the frequency domain and vice versa. Moreover, the network is quite general in nature and well suited for other applications in computer vision and image processing which are traditionally tackled in the frequency domain. We have presented results an comparison with alternatives elsewhere in the literature. In our experiments, our network is up to more than two orders of magnitude faster than the alternatives with an imperceptible loss of performance.

References

- [1] S. Baker and T. Kanade. Limits on super-resolution and how to break them. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1167–1183, 2002. 1
- [2] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. 2012. 6, 7
- [3] T. Bishop, S. Zanetti, and P. Favaro. Light field super-resolution. In *IEEE International Conference on Computational Photography*, 2009. 1
- [4] N. K. Bose, H. C. Kim, and H. M. Valenzuela. Recursive implementation of total least squares algorithm for image reconstruction from noisy, undersampled multiframes. In *IEEE Conference on Acoustics, Speech and Signal Processing*, volume 5, pages 269–272, 1993. 1
- [5] R. N. Bracewell. The fast hartley transform. *Proceedings of the IEEE*, 72(9):10101018, 1984. 5
- [6] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2016. 1, 3, 5, 7
- [7] P. E. Eren, M. I. Sezan, and A. M. Tekalp. Robust, object-based high resolution image reconstruction from low-resolution video. *IEEE Transactions on Image Processing*, 6(10):1446–1451, 1997. 1
- [8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) chal-

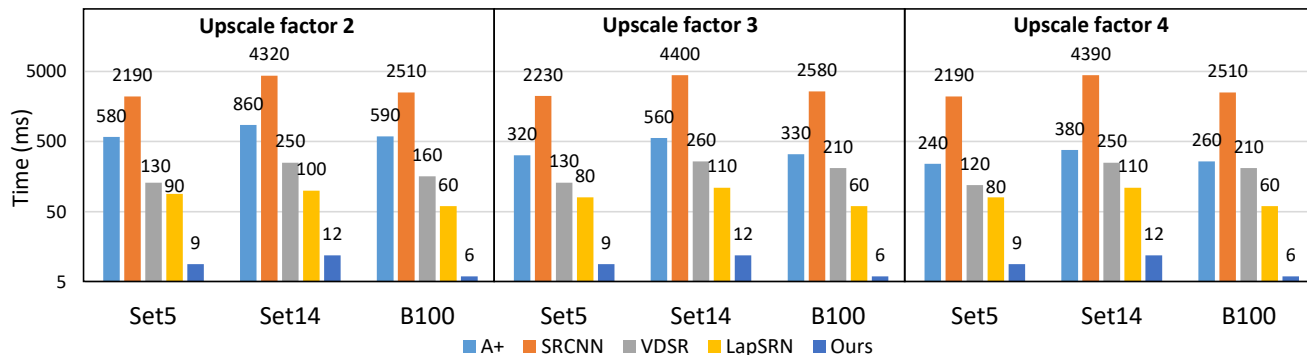


Figure 8. Average per-image testing time for our method and the alternatives in milliseconds for upscale factors 2, 3 and 4 on the three testing datasets. Our methods is a least 10 times faster than LapSRN, 20 times faster than VDSR, 40 times faster and A+ and 200 times faster than SRCNN.

- lence. *International Journal of Computer Vision*, 88(2):303–338, June 2010. 6
- [9] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar. Fast and robust multi-frame super-resolution. *IEEE Transactions on Image Processing*, 13:1327–1344, 2003. 1
- [10] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011. 2
- [11] R. Hartley. A more symmetrical fourier analysis applied to transmission problems. 30:144 – 150, 04 1942. 5
- [12] J.-B. Huang, A. Singh, and N. Ahuja. Single image super-resolution from transformed self-exemplars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5197–5206, 2015. 7
- [13] A. Kappeler, S. Yoo, Q. Dai, and A. K. Katsaggelos. Video super-resolution with convolutional neural networks. *IEEE Transactions on Computational Imaging*, 2(2):109–122, 2016. 1
- [14] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1646–1654, 2016. 1, 5, 6, 7
- [15] S. P. Kim, N. K. Bose, and H. M. Valenzuela. Recursive reconstruction of high resolution image from noisy undersampled multiframe. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38(6):1013–1027, 1990. 1
- [16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 2, 3
- [17] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 6, 7
- [18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 5
- [19] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int’l Conf. Computer Vision*, volume 2, pages 416–423, July 2001. 6, 7
- [20] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013. 5
- [21] C. Poynton. *Digital video and HD: Algorithms and Interfaces*. Elsevier, 2012. 6
- [22] M. Protter and M. Elad. Super resolution with probabilistic motion estimation. *IEEE Transactions on Image Processing*, 18(8):1899–1904, 2009. 1
- [23] O. Rippel, J. Snoek, and R. P. Adams. Spectral representations for convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 2449–2457, 2015. 2
- [24] D. Salomon. *Data compression: the complete reference*. Springer Science & Business Media, 2004. 8
- [25] S. Schuler, C. Leistner, and H. Bischof. Fast and accurate image upscaling with super-resolution forests. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3791–3799, 2015. 6
- [26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014. 1
- [27] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. 6
- [28] R. Timofte, V. De Smet, and L. Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *Asian Conference on Computer Vision*, pages 111–126. Springer, 2014. 7
- [29] R. Y. Tsai and T. S. Huang. Multipleframe image restoration and registration. In *Advances in Computer Vision and Image Processing*, pages 317–339, 1984. 1
- [30] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 8
- [31] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *IEEE transactions on image processing*, 19(11):2861–2873, 2010. 1, 6
- [32] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In *International conference on curves and surfaces*, pages 711–730. Springer, 2010. 6, 7