Memory- and time-efficient dense network for single-image super-resolution

Imanpour, Nasrin; Naghsh-Nilchi, Ahmad Reza; Monadjemi, Amirhassan; Karshenas, Hossein; Nasrollahi, Kamal; Moeslund, Thomas B.

Published in:
IET Signal Processing

DOI (link to publication from Publisher):
10.1049/sil2.12020

Creative Commons License
CC BY 4.0

Publication date:
2021

Document Version
Publisher's PDF, also known as Version of record

Link to publication from Aalborg University

Citation for published version (APA):
Imanpour, N., Naghsh-Nilchi, A. R., Monadjemi, A., Karshenas, H., Nasrollahi, K., & Moeslund, T. B. (2021). Memory- and time-efficient dense network for single-image super-resolution. *IET Signal Processing*, *15*(2), 141-152. Advance online publication. https://doi.org/10.1049/sil2.12020

Received: 7 July 2020 | Revised: 22 October 2020 | Accepted: 26 October 2020

DOI: 10.1049/sil2.12020

The Institution of Engineering and Technology WILEY

ORIGINAL RESEARCH PAPER

Memory- and time-efficient dense network for single-image super-resolution

Nasrin Imanpour[1] | **Ahmad R. Naghsh-Nilchi**[1] | **Amirhassan Monadjemi**[2] | **Hossein Karshenas**[1] | **Kamal Nasrollahi**[3,4] | **Thomas B. Moeslund**[3]

[1]Department of Computer Engineering, University of Isfahan, Isfahan, Iran

[2]School of Continuing and Lifelong Education, National University of Singapore, Singapore 138607

[3]Department of Architecture Design and Media Technology, Aalborg University, Aalborg, Denmark

[4]Research Department of Milestone Systems A/S, Copenhagen, Denmark

Correspondence
Ahmad R. Naghsh-Nilchi, Department of Computer Engineering, University of Isfahan, Isfahan, Iran.
Email: nilchi@eng.ui.ac.ir

Abstract

Dense connections in convolutional neural networks (CNNs), which connect each layer to every other layer, can compensate for mid/high-frequency information loss and further enhance high-frequency signals. However, dense CNNs suffer from high memory usage due to the accumulation of concatenating feature-maps stored in memory. To overcome this problem, a two-step approach is proposed that learns the representative concatenating feature-maps. Specifically, a convolutional layer with many more filters is used before concatenating layers to learn richer feature-maps. Therefore, the irrelevant and redundant feature-maps are discarded in the concatenating layers. The proposed method results in 24% and 6% less memory usage and test time, respectively, in comparison to single-image super-resolution (SISR) with the basic dense block. It also improves the peak signal-to-noise ratio by 0.24 dB. Moreover, the proposed method, while producing competitive results, decreases the number of filters in concatenating layers by at least a factor of 2 and reduces the memory consumption and test time by 40% and 12%, respectively. These results suggest that the proposed approach is a more practical method for SISR.

1 | INTRODUCTION

Single-image super-resolution (SISR), which aims to restore rich details and/or pleasant visual quality to an image, is favoured in many fields, including surveillance, remote sensing, and medical imaging. SISR is a classic problem, nevertheless a challenging open research problem in computer vision because of its ill-posed nature, that is, being under-determined. In detail, the low-resolution (LR) image ($\boldsymbol{y}$) is formed using [1]:

$$\boldsymbol{y} = \boldsymbol{Dx} + \boldsymbol{v}, \tag{1}$$

where, D, $\boldsymbol{x}$, and $\boldsymbol{v}$ stand for degradation process, high-resolution (HR) image, and additive noise, respectively. Degradation operators mostly include blurring and down-sampling. The information loss in the degradation process is high. Therefore, there exist various images that can be reduced to the observed LR image by applying Equation (1). That is especially problematic in larger up-scaling factors because the SISR is more ill-posed in these cases.

For decades, there has been consistent progress in developing and improving SISR techniques, which are documented in several surveys [1,2]. These techniques are in three main categories, namely interpolation-based, reconstruction-based, and learning-based methods [3–6]. Interpolation- and reconstruction-based methods have the problem of preserving information. Due to the significant learning ability of deep convolutional neural networks (CNNs) and their hierarchical property, they have been widely used in the single-image super-resolution task recently. CNNs learn an end-to-end mapping between the LR image and its counterpart HR image.

As a pioneer, Dong et al. proposed the first convolutional neural network for the SISR problem [3]. The problem with this network is the slow convergence that prevents it from increasing in depth. Kim et al. addressed this problem with a skip connection that adds the input and output of the network via element-wise addition and proposed two 20-layer CNNs, by increasing the recursion depth [7] or stacking weight layers [8]. Residual connection alleviates the vanishing gradient problem in training deeper networks. In other research, Tai et al. proposed another recursive network with depth 52 [9]. They used recursion on the local residual unit. Ledig et al. also proposed a residual CNN that combines local and global residual learning [10]. However, their proposed CNN uses a late up-scaling strategy and does not use shared weights, reaching promising results. Lim et al. improved Ledig et al.'s method in different ways, including by omitting batch normalisation [11].

With the advent of dense CNNs, recent SISR methods using them have achieved superior results. The number of output feature-maps of dense network layers is defined as the growth rate. Dense networks ensure maximum information flow by connecting each layer to every other layer in a feed-forward manner, as shown in Figure 1. These networks using channel-wise concatenations provide several other advantages, for example, they use the collective knowledge of hierarchical features and avoid learning redundant feature-maps. However, this kind of CNN consumes high GPU memory due to the dense concatenation.

Tong et al. used dense connections in the whole network, and set the growth rate to 16 to prevent the network from growing too wide [12]. Tai et al. proposed dense connections for image restoration in a global way and between modules named memory blocks [13]. Zhang et al. proposed dense blocks for SISR, which employ dense connections inside blocks of limited depth [14], as shown in Figure 1. They also use a 1 × 1 convolutional layer to reduce the channel number. Recent SISR methods have used either local [15–17] or global dense connections [18] in their proposed CNNs with promising results. Local dense connections are between convolutional layers [15,16] or residual units [17]. Utilising a larger growth rate enriches the concatenating feature-maps and leads to an overall superior discriminative ability. However, existing methods do not handle the memory problem caused by larger growth rates.

Increasing the growth rate does indeed produce richer features, but it also produces some irrelevant feature-maps that increase memory usage and testing. A composite layer for learning concatenating feature-maps is therefore proposed. In other words, a wider convolutional layer is used before concatenating layers to learn richer feature-maps. This layer is then followed by a slim layer to extract relevant information from the input feature-maps. The proposed method, which is shown in Figure 2, is inspired by the concept of dimensionality reduction to reduce the memory usage of dense CNNs. It has the flexibility to tune the number of filters in the odd and even layers to get an efficient trade-off between the representational power and memory usage of dense CNNs.

Comprehensive experiments justify the efficiency of the proposed method. It has been examined at four depths: 4, 8, 16, 32. On average, the proposed method decreases the number of concatenating feature-maps, resulting in 24% and 6% less GPU memory usage and test time, respectively, compared to the basic dense method. It also improves the peak signal-to-noise ratio by 0.24 dB. Moreover, with the proposed method, the growth rate can be reduced by at least a factor of 2 to lower the memory and time usage by 40% and 12%, respectively, while keeping the results competitive.

In summary, the proposed method has the following advantages:

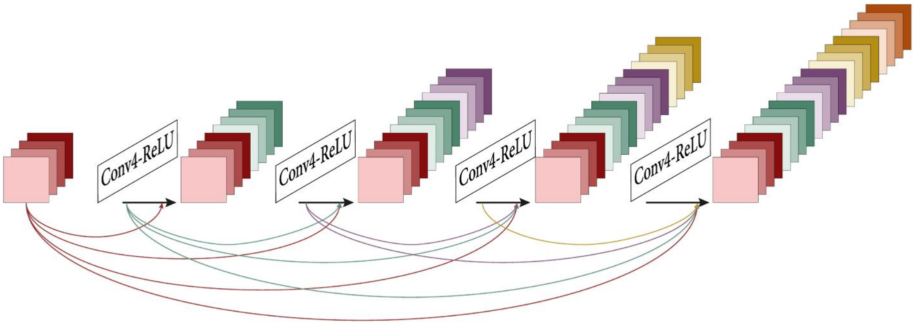

FIGURE 1 The basic dense method (Conv4 stands for a convolutional layer with four filters)

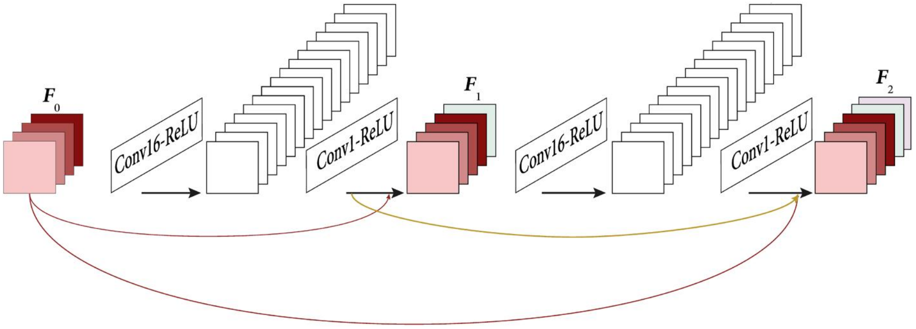

FIGURE 2 The proposed dense method (Conv16 stands for a convolutional layer with 16 filters, and Conv1 stands for a convolutional layer with one filter)

- It lowers the need for larger growth rates to increase the discriminative capability of dense CNNs, and
- It significantly reduces GPU memory consumption and test time by propagating only the representative concatenating feature-maps.

A review of related works is provided in Section 2. The proposed method is detailed in Section 3. Then, the data sets, training setup, network parameters, and experimental results are described in Section 4, and finally, the paper is concluded in Section 5.

2 | RELATED WORK

With the ability to automatically learn informative hierarchical features, CNNs have been extensively studied in recent years. In the following subsections, related concepts from dense CNNs and dimensionality reduction, which help to comprehend the proposed method, are presented.

2.1 | Dense CNNs

Recent works have shown that shorter connections between layers close to the input and those close to the output of convolutional networks make them substantially deeper, more accurate, and efficient to train. Dense CNNs concat feature-maps in other layer's input (with matching feature-map sizes) in a feed-forward manner [19], shown in Figure 1. Dense CNNs have several advantages: they alleviate the vanishing gradient problem, strengthen feature propagation, and reduce redundant feature-maps.

Recent SISR methods have also witnessed these advantages from dense CNNs. Tai et al. [13] used a densely connected structure in a global way and between memory blocks. In each memory block, they used successive residual units to learn multi-level representations of the current state, concatenating with outputs of previous memory blocks. At the end of each memory block, they applied a pre-activated 1×1 convolutional layer. Their experimental results show the efficiency of the long-term dense connections for image restoration tasks. Tong et al. [12] and Zhang et al. [14] proposed dense blocks for SISR similar to that shown in Figure 1. Tong et al. also used dense connections between blocks for improved results and set the growth rate to 16, preventing the network from growing too wide, and used a 1×1 convolutional layer after all blocks, to reduce the number of feature-maps. Zhang et al. used local residual learning between the input and output of dense for improved performance. The output of each block has a direct connection to all the layers of the next block to support contiguous memory among blocks. Finally, they concatenated the outputs of all blocks to use the hierarchical features of the input LR image for reconstruction. They also utilised 1×1 convolutional layers to adaptively preserve features and stabilise the training of the wider network.

More recently researchers have used dense connections in their proposed methods. Shamsolmoali et al. used dense blocks with dilated convolutional layers to increase the receptive field [15]. Anwar et al. proposed a densely residual Laplacian network, and used a local dense connection between residual units [15]. Qin et al. proposed a multi-resolution space-attended residual dense network with an adaptive fusion block based on channel-wise sub-network attention [15]. Dai et al. used channel-wise attention mechanisms to extract more informative and discriminative representations [18]. The main problem arising from dense CNNs is the memory consumption, especially in larger growth rates.

In a dense block, if each layer produces G feature-maps, then the l^{th} layer receives $G_0 + G \times (l - 1)$ feature-maps of previous layers, where G_0 is the number of received feature-maps in the first layer and G is referred to as the growth rate. Using larger depth (L) and growth rate (G) increases the number of feature-maps to be kept in memory. Previous methods prevented the memory problem of dense CNNs by various means, including using less concatenate layers, using a smaller growth rate, or using a 1×1 convolutional layer to reduce the channel number. The concept of dimensionality reduction, a two-step concatenate feature-map learning, is proposed to produce reduced and representative concatenate feature-maps. The proposed method described in the following sections significantly improves the memory usage without loss of information.

2.2 | Dimensionality reduction

In general, well-performing features have several characteristics, including (1) being representative to provide a concise description, and (2) being independent, as dependent features are redundant. Dimensionality reduction is concerned with reducing the number of features to generate more compact and representative features. The main problems with high-dimensional data are when many features are irrelevant or redundant. Therefore, such features increase memory usage and test time without useful function.

There are two general approaches for dimensionality reduction: feature selection and feature extraction. The central premise when using a feature selection technique is that the data contain some features that are either redundant or irrelevant and can thus be removed without incurring much loss of information [20]. Feature extraction creates new features based on the original feature set intended to be informative and non-redundant. It usually involves transforms to get relevant information from the input features, so that the desired task is performed by using this reduced representation instead of the complete initial one. The transforms may be linear or non-linear. However, the best transform is most likely a non-linear function.

A novel approach is proposed here for reducing the memory consumption and test time of dense CNNs inspired by the dimensionality reduction concepts mentioned above.

3 | PROPOSED METHOD

3.1 | Network architecture

The network architecture used for experiments is shown in Figure 3. This is the common architecture being used in most SISR techniques. The network learns an end-to-end mapping from LR images (I_{LR}) to HR images (I_{HR}). The output of the network is named I_{SR}, and is an approximation of I_{HR}.

Low-level feature-maps F_{-1} and F_0 are extracted using two convolutional layers:

$$\boldsymbol{F}_{-1} = \boldsymbol{W}^{-1}\boldsymbol{I}_{\mathrm{LR}}, \tag{2}$$

$$\boldsymbol{F}_{0} = \boldsymbol{W}^{0}\boldsymbol{F}_{-1}, \tag{3}$$

where W^{-1} and W^{0} are the weights of these two convolutional layers. The bias term is omitted for simplicity.

The proposed dense method has F_0 as input and learns residual multi-level feature-maps. If the number of levels of the proposed method is considered to be D, the input to the d-th level is denoted by F_d, for $d = 1, 2, \ldots, D$. Each level applies a non-linear transform $\mathrm{H}_d(.)$ consisting of two convolutional layers:

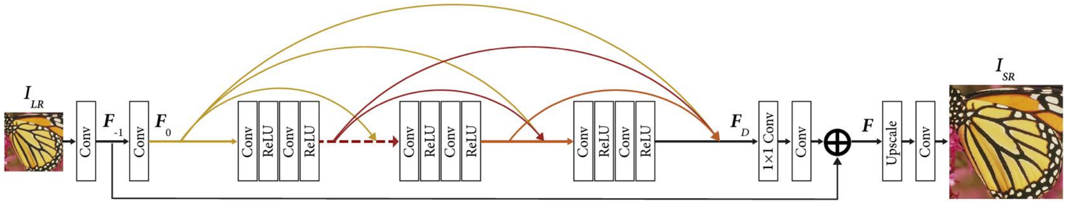

FIGURE 3 The architecture of the network used for SISR

TABLE 1 Results for $L = 4$

Model	Memory (MiB)	Time (ms)	Parameters (million)	Set5		Set14		B100		Urban100		Manga109		Average	
‘16’	669	200	0.3	37.24	0.9575	32.82	0.9106	31.59	0.8920	30.07	0.9046	36.79	0.9719	32.99	0.9242
‘32’	695	200	0.4	37.38	0.9581	32.94	0.9118	31.73	0.8940	30.56	0.9116	36.88	0.9727	33.22	0.9273
‘64’	761	198	06	37.46	0.9586	33.085	0.9136	31.84	0.8954	30.84	0.9148	37.22	0.9741	33.46	0.9292
‘128’	857	199	1.4	37.60	0.9592	33.18	0.9146	31.92	0.8966	31.10	0.9178	37.55	0.9748	33.68	0.9308
‘256’	1131	201	4.4	37.67	0.9594	33.28	0.9154	31.98	0.8973	31.31	0.9202	37.73	0.9752	33.83	0.9319
‘64-16’	655	196	0.3	37.32	0.9580	32.92	0.9118	31.71	0.8939	30.44	0.9099	36.96	0.9730	33.20	0.9268
‘128-16’	663	197	0.4	37.45	0.9585	33.00	0.9126	31.77	0.8946	30.57	0.9116	37.23	0.9738	33.36	0.9278
‘256-16’	687	195	0.6	37.38	0.9583	33.00	0.9130	31.78	0.8950	30.69	0.9131	37.15	0.9737	33.37	0.9284
‘512-16’	767	196	1.0	37.54	0.9588	33.09	0.9134	31.85	0.8955	30.85	0.9145	37.44	0.9743	33.54	0.9292
‘1024-16’	917	196	1.9	37.56	0.9588	33.12	0.9139	31.87	0.8959	30.97	0.9163	37.38	0.9742	33.57	0.9299
‘128-32’	673	196	0.5	37.53	0.9586	33.07	0.9133	31.84	0.8955	30.81	0.9144	37.35	0.9741	33.50	0.9291
‘256-32’	695	195	0.8	37.38	0.9587	33.07	0.9137	31.84	0.8957	30.91	0.9158	37.17	0.9742	33.46	0.9296
‘512-32’	783	196	1.3	37.60	0.9591	33.17	0.9144	31.90	0.8964	31.03	0.9170	37.54	0.9747	33.65	0.9304
‘1024-32’	933	195	2.3	37.64	0.9592	33.21	0.9147	31.92	0.8965	31.14	0.9183	37.46	0.9747	33.67	0.9309
‘256-64’	721	196	1.0	37.58	0.9593	33.17	0.9145	31.90	0.8963	31.04	0.9171	37.49	0.9748	33.64	0.9305
‘512-64’	811	193	1.7	37.65	0.9594	33.17	0.9145	31.93	0.8968	31.15	0.9187	37.52	0.9748	33.69	0.9311
‘1024-64’	967	197	3.2	37.70	0.9594	33.27	0.9151	31.97	0.8970	31.24	0.9192	37.71	0.9751	33.80	0.9314
‘512-128’	845	197	2.6	37.76	0.9596	33.32	0.9153	31.99	0.8971	31.29	0.9198	37.89	0.9754	33.88	0.9318
‘1024-128’	1027	199	5.0	37.75	0.9594	33.28	0.9150	31.98	0.8973	31.39	0.9209	37.75	0.9752	33.86	0.9321
‘512-256’	947	196	4.4	37.78	0.9598	33.32	0.9156	32.01	0.8975	31.45	0.9217	37.93	0.9757	33.95	0.9326

Note: In the first column, symbol G indicates the number of filters in each layer of the basic dense block, $G_1 - G_2$ stands for the number of filters of the proposed method in layers before concatenate layers, and the concatenate layers, respectively. Other columns represent the memory usage, average test time for the B100 data set, number of network parameters, average PSNR/SSIM for each test data set, and average PSNR/SSIM for all five test data sets, respectively.

TABLE 2 Results for $L = 8$

Model	Memory (MiB)	Time (ms)	Parameters (million)	Set5		Set14		B100		Urban100		Manga109		Average	
'16'	759	235	0.4	37.46	0.9585	33.02	0.9126	31.79	0.8948	30.69	0.9129	37.20	0.9736	33.39	0.9282
'32'	847	232	0.7	37.60	0.9590	33.16	0.9140	31.91	0.8962	31.04	0.9168	37.51	0.9747	33.65	0.9303
'64'	1025	233	1.6	37.76	0.9596	33.28	0.9149	31.97	0.8969	31.26	0.9196	37.79	0.9753	33.83	0.9316
'128'	1391	233	5.0	37.88	0.9601	33.42	0.9162	32.08	0.8983	31.68	0.9238	38.04	0.9760	34.08	0.9336
'256'	2303	254	18.1	37.93	0.9603	33.49	0.9171	32.13	0.8990	31.86	0.9260	38.07	0.9762	34.17	0.9346
'64-16'	701	229	0.5	37.58	0.9590	33.13	0.9138	31.88	0.8959	30.91	0.9154	37.43	0.9745	33.57	0.9297
'128-16'	735	227	0.7	37.58	0.9592	33.18	0.9144	31.92	0.8967	31.11	0.9181	37.48	0.9747	33.66	0.9309
'256-16'	769	226	1.2	37.71	0.9596	33.27	0.9149	31.97	0.8971	31.24	0.9192	37.77	0.9752	33.82	0.9315
'512-16'	881	227	2.2	37.76	0.9596	33.30	0.9152	32.00	0.8974	31.39	0.9210	37.74	0.9753	33.86	0.9322
'1024-16'	1101	228	4.1	37.79	0.9598	33.38	0.9155	32.03	0.8979	31.49	0.9222	37.73	0.9752	33.91	0.9327
'128-32'	753	225	0.9	37.66	0.9593	33.25	0.9148	31.98	0.8973	31.29	0.9200	37.64	0.9750	33.79	0.9317
'256-32'	795	227	1.6	37.76	0.9598	33.30	0.9156	32.03	0.8978	31.44	0.9215	37.86	0.9755	33.93	0.9326
'512-32'	911	230	2.9	37.85	0.9599	33.38	0.9159	32.06	0.8982	31.56	0.9228	38.01	0.9759	34.03	0.9332
'1024-32'	1159	228	5.6	37.85	0.9600	33.40	0.9160	32.07	0.8984	31.64	0.9236	37.89	0.9758	34.02	0.9335
'256-64'	875	225	2.3	37.78	0.9597	33.33	0.9158	32.04	0.8981	31.53	0.9225	37.73	0.9753	33.92	0.9329
'512-64'	999	226	4.4	37.85	0.9600	33.39	0.9158	32.09	0.8985	31.68	0.9243	37.89	0.9758	34.04	0.9337
'1024-64'	1273	226	8.5	37.94	0.9603	33.50	0.9171	32.09	0.8984	31.77	0.9249	38.18	0.9762	34.17	0.9341
'512-128'	1165	226	7.3	37.89	0.9602	33.48	0.9170	32.10	0.8985	31.79	0.9249	38.02	0.9760	34.12	0.9340
'1024-128'	1511	237	14.4	37.96	0.9603	33.51	0.9168	32.14	0.8991	31.91	0.9266	38.18	0.9763	34.22	0.9348
'512-256'	1481	233	13.3	37.95	0.9604	33.52	0.9168	32.14	0.8991	31.92	0.9264	38.18	0.9763	34.23	0.9348

Note: In the first column, symbol G indicates the number of filters in each layer of the basic dense block, $G_1 - G_2$ stands for the number of filters of the proposed method in layers before concatenate layers, and the concatenate layers, respectively. Other columns represent the memory usage, average test time for the B100 data set, number of network parameters, average PSNR/SSIM for each test data set, and average PSNR/SSIM for all five test data sets, respectively.

$$\mathrm{H}_d = \sigma\left(\boldsymbol{W}_d^2 \sigma\left(\boldsymbol{W}_d^1 \boldsymbol{F}_{d-1}\right)\right), \tag{4}$$

where $\boldsymbol{W}_d^i, i = 1, 2$ stands for the weights of the i-th convolutional layer in level d, σ denotes the ReLU activation function [21], and d is the index of the level. The size of $\boldsymbol{W}_d^i$ is $3 \times 3 \times n_i$ in which n_1 is much larger than n_2.

Each level gets the concatenation of feature-maps of all preceding levels as input:

$$\boldsymbol{F}_d = [\boldsymbol{F}_0, \boldsymbol{F}_1, \ldots, \boldsymbol{F}_{d-1}], \tag{5}$$

where $[\cdot, \cdot]$ refers to the concatenation of feature-maps.

The output of the proposed method, F_D, is fed into a 1×1 convolutional layer, namely feature fusion layer, to control the output information and adaptively fuse multi-level feature-maps. A 3×3 convolutional layer is used to extract features for residual learning. The final multi-level feature-maps after residual learning formulate as:

$$\boldsymbol{F} = \boldsymbol{F}_{-1} + \boldsymbol{W}^{L+2} \boldsymbol{W}^{L+1} \boldsymbol{F}_D \tag{6}$$

where W^{L+1} and W^{L+2} represent the weights of 1×1 and 3×3 convolutional layers, respectively. Up-scaling is done on these multi-level feature-maps using ESPCNN [22], followed by a convolutional layer outputting the I_{SR}.

3.2 | Representative dense feature learning

In the basic dense block shown in Figure 1, the network's discriminative ability increases by using a larger growth rate. However, the larger growth rate is associated with huge memory usage due to the accumulation of concatenating feature-maps stored in memory. Therefore, the memory problem does not allow the growth rate to be increased very much in these networks.

Increasing the growth rate also produces irrelevant feature-maps that do not affect the network's discriminative ability but increase its GPU memory usage. Therefore, a new dense method is proposed that determines the network's discriminative capability using two hyper-parameters. In other words, the concatenating feature-maps are learnt in two consecutive

TABLE 3 Results for $L = 16$

Model	Memory (MiB)	Time (ms)	Parameters (million)	Set5		Set14		B100		Urban100		Manga109		Average	
'16'	1031	309	0.7	37.60	0.9592	33.19	0.9144	31.94	0.8968	31.16	0.9183	37.54	0.9748	33.70	0.9310
'32'	1313	303	1.7	37.79	0.9597	33.32	0.9155	32.04	0.8981	31.48	0.9222	37.76	0.9755	33.91	0.9329
'64'	1901	305	5.3	37.85	0.9603	33.47	0.9167	32.10	0.8990	31.77	0.9253	38.04	0.9763	34.12	0.9344
'128'	3215	305	19.2	37.93	0.9605	33.47	0.9169	32.14	0.8989	31.96	0.9267	38.18	0.9767	34.24	0.9349
'256'	6493	761	73.6	38.00	0.9603	33.63	0.9179	32.20	0.8997	32.19	0.9288	38.58	0.9769	34.47	0.9359
'64-16'	835	291	0.9	37.74	0.9596	33.30	0.9151	32.00	0.8973	31.32	0.9202	37.83	0.9755	33.87	0.9320
'128-16'	877	290	1.5	37.72	0.9596	33.27	0.9151	32.01	0.8977	31.44	0.9216	37.75	0.9752	33.88	0.9324
'256-16'	981	290	2.7	37.85	0.9599	33.40	0.9158	32.08	0.8982	31.72	0.9242	38.01	0.9759	34.09	0.9336
'512-16'	1185	290	5.3	37.89	0.9601	33.44	0.9166	32.12	0.8989	31.84	0.9258	38.13	0.9761	34.18	0.9344
'1024-16'	1593	295	10.3	37.91	0.9603	33.51	0.9169	32.14	0.8995	31.91	0.9262	38.10	0.9763	34.20	0.9348
'128-32'	971	292	2.2	37.85	0.9601	33.41	0.9161	32.09	0.8986	31.65	0.9239	38.12	0.9761	34.10	0.9338
'256-32'	1075	295	4.1	37.92	0.9602	33.45	0.9167	32.11	0.8988	31.79	0.9252	38.28	0.9763	34.21	0.9343
'512-32'	1315	289	7.9	37.96	0.9605	33.48	0.9165	32.12	0.8987	31.90	0.9264	38.17	0.9764	34.21	0.9347
'1024-32'	1793	307	15.6	37.97	0.9605	33.60	0.9172	32.15	0.8994	31.96	0.9268	38.28	0.9765	34.28	0.9351
'256-64'	1289	288	6.8	37.98	0.9604	33.52	0.9173	32.15	0.8993	31.97	0.9269	38.33	0.9766	34.30	0.9351
'512-64'	1583	294	13.2	38.01	0.9606	33.59	0.9179	32.15	0.8991	32.07	0.9282	38.43	0.9768	34.36	0.9355
'1024-64'	2197	401	26.2	38.03	0.9606	33.57	0.9172	32.18	0.8996	32.17	0.9289	38.34	0.9765	34.37	0.9358
'512-128'	2109	324	239	38.06	0.9607	33.62	0.9178	32.19	0.8994	32.16	0.9284	38.50	0.9769	34.43	0.9357
'1024-128'	2951	488	47.5	38.00	0.9607	33.64	0.9184	32.21	0.9002	32.26	0.9298	38.34	0.9767	34.41	0.9363
'512-256'	3193	491	45.2	38.08	0.9608	33.75	0.9193	32.22	0.9002	32.27	0.9299	38.50	0.9769	34.48	0.9365

Note: In the first column, symbol G indicates the number of filters in each layer of the basic dense block, $G_1 - G_2$ stands for the number of filters of the proposed method in layers before concatenate layers, and the concatenate layers, respectively. Other columns represent the memory usage, average test time for the B100 data set, number of network parameters, average PSNR/SSIM for each test data set, and average PSNR/SSIM for all five test data sets, respectively.

layers. In the proposed method shown in Figure 2, the richer feature-maps can also be learnt with a wider layer before the concatenating layer. The concise and representative concatenating feature-maps are then extracted from these features using a thin concatenating layer. As a result, in the proposed dense block memory usage efficiently decreases, as discussed in the following subsection. This is while the proposed method does not reduce the discriminative capability of dense CNNs because it keeps the propagating feature-maps as representative as before.

3.3 | Discussion

By assuming the number of convolutional layers to be even, if the odd and even layers of the proposed dense block produce G_1 and G_2 feature-maps, respectively, then the input to the l-th layer (to be odd) has $G_0 + G_2 \times (l/2)$ channels, which is almost half of the basic dense block $G_0 + G \times (l - 1)$ by setting $G = G_2$. Therefore, with the same depth and growth rate, the proposed block is expected to have a lower memory requirement and shorter test time than the basic dense block.

With the help of the wider layer used before the concatenating layer, the proposed method learns discriminative feature-maps. Therefore, the growth rate can be reduced (G_2) compared to the growth rate of the basic dense block (G) without loss of information. That produces more representative concatenating feature-maps and can more reduce the GPU memory usage and test time.

4 | EXPERIMENTS

The basic dense block is used instead of the proposed dense block in the network architecture to compare the results. These models are trained with different numbers of convolutional layers. The results are reported in Tables 1–4, and are discussed below.

4.1 | Data sets and metrics

The DIVerse 2K resolution high-quality image data set (DIV2K) contains 800 training images, 100 validation images, and 100 test

TABLE 4 Results for $L = 32$

Model	Memory (MiB	Time (ms)	Parameters (million)	Set5		Set14		B100		Urban100		Manga109		Average	
'16'	1911	436	1.7	37.80	0.9597	33.34	0.9151	32.05	0.8978	31.56	0.9224	37.99	0.9757	34.02	0.9329
'32'	2929	440	5.5	37.93	0.9603	33.50	0.9171	32.13	0.8992	31.88	0.9262	38.20	0.9764	34.22	0.9348
'64'	5121	507	19.8	37.98	0.9605	33.58	0.9168	32.18	0.8997	32.10	0.9282	38.35	0.9766	34.35	0.9356
'128'	10,065	911	76.0	38.04	0.9607	33.66	0.9182	32.21	0.8999	32.18	0.9289	38.49	0.9768	34.44	0.9360
'256'	Out of memory														
'64-16'	1167	416	2.1	37.81	0.9599	33.37	0.9161	32.06	0.8983	31.66	0.9238	37.98	0.9758	34.05	0.9335
'128-16'	1277	419	3.9	37.92	0.9603	33.47	0.9166	32.13	0.8991	31.84	0.9258	38.23	0.9766	34.21	0.9347
'256-16'	1501	418	7.6	37.95	0.9604	33.51	0.9170	32.16	0.8993	32.00	0.9273	38.40	0.9767	34.33	0.9352
'512-16'	1929	427	15.0	37.97	0.9604	33.57	0.9175	32.19	0.8999	32.07	0.9286	38.27	0.9766	34.32	0.9358
'1024-16'	2831	539	29.8	38.02	0.9606	33.61	0.9173	32.20	0.9000	32.20	0.9294	38.38	0.9768	34.40	0.9361
'128-32'	1597	423	6.5	37.97	0.9603	33.50	0.9162	32.14	0.8987	31.96	0.9266	38.41	0.9767	34.32	0.9348
'256-32'	1861	417	12.7	38.00	0.9605	33.56	0.9172	32.19	0.8997	32.12	0.9287	38.35	0.9766	34.36	0.9358
'512-32'	2387	429	25.0	38.02	0.9605	33.62	0.9177	32.20	0.8998	32.24	0.9293	38.53	0.9770	34.47	0.9361
'1024-32'	3513	701	49.8	38.07	0.9606	33.68	0.9185	32.20	0.8997	32.21	0.9295	38.51	0.9769	34.45	0.9362
'256-64'	2585	418	22.7	37.81	0.9605	33.52	0.9177	32.18	0.8997	32.13	0.9290	38.09	0.9767	34.27	0.9359
'512-64'	3337	547	45.1	38.06	0.9607	33.73	0.9187	32.23	0.9002	32.31	0.9300	38.62	0.9770	34.53	0.9365
'1024-64'	4869	941	90.0	38.05	0.9607	33.69	0.9184	32.23	0.9004	32.33	0.9303	38.54	0.9769	34.51	0.9366
'512-128'	5205	828	85.3	38.09	0.9610	33.79	0.9189	32.25	0.9004	32.39	0.9312	38.68	0.9772	34.58	0.9370
'1024-128'	7561	1394	170.2	38.07	0.9608	33.69	0.9187	32.22	0.9001	32.35	0.9306	38.74	0.9773	34.58	0.9368
'512-256'	8955	1534	165.7	38.07	0.9608	33.82	0.9201	32.24	0.9003	32.39	0.9308	38.57	0.9770	34.55	0.9369

Note: In the first column, symbol G indicates the number of filters in each layer of the basic dense block, $G_1 - G_2$ stands for the number of filters of the proposed method in layers before concatenate layers, and the concatenate layers, respectively. Other columns represent the memory usage, average test time for the B100 data set, number of network parameters, average PSNR/SSIM for each test data set, and average PSNR/SSIM for all five test data sets, respectively.

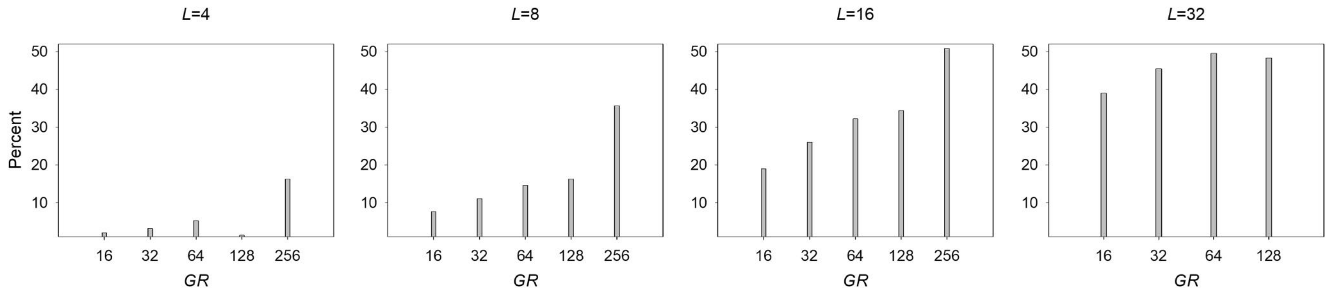

FIGURE 4 The percentage memory usage reduction of the proposed method compared to the basic method, while the growth rate (GR) is the same for both methods. L represents the number of convolutional layers

images [23]. The DIV2K data set is used for training and validation. Only five validation images are used in experiments to reduce training time. Set5 [24], Set14 [25], B100 [26], Urban100 [27], and Manga109 [28] are used as the five standard test data sets.

The HR images are degraded by the bicubic downscaling (using 'imresize' function of MATLAB) with a scale factor of 2 to form the LR images. Peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) [29] metrics are calculated on the Y channel of transformed images in YCbCr space, in both validation and test steps.

4.2 | Training setup

In each training batch, 16 LR RGB patches of size 48×48 are randomly cropped as inputs. These patches are randomly augmented by flipping horizontally or vertically and rotating 90°. Each input patch to the network is subtracted with the mean RGB value of the DIV2K data set. This mean value is added back to the output of the network. The learning rate is initialised to 10^{-4} for all layers. The network is implemented with the Torch7 framework. Adam optimiser [30] is used by $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. An epoch contains 1000 iterations of back-propagation. The results are reported after 200 epochs of training. Two NVIDIA GTX 1080Ti GPUs are used for training, validation, and testing.

TABLE 5 The percentage of memory usage reduction

Selected models	Memory improvement
'64-16' versus '16'	17%
'128-32' versus '32'	21%
'256-64' versus '64'	25%
'512-128' versus '128'	25%
'512-256' versus '256'	34%

TABLE 6 Results for L = 16

Model	Memory (MiB)	Time (ms)	PSNR/SSIM	
'32'	1313	303	33.91	0.9329
'64'	1901	305	34.12	0.9344
'128'	3215	305	34.24	0.9349
'256'	6493	761	34.47	0.9359
'64-16'	835	291	33.87	0.9320
'128-32'	971	292	34.10	0.9338
'256-32'	1075	295	34.21	0.9343
'512-128'	2109	324	34.43	0.9357

4.3 | Network parameters

All convolutional kernels have a size of 3×3, except the feature fusion layer whose kernel size is 1×1. Zero-padding is used in all 3×3 convolutional layers because using a kernel size of 3×3 reduces the feature-map size. The efficiency of zero-padding has been shown by Kim et al. [8].

The number of filters in the first and second convolutional layers, feature fusion layer, and the next coming 3×3 convolutional layer is 64. Three filters are used in the last convolutional layer to output a colour image.

4.4 | Results

The proposed dense method in Figure 3 is replaced with basic dense method 1 to compare the results. The symbols 'G' and '$G_1 - G_2$' are used in the first column of tables to present the names of the basic method and the proposed method, respectively. G stands for the number of filters in each layer of the basic dense block. G_1 and G_2 represent the number of filters in odd and even layers of the proposed block, respectively. In almost all experiments, the value of G_1 is larger/equal to four times the value of G_2. Larger G_1 boosts the results.

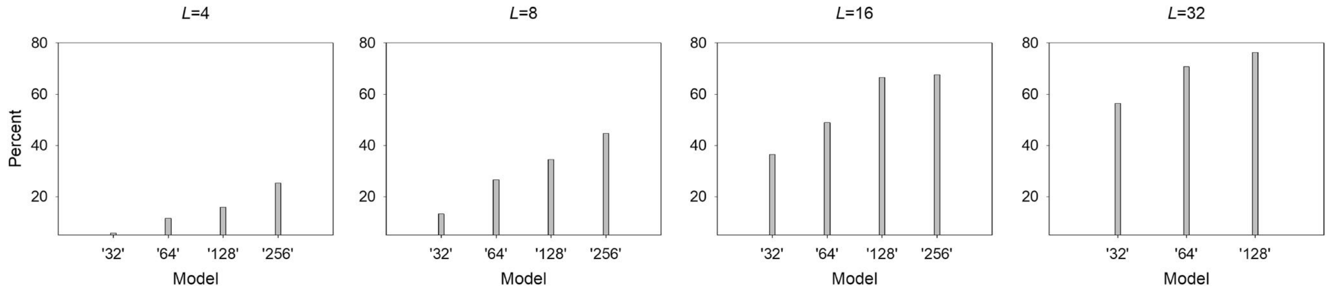

FIGURE 5 The percentage of memory usage reduction of the proposed method compared to the basic method. The horizontal axes show the basic models. The vertical axes represent the percentage of memory improvement achieved by the proposed method with similar PSNR and lower growth rate. L represents the number of convolutional layers

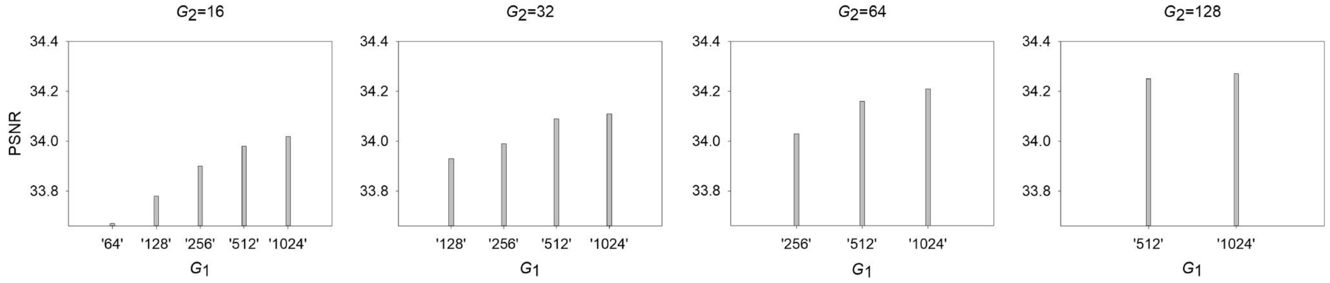

FIGURE 6 PSNR values of the proposed method with different values of filters in its first layer (G_1) at any fixed growth rate (G_2)

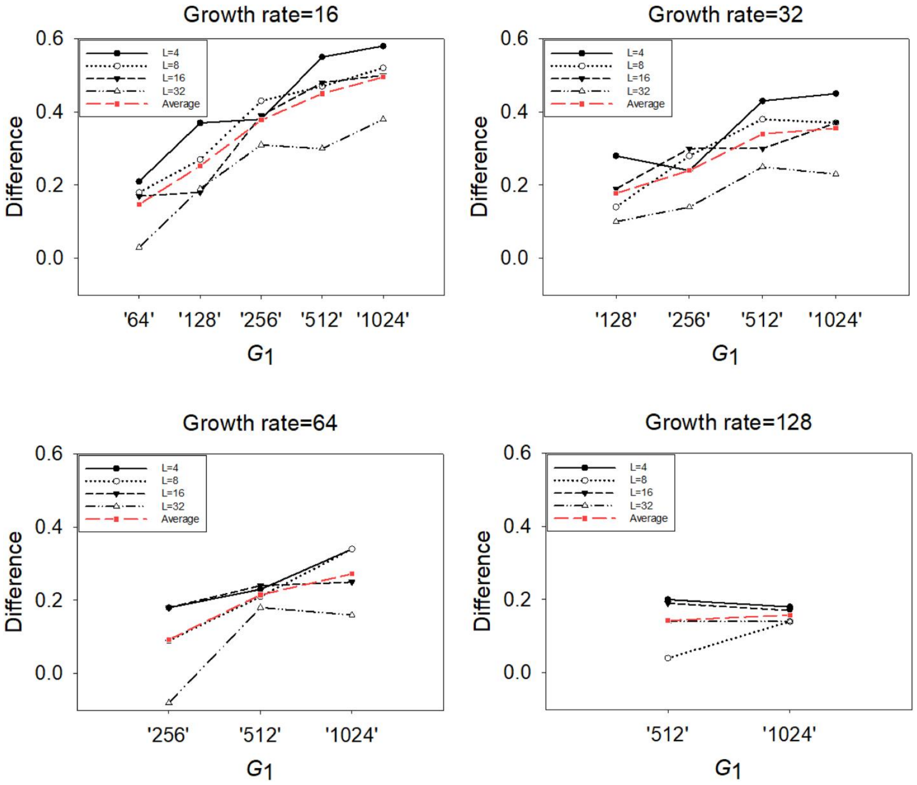

FIGURE 7 PSNR improvement of the proposed method compared to the basic method. The improvement is shown for different growth rates (G_2 in the proposed method and G in the basic dense method). Improvements for each depth (L = 4, 8, 16, 32) are shown with a plot. The average plot for all depths is depicted in red

TABLE 7 PSNR improvement

Selected models	PSNR improvement
'512-16' versus '16'	0.45
'512-32' versus '32'	0.34
'512-64' versus '64'	0.22
'512-128' versus '128'	0.14
'512-256' versus '256'	0.06

4.4.1 | Memory/time investigation

As formulated in Subsection 3.3, the proposed method is expected to reduce GPU memory usage and test time, while the growth rate is the same for both methods. Comprehensive experimental results justify this hypothesis. By setting the growth rate to be the same for both the basic and proposed dense methods, the models are selected based on their PSNR/SSIM values to be competitive with the basic dense method. The percentage of memory usage reduction of the proposed method compared to the basic method is calculated at each depth and depicted in Figure 4. Furthermore, the average value of four depths for selected models is reported in Table 5. The overall average of all models is 24%. By the same calculations, the proposed method is 6% time-efficient.

As discussed in Subsection 3.3, memory usage is directly proportional to the growth rate, and the proposed method can decrease the growth rate while achieving competitive results, that is considering the PSNR/SSIM values at depth 16, models '32', '64', '128', and '256' are comparable with '64-16', '128-32', '256-32', and '512-128', respectively, reported in Table 6.

At each growth rate for the basic dense method, the percentage of memory improvement achieved by the proposed dense method is illustrated in Figure 5. The horizontal axes show the growth rate of the basic connectivity pattern. The vertical axes represent the percentage of memory improvement achieved by the proposed method with similar PSNR and reduced growth rate. The average memory improvement of all models at all depths is 40%. The proposed method improves the test time by 12% with similar calculations.

4.4.2 | Investigation of the number of filters

The value of G_2 is assumed to be constant for investigating G_1. Each sub-figure in Figure 6 shows the average PSNR values of the four depths in each value of G_2. The larger G_1 results in better SISR performance at any fixed value for G_2. This is conceivable because larger G_1 enriches concatenating feature-maps. Results converge at G_1 = 512. Therefore, the '512-X' models have been selected to compare with the 'X' models.

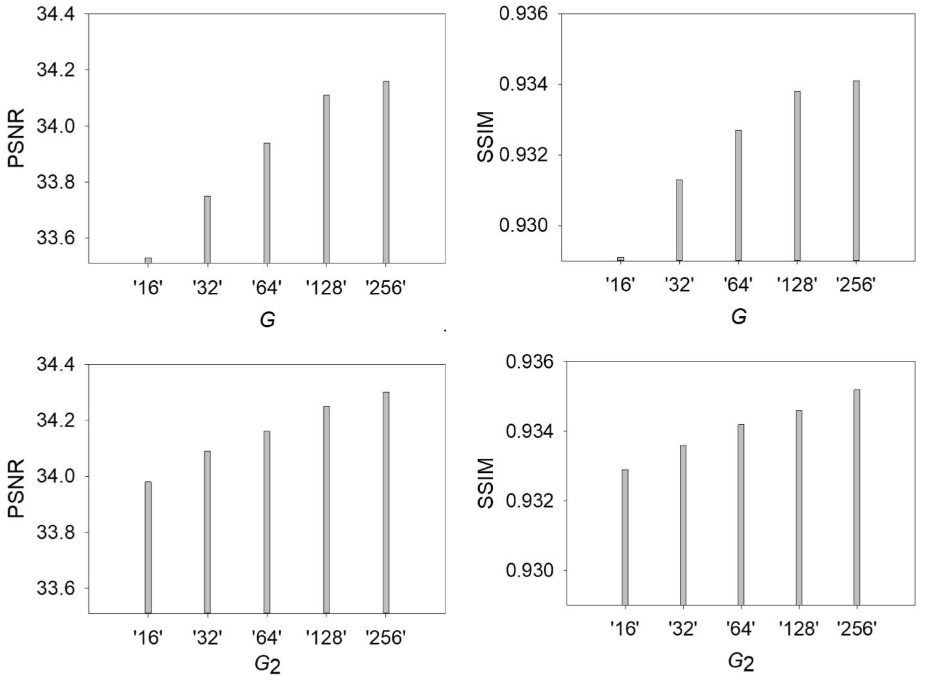

FIGURE 8 PSNR/SSIM in different growth rate values (G and G_2). For the proposed method the G_1 is fixed to 512. Increasing the growth rate improves the results in both methods

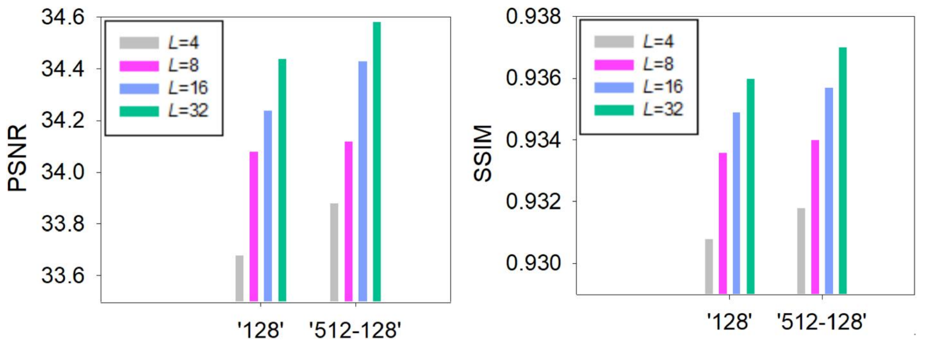

FIGURE 9 PSNR/SSIM with different values of convolutional layers (L) in models '128' and '512-128'

PSNR improvement of the proposed method compared to the basic method is shown in Figure 7. The improvement is shown for different growth rates (G_2 in the proposed method and G in the basic dense method), and different depths ($L = 4$, 8, 16, 32). The proposed method has a larger PSNR than the basic method pattern at almost all growth rates and for different values of G_1 and L.

PSNR improvement of '512-X' models compared to X models, averaged at all four depths, is reported in Table 7. On average, for all values of X, an improvement of 0.24 dB is obtained.

For investigating the growth rate (G in the basic dense block and G_2 in proposed method), G_1 is fixed to 512 in the proposed method. PSNR values, averaged at four depths, for different growth rates are depicted in Figure 8. From this figure it can be inferred that increasing the growth rate improves the results in both methods.

4.4.3 | The effect of the number of layers

Increasing the number of convolutional layers improves the PSNR/SSIM values in both methods. An sample is shown in Figure 9 for models '128' and '512-128'.

4.4.4 | Visual results

A visual comparison is shown in Figure 10 for models '128' and '512-128', and images 'img047' and 'img52' from

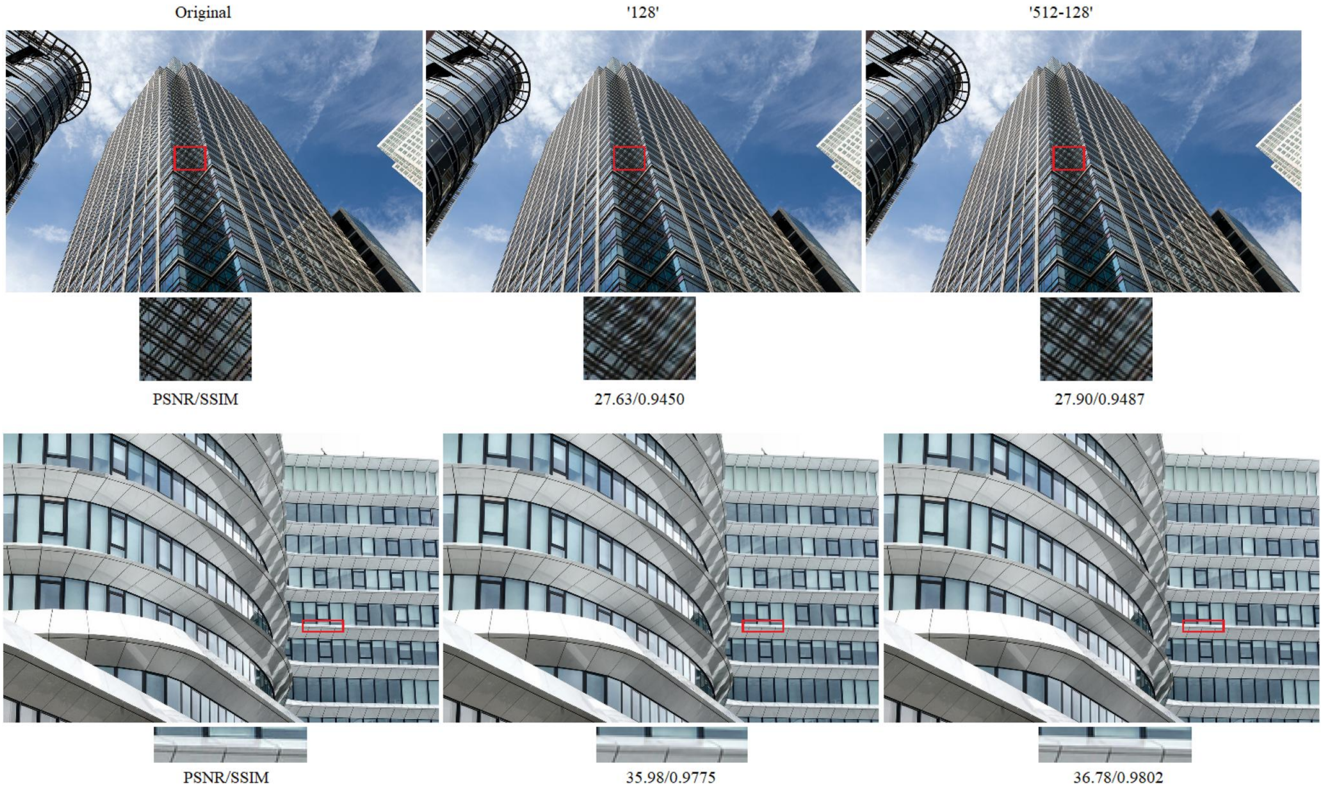

FIGURE 10 Visual results for images 'img047′ and 'img052′ from Urban100

TABLE 8 PSNR/SSIM results in scale factor of ×2

Model	Set5		Set14		B100		Urban100		Manga109		Average	
MARDN	37.99	0.9606	33.57	0.9178	32.19	0.8997	32.10	0.9285	38.31	0.9767	34.34	0.9358
'256-32'	38.02	0.9606	33.63	0.9177	32.21	0.9003	32.26	0.9300	38.50	0.9769	34.46	0.9365
RDN	38.12	0.9610	33.58	0.9185	32.25	0.9007	32.38	0.9313	38.71	0.9773	34.58	0.9372
'512-128'	38.16	0.9612	33.81	0.9202	32.30	0.9011	32.62	0.9330	38.90	0.9776	34.74	0.9380

Urban100. These models are trained with a scale factor of ×2 and $L = 32$ convolutional layers. The basic dense block produces noticeable artefacts and blurred edges. In contrast, the proposed method can recover sharper and clear edges.

4.4.5 | Comparison with the state-of-the-art

Dense blocks of two recent dense CNNs are replaced with the proposed dense blocks [14,16] to compare with the state-of-the-art. All networks are trained with 200 epochs, and the results are reported in Table 8. The blocks of RDN [14] have depth 8 and growth rate 64, which are replaced with '512-128' of depth 8. The blocks of MARDN [16] have depth 4 and growth rate 32, replaced with '256-32' of depth 4. The proposed method improves the PSNR/SSIM values in both RDN and MARDN.

5 | CONCLUSION

A novel dense block is proposed, producing more representative concatenating feature-maps. It uses a convolutional layer with more filters before concatenating layers. The proposed method keeps the discriminative ability of dense CNNs, while it reduces the GPU memory usage significantly. It improves the PSNR of the basic dense CNN by 0.24, recovers sharper and clear edges, and reduces memory consumption and test time by 24% and 6%, respectively. It decreases the need for a larger growth rate. Therefore, it achieves 40% and 12% less memory consumption and test time than the basic dense method. The highest improvements are obtained on the very challenging Urban100 data set. These results justify the limitation of basic dense CNNs, relying only on the growth rate value to achieve better hierarchical features.

ORCID

Kamal Nasrollahi https://orcid.org/0000-0002-1953-0429

REFERENCES

1. Nasrollahi, K., Moeslund, T.B.: Super-resolution: a comprehensive survey. Mach. Vis. Appl. 25, 1423–1468 (2014)
2. Wang, Z., Chen, J., Hoi, S.C.: Deep learning for image super-resolution: a survey. IEEE Trans. Pattern. Anal. Mach. Intell. (2020)
3. Dong, C., et al.: Image super-resolution using deep convolutional networks. IEEE Trans. Pattern. Anal. Mach. Intell. 38, 295–307 (2015)
4. Shabaninia, E., Naghsh-Nilchi, A.R., Kasaei, S.: High-order Markov random field for single depth image super-resolution. IET Comput. Vis. 11, 683–690 (2017)
5. Abbasi, A., et al.: Optical coherence tomography retinal image reconstruction via nonlocal weighted sparse representation. J Biomed. Optic. 23 (2018), 036011
6. Yang, J., et al.: Image super-resolution via sparse representation. IEEE Trans. Image Process. 19, 2861–2873 (2010)
7. Kim, J., Kwon Lee, J., Mu Lee, K.: Deeply-recursive convolutional network for image super-resolution. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, Nevada, pp. 1637–1645 (2016)
8. Kim, J., Kwon Lee, J., Mu Lee, K.: Accurate image super-resolution using very deep convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, Nevada, pp. 1646–1654 (2016)
9. Tai, Y., Yang, J., Liu, X.: Image super-resolution via deep recursive residual network. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Honolulu, Hawaii, pp. 3147–3155 (2017)
10. Ledig, C., et al: Photo-realistic single image super-resolution using a generative adversarial network. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Honolulu, Hawaii, pp. 4681–4690 (2017)
11. Lim, B. et al.: Enhanced deep residual networks for single image super-resolution. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Honolulu, Hawaii, pp. 136–144 (2017)
12. Tong, T. et al.: Image super-resolution using dense skip connections. In: Proceedings of the IEEE international conference on computer vision, Venice, Italy, pp. 4799–4807 (2017)
13. Tai, Y., et al.: Memnet: a persistent memory network for image restoration. In: Proceedings of the IEEE international conference on computer vision, Venice, Italy, pp. 4539–4547 (2017)
14. Zhang, Y., et al.: Residual dense network for image super-resolution. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Salt Lake City, Utah, pp. 2472–2481 (2018)
15. Shamsolmoali, P., et al: Image super resolution by dilated dense progressive network. Image Vis. Comput. 88, 9–18 (2019)
16. Qin, J., et al.: Multi-Resolution space-attended residual dense network for single image super-resolution. IEEE Access. 8, 40499–40511 (2020)
17. Anwar, S., Barnes, N.: Densely residual laplacian super-resolution. IEEE Trans. Pattern. Anal. Mach. Intell. (2020)
18. Dai, T., et al.: Second-order attention network for single image super-resolution. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 11065–11074 (2019). Long Beach, California
19. Huang, G., et al.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Venice, Italy, pp. 4700–4708 (2017)
20. Bermingham, M.L., et al: Application of high-dimensional feature selection: evaluation for genomic prediction in man. Sci Rep. 5 (2015), 10312
21. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics, Fort Lauderdale, Florida, pp. 315–323 (2011)
22. Shi, W., et al: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, Nevada, pp. 1874–1883 (2016)
23. Agustsson, E., Timofte, R.: Ntire 2017 challenge on single image super-resolution: dataset and study. In: Proceedings of the IEEE conference on computer vision and pattern recognition Workshops, Honolulu, Hawaii, pp. 126–135 (2017)
24. Bevilacqua, M., et al.: Low-complexity single-image super-resolution based on nonnegative neighbour embedding (2012)
25. Zeyde, R., Elad, M., Protter, M.: On single image scale-up using sparse-representations. In: International conference on curves and surfaces, Avignon, France, pp. 711–730 (2010)
26. Martin, D., et al.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proceedings Eighth IEEE international conference on computer vision, Canada, Vancouver, pp. 416–423 (2001)
27. Huang, J., Singh, A., Ahuja, N.: Single image super-resolution from transformed self-exemplars. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Massachusetts, Boston, pp. 5197–5206 (2015)
28. Matsui, Y., et al: Sketch-based manga retrieval using manga109 dataset. Multimed. Tool Appl. 76, 21811–21838 (2017)
29. Wang, Z., et al.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. 13, 600–612 (2004)
30. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization (2014). arXiv preprint arXiv:1412.6980

How to cite this article: Imanpour N, Naghsh-Nilchi AR, Monadjemi A, Karshenas H, Nasrollahi K, Moeslund TB. Memory- and time-efficient dense network for single-image super-resolution. *IET Signal Process*. 2021;15:141–152. https://doi.org/10.1049/sil2.12020