# UDenseNet: A Universal Dense Convolutional Network for Image Recognition

Liang Wang[1](✉), Changshuang Zhao[1], Ling Shao[2], and Yihong Wu[3]

[1] Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China
wangliang@bjut.edu.cn
[2] Inception Institute of Artificial Intelligence, Abu Dhabi, UAE
[3] National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100080, China

**Abstract.** Densely connected convolutional networks (DenseNet) have reached unprecedented parameter-performance efficiencies and alleviated problems of vanishing gradients by concatenating each layer to every other layer. However, with the increase in network depth, the cross-channel interaction of dense blocks has become increasingly complex. Hence, it is now more difficult to optimize networks. Moreover, the way combining features in DenseNet restricts its flexibility and scalability in learning more expressive combination strategies. In this study, we aim to answer the question of how to simultaneously ensure the benefits of feature reuse, reduce the complexity of cross-channel interactions, and increase the flexibility of the network. Hence, the components of DenseNet are refined and then used as a basis to develop a universal densely connected convolutional network (UDenseNet). Based on the proposed architecture, the impact of different component configurations on the network performance is empirically analyzed to determine the optimal architectural configuration. Extensive experiments are conducted to validate the proposed UDenseNet on benchmark datasets (CIFAR, SVHN and ImageNet). Results show that, compared to most other methods, the proposed UDenseNet can significantly improve performance in image recognition tasks.

**Keywords:** Deep learning · Neural networks · Dense connection · Image recognition

## 1 Introduction

Deep convolutional networks have achieved promising results in many visual recognition tasks [1–3], with significantly improved feature representations. High-level semantic information is known to play a crucial role in this. One simple and intuitive way to extract more high-level information is to increase the network depth. Residual network (ResNets) [4] trains a much deeper network architecture by leveraging the concept of

This work was partially supported by the National Natural Science Foundation of China under grant numbers of 61772050 and 61836015.


Y. Peng et al. (Eds.): PRCV 2020, LNCS 12307, pp. 209–221, 2020.
https://doi.org/10.1007/978-3-030-60636-7_18

shortcut connections. However, when the depth exceeds 1000 layers, the performance of ResNets reaches saturation owing to information loss caused by the simple summation-based feature aggregation. This occurs because the information carried by prior feature maps becomes corrupted or washed out by later features.

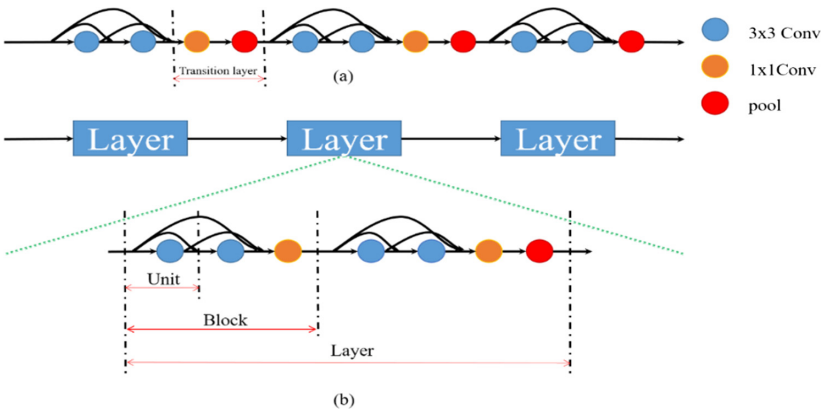


**Fig. 1.** Comparison of network architecture of (a) DenseNet and (b) the proposed UDenseNet.

To resolve this problem faced by ResNets, the densely connected convolutional network (DenseNet) [5] was proposed. Dense connections allow each convolutional layer to connect to every other layer in a feed-forward way. Then, a series of concatenations is used to integrate semantic information from different layers into one entity as the final input. The better parameter-performance efficiency of DenseNet over ResNets may be due to aggregation style, which combines features via direct concatenation, while preserving their original form. However, concatenation increases the complexity of cross-channel interactions and the difficulty of optimization, respectively. Moreover, the way DenseNet combines features restricts its flexibility and scalability to learn more expressive combination strategies.

In this study, we revisit DenseNet and propose a universal densely connected convolutional network (UDenseNet) to extend network flexibility and versatility, while preserving the efficiency of feature reuse. As shown in Fig. 1, unlike DenseNet, the proposed UDenseNet consists of three main components: *unit*, *block*, and *layer*. Hierarchically, a *unit* is not only the most fundamental part of the network but also the minimal entity of dense connections. A *block* consists of a series of units followed by a $1 \times 1$ convolution, which can produce cross-channel interactions. Finally, a *layer* comprises several layers of *blocks*, followed by a $2 \times 2$ average pooling layer, which fulfils the task of down-sampling. The entire network resembles a building made of Lego bricks; it is built step-by-step from the most fundamental *unit*. The feature maps produced by each *block* have the same size as the input. Additionally, the neuron of a *layer* encodes higher-level semantics by processing larger receptive fields with the pooling layers. The three components perform distinct roles and combine for desirable outcomes. The difference between the proposed UDenseNet and DenseNet is shown in Fig. 1. The proposed UDenseNet (Fig. 1(b)) is an extension of DenseNet (Fig. 1(a)), which is a combination

of multiple *units*, *blocks*, and *layers*. This hierarchy significantly expands the flexibility of the network's construction. It is worth noting that, unlike DenseNet, the last layer of UDenseNet has a $1 \times 1$ Conv. The proposed UDenseNet is evaluated on several benchmark datasets (CIFAR [6], SVHN [7], and ImageNet [8]) for image recognition and significantly outperforms the state-of-the-art methods on most.

The main contributions of this paper can be summarized as follows:

- A universal densely connected convolutional network is proposed, which can reasonably extend network flexibility and versatility, while preserving efficient feature reuse.
- The proposed UDenseNet architecture enhances the performance of dense connections, enabling it to achieve high performance and dramatically improve accuracy and effectiveness in comparison with existing deep convolutional networks.
- To the best of our knowledge, this is the first time that an optimal architectural configuration is provided for a densely connected convolutional network. This was achieved by fixing the total number of network layers and performing extensive experiments with varying number of network components.

The remainder of this paper is organized as follows. Section 2 reviews related works. In Sect. 3, the proposed model and its components are illustrated in more detail. Extensive experiments are reported in Sect. 4 and Sect. 5 concludes the paper.

## 2 Related Works

Since AlexNet [1] won the 2012 ImageNet Challenge [8], deep convolutional networks have become an important machine learning method in computer vision. With the development of hardware and the emergence of large-scale datasets, it is now possible to train very deep neural networks. In the last few years, the number of network layers has gradually increased. From the initial LeNet5 [9] of only 5 layers, to the later VGG [10] and Inception [11] of dozens of layers as well as the more recent highway networks [12], ResNets [4] and DenseNet [5] which have hundreds of layers, the performance of deep convolutional networks has steadily and dramatically improved for many image recognition tasks. This reveals that network depth is crucial. However, it does not mean that learning better networks is as easy as simply adding more layers. One reason for this is the notorious problems of vanishing/exploding gradients [13], which hinders convergence from the start. Furthermore, with increasing network depth, difficulties with feature reuse have been highlighted, thereby resulting in accuracy saturation. Many recent publications have addressed related issues [13, 14]. In the following paragraphs, we review several works that are closely related to this present study from micro and macro perspectives.

To alleviate the problems of deep convolutional networks, earlier works adopted better initialization schemes. For example, [13] proposed that a properly scaled uniform distribution should be adopted for initialization. Unfortunately, this initialization scheme is only valid for linear activations, whereas ReLU and PReLU are nonlinear. Because of this dilemma, [14] proposed a better, more theoretically sound initialization by considering ReLU/PReLU, and this enabled very deep models to converge where [13] method

failed. A recently proposed, notable contribution for training extremely deep networks is BN [15], which can standardize each mini-batch hidden layer's mean and variance. It not only decreases the vanishing gradients problem, but also yields a strong regularizing effect. However, dataset distributions are generally uneven, especially for a mini-batch of data taken from a dataset. In these situations, batch normalization fails to adapt to some data.

On the other hand, some works have focused on the macro design of the network structure. Most of them improved information flow during forward and backward propagation by introducing extra skip connections. Typical works include highway networks [12], ResNets [4], and many other residual networks [16–18]. Highway networks [12] introduce gating units to regulate the information flow of the networks, thereby allowing information to propagate unimpeded on information highways across several layers. Unlike highway networks, ResNets [4] combine features through direct summation defined by identity shortcut connections before features are passed into a layer. This is performed instead of the gated summation defined by shortcut connections with the gating function. This simplification significantly improves training efficiency, simplicity, and scalability. Instead of focusing on the depth, wide residual networks [17] focus on the width of the network. [18] analyses the success of highway networks and ResNets, which may owe to the dropout mechanism applied on network layers during training. This makes it possible that the resulting networks are short during training but deep during tests. In summary, these findings enable information flow to bypass some dropped layers via skip connections. Afterward, DenseNet [5] was proposed by building shorter paths from early layers to later layers, which concatenates all layers together in a feed-forward way. This has allowed DenseNet to attain high parameter-performance efficiency. Since its original construction, several extensions of DenseNet have been proposed [19, 20]. The notion of local dense connectivity was introduced in [19], which suggests that less connectivity can achieve more efficient feature reuse and higher accuracy in dense architectures, thereby allowing for an increased growth rate at a fixed network capacity. The closest study to ours is [20], which explores a key architectural aspect of densely connected convolutional networks, dense feature aggregation, including the summing scheme of ResNets and concatenating scheme of DenseNet. However, it mainly focused on the skip layer connections instead of the entire architecture of densely connected convolutional networks.

In this paper, we separate the fully-dense connectivity of dense block by introducing $1 \times 1$ Conv and redefine the components of densely connected convolutional network. A universal densely connected convolutional network is proposed. Based on this, the impact on network performance of different component configurations is empirically analyzed to determine the optimal architectural configuration. To the best of our knowledge, this is the first study that provides an optimal configuration for the densely connected convolutional network architecture, when the total number of layers, i.e., the depth, is fixed.

## 3 Universal Densely Connected Convolutional Network

In DenseNet, each layer that is densely connected by concatenation receives the feature maps of all previous layers. In contrast to common connection methods, dense connections ensure maximum information flow between network layers, which is beneficial for information transmission in the network. Consequently, the feature maps of the $i^{th}$ layer can be represented by the following formula:

$$x_i = H_i\left(\left[x_0, x_1, \cdots, x_{i-1}\right]\right) \tag{1}$$

where $x_j (0 \leq j \leq i)$ is the output of the $j^{th}$ layer, $[\cdot]$ refers to the concatenation of the feature maps, and $H_i(\cdot)$ denotes the non-linear transformation function corresponding to the $i^{th}$ layer. In [5], DenseNet consists of sequentially stacked dense blocks, as shown in Fig. 1(a). A detailed description is as follows. DenseNet is divided into three dense blocks, each of which has an equal number of layers. There is a transition layer between dense blocks to reduce the number of accumulated channels per concatenation while one $2 \times 2$ average pooling operation is used for down-sampling to increase the size of the network's receptive field.

In the remaining parts of this section, we introduce the network architecture of UDenseNet. A schematic illustration is shown in Fig. 1(b) to facilitate an understanding of the network architecture. Each component of the network is now described.

### 3.1 Unit

A *unit* refers to the minimal dense connection in the network, which connects the input with its output by concatenation to ensure the benefits of feature reuse (see Fig. 1(b)). Following DenseNet [5], two types of *units* are used in UDenseNet:

**Basic** - BN and ReLU are followed by one $3 \times 3$ convolution: BN-ReLU-$3 \times 3$ Conv, with which the network is denoted as UDenseNet.

**Bottleneck** - one $1 \times 1$ convolution is added before the $3 \times 3$ convolution of the **Basic** *unit* to reduce the feature dimension: BN-ReLU-$1 \times 1$ Conv-BN-ReLU-$3 \times 3$ Conv, with which the networks is denoted as UDenseNet-B.

As in [5], the $3 \times 3$ convolution kernel is adopted here without considering other convolutional kernels with different sizes.

### 3.2 Block

Skip connections allow useful features to be transferred from shallower layers to deeper layers, where each layer is directly supervised by the final output layer. DenseNet implements skip connections by concatenating outputs of all previous layers. However, a potential drawback is that aggregation by concatenation results in the isolation between feature maps. Therefore, to learn complex cross channel interactions, an additional convolutional operation with a kernel size of $1 \times 1$ is added before down-sampling in transition layers. Thus, the $1 \times 1$ Conv is a necessary component of a *block*. In an ablation study, we performed extensive experiments to show the effect of $1 \times 1$ Conv, as described in Sect. 4.2. In UDenseNet, the $1 \times 1$ Conv in the transition layer in the

original DenseNet was moved to the end of the *block*. Then, we redefined the concept of the *block*, that is, a group of dense connections ending with one $1 \times 1$ Conv form the new *block*. Note that the $1 \times 1$ Conv is connected in a traditional way rather than as a dense connection. Further, a *unit* factor $u$, which is the number of *units* (including both **Basic** and **Bottleneck**) in each *block*, was also introduced.

### 3.3 Layer

Inspired by residual networks, in which multiple residual blocks process feature maps of the same scale, we took advantage of a few dense *blocks* for this. The down-sampling layer is essential for changing the size of feature maps in the network. As in [5], the pooling operation was also used. Here, we introduced the concept of a *layer*, which consists of several serially connected *blocks* followed by one $2 \times 2$ pooling layer (see Fig. 1(b)). Additionally, two factors, *block* factor $b$ and *layer* factor $l$, were also introduced, where $b$ is the number of *blocks* in each *layer* and $l$ is the number of *layers* in UDenseNet. From the perspective of UDenseNet, the original densely connected convolutional network (i.e., 40-layer DenseNet [5]) has 3 *layers*, where each *layer* has 1 *block*, and each *block* has 12 *units*, but the last *layer* has no $1 \times 1$ Conv.

### 3.4 The General Expression and Structure

Various components of the densely connected convolutional network are refined above. In this section, we will present a concrete mathematical expression that indicates the relationship between the total number of network layers and the number of components. Additionally, to express the network architecture, we present a description to clarify the specific general structure of the network.

A list is used to express an N-layer densely connected network, as follows:

$$[\mathrm{N}(u_1, b_1)(u_2, b_2) \cdots (u_l, b_l)] \tag{2}$$

where $l$ refers to the number of *layers* in the network. Each *layer* is in the form of parentheses with two elements, where the first element, i.e., $u_i (1 \leqslant i \leqslant l)$, is the number of *units* in each *block*, and the second element, i.e., $b_j (1 \leqslant j \leqslant l)$, is the number of *blocks* within each *layer*. The relationship between the total number of network *layers* and the number of *units* and *blocks* can be represented by the following formula:

$$N = \sum_{i=1}^{l} (\alpha u_i + 1) \times b_i + 2 \tag{3}$$

where the coefficient $\alpha = 1$ for UDenseNet and $\alpha = 2$ for UDenseNet-B.

Several variants of the network structure can be obtained for different $l$, $b$, and $u$, respectively. The above analysis shows that the architecture of UDenseNet more flexible for learning more expressive merge strategies. For example, a UDenseNet version of original DenseNet [5], which is a 40-layer DenseNet, can be described as follows:

$$[40(5, 1)(7, 2)(7, 2)] \tag{4}$$

From Eq. (3), it can be seen that Eq. (4) is not the sole UDenseNet architecture corresponding to the original 40-layer DenseNet. For example, [40(4, 2)(6, 2)(6, 2)] and [40(9, 1)(6, 2)(6, 2)] are also corresponding UDenseNet architectures. This shows that UDenseNet has more flexibility than DenseNet.

Table 1 illustrates the general structure of UDenseNet. It consists of an initial convolution Conv; 3 sequential layers L1, L2, and L3, each of which is followed by average pooling, which is not shown in this table; and the final classification layer, which is also omitted for clarity. In the example shown, the network uses a **Basic** *unit*. As in [5], we also introduced the compression factor $r$ to scale the channel of the $1 \times 1$ Conv in the last *block* of one *layer*, and the growth rate $k$ to describe the number of feature maps (or channels) added by per dense connection. A UDenseNet with compression factor $r$ is called UDenseNet-C, and in the following experiments, it is set to $r = 0.5$. When both the bottleneck and compression factor $r$ are used, the corresponding UDenseNet is called UDenseNet-BC. We analyzed the impacts of the *unit*, *block*, and *layer* on the performance of dense connections through experiments. Several modifications of the densely connected network architecture are further reported and compared in the following section of experiments.

**Table 1** Structure of universal densely connected convolutional networks. The number of *units* in each *block* is denoted by factor $u_i$; the parameters $k$ and $r$ are the growth rate and the compression ratio of the network, respectively; and groups of convolutions are shown in brackets, where $b_i$ is the number of *blocks* in each *layer*.

| Group | Input size | Output size | Layer type |
|---|---|---|---|
| Conv | $32 \times 32$ | $32 \times 32$ | $(3 \times 3, 2k)$ |
| L1 | $32 \times 32$ | $16 \times 16$ | $\begin{pmatrix} (3 \times 3, k) \times u_1 \\ 1 \times 1, r \end{pmatrix} \times b_1$ |
| L2 | $16 \times 16$ | $8 \times 8$ | $\begin{pmatrix} (3 \times 3, k) \times u_2 \\ 1 \times 1, r \end{pmatrix} \times b_2$ |
| L3 | $8 \times 8$ | $1 \times 1$ | $\begin{pmatrix} (3 \times 3, k) \times u_3 \\ 1 \times 1, r \end{pmatrix} \times b_3$ |

## 4 Experiments

### 4.1 Implementation Details

We empirically validated UDenseNet on a series of benchmark datasets (including CIFAR [6], and SVHN [7]) for image recognition tasks. Comparisons with the state-of-the-art methods were also performed. The main purpose of Subsect. 4.2 is to explore the internal structure of the dense convolutional network, i.e., the relationship between *unit*,

*block*, and *layer*, and its impact on network performance, rather than to focus on higher precision. In contrast, Subsect. 4.3 focuses on the high precision. In the experiments, the DenseNet [5] was taken as the reference network. For all datasets, we compared the results of the reference network with those of the proposed UDenseNet.

Similar to [5], for the CIFAR datasets [6], the training and test sets have 50,000 and 10,000 images, respectively, and 5,000 training images were taken as a validation set. A standard data augmentation scheme (mirroring/shifting) was adopted, and a "+" mark after the dataset name denotes this data augmentation in the following tables. For the Street View House Numbers (SVHN) dataset [7], the training set contains 73,257 images and additional 531,131 images, and 6,000 images were used as a validation set. The test set contains 26,302 images. No data augmentation was adopted for this dataset. Each UDenseNet has three dense layers. For the CIFAR dataset, networks did not adopt the dropout, whereas the dropout rate is 0.2 for the SVHN. Further, we also verified the proposed UDenseNet on the ImageNet2012 [8] dataset and compared it with the DenseNet. In the experiments on ImageNet, we used a 121-layer UDenseNet-BC structure. And UDenseNet-BC has four dense layers.

As in [5], all networks utilize stochastic gradient descent (SGD) with a weight decay of $10^{-4}$ as a learning optimizer and adopt the parameter initialization introduced by [14]. For the CIFAR and SVHN datasets, the batch size of the network is set to 64 and is trained for 4 and 5 epochs, respectively. The initial learning rate is set to 0.1, and is divided by 10 at 50% and 70% of the total number of training epochs. For the ImageNet, model is trained for 90 epochs with a batch size of 256. The initial learning rate of the model is 0.1 and every 30 rounds divided by 10 until 90 epochs. Our implementations were performed in Pytorch [21].

### 4.2 Trade-Off Between Unit and Block

First, experiments were performed to explore the relationship between unit factor $u$ and block factor $b$ and its impact on network performance. The comparison was performed among networks of equal depths, thus we established networks with different $u$ and $b$ to ensure that the total number of convolutional layers were equal. This means, for instance, that $u$ should increase while b decreases. Results are shown in Table 2, where the total number of convolutional layers of UDenseNet, 38, is fixed, and several networks with different unit factors $u$ and block factors $b$ were trained and tested. Further, results of a UDenseNet with the same depth as the original 40-layer DenseNet and a UDenseNet with similar layer type (12, 1)(12, 1)(12, 1) as the original DenseNet are also shown in Table 2. It can be seen that both significantly outperformed the original DenseNet. Notably, the difference in network architectures makes that UDenseNet and DenseNet unable to have the same depth and similar layer type simultaneously. Considering that UDenseNet outperformed the corresponding DenseNet with the same depth, 38-layer UDenseNet were used here for perform analysis.

As seen in Table 2, [38(5, 2)(5, 2)(5, 2)] and [38(3, 3)(3, 3)(3, 3)] turn out to be the better, whereas [38(11, 1)(11, 1)(11, 1)] and [38(16)(1, 6)(1, 6)] have the poorer performances. This is probably due to the increased difficulty in optimization as a result of the increased number of dense connections in [38(11, 1)(11, 1)(11, 1)]. Similarly, it is speculated that feature extraction may not be sufficient in [38(1, 6)(1, 6)(1, 6)]. Thus,

when the depth of the network is fixed, too many or too few units in each block result in poor network performance. This means that the optimal architecture of UDenseNet can be obtained when the unit factor $u$ and block factor $b$ reach the best trade-off. Further, we also discuss the impact of the last layer, with and without the $1 \times 1$ convolution, on network performance. Table 2 shows that [41(12, 1)(12, 1)(12, 1)] outperformed the reference network. On CIFAR-10, the error dropped from 5.24% to 5.13%. Following a similar trend, the error dropped from 24.42% to 23.23% on CIFAR-100. Notably, the performance of the proposed UDenseNet with 38 layers exceeded that of the reference network everywhere except for [38(1, 6)(1, 6)(1, 6)]. The number of network layers, meanwhile, was smaller, which increased the compactness of the densely connected network. From the point of view of the number of FLOPs, it is also consistent with the above conclusion. For networks listed from top to bottom in Table 2, the number of FLOPs is 0.29, 0.29, 0.30, 0.18, 0.26, 0.24, 0.22, 0.20, and 0.15, respectively. It can be seen that UDenseNet can improve the network performance with similar FLOPs and parameters in comparison with DenseNet.

Similar to the case of the 40-layer UDenseNet stated in Sect. 3.4, there are some other 38-layer UDenseNet. They have similar performances with the ones reported. The relationship between $u$ and $b$ and its impact on network performance of them follow the trend summarized above, too. For clarity and space limitation, we do not list them here.

**Table 2** Test error on CIFAR-10+ and CIFAR-100+ of dense convolutional network with varying u and b. The overall best results are in blue, and the best results of the networks with the same depth are in bold.

| Network | Depth | Params | FLOPs ($10^9$) | CIFAR-10+ | CIFAR-100+ |
|---|---|---|---|---|---|
| DenseNet [5] | 40 | 1.0M | 0.29 | 5.24 | 24.42 |
| Our implementation | 40 | 1.0M | 0.29 | 5.38 | 24.47 |
| UDenseNet [41(12, 1)(12, 1)(12, 1)] | 41 | 1.2M | 0.30 | 5.13 | 23.23 |
| UDenseNet [40(5, 1)(7, 2)(7, 2)] | 40 | 1.1M | 0.18 | 4.74 | **22.88** |
| UDenseNet [38(11, 1)(11, 1)(11, 1)] | 38 | 1.0M | 0.26 | 5.18 | 23.64 |
| UDenseNet [38(5, 2)(5, 2)(5, 2)] | 38 | 1.0M | 0.24 | 4.84 | 22.83 |
| UDenseNet [38(3, 3)(3, 3)(3, 3)] | 38 | 0.9M | 0.22 | **4.83** | 22.61 |
| UDenseNet [38(2, 4)(2, 4)(2, 4)] | 38 | 0.8M | 0.20 | 5.07 | 23.20 |
| UDenseNet [38(1, 6)(1, 6)(1, 6)] | 38 | 0.6M | 0.15 | 5.93 | 25.80 |

### 4.3 Performance Evaluation

Experiments were also performed to evaluate the UDenseNet architecture. To make comparisons, the DenseNet and other state-of-the-art methods were also applied on

benchmark datasets. Experiments were performed on datasets of CIFAR-10, CIFAR-100 and SVHN. The top1 error rate was used to evaluate the performance. Experimental results are shown in Table 3, which shows that UDenseNet-BC [250(11, 4) (13, 3)(12, 3)] with depth = 250 and k = 24 outperformed the existing DenseNet and other state-of-the-art methods on all CIFAR datasets. Its error rates of 3.56% on CIFAR-10+ and 17.48% on CIFAR-100+ are significantly lower than the error rates achieved by the variations of ResNets. Notably, the performance of UDenseNet is improved in comparison with the corresponding DenseNet variant structure.

**Table 3** Comparisons of test error (%) of UDenseNet and existing methods. The overall best results are in blue. The symbol+ in the dataset name indicates standard data augmentation (translation and/or mirroring).

| Network | Depth | Params | CIFAR-10+ | CIFAR-100+ | SVHN |
|---|---|---|---|---|---|
| Network in network [22] | – | – | 8.81 | – | 2.35 |
| All-CNN [23] | – | – | 7.25 | 33.71 | – |
| Highway network [12] | – | – | 7.72 | 32.39 | – |
| Scalable BO [24] | – | – | 6.37 | 27.40 | 1.77 |
| FractalNets [25] | 21 | 38.6M | 5.22 | 23.30 | 2.01 |
| Wide ResNets [17] | 16 | 11.0M | 4.81 | 22.07 | – |
| ResNets [4] | 110 | 1.7M | 6.41 | 27.22 | 2.01 |
| ResNets (pre-activation) [16] | 164 | 1.7M | 5.46 | 24.33 | – |
| ResNets with stochastic depth [18] | 110 | 1.7M | 5.23 | 24.58 | 1.75 |
| DenseNet [5] | 40 | 1.0M | 5.24 | 24.42 | 1.79 |
| DenseNet [5] | 100 | 7.0M | 4.10 | 20.20 | **1.67** |
| DenseNet-BC [5] | 100 | 0.8M | 4.51 | 22.27 | 1.76 |
| DenseNet-BC (k = 24) [5] | 250 | 15.3M | 3.62 | 17.60 | 1.74 |
| UDenseNet-BC [40(5, 1)(7, 2)(7, 2)] | 40 | 1.1M | 4.74 | 22.88 | 1.80 |
| UDenseNet-BC [100(12, 2)(8, 4)(8, 4)] | 100 | 10.25M | 3.83 | 19.17 | 1.74 |
| UDenseNet-BC [100(7, 2)(8, 2)(8, 2)] | 100 | 0.9M | 4.40 | 21.31 | 1.88 |
| UDenseNet-BC [250(11, 4)(13, 3)(12, 3)] | 250 | 21.99M | **3.56** | **17.48** | 1.71 |

For example, the DenseNet with depth = 100 and k = 12 obtained a competitive error rate of 4.10% on the test dataset of CIFAR-10+. Meanwhile, the test error of the UDenseNet [100(12, 2)(8, 4)(8, 4)] with depth = 100 and k = 12 reached 3.83% on CIFAR-10+, which is a relative improvement of 6.59% with respect to the aforementioned DenseNet. On CIFAR-100+, we observed a similar trend. However, the UDenseNet performed poorer than the DenseNet on SVHN. This was partly because the content of the SVHN was comparatively simple, thus overfitting to the training

dataset might have occured. In summary, the proposed UDenseNet surpassed most state-of-the-art methods on CIFAR-10, CIFAR-100, and SVHN.

**Table 4.** The top1 and top5 errors (%) of UDenseNet and DenseNet on ImageNet validation set, with single-crop testing. All the networks have the same growth rate k = 32 and depth 121. The overall best results are in blue.

| Network | Params | FLOPs ($10^9$) | top1 | top5 |
|---|---|---|---|---|
| DenseNet-BC | 7.98M | 0.57 | 25.02 | 7.71 |
| UDenseNet [121(12, 1)(7, 2)(7, 2)(8, 2)] | 7.99M | 0.50 | **24.69** | **7.30** |
| UDenseNet [121(7, 2)(7, 2)(5, 3)(6, 2)] | 8.29M | 0.61 | 24.75 | 7.38 |
| UDenseNet [121(5, 3)(7, 2)(7, 2)(6, 2)] | 8.01M | 0.68 | 24.80 | 7.36 |

To show efficiency on a larger-scale dataset, the different configurations of UDenseNet were tested and compared with DenseNet on ImageNet [8]. All models had the same preprocessing methods and hyperparameters during training. The validation error for ImageNet2012 is reported in Table 4. These results show that the better accuracy exhibited by UDenseNet over DenseNet extended to ImageNet. UDenseNets have similar numbers of parameters to those of state-of-the-art DenseNets, while obtain significantly higher precision. For example, UDenseNet-BC (k = 32) [121(12, 1) (7, 2)(7, 2)(8, 2)] (7.99 M params) yielded lower validation error to DenseNet-BC (k = 32) (7.98 M params). Table 4 shows it can be seen that different network structures can be constructed by controlling $u$ and $b$ when the network depth remains unchanged, which can also reflects the flexibility of the UDenseNet.

## 5 Conclusions

In this study, DenseNet was revisited, and components of densely connected convolutional networks were refined to propose a universal dense convolutional network, UDenseNet. The proposed UDenseNet guarantees the benefits of feature reuse while reducing the complexity of cross-channel interactions and increasing network flexibility. Additionally, it can also significantly improve the performance of the densely connected convolutional networks. Based on the proposed network architecture, the impact of different component configurations on network performance was analyzed empirically to investigate the optimal architecture configuration of the densely connected convolutional networks. Extensive experiments proved the feasibility of the proposed UDenseNet. Moreover, the proposed UDenseNet also outperforms most state-of-the-art methods on the datasets of CIFAR-10, CIFAR-100, SVHN, and ImageNet. We believe that these results will help the further study of densely connected convolutional networks. In a future study, we plan to focus on the optimal configuration of densely connected convolutional networks in theory and apply the proposed UDenseNet for various purposes.

## References


1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Proceedings of NIPS 2012, Lake Tahoe, NV, USA, pp. 1106–1114 (2012)
2. Hu, H., Ma, B., Shen, J., Sun, H., Shao, L., Porikli, F.: Robust object tracking using manifold regularized convolutional neural networks. IEEE Trans. Multimedia **21**(2), 510–521 (2019)
3. Hsu, Y.C., Lv, Z., Schlosser, J., Odom, P., Kira, Z.: Multiclass classification without multi-class labels. In: Proceedings of ICLR 2019, New Orleans, LA, USA, pp. 1–16 (2019)
4. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of CVPR 2016, Seattle, WA, USA, pp. 770–778 (2016)
5. Huang, G., Liu, Z., Weinberger, K.Q., Van Der Maaten, L.: Densely connected convolutional networks. In: Proceedings of CVPR 2017, Honolulu, HI, USA, pp. 2261–2269 (2017)
6. Krizhevsky, A.: Learning multiple layers of features from tiny images. Technical report, Department of Computer Science, University of Toronto (2009)
7. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.: Reading digits in natural images with unsupervised feature learning. In: NIPS Workshop on Deep Learning and Unsupervised Feature Learning (2011)
8. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Li, F.F.: ImageNet: a large-scale hierarchical image database. In: Proceedings of CVPR 2009, Miami, FL, USA, pp. 248–255 (2009)
9. Sermanet, P., Chintala, S., LeCun, Y.: Convolutional neural networks applied to house numbers digit classification. In: Proceedings of ICPR 2012, Tsukuba, Japan, pp. 3288–3291 (2012)
10. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large scale image recognition. In: Proceedings ICLR 2015, Vancouver, BC, Canada, pp. 1–14 (2015)
11. Szegedy, C., et al.: Going deeper with convolutions. In: Proceedings of CVPR 2015, Boston, MA, USA, pp. 1–9 (2015)
12. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks. arXiv:1505.00387 (2015)
13. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feed forward neural networks. In: Proceedings of International Conference on Artificial Intelligence and Statistics, PMLR, vol. 9, pp. 249–256 (2010)
14. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings ICCV 2015, Santiago, Chile, pp. 1026–1034 (2015)
15. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Proceedings of ICML 2015, Lille, France, pp. 448–456 (2015)
16. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 630–645. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_38
17. Zagoruyko, S., Komodakis, N.: Wide residual networks. In: Proceedings of BMVC 2016, York, UK, pp. 87.1–87.12 (2016)
18. Huang, G., Sun, Yu., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 646–661. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_39
19. Hess, A.: Exploring feature reuse in DenseNet architectures. arXiv:1086.01935v1 (2018)
20. Zhu, L., Deng, R., Maire, M., Deng, Z., Mori, G., Tan, P.: Sparsely aggregated convolutional networks. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11216, pp. 192–208. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01258-8_12
21. Paszke, A., et al.: Automatic differentiation in PyTorch. In: AutoDiff Workshop of NIPS 2017, Long Beach, CA, USA (2017)

22. Lin, M., Chen, Q., Yan, S.: Network in network. In: Proceedings of ICLR 2014, Banff, Canada (2014)
23. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: the all convolutional net. In: Proceedings of ICLR 2015, San Diego, CA, USA (2015)
24. Shoek, J., et al.: Scalable Bayesian optimization using deep neural networks. In: Proceedings of ICML 2015, Lille, France, pp. 2171–2180 (2015)
25. Larsson, G., Maire, M., Shakhnarovich, G.: FractalNet: ultra-deep neural networks without residuals. In: Proceedings of ICLR 2017, Toulon, France (2017)