Fast DenseNet: Towards Efficient and Accurate Text Recognition with Fast Dense Networks

Zhao Zhang 1,2, Zemin Tang 2, Yang Wang 1, Zheng Zhang 3, Shuicheng Yan 4, Meng Wang 1

1 Hefei University of Technology, 2 Soochow University, 3 Harbin Institute of Technology
4 National University of Singapore

Abstract

Convolutional Recurrent Neural Network (CRNN) is a popular network for recognizing texts in images. Advances like the variants of CRNN, such as Dense Convolutional Network with Connectionist Temporal Classification, has reduced the running time of the networks, but exposing the inner computation cost of the convolutional networks as a bottleneck. Specifically, DenseNet based frameworks use the dense blocks as the core module, but the inner features are combined in the form of concatenation in dense blocks. As a result, the number of channels of combined features delivered as the input of the layers close to the output and the relevant computational cost grows rapidly with the dense blocks getting deeper. This will severely bring heavy computational cost and restrict the depth of dense blocks.

In this paper, we propose an efficient convolutional block called Fast Dense Block (FDB). To reduce the computing cost, we redefine and design the way of combining internal features of dense blocks. FDB is a convolutional block similarly as the dense block, but it applies both sum and concatenating operations to connect the inner features in blocks, which can reduce the computation cost to \((1/L, 2/L)\), compared with the original dense block, where \(L\) is the number of layers in the dense block. Importantly, since the parameters of standard dense block and our new FDB keep consistent except the way of combining features, and their inputs and outputs have the same size and same number of channels, so FDB can be easily used to replace the original dense block in any DenseNet based framework. Based on the designed FDBs, we further propose a fast network of DenseNet to improve the text recognition performance in images. The recognition results on a large-scale Chinese string and MNIST datasets show that our model can obtain more accurate result with higher efficiency, compared with other related popular text recognition networks.

1. Introduction

Texts and images are two of the most popular vision data in the area of computer vision. It is common in practice that texts are always embedded in images, so how to detect and recognize the texts or characters in images accurately by a learning algorithm is still challenging and an important topic in the field of visual pattern recognition [36, 37], such as the optical character recognition (OCR) [6]. OCR is a long-standing topic but still a very challenging task due to complicated background and complex contents in images [2]. Recent years have witness the fast development and continuous breakthroughs in computer vision [12-14] and deep learning [7-11][46][52], lots of advanced end-to-end deep learning methods have been proposed [24].

For OCR, two crucial sub-tasks are text line extraction and text line recognition. The first task is to extract the regions of texts in images and the second one recognizes the textual contents of the identified region. To handle the OCR, there are two mainstream frameworks at present. The first one is to train an end-to-end network that jointly solves the tasks of text line extraction and recognition, such as arbitrary orientation network (AON) [2]. The other popular one is a two-stage scheme, i.e., training two networks for two sub-tasks, for instance Convolutional Recurrent Neural Network (CRNN) [29]. Generally, the unified models have stronger adaptability and faster speed, but the results are slightly lower. The two-stage models usually have higher
Technically, we propose a new block termed Fast Dense Block (FDB) and then derive a fast convolutional neural network based on the designed FDBs. Specifically, we redefine and redesign the way of combining the internal features over the original dense block to enable a more efficient block. Compared with dense block, FDB uses both sum and concatenating operations to connect inner features in each block, which can reduce the computing cost to \( \frac{1}{L}N \times H \times W \times C \), where \( L \) is the number of layers in dense block. The parameters of original dense block and FDB are also the same except for the way of combining features, and their inputs and outputs have the same size and same number of channels. Thus, FDB is applicable to any networks that the dense block can be used. Figure 1(d) illustrates the layout of our fast dense block, where the convolution layer includes the functions of Batch Normalization (BN) [22], ReLU and Convolution.

- We mainly evaluate the capability of the designed FDBs for improving the text recognition performance and efficiency. Specifically, we propose a fast text recognition network, called Fast DenseNet with Up-sampling block (shortly, FDenseNet-U). For FDenseNet-U, we define an up-sampling block using several designed FDBs and deconvolution operation for up-sampling the features. To avoid important feature information loss and make the related parameters of FDenseNet-U learnable, we use the convolution operation with stride 2 to replace the pooling operation. In order to improve the efficiency of extracting dense features, we apply the depth-wise separable convolution to replace the original convolution.

- We conduct the text recognition simulations on Chinese string and handwritten digits, which demonstrates that our model can obtain enhanced performance. Moreover, the process of recognizing the texts in image is efficient.

2. Related Work

2.1. Depth-wise Separable Convolution

The depth separable convolution, which is a new variant of the original convolution, is first presented in MobileNet [17]. Different from the original convolution that considers the channel and region changes jointly, the deep separable convolution clearly separates the channel and region and divides the convolution operation into two sub-steps, that is, depth-wise and point-wise processes.

The depth-wise process divides the input features with the form of \( N \times H \times W \times C \) into \( C \) groups and then performs convolution operation to each group, where \( N \) is the number of features, \( H \) and \( W \) are the height and width of features, and \( C \) is the number of channels of features. The depth-wise process collects spatial features of each channel, namely, depth-wise features. The point-wise mainly refers to a process that conducts \( 1 \times 1 \) convolution operation using \( k \) filters to the output features from the depth-wise process, which collects the features from each point, i.e., point-wise features. Figure 2 shows the depth separable convolution in the case of padding “same”, where \( K \) is convolution kernel.

2.2. Convolutional Recurrent Neural Network

CRNN is an end to end learning framework recognizing the text sequences in scenes [29]. CRNN takes advantage of the CNNs for the local feature extraction and RNN for the temporal summarization of extracted features. As shown in...
Figure 3. The learning architecture of CRNN for text recognition.

3. Proposed Fast Dense Block (FDB)

We propose the definition and structure of FDB, which is the core of our fast text recognition framework. Since the structure of our FDB is inspired by the residual block [16], dense block [5] and residual dense block [50], we firstly review them and also show their structures in Figure 4 for comparison. Then, we detail the structure of our FDB.

3.1. Reviews of Related Blocks

Residual block. The Residual Network (ResNet) [16] was mainly proposed for addressing the issue of network degradation. The core idea of ResNet is embodied in the design of residual block. Figure 1(a) shows the structure of the standard residual block. Let \( x, F(x) \) and \( H(x) \) represent the input features, output features without and with a short connection respectively, we can obtain

\[
F(x) = w_2(\sigma(w_1x)), \quad H(x) = F(x) + x, \tag{1}
\]

where \( w_1 \) and \( w_2 \) are weights of the convolutional layers, \( \sigma(\bullet) \) is the function of Rectified Liner Units (ReLU) [23]. The identical maps can be constructed in two cases that either \( F(x) = x \) without short connection or \( F(x) = H(x) - x = 0 \) with short connection. The optimization with a short connection is easier in training. The short connection in residual block can also help enhance the feature flow.

Dense block. Different from ResNet, DenseNet mainly creates short paths between layers, and in this process a simple connectivity pattern, referred to as dense block [5], is derived. To ensure maximum information flow between layers, all layers with matching feature-map sizes in a dense block are connected directly with each other [5], and the related features are combined by concatenating them as

\[
X_i = \phi([X_0, X_1, \ldots, X_{i-1}])
\]

where \( X_i \) represents the features of \( i \)-th layer, \( X_0 \) represent the input features and \( \phi(\bullet) \) denotes the according function of convolution layer. To preserve the feed-forward nature, each layer obtains additional inputs from preceding layers and passes on its own feature-maps to subsequent layers [5]. Figure 1(b) illustrates the layout of dense block. DenseNet uses many short connections to enhance feature flow, so it has a strong feature learning ability. But the approach of combining the features by concatenating them will bring about sharp increase in terms of number of feature channels and huge computing efforts with the dense block getting deeper, which will restrict the depth of the networks using dense blocks, for instance DenseNet, directly.

Residual dense block. The residual dense block (RDB) is proposed in the residual dense network (RDN) [50] that aims to make full use of all hierarchical features from the original image. The core idea of RDN is to utilize both residual and dense blocks to improve the feature learning ability by enhancing the information flow, learning residual features and enhancing the local feature fusion. Figure 1(c) illustrates the structure of RDB. It should be noted that RDB utilizes the concatenating operation as dense block to combine the features of the former RDBs and current layers, and use the residual outside the dense block to enhance the representation learning. As such, the proposed RDB will also suffer from huge computing cost as the dense block. But its structural design gives us inspiration.

3.2. Structure of our Fast Dense Block (FDB)

We describe the structure of FDB. The main idea of our FDB uses both the sum operation in residual block and the concatenating operation in dense block. But the difference is that RDB mainly improves the feature learning ability, using the sum operation to construct a residual outside the dense block, while our FDB mainly uses the sum operation to change the feature fusion mode inside the dense block such that the computing cost of dense blocks can be clearly reduced. Specifically, FDB is proposed by redefining and designing the way of combining the internal features of the dense blocks. Generally, the number of channels of input
features is more than that of inner layers in dense block. So, we propose to reduce the computing cost by using the sum operation to combine features instead of concatenating for all the inner layers of the dense blocks, except the input and output layer, which is performed as follows:

\[ X_i = \phi(X_{i-1}) \]

\[ X_j = \phi(X_1 + \ldots + X_{i-1}), \quad i > 1 \]  

(3)

We obtain the output features by concatenating all the features of different layers in FDBs. As such, we can fully utilize feature information of all layers and can also ensure that our feature maps have the same size and same number of channels as that of dense block. As such, our proposed FDB will be applicable to any deep network structures that the original dense block can be used.

### 3.3. Theoretical Computational Analysis of FDB

As claimed above, FDB can reduce the computing cost of the dense block by redefining and designing the way of combining internal features, so we would like to present the theoretical computational analysis. More specifically, we summarize the analysis in Theorem 1.

**Theorem 1.** For each block, our FDB is able to reduce the computing cost to \((1/L, 2/L)\), compared with the original dense block, where \(L\) is the number of layers in block.

**Proof.** Recalling the computation of combining features in the original dense blocks and our FDBs in formulas (2) and (3). For the original dense block, the input features of the \(i\)-th layer are the combined features of its all preceding layers by concatenating. For FDB, the input feature of the first convolution layer is original data and input features of the \(i\)-th layer are combined features of its preceding layers by summing over the \(i\)-th to \((i-1)\)-th layers. Thus, we find that the number of channels of the input features grows in the original dense blocks but keeps unchanged in the middle layers of our proposed FDB. Next, we present the quantitative comparison of the computational cost between the original dense block and our FDB.

Denote by \(M, N, L, H\) the number of channels of input data, the number of kernels in each layer, the number of the convolutional layers in a dense block/our proposed FDB, and the computing cost of each channel of each kernel, respectively. By setting the related parameters of the two blocks to be same, we can obtain the following computing efforts for original dense block and our proposed FDB:

\[
C_{\text{Dense}} = C_1 + C_2 + \ldots + C_i \\
= H \ast \left( N \ast M + N \ast (M + N) + \ldots + N \ast (M + N \ast (L-1)) \right) \\
= H \ast N \ast \left( M \ast L + N \ast L \ast (L-1)/2 \right)
\]

\[
C_{\text{FDB}} = C_1 + C_2 + \ldots + C_i \\
= H \ast \left( N \ast M + N \ast N + \ldots + N \ast N \right), \\
= H \ast N \ast \left( M + N \ast (L-1) \right)
\]

where \(C_{\text{Dense}}, C_{\text{FDB}}\), and \(C_i (i = 1, 2, \ldots)\) denote the computing efforts of the original dense block, FDB and each layer of the two blocks, respectively. Then, we have the following relation between \(C_{\text{Dense}}\) and \(C_{\text{FDB}}\):

\[
\frac{C_{\text{FDB}}}{C_{\text{Dense}}} = \frac{1+1/\vartheta}{L}, \quad \text{where } \vartheta = 1+\frac{2M}{N \ast (L-1)},
\]

We can find when the value of \(M\) is fixed, if the values of \(L\) and \(N\) are very large, the reduced computational effort tends to \(2/L\), else the reduced cost tends to \(1/L\). As such, one can easily conclude that \(C_{\text{FDB}} / C_{\text{Dense}} = (1/L, 2/L)\). Clearly, FDB achieves an obvious computing effort reduction. Specifically, by comparing with the dense block, we can draw the following conclusion that the computation of our FDB can be reduced to nearly \(2/L\) with the increase of \(L\) and \(N\), but the value of \(L\) is usually large at this time; otherwise, the cost reduction is about \(1/L\), but the \(L\) is smaller. This also fully demonstrates the rationality of our proposed FDB. Based on the computational analysis, reducing the cost of calculation has a clear restriction with FDB getting deeper, i.e., the reduced cost cannot be increased indefinitely with the growth of \(L\) and \(N\). Specifically, with the increasing number of the convolution layers inside FDB, the reduced cost is close to the lower limit, namely, \(2/L\). Note that the computational efforts discussed here mainly refer to the computation cost of operating the convolutional kernels on features in the original dense block and our FDB, excluding the calculation cost of concatenation in dense block and the calculation cost of the feature addition after conversion to sum in FDB. The main reasons are twofold. On one hand, the calculation cost of the applied sum and concatenation operations on extracted features is far less than that of the convolution kernels, which is not in one order of magnitude. On the other hand, the calculation cost of the concatenation operator is not easy to measure, so we mainly focus on computing the cost of convolution kernels.
4. Fast DenseNet for Text Recognition

We propose the architecture of the fast DenseNet with an up-sampling block, termed FDenseNet-U. FDenseNet-U includes two parts, i.e., convolution layers and transcription layer. Traditional CNNs usually extend the network depth by stacking more convolutional and down-sampling layers. But the size of stacked features cannot be reduced forever, and the down-sampling layer may also cause some useful feature information loss. To consummate these deficiencies, we use deconvolution to define an up-sampling block based on the original CNNs, similarly as [52], which can help to extend the depth and restore lost feature information to a certain extent. To improve the model efficiency of our FDenseNet-U, we use the depth-wise separable convolution to replace the original convolution operation. Note that all the convolution operations refer to an operation group that includes BN, ReLU and Depth separable convolution in this paper. Besides, we apply the convolutional operation with stride 2 to replace the pooling layer in FDenseNet-U as down-sampling strategy to prevent the feature information being lost and make the parameters of the whole framework learnable at the same time in transition layer. The structure of our FDenseNet-U framework has two parts, i.e., FDB based encoding module and up-sampling block, which will be described shortly. Figure 4 illustrates the convolutional network part of our FDenseNet-U.

**FDB encoding module.** This feature encoding module includes a convolution layer, three FDBs and two transition layers. The first convolution layer, i.e., Conv1, is mainly applied to extract shallow features and also plays a role of down-sampling when encountering with large-size features. It is worth noting that this module mainly operates the convolution and down-sampling on the input images and outputs dense encoding features.

**Up-sampling block.** This up-sampling block optimizes the dense features further from the FDB encoding module and outputs the up-sampling features as the input of the transcription layer. Up-sampling block is the core of our proposed FDenseNet-U, which contains the operations of up-sampling, two fast dense blocks, a transition layer and a convolution layer. The most important part of this block is up-sampling, and a good up-sampling way can effectively restore the lost feature information, which is widely-used in image restoration. In addition, the up-sampling block is helpful to extend the depth of network for delivering deeper features. On the basis of the original image pixels, we use the deconvolution [49] to construct our up-sampling block. Note that for the study on CNNs, the deconvolution usually refers to the inverse process of the convolution operation. Similar to the convolution, deconvolution also involves the multiplication and the addition operation. In addition, the deconvolution can be used to up-sample the CNN features to the resolution of original images.

**Transcription layer.** This transcription layer is mainly used to transform the prediction of each frame into the final label sequence, which includes the operations of soft-max and CTC. The soft-max function is employed to output the predictions of the learned features from convolution parts, and CTC can transform the predictions into the final label sequence. In our FDenseNet-U framework, CTC needs to input data of each column of a picture containing text as a sequence and outputs the corresponding characters.

5. Experimental Results and Analysis

In this section, we evaluate our proposed FDenseNet-U on two text recognition tasks: (1) recognizing the texts in images [38-40]; (2) recognizing the handwritten digits from images. For the first task, we compare the results of our FDenseNet-U with those of several related deep models, where the CPUs and GPUs of all the evaluated methods in experiments are Xeon E3 1230 and 1080 Ti respectively and the applied convolution architectures are based on the framework of Caffe [31]. A large-scale synthesis Chinese String dataset [30] is used for evaluations. For the second task of the handwritten digits recognition, we compare the recognition results with some popular methods on MNIST [15]. It is noteworthy to point out that CTC is required in the first task of text recognition, but it is not required in the second task of handwritten digits recognition. Because the second task only needs to perform the single character recognition, we use the convolution parts of FDenseNet-U to extract deep features and then use the soft-max function [3] as the classifier to predict the labels of samples.

5.1. Handwritten Digits Recognition

We first evaluate each deep framework for recognizing the handwritten digits based on images using the popular MNIST database [15]. Note that MNIST is a widely-used handwritten digit dataset, where the goal is to classify the images with 28x28 pixel as one of the 10 digital class. The MNIST handwritten digit dataset contains 60,000 training samples and 10,000 testing samples. MNIST can evaluate the feature learning ability of a learning algorithm.

**Implementation details.** For MNIST, the batch size is set to 128 and the epoch size is 200 in this experiment. The initial learning rate is set to 0.001, which will be adjusted to 0.0001 at interval between 50 and 100, and to 0.00001 after 100 epochs. Note that we add a fully-connected layer after the last convolutional layer in our FDenseNet-U so that the output features can be transformed into the required form of softmax. To prevent the overfitting, we add four dropout layers after Conv1, FDB3, Conv2 and the fully-connected layer, and set the value to 0.5 in FDenseNet-U. In this study, the performance of FDenseNet-U are compared with those of six popular deep models, including Deep L2-SVM [41],
Max-out Network [42], BinaryConnect [43], PCANet-1 [44], gcForest [45] and Simple CNN with BaikalCMA loss.

**Recognition results.** The handwritten digit recognition results in terms of accuracy on MNIST are described in Table 1. We see that our frameworks achieve the enhanced results, and the accuracies reach 99.68, which implies that the proposed FDBs have strong representation ability.

**Visualization of the training process.** We illustrate the training curves of our framework that is trained by using the cross-entropy loss. The obtained training curves of our FDenseNet-U are shown in Figure 5, where the top figure shows the curve of the cross-entropy loss and the bottom one illustrates the recognition accuracy. From the results, we can find that the cross-entropy loss is well fitted, which implies that our framework has strong abilities for feature representation learning and text recognition.

### 5.2. Text Recognition in Images

In this section, we evaluate each deep framework model for recognizing the texts in images by using the synthetic Chinese string dataset. This string dataset is generated from Chinese corpus, including news and classical Chinese, by changing fonts, sizes, gray levels, blurring, perspective and stretching, which is made by following the procedures in [30]. The dictionary has about 5990 characters, including Chinese, punctuation, English and numbers. Each sample is fixed to 10 characters, and those characters are randomly intercepted from the corpus. The resolution of text pictures is unified into 280×32. A total of about 3 million 600 thousand images are generated, which are divided into a training set and a test set according to 9:1. Figure 6 shows some image examples of the Chinese string dataset.

**Implementation details.** We use the stochastic gradient descent (SGD) [28] for training and take Tensorflow [32] and Keras as our experiment architectures. The training of the deep network is implemented on TITAN Xp. The batch size of FDenseNet-U is set to 32. The epoch size is 10. The initial learning rate is set to 0.001, which will be adjusted at each epoch with algorithm of 0.005*0.4**epoch, where “**” denotes the power calculation. The weight decay is set to 0.0001. In our FDenseNet-U, we add a dropout layer [27] after the Conv2 and set the dropout rate to 0.2 to prevent overfitting. We use the value of test loss as a metric, and the training stops when the loss values do not descend. The weights are kept when the training of each epoch finishes.

**Text recognition results.** We illustrate the recognition result of each model in Table 2, where “Accuracy” refers to the correct proportion of the whole string and statistics on the test set, “Test time” refers to the testing time of single image on GPU. For the compared models and our models, the text recognition results are based on the frameworks of CRNN/DenseNet+CTC. Specifically, the frameworks with suffix “res-blstm” denotes the models with blstm [25] in the form of residuals, the frameworks with suffix “no-blstm” means that there is no LSTM layer. “DenseNet-sum-blstm-full-res-blstm” has two changes over “Densenet-res-blstm” framework: (1) the approach of combining the two lstms into blstm changes from concat to sum; (2) both layers of blstm are connected by the residual way. “DenseNet-no-blstm-vertical-feature” removes the pooling operations [26] of 1x4 to “Densenet-no-blstm” relatively. “DenseNet-UB” denotes the original DenseNet with an un-sampling block [33], where the bilinear interpolation and Deconvolution are used to construct the un-sampling block. “DenseNet” represents the original framework of the “DenseNet-UB” without up-sampling block, which is over-fitting, therefore we cannot give the result of this method. We see that our FDenseNet-U can achieve higher accuracies up to 99.45, compared with other related models. About the testing time consumption, our FDenseNet-U can reduce the recognition time of a single text image to 0.64s. The experimental results once again demonstrate that our proposed fast dense neural network with fast dense blocks can obtain better results greatly reduced computation complexity. The used dataset and results of most compared methods are publicly available at https://github.com/senlinuc/caffe_ocr.

### 5.3. Visualization of Recognized Texts in Images

In addition to the above quantitative evaluation results, we also visualize some recognized texts in images by using our
final performance is also expected to be improved. Further extend the depth of our proposed network and the transpose convolution. Under these circumstances, we can magnify the original input images by up-sampling such as strategy to features with very large size. In fact, we can also resources, we usually need to adopt the down-sampling information. But due to the limit of computing power and the input images with larger sizes that contain more feature theory, we can obtain better results with deeper networks and input images with larger sizes that contain more feature information. As such, we can potentially obtain stronger feature leaning ability.

6.3. Discussion on Up-sampling

To define the up-sampling block in our FDenseNet-U, different up-sampling methods can be applied, for instance traditional interpolation, edge-based interpolation, bilinear interpolation [47], region-based interpolation [48] and deconvolution methods [49]. But choosing an appropriate interpolation algorithm to insert new elements between the pixels is important. By comparing with the deconvolution, other interpolation methods mentioned-above are based on the fixed algorithm to sample the images or features, so the results are different due to different algorithms. But a bad up-sampling method fails to effectively recover the feature information. As such, we choose the deconvolution as our up-sampling method, because it can learn the parameters adaptively in the training process.

6.4. Discussion on Application Areas

By the theoretical analysis and experimental verification, we conclude that our FDB can reduce the computing cost significantly and also obtains more promising performance in the recognition tasks, compared with the original dense blocks. In addition, FDB is applicable to any deep networks that the original dense block can be used. Due to the fact that DenseNet has a very wide range of application areas and the dense blocks have also been used by many existing deep network models, we believe that we can replace the original dense block to construct different FDB based deep network models to deal with different recognition problems, including the other computer vision tasks. Moreover, since our frameworks are indeed full-convolutional networks, we can infer that FDenseNet-U will obtain promising results potentially in the task of object segmentation [34-35].

7. Conclusion and Future Work

In this paper, we have proposed a novel and efficient convolutional block called fast dense block (FDB), which can significantly reduce the computing cost to (1/L, 2/L), compared with the dense blocks of the classical DenseNet. Based on the designed FDBs, we also propose a fast dense neural network to improve the text recognition result in
images. To restore feature information and extend the depth of network, FDenseNet-U constructs an up-sampling block by utilizing several FDBs and deconvolution.

We examined the performance of our FDenseNet-U for both Chinese and handwritten digits recognition. From the investigated cases, enhanced recognition results with high efficiency are obtained by our network, compared with the other related models. Although promising results have been obtained by our models, there are some issues that are still worthy to explore. For example, more effective ways to reduce the computing cost of the dense blocks and retain the important feature information are highly-desired to be studied. How to determine the optimal number of layers in deep learning networks, including our FDenseNet-U, still remains an open issue, which is another research problem. Besides, it is also interesting to employ the proposed fast dense block to the other deep convolutional networks and extend the presented fast FDenseNet-U to the other popular low-level or high-level computer vision tasks.

**References**

[1] A. Graves, S. Fernández, F. Gomez, J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML*, 2006.

[2] Z. Cheng, Z. Xu, F. Bai, Y. Niu, S. Pu, and S. Zhou. Aon: Towards arbitrarily-oriented text recognition. In *Proceedings of IEEE CVPR*, 2018.

[3] K. Duan, S. Keerthi, W. Chu, S. Shevade, and A. Poo. Multi-category Classification by Soft-Max Combination of Binary Classifiers. In *Proceedings of the International Workshop on Multiple Classifier Systems*, 2003.

[4] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao. Detecting text in natural image with connectionist text proposal network. In *Proceedings of ECCV*, 2016.

[5] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely Connected Convolutional Networks. In *Proceedings of IEEE CVPR*, 2017.

[6] H. J. Caulfield, and W. T. Maloney. Improved discrimination in optical character recognition. *Applied Optics*, 8(11):2354-2356, 1969.

[7] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[8] A. Krizhevsky, I. Sutskever, G. Hinton. ImageNet classification with deep convolutional neural networks. In *Proceedings of ICML*, 2012.

[9] K. Simonyan, and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of ICLR*, 2015.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich. Going deeper with convolutions. In *Proceedings of IEEE CVPR*, 2015.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of IEEE CVPR*, 2016.

[12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, and A.C. Berg. Imagenet large scale visual recognition
