Journal of Intelligent & Fuzzy Systems 40 (2021) 5015–5025
DOI:10.3233/JIFS-201758
IOS Press

Additive DenseNet: Dense connections based on simple addition operations

Dawei Yu[a], Jie Yang[b,*], Yun Zhang[b] and Shujuan Yu[b]
[a]*College of Computer Science and Technology, Suzhou College of Information and Technology, Suzhou, China*
[b]*College of Electronic and Optical Engineering, Nanjing University of Posts and Telecommunications, Nanjing, China*

Abstract. The Densely Connected Network (DenseNet) has been widely recognized as a highly competitive architecture in Deep Neural Networks. And its most outstanding property is called Dense Connections, which represent each layer's input by concatenating all the preceding layers' outputs and thus improve the performance by encouraging feature reuse to the extreme. However, it is Dense Connections that cause the challenge of dimension-enlarging, making DenseNet very resource-intensive and low efficiency. In the light of this, inspired by the Residual Network (ResNet), we propose an improved DenseNet named Additive DenseNet, which features replacing concatenation operations (used in Dense Connections) with addition operations (used in ResNet), and in terms of feature reuse, it upgrades addition operations to accumulating operations (namely $\sum(\cdot)$), thus enables each layer's input to be the summation of all the preceding layers' outputs. Consequently, Additive DenseNet can not only preserve the dimension of input from enlarging, but also retain the effect of Dense Connections. In this paper, Additive DenseNet is applied to text classification task. The experimental results reveal that compared to DenseNet, our Additive DenseNet can reduce the model complexity by a large margin, such as GPU memory usage and quantity of parameters. And despite its high resource economy, Additive DenseNet can still outperform DenseNet on 6 text classification datasets in terms of accuracy and show competitive performance for model training.

Keywords: DenseNet, ResNet, deep learning, text classification

1. Introduction

Recently, Deep Neural Networks (DNNs) have achieved unprecedented successes in machine learning [1–3] as they made remarkable breakthroughs in various domains, including but not limited to speech recognition [4], computer vision [5, 6] and natural language processing (NLP) [7, 8]. And it is widely acknowledged that network depth plays an essential role in the great success of DNNs [9, 10]. Substantial evidences indicate that by stacking layers successively, DNNs can naturally learn and integrate multiple levels of abstractions from data [11, 12], and thus be exponentially better than shallow ones at approximating the target function [13] and enriching feature hierarchies [14]. Hence, the exploration of building deeper networks has become a popular research trend in the past few years [15].

However, with the unlimited increase of depth, DNNs suffer a lot from the well-known vanishing gradient problem [16, 17], which results in high training and testing error [18]. Moreover, training DNNs becomes a really challenging task [14, 19, 20]. As a result, boosting DNNs performance by stacking more layers is proven to be unfeasible [9, 15, 25].

Aim at alleviating the vanishing gradient problem, Skip Connections [1, 12, 21, 22] have recently attracted researches' attention. By adding connections (or paths) across layers [23], Skip Connections

*Corresponding author. Jie Yang, College of Electronic and Optical Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210023, China. E-mail: 1017020718@njupt.edu.cn.

ISSN 1064-1246/$35.00 © 2021 – IOS Press. All rights reserved.

allow information (or gradients) to "bypass" several intermediate layers and thus effectively alleviate attenuation. [9, 15, 24]. Encouraged by Skip Connections, a variety of innovations have been proposed, and the Densely Connected Network (DenseNet) [15] as well as Residual Network (ResNet) [25, 26, 28, 29] may be the most successful architectures among them.

The ResNet was initially proposed in [25] and extended in [29]. The core idea of it is building connections between each layer's input and output, which is known as identity mapping [26]. Specially, for each layer in ResNet, identity mapping combines the layer's input and output by summation (or an addition operation [27]). As a result, ResNet enables the feature reuse among different layers. In spite of its simplicity, ResNet can significantly reduce the difficulty of training DNNs [28] and achieve record-breaking performance on many tasks, such as ImageNet and COCO object detection [17, 29].

The success of ResNet laid the foundation for the following DenseNet, whose most distinguishing contribution is introducing a novel connectivity pattern called Dense Connections, which enable the input of each layer to be a concatenation of all the preceding layers' outputs [30]. And in this way, each layer can enjoy a dense feature reuse in contrast to ResNet. In practice, DenseNet significantly boosts the performance of DNNs and shows great superiority over ResNet [31].

Despite its great successes, some problems have arisen in the applications of DenseNet. Recall that quantitatively, there are $L(L+1)/2$ connections instead of L for the L^{th} layer in DenseNet due to Dense Connections [41, 54], and each layer's input has to concatenate the outputs of all preceding layers dimensionally, which means with the increase of depth, the dimension of input will enlarge dramatically. Besides, the later layers are forced to learn spatial and cross-layer features altogether, entailing heavy amount of parameters and computations [30, 32]. As a result, in order to ease the difficulty of training, the quantity of connections in Dense Connections is strictly limited for practical applications [27]. Generally speaking, due to its connectivity pattern, DenseNet suffers excessive model complexity [30, 32], which further leads to many serious issues, such as memory-hungry, time-consuming [33], overfitting [34], and low efficiency on computation and parameters [41].

In the light of this, by taking inspiration from ResNet, we propose an improved version of DenseNet named Additive DenseNet (A. DenseNet), which explicitly features replacing concatenation operations with simple addition operations in terms of achieving feature reuse among layers. Specifically, in order to achieve the dense feature reuse like DenseNet, Additive DenseNet further upgrades addition operations to accumulating operations thus enables each layer's input to be the summation of all preceding layers' outputs, not just the former 2 layers in ResNet. In this way, Additive DenseNet can not only retain the effect of Dense Connections, but also overcome the enlarging dimension problem in DenseNet, and thus results in a massive reduce on model complexity, such as GPU memory usage or the number of parameters.

In this paper, as we focus on text classification task, thus we conduct Additive DenseNet and DenseNet based on multi-layer bidirectional Long Short-Term Memory (BiLSTM) [4]. And we implement two sets of experiments so as to make a comprehensive comparison between Additive DenseNet, DenseNet, and plain DNN (i. e., BiLSTM). Firstly, we elaborately compare the model complexity of three models, and the results show that in terms of parameter number, GPU memory usage and floating-number operations (FLOPs), Additive DenseNet costs nearly the same amount of resources as plain DNN, while DenseNet costs dramatically more resources than others; Secondly, we continue to evaluate the performance of three models on 6 text classification datasets, and the results show that in terms of accuracy, Additive DenseNet can outperform others at all datasets, and for model training, it can achieve competitive performance as DenseNet.

To sum up, compared to DenseNet, Additive DenseNet shows superiority on performance and resource economy. Besides, for its counter parts, it can enjoy a fully dense feature reuse without any compromise, and show great simplicity on theory.

2. Related work

The rapid increase of network depth indeed used to accelerate the success of DNNs, while the excessive depth also introduces the vanishing gradient problem and thus hampers the further advances of DNNs [9, 24, 31]. In literature, Skip Connections are proven to be a powerful category of approaches for overcoming this problem [18, 35]. And among all the variants of them, ResNet and DenseNet have witnessed wide applications and a fast speed of evolution in recent

years [36, 40], which paved the way for our Additive DenseNet.

As far as ResNet is concerned, Li et al. [26] proposed an improved ResNet by parameterizing the identity mapping. For each layer in ResNet, they assigned a parameter k to adjust the identity mapping, and thus provided an appropriate gradient increment for parameters among layers; To figured out why ResNet works so well, He et al. [37] conducted a theoretical analysis on it by studying the influence of identity mapping on the hypothesis complexity of the neural network in terms of the covering number of its hypothesis space. And they finally provided a covering bound and a generalization bound for ResNet; Besides, Zhang et al. [38] proposed a simple yet principled approach to boosting the robustness of ResNet via characterizing it based on an explicit Euler method, and they further proved that a small step factor h in Euler method can effectively improve the training and generalization robustness of ResNet; At present, as the novel COVID-19 pandemic has spread all over the world, Farooq et al. [39] proposed the COVID-ResNet to better identify the presence of COVID-19 and distinguish it from other forms of flus with ResNet.

As for DenseNet, nowadays researchers have reached a consensus that Dense Connections are prohibitively expensive and remain much scope for improvement [40]. Chen et al. [27] proposed a novel model which combines DenseNet and Attention mechanism. And those connections carrying important features can be highlighted by Attention mechanism; Further, Tong et al. [54] proposed the Channel-Attention-Based DenseNet by integrating channel-level Attention mechanism in channel domain to adaptively adjust feature weights; Considering that down-sampling is widely utilized in DenseNet to reduce the size of features by dropping some feature information, Tang [55] refined DenseNet by explicitly introducing up-sampling to enable better feature reuse without loss. In order to alleviate excessive computational complexity and parameter growth of DenseNet, other researchers have also focused on simplifying its connectivity pattern. Actually, it is the inventors of DenseNet that initially implemented correlated research, and thus DenseNet-BC was proposed [15] by introducing a compression factor θ to adjust the compactness of Dense Connections with just keeping θ% of them; Huang et al. [30] proposed the CondenseNet by utilizing a module named learned group convolution so as to adaptively remove certain unnecessary connections in Dense Connections during training, and thus facilitates CondenseNet to be more efficient; Moreover, Liu et al. [41] proposed SparseNet, which features removing the connections from middle layers and only retaining the connections from most two previous layers, then extending network's width or depth as much as possible. The results revealed that under same quantity of parameters, SparseNet can outperform DenseNet; Similar to CondenseNet, Feng et al. [34] also simplified Dense Connections by trimming them in two ways: one is to fix the trimming rate (e. g., 50%), and the other is to fix the quantity of Dense Connections. The results revealed that even trimming 75% Dense Connections will not harm the performance, but obtain a certain improvement, which proves that Dense Connections indeed have much redundancy. Overall, the core of DenseNet-BC, CondenseNet, and SparseNet can be classified to the extensions of down-sampling, as they aim at restricting the quantity of Dense Connections, which equals to restrict the scale of feature reuse. Thus as reported in [55], they may also lose some vital features.

As described above, recent works mainly focused on improving the connectivity pattern of Dense Connections, while ignoring another key point, i. e. it is concatenation operations that should also be blamed for the high complexity of Dense Connections as they enlarge the inputs' dimensions. And obviously, above works are unable to deal with this issue, and their methods are quite complex. Thus to some extent, the main drawback of DenseNet has not be addressed yet.

Different from all of above works, our Additive DenseNet aims at improving the way to achieve feature reuse rather than connectivity pattern of DenseNet. Learning from ResNet, Additive DenseNet utilizes accumulating operations (upgraded from addition operations in ResNet) to retain dense feature reuse as well as preserve dimensions from enlarging. In this way, Additive DenseNet can effectively reduce the model complexity compared to DenseNet, and further achieve more competitive performance. Moreover, in comparison to DenseNet-BC, CondenseNet, and etc., Additive DenseNet features retaining the integrity of Dense Connections without any compromise on the quantity of connections, or feature reuse.

3. Background

In this section, we firstly review the necessary background information of BiLSTM, ResNet and

DenseNet. And then we delve into the proposed Additive DenseNet.

3.1. BiLSTM

Long Short-Term Memory (LSTM) was firstly proposed in [42] and modified in [43], and it has become one of the most popular models in NLP [44]. Given a sentence$S = [x_1, x_2, ..., x_k]$, k denotes the length of S. And at time $t(1 \leq t \leq k)$, a LSTM computes the hidden state h_t of x_l by iterating the following equation:

$$h_t = LSTM(h_{t-1}, x_t) \tag{1}$$

Where $LSTM(\cdot)$ denotes the computations of LSTM, which can be written as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3}$$

$$\widetilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{4}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C}_t \tag{5}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{6}$$

$$h_t = o_t \cdot \tanh(C_t) \tag{7}$$

Where Equations (2), (3), and (6) denote the formulas of forget gate, input gate and output gate with corresponding weight matrixes W_f, W_i, W_o and bias vectors b_f, b_i, b_o. The symbol $\odot$ denotes the element-wise multiplication. Besides, Equations (4) and (5) denote the formulas of candidate cell state and cell state respectively, and W_C, b_C represent the weight matrix and bias vector. Finally, Equation (7) denotes the formula of hidden state at time t.

The vanilla LSTM only processes text according to temporal order, in other words, it only captures features unidirectionally. Consequently, Bidirectional LSTM (BiLSTM) [4, 45] was proposed to process text in bidirectional way: forward and backward. Specially, a Bidirectional LSTM uses two independent LSTM to process text forwardly and backwardly, and then obtains the forward and backward hidden states, which can be denoted as $\overrightarrow{h_t}$, $\overleftarrow{h_t}$ respectively, the formulas of $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ can be written as follows:

$$\overrightarrow{h_t} = LSTM(h_{t-1}, x_t) \tag{8}$$

$$\overleftarrow{h_t} = LSTM(h_{t-1}, x_t) \tag{9}$$

And the hidden state of BiLSTM at time t is the concatenation of $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$, which can be denoted as follows:

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}] \tag{10}$$

Where $[\cdot]$ denotes the concatenation operation.

3.2. ResNet

Given a plain DNN, and consider that x_l and y_l are the input and output for the l^{th} layer $(1 \leq l)$ in the given DNN, respectively. $H_l(\cdot)$ represents the non-liner transformation performed by the l^{th} layer [38]. Hense, the output of the l^{th} layer can be written as:

$$y_l = H_l(x_l) \tag{11}$$

Where $x_l = y_{l-1}$, and specially, y_0 denotes the output of the input layer.

Equation (11) denotes that the (actual) output of the l^{th} layer (y_l) equals the result of l^{th} layer's non-liner transformation ($H_l(\cdot)$) given the input x_l, or in other words, the hidden output of the l^{th} layer. And we can observe that there is only one term for each layer's output. The structure of a plain DNN is visualized in Fig. 1(a).

The most remarkable contribution of the ResNet is that it builds a direct path (i.e., identity mapping) which connects the output and input at each layer. In ResNet, the output of the l^{th} layer $(1 \leq l)$ can be written as:

$$y_l = H_l(x_l) + x_l = H_l(y_{l-1}) + y_{l-1} \tag{12}$$

Equation (12) denotes that there are two terms for each layer's (actual) output in the ResNet, namely $H_l(x_l)$ and x_l. Technologically, ResNet enables the input of each layer to directly be a part of the output via identity mapping, or more mathematically, an addition operation. In other words, information flowed in the ResNet features "bypassing" one layer and directly connecting to the deeper layers. What's more, observed by Equation (12), identity mapping does not introduce any extra parameters for training, thus ResNet is resource-efficient [53]. And the structure of ResNet is visualized in Fig. 1(b).

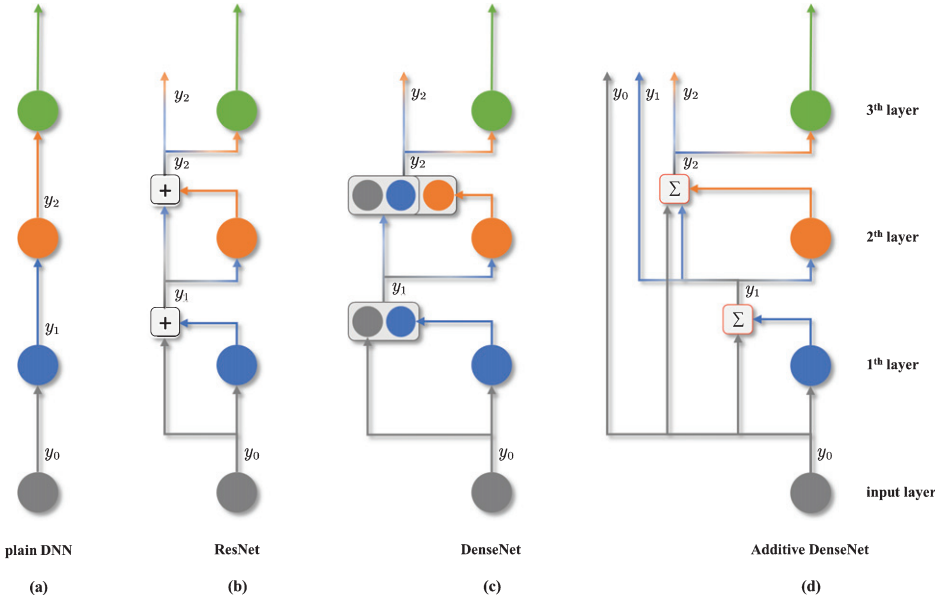

Fig. 1. A simple visualization of structures of plain DNN, ResNet, DenseNet and Additive DenseNet. Specially, only the most 3 previous layers are depicted.

3.3. DenseNet

As mentioned before, DenseNet features the so-called Dense Connections which enable each layer's input to be the concatenation of all the preceding layers' outputs. In this way, each layer in DenseNet can enjoy a "dense" feature reuse since the input obtains the information learned by all the preceding layers. Specially, the output of the l^{th} layer $(1 \leq l)$ in DenseNet can be written as:

$$y_l = H_l(x_l) = H_l([y_0; y_1; ...; y_{l-2}; y_{l-1}]) \quad (13)$$

Where $x_l = [y_0; y_1; ...; y_{l-2}; y_{l-1}]$.

The symbol $[y_0; y_1; ...; y_{l-2}; y_{l-1}]$ represents the effect of Dense Connections, which concatenates the outputs of former $l-1$ layers at a certain dimension. And the structure of DenseNet is visualized in Fig. 1(c).

From Equation (13), we can easily deduce that the dimensions of input x_l will become larger and larger with the increase of the depth. For example, assume that the dimension of $H_l(x_l)$ $(1 \leq l)$ and y_0 is 50, thus for x_l, the dimension equals $50 * l$. Consider that a large value of l (namely a very deep network) will bring excessive dimensions as well as computation complexity for later layers, which imposes heavy press on hardware and training efficiency. Specially, in this paper we refer to this problem as enlarging dimension problem. To sum up, it is Dense Connections that make DenseNet a promising architecture, as they achieve dense feature reuse. However, from another point of view, Dense Connections inevitably set a high standard on resources and consequently lower the efficiency of DenseNet, as they result in the annoying enlarging dimension problem, which undoubtedly should blame on concatenation operations.

4. Additive DenseNet

The above discussion inspires us that if one wants to make an improvement on vanilla DenseNet, Dense Connections are the right place to start. In response to this point, we propose an improved version of DenseNet called Additive DenseNet. Considering that concatenation operations lead to the enlarging dimension problem, thus motivated by the work of ResNet, the proposed Additive DenseNet replaces (complex) concatenation operations with (simple) addition operations used in Equation (12). And specially, the output of the l^{th} layer in Additive DenseNet can be written as:

$$y_l = H_l(x_l) = H_l\left(\sum_{i=0}^{l-1} y_i\right) \quad (14)$$

Equation (14) denotes that in order to fully achieve a dense feature reuse as DenseNet does,

our Additive DenseNet utilizes accumulating operations (upgraded from addition operations) to replace concatenation operations (denoted in Equation (13)), which enables each layer's input to be the sum of all the preceding layers' outputs. Compared to Equation (13), because of accumulating operations, our Additive DenseNet can keep the dimension of input unchanged like ResNet, and thus completely avoid the enlarging dimension problem which DenseNet suffers a lot. Moreover, compared to Equation (12), our Additive DenseNet effectively enriches the information for y_l. Concretely speaking, there are l terms for the y_l in Additive DenseNet, while only 2 in ResNet, which means Additive DenseNet can enjoy more diverse features. And the structure of Additive DenseNet is visualized in Fig. 1(d).

Finally, as Additive DenseNet retains the integrity of Dense Connections, in other words, each layer can still enjoy a dense feature reuse from all the previous layers. Thus it does not make any comprise on feature reuse, which distinguishes it from other counterparts like DenseNet-BC, CondenseNet, and etc. From this perspective, Additive DenseNet can be regarded as a "densely connected" ResNet.

5. Experiments

5.1. Implementing details

We utilize 6 text classification datasets as follows:

- MR: Movie Review (MR) is a sentiment classification dataset proposed by [46]. Each review has binary labels (negative, positive) and contains only one sentence.
- SST2: Stanford Sentiment Treebank is an extension of MR [47]. And each review has binary labels (negative, positive), thus this dataset is denoted as SST2. Specifically, followed with [48], in all experiments, phrases and sentences are used to train the model, but only sentences are scored at test time.
- SUBJ: proposed by [49], this dataset is to classify a sentence into two classes: subjective or objective.
- TREC: TREC is a dataset for question type classification task [50]. The sentences are questions from 6 classes (person, location, numeric information, etc.)
- MPQA: MPQA is a serialized Pandas Data Frame with about 8000 labeled feature vectors that were derived from the annotated MPQA Corpus to model phrase and sentence level sentiment (polarity) in news and editorial content [51].
- CR: CR [52] is a dataset based on customer reviews of various products. Each review has binary labels (negative, positive).

First of all, for modeling details, since this paper focuses on text classification task, the Densely Connected BiLSTM (DC-BiLSTM) [48], the application of DenseNet in BiLSTM, is chosen as the instance and representative of DenseNet, and for simplicity, in all experiments, we refer to DC-BiLSTM as DenseNet. Based on DC-BiLSTM, we further conduct our Additive DenseNet according to Equation (14). What's more, considering above two architectures are based on BiLSTM, the vanilla multi-layer BiLSTM is reasonably chosen as the baseline for all the experiments, and it is also regarded as the instance and representative of plain DNN.

For setting details, the hidden size of BiLSTM is fixed to 100, and the dimension of word embedding is 300, and we use the 300-dimension Glove vectors which were trained on 42 billion words for word embedding.

For training details, we use stochastic gradient descent algorithm and adam update rule with shuffled minibatch. The batch size and learning rate are set to 128 and 0.005 respectively. As for regularization, dropout is applied with rate of 0.5 for word embedding.

5.2. Comparison on model complexity

As discussed before, due to the utilization of accumulating operations, the proposed Additive DenseNet is free of the enlarging dimension problem that vanilla DenseNet suffers a lot. In order to have a deep understanding of this point, we firstly conduct a set of experiments to demonstrate the model complexity of BiLSTM, DenseNet, and Additive DenseNet, in terms of three metrics: quantity of parameters, GPU memory usage, and floating-number operations (FLOPs). Specially, we count FLOPs based on the one epoch of test set of MR dataset. The results are listed in Table 1, and specially, abbreviation A. DenseNet represents our Additive DenseNet.

Observed by Table 1, we can conclude that the proposed Additive DenseNet consumes exactly or nearly the same amount of resources as multi-layer

Table 1
The comparison of model complexity of 3 models

Depth	Model	Quantity of parameters	GPU memory usage (MB)	FLOPs
5-layer	BiLSTM	226,702	343	2,675,475,660
	A. DenseNet	226,702	343	2,677,779,660
	DenseNet	526,702	692	6,210,375,660
10-layer	BiLSTM	453,202	706	5,350,270,040
	A. DenseNet	453,202	706	5,355,454,040
	DenseNet	1,803,202	2,239	21,257,320,040
15-layer	BiLSTM	679,702	1,156	8,025,064,420
	A. DenseNet	679,702	1,156	8,033,128,420
	DenseNet	3,829,702	4,293	45,141,414,742
20-layer	BiLSTM	906,202	1,654	10,699,760,072
	A. DenseNet	906,202	1,654	10,710,704,072
	DenseNet	6,606,222	7,668	77,862,860,072

BiLSTM under all three metrics, while DenseNet consumes dramatically more resources than other two models.

Concretely speaking, with regard to our Additive DenseNet, Equation (14) has denoted that Additive DenseNet causes neither extra parameters nor dimension enlarging, which is proven by the statistics listed in Table 1, as Additive DenseNet consumes exactly the same quantity of parameters and GPU memory as BiLSTM does at same depths. What's more, compared to BiLSTM, our Additive DenseNet merely introduces small amount of computation complexity for performing accumulating operations. Specifically, at a depth of 5, 10, 15, 20, our Additive DenseNet needs merely 0.086%, 0.097%, 0.100%, 0.102% more FLOPs than BiLSTM, respectively. Hence, in terms of computation complexity, our Additive DenseNet is almost the same as BiLSTM.

However, we can also learn from Table 1 that in general, DenseNet is quite a resource-expensive model. With the increase of depth, the amount of consumed resources grows extremely faster. On the one hand, at a depth of 5, 10, 15, 20, DenseNet costs 132.3%, 297.9%, 463.4%, and 629.0% more parameters than BiLSTM (and Additive DenseNet), respectively; As for GPU memory usage, the statistics are 101.7%, 217.1%, 271.4%, and 363.6%, respectively; And for computation complexity, the statistics are 132.1%, 297.3%, 462.5%, and 627.7%, respectively; On the other hand, if we set the model complexity at 5-layer as baseline, we can find that for DenseNet, it costs 242.3%, 627.1%, 1154.2% more parameters, 223.6%, 520.4%, 1008.1% more GPU memory, and 242.3%, 626.9%, 1153.8% more FLOPs at 10-layer, 15-layer, 20-layer compared to its baseline. And in sharp contrast with DenseNet, Additive DenseNet only costs 99.9%, 199.8%, 299.7%

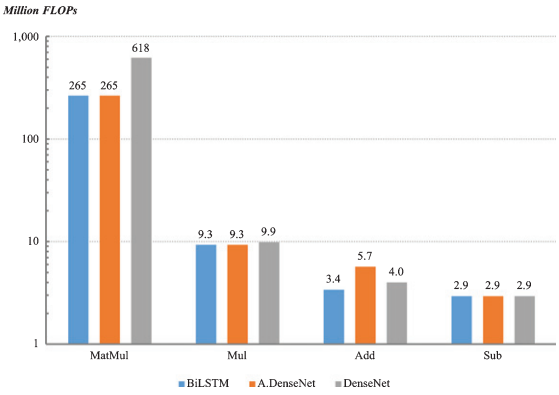

Fig. 2. The main 4 components of FLOPs at 5-layer.

more parameters, 105.8%, 237.0%, 382.2% more GPU memory, and 100.0%, 200.0%, 300.0% more FLOPs compared to its baseline.

The above statistical analysis substantially unveils an astonishing truth, i. e., the cost of DenseNet will grow at an extremely fast speed with the increase of depth, while our Additive DenseNet grows at a moderate speed. From this point of view, DenseNet indeed is inadequate for building DNNs.

In order to figure out why DenseNet is so resource-expensive, as show in Figs. 2–5, we visualize the main 4 components of FLOPs for three models at a depth of 5-layer, 10-layer, 15-layer and 20-layer respectively. For Figs. 2–5, abbreviations MatMul, Mul, Add, and Sub represent matrix multiplication, multiplication, addition, and subtraction operations, respectively.

Figures 2–5 are informative and instructional. To begin with, it is noticeable that matrix multiplication operations continuously dominate over the models' total FLOPs at all depths, e. g., at a depth of 5, they account for about 99.1%, 99.0%, 99.5% of

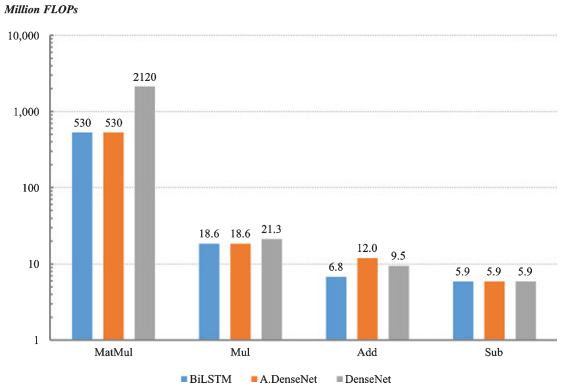

Fig. 3. The main 4 components of FLOPs at 10-layer.

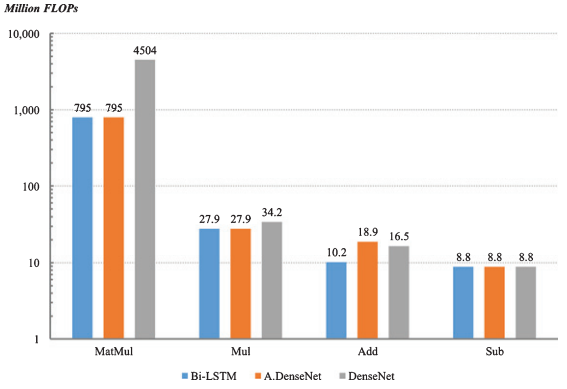

Fig. 4. The main 4 components of FLOPs at 15-layer.

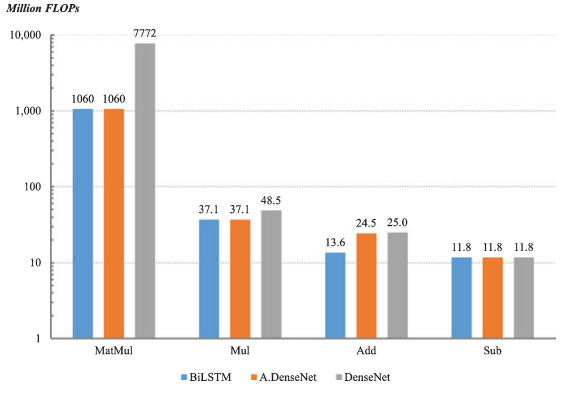

Fig. 5. The main 4 components of FLOPs at 20-layer.

total FLOPs for BiLSTM, Additive DenseNet and DenseNet, and similar conclusions can be found at other depths.

Secondly, it is easy to observe that DenseNet consumes dramatically more matrix multiplication operations than other models, and in terms of the rest 3 types of operations, FLOPs on all 3 models are exactly or nearly the same. Specifically, if we set the 5-layer's FLOPs as baseline, DenseNet costs 243.0 %, 628.9%, and 1157.6% more matrix multiplication operations at 10-layer, 15-layer, 20-layer, respectively, while Additive DenseNet only costs 200%, 300%, 400% more matrix multiplication operations. Thus we can conclude that in DenseNet, matrix multiplication operations will grow at a terribly fast speed with the increase of network depth. Recall that in DenseNet, dimension of input will gradually enlarge with the increase of network depth, and unfortunately, according to Equations (2)–(7), there are many affine transforms ($Wx+b$) in the calculations of LSTM. Thus this drawback will naturally cause the enlarging of weight matrixes as well as the ballooning of matrix multiplication operations, which are precisely demonstrated by Figs. 2–5. As a result, DenseNet is inherently resource-expensive.

Thirdly, as for Additive DenseNet, it costs a little more FLOPs merely on addition operations compared to BiLSTM. For instance, Additive DenseNet only introduces 2.3 m, 5.2 m, 8.7 m, 10.9 m more FLOPs on addition operations at a depth of 5, 10, 15, 20, which is a negligible cost for overall computation complexity. As for the rest 3 types of operations, it costs exactly the same FLOPs as BiLSTM. And this phenomenon is in accordance with the effect of Equation (14), in which accumulating operations are introduced. Moreover, note that addition operations make up a very small percent of total FLOPs, thus generally speaking, Additive DenseNet keeps almost the same as BiLSTM in terms of FLOPs, which is a huge advantage to DenseNet.

To sum up, the above statistical analysis reveals that our Additive DenseNet enjoys a high level of resource economy as it immensely reduces the model complexity compared to DenseNet and costs almost the same amount of resources as plain DNN (BiLSTM) at same depth.

5.3. Comparison on performance

This section we further evaluate the performance of three models based on chosen datasets at 5-layer and 10-layer. The metrics are accuracy and training loss value. The statistics of accuracies are listed in Tables 2 and 3 respectively. Moreover, the training loss curves at both depths are demonstrated in Figs. 6 and 7. Specially, training loss values are all evaluated on SUBJ dataset.

Firstly, observed by Table 2, we can find that the proposed Additive DenseNet continuously shows

Table 2
The comparison of accuracies for 3 models at 5-layer

Model	Accuracy (%)					
	MR	SUBJ	TREC	CR	MPQA	SST2
BiLSTM	79.96	92.90	82.07	81.49	88.86	83.08
A.DenseNet	80.48	93.82	85.38	82.85	89.66	83.84
DenseNet	80.13	93.57	84.97	82.26	89.38	83.48

Table 3
The comparison of accuracies for 3 models at 10-layer

Model	Accuracy (%)					
	MR	SUBJ	TREC	CR	MPQA	SST2
BiLSTM	51.08	54.75	65.09	66.43	80.89	50.34
A.DenseNet	81.30	94.21	86.85	83.38	90.34	84.77
DenseNet	81.03	93.91	86.57	83.05	89.90	84.30

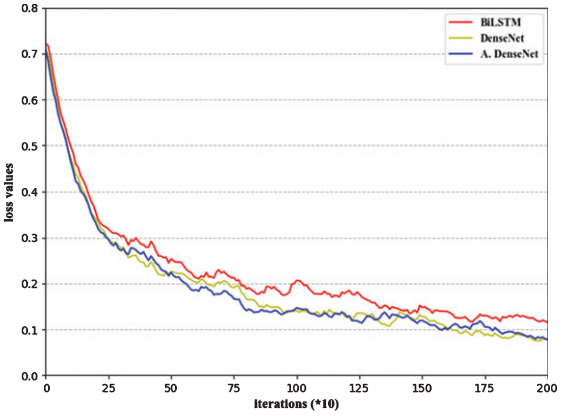

Fig. 6. The training loss curves at 5-layer.

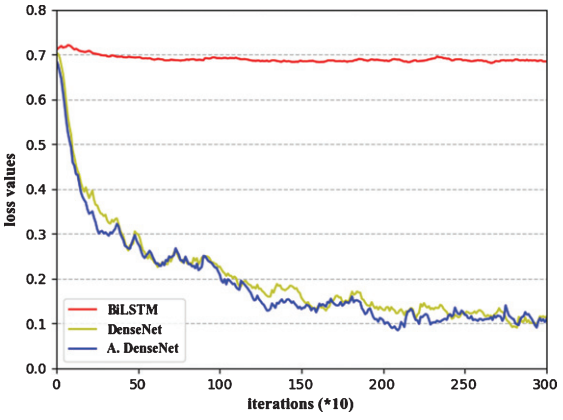

Fig. 7. The training loss curves at 10-layer.

certain improvements over other two models among all 6 datasets. For instance, compared to DenseNet, our Additive DenseNet can obtain a gain of 0.35%, 0.25%, 0.41%, 0.28%, and 0.36% on MR, SUBJ, TREC, CR, MPQA, and SST2, respectively. And with regard to model training, we can also learn from Fig. 6 that Additive DenseNet and DenseNet can perform better than BiLSTM as they converge faster and achieve smaller training losses. Besides, with regard to model training, Additive DenseNet performs almost the same as DenseNet at this depth.

Secondly, observed by Table 3, it is easy to note that a striking phenomenon occurs, namely at a depth of 10-layer, vanilla BiLSTM starts to suffer from the vanishing gradient problem as its performance degrades severely compared to 5-layer, and from Fig. 7 we can observe that at this point BiLSTM is no longer trainable because its training loss curve is almost flat with large loss values. On the contrary, other two models can continue to enjoy the gain of increased depth and obtain certain improvements compared to 5-layer. Moreover, our Additive DenseNet still outperforms DenseNet among all 6 datasets. For example, compared to DenseNet, our Additive DenseNet can obtain a gain of 0.27%, 0.30%, 0.28%, 0.33%, 0.44%, and 0.47% on MR, SUBJ, TREC, CR, MPQA, SST2, respectively. Again, in terms of model training, we can learn from Fig. 7 that Additive DenseNet performs better than DenseNet as it can converge slightly faster.

In summation, according to the above discussions, our Additive DenseNet is proven to be a promising architecture as it can immensely reduce the model complexity and obtain better performance. In detail, Additive DenseNet can continuously outperform DenseNet on accuracy and in the meantime, it can also show competitive performance in terms of model training.

6. Conclusion & future work

This paper we propose an improved version of DenseNet named Additive DenseNet. Different from many existing works, our Additive DenseNet features focusing on improving the way to achieve dense feature reuse, rather than the connectivity pattern of DenseNet as its counterparts mainly do. In terms of feature reuse, Additive DenseNet replaces the concatenation operations with addition operations to keep dimension of the each layer's input from enlarging, and further upgrades addition operations to accumulating operations to achieve dense feature reuse. Thus to some extent, Additive DenseNet can be regarded as a "densely connected" ResNet. Accord-

ing to our experimental results, Additive DenseNet can outperform DenseNet on all 6 datasets under the premise of substantially reducing the model complexity to the level of a plain DNN. And in contrast to existing methods, Additive DenseNet retains the integrity of Dense Connections without any loss on feature reuse, and shows great simplicity on theory.

As for future work, apparently in this paper we only evaluate Additive DenseNet on text classification task, and considering that the vanilla DenseNet and ReNet were initially applied to the field of computer version, thus we plan to apply our Additive DenseNet to the same filed in the future.

Acknowledgments

This work was supported in part by the Second Batch of Excellent Teaching Teams in Suzhou Vocational Colleges under Grant 201905, and in part by Suzhou Excellent Science and Technology Service Team Fund under Grant 201509.

References

[1] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, Daniel Fink, et al., "Evolving deep neural networks," [Online] (2017). Avaliable: https://arxiv.org/abs/1703.00548

[2] Y. Qi et al., Hedging deep features for visual tracking, *IEEE Trans. Pattern Anal. Mach. Intel.* **41**(5) (2019), 1116-1130.

[3] S. Yu, J. Yang, D. Liu, R. Li, Y. Zhang and S. Zhao, Hierarchical data augmentation and the application in text classification, *IEEE Access* **7** (2019), 185476–185485.

[4] A. Graves, A. Mohamed and G.E. Hinton, Speech recognition with deep recurrent neural networks, *Int. Confer. Acoust., Speech, Signal Process.*, Vancouver, BC, Canada, 2013, 6645-6649.

[5] G. Wu, J. Han, Z. Lin, G. Ding, B. Zhang and Q. Ni, Joint image-text hashing for fast large-scale cross-media retrieval using self-supervised deep learning, *IEEE Trans. Ind. Electron.* **66**(12) (2019), 9868–9877.

[6] S. Zhao, H. Yao, Y. Gao, G. Ding and T. Chua, Predicting personalized image emotion perceptions in social networks, *IEEE Trans. Affective Comput.* **9**(4) (2018), 526-540.

[7] W. Zhang, Y. Li and S. Wang, Learning document representation via topic-enhanced LSTM model, *Knowl.-Based Syst.* **174** (2019), 194–204.

[8] X. Li, S. Feng, D. Wang and Y. Zhang, Context-aware emotion cause analysis with multi-attention-based neural network, *Knowl.-Based Syst.* **174** (2019), 205–218.

[9] R.K. Srivastava, K. Greff and J. Schmidhuber, Training very deep networks, *Adv. Neural Infor. Process. Syst*. Lile, France, 2015, 2377–2385.

[10] S. Zagoruyko and N. Komodakis, "Wide residual networks," [Online] (2016). Available: https://arxiv.org/abs/1605.07146

[11] M.D. Zeiler and R. Fergus, Visualizing and understanding convolutional neural networks, *Proc. Eur. Confer. Comput. Vision Pattern Recognit.*, Zurich, Switzerland (2014), 6-12.

[12] M.R. Minar and J. Naher, "Recent advances in deep learning: an overview." [Online] (2018). Available: https://arxiv.org/abs/1807.08169

[13] M. Bianchini and F. Scarselli, On the complexity of neural network classifiers: A comparison between shallow and deep architectures, *IEEE Trans. Neural Networks Learn. Syst.* **25**(8) (2014), 1553–1565.

[14] A. Khan, A. Sohail, U. Zahoora and A.S. Qureshi, A survey of the recent architectures of deep convolutional neural networks, *Artif. Intell. Rev.* **1** (2019), 1–62.

[15] G. Huang, Z. Liu, L.V. Maaten and K.Q. Weinberger, Densely connected convolutional networks, *Proc. IEEE Confer. Computer Vision Pattern Recognit.*, Puerto, Rico, USA (2017), 4700–4708.

[16] Y. Bengio, P. Simard and P. Frasconi, Learning long-term dependencies with gradient descent is difficult, *IEEE Trans. Neural Networks* **5**(2) (1994), 157–166.

[17] A. Zoheb, and N. Rajmalwar, "DenseNet Models for Tiny ImageNet Classification," [Online]. Available: https://arxiv.org/abs/1904.10429

[18] Y.N. Dauphin, A. Fan, M. Auli and D. Grangier, Language modeling with gated convolutional networks, *Proc. Int. Confer. Mach. Learn*., Sydney, Australia (2017), 933–941.

[19] Z. Wu, C. Shen and A.V. Hengel, Wider or deeper: Revisiting the resnet model for visual recognition, *Pattern Recognit.* **90** 119–133, 201p.

[20] J. Kuen, X. Kong, G. Wang and Y. Tan, DelugeNets: deep networks with efficient and flexible cross-layer information inflows, *Proc. IEEE Int. Confer. Comput. Vision Workshops*, Lagos, Nigeria (2017), 958-966.

[21] Maaløe Lars, Casper Kaae Sønderby, Søren Kaae Sønderby and Ole Winther, "Auxiliary deep generative models," [Online] (2016). Avaliable: https://arxiv.org/abs/1602.05473

[22] G. Larsson, M. Maire and G. Shakhnarovich, "Fractalnet: Ultra-deep neural networks without residuals," [Online] (2017). Available: https://arxiv.org/abs/1605.07648v1

[23] Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang and Bowen Zhou, "Improved neural relation detection for knowledge base question answering," [Online] (2017). Available: https://arxiv.org/abs/1704.06194

[24] J. Yang, S. Yu and Y. Zhang, Highway II, an extended version of highway networks and its application to densely connected Bi-LSTM, *Jour. Intel. Fuzzy Syst.* **37**(3), (2019) 4021-4032.

[25] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, *Proc. IEEE Confer. Comput. Vision Pattern Recognit.*, Las Vegas, Nevada, USA (2016), 770–778.

[26] B. Li and Y. He, An improved resnet based on the adjustable shortcut connections, *IEEE Access* **6** (2018), 18967-18974.

[27] B. Chen, J. Wang and Z. Chi, Improved DenseNet with convolutional attention module for brain tumor segmentation, *Proc. Int. Symp. Image Comput. Digital Med.*, Xi'an, Shaanxi, China (2019), 22–26.

[28] S. Li, J. Jiao, Y. Han and T. Weissman, "Demystifying ResNet," [Online] (2016). Available: https://arxiv.org/abs/1611.01186

[29] K. He, X. Zhang, S. Ren and J. Sun, "Identity mappings in deep residual networks," *Eur. Confer. Comput. Vision*, Amsterdam, Netherlands (2016), 630-645.

[30] G. Huang, S. Liu, L. Maaten and K.Q. Weinberger, Condensenet: An efficient densenet using learned group convolutions, *Proc. IEEE Confer. Comput. Vision Pattern Recognit.*, Salt Lake City, UT, USA, pp. 2752-2761.

[31] M. Zeng and N. Xiao, Effective combination of DenseNet and BiLSTM for keyword spotting, *IEEE Access* **7** (2019), 10767-10775.

[32] J. Kuen, X. Kong, G. Wang and Y. Tan, DelugeNets: deep networks with efficient and flexible cross-layer information inflows, *Proc. IEEE Int. Conf. Comput. Vision Workshops*, Venice, Italy, 2017, 958–966.

[33] G. Pleiss, D. Chen, G. Huang, T. Li, L. Maaten and K.Q. Weinberger, Memory-Efficient Implementation of DenseNets. [Online] (2017). Available: https://arxiv.org/abs/1707.06990

[34] M. Zeng and N. Xiao, Effective combination of DenseNet and BiLSTM for keyword spotting, *IEEE Access* **7** (2019), 10767–10775.

[35] I. Koné and L. Boulmane, Hierarchical resnext models for breast cancer histology image classification, *Int. Confer. Image Anal. Recognit.*, Póvoa de Varzim, Portugal (2018), 796–803.

[36] S. Zagoruyko and N. Komodakis, "Wide residual networks," [Online] (2016), Available: https://arxiv.org/abs/1605.07146

[37] F. He, T. Liu and D. Tao, Why resnet works? residuals generalize," [Online] (2019). Available: https://arxiv.org/abs/1904.01367

[38] J. Zhang, B. Han, L, Wynter, KH. L and Kankanhalli M. "Towards robust resnet: A small step but a giant leap," [Online] (2019). Available: https://arxiv.org/abs/1902.10887

[39] M. Farooq and A. Hafeez, "Covid-resnet: A deep learning framework for screening of covid19 from radiographs," [Online] (2020). Available: https://arxiv.org/abs/2003.14395

[40] H. Hu, D. Dey, A.D. Giorno, M. Hebert and J.A. Bagnell, "Log-DenseNet: How to sparsify a densenet," [Online] (2017). Available: https://arxiv.org/abs/1711.00002

[41] Wenqi Liu and Kun Zeng, "SparseNet: A sparse DenseNet for image classification," [Online] (2018). Available: https://arxiv.org/abs/1804.05340

[42] S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Comput.* **9**(8) (1997), 1735–1780.

[43] X. Shi, Z. Chen, H. Wang, D. Y. Yeung, W. Wong and W. Woo, Convolutional lstm network: A machine learning approach for precipitation nowcasting, *Neural Infor. Process. Syst.*, Montreal, Canada (2015), 802– 810.

[44] Alex Graves, "Generating Sequences with Recurrent Neural Networks," [Online] (2013). Available: https://arxiv.org/abs/1308.0850

[45] A. Graves, N. Jaitly and A. Mohamed, Hybrid speech recognition with Deep Bidirectional LSTM, *IEEE Workshop Autom. Speech Recognit. Understanding*, Olomouc, Czech (2013), 273–278.

[46] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," [Online] (2005). Available: https://arxiv.org/abs/cs/0506075

[47] R. Socher, A. Perelygin, J. Wu, et al., "Recursive deep models for semantic compositionality over a sentiment treebank." *Proc. Confer. Empirical Methods Nat. Lang. Process.*, Seattle, WA, USA (2013), 1631–1642.

[48] Z. Ding, R. Xia, J. Yu, X. Li and J. Yang, Densely connected bidirectional lstm with applications to sentence classification, *CCF Int. Confer. Nat. Lang. Process. Chinese Comput.*, Hohhot, Inner Mongolia, China (2018), 278–287.

[49] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," [Online] (2004). Available: https://arxiv.org/abs/cs/0409058

[50] X. Li and D. Roth, "Learning question classifiers," [Online] (2005). Available: https://www.aclweb.org/anthology/C02-1150.pdf

[51] T. Wilson, J. Wiebe and P. Hoffmann, "Recognizing contextual polarity in phrase-level sentiment analysis," [Online] (2005). Available: https://www.aclweb.org/anthology/H05-1044.pdf

[52] CR dataset: M. Hu, and B. Liu, "Mining and summarizing customer reviews," [Online] (2004). Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.76.2378&rep=rep1&type=pdf

[53] Xinjie Feng, Hongxun Yao and Shengping Zhang, An efficient way to refine DenseNet, *SIViP.* **13** (2019), 959–965.

[54] W. Tong, W. Chen, W. Han and L. Wang, Channel-attention-based densenet network for remote sensing image scene classification, *IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens.* **13** (2020), 4121-4132.

[55] Z. Tang, W. Jiang, Z. Zhang, M. Zhao, L. Zhang and M. Wang, Densenet with up-sampling block for recognizing texts in images, *Neural Comput. Appl.* **32** (2020), 7553–7561.