Recurrent Aggregation of Convolutional Neural Network

Anonymous ACCV 2018 submission

Paper ID 536

Abstract. Standard convolutional neural networks assemble multiple convolutional layers to extract high-level features. Recent efforts keep designing deeper and wider architectures. Even with skip connections applied to combine different layers, the useful low-level features are not effectively utilized. Some deep layer aggregation methods have been proposed to aggregate features of all levels, using simple linear combination or complex non-linear transformation. In this paper, we treat convolutional features as a sequence, and propose our Recurrent Aggregation of Convolutional Neural Network (CNN-RA). Our aggregation method splits a standard CNN into blocks and maps their feature matrices to a sequence of vectors of the same length. LSTM is employed to connect to the sequence and better fuse features across blocks. Our proposed CNN-RA can be directly appended to any standard CNN without any modifications. Experiments show remarkable improvements of CNN-RA over the original architectures across datasets.

1 Introduction

In recent years, Convolutional Neural Networks (CNNs) have shown remarkable advantage on computer vision tasks such as image classification [1], object detection [2] and semantic segmentation [3]. The basic architecture of convolutional layer consists of two levels, feature extraction and feature mapping. In feature extraction level, the input of each convolutional neuron is connected to local receptive domain and the local characteristics are extracted. Feature mapping level employs multiple convolutional kernels to focus on different aspects of the characteristics. The results of different convolutional layers are customarily regarded as features containing spatial and channel-wise information. A series of convolutional layers are stacked together to expand the field of reception and to generate higher level features. The evolution of CNNs from LeNet [4], AlexNet [5] to ResNet [6] and DenseNet [7] increases both the performance and the size of the network, which yields deeper and wider network structures.

From the first application in ResNet [6], skip connections have been introduced into CNN structures, and proven effective on various vision tasks. Skip connections combine the output of previous layer and the current layer to deal with the gradient vanishing problem. DenseNet [7] connects densely in a block to make better use of previous features. To further utilize features from different layers, Yu [8] extends the current skip connection approach and proposes deep layer aggregation (DLA) architectures. These architectures simply combine features of different level by concatenation or addition, without considering the interior relationship between low-level and high-level feature representations.

Recurrent Neural Networks (RNNs) [9] have been proposed to deal with sequential data like text and speech. Different from feedforward neural networks, RNNs build connections between neurons which are in the same layer. RNNs can be unfolded as a directed graph along the time steps, with all the layers sharing the same weights. This makes RNNs applicable to sequential tasks such as text classification. Long Short Term Memory (LSTM) [10] is a special RNN, which makes use of three gates to select valuable information from all the memories. LSTM has proven to be more efficient than normal RNNs in most tasks on sequences.

In this work, we propose a brand new approach to convolutional layer aggregation, by introducing a new architecture which is named as '*Convolutional Neural Network-Recurrent Aggregation*' (CNN-RA). Our goal is to use LSTM to aggregate outputs of multiple layers and retrieve more expressive features. To achieve this, we build CNN-RA by creating parallel connection between a CNN and a LSTM. Features from lower convolutional layers to higher layers naturally form a sequence with a various granularity of information. This kind of sequence contains both the features themselves and the transformation relationship between different levels of features, which directly leads us to RNNs. We create information between outputs of convolutional layers and the inputs of LSTM, and employ the outputs of LSTM as the final feature for tasks such as image classification.

The receptive fields and feature maps sizes of different convolutional layers vary from each other, especially for consecutive layers with a pooling layer in-between. We propose an algorithm to transform different shape of feature matrices to vectors with the same dimension. Then transformed vectors are stacked together as inputs of LSTM. The number of chosen features is the step length of the LSTM.

The development of new network architectures is always a time consuming task with abundant hyper parameters to determine. Previous work on layer aggregation such as DLA [8] brings with huge change on the original network architecture, which can even be much more complicated. However, our proposed CNN-RA won't do any modification on the original network, by only connecting it with a parallel LSTM. This property enables CNN-RA to be easily applicable to multiple convolutional network structures.

Our evaluation experiments extend famous network structures VGGNet [11] and ResNet [6] for image classification task on benchmark datasets. The testing results show improvements across different network structures and datasets over other state-of-the-art aggregation methods. The connected LSTM brings with higher performance without increasing much in parameter numbers. The experiment processes show that the relationship between two convolutional blocks with a down sampling layer inside has the most important contribution to the model.

2 Related Work

Convolutional neural networks (CNNs) have proven their great advantage on computer vision tasks. The successful application of LeNet [4] and AlexNet [5] on ImageNet [1] classification tasks first demonstrated the effectiveness of convolutional architecture for visual recognition. Inspired by the combination of convolutional, pooling and full-connected layers, VGGNet [11] proposed to assemble several layers with small convolutional kernels to replace a large kernel. It achieved huge improvement over AlexNet on almost every visual problem. ResNet [6] employed residual blocks to avoid vanishing gradient problem and enabled very deep network to work well. Skip connection was also first proposed in ResNet and helped improve performance over various visual problems. DenseNet [7] was an advanced edition of skip connection, because it connected each two layers within a block to maximize the data flow. DenseNet made better use of information within a block and show excellent accuracy over image classification tasks. CliqueNet [12] increased the complexity by proposing clique blocks. It applied bi-directional connection between each two layers in a block and built a bi-directional fully connected graph. CliqueNet made block layers recurrent and was denser than DenseNet.

Recurrent neural networks (RNNs) have been designed to deal with sequential data, which build connections between units in each layer. RNN was first proposed by [9] to learn long term dependencies. The architecture was designed to capture the contextual information and has been successfully applied in text recognition [13], automatic speech recognition [14] and video processing [15]. Long-short Term Memory (LSTM) [10] employed forget gate to consider both short-term and long-term memory. It was capable of remembering the long-term information and extracting effective information from a long sequence. Bi-directional LSTM (BLSTM) [16] focused on the imperfection of LSTM which only took previous information into consideration. BLSTM employed two LSTM structure with opposite direction and fused information of the whole sequence.

Layer aggregation fuses information of different layers to better utilize the features. FCN [17], FPN [18] and U-Net [19] employed simple skip connection to combine cross-layer information, which directly connected shallow layers to the end. Deep Layer Aggregation [8] proposed iterative deep aggregation (IDA) and hierarchical deep aggregation (HDA). IDA iteratively merged lower layers and deeper layers, to aggregate shallower features more sufficiently. This helped refine shallow features. HDA merged convolutional blocks in a tree, and combined different layers in parallel as well as in series. However, DLA made the network structures very complicated and did not aggregate features of different levels innovatively. The main idea of DLA is still skip connections ever different layers and stages.

3 Methodology

Convolutional layer aggregation is a combination of features from different layers. The output matrices of convolutional layers are regarded as expressive features for vision tasks. In general, shallow layers contain low-level features and deep layers contain high-level features. Existing work simply uses high-level features or a combination of all layers. In this work, we take into account both feature combination and the sequential transformation relationship between all levels. Output features of convolutional layers form a sequence and RNNs are designed to deal with sequential data. LSTM is employed to do the layer aggregation on CNN in this paper. The output shape of different layers varies, therefore we use mapping nodes to map selected features to the same dimension first, and then aggregate them to be the input of LSTM. The structure diagram of CNN-RA is shown in Fig. 1.

Recent network design tends to assemble several blocks together and simplifies the network representation. Several layers are grouped into a single block which is capable of realizing some feature extraction and transformation function. Our proposed aggregation method employs the output of blocks instead of all layers to focus on useful but not redundant information.

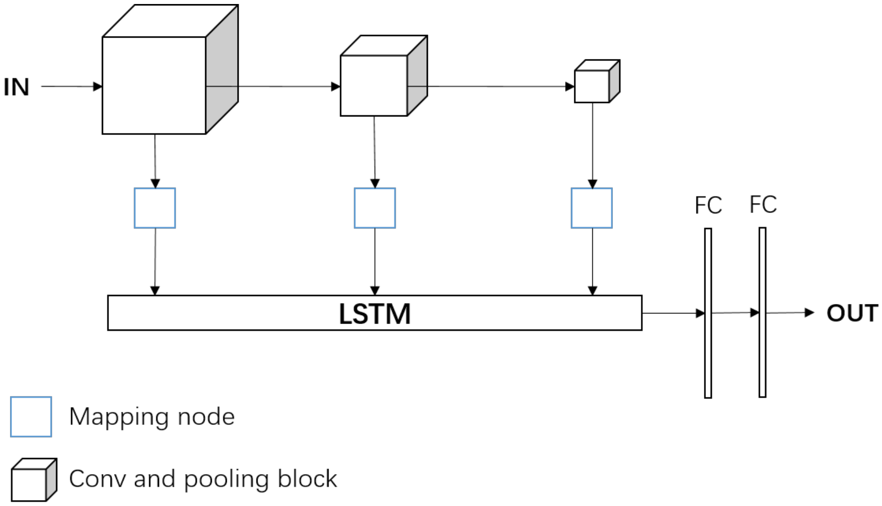

Fig. 1. Structure of CNN-RA

3.1 Convolutional Feature Mapping

Convolutional feature mapping helps transform feature matrices of different blocks into vectors with the same dimension. The feature resolution and number of channels vary among blocks because of convolution and down-sampling operations. Then the blocks are grouped into stages according to their output shape. Blocks within the same stage share common mapping operation which is different from other stages. In general, deeper layer holds smaller feature resolution and more channels. We propose to take the feature tensor of the last block as the standard $S \in \mathbb{R}^{H\times W\times C}$ and transform other matrices towards its dimension.

The standard S contains high-level features and each unit of S holds a large receptive field size due to previous convolution and down-sampling operations. For such output features we tend to exploit their channel dependencies instead of total information. Therefore, we do global average pooling on S to squeeze the spatial information into a channel-wise vector. Then a channel descriptor $D \in \mathbb{R}^C$ is generated with the c_{th} element D_c calculated by:

$$D_c = F_{gp}(S_c) = \frac{1}{H\times W}\sum_{i=1}^{H}\sum_{j=1}^{W} S_c(i,j). \tag{1}$$

Shallow blocks, where each unit can cover a limited region of receptive field, contain local information and this information helps extract fine features. Our proposed layer aggregation pays attention to both low-level and high-level features, and thus global average pooling operation is not suitable for these shallow blocks. We embed a convolutional layer with kernel size $k\times k$ and stride $l = k$ to combine local information, and then use average pooling layer with kernel size equal to the feature map shape of S. An output matrix $U \in \mathbb{R}^{h\times w\times c}$ is generated with $h\times w\times c = C$. We then expand the matrix U to a vector with the same length as D.

The receptive fields of blocks in middle stages are large enough to cover the input image, and the numbers of channels are normally less than C. For these kinds of blocks, we first embed a convolutional layer with channel number C to extend the channel number, and then take global average pooling on the output. A vector with length C is then generated.

The generated vectors share the same length and can be arranged into a matrix. Given a network structure with N blocks, we can generate a feature matrix $V \in \mathbb{R}^{N\times C}$.

3.2 Recurrent Aggregation using LSTM

After convolutional feature mapping we get a matrix $V \in \mathbb{R}^{N\times C}$. The matrix is constructed by assembling a sequence of N feature vectors $v \in \mathbb{R}^{1\times C}$ and we denote $V = [v_1, v_2, \cdots, v_N]$. As shown in Fig. 1, the sequential vectors are arranged according to the generated order in the CNN structure, and $[v_1$ to $v_N]$ denote features from low-level to high-level. Existing work on convolutional layer aggregation focuses on the linear and non-linear combination of different layers. Fig. 2(b) shows the shallowest aggregation, which simply makes linear combination on all blocks. Yu [8] proposed iterative deep aggregation (IDA) and hierarchical deep aggregation (HDA) as shown in Fig. 2(c) and Fig. 2(d). IDA iteratively merges lower layers with deeper layers to refine shallow features. HDA employs deep and branching structure to better preserve features.

Recurrent aggregation of CNN (CNN-RA) merges blocks as a sequence to both contain the spatial and channel-wise information and explore the transformation relationship according to the order. The mapped feature vectors share

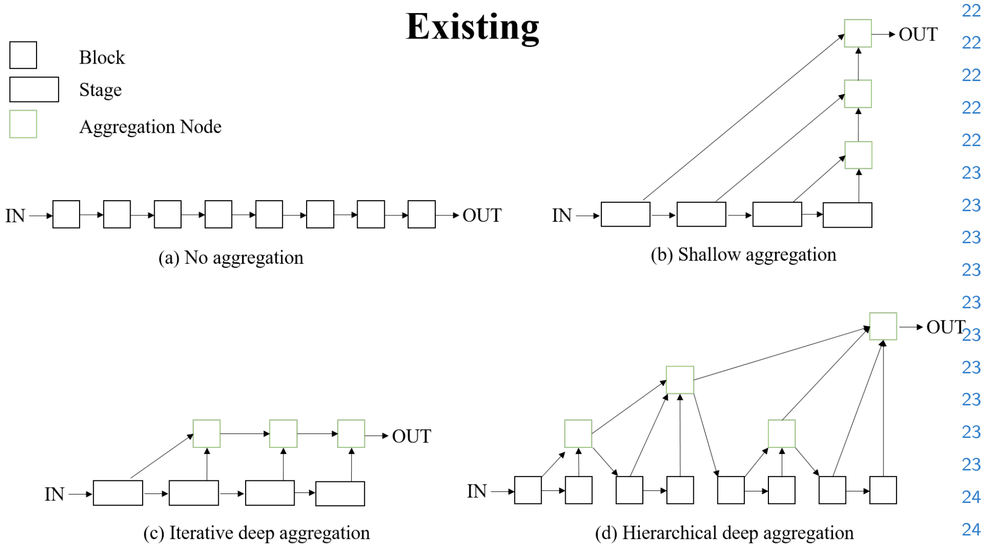

Fig. 2. Different existing approaches to aggregation. (a)default classification and regression networks without aggregation. (b)shallow linear combination of all layers by skip connection. (c)iterative deep aggregation to better aggregate shallow layers. (d)hierarchical deep aggregation to better preserve previous features.

the same length and form a regular sequence. One typical sequential data is text, and RNN has shown their tremendous advantage on processing text data. The successful application of RNNs mainly relies on their memory on contextual information. The memory mechanism helps combine all the input information together and is capable of detecting the correlation among all steps. LSTM employs input gate, forget gate and output gate to realize the long short-term memory, which naturally introduces attention mechanism to focus on useful information. LSTM takes input of n steps, with all the input vector sharing the same dimension and the mapped feature matrix $V \in \mathbb{R}^{N\times C}$ satisfies the requirements. The output vector of LSTM U can be calculated by:

$$U = F_{LSTM}(V) \tag{2}$$

where F_{LSTM} denote a standard LSTM structure with forget gates. The inputs of LSTM connect to the convolutional blocks through mapping nodes, and the gradient can be directly back propagated to all the blocks. These connections contribute to avoiding vanishing gradient problem and help convolutional blocks to extract more expressive sequential features.

The output vector U of LSTM contains squeezed and abstract information through the convolutional network. To utilize this abstract feature, we embed a simple non-linear combination before the output layer as:

$$u = \sigma(WU) \tag{3}$$

where σ denote the ReLU [20] function.

3.3 Examples: VGG-RA and ResNet-RA

The flexibility of recurrent aggregation using LSTM means that it can be directly appended to standard convolutional neural networks, without any modification on the original structure. Given a normal CNN, all we need to do is selecting output features to be employed, mapping them to vectors with the same length and arrange them into a matrix as the input of LSTM. In this paper, we apply CNN-RA to modern architectures VGGNet and ResNet to show the efficiency.

For non-blocked networks such as VGGNet, feature selection needs to be done before layer aggregation. In general, modern CNN architectures repeat serval convolution followed by a down-sampling operation to extract features. We classify the convolutional layers sharing the same feature resolution into the same stage, and split each stage into 1 or more blocks manually. Then the network is grouped into blocks with each blocks containing more than one convolutional layers. We construct VGG-RA network by simply choose the first three stages of VGGNet and aggregate it with a LSTM network. The VGG-RA network structure is shown in Fig. 3.

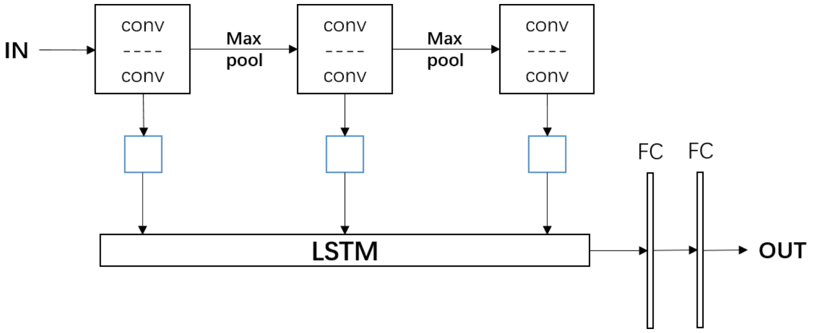

Fig. 3. Structure of VGG-RA

Blocked networks like ResNet are easy to apply CNN-RA, with each block outputing a feature matrix. For complex blocks such as a residual block with 20 or more convolutional layers, we can also split them into more sub-blocks. We employ ResNet with 3 residual blocks to build ResNet-RA, and the structure diagram is shown in Fig. 4.

4 Training Details

4.1 Feature selection

Given a standard CNN structure, we just select some of the convolutional layers to be aggregated. A large number of outputs bring with complexity and much

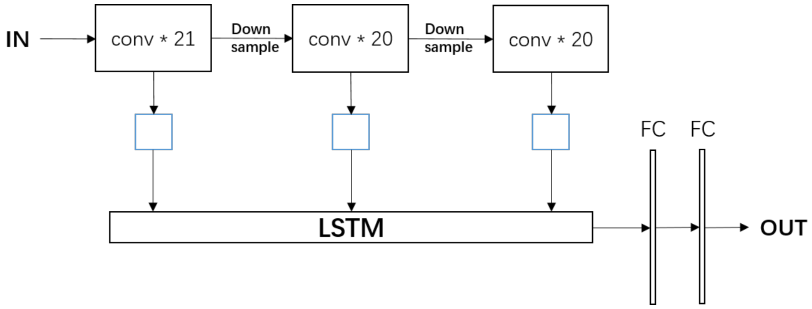

Fig. 4. Structure of ResNet-RA

redundant information, which brings down the appearance and convergence rate of LSTM. The selected blocks have to satisfy three rules we proposed: (a)each block contains at least 2 convolutional layers, (b)each stage contains at most 2 blocks, (c)the selected layer in each stage should be located before the corresponding down-sampling layer. According to these three rules, we split VGGNet and ResNet into 3 blocks, which is shown in Table 1.

4.2 Feature Mapping

Feature mapping nodes transform the feature matrix of CNN layers to the input vectors of LSTM. The feature resolution and channel number vary from shallow layers to deep layers, however, LSTM requires inputs of a common dimension. We first take global average pooling on the last block output and generate a standard vector. In general, previous blocks hold larger feature map and fewer channels. We propose to treat different blocks according to their feature resolutions, and group them into three levels, shallow level, middle level and last level. The last level only contains the final block. The middle level contains blocks whose feature resolutions is 2× or 1× larger than that of the final blocks. The shallow blocks contains the rest blocks.

The last block outputs a standard vector v_e with the same dimension as the last channel number C by global average pooling. For middle level blocks, we employ a single convolutional layer to transform the feature resolution and channel number into the same as the final block and then take the global average pooling operation. Blocks in shallow levels are handled specially, because their feature resolutions are at least 4× larger than the final block and the receptive fields may not cover the input size. These blocks contain more details information which we hope to utilize. Thus we first combine the spatial information by a convolutional layer with stride 2, and take average pooling with the kernel size equal to the shape of the final block. A feature matrix $u \in \mathbb{R}^{h \times w \times c}$ is then generated with $h \times w \times c = C$ and is expanded to v_s. All feature vectors are arranged into a matrix V as the input of LSTM.

4.3 Implementation Details

We verify the effectiveness of CNN-RA on three famous classification benchmark Cifar10 [21], SVHN [22] and Cifar100 [21]. All these three datasets contain images with size $32 \times 32 \times 3$. We do data augmentation on Cifar10 and Cifar100 by first resizing the images to $40 \times 40 \times 3$ and then randomly cropped them to $32 \times 32 \times 3$. We also flip the images at random.

Considering the small resolution of images, we do some manipulation on the original network structure. Standard VGGNet consists of five down-sampling layers and cannot be directly applied to small images. We select the first three stages with each containing two convolutional layers and a pooling layer, which we denote as VGG-1 in this paper. The selected network contains totally 6 convolutional layers and we group them into 3 blocks with each containing 2 layers. VGG-1 is shallow and each stage contains only two layers. We then double each block to 4 layers and generate a new network denoted as VGG-2. VGG-2 contains 12 layers and is also grouped into 3 blocks according to their feature resolutions. ResNet is block-wise and we assemble three blocks with each containing 20 convolutional layers. To eliminate the influence of complexity by applying LSTM, we embed a fully-connected layers before output layer in the re-implementation.

We apply CNN-RA to the three networks by appending a normal LSTM structure. The aggregated network is then correspondingly denoted as VGG-1-RA, VGG-2-RA and ResNet-RA. The structure parameters are listed in Table 1. We employ Batch-Normalization in each model to accelerate the convergence, and learning rate decay to optimize the convergence results.

Table 1. Network parameters of re-implementation and aggregation. GAP denotes global average pooling.

Network	Block1	Block2	Block3	Classifier
VGG-1	conv,128 × 2	conv,256 × 2	conv,512 × 2	1024d-fc×2, softmax
VGG-2	conv,128 × 2	conv,256 × 2	conv,512 × 2	1024d-fc×2, softmax
ResNet	conv,64 × 21	conv,128 × 20	conv,256 × 20	GAP, 256d-fc, softmax
VGG-1-RA	conv,128 × 2	conv,256 × 2	conv,512 × 2	256d-fc, softmax
	LSTM 256			
VGG-2-RA	conv,128 × 2	conv,256 × 2	conv,512 × 2	256d-fc, softmax
	LSTM 256			
ResNet-RA	conv,64 × 21	conv,128 × 20	conv,256 × 20	256d-fc, softmax
	LSTM 256			

5 Results

5.1 Classification with CNN-RA

We evaluate our recurrent aggregation methods on three image classification benchmark datasets Cifar10, SVHN and Cifar100 using three standard networks. We first re-implement VGG-1, VGG-2 and ResNet shown in Table 1, and train them on the three datasets. The networks are trained by SGD with momentum 0.9. We set weight decay as 10^{-4} and batch size as 64. For all the three datasets, we train 100 epochs. For Cifar10 and SVHN, the learning rates starts at 0.01, and is reduced by 10 at 50_{th} and 75_{th} epoch. For Cifar100, the learning rates starts at 0.005 and is reduced at 80_{th} epoch. The data augmentation is operated on Cifar10 and Cifar100 with random-cropping and flipping. Then we evaluate the trained model on the three test sets and take the results as the baselines and are shown in Table 2.

After the training and evaluation of the re-implemented network, we build VGG-1-RA, VGG-2-RA and ResNet-RA by joining LSTM and the original network with mapping nodes. To be fair, we train the aggregated network with the same procedure and hyper-parameters. We evaluate our aggregated network on the three test sets, and the results are shown in Table 2.

Table 2. Validation accuracies on three datasets. The number in parenthesis represents the improvements of network with CNN-RA over the original network.

Network	Cifar10	SVHN	Cifar100
VGG-1	93.73	97.97	73.01
VGG-1-RA	94.49(0.76)	98.11(0.14)	74.81(1.80)
VGG-2	94.91	98.14	75.66
VGG-2-RA	95.30(0.39)	98.32(0.18)	77.48(1.82)
ResNet	93.74	98.00	75.87
ResNet-RA	94.78(**1.04**)	98.19(**0.19**)	77.85(**1.98**)

In Table 2, the digits represent the improvement on testing accuracy of networks with CNN-RA over the original ones. For all three networks over the three benchmark datasets, our proposed recurrent aggregation method makes improvements. On Cifar10, ResNet-RA achieves over 1% on testing accuracy. On Cifar100, all the three networks with CNN-RA achieves more than 1% improvements, and ResNet-RA even increase remarkably 1.98%.

These comparison experiments show the effectiveness of CNN-RA. Considering the complexity and difficulty of the datasets themselves, Cifar100 contains 100 classes while Cifar10 and SVHN contains 10. The otherness of intra-class images in Cifar10 is much larger than SVHN. Thus we rank them as Cifar100, Cifar10 and SVHN according to their complexity. The testing results show that, all these three networks show better performance on simple datasets over the complex one, and networks with CNN-RA show the same phenomenon. Paying attention to the improvements on the three datasets and we notice that, the increment of accuracy of three networks is larger when the task is more complex. The improvement on SVHN is less than 0.2%, and that of Cifar10 achieves 1.04%, while the accuracy increases up to 1.98% over Cifar100. The difference of testing results is caused by two reasons. The first is that these networks are capable to deal with simple tasks well, and the upside potential is relatively smaller than the complex tasks. The second reason is concluded as the effectiveness of information aggregation. For simple tasks such as SVHN classification, high-level features are enough to distinguish different classes. However, for complex problems like Cifar100, the intra-class information is complicated with backgrounds and a variety samples of the same class. Then we need to utilize the features from low-level to high-level and their combination patterns. The results indicate that the recurrent aggregation method works better on more complex tasks. Another remarkable fact is that, the CNN-RA shows better results on ResNet which has totally 61 layers. In ResNet, each block contains 20 convolutional layers, which brings significant difference between two blocks. CNN-RA seems to work better on capturing feature transformation relationship of deeper blocks.

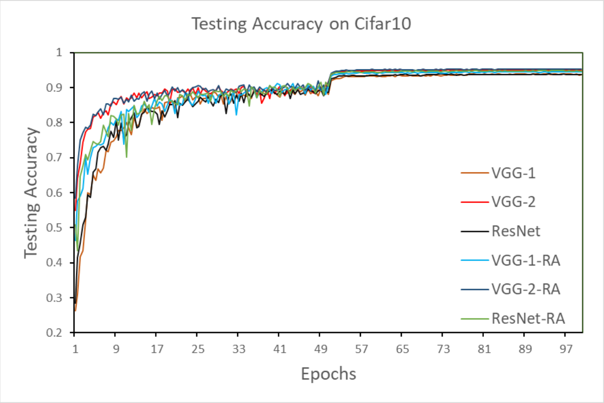

Fig. 5. Testing Accuracy on Cifar10

Our model takes different learning decay between Cifar10, SVHN and Cifar100. In Fig. 5 and Fig. 6 we show the training procedure on Cifar10 and Cifar100. It can be seen that, on Cifar10, recurrent aggregation method helps the network converge faster and show better final results. On Cifar100, all the three networks converge slower at the beginning with CNN-RA, however, they achieve higher performance after learning rate decay. ResNet-RA increases the most at the learning rate decay point. These learning curves show the different

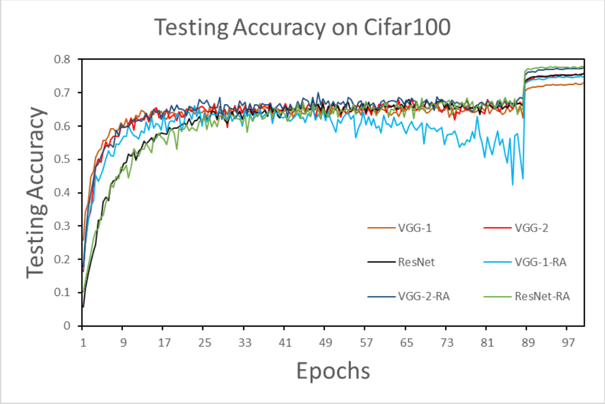

Fig. 6. Testing Accuracy on Cifar100

effect of CNN-RA on various tasks. For simple tasks, networks with aggregation converge faster and achieve better results in the end. For complex tasks, recurrent aggregation brings with lower convergence but higher accuracy eventually. The appended LSTM helps extract more expressive features, and it takes longer to better fit the complex dataset.

5.2 More Explorations

To better understand the effectiveness of our proposed CNN-RA, we explore some other methods to employ the outputs of LSTM.

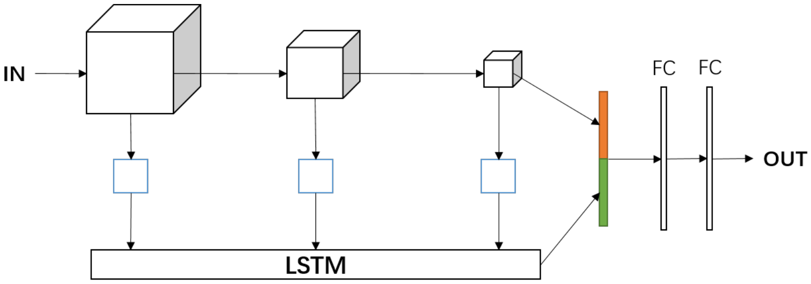

Fig. 7. Structure of CNN-RA-B

Given a network with N blocks, we employ a LSTM with N steps to aggregrate the features. In Fig. 1, only the final state is considered and directly used to do the classification. Fig. 7 show a combination of aggregated feature and the original high-level feature. We concatenate the output of LSTM and the feature

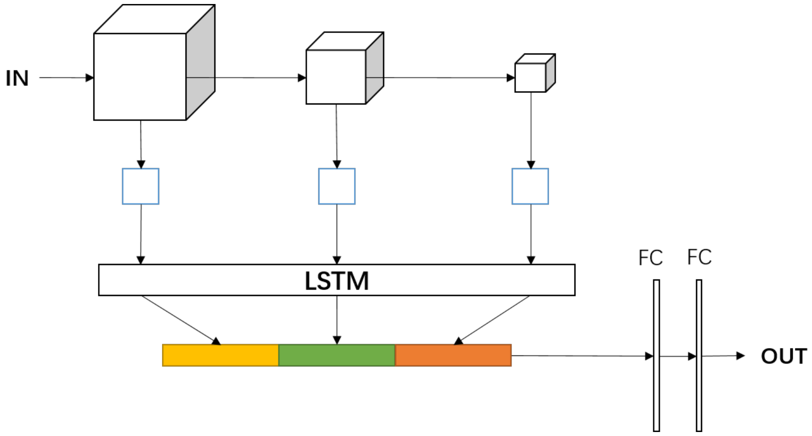

Fig. 8. Structure of CNN-RA-C

vector of the last block as a new vector, and apply it to the classifier. We denote this structure as CNN-RA-B. In Fig. 8, we concatenate the corresponding N output vectors together as the final feature, and employ it for classification. We denote the structure as CNN-RA-C.

To evaluate the performance of these two transformative network structures, we employ ResNet as the baseline and test on all three dataset. The evaluation results are shown in Table 3. CNN-RA improves the performance of ResNet on all three datasets. The combination of aggregated features and the original high-level features at least maintains the accuracy of ResNet, and shows obvious advantage over Cifar10. However, the results of ResNet-RA-B are worse than ResNet-RA over all three datasets. This is probably because the propagation through the original architecture is more straightforward than the aggregated structure and the aggregated features are suppressed during the training process and only play a weak role on the task. ResNet-RA-C concatenates output features of all steps in LSTM to further combine low-level to high-level information. The output of LSTM naturally relates to the information containing previous steps. The structure shows remarkable improvement on SVHN, and is even better than ResNet-RA. However, on Cifar10 and Cifar100, ResNet-RA-C just shows small advantage, and is even worse than ResNet-RA-B. This might because in ResNet-RA-C, the shallow features are even directly employed to do the classification, and they bring much redundant information and has bad influence on the network performance.

These two explorations show some different methods of combining our recurrent aggregated features with other information. In these experiments, all structures achieve large or small improvements across networks and datasets. On the whole, ResNet-RA shows better performance than ResNet-RA-B and ResNet-RA-C, which definitely proves the effectiveness of our recurrently aggregated features.

Table 3. Validation accuracies of ResNet, ResNet-RA, ResNet-RA-B and ResNet-RA-C over three datasets.

Network	Cifar10	SVHN	Cifar100
ResNet	93.74	98.00	75.87
ResNet-RA	94.78	98.19	77.85
ResNet-RA-B	94.42	98.01	75.91
ResNet-RA-C	93.85	98.31	75.89

6 Conclusion

In this paper, we propose a brand new method of convolutional layer aggregation method which is denoted as CNN-RA. Recurrent layer aggregation first splits a CNN into convolutional blocks and arranges their output features into a matrix. A LSTM is then employed to combine the low-level to high-level features and extract the sequential relationship between them. CNN-RA can be simply applied to any standard CNN without any modification on the original architectures. We evaluate our aggregation methods over Cifar10, SVHN and Cifar100 using VGG-1, VGG-2 and ResNet. The results show remarkable improvements with CNN-RA. Our proposed recurrent aggregation method is the first to put forward treating the convolutional layer features as a sequence and combining CNN and RNN parallelly together. The evaluation experiments verify the effectiveness of CNN-RA and show its great potential on utilizing the features in CNNs.

References

1. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision **115** (2015) 211–252
2. Zhang, L., Lin, L., Liang, X., He, K.: Is faster r-cnn doing well for pedestrian detection? In: European Conference on Computer Vision, Springer (2016) 443–457
3. Lin, G., Milan, A., Shen, C., Reid, I.: Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In: IEEE conference on computer vision and pattern recognition (CVPR). Volume 1. (2017) 3
4. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86** (1998) 2278–2324
5. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. (2012) 1097–1105
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 770–778
7. Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. Volume 1. (2017) 3
8. Yu, F., Wang, D., Darrell, T.: Deep layer aggregation. arXiv preprint arXiv:1707.06484 (2017)
9. Sutskever, I.: Training recurrent neural networks. University of Toronto, Ontario, Canada (2013)
10. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: Continual prediction with lstm. (1999)
11. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
12. Yang, Y., Zhong, Z., Shen, T., Lin, Z.: Convolutional neural networks with alternately updated clique. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 2413–2422
13. Su, B., Lu, S.: Accurate scene text recognition based on recurrent neural network. In: Asian Conference on Computer Vision, Springer (2014) 35–48
14. Graves, A., Mohamed, A.r., Hinton, G.: Speech recognition with deep recurrent neural networks. In: Acoustics, speech and signal processing (icassp), 2013 ieee international conference on, IEEE (2013) 6645–6649
15. Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., Baskurt, A.: Sequential deep learning for human action recognition. In: International Workshop on Human Behavior Understanding, Springer (2011) 29–39
16. Huang, Z., Xu, W., Yu, K.: Bidirectional lstm-crf models for sequence tagging. arXiv preprint arXiv:1508.01991 (2015)

17. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015) 3431–3440
18. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR. Volume 1. (2017) 4
19. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention, Springer (2015) 234–241
20. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10). (2010) 807–814
21. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Technical report, Citeseer (2009)
22. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: NIPS workshop on deep learning and unsupervised feature learning. Volume 2011. (2011) 5