Recurrent Convolutional Neural Network for Object Recognition

Ming Liang Xiaolin Hu
State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology (TNList)
Department of Computer Science and Technology
Center for Brain-Inspired Computing Research (CBICR)
Tsinghua University, Beijing 100084, China
liangm07@mails.tsinghua.edu.cn, xlhu@tsinghua.edu.cn

Abstract

In recent years, the convolutional neural network (CNN) has achieved great success in many computer vision tasks. Partially inspired by neuroscience, CNN shares many properties with the visual system of the brain. A prominent difference is that CNN is typically a feed-forward architecture while in the visual system recurrent connections are abundant. Inspired by this fact, we propose a recurrent CNN (RCNN) for object recognition by incorporating recurrent connections into each convolutional layer. Though the input is static, the activities of RCNN units evolve over time so that the activity of each unit is modulated by the activities of its neighboring units. This property enhances the ability of the model to integrate the context information, which is important for object recognition. Like other recurrent neural networks, unfolding the RCNN through time can result in an arbitrarily deep network with a fixed number of parameters. Furthermore, the unfolded network has multiple paths, which can facilitate the learning process. The model is tested on four benchmark object recognition datasets: CIFAR-10, CIFAR-100, MNIST and SVHN. With fewer trainable parameters, RCNN outperforms the state-of-the-art models on all of these datasets. Increasing the number of parameters leads to even better performance. These results demonstrate the advantage of the recurrent structure over purely feed-forward structure for object recognition.

1. Introduction

The past few years have witnessed the bloom of convolutional neural network (CNN) in computer vision. Over many benchmark datasets CNN has substantially advanced the state-of-the-art accuracies of object recognition [26, 50, 33, 5, 43]. For example, after training on 1.2 million images from ImageNet [8], CNN [26] has achieved better performance than handcraft features by a significant margin in classifying objects into 1000 categories. Furthermore, the pretrained CNN features on this dataset have been transfered to other datasets to achieve remarkable results [5, 43].

CNN is a type of artificial neural network, which originates from neuroscience dating back to the proposal of the first artificial neuron in 1943 [34]. In fact, CNN, as well as other hierarchical models including Neocognitron [13] and HMAX [38], is closely related to Hubel and Wiesel's findings about simple cells and complex cells in the primary visual cortex (V1)[23, 22]. All of these models have purely feed-forward architectures, which can be viewed as crude approximations of the biological neural network in the brain. Anatomical evidences have shown that recurrent connections ubiquitously exist in the neocortex, and recurrent synapses typically outnumber feed-forward and top-down (or feedback) synapses [6]. Due to the presence of recurrent and top-down synapses, object recognition is actually a dynamic process though the input is static. Specific functions of these synapses remain unclear, but it is generally believed that recurrent synapses play an important role in context modulation. The processing of visual signals is strongly modulated by their context [1]. Normally we do not perceive this effect without attention, but the effect gets prominent in perceptual illusions, *e.g.*, the famous Fraser spiral illusion [12]. Context modulation is also observed in the responses of individual neurons in the visual system. For instance, the response properties of V1 neurons can be altered in many ways by changing the context around their classical receptive fields (RFs) [42]. This phenomenon is suggested to be induced by recurrent synapses in V1 [7, 54].

The context is important for object recognition (Figure 1). A feed-forward model can only capture the context (*e.g.*, the face in Figure 1) in higher layers where units have larger RFs, but this information cannot modulate the activities of units in lower layers responsible for recognizing smaller ob-

Figure 1. Importance of context for object recognition. Without the context (face), it is hard to recognize the black curve in the middle area as a nose.

jects (*e.g.*, the nose in Figure 1). To utilize this information, one strategy is to use top-down (or feedback) connections to propagate it downwards [32], which is adopted in the convolutional deep belief networks (CDBN) [31]. In this study, we take a different strategy, that is, use recurrent connections within the same layer of deep learning models. It is expected that, equipped with context modulation ability, these lateral connections may boost the performance of deep learning models.

In the paper, we present a recurrent CNN for static object recognition. The architecture is illustrated in Figure 2, where both feed-forward and recurrent connections have local connectivity and shared weights among different locations. This architecture is very similar to the recurrent multilayer perceptron (RMLP) which is often used for dynamic control [11, 37] (Figure 2, middle). The main difference is that the full connections in RMLP are replaced by shared local connections, just as the difference between MLP [40] and CNN. For this reason, the proposed model is called the *recurrent convolutional neural network (RCNN)*.

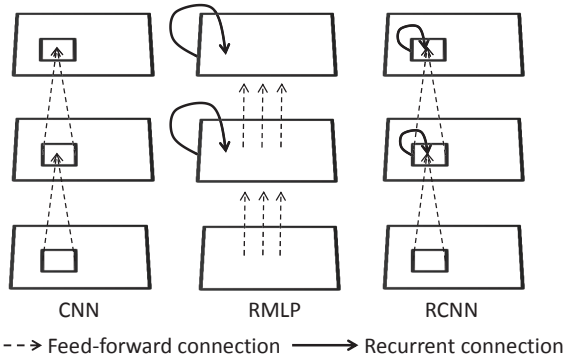

Figure 2. Illustration of the architectures of CNN, RMLP and RCNN. For each model two hidden layers are shown.

The proposed RCNN was tested on several benchmark object recognition datasets. With fewer parameters, RCNN achieved better results than the state-of-the-art CNNs over all of these datasets, which validates the advantage of RCNN over CNN. The remaining content is organized as follows. Section 2 reviews some related work. Section 3 describes the architecture of RCNN. Section 4 presents the experimental results and analysis. Finally, Section 5 concludes the paper.

2. Related work

2.1. Convolutional neural networks

Inspired by Hubel and Wiesel's breakthrough findings in cat [23][22], Fukushima [13] proposed a hierarchical model called Neocognitron, which consisted of stacked pairs of simple unit layer and complex unit layer. The first CNN was proposed by LeCun *et al*. [28][27]. Essentially CNN differs from the Neocognitron by incorporating the back-propagation (BP) algorithm for learning the receptive fields of simple units. Since its birth, CNN has been characterized by local connections, weight sharing and local pooling. The first two properties enable the model to discover local informative visual patterns with fewer adjustable parameters than MLP. The third property equips the network with some translation invariance. As suggested by Saxe *et al*. [41], the excellent performance of CNN can be largely attributed to these properties as certain structures with random weights could also achieve good results.

Over the past years, many techniques have have been developed for improving the performance of CNN. The rectified linear function [14] becomes the most commonly used activation function due to its resistance to the notorious gradient vanishing effect in the BP algorithm. Dropout [48] is an effective technique to prevent neural networks from over-fitting in training. To leverage the model averaging ability of dropout, Goodfellow *et al*. [17] used max pooling over feature channels as the activation function. To strengthen the nonlinearity of convolutional units, Lin *et al*. [33] proposed the network in network (NIN) structure, in which convolution was replaced by the local multilayer perceptron (MLP) [39] sliding over input feature maps. To prevent NIN from over-fitting, the fully connected layers were replaced by the global average pooling layer. Simonyan and Zisserman [44] used 3×3 convolutions to build very deep networks, considering that a stack of small filters have stronger nonlinearity than a large filter with the same amount of parameters. Szegedy *et al*. [50] proposed the multi-scale inception modules and built the GoogLeNet based on them. Small filters were also favored in this model. CNN is a computation-intensive model and is usually hard to run on CPU. The use of GPU has greatly facilitated the training and testing of CNN on large-scale datasets. The first successful GPU implementation of CNN refers to the AlexNet [26] which won the recognition competition in the ImageNet [8] Large Scale Visual Recognition Challenge (ILSVRC) 2012. Since then, most submissions to this annual competition were based on GPU implemented CNN.

2.2. Recurrent neural networks

Recurrent neural network (RNN) has a long history in the artificial neural network community [4, 21, 11, 37, 10, 24], but most successful applications refer to the modeling of sequential data such as handwriting recognition [18] and speech recognition[19]. A few studies about RNN for static visual signal processing are briefly reviewed below.

In [20] a multi-dimensional RNN (MDRNN) is proposed for off-line handwriting recognition. MDRNN has a directed structure in that it treats the image as 2D sequential data. Furthermore, MDRNN has a single hidden layer, which cannot produce the feature hierarchy as CNN.

In [2] a hierarchical RNN called the Neural Abstraction Pyramid (NAP) is proposed for image processing. NAP is a biology-inspired architecture with both vertical and lateral recurrent connectivity, through which the image interpretation is gradually refined to resolve visual ambiguities. In designing the structure, biological plausibility is stressed. For example, it employs excitatory and inhibitory units, which are not considered in most deep learning models. It is unclear whether these more biologically plausible techniques would make NAP more effective than state-of-the-art deep learning models. More importantly, though the general framework of NAP has recurrent and feedback connections, for object recognition only a feed-forward version was tested. The recurrent NAP was used for other tasks such as image reconstruction.

Besides NAP, top-down connections have been used in some other hierarchical models. Lee *et al*. [31] proposed CDBN for unsupervised feature learning. During inference the information in the top layer could be propagated to the bottom layer through the intermediate layers between them. Different from this layer-by-layer propagation idea, Pinheiro and Collobert [36] used extra connections from the top layer to the bottom layer of a CNN directly. This model was used for scene labeling. These models are different from RCNN where recurrent connections exist within the same layer, not between layers.

There is an interesting relationship between RCNN and some sparse coding models [15] where fixed-point updates are used in inference. The iterative optimization procedures implicitly define recurrent neural networks. Note that supervised learning techniques can be incorporated into the unsupervised learning framework of sparse coding models [3]. But these techniques have not made the sparse coding models competitive with CNN for object recognition.

Finally, our model is also related to the recursive neural network [46], in which a recursive layer is unfolded to a stack of layers with tied weights. Socher *et al*. [45] used a recursive neural network to perform scene parsing. Eigen *et al*. [9] studied the factors that influence the performance of CNN by employing a recursive convolutional neural network, which is equivalent to the time-unfolded version of RCNN but without feed-forward input to each unfolded layer.

3. RCNN Model

3.1. Recurrent convolutional layer

The key module of RCNN is the recurrent convolutional layer (RCL). The states of RCL units evolve over discrete time steps. For a unit located at (i, j) on the kth feature map in an RCL, its net input $z_{ijk}(t)$ at time step t is given by:

$$z_{ijk}(t) = (\mathbf{w}_k^f)^T \mathbf{u}^{(i,j)}(t) + (\mathbf{w}_k^r)^T \mathbf{x}^{(i,j)}(t-1) + b_k. \quad (1)$$

In the equation $\mathbf{u}^{(i,j)}(t)$ and $\mathbf{x}^{(i,j)}(t-1)$ denote the feed-forward and recurrent input, respectively, which are the vectorized patches centered at (i, j) of the feature maps in the previous and current layer, $\mathbf{w}_k^f$ and $\mathbf{w}_k^r$ denote the vectorized feed-forward weights and recurrent weights, respectively, and b_k is the bias. The first term in (1) is used in standard CNN and the second term is induced by the recurrent connections.

The activity or state of this unit is a function of its net input

$$x_{ijk}(t) = g(f(z_{ijk}(t))), \quad (2)$$

where f is the rectified linear activation function

$$f(z_{ijk}(t)) = \max(z_{ijk}(t), 0), \quad (3)$$

and g is the local response normalization (LRN) function [26]

$$g(f_{ijk}(t)) = \frac{f_{ijk}(t)}{\left(1 + \frac{\alpha}{N} \sum_{k'=\max(0,k-N/2)}^{\min(K,k+N/2)} (f_{ijk'})^2\right)^{\beta}} \quad (4)$$

where K is the total number of feature maps in the current layer. Note that in the denominator in (4) the sum runs over N feature maps at the same location (i, j) (usually $N < K$), and α and β are constants controlling the amplitude of normalization. In addition $f(z_{ijk}(t))$ has been abbreviated as $f_{ijk}(t)$. LRN mimics the lateral inhibition in the cortex, where different features compete for large responses. LRN is used in our model for preventing the states from exploding.

Equations (1) and (2) describe the dynamic behavior of the RCL. Unfolding this layer for T time steps results in a feed-forward subnetwork of depth $T + 1$. See the top left of Figure 3 for an example with $T = 3$. While the recurrent input evolves over iterations, the feed-forward input remains the same in all iterations. When $t = 0$ only the feed-forward input is present. The subnetwork has several paths from the input layer to the output layer. The longest path goes through all unfolded recurrent connections (therefore

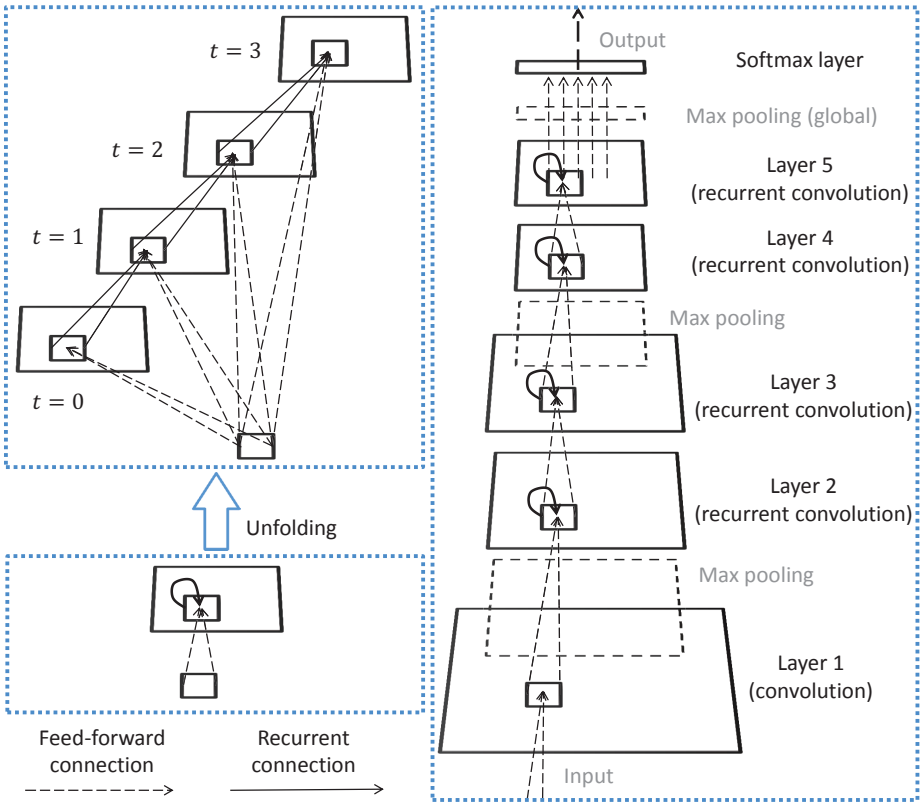

Figure 3. The overall architecture of RCNN. Left: An RCL is unfolded for $T = 3$ time steps, leading to a feed-forward subnetwork with the largest depth of 4 and the smallest depth of 1. At $t = 0$ only feed-forward computation takes place. Right: The RCNN used in this paper contains one convolutional layer, four RCLs, three max pooling layers and one softmax layer.

$length = T + 1$), while the shortest path goes through the feed-forward connection only (therefore $length = 1$). The effective RF of an RCL unit in the feature maps of the previous layer expands when the iteration number increases. If both input and recurrent filters in equation (1) have square shapes in each feature map, then the effective RF of an RCL unit is also square, whose side length is $(L_{rec}-1)T+L_{feed}$, where L_{feed} and L_{rec} denote the side lengths of the input and recurrent filters, respectively.

3.2. Overall architecture

RCNN contains a stack of RCLs, optionally interleaved with max pooling layers. See Figure 3 for the architecture used in this work. To save computation, layer 1 is the standard feed-forward convolutional layer without recurrent connections, followed by max pooling. On top of this, four RCLs are used with a max pooling layer in the middle. Between neighboring RCLs there are only feed-forward connections. Both pooling operations have stride 2 and size 3. The output of the fourth RCL follows a global max pooling layer, which outputs the maximum over every feature map, yielding a feature vector representing the image. This is different from the model in [26] where fully connected layers are used or the models in [33, 50] where global average pooling is used. Finally a softmax layer is used to classify the feature vectors to C categories whose output is given by

$$y_k = \frac{\exp\left(\mathbf{w}_k^T \mathbf{x}\right)}{\sum_{k'} \exp\left(\mathbf{w}_{k'}^T \mathbf{x}\right)} \qquad (k = 1, 2, ..., C) \qquad (5)$$

where y_k is the predicted probability belonging to the kth category, and $\mathbf{x}$ is the feature vector generated by the global max pooling. Training is performed by minimizing the cross-entropy loss function using the backpropagation throught time (BPTT) algorithm [52]. This is equivalent to using the standard BP algorithm on the time-unfolded network. The final gradient of a shared weight is the sum of its gradients over all time steps.

If we unfold the recurrent connections for T time steps, the model becomes a very deep feed-forward network with $4(T+1)+2$ parameterized layers, where $T+1$ is the depth of each RCL. But $4(T + 1) + 2$ is only the length of the longest path from the input layer to the output layer, and there are many other paths with different lengths. Among them the shortest path has length 6, which is the feed-forward path bypassing all recurrent connections.

3.3. Discussion

From the computational perspective, the recurrent connections in RCNN offer several advantages. First, they enable every unit to incorporate context information in an arbitrarily large region in the current layer. In fact, as the time steps increase, the state of every unit is influenced by other units in a larger and larger neighborhood in the current layer (equation (1)); as a consequence, the size of regions that the unit can "watch" in the input space also increases. In CNN, the size of the RFs of the units in the current layer is fixed, and "watching" a larger region is only possible for units in higher layers. But unfortunately the context seen by higher-level units cannot influence the states of the units in the current layer without top-down connections. Second, the recurrent connections increase the network depth while keep the number of adjustable parameters constant by weight sharing. This is consistent with the trend of modern CNN architecture: going deeper with relatively small number of parameters [33, 44, 50]. Note that simply increasing the depth of CNN by sharing weights between layers can result in the same depth and the same number parameters as RCNN, but such a model may not compete with RCNN in performance, as verified in our experiments (see Section 4.2.1). We attribute this fact to the difficulty in learning such a deep model. Then here comes the third advantage of RCNN — the time-unfolded RCNN is actually a CNN with multiple paths between the input layer to the output layer (Figure 3), which may facilitate the learning. On one hand, the existence of longer paths makes it possible for the model to learn highly complex features. On the other hand, the existence of shorter paths may help gradient backpropagation during training. Multi-path is also used in [50, 30], but there extra objective functions are used in hidden layers to alleviate the difficulty in training deep networks, which are not used in RCNN.

3.4. Implementation

Many hyper-parameters may affect the performance of RCNN such as the numbers of feature maps and the filter size in each layer. We did not explore the best configuration. Instead, we limited the search in a constrained hyper-parameter space. First, layers 1 to 5 were constrained to have the same number of feature maps K. As a consequence the model can be denoted by RCNN-K. For example, RCNN-96 indicates that each of the five layers has 96 feature maps. Second, the feed-forward filter size in layer 1 was 5×5, the feed-forward and recurrent filter sizes in layers 2 to 4 were all 3×3. So the total number of parameters of RCNN-K was approximately $72K^2$, which only accounted for the weights in RCLs because other layers had far fewer parameters.

The hyper-parameters of LRN in (4) were set as $\alpha = 0.001$, $\beta = 0.75$ and $N = K/8 + 1$. Dropout [48] was used after each RCL except layer 5, which was connected to the softmax layer. If the RCL was followed by a pooling layer, dropout was placed after pooling.

4. Experiments

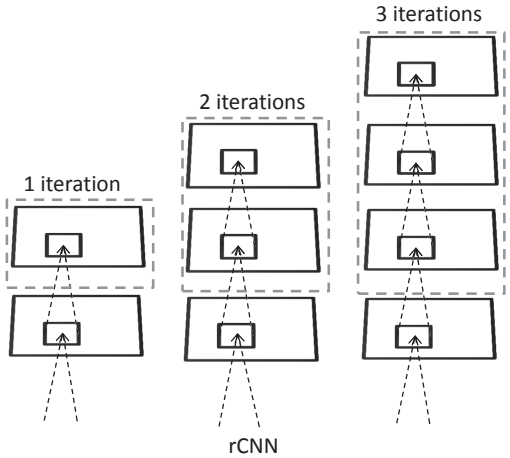

Figure 4. The subnetworks used to construct rCNNs. They are used to replace the RCLs in RCNN. The layers surrounded by the dotted box have tied weights. From left to right, the subnetworks correspond to the RCL with one, two and three iterations, respectively.

4.1. Overall settings

The RCNN was implemented using cuda-convnet2 [26] developed by Alex krizhevsky. Experiments were run on two GPUs with data parallelism. The models were evaluated on four benchmark object classification datasets, CIFAR-10 [25], CIFAR-100 [25], MNIST [29] and SVHN [35]. We found that a few iterations of the dynamic process of RCNN were able to produce excellent results. Except on CIFAR-10, where different number of iterations were compared, on the other three datasets, the iteration number was set to 3.

The training procedure followed [26]. The model was trained using the BPTT algorithm in combination with stochastic gradient descent. The initial learning rate was set heuristically and annealed according to a schedule predetermined on the validation set. When the accuracy stopped improving, the learning rate was decreased to its $1/10$. Three annealing steps were used so that the final learning rate was $1/1000$ of the initial value. The momentum for all datasets were fixed at 0.9. Weight decay was used as another regularizer besides dropout. All weights of the model were set to have the same decay term. Dropout probabilities and weight decay rate were tuned. For CIFAR-10, CIFAR100 and SVHN, the images were preprocessed by removing the per-pixel mean value calculated over the training set. For MNIST the raw images were used as input.

4.2. CIFAR-10

The CIFAR-10 dataset [26] consists of 60000 color images of 32×32 pixels in 10 classes. The total dataset was split into 50000 training images and 10000 testing images. The last 10000 training images were used for validation. After the hyper-parameters were determined, the model was trained from scratch on all 50000 training images.

4.2.1 Comparison with baseline models

Model	No. of Param.	Error (%)	
		Training	Testing
rCNN-96 (1 iter)	0.67 M	4.61	12.65
rCNN-96 (2 iters)	0.67 M	2.26	12.99
rCNN-96 (3 iters)	0.67 M	1.24	14.92
WCNN-128	0.60 M	3.45	9.98
RCNN-96 (1 iter)	0.67 M	4.99	9.95
RCNN-96 (2 iters)	0.67 M	3.58	9.63
RCNN-96 (3 iters)	0.67 M	3.06	9.31

Table 1. Comparison with the baseline models on CIFAR-10

We analyzed the properties of RCNN by comparing it with two baseline models. The first baseline model was constructed by removing the recurrent connections in RCNN, which becomes a conventional CNN. For fair comparison, we used more feature maps in each layer to make its number of parameters approximately the same as RCNN. To emphasize this point, the model was denoted by WCNN (wide CNN). Note that WCNN was similar to the VGG very deep model [44] as most layers used 3×3 filters. The second baseline model was constructed by removing the recurrent connections in each RCL of RCNN but adding a cascade of duplicated convolutional layers. This was called recursive CNN [9] or rCNN for short. The cascade of duplicated convolutional layers can be understood as the time-unfolded version of RCL starting at $t = 1$ without feed-forward input (Figure 4). In other words in iterations of RCNN, the feed-forward input (the first term in equation (1)) was always there, but in iterations of rCNN it was absent. Please compare the top-left of Figure 3 and Figure 4. Note that rCNN had exactly the same number of parameters as RCNN.

RCNN-96 was used for comparison. Because cuda-convnet2 [26] requires the number of filters to be multiples of 16, we selected WCNN-128 (0.6 million parameters) which has the closest complexity with RCNN-96 and rCNN-96 (0.67 million parameters). For both RCNN and rCNN, 1, 2 and 3 iterations were tested. Table 1 shows the comparison results.

WCNN-128 achieved much better performance than rCNN in terms of testing accuracy. In fact, WCNN-128 already surpassed most of the models shown in Table 2, which validates the effectiveness of extensive use of 3×3 filters [44]. However, WCNN was significantly outperformed by RCNN with a few iterations.

More iterations in RCNN led to both lower training error and lower testing error, and more iterations in rCNN led to lower training error but higher testing error (Table 1). In fact, the training errors of the three rCNNs were even lower than the corresponding RCNNs. Clearly, overfitting has occurred in rCNN, a phenomenon also reported in [9]. The comparison indicates that the multi-path structure of RCNN is less prone to overfitting than the chain structure of rCNN.

4.2.2 Comparison with state-of-the-art models

Model	No. of Param.	Testing Error (%)
Without Data Augmentation		
Maxout [17]	> 5 M	11.68
Prob maxout [47]	> 5 M	11.35
NIN [33]	0.97 M	10.41
DSN [30]	0.97 M	9.69
RCNN-96	0.67 M	**9.31**
RCNN-128	1.19 M	**8.98**
RCNN-160	1.86 M	**8.69**
RCNN-96 (no dropout)	0.67 M	13.56
NIN (no dropout) [33]	0.97 M	14.51
With Data Augmentation		
Prob maxout [47]	> 5 M	9.39
Maxout [17]	> 5 M	9.38
DropConnect (12 nets) [51]	—	9.32
NIN [33]	0.97 M	8.81
DSN [30]	0.97 M	7.97
RCNN-96	0.67 M	**7.37**
RCNN-128	1.19 M	**7.24**
RCNN-160	1.86 M	**7.09**

Table 2. Comparison with existing models on CIFAR-10

We then compared RCNN with the state-of-the-art models on this dataset. Three models with different K's were tested: RCNN-96, RCNN-128 and RCNN-160. The number of iterations was set to 3. All of them outperformed existing models, and the performance was steadily improved with more features maps (Table 2). Among the existing models for comparison NIN and DSN had the fewest parameters, about 1 million. RCNN-96 had even fewer parameters, 0.67 million, but achieved better results. Note that the maxout networks, NIN and DSN pre-processed the data with global contrast normalization and ZCA whitening, while we simply subtracted the per-pixel mean value from the data.

Dropout played an important role for RCNN to achieve these remarkable results. Without dropout, the error rate of RCNN-96 was 13.56%, much higher than 9.31% with dropout. But this error rate was lower than that of NIN without dropout (14.51%).

We also tested RCNN when data was augmented with translations and horizontal reflections as in [17]. In the training phase, crops of 24×24 pixels were randomly extracted from the original image and randomly horizontally reflected. In the testing phase, nine crops were uniformly extracted from the image so that the interval between neighboring crops was four pixels. All of the nine outputs were averaged to give the final output. RCNN-96 achieved significantly better result than the state-of-the-art models using much fewer parameters. And again, simply increasing K led to even better results. See Table 2 for details.

4.3. CIFAR-100

Model	**No. of Param.**	**Testing Error** (%)
Maxout [17]	> 5 M	38.57
Prob maxout [47]	> 5 M	38.14
Tree based priors [49]	—	36.85
NIN [33]	0.98 M	35.68
DSN [30]	0.98 M	34.57
RCNN-96	0.68 M	**34.18**
RCNN-128	1.20 M	**32.59**
RCNN-160	1.87 M	**31.75**

Table 3. Comparison with existing models on CIFAR-100

CIFAR-100 has 100 classes of images in the same format as CIFAR-10. The two datasets have the same size, so the number of images in each CIFAR-100 class is only $1/10$ of that in CIFAR-10. We tested three RCNNs without data augmentation. The same setting as in CIFAR-10 was adopted here without further tuning the hyper-parameters. Again RCNN-96 outperformed the state-of-the-art models with fewer parameters, and the performance kept improving by increasing K. Table 3 shows the comparison result.

4.4. MNIST

Model	**No. of Param.**	**Testing Error** (%)
NIN [33]	0.35 M	0.47
Maxout [17]	0.42 M	0.45
DSN [30]	0.35 M	0.39
RCNN-32	0.08 M	0.42
RCNN-64	0.30 M	**0.32**
RCNN-96	0.67 M	**0.31**

Table 4. Comparison with existing models on MNIST

MNIST [29] is one of the most well known datasets in the machine learning community. It consists of hand written digits of 0 to 9. There are 60000 training images and 10000 testing images. The images are in gray scale with size 28×28 pixels. We compared RCNN with other models without data augmentation and model averaging techniques. This benchmark is much easier than the two CIFAR datasets, and we preferred smaller Ks. The results are shown in Table 4. RCNN-64 outperformed other models using only 0.30 million parameters. In contrast, 0.42 million parameters were used in Maxout networks and 0.35 million parameters were used in NIN and DSN. No preprocessing was used by RCNN while the other models used global contrast normalization and ZCA whitening.

4.5. SVHN

Model	**No. of Param.**	**Testing Error (%)**
Without Data Augmentation		
Maxout [17]	> 5 M	2.47
Prob maxout [47]	> 5 M	2.39
NIN [33]	1.98 M	2.35
DSN [30]	1.98 M	1.92
RCNN-128	1.19 M	**1.87**
RCNN-160	1.86 M	**1.80**
RCNN-192	2.67 M	**1.77**
With Data Augmentation		
Multi-digit number recognition [16]	> 5 M	2.16
DropConnect (5 nets) [51]	—	1.94

Table 5. Comparison with existing models on SVHN

SVHN consists of real-world house numbers collected from Google Street View images. The dataset has two formats and we used the second format. Totally there are 630,420 color images of size 32×32 pixels, which are split into three sets. The training set contains 73,257 digits, the testing set contains 26,032 digits and an extra set contains 531,131 additional less difficult samples. Multiple digits may coexist in an image, and the task is to classify the center digit. We followed the training procedure described in [17]. 400 samples per class were randomly selected from the training set, and 200 samples per class were randomly selected from the extra set. These samples composed the validation set. The other images in the training set and extra set composed the training set. The validation set was only used for tuning hyper-parameters and not used in training.

SVHN is much more difficult than MNIST due to large variations of color and brightness. Local contrast normalization was suggested to be an effective preprocessing step [53] and was adopted by many models including the maxout networks, NIN, DSN and DropConnect. We only subtracted the mean value from each pixel. With this simple preprocessing step, RCNN-128 outperformed the state-of-the-art models without data augmentation and two models with data augmentation (note that DropConnect used model averaging of five networks). See Table 5 for details. RCNN-128 had much fewer parameters than NIN (1.19 million *versus* 1.98 million), and increasing K kept improving the accuracy.

5. Conclusion

Inspired by the fact of abundant recurrent synapses in the brain, we proposed a recurrent convolutional neural network (RCNN) for (static) object recognition. The basic idea was to add recurrent connections within every convolutional layer of the feed-forward CNN. This structure enabled the units to be modulated by other units in the same layer, which enhanced the capability of the CNN to capture statistical regularities in the context of the object. The recurrent connections increased the depth of the original CNN while kept the number of parameters constant by weight sharing between layers. Experimental results demonstrated the advantage of RCNN over CNN for object recognition. Over four benchmark datasets, with fewer parameters RCNN outperformed the state-of-the-art models. Increasing the number of parameters led to even better performance. This work shows that it is possible to boost the performance of CNN by incorporating more facts of the brain. It would be interesting to see other fact of the brain to be integrated into deep learning models in future.

Acknowledgements

We are grateful to the anonymous reviewers for their valuable comments. This work was supported in part by the National Basic Research Program (973 Program) of China under Grant 2012CB316301, in part by the National Natural Science Foundation of China under Grant 61273023, Grant 91420201, and Grant 61332007, in part by the Natural Science Foundation of Beijing under Grant 4132046, and in part by the Tsinghua University Initiative Scientific Research Program under Grant 20141080934.

References

[1] T. D. Albright and G. R. Stoner. Contextual influences on visual processing. *Annual review of neuroscience*, 25(1):339–379, 2002.

[2] S. Behnke. *Hierarchical Neural Networks for Image Interpretation*, volume 2766 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[3] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2559–2566, 2010.

[4] G. A. Carpenter and S. Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer vision, graphics, and image processing*, 37(1):54–115, 1987.

[5] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.

[6] P. Dayan and L. F. Abbott. *Theoretical neuroscience*. Cambridge, MA: MIT Press, 2001.

[7] G. Deco and T. S. Lee. The role of early visual cortex in visual integration: a neural model of recurrent interaction. *European Journal of Neuroscience*, 20(4):1089–1100, 2004.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[9] D. Eigen, J. Rolfe, R. Fergus, and Y. LeCun. Understanding deep architectures using a recursive convolutional network. In *International Conference on Learning Representations (ICLR)*, 2014.

[10] J. L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[11] B. Fernandez, A. G. Parlos, and W. Tsai. Nonlinear dynamic system identification using artificial neural networks (anns). In *International Joint Conference on Neural Networks (IJCNN)*, pages 133–141, 1990.

[12] J. Fraser. A new visual illusion of direction. *British Journal of Psychiatry*, 2:307–320, 1908.

[13] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[14] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, volume 15, pages 315–323, 2011.

[15] I. Goodfellow, A. Couville, and Y. Bengio. Large-scale feature learning with spike-and-slab sparse coding. In *International Conference on Machine Learning (ICML)*. 2012.

[16] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[17] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio. Maxout networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 1319–1327, 2013.

[18] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 31(5):855–868, 2009.

[19] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6645–6649, 2013.

[20] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 545–552, 2009.

[21] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[22] D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574, 1959.

[23] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106, 1962.

[24] M. I. Jordan. attractor dynamics and parallelism in a connectionist sequential machine. pages 112–127. IEEE Press, 1990.

[25] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Computer Science Department, University of Toronto, Tech. Rep*, 2009.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[27] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems (NIPS)*, pages 396–404, 1990.

[28] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[30] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *Advances in neural information processing systems (NIPS), Deep Learning and Representation Learning Workshop*, 2014.

[31] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, pages 609–616, 2009.

[32] T. S. Lee and D. Mumford. Hierarchical bayesian inference in the visual cortex. *Journal of the Optical Society of America A*, 20(7):1434–1448, 2003.

[33] M. Lin, Q. Chen, and S. Yan. Network in network. In *International Conference on Learning Representations (ICLR)*, 2014.

[34] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[35] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on Deep Learning and Unsupervised Feature Learning*, volume 2011, page 4, 2011.

[36] P. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene labeling. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 82–90, 2014.

[37] G. V. Puskorius and L. A. Feldkamp. Neurocontrol of nonlinear dynamical systems with kalman filter trained recurrent networks. *IEEE Transactions on Neural Networks*, 5(2):279–297, 1994.

[38] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2:1019–1025, 1999.

[39] F. Rosenblatt. *Principles of neurodynamics; perceptrons and the theory of brain mechanisms*. Spartan Books Washington, 1962.

[40] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, 1986.

[41] A. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Y. Ng. On random weights and unsupervised feature learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 1089–1096, 2011.

[42] P. Series, J. Lorenceau, and Y. Frégnac. The silent surround of V1 receptive fields: theory and experiments. *Journal of physiology-Paris*, 97(4):453–474, 2003.

[43] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2014.

[44] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[45] R. Socher, C. C. Lin, C. Manning, and A. Y. Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 129–136, 2011.

[46] R. Socher, C. D. Manning, and A. Y. Ng. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Advances in neural information processing systems (NIPS), Deep Learning and Representation Learning Workshop*, pages 1–9, 2010.

[47] J. T. Springenberg and M. Riedmiller. Improving deep neural networks with probabilistic maxout units. In *International Conference on Learning Representations (ICLR)*, 2014.

[48] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[49] N. Srivastava and R. Salakhutdinov. Discriminative transfer learning with tree-based priors. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2094–2102, 2013.

[50] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[51] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 1058–1066, 2013.

[52] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[53] M. D. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In *International Conference on Learning Representations (ICLR)*, 2013.

[54] M. Zhu and C. Rozell. Visual nonclassical receptive field effects emerge from sparse coding in a dynamical system. *PLOS Computational Biology*, 9(8):e1003191, 2013.