Recurrent Neural Networks (RNN) are a class of artificial neural network which became more popular in the recent years. The RNN is a special network, which has unlike feedforward networks recurrent connections. The major benefit is that with these connections the network is able to refer to last states and can therefore process arbitrary sequences of input. RNN are a very huge topic and are here introduced very shortly. This article specializes on the combination of RNN with CNN.

Introduction

The basic idea

The basic difference between a feed forward neuron and a recurrent neuron is shown in figure 1. The feed forward neuron has only connections from his input to his output. In the example of figure 1 the neuron has two weights. The recurrent neuron instead has also a connection from his output again to his input and therefore it has in this example three weights. This third extra connection is called feed-back connection and with that the activation can flow round in a loop. When many feed forward and recurrent neurons are connected, they form a recurrent neural network (5).
The RNN offers two major advantages:

1. **Store Information**

   The recurrent network can use the feedback connection to store information over time in form of activations (11). This ability is significant for many applications. In (5) the recurrent networks are described that they have some form of memory.

2. **Learn Sequential Data**

   The RNN can handle sequential data of arbitrary length. What this exactly means is explained in figure 2: On the left the default feed forward network is shown which can just compute one fixed size input to one fixed size output. With the recurrent approach also one to many, many to one and many to many inputs to outputs are possible. One example for one to many networks is that you label a image with a sentence. The many to one approach could handle a sequence of images (for example a video) and produce one sentence for it and finally the many to many approach can be used for language translations. Other use cases for the many to many approach could be to label each image of a video sequence (100).
Figure 2: The different kinds of sequential data, which can be handled by a recurrent neural net (100)

**Training of Recurrent Nets**

The training of almost all networks is done by back-propagation, but with the recurrent connection it has to be adapted. This is simply done by unfolding the net like it is shown in figure 3. It is shown that the network consists of one recurrent layer and one feed forward layer. The network can be unfolded to \( k \) instances of \( f \). In the example in figure 3 the network is unfolded with a depth of \( k = 3 \). After unfolding, the network can be trained in the same way as a feed forward network with Backpropagation, except that each epoch has to run through each unfolded layer. The algorithm for recurrent nets is then called Backpropagation through time (BPTT).

Figure 3: Unfolding of three time steps (102)

**History**
Recurrent Neural Networks (RNN) have a long history and were already developed during the 1980s. The Hopfield Network, which was introduced in 1982 by J.J. Hopfield, can be considered as one of the first network with recurrent connections (10). In the following years learning algorithms for fully connected neural networks were mentioned in 1989 (9) and the famous Elman network was introduced in 1990 (11). The Elman network was inspired by the architecture used by Jordan, therefore they are often mentioned together as Elman and Jordan networks. The historical architecture used by Jordan is shown in figure 4. Schmidhuber discovered in 1992 the vanishing gradient problem and therefore improved with Hochreiter the RNN to the Long Short-Term Memory (LSTM) in 1997 (8). The LSTM are more stable to the vanishing gradient problem and can better handle long-term dependencies. Furthermore, the Bidirectional Recurrent Neural Networks (BRNN) was a further big contribution in 1997 (13). In (21) a hierarchical RNN for image processing is proposed. This biology-inspired RNN is called Neural Abstraction Pyramid (NAP) and has both vertical and lateral recurrent connections. After that no major improvement happened a long time until in 2014 the Gated Recurrent Neural Networks (GRU) were introduced, which are kind of similar to the LSTM (21). Over the last few years several people tried to combine RNN with CNN and called them sometimes RCNN. (last paragraph Combination of Recurrent and Convolutional Neural Networks)

![Historical Jordan network (1986). Connections from output to state unit are one-for-one. This network was used by Elman in his Finding Structure in Time paper in 1990. (11)](image)

**Why Recurrent Neural Networks?**

The recurrent connections often offer advantages. They make every unit to use their context information and especially in image recognition tasks this is very helpful. As the time steps increase, the unit gets influenced by larger and larger neighborhood. With that information recurrent networks can watch large regions in the input space. In CNN this ability is limited to units in higher layers. Furthermore the recurrent connections increase the network depth while they keep the number of parameters low by weight sharing. Reducing the parameters is also a modern trend of CNN architectures:

"... going deeper with relatively small number of parameters ... " (6)
Additionally the recurrent connections yield to an ability of handling sequential data. This ability is very useful for many tasks (refer Applications). As last point recurrent connections of neurons are biological inspired and are used for many tasks in the brain. Therefore using such connections can enhance artificial networks and bring interesting behaviors. (15) The last big advantage is that RNN offer some kind of memory, which can be used in many applications.

Applications

The ability to store information is significant for applications like speech processing, non-Markovian control and music composition (8). In addition RNN are used successfully for sequential data such as handwriting recognition and speech recognition. The big advantage in comparison to feed forward networks is, that RNN can handle sequential data as described in the paragraph before. In (2) a single RNN is proposed for sequence labeling. Most successful applications of RNN refer to tasks like handwriting recognition and speech recognition (6). They are also used in (16) for Clinical decision support systems. They used a network based on the Jordan/Elman neural network. Furthermore in (17) a recurrent fuzzy neural network for control of dynamic systems is proposed. Newer application which use combinations of RNN with CNN are for scene labeling and object detection (last paragraph).

Improvements

Long Short Term Memory (LSTM)

How does LSTM improve a Recurrent Neuron?

One major drawback of RNNs is that the range of contextual information is limited and the Back-Propagation through time does not work properly. This is noticeable in either vanishing or exploding outputs of the network. In literature this problem is called vanishing gradient problem or exploding gradient. When the network is learning to bridge long time lags, it takes a huge amount of time or does not work at all, because of the vanishing gradient problem. The exploding gradient leads to oscillating weights, which also reduces the quality of the network. In practice this means when a recurrent network is learning to store information over extend time intervals, it takes a very long time, due to insufficient, decaying error back flow. To address exactly these problems, Hochreiter and Schmidhuber are introducing in (8) their

... novel, efficient, gradient-based method called “Long Short-Term Memory” (LSTM). (8)

The LSTM is designed to overcome the error back flow problems through carousels in their special units. This is all done with still a low computational complexity of $O(1)$ and additionally the LSTM improves the RNN with the ability to bridge time intervals.

How does the LSTM work?

Each LSTM block consists of a forget gate, input gate and an output gate. In figure 5 on the bottom a basic LSTM cell with a step wise explanation of the gates is shown and on the top an other illustration of the cell connected into a network is shown. In the first step the forget gate looks at $h_{t-1}$ and $x_t$ to compute the output $f_t$, which is a number between 0 and 1. This is multiplied by the cell state $C_{t-1}$ and yield the cell to either forget everything or keep the information. For example a value of 0.5 means that the cell forgets 50% of its information. In the next step the input gate is computing the update for the cell by first multiplying the outputs $i_t$ and $\hat{C}_t$ and then adding this output to the input $C_{t-1} * f_t$, which was computed in the step before. Finally the output value has to be computed, which is done by multiplying $o_t$ with the tanh of the result of the previous step, which yields to: $h_t = o_t * \tanh(C_t)$ and $o_t = \sigma(*) (W_o [h_{t-1}, x_t] + b_o)$. The formulas are also shown in figure 5 and are displayed in LSTM (10).

Today many people use the LSTM instead of the basic RNN and they work tremendously well on a large variety of problems. Most remarkable results are achieved with LSTM instead of RNN and many authors talk about RNN, which are using the LSTM cells.
Figure 5: On the top: Different illustration of the LSTM. Sigmoid and tanh functions are shown (Based on source 102).

On the bottom: LSTM memory block. The steps of the cell. Forget, Input and Output gate with formula and active region (Based on source 102).

**Connectionist Temporal Classification (CTC)**
RNNs are limited in detecting cursive handwriting, where segmentation is difficult to determine. Therefore, Connectionist temporal classification is added as output layer for sequence labeling tasks. The main different step to RNN is that the network output gets transformed into a conditional probability distribution over, for example, label sequences. Then the most probable labeling for a given input sequence is chosen. The key benefits of CTC are that it does not explicitly model dependencies between labels and it obviates the need for segmented data. Furthermore, it allows to train directly for sequence labeling (18).

**Gated Recurrent Unit (GRU)**

The Gated Recurrent Unit was introduced in 2014 and is similar to the LSTM. It uses also the gating mechanism and is designed to adaptively reset or update its memory content. The GRU uses a reset and an update gate, which both can be compared with the forget and the input gate of the LSTM. Differently to the LSTM, the GRU fully exposes its memory at each time step and has no separate memory cells. The output balances between its last state and the new state. In (20) is shown that the LSTM and GRU outperform the traditional tanh-unit. However, the paper could not find a big performance difference between the LSTM and GRU (19).

**Bidirectional Recurrent Neural Networks (BRNN or BLSTM)**

In many cases it is useful to have access to both, the past and the future context. Therefore, Bidirectional Recurrent Neural Networks (BRNN) were introduced in 1997 by Schuster and Paliwal. In these networks the input space is increased and they have forward and backward connections. For more details, refer to (21).

**Combination of Recurrent and Convolutional Neural Networks**

Recurrent and Convolutional Neural Networks can be combined in different ways. In some paper Recurrent Convolutional Neural Networks are proposed. There is a little confusion about these networks and especially the abbreviation RCNN. This abbreviation refers in some papers to Region Based CNN (7), in others to Recursive CNN (3) and in some to Recurrent CNN (6). Furthermore not all described Recurrent CNN have the same architecture. In the following, two approaches are described in more detail. The first approach is described in the paper of Andrej Karpathy and Li Fei-Fei: They connect a CNN and RNN in series and use this for labeling a scene with a whole sentence (14). The second approach from Ming Liang and Xiaolin Hu mixes a CNN with a RNN and use this architecture for better object detection (6).

**CNN and afterwards RNN**

**General Structure**

The alignment model described in the paper is a CNN over image region combined with a bidirectional RNN and afterwards a Multimodal RNN architecture, which uses the input of the previous net. This multimodal RNN can finally generate novel descriptions of image regions. In conclusion two single models are combined to one more powerful model, which is used to label images with sentences. To make the architecture more clear the two models are shown in figure 6. The first modal is represented as the left VGGNET and the second module is shown on the bottom right as the RNN.

**How does the net work in detail?**
The proposed model is trained with a set of images and their corresponding sentence descriptions. It is assumed, that the sentences written by people refer to a particular but unknown region of the image. The first model aligns sentence snippets to the visual image regions. Afterwards the second multimodal RNN gets trained with the output of the first and learn how to generate sentences. The CNN has to learn how to align visual and language data. Therefore the net uses a method described by Girshick et al. to detect objects in every image with a CNN, which is pre-trained on ImageNet. This pre-trained network is very similar to the VGGNET with the only difference, that they cut the last two fully connected layers. Karpathy and Fei-Fei propose a BRNN, that is used to represent sentences. Finally after aligning the data, the output of the first model is fed to the Multimodal Recurrent Neural Network. This Network has a typical hidden layer of 512 neurons. It is shown in figure 6 that the input of the next recurrent layer is always the output of the layer before. The network is trained to combine a word \( x \) and the previous context \( h_{t-1} \) to predict a new word \( y \). For example, figure 6 shows that the first recurrent neuron outputs “straw” and therefore the next gets “straw” as input. With this input the neuron can compute the next word referring to the last. With this technique this kind of RCNN is able to create a whole meaningful sentence to describe an arbitrary image (14).

![VGGNET](image)

Figure 6: This image shows the architecture of a RCNN. In this case first a default CNN is used and after that a RNN is used to label the image with a sentence. (Based on source 14)

Examples of the output

In general the network generates very accurate and sensible description of images. Examples of generated images with text description are shown in figure 7. In these examples the network works pretty well except for the last two where the "wakeboard" and "two young girls" are considered as wrong. Surprisingly, the first description of the "mans in black shirt is playing guitar" does not appear in the training set. But "man in black shirt" has 20 occurrences and "is playing guitar" has 60. Therefore the network really learns how to combine these and generates a meaningful result. Although these results look very impressive, there are some limitations of the network. For example the model can only handle one specific array of pixels with fixed resolution. Furthermore this concept is based on two separate networks. Going directly from an image sentence to the region-level annotations of a single network remains an open problem (14).
Mixed CNN and RNN

Architecture

In a mixed CNN and RNN architecture the positive features of a RNN are used to improve the CNN. Liang and Hu are describing an architecture for object detection in (6) and in (2) a similar architecture for scene labeling is proposed. In these papers the combined network is called RCNN. The following quote describe what their main idea is:

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. (6)

The key module of this RCNN are the recurrent convolution layers (RCL), which introduce recurrent connection into a convolution layer. With these connections the network can evolve over time though the input is static and each unit is influenced by its neighboring units. This property integrates the context information of an image, which is important for object detection. The importance of context information is shown in figure 8. In this figure it is very hard to recognize the mouth or the nose without the context (6).

Figure 8: Describe the importance of context information to recognize the nose or the mouth. (6)
The network is trained with BPTT and therefore with the same unfolding algorithm described in the first paragraph (Introduction). The unfolding facilitates the learning process and sharing weights reduce the parameters. In figure 9 on the left the unfolding for three time steps of the recurrent connection is shown and on the right the architecture of the whole used RCNN. The network consists of first a convolutional layer to save computations, which is followed by a max pooling layer. On top of that two RCL, one max pooling and then again two RCL layer are used. Finally one global max pooling and a softmax layer are used. The pooling operation have stride two and size three. The global max pooling layer outputs the maximum over every feature map, yielding to a feature vector that represents the image.

Figure 9: Mixed CNN and RNN architecture. On the left the a RCL is unfolded for three time steps, leading to a feed-forward network with largest depth of four and smallest depth of one. On the right the RCNN used in paper (6) is shown with one convolutional layer, four RCL, three max pooling and one softmax layer. (Source 6)

Results of the network

The authors show in their paper that the recurrent connections perform better as a CNN with the same number of layers. For example, if the RCL in the RCNN uses three time steps for unfolding, they added three feed forward layers to the CNN and compared these networks. This comparison indicates that the multi-path structure of the RCL is less prone to overfitting and performs better than the extended CNN. In their next experiment they compared this RCNN with state-of-the-art models, which is shown in figure 6. The layers one to five (Figure 4) are constrained to have the same number of feature maps K. Thereby RCNN-K denotes a network with K feature maps in layer one to five and RCNN-96 has for example 96 feature maps. Table 1 compares some results on the CIFAR-10 dataset. This dataset consists of 60000 color images of 32x32 pixels in ten classes. For the RCNN 50000 images were used for training and 10000 images were used for testing. The last 10000 images of the training set were used for validation of the net. The RCNN has still remarkable results compared to many other nets. It has with a very low number of parameter very good results and the RCNN-160 reduces the testing error to 7.09%. The table shows state-of-the-art networks and compared to other networks the RCNN performs pretty good but there are some better performing nets. For example the Deep Residual Network which is also introduced in the wiki. Currently the best performing net on the CIFAR-10 dataset is the Fraction Max Pooling (4), which achieves a testing error of 3.47%.
<table>
<thead>
<tr>
<th>Model</th>
<th>Number of Parameter</th>
<th>Testing Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Without Data Augmentation</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maxout</td>
<td>&gt; 5 M</td>
<td>11.68</td>
</tr>
<tr>
<td>Prob maxout</td>
<td>&gt; 5 M</td>
<td>11.35</td>
</tr>
<tr>
<td>NIN</td>
<td>0.97 M</td>
<td>10.41</td>
</tr>
<tr>
<td>RCNN-96</td>
<td>0.67 M</td>
<td>9.31</td>
</tr>
<tr>
<td>RCNN-128</td>
<td>1.19 M</td>
<td>8.98</td>
</tr>
<tr>
<td>RCNN-160</td>
<td>1.86 M</td>
<td>8.69</td>
</tr>
<tr>
<td>RCNN-96 (no Dropout)</td>
<td>0.67 M</td>
<td>13.56</td>
</tr>
<tr>
<td><strong>With Data Augmentation</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maxout</td>
<td>&gt; 5 M</td>
<td>9.39</td>
</tr>
<tr>
<td>Prob maxout</td>
<td>&gt; 5 M</td>
<td>9.38</td>
</tr>
<tr>
<td>NIN</td>
<td>0.97 M</td>
<td>8.81</td>
</tr>
<tr>
<td>RCNN-96</td>
<td>0.67 M</td>
<td>7.37</td>
</tr>
<tr>
<td>RCNN-128</td>
<td>1.19 M</td>
<td>7.24</td>
</tr>
<tr>
<td>RCNN-160</td>
<td>1.86 M</td>
<td>7.09</td>
</tr>
<tr>
<td>Deep Residual Networks</td>
<td>1.7 M</td>
<td>6.43</td>
</tr>
<tr>
<td>Fraction Max Pooling (4)</td>
<td>-</td>
<td>3.47</td>
</tr>
</tbody>
</table>

Table 1: Results of the Recurrent CNN compared to other state of the art solutions, (based on source 6 and added some relevant networks)

**Conclusion**

Recurrent Networks are very exciting and have already a very long history. In this history there researchers were able to get a good understanding and feeling about the recurrent network. The fact that it is biological inspired is very promising for getting better performance out of RNN. Furthermore the basic idea of the RNN evolved over the time and many remarkable contributions were made. For example the LSTM, which enhances many properties of the basic RNN. In future it can be assumed that the combination of RNN with other networks, especially the CNN, will be continued. The improvement and the ability to handle sequential data enhance the CNN a lot and brings new unexplored behavior. This is an exciting and promising area of artificial intelligence.

**Literature**

1. A Field Guide to Dynamical Recurrent Networks, 2001 Book
2. Convolutional Neural Networks with Intra-layer Recurrent Connections for Scene Labeling, 2015
3. Understanding deep architectures using a recursive convolutional network, 2013
4. Fractional Max-Pooling (2015, Benjamin Graham)
5. Recurrent Neural Network Architectures (2015, John A. Bullinaria)
6. Recurrent Convolutional Neural Network for Object Recognition (2015, Ming Liang, Xiaolin Hu)
(8) **LONG SHORT-TERM MEMORY** (1997, Sepp Hochreiter, Jürgen Schmidhuber)

(9) **A Learning Algorithm for Continually Running Fully Recurrent Neural Networks** (1989, Ronald J. Williams)

(10) Neurons with graded response have collective computational properties like those of two-state neuron (1884, J.J. Hopfield)

(11) **Finding Structure in Time** (1990, Jeffrey L. Elman)

(13) **Bidirectional Recurrent Neural Networks** Schuster (1997, M. Schuster)

(14) **Deep Visual-Semantic Alignments for Generating Image Descriptions**, (2015, Andrej Karpathy, Li Fei-Fei)

(15) **Biologically Inspired Modular Neural Networks**, (2000)

(16) **Clinical decision support system based on Jordan/Elman neural networks** (2011, P. A. Kharat, S.V. Dudul)

(17) **Identification and Control of Dynamic Systems Using Recurrent Fuzzy Neural Networks** (2002, Ching-Hung Lee, Ching-Cheng Teng)

(18) **Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks**, (2006, Alex Grave, Santiago Fernandez, Faustino Gomez, Jürgen Schmidhuber)

(19) **Gated Feedback Recurrent Neural Networks**, (2015, J Chung)

(20) **Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling** (2014, Chung et al.)


**Weblinks**

(100) [http://karpathy.github.io/2015/05/21/rnn-effectiveness/](http://karpathy.github.io/2015/05/21/rnn-effectiveness/)

(101) [http://cs231n.stanford.edu/](http://cs231n.stanford.edu/)

(102) [http://colah.github.io/posts/2015-08-Understanding-LSTMs/](http://colah.github.io/posts/2015-08-Understanding-LSTMs/)