Recurrent Nested Model for Sequence Generation*

Wenhao Jiang,[1] **Lin Ma,**[1] **Wei Lu**[2]
[1]Tencent AI Lab
[2]University of Electronic Science and Technology of China
{cswhjiang, forest.linma}@gmail.com, luwei@uestc.edu.cn

Abstract

Depth has been shown beneficial to neural network models. In this paper, we make an attempt to make the encoder-decoder model deeper for sequence generation. We propose a module that can be plugged into the middle between the encoder and decoder to increase the depth of the whole model. The proposed module follows a nested structure, which is divided into blocks with each block containing several recurrent transition steps. To reduce the training difficulty and preserve the necessary information for the decoder during transitions, inter-block connections and intra-block connections are constructed in our model. The inter-block connections provide the thought vectors from the current block to all the subsequent blocks. The intra-block connections connect all the hidden states entering the current block to the current transition step. The advantages of our model are illustrated on the image captioning and code captioning tasks.

Introduction

The general encoder-decoder framework is usually a framework that learns a transformation of representations. In this framework, an encoder is used to encode the input into a vector and a decoder generates the output based on the encoded vector. Recently, the encoder-decoder framework has been applied to the sequence learning problem based on recurrent neural networks (RNNs) (Sutskever, Vinyals, and Le 2014). The decoder for sequence learning is usually a RNN, but the encoder can be a RNN or a convolutional neural network (CNN), depending on the specific tasks.

It is known that deep feedforward neural networks are more expressive than shallow neural networks (Bengio and others 2009; Pascanu, Montufar, and Bengio 2014). However, deep neural networks are difficult to train. The gradient might vanish or explode. For recurrent neural networks, long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997) was proposed to solve this problem by introducing information gates. Inspired by LSTM, skip connections (He et al. 2016; Srivastava, Greff, and Schmidhuber 2015) were proposed to ease the training and has achieved great success in convolution neural networks (CNNs). Recently, dense connections among layers (Huang et al. 2017) were found to be beneficial for the training of very deep convolutional neural networks.

In recurrent neural networks, increasing the "depth" can also increase the expressive power (Pascanu et al. 2013; Zilly et al. 2017). In (Pascanu et al. 2013), the authors observed that the relationship between consecutive hidden states and that between the hidden states and outputs are not deep enough. And alternative RNNs were proposed by making those shallow relationships to be modeled with deep neural networks. In addition to the methods mentioned above, recurrent highway networks (RHN) (Zilly et al. 2017) made the transition functions much deeper by adding more micro RNN steps and highway connections. Empirical evaluation showed that deeper RNN provide better performance. Hence, increasing the depth also makes RNNs more powerful.

For an encoder-decoder model, except adopting the depth increasing techniques from RNN mentioned before, we can also inserts several transition steps between the encoder and decoder to make the whole model deeper. It is the strategy proposed by Review Net (Yang et al. 2016), and the additional transition steps are called review steps. The review steps processes the information from encoder and passes the results to the decoder. The hidden states from review steps are used as inputs of attention in the decoder. Hence, review steps provide the decoder with better source representations and attention inputs. For Review Net, the best number of review steps selected on validation set is usually around 8 (Yang et al. 2016). And if the number of review steps is larger than that, performance will not be improved further by adding review steps.

In this work, we explore deep extensions of the encoder-decoder model by increasing the number of non-linear transformations from source to target, without changing the designs of the encoder and decoder. Similarly to Review Net (Yang et al. 2016), we insert an intermediate module between the encoder and decoder, which contains several RNN transition steps and learns a transformation for the en-

*This work is partly supported by National Natural Science Foundation of China (Grant No. 71901050) and the General Programs of the Sichuan Province Social Science Association (Grant No. SC19B035).

coded vector. The module between the encoder and decoder is called **reviewer** in this paper. The reviewer introduces extra transition steps and makes the whole encoder-decoder model deeper. The total number of RNN steps in the reviewer is called **review depth**. In our model, the reviewer follows a nested structure, which contains several blocks, and each block consists of multiple RNNs. To ease the training process and preserve information of source for the decoder, *inter-block connections and intra-block connections* are constructed to facilitate error propagation and reuse of hidden states. These connections connect preceding hidden states with subsequence RNN steps. With the novel architecture proposed, the review depth is increased from 8 (Review Net) to 64. The advantages of our model are illustrated on the image captioning and code captioning tasks.

Related Works

Depth has been shown beneficial for the feed-forward neural networks intuitively , theoretically and empirically . It was shown in (Bengio and others 2009) that, deep sum-product networks require exponentially less node to express some families of polynomials compared to the shallow ones. Deep neural networks with piecewise linear activation are more expressive than shallow ones (Montufar et al. 2014; Pascanu, Montufar, and Bengio 2014). Hence, to approximate the complex functions, the neural networks have to be sufficiently deep. However, deep neural networks are difficult to optimize. Skip connections (He et al. 2016) and dense connections (Huang et al. 2017) are proposed to ease the training process. They both provide a better way for error to propagate back to the shallow parts.

For RNNs, the traditional way to increase the depth is to stack multiple layers of RNNs and performance improvements were observed (Graves 2013). In (Chung et al. 2015), extra connections between all state across consecutive time steps in a stacked RNN were proposed. In (Pascanu et al. 2013), it was shown that in the traditional RNNs the hidden-to-hidden, hidden-to-output and input-to-hidden functions are shallow. Hence, to make RNNs deep, these functions are needed to be replaced by deeper functions. Based on that observations, two extensions were proposed in (Pascanu et al. 2013) and promising performance are achieved on various tasks. In (Zilly et al. 2017), the step-to-step transition functions of RNN were made deep by combining the micro time steps and highway connections.

For an encoder-decoder model, the depth is determined by the types of the encoder and decoder. In order to make the whole model deeper, an intermediate module can be inserted between the encoder and decoder. Review Net (Yang et al. 2016) adopted such strategy and the intermediate step inserted was called review step, which reviews the encoded vector from the encoder and passes it to the decoder. Hence, the review step reprocesses the information from encoder and great improvements were achieved (Yang et al. 2016). The review depth of Review Net is usually 8-10. In this paper, we try to make reviewer much deeper.

Background

To provide a clear description of our method, we present a short review of the encoder-decoder framework for sequence generation in this section.

Long Short-Term Memory. RNN is the basic building block for the encoder-decoder based sequence learning model. In this paper, we use long short-term memory LSTM (Hochreiter and Schmidhuber 1997) in our model. Recall that a LSTM is a function that perform a state transition based on the current hidden state and current input. The state transition of LSTM unit can be expressed as follows:

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \mathbf{g}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{T}\mathbf{H}_t, \tag{1}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t, \tag{2}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \tag{3}$$

where $\mathbf{i}_t, \mathbf{f}_t, \mathbf{c}_t, \mathbf{o}_t$ and $\mathbf{h}_t$ are input gate, forget gate, memory cell, output gate and hidden state of the LSTM, respectively. Here, $\mathbf{H}_t = \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix}$ is the concatenation of input $\mathbf{x}_t$ for the time step t and hidden state $\mathbf{h}_{t-1}$, $\mathbf{T}$ is a linear transformation operator. In this paper, we use the shorthand notation

$$[\mathbf{h}_t, \mathbf{c}_t] = \text{LSTM}(\mathbf{H}_t, , \mathbf{c}_{t-1}) \tag{4}$$

to express the above equations.

In (Bahdanau, Cho, and Bengio 2015; Xu et al. 2015), attention mechanism was introduced into the decoder. At each time step, the attention model performs attention on the set of annotation vectors $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_M\}$ and the context vector $\mathbf{z}_t$ is obtained with an attention model. More specifically, the attention model can be expressed as

$$e_{ti} = \text{sim}(\mathbf{a}_i, \mathbf{h}_{t-1}), \alpha_{ti} \quad = \frac{\exp(e_{ti})}{\sum_{j=1}^{k}\exp(e_{tj})}, \mathbf{z}_t = \sum_{i=1}^{k} \alpha_{ti}\mathbf{a}_i,$$

where $\text{sim}(\mathbf{a}_i, \mathbf{h}_{t-1})$ is a function to measure the similarity between $\mathbf{a}_i$ and $\mathbf{h}_{t-1}$ and is usually a multilayer perceptron (MLP). For LSTM with attention, the input for each time step is $\mathbf{H}_t = \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \\ \mathbf{z}_t \end{bmatrix}$. The LSTM equipped with attention mechanism usually performs better than the vanilla LSTM.

Encoder. In the encoder-decoder model, the encoder transforms the input into a vector $\mathbf{a}$ and a set of annotation vectors A. The vector $\mathbf{a}$ contains the information of the whole input, and the vectors in A contains the information of a certain part of the input. The encoder can be a RNN or a CNN, depending on the particular tasks.

For tasks like machine translation, the goal is to translate the source sequence into target sequence. To achieve that, the source sequence is first encoded and then a target sequence is generated by the decoder based on the information from the encoder. Hence, the encoder for such tasks is a RNN. We denote the source sequence as $(y_1, y_2, \cdots, y_M)$, where M is the length. The source sequence is fed into the encoder and the hidden states $\mathbf{a}_m$ at each time step form the annotation vectors $A = \{\mathbf{a}_1, \dots, \mathbf{a}_M\}$. The hidden state $\mathbf{a}_m$ only contains the information of the prefix of length m. The last hidden state encodes the information about the whole input sequence, hence $\mathbf{a} = \mathbf{a}_M$.

For tasks like image captioning, the goal is to translate the source image into a natural sentence. A CNN trained for image classification task is usually employed as an encoder, which extracts the global representations and subregion representations of the input image. The global representation is usually the outputs of full connecting layers and subregion representations are usually the outputs of convolutional layers. The extracted global representation and subregion representations are denoted as $\mathbf{a}$ and $A = \{\mathbf{a}_1, \dots, \mathbf{a}_M\}$ respectively, where M is the number of subregions.

Decoder. Given the representations $\mathbf{a}$ and A from the encoder, a decoder is employed to translate the encoded information into target sequence, like natural sentence. In this paper, we use LSTM equipped with soft attention mechanism as the basic unit of the decoder. The aim of encode-decoder model is to generate a sequence $\mathcal{C} = (y_1, y_2, \cdots, y_N)$ for given input. The objective adopted is usually to minimize a negative log-likelihood:

$$\mathcal{L} = -\log\ p(\mathcal{C}|\mathcal{I}) = -\sum_{t=0}^{N-1} \log p(y_{t+1}|y_t), \tag{5}$$

where $p(y_{t+1}|y_t) = \text{Softmax}(\mathbf{W}\mathbf{h}_t)$ and $\mathbf{h}_t$ is computed by the LSTM unit setting $\mathbf{x}_t = \mathbf{E}\mathbf{y}_t$. Here, $\mathbf{W}$ is a matrix for linear transformation, y_0 is the sign for the start of sentences, and $\mathbf{E}\mathbf{y}_t$ denotes the distributed representation of the term $\mathbf{y}_t$, in which $\mathbf{y}_t$ is the one-hot representation for the word y_t and $\mathbf{E}$ is the word embedding matrix.

Reviewer. A reviewer contains several RNN steps and is inserted between the encoder and decoder. The RNN unit is usually a LSTM with attention mechanism and the input at each time step is $\mathbf{H}_t = \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{z}_t \end{bmatrix}$. Note that the input for the review step does not contain any information of target sequence. The hidden states of reviewer are outputted as thought vectors and will be used as input of the attention model for the decoder. The review step learns better annotation vectors for attention model and better hidden states for the decoder. It was observed that adding the reviewer module provides promising performance on image captioning and code captioning tasks (Yang et al. 2016).

The Proposed Model

In Review Net, the hidden states in the review step will be used as input of the attention model in the decoder step. Hence, the error from decoder can be propagated back to the review step. However, the review depth of is usually about 8 and the performance can not be improved further by adding more RNN units. A Review Net with more review steps is more expressive than one with less review steps. The first reason for the performance can not be further improved by adding more review steps might be that the model with larger review depth is more difficult to optimize. The training error of Review Net does not always decrease as the number of review steps increase. The second reason might be that the reviewer can not preserve enough information, especially when review depth is large. If the hidden state after the last step of reviewer can not preserve enough information about the input, the decoder will not decode properly. Hence, we need to design a structure that is easy to optimize and preserve necessary information. In this paper, we achieve that goal by adopting a nested structure and constructing inter- and intra-block connections.

Recurrent Nested Model

The framework of our model is shown in Fig. 1. We can see that the reviewer contains several recurrent blocks, which contains several transition steps. Each block generates a set of thought vectors. The LSTM units are densely connected in our model, and the connections can be categorized into two types: *inter-block connections* and *intra-block connections*. In our framework, each review block takes the thought vectors generated by all the preceding blocks as inputs of the attention models and its thought vectors will be used by attention models in all the subsequent review blocks, which are inter-block connections. The inter-block connections provide the thought vectors from current block to all the subsequence block. Hence, the thought vectors can be utilized immediately after they are generated. Inside the review block, the input at each time step contains the hidden states from all previous steps in the same block. They are variants of high-order RNN (Soltani and Jiang 2016). Hence, the intra-block connections can better capture long term dependency. The decoder takes only the thought vectors from the last review block as the input for the attention model. By the inter- and inter-block connections, the hidden states in the reviewer are reused heavily, which helps propagate error and preserve necessary information about the input. We provide formal and detailed descriptions as follows.

We denote the number of transition steps within the review block as L and the number of review blocks as B.The set of hidden states from encoder is denoted as $A^{(0)}$, and the set of thought vectors from the b-th ($b \geq 1$) review block is denoted as $A^{(b)}$. The RNNs in the reviewer are all LSTM units, and the i-th ($i \geq 1$) LSTM in the b-th review block is denoted as $\text{LSTM}^{(b,i)}$. Assume the global time step for the i-th LSTM unit in the b-th block is t and the transition step of is expressed as

$$[\mathbf{h}_t, \mathbf{c}_t] = \text{LSTM}^{(b,i)}\left(\mathbf{H}_t, \mathbf{c}_{t-1}\right). \tag{6}$$

It is a LSTM unit equipped with b attention models and we denote the j-th ($0 \leq j \leq b-1$) as $f^{(b,i,j)}_{\text{review-att}}(A^{(j)}, \mathbf{h}_{t-1})$. The input $\mathbf{H}_t$ is the concatenation of two vectors, which represent intra- and inter-block connections. The first vector $\mathbf{H}_t^{(intra)}$ is the concatenation of the preceding i hidden states, which can be expressed as

$$\mathbf{H}_t^{(intra)} = \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{h}_{t-2} \\ \vdots \\ \mathbf{h}_{t-i} \end{bmatrix}. \tag{7}$$

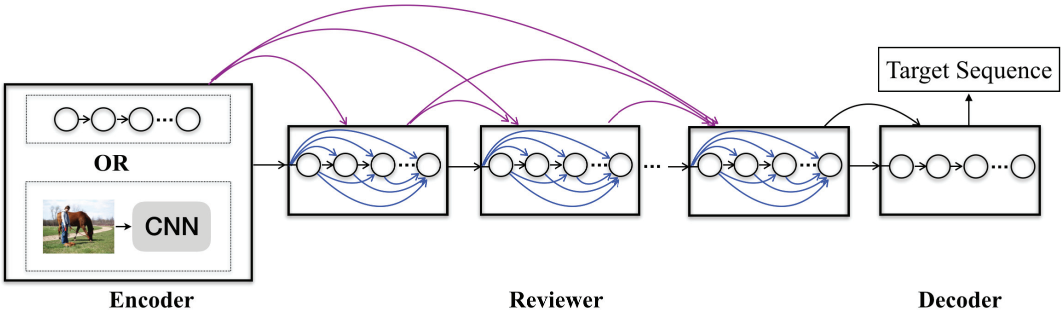

Figure 1: The diagrammatic illustration of our model. The reviewer contains recurrent blocks, each of which contains nested transition steps. The purple lines represent the inter-block connections, which connect the thought vectors from preceding blocks with attention models in the subsequent blocks. The blue lines represent the intra-block connections which connect the preceding hidden states to the input of subsequent transition steps. The intra-block connections capture higher-order dependency in the steps of reviewer. The decoder takes only the last thought vectors as input of the attention model.

For $i = 1$, $\mathbf{H}_t^{(intra)}$ contains only one hidden state $\mathbf{h}_{t-1}$, which is the one entering the current block from previous block. For $i > 1$, $\mathbf{H}_t^{(intra)}$ contains all the preceding states within the same review block and the hidden state after the last step of previous block. Hence, with $\mathbf{H}_t^{(intra)}$, the LSTM become a higher-order RNN (Soltani and Jiang 2016) and can learn long term dependency in sequences.

The second vector $\mathbf{H}_t^{(inter)}$ is the concatenation of the outputs of b attention models, which can be expressed as

$$\mathbf{H}_t^{(inter)} = \begin{bmatrix} f_{\text{review-att}}^{(b,i,0)}(A^{(0)}, \mathbf{h}_{t-1}) \\ f_{\text{review-att}}^{(b,i,1)}(A^{(1)}, \mathbf{h}_{t-1}) \\ \vdots \\ f_{\text{review-att}}^{(b,i,b-1)}(A^{(b-1)}, \mathbf{h}_{t-1}) \end{bmatrix}. \quad (8)$$

With $\mathbf{H}_t^{(intra)}$ and $\mathbf{H}_t^{(inter)}$, the input for the i-th LSTM in the b-th block is expressed as $\mathbf{H}_t = \begin{bmatrix} \mathbf{H}_t^{(intra)} \\ \mathbf{H}_t^{(inter)} \end{bmatrix}$. Since $\mathbf{H}_t^{(intra)}$ is for high-order connections and $\mathbf{H}_t^{(inter)}$ is for connections from multiple attention models, our transition unit is a higher-order LSTM with multiple attention models.

Discriminative Supervision

Discriminant supervision (Fang et al. 2015; Yang et al. 2016) is adopted in our model to further boost image captioning. Given a set of thought vectors A, a matrix $\mathbf{A}$ is formed by selecting elements from V as column vectors. A score vector $\mathbf{s}$ of words is then calculated as $\mathbf{s}(\mathbf{A}) = \text{Row-Max-Pool}\left(\mathbf{WA}\right)$, where $\mathbf{s}(\mathbf{A})$ means $\mathbf{s}$ is vector depending on $\mathbf{A}$, $\mathbf{W}$ is a linear transformation matrix and Row-Max-Pool$(\cdot)$ is a max pooling operator along the rows of the input matrix. The i-th element of $\mathbf{s}$ is denoted s_i, which represents the score for the i-th word. Adopting multi-label margin loss, we obtain the loss function for discriminant supervision written as:

$$\mathcal{L}_{dis}\left(A\right) = \sum_{j \in \mathcal{W}} \sum_{i \notin \mathcal{W}} \max\left(0, 1 - (s_j - s_i)\right), \quad (9)$$

where $\mathcal{W}$ is the set of all frequent words in the current sample. In this paper, we only consider the 1,000 most frequent words in training set. With the above discriminant supervision loss function, the complete loss function of our model is expressed as:

$$\mathcal{L}_{all} = \mathcal{L} + \frac{\lambda}{B} \sum_{b=1}^{B} \mathcal{L}_{dis}\left(A^{(b)}\right), \quad (10)$$

where λ is a trade-off parameter.

The discriminative supervision can provide auxiliary supervision signal for the hidden states from the review steps. Hence, this technique can ease the training to some extent and improve the quality of the thought vectors.

Connections to Review Net

Our model is a generalization of Review Net. If we remove all the inter- and intra-block connections and set $B = 1$, our model will be exactly the same as Review Net. The reviewer in Review Net only contains one review block and can only generate one set of thought vectors. The effective review depth is only about 8 for image captioning and code captioning (Yang et al. 2016). However, with the inter- and intra-block connections, our model can achieve better performance with larger review depth.

Experiments

We illustrate the advantages of our model on two tasks, *i.e.,* image captioning and code captioning. The types of encoders are different for models on those two tasks. For image captioning, the encoder is a CNN, while for the code captioning, a RNN is employed.

Image Captioning

Dataset. The MS COCO dataset[1] is the largest dataset for the image captioning task. This dataset contains $82,783$, $40,504$ and $40,775$ images for training, validation and test

[1] http://mscoco.org/

sets respectively. The captioning task on this dataset is challenging, because most images contain multiple objects in complex scenes. Each image in this dataset has 5 captions annotated by human. For offline evaluation, we following the conventional evaluation procedure (Mun, Cho, and Han 2017; Yao et al. 2017; Yang et al. 2016), and employ the Karpathy's split (Karpathy and Fei-Fei 2015), which contains 5,000 images for validation, $5,000$ images for test and $113,287$ images for training.

Training Settings. We use ResNet (He et al. 2016) as the encoder for the image captioning task. Specifically, the outputs of the last convolutional layer (before pooling) are extracted as subregion features. The parameters for encoders are fixed during the training encoder-decoder model. The commonly used techniques including scheduled sampling, label-smoothing regularization (LSR), dropout and early stopping are adopted. For scheduled sampling, the probability of sampling a token from model is $\min(0.25, \frac{epoch}{100})$, where $epoch$ is the number of passes sweeping over training data. For LSR, the prior distribution over labels is uniform distribution and the smoothing parameter is set to 0.1. Dropout is only applied on the hidden states in reviewer and decoder and the probability is set to 0.3. If the evaluation measurement on validation set, specifically the CIDEr, reaches the maximum value, we terminate the training procedure. The LSTM size is set as 512 for all LSTM units in our model. The Adam (Kingma and Ba 2015) is applied to optimize the network, and learning rate is set to 5×10^{-4} and decay every 3 epochs by a factor 0.8 when training with cross entropy loss. Each mini-batch contains 10 images.

For sentence generation in testing stage, beam search strategy usually provides better performance for models trained with cross entropy loss, but greedy search is faster. Hence, greedy search is used for evaluation on the validation set and beam search is used to generate the final results.

Evaluation metrics. Following the standard evaluation process, five types of metrics are used for performance comparisons, specifically the BLEU (Papineni et al. 2002), METEOR (Banerjee and Lavie 2005), ROUGE-L (Lin 2004), CIDEr (Vedantam, Lawrence Zitnick, and Parikh 2015), and SPICE (Anderson et al. 2016). These metrics measure the similarity between generated sentences and the ground truth sentences from different viewpoints. To provide convincing comparisons, we provide the performance measured by all the above metrics. We use the official MSCOCO caption evaluation scripts[2] and source code of SPICE[3] for the performance evaluation.

The captioning performance of competing models by beam search is presented in Table 2, including the performance of single model and ensemble of models. For our model and Review Net, the ensembles of 4 models are evaluated. We can observe that the performance of our single model is better than the baseline method Review Net, which proves that our strategy of constructing dense connections in reviewer can improve the performance. Moreover, ensemble of our models is also better than the ensemble of Review Net. And the ensemble of our models performs the best in competing models. The results of the same ensemble model are submitted to the evaluation server for more complete comparisons. The results of online evaluation are shown in Table 3. We can see that our method outperforms the competing methods trained with cross entropy loss. In addition, the models Att2all* (Rennie et al. 2017) and Up-Down* (Anderson et al. 2017) are better than ours. However, they were trained with reinforcement learning and can not be used to compare the model ability with our model. Note that our model is better than Att2all trained with cross entropy loss, as shown in Table 2.

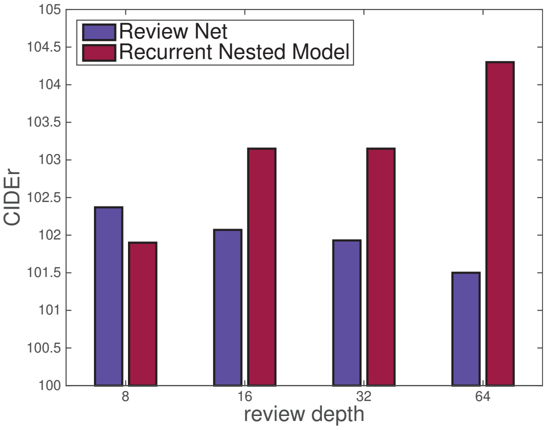

Figure 2: The effects of review depth.

Comparison with Review Net. To show the advantages of our model, we also compare our model with Review Net with different review depths. The CIDEr scores with greedy search on validation set are plotted in Fig 2. For our model, the number of steps within each block is set to 8. We can see that our model becomes better as the review depth increases. However, Review Net shows opposite trends. This phenomenon are attributes to the nested structure and the inter- and intra-block connections adopted in our model. Moreover, we can see that our model performs the best when review depth is 64, *i.e.* the number of blocks is 8. Hence, B is set to 8 in all our experiments.

The Effects of L. The CIDEr scores on the validation dataset with different values of L are studies in Table 3, in which the number of blocks is fixed to 8. We can see that the performance of model does not vary greatly. The model with $L = 8$ is the best and the model with $L = 2$ performs the worst. Note that L is also the number of the thought vectors generated by each block. Hence, if L is too small, the thought vectors do not contain the necessary information for the decoder. Larger L will lead to more memory occupation and it seems that if L is bigger than 8, increase L does not improve the performance. Hence, L is set to 8 in all our experiments.

[2] https://github.com/tylin/coco-caption

[3] https://github.com/peteanderson80/coco-caption

Table 1: Performance comparisons on the test set of Karpathy's split (Karpathy and Fei-Fei 2015). All image captioning models are trained with the cross-entropy loss. Σ indicates an ensemble, † indicates a different split, and $(-)$ indicates an unknown metric. All values are reported as percentage (%), and the highest value of each entry has been highlighted in boldface. The results are obtained using beam search with beam size 3.

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L	CIDEr	SPICE
Soft Attention (Xu et al. 2015)	70.7	49.2	34.4	24.3	23.9	-	-	-
Review Net(Yang et al. 2016)	-	-	-	29.0	23.7	-	88.6	-
LSTM-A3 (Yao et al. 2017)	73.5	56.6	42.9	32.4	25.5	53.9	99.8	18.5
Text Attention (Mun, Cho, and Han 2017)	74.9	58.1	43.7	32.6	25.7	-	102.4	-
Attribute LSTM (Wu et al. 2016)	74.0	56.0	42.0	31.0	26.0	-	94.0	-
RIC (Liu et al. 2016)	73.4	53.5	38.5	29.9	25.4	-	-	-
CNN$_{\mathcal{L}}$ +RHN (Gu et al. 2017)	72.3	55.3	41.3	30.6	25.2	-	98.9	-
Adaptive (Lu et al. 2016)†	74.2	58.0	43.9	33.2	26.6	-	108.5	-
Att2in (Rennie et al. 2017)	-	-	-	31.3	26.0	54.3	101.3	-
Att2all (Rennie et al. 2017)	-	-	-	30.0	25.9	53.4	99.4	-
ReviewNet	74.8	58.3	44.5	33.9	26.4	54.9	106.5	19.6
Recurrent Nested Model	**75.7**	**59.6**	**45.8**	**35.1**	**26.9**	**55.7**	**109.7**	**20.1**

Table 2: Performance of model ensemble on the test set of Karpathy's split (Karpathy and Fei-Fei 2015). All image captioning models are trained with the cross-entropy loss. Σ indicates an ensemble, † indicates a different split, and $(-)$ indicates an unknown metric. All values are reported as percentage (%), and the highest value of each entry has been highlighted in boldface. The results are obtained using beam search with beam size 3.

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L	CIDEr	SPICE
Attribute Attention (You et al. 2016)$^{\Sigma}$	70.9	53.7	40.2	30.4	24.3	-	-	-
NIC (Vinyals et al. 2015)$^{\dagger\Sigma}$	-	-	-	32.1	25.7	-	99.8	-
Att2in (Rennie et al. 2017)$^{\Sigma}$	-	-	-	32.8	26.7	55.1	106.5	-
Att2all (Rennie et al. 2017)$^{\Sigma}$	-	-	-	32.2	26.7	54.8	104.7	-
ReviewNet$^{\Sigma}$	76.4	60.3	46.6	35.7	27.3	56.2	111.7	20.3
Recurrent Nested Model$^{\Sigma}$	**76.7**	**60.8**	**46.9**	**36.2**	**27.5**	**56.5**	**113.1**	**20.4**

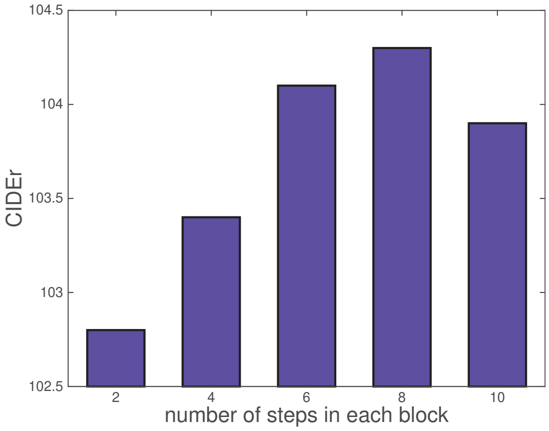

Figure 3: The effects of the number of steps in each block.

Ablation Studies. To show the effects of the two types of dense connections, we perform an ablation study. We compare our model with the following variants:

- **Recurrent Nested Model - All Connections**: All the dense connections are removed and the rest stays the same with our model. It is similar to the Review Net, but the RNN steps in reviewer are divided into blocks.
- **Recurrent Nested Model - Intra-block Connections**: Only the high-order intra-block connections are removed and the rest stays the same with our model. Each transition unit is connected to the units belonging to the preceding blocks.
- **Recurrent Nested Model - Inter-block Connections**: Only the inter-block connections are removed and the rest stays the same with our model. Each transition unit is connected to the units belonging to the same block and the last step of the previous block.

For all the models, both B and L are set to 8. The CIDEr scores by greedy search on the validation set are presented in Table 4. We can see that removing any kind of connections lead to an inferior model and the model without the two types of connection performs the worst. Hence, the dense connections are helpful and necessary for our model.

Code Captioning

Dataset. In addition to the task with a CNN as the encoder, we also evaluate our model on the code captioning, on which the encoder is a RNN. The task of code captioning is to generate the comment for the given source code. The HabeasCorpus (Movshovitz-Attias and Cohen 2013) dataset is used to illustrate the effectiveness of our method. In this dataset, 9 popular open source Java code are collected. The

Table 3: Performance of different models on the MS COCO evaluation server. All values are reported as percentage (%). $*$ indicates models trained with reinforcement learning.

	BLEU-1		BLEU-2		BLEU-3		BLEU-4		METEOR		ROUGE-L		CIDEr		SPICE	
Methods	C5	C40	C5	C40	C5	C40	C5	C40	C5	C40	C5	C40	C5	C40	C5	C40
NIC (Vinyals et al. 2015)	71.3	89.5	54.2	80.2	40.7	69.4	30.9	58.7	25.4	34.6	53.0	68.2	94.3	94.6	18.2	63.6
Captivator (Fang et al. 2015)	71.5	90.7	54.3	81.9	40.7	71.0	30.8	60.1	24.8	33.9	52.6	68.0	93.1	93.7	18.0	60.9
LRCN (Donahue et al. 2015)	71.8	89.5	54.8	80.4	40.9	69.5	30.6	58.5	24.7	33.5	52.8	67.8	92.1	93.4	17.7	59.9
Hard-Attention (Xu et al. 2015)	70.5	88.1	52.8	77.9	38.3	65.8	27.7	53.7	24.1	32.2	51.6	65.4	86.5	89.3	17.2	59.8
ATT-FCN (You et al. 2016)	73.1	90.0	56.5	81.5	42.4	70.9	31.6	59.9	25.0	33.5	53.5	68.2	94.3	95.8	18.2	63.1
Adaptive (Lu et al. 2016)	74.8	92.0	58.4	84.5	44.4	74.4	33.6	63.7	26.4	35.9	55.0	70.5	104.2	105.9	19.7	67.3
PG-BCMR (Liu et al. 2017)	75.4	91.8	59.1	84.1	44.5	73.8	33.2	62.4	25.7	34.0	55.0	69.5	101.3	103.2	-	-
Review Net (Yang et al. 2016)	72.0	90.0	55.0	81.2	41.4	70.5	31.3	59.7	25.6	34.7	53.3	68.6	96.5	96.9	18.5	64.9
Att2all (Rennie et al. 2017)*	78.1	93.7	61.9	86.0	47.0	75.9	35.2	64.5	27.0	35.5	56.3	70.7	114.7	116.7	20.7	68.9
Up-Down (Anderson et al. 2017)*	80.2	95.2	64.1	88.8	49.1	79.4	36.9	68.5	27.6	36.7	57.1	72.4	117.9	120.5	21.5	71.5
AoANet (Huang et al. 2019)	81.0	95.0	65.8	89.6	51.4	81.3	39.4	71.2	29.1	38.5	58.9	74.5	126.9	129.6		
Our model	76.3	93.6	60.1	86.6	46.3	77.1	35.5	66.6	27.1	36.9	56.0	72.0	108.4	108.9	-	-

Table 4: Ablation study of inter- and intra-block connections. The CIDEr scores on the validation set are reported.

Method	CIDEr
Recurrent Nested Model	104.3
Recurrent Nested Model - All Connections	102.6
Recurrent Nested Model - Intra-block Connections	103.6
Recurrent Nested Model - Inter-block Connections	103.0

Table 5: Comparison of models on the HabeasCorpus code captioning dataset."CS-k" refers to top-k character savings.

Model	CS-1	CS-2	CS-3	CS-4	CS-5
LSTM	69.5	76.9	80.3	82.2	83.5
SoftAttention	71.2	78.6	81.7	83.4	84.7
Review Net	**71.9**	79.1	82.1	83.8	85.0
Our model	**71.9**	**79.2**	**82.4**	**84.1**	**85.4**

dataset contains 6K Java source code files with 7M source code tokens and 251K comment word tokens. About 10% of the files are randomly selected as the test set and validation set, respectively, and the rest 80% is used for training.

Metrics. All the methods are evaluated with the task of code comment completion. We apply the model to predict the next token and compute the percentage of characters that can be saved. The metric of top-k character savings (CS-k) (Movshovitz-Attias and Cohen 2013) is used for evaluation. Specially, for a word of length l, we denote the minimum number of prefix characters needed to be typed such that the actual word ranks among the top-k list based on the given model as n, and the number of saved characters would be $l - n$. To obtain CS-k, the average percentage of saved characters per comment is calculated. We follow the same preprocessing as in (Yang et al. 2016) and set the dimension of LSTM unit to 512, $L = 8$ and $B = 8$.

Results. We compare our model with the encoder-decoder models whose decoder or reviewer are LSTM, soft attention (Bahdanau, Cho, and Bengio 2015) and Review Net (Yang et al. 2016). For the all the models compared, the encoders are all LSTMs. Hence, we use the types of decoder or reviewer to distinguish them. So, "LSTM" means an encoder-decoder model with LSTM as both encoder and decoder, and "SoftAttention" means an encoder-decoder model with LSTM as encoder and LSTM with soft attention mechanism as decoder.

The performance measured with CS-k is presented in Table 5[4]. We can see that introducing soft attention mechanism and reviewer step can improve the performance over the vanilla encoder-decoder. But our model can further improve the performance, which shows that the dense review network is also effective for the setting with RNN encoder.

Conclusion

In this paper, a recurrent nested model was proposed, which can make the encoder-decoder model for sequence learning deeper. Thus, the effective review depth is increased from 8 to 64. In our model, the reviewer is divided into blocks, which contains nested transition steps. Two types of connections between RNN steps were introduced, *i.e.*, inter-block connections and intra-block connections. The intra-block connections make the transition steps be aware of the all the states of preceding steps within the same block, and the inter-block connections make the block be aware of the thought vectors from all preceding blocks. By virtue of the connections introduced, the hidden states in the reviewer are reused heavily, thus helping propagate errors and preserve necessary information about the inputs. The effectiveness of the proposed models has been verified sufficiently on image captioning and code captioning tasks.

References

Anderson, P.; Fernando, B.; Johnson, M.; and Gould, S. 2016. Spice: Semantic propositional image caption evalu-

[4] The results in Table 3 of (Yang et al. 2016) is not correct due to a bug in the computation of CS-k metric. The bug is fixed in our code.

ation. In *ECCV*, 382–398.

Anderson, P.; He, X.; Buehler, C.; Teney, D.; Johnson, M.; Gould, S.; and Zhang, L. 2017. Bottom-up and top-down attention for image captioning and vqa. *arXiv preprint arXiv:1707.07998*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. *ICLR-15*.

Banerjee, S., and Lavie, A. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *ACL workshop*.

Bengio, Y., et al. 2009. Learning deep architectures for ai. *Foundations and trends® in Machine Learning* 2(1):1–127.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2015. Gated feedback recurrent neural networks. In *ICML*, 2067–2075.

Donahue, J.; Anne Hendricks, L.; Guadarrama, S.; Rohrbach, M.; Venugopalan, S.; Saenko, K.; and Darrell, T. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2625–2634.

Fang, H.; Gupta, S.; Iandola, F.; Srivastava, R. K.; Deng, L.; Dollár, P.; Gao, J.; He, X.; Mitchell, M.; Platt, J. C.; et al. 2015. From captions to visual concepts and back. In *CVPR*, 1473–1482.

Graves, A. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Gu, J.; Wang, G.; Cai, J.; and Chen, T. 2017. An empirical study of language cnn for image captioning. In *ICCV*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *CVPR*, 4700–4708.

Huang, L.; Wang, W.; Chen, J.; and Wei, X.-Y. 2019. Attention on attention for image captioning. In *ICCV*, 4634–4643.

Karpathy, A., and Fei-Fei, L. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, 3128–3137.

Kingma, D., and Ba, J. 2015. Adam: A method for stochastic optimization. *ICLR*.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. In *ACL-04 workshop*.

Liu, H.; Yang, Y.; Shen, F.; Duan, L.; and Shen, H. T. 2016. Recurrent image captioner: Describing images with spatial-invariant trnsformation and attention fieltering. *arXiv:1612.04949*.

Liu, S.; Zhu, Z.; Ye, N.; Guadarrama, S.; and Murphy, K. 2017. Improved image captioning via policy gradient optimization of spider. In *ICCV*.

Lu, J.; Xiong, C.; Parikh, D.; and Socher, R. 2016. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. *arXiv preprint arXiv:1612.01887*.

Montufar, G. F.; Pascanu, R.; Cho, K.; and Bengio, Y. 2014. On the number of linear regions of deep neural networks. In *NeurIPS*, 2924–2932.

Movshovitz-Attias, D., and Cohen, W. W. 2013. Natural language models for predicting programming comments. In *ACL*, 35–40.

Mun, J.; Cho, M.; and Han, B. 2017. Text-guided attention model for image captioning. *AAAI*.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 311–318.

Pascanu, R.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2013. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

Pascanu, R.; Montufar, G.; and Bengio, Y. 2014. On the number of response regions of deep feed forward networks with piece-wise linear activations. In *ICLR 2014*.

Rennie, S. J.; Marcheret, E.; Mroueh, Y.; Ross, J.; and Goel, V. 2017. Self-critical sequence training for image captioning. In *CVPR*, 7008–7024.

Soltani, R., and Jiang, H. 2016. Higher order recurrent neural networks. *arXiv preprint arXiv:1605.00064*.

Srivastava, R. K.; Greff, K.; and Schmidhuber, J. 2015. Training very deep networks. In *NeurIPS*, 2377–2385.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *NeurIPS*. 3104–3112.

Vedantam, R.; Lawrence Zitnick, C.; and Parikh, D. 2015. Cider: Consensus-based image description evaluation. In *CVPR*.

Vinyals, O.; Toshev, A.; Bengio, S.; and Erhan, D. 2015. Show and tell: A neural image caption generator. In *CVPR*, 3156–3164.

Wu, Q.; Shen, C.; Liu, L.; Dick, A.; and van den Hengel, A. 2016. What value do explicit high level concepts have in vision to language problems? In *CVPR*.

Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; and Bengio, Y. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2048–2057.

Yang, Z.; Yuan, Y.; Wu, Y.; Cohen, W. W.; and Salakhutdinov, R. R. 2016. Review networks for caption generation. In *NeurIPS*, 2361–2369.

Yao, T.; Pan, Y.; Li, Y.; Qiu, Z.; and Mei, T. 2017. Boosting image captioning with attributes. In *ICCV*, 4894–4902.

You, Q.; Jin, H.; Wang, Z.; Fang, C.; and Luo, J. 2016. Image captioning with semantic attention. In *CVPR*, 4651–4659.

Zilly, J. G.; Srivastava, R. K.; Koutník, J.; and Schmidhuber, J. 2017. Recurrent highway networks. In *ICML*, 4189–4198.