CNN+CNN: Convolutional Decoders for Image Captioning

Qingzhong Wang and Antoni B. Chan
Department of Computer Science, City University of Hong Kong
qingzwang2-c@my.cityu.edu.hk, abchan@cityu.edu.hk

Abstract

Image captioning is a challenging task that combines the field of computer vision and natural language processing. A variety of approaches have been proposed to achieve the goal of automatically describing an image, and recurrent neural network (RNN) or long-short term memory (LSTM) based models dominate this field. However, RNNs or LSTMs cannot be calculated in parallel and ignore the underlying hierarchical structure of a sentence. In this paper, we propose a framework that only employs convolutional neural networks (CNNs) to generate captions. Owing to parallel computing, our basic model is around 3× faster than NIC (an LSTM-based model) during training time, while also providing better results. We conduct extensive experiments on MSCOCO and investigate the influence of the model width and depth. Compared with LSTM-based models that apply similar attention mechanisms, our proposed models achieves comparable scores of BLEU-1,2,3,4 and METEOR, and higher scores of CIDEr. We also test our model on the paragraph annotation dataset [22], and get higher CIDEr score compared with hierarchical LSTMs.

1. Introduction

Human beings are able to describe what they see, and image captioning is a task that can make a computer have this ability. To achieve this goal requires at least three models: (1) vision model—to extract visual features from images, (2) language model—to generate captions, (3) the connection between vision and language models. Image captioning combines two fields—computer vision and natural language processing (NLP) to address the challenge of understanding both images and their descriptions.

Recently, increasing research has been devoted to image captioning, and a variety of methods have been proposed [16, 8, 21, 31, 34, 36, 33, 2, 19, 9, 35]. Almost all of the current proposed methods are under the framework of CNN+RNN, in which a CNN is used for the vision model, and an RNN is employed to generate sentences. Moreover, there are different ways to connect the CNN and RNN. A naive way is directly feeding the output of the CNN into the RNN [21, 31]. However, this naive approach treats objects in an image the same and ignores the salient objects when generating one word. To imitate the visual attention mechanism of humans, attention modules have been introduced into the CNN+RNN framework [34, 36, 19].

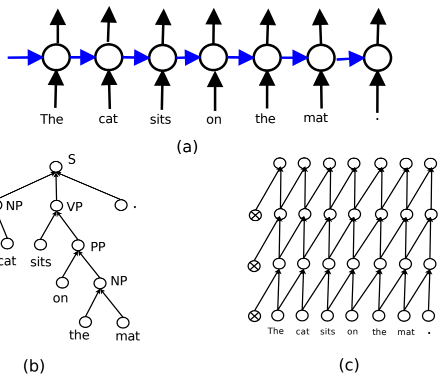


Figure 1. Language models: (a) RNN-based model, which takes the current word and the previous hidden states as input [31]; (b) the tree structure of a sentence, where {S, NP, VP, PP} denote sentence, noun phrase, verb phrase, and preposition phrase [26, 25]; (3) CNN-based model, which is much denser than a tree structure, but is able to learn a tree structure [20]. ⊗ denotes zero padding.

Although the CNN+RNN framework is popular and provides satisfying results, there are disadvantages of sequential RNNs: (1) RNNs have to be calculated step-by-step, which is not amenable to parallel computing during training; (2) there is a long path between the start and end of the sentence using RNNs (see Fig. 1a), which easily forgets the long-range information. Tree structures can make a shorter path between the start and end words (see Fig. 1b), but trees require special processing that is not easily parallelizable. Also trees are defined by NLP (e.g., noun phrase, adjectives, verb phrase, etc), which is a hand-crafted structure that may not be optimal for the captioning task.

An alternative language model to RNNs and trees are CNNs applied to the sentence to merge words layer by layer [22, 20], thus learning a tree structure of the sentence (see Fig. 1c). CNNs can be implemented in parallel, and have a larger receptive field size (can see more words) using less layers. E.g., given a sentence composed of 10 words, a RNN needs to iterate 10 times to get a representation of the sentence (equivalent to 10 layers), while a CNN with kernel size 3 only needs 5 layers. Using wider kernels, CNNs are able to tackle longer sentences with less layers. CNN have been widely applied to the field of NLP, leading to improved performances in many tasks, such as language modeling, machine translation and text classification [5, 10, 11, 13, 4].

Inspired by the applications of CNNs in the field of NLP, we develop a framework that only employ CNNs for image captioning. The main contributions of this paper are:

1. We propose a CNN+CNN framework for image captioning, which is faster than LSTM-based models and outperforms them on some metrics.
2. We propose a hierarchical attention module to connect the vision CNN with the language CNN, which significantly improves the performance.
3. We investigate the influence of the hyper-parameters, including the number of layers and the kernel width of the language CNN. The receptive field of the language CNN can be increased by stacking more layers or increasing the kernel width, and our experiments show that increasing the kernel width is better.

2. Related work

CNN+RNN models Many methods have been proposed to automatically generate captions of images, and the most popular framework is CNN+RNN. Most work endeavors to modify the language model and the connection between the vision and language models to improve the performance.

The m-RNN model [21] uses a vanilla RNN combined with different CNNs (AlexNet or VGG-net) – the RNN hidden states and the CNN output are fed into a multimodal block to fuse the image and language features at each time step, and a softmax layer predicts the next word. However, the vanilla RNNs suffers from the vanishing gradient problem. Therefore, the neural image captioning (NIC) model employs a LSTM as a decoder [31]. At the beginning, the NIC model takes the image feature vector as input, and then the visual information is passed through the recurrent path.

In both m-RNN and NIC, an image is represented by a single vector, which ignores different areas and objects in the image. A spatial attention mechanism is introduced into image captioning model in [34], which allows the model to pay attention to different areas at each time step.

Most recently, [36, 33, 35] have included semantics, which are image annotations produced by an image classifier, into image captioning. A semantic attention model is proposed in [36], which applies a fully convolutional network (FCN) to detect semantics first and then computes a weight for each semantic at each time step. Although this semantic attention significantly improves the performance, to some extent, it has a problem of prediction error accumulation along generated sequences [35]. Alternatively, [9] uses semantics differently, by generating the parameters of an RNN or LSTM from the output of the semantic detector. This model can be trained in a end-to-end manner.

LSTMs have also been used hierarchically to model a shallow tree structure. [28] proposes a phrase LSTM model, which has two levels of LSTMs, one to model the sentence composed of phrases, and another to generate words in a phrase. Similarly, [32] applies a skeleton LSTM to generate the skeleton words of the sentence, and an attribute LSTM to generate adjective words that modify the skeleton words, which is a coarse-to-fine model. Hierarchical LSTMs have also been used to generate a paragraph description consisting of multiple sentences to describe an image [15].

To our knowledge, our work is the first that applies CNNs to completely replace RNNs for image captioning. Inspired by [20], we propose a hierarchical attention module, which aims to learn the co-relationship between concepts at each level and image areas. Furthermore, our attention module uses the dot-product, which has less parameters and can be calculated faster than the MLP-based attention in [34, 19].

CNNs in NLP Deep learning methods have dominated the field of NLP, and CNNs are an important tool to solve NLP problems. In [3], CNNs are applied to several NLP tasks, such as chunking, part-of-speech tagging, named entity recognition, and semantic role labeling. This CNN-based model provides accurate results with fast speed, and is capable of learning representations instead of employing hand-crafted features.

[13] develops a CNN model with a max-over-time pooling layer for sentence classification. [4] experiments with very deep CNNs for text classification, and the results suggest that depth and max-pooling improve the performance.

In terms of language modeling, [5] introduces a new activation function called gated linear units (GLU), which applies an output gate to each unit, and can be trained faster than ReLUs. In [10], GLUs are adopted for machine translation, where a convolutional sequence-to-sequence model outperforms the popular LSTM-based sequence-to-sequence model [27, 1]. In [11], dilated convolution is employed to increase the receptive field of the CNN.

In this paper, our proposed model adopts language CNNs without pooling layers, which is different from [13, 4]. Furthermore, we use *causal* convolution (similar to [29]), so that our model can generate captions word-by-word.

3. Model

CNNs show relatively strong abilities to tackle very long sequences [29]. Inspired by CNNs used for NLP, we propose a CNN+CNN frame work for image captioning. There

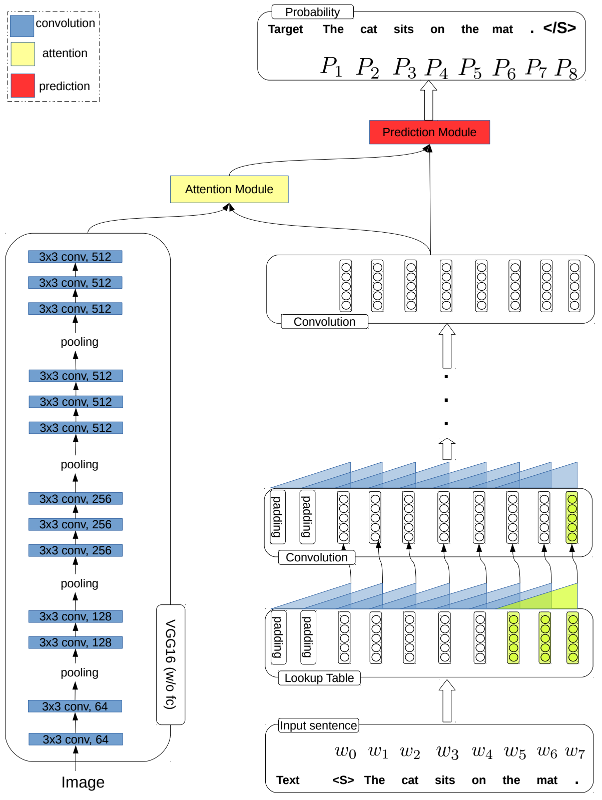


Figure 2. The CNN+CNN model for image captioning. The vision CNN extracts features from the image, and the language CNN is applied to model the sentence. The attention module and prediction module fuse the information from image and word contexts. The convolutions in the language CNN use *causal* filters (depending only on the present and past words), to enable feed-forward sentence generation. For each convolutional layer in the language module, we use $k-1$ zero vectors to pad the sentence matrix along the length axis, where k denotes the kernel size (here, $k=3$).

are four modules in our framework: (1) vision module, which is adopted to "watch" images; (2) language module, which is to model sentences; (3) attention module, which connects the vision module with the language module; (4) prediction module, which takes the visual features from the attention module and concepts from the language module as input and predicts the next word. Fig. 2 illustrates the proposed framework for image captioning.

The vision module The vision module is a CNN without fully connected layer, whose output is a $d \times d \times D_c$ feature map. In each position of the feature map, the D_c dimensional feature vector represents part of the image. Define $\mathbf{v} = [v_1, \cdots, v_N]$ to be the list of image feature vectors, where $v_i \in \mathbb{R}^{D_c}$, i is the position index in the feature map, and $N = d^2$ is the number of positions. In this paper, we use VGG-16 [24] as the CNN for the vision module.

The language module The language model is based on a CNN without pooling, which is very different from the typical RNN-based framework (e.g., [21, 31, 7, 34]). RNNs adopt a recurrent path to memorize the context, while CNNs use kernels and stack multiple layers to model the context.

Let $S = [w_1, w_2, \cdots, w_L]$ be a sentence with L words. We first use a look-up table to project each word into an embedding space with D_e dimensions, and calculate the embeddings $E = [e_1, e_2, \cdots, e_L]$, where $e_j \in \mathbb{R}^{D_e}$. In this stage, a sentence is represented by a $L \times D_e$ matrix. A stack of convolutional layers with gated linear units (GLUs) follows the embedding layer, which is calculated using the following equations:

$$h_a^l = W_a^l * h^{l-1} + b_a^l, \tag{1}$$

$$h_b^l = W_b^l * h^{l-1} + b_b^l, \tag{2}$$

$$h^l = h_a^l \odot \sigma(h_b^l), \tag{3}$$

where $W_a^l, W_b^l \in \mathbb{R}^{k \times D_e}$ denote the kernels of the lth layer, b_a^l and b_b^l are biases, $*$ denotes the convolution operator, $\odot$ denotes the element-wise multiplication, and $\sigma(x) = \frac{1}{1+e^{-x}}$. h_b^l plays the role of a gate and h_a^l is a linear transformation. In our framework $h^0 = E$. Note that the convolution filters are *causal* filters, only depending on the current and past inputs. During inference, this structure allows a sentence to be generated using a feed-forward process where the predicted word at the output layer is used as the next word input.

For modeling sentences, the length of the output is required to be the same as the input sentence. Since there are no pooling layers and fully connected layers, we need to add zero-padding at the beginning of the input and hidden layers. If the convolutional kernel width is k, which indicates that it considers k concepts[1] in each step, the word/concept matrices needs to be zero padded with $k-1$ zero vectors before convolution. The output of the CNN is a set of concepts $\mathbf{c} = [c_1, c_2, \cdots, c_L]$, where $c_j \in \mathbb{R}^{D_e}$.

The attention module Intuitively, to predict different words, different objects in the images should be attended and input into the prediction module. The attention module takes the visual features $\mathbf{v}$ and the concepts $\mathbf{c}$ at the top level as input, which is shown in Fig. 3. For each concept c_j and visual feature vector v_i, we calculate a score $s_{i,j}$ as follows:

$$s_{i,j} = c_j^T U v_i, \tag{4}$$

where $U \in \mathbb{R}^{D_e \times D_c}$ is a parameter matrix. Therefore each concept c_j corresponds to a score vector $\mathbf{s}_j = [s_{1,j}, s_{2,j}, \cdots, s_{N,j}]$, indicating matches with the image

[1]The bottom-level represents words, whereas hidden layers represent different concepts.

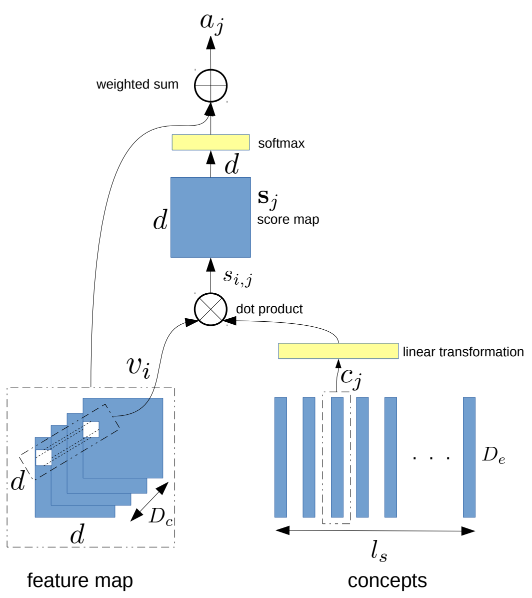


Figure 3. The attention module. We apply dot-product and softmax to calculate the weight for each position and concept, which only has one parameter matrix (linear transformation).

features. Then $\mathbf{s}_j$ is fed into a softmax layer, which assigns a weight $w_{i,j}$ for the visual feature vector v_i,

$$w_{i,j} = \frac{e^{s_{i,j}}}{\sum_{i=1}^{N} e^{s_{i,j}}}. \tag{5}$$

We use the weighted sum operation to calculate the final attention vector as follows,

$$a_j = \sum_{i=1}^{N} w_{i,j} v_i. \tag{6}$$

The output of our attention module is the attention features $\mathbf{a} = [a_1, a_2, \cdots, a_L]$, where $a_i \in \mathbb{R}^{D_c}$, corresponding to each word in the sentence.

The prediction module As shown in Fig. 2, our proposed prediction module takes the attention features $\mathbf{a}$ and concepts $\mathbf{c}$ as input. Our prediction module is a one-hidden layer neural network. At the jth sentence position, a_j and c_j are fed into the network, the output is the prediction probability of next word P_{j+1},

$$h^p_j = f\left(W^p_a a_j + W^p_c c_j + b^p\right), \tag{7}$$
$$P_{j+1} = \text{softmax}(U^p h^p_j), \tag{8}$$

where $W^p_a \in \mathbb{R}^{D^p_h \times D_c}$ and $W^p_c \in \mathbb{R}^{D^p_h \times D_e}$ are the parameters, and D^p_h denotes the number of hidden units in the prediction module. $f(x) = \max(x, 0.1x)$ is a leaky ReLU, and $U^p \in \mathbb{R}^{V \times D^p_h}$, where V represents the vocabulary size. During training the input image-sentence pair is given, therefore the prediction module can be implemented as a 1×1 convolutional kernel, which is faster than RNN-based models.

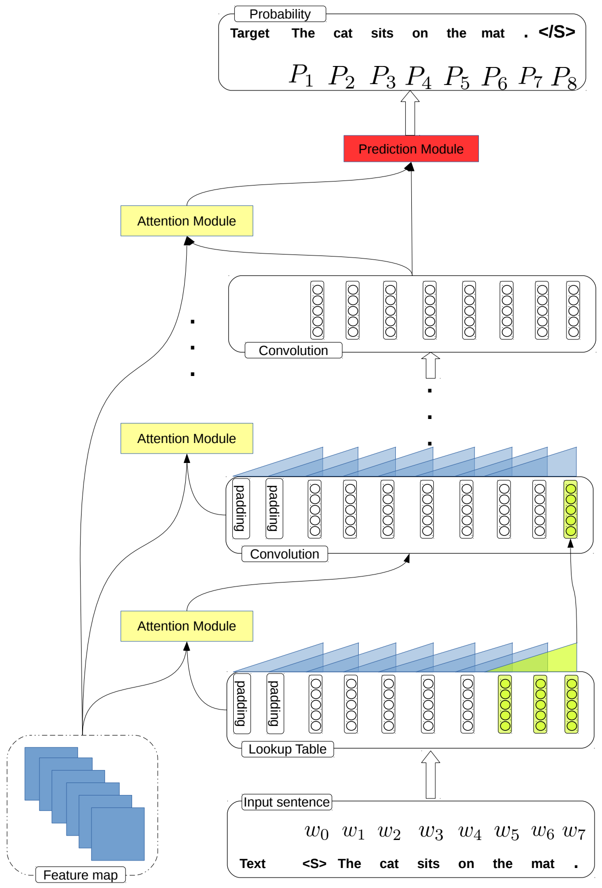


Figure 4. The hierarchical attention model. The structure of hierarchical attention moves up the layers, not to the right, so as to prevent sideways connections in the same layer (as in an RNN).

3.1. Hierarchical attention

The basic CNN+CNN model in the previous sub-section extracts attention features from the top-level of the language model. However, different levels of the language CNN represent different concept levels that could benefit from visual inputs. Hence, we also consider a hierarchical attention (hier-att) module (see Fig. 4), where attention vectors are computed at each level of the language model and fed into the next level,

$$a^{l-1}_j = \sum_{i=1}^{N} w^{l-1}_{i,j} v_i, \tag{9}$$
$$h^l_a = W^l_a * h^{l-1} + W^{att}_a * \mathbf{a}^{l-1} + b^l_a, \tag{10}$$
$$h^l_b = W^l_b * h^{l-1} + W^{att}_b * \mathbf{a}^{l-1} + b^l_b, \tag{11}$$
$$h^l = h^l_a \odot \sigma(h^l_b), \tag{12}$$

where $w_{i,j}^{l-1}$ is computed using equations (4-5) on the concepts $\mathbf{c}^{l-1}$at level $l-1$, $W_a^{att}, W_b^{att} \in \mathbb{R}^{D_e \times D_c}$ and $\mathbf{a}^{l-1} = [a_j^{l-1}]_{j=1}^{L}$.

In contrast to RNN-based models that calculate attention maps in a left-right (word-by-word) manner, our attention maps are calculated in a bottom-up (layer-by-layer) manner, so as to prevent sideways connections in the same layer. This allows our model to be trained in parallel over all words in the sentence, rather than word-by-word. Note that the model still sees attended features of previous words (from the lower layer) through the causal convolution layer.

3.2. Training and inference

During training, as image-caption pairs are given, the convolution structures are applied in the normal way, and the loss function for each sentence is the cross-entropy,

$$\mathcal{L} = -\sum_{j=0}^{L-1} y_{j+1} \log(P_{j+1}) + \frac{\lambda}{2} \sum_{W} ||W||^2, \qquad (13)$$

where y_{j+1} denotes the ground-truth label and P_{j+1} is the prediction probability. The first term is the same with the loss function in [31], and the second term is the regularization term, where W denotes the weights in our framework.

During inference, the caption is generated given the image using a feed-forward process. The caption is initialized as zero padding and a start-token <S>, and is fed as the input sentence to the model to predict the probability of the next word. The predicted word is appended to the caption, and the process is repeated until the ending token </S> is predicted, or the maximum length is reached. The predicted words are selected using a naive greedy method (most probable word), which is equivalent to a beam-search algorithm with beam width of 1.

4. Experiments

We present experiments using our proposed CNN+CNN model on three datasets.

4.1. Dataset and experimental setup

MSCOCO MSCOCO is the most popular dataset for image captioning, comprising 82,783 training and 40,504 validation images. Each image has 5 human annotated captions. As in [12, 34], we split the images into 3 datasets, consisting of 5,000 validation and 5,000 testing images, and 113,287 training images. Our vocabulary contains 10,000 words.

Flickr30k Flickr30k contains 31,783 images, each of which has 5 captions. Following [12], we use 1,000 images for validation and 1000 images for testing, and the remainder for training. The vocabulary size is 10,000.

Paragraph annotation dataset (PAD) A dataset with 19,551 images from MSCOCO and visual genome (VG). Each image is described by a paragraph comprising several sentences. The average length of the paragraph is 67.5 words, while that of MSCOCO dataset is 11.3. Generating such long captions is much more challenging. We use the same split as [15], and our vocabulary size is 5,000.

Implementation details For all datasets, we set $D_e = 300$. In our framework, the vision module is the VGG-16 network without fully connected layers, which is pretrained on ImageNet.[2] We use conv5_3 feature map to compute attention features ($D_c = 512$ and $d = 14$). The size of the hidden layer in the prediction module is 1024, and λ=1e-5. We apply Adam optimizer [14] with mini-batch size 10 to train our model. For VGG-16, we set the learning rate to 1e-5, and for other parameters, the learning rate starts from 1e-3 and decays every 50k steps.[3] The stopping criteria is based on the validation loss. The weights are initialized by the truncated normal initializer with $stddev = 0.01$. The model was implemented in TensorFlow.

During training time, the images are resized to 299×299 and then we randomly crop 224×224 patches as inputs. During testing time, the images are resize into 224×224 without cropping.[4] The maximum caption length is 70 for MSCOCO and Flickr30k, and 150 for PAD.

Compared models We compare our proposed model with and without hierarchical attention. We also compare other popular models: DeepVS [12], m-RNN [21], Google NIC [31], LRCN [7], hard-ATT and soft-ATT [34] on MSCOCO and Flickr30k datasets, and Sentence-Concat, DenseCap-Concat, Image-Flat, Regions-Flat-Scratch, Regions-Flat-Pretrained, and Regions-Hierarchical [15] on PAD.

Metrics We report the following metrics: **B**LEU-1,2,3,4 [23], **M**eteor [6], **R**ouge-L [17] and **CIDEr** [30], which are computed using the MSCOCO toolkit [18]. For all metrics, higher values indicate better performance.

4.2. Results on MSCOCO and Flickr30k

Table 1 shows the results on MSCOCO and Flickr30k. Note that all models only use the image information without semantics or attributes boosting. In terms of the B-1 score, which only considers bigrams, our model performs slightly worse than Hard- and Soft-ATT models, but better than NIC

[2]https://github.com/tensorflow/models/tree/master/research/slim

[3]For PAD, the learning rate decays every 10k steps.

[4]We employ the code provided by the following link: https://github.com/tensorflow/models/tree/master/research/im2txt

Dataset	Model	B-1	B-2	B-3	B-4	M	R	C
MSCOCO	DeepVS [12]	0.625	0.450	0.321	0.230	0.195	-	0.660
	m-RNN [21]	0.670	0.490	0.350	0.250	-	-	-
	NIC [31]	0.666	0.461	0.329	0.246	-	-	-
	LRCN [7]	0.697	**0.519**	**0.380**	**0.278**	0.229	0.508	0.837
	Hard-ATT [34]	**0.718**	0.504	0.357	0.250	0.230	-	-
	Soft-ATT [34]	0.707	0.492	0.344	0.243	**0.239**	-	-
	Ours (w/o hier-att)	0.688	0.513	0.370	0.265	0.234	0.507	0.839
	Ours (w/ hier-att)	0.685	0.511	0.369	0.267	0.234	**0.510**	**0.844**
Flickr30k	DeepVS [12]	0.573	0.369	0.240	0.157	0.153	-	0.247
	m-RNN [21]	0.60	0.41	0.28	0.19	-	-	-
	NIC [31]	0.663	0.423	0.277	0.183	-	-	-
	LRCN [7]	0.587	0.39	0.25	0.165	-	-	-
	Hard-ATT [34]	**0.669**	**0.439**	**0.296**	**0.199**	0.185	-	-
	Soft-ATT [34]	0.667	0.434	0.288	0.191	0.185	-	-
	Ours, w/o hier-att	0.577	0.401	0.276	0.190	0.184	0.425	0.352
	Ours, w/ hier-att	0.607	0.425	0.292	**0.199**	**0.191**	**0.442**	**0.395**

Table 1. Performance on MSCOCO and Flickr30k dataset. B, M, R, C stand for BLEU, Meteor, Rouge-L and CIDEr, respectively. The bold, red, and blue numbers are the highest value, 2nd highest and 3rd highest values, respectively.

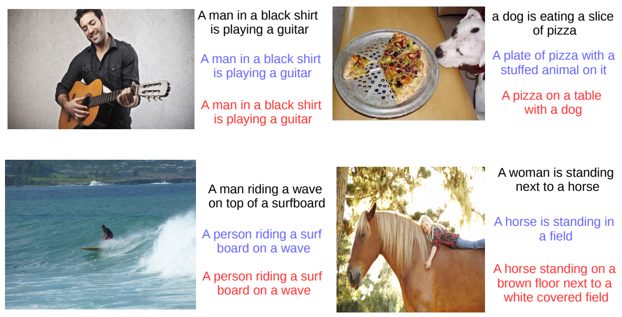

A man in a black shirt is playing a guitar
A man in a black shirt is playing a guitar
A man in a black shirt is playing a guitar
a dog is eating a slice of pizza
A plate of pizza with a stuffed animal on it
A pizza on a table with a dog
A man riding a wave on top of a surfboard
A person riding a surf board on a wave
A person riding a surf board on a wave
A woman is standing next to a horse
A horse is standing in a field
A horse standing on a brown floor next to a white covered field

Figure 5. Examples of generated descriptions. Black, blue and red text correspond to captions from NIC, our model, and our model with hier-att.

and m-RNN on MSCOCO. In terms of B-2,3,4 and M metrics, the performance of our model ranks 2nd, and the values are marginally lower than the LRCN model, which applies multilayer LSTMs to generate captions. This suggests that adopting CNNs as decoder is competitive or can be better than using LSTMs. When we apply our proposed hierarchical attention model, the B-4, M, R and CIDEr scores exhibit improvement. Using hier-att with our model improves the CIDEr score from 0.839 (no hier-att) to 0.844 and the Meteor score from 0.507 to 0.510.

On Flickr30k, our proposed model also provides the comparable results, and can slightly improve the M score compared with LSTM-based models. Moreover, our proposed hierarchical attention significantly improves all scores compared with the model without hierarchical attention. The CIDEr score is improved from 0.352 to 0.395 (12% increase), B-1 score is improved by 3% from 0.507 to 0.607 and other scores are improved by around 1%.

An advantage of our proposed CNN+CNN model is the training speed. We compare our 6-layer model without hierarchical attention with NIC. When we set the batch size to 32, and train our model and NIC on the same machine with a GTX1080 GPU, it takes 0.24 seconds per batch for our model, and 0.64 seconds for NIC. Our model is about $3\times$ faster than NIC for training.[5] After training 50k steps (each step uses one batch), the loss decreases by around 70%, and the model is able to output reasonable descriptions. In the inference stage, both our CNN+CNN model and LSTM-based model generate one word at a time, and thus, the inference speed is almost the same.

Note that the NIC model contains about 6.7M parameters, while our proposed model contains more than 16.0M parameters. Although our model is more complex, it can be trained faster because it can better take advantage of parallelization since there are no recurrent (left-right) connections between words within the same layer. Examples of generated descriptions are shown in Fig. 5.

4.3. Results on paragraph generation

We evaluate our model on the paragraph annotation dataset, in which an image has a much longer caption than MSCOCO and Flickr30k. Table 2 shows the results. Our proposed model exhibits comparable performance with Image-Flat when considering B-1,2,3,4, which is lower than the Regions-Hierarchical. Our model with kernel size 7 significantly improves the CIDEr score, which is specifically designed to evaluate image captioning models. Some examples generated by our 20-layer model with kernel size 7 are shown in Fig. 6.

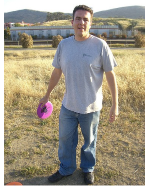
[5]NIC is the simplest among the LSTM-based models.

Model	B-1	B-2	B-3	B-4	M	R	C
Sentence-Concat*	0.311	0.151	0.076	0.040	0.120	-	0.068
DenseCap-Concat*	0.332	0.169	0.085	0.045	0.127	-	0.125
Image-Flat* [12]	0.340	0.200	0.122	0.077	0.128	-	0.111
Regions-Flat-Scratch*	0.373	0.217	0.131	0.081	0.135	-	0.111
Regions-Flat-Pretrained*	0.383	0.229	0.142	**0.090**	0.142	-	0.121
Regions-Hierarchical*	**0.419**	**0.241**	**0.142**	0.087	**0.160**	-	0.135
Ours, 20-layer, $k = 3$	0.350	0.202	0.118	0.068	0.140	0.263	0.106
Ours, 20-layer, $k = 5$	0.311	0.171	0.099	0.057	0.127	0.263	0.120
Ours, 20-layer, $k = 7$	0.350	0.194	0.107	0.059	0.133	0.259	**0.152**

Table 2. Results on the paragraph annotation dataset. The bold numbers represent the highest score. We use a 20-layer CNN with skip connections every 3 layers, and we do not use hierarchical attention for this task. *The results are from [15].

A man is standing on a baseball field. He is wearing a blue and white jersey uniform. A black helmet is the jersey , and his hat is red. His shoes are white and black. He is holding a black bat in his hand. The other man is wearing a cap , a gray shirt , suspenders and gloves. More of a chain link fence is surrounding the field. Tall trees can be seen in the background .

A brown horse with a black mane is standing in the carriage down a wooded area. There are trees in the background behind the man. Trees are visible in the distance behind the trees .


A man is standing on a grassy field outside on a sunny day. He is reaching for a white frisbee that is soaring in front of him. The man is wearing a shirt and blue jean shorts. Trees can be seen sitting near a sidewalk in the background. Part of a building can be seen behind the trees. A person is walking towards the sidewalk in front of the building .

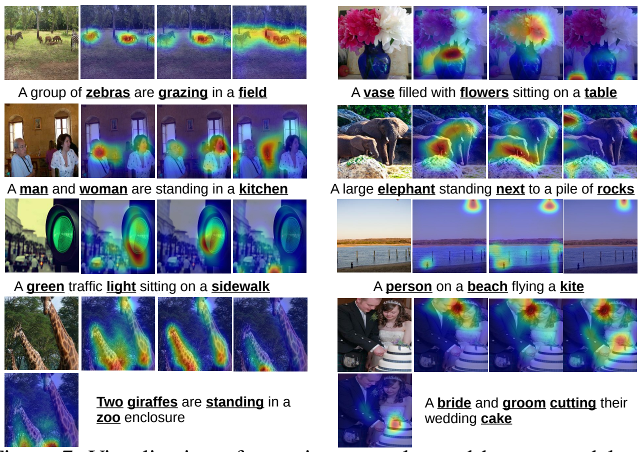
A bunch of cows are grazing in a field. There are a bunch of trees and trees surrounding the house. The sky is blue and has a few white buildings .

Figure 6. Examples of generated paragraph description.

4.4. Wider or Deeper?

In this experiment, we compare the performance of our model with different kernel widths and depths, with results shown in Table 3. Designing and training deeper networks is popular, yet in our experiments on image captioning it is unnecessary to use deeper network on these datasets. The results suggest that for MSCOCO and Flickr30k datasets 6-layer networks with kernel size 3, and hierarchical attention is the optimal choice.

For our proposed model without hierarchical attention, increasing the kernel width is a better choice both on MSCOCO and Flickr30k. When the kernel size increases from 2 to 7, the CIDEr score increases from 0.827 to 0.839 on MSCOCO, and from 0.342 to 0.352 on Flickr30k, which is better than the 20-layer network.


A group of **zebras** are **grazing** in a **field**

A **vase** filled with **flowers** sitting on a **table**

A **man** and **woman** are standing in a **kitchen**

A large **elephant** standing **next** to a pile of **rocks**

A **green** traffic **light** sitting on a **sidewalk**

A **person** on a **beach** flying a **kite**

Two giraffes are **standing** in a **zoo** enclosure

A **bride** and **groom** **cutting** their wedding **cake**

Figure 7. Visualization of attention maps learned by our model. The left-most images are the input images, and each bold-underline word corresponds to one attention map. Here we show nouns, verbs, prepositions and numbers.

Note that the average length of the captions in MSCOCO is 11.6, and the receptive field of a 6-layer CNN with kernel size 3 is 13, which is longer than many sentences in MSCOCO. Increasing the kernel width or the network depth increases the complexity of the model, which makes it is more difficult to train and easier to overfit. Based on our experiments, we suggest using $k = 3$ and setting the depth such that the receptive field is the average length of the sentences, which gives relatively good balance between the complexity and model performance.

4.5. Attention module analysis

We use the MSCOCO test images to visualize the attention maps and descriptions. The attention maps are visualized by upsampling the 14×14 attention maps with bilinear interpolation and Gaussian smoothing (see Fig. 7). Our proposed model is able to pay attention to correct area when predicting the corresponding words. However, it is difficult to quantify the attention module. E.g., when the model predicts the word "man" (second row in Fig. 7), it also pays attention to the woman, and vice versa. For the prepositions and verbs, our model attends to the correct area, *e.g.* when the word "next" is present, the areas around the elephant are

Dataset	#layers	k	#para	B-1	B-2	B-3	B-4	M	R	C
MSCOCO	6, w/o hier-att	2	16.0 M	0.679	0.504	0.363	0.262	0.231	0.505	0.827
		3	16.9 M	0.683	0.509	0.367	0.264	0.234	0.507	0.829
		5	18.7 M	0.687	0.511	0.368	0.264	0.233	0.509	0.838
		7	20.5 M	**0.688**	0.513	0.370	0.265	0.234	0.507	0.839
	6, w/ hier-att	2	16.0 M	0.682	0.508	0.368	0.266	0.233	0.508	0.836
		3	16.9 M	0.684	**0.514**	**0.372**	**0.269**	**0.235**	**0.511**	0.842
		5	18.7 M	0.685	0.511	0.369	0.267	0.234	0.510	**0.844**
		7	20.5 M	0.686	0.510	0.367	0.264	0.234	0.510	0.834
	20 w/o hier-att	3	24.5 M	0.683	0.505	0.365	0.264	0.234	0.505	0.838
Flickr30k	6, w/o hier-att	2	16.0 M	0.552	0.378	0.257	0.175	0.179	0.421	0.342
		3	16.9 M	0.577	0.396	0.267	0.181	0.180	0.426	0.350
		5	18.7 M	0.577	0.396	0.268	0.183	0.183	0.424	0.340
		7	20.5 M	0.577	0.401	0.276	0.190	0.184	0.425	0.352
	6, w/ hier-att	2	16.0 M	0.581	0.403	0.277	0.190	0.182	0.433	0.368
		3	16.9 M	**0.607**	**0.425**	**0.292**	**0.199**	**0.191**	**0.442**	**0.395**
		5	18.7 M	0.588	0.402	0.270	0.185	0.183	0.428	0.352
		7	20.5 M	0.593	0.412	0.282	0.193	0.187	0.437	0.386
	20 w/o hier-att	3	24.5 M	0.589	0.411	0.284	0.198	0.175	0.425	0.320

Table 3. The influence of the kernel width and number of layers. #layers denotes the number of layers, k denotes the width of kernel, and #para is the number of parameters of the model, which reflects the complexity of the model.

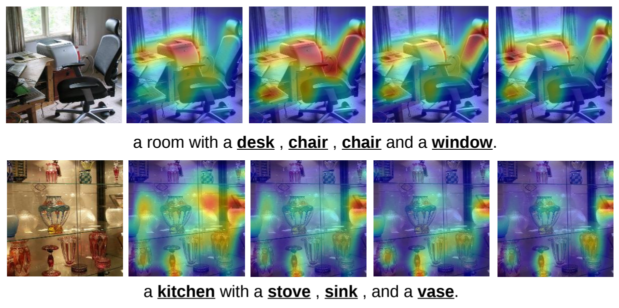

a room with a desk , chair , chair and a window.
a kitchen with a stove , sink , and a vase.

Figure 8. Example failure cases. Scenes with many objects are challenging for the visual attention model.

highlighted.

When there are many objects in an image, although the model can pay attention to the areas that contain objects, it predicts the wrong words or repeats a words several times, as seen in Fig. 8. The top row shows that the model predicts the word "chair" two times, and the attention maps are similar. The bottom row shows that although the objects are attended, the model predicts the wrong words "stove" and "sink", which are likely influenced by the preceding context "a kitchen with a".

We also visualize our model with hierarchical attention in Fig. 9, which shows the attention maps at the top-level for the predicted words. The model with hierarchical attention tends to pay attention to small areas at the top level, and for the scenes with many objects it is able to focus on the relatively correct areas, which is similar to the coarse-to-fine procedure. Compared with the failure cases in Fig. 8, using hierarchical attention provides more details, which could benefit caption generation.

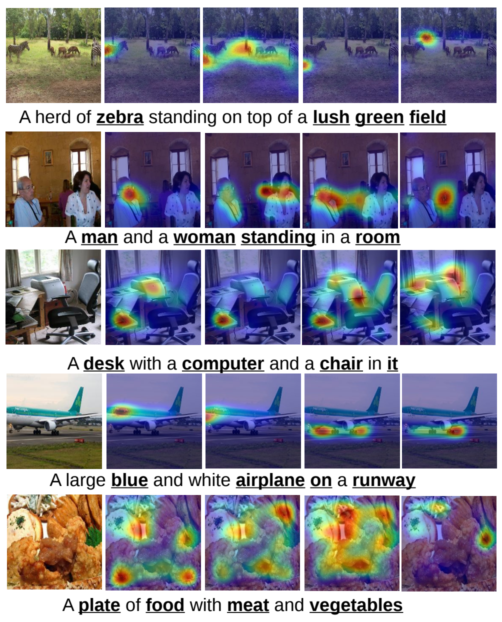

A herd of zebra standing on top of a lush green field
A man and a woman standing in a room
A desk with a computer and a chair in it
A large blue and white airplane on a runway
A plate of food with meat and vegetables

Figure 9. Visualization of the attention maps at the top-level of our hierarchical attention. The hierarchical attention can handle many-object scenes.

5. Conclusions

We have developed a CNN+CNN framework for image captioning and explored the influence of the kernel width and the layer depth of the language CNN. We have shown that the ability of the CNN-based framework is competitive to LSTM-based models, but can be trained faster. We also visualize the learned attention maps to show that the model is able to learn concepts and pay attention to the corresponding areas in the images in a meaningful way.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015. 2

[2] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, and T.-S. Chua. Sca-cnn: spatial and channel-wise attention in convolutional networks for image captioning. *CVPR*, 2017. 1

[3] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *JMLR*, 12:2493–2537, 2011. 2

[4] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun. Very deep convolutional networks for text classification. *EACL*, 2017. 2

[5] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. *arXiv*, 2016. 2

[6] M. Denkowski and A. Lavie. Meteor universal: Language specific translation evaluation for any target language. *EACL 2014 Workshop on Statistical Machine Translation*, 2014. 5

[7] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, , and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. *CVPR*, 2015. 3, 5, 6

[8] H. Fang, S. Gupta, F. Iandola, R. K. Srivastava, L. Deng, P. Dollar, J. Gao, X. He, M. Mitchell, J. C. Platt, C. L. Zitnick, and G. Zweig. From captions to visual concepts and back. *CVPR*, 2015. 1

[9] Z. Gan, C. Gan, X. He, Y. Pu, K. Tran, J. Gao, L. Carin, and L. Deng. Semantic compositional networks for visual captioning. *CVPR*, 2017. 1, 2

[10] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. *arXiv*, 2017. 2

[11] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. van den Oord, A. Graves, and K. Kavukcuoglu. Neural machine translation in linear time. *arXiv*, 2016. 2

[12] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *CVPR*, 2015. 5, 6, 7

[13] Y. Kim. Convolutional neural networks for sentence classification. *EMNLP*, 2014. 2

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 5

[15] J. Krause, J. Johnson, R. Krishna, and L. Fei-Fei. A hierarchical approach for generating descriptive image paragraphs. *CVPR*, 2017. 2, 5, 7

[16] G. Kulkarni, V. Premraj, V. Ordonez, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg. Babytalk: understanding and generating simple image descriptions. *IEEE Trans. on PAMI*, 35(12):2891–2903, 2013. 1

[17] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. *ACL 2004 Workshop*, 2004. 5

[18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick. Microsoft coco: Common objects in context. *ECCV*, 2014. 5

[19] J. Lu, C. Xiong, D. Parikh, and R. Socher. Knowing when to look: adaptive attention via a visual sentinel for image captioning. *CVPR*, 2017. 1, 2

[20] L. Ma, Z. Lu, L. Shang, and H. Li. Multimodal convolutional neural networks for matching image and sentence. *ICCV*, 2015. 1, 2

[21] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *ICLR*, 2015. 1, 2, 3, 5, 6

[22] Z. Niu, M. Zhou, L. Wang, X. Gao, and G. Hua. Hierarchical multimodal lstm for dense visual-semantic embedding. *ICCV*, 2017. 1

[23] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. *ACL*, 2002. 5

[24] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015. 3

[25] R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng. Parsing with compositional vector grammars. *ACL*, 2013. 1

[26] R. Socher, C. Lin, A. Y. Ng, and C. D. Manning. Parsing natural scenes and natural language with recursive neural networks. *ICML*, 2011. 1

[27] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *NIPS*, 2014. 2

[28] Y. H. Tan and C. S. Chan. phi-lstm: A phrase-based hierarchical lstm model for image captioning. *ACCV*, 2016. 2

[29] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv*, 2016. 2

[30] R. Vedantam, C. L. Zitnick, and D. Parikh. Cider: Consensus-based image description evaluation. *CVPR*, 2015. 5

[31] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. *CVPR*, 2015. 1, 2, 3, 5, 6

[32] Y. Wang, Z. Lin, X. Shen, S. Cohen, and G. W. Cottrell. Skeleton key: Image captioning by skeleton-attribute decomposition. *CVPR*, 2017. 2

[33] Q. Wu, C. Shen, L. Liu, A. Dick, , and A. v. d. Hengel. What value do explicit high level concepts have in vision to language problems? *CVPR*, 2016. 1, 2

[34] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *ICML*, 2015. 1, 2, 3, 5, 6

[35] T. Yao, Y. Pan, Y. Li, Z. Qiu, and T. Mei. Boosting image captioning with attributes. *ICCV*, 2017. 1, 2

[36] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo. Image captioning with semantic attention. *CVPR*, 2016. 1, 2