What is the Role of Recurrent Neural Networks (RNNs) in an Image Caption Generator?

Marc Tanti **Albert Gatt**
Institute of Linguistics
and Language Technology
University of Malta
marc.tanti.06@um.edu.mt
albert.gatt@um.edu.mt

Kenneth P. Camilleri
Deptartment of Systems
and Control Engineering
University of Malta
kenneth.camilleri@um.edu.mt

Abstract

In neural image captioning systems, a recurrent neural network (RNN) is typically viewed as the primary 'generation' component. This view suggests that the image features should be 'injected' into the RNN. This is in fact the dominant view in the literature. Alternatively, the RNN can instead be viewed as only encoding the previously generated words. This view suggests that the RNN should only be used to encode linguistic features and that only the final representation should be 'merged' with the image features at a later stage. This paper compares these two architectures. We find that, in general, late merging outperforms injection, suggesting that RNNs are better viewed as encoders, rather than generators.

1 Introduction

Image captioning (Bernardi et al., 2016) has emerged as an important testbed for solutions to the fundamental AI challenge of grounding symbolic or linguistic information in perceptual data (Harnad, 1990; Roy and Reiter, 2005). Most captioning systems focus on what Hodosh et al. (2013) refer to as *concrete conceptual* descriptions, that is, captions that describe what is strictly within the image, although recently, there has been growing interest in moving beyond this, with research on visual question-answering (Antol et al., 2015) and image-grounded narrative generation (Huang et al., 2016) among others.

Approaches to image captioning can be divided into three main classes (Bernardi et al., 2016):

1. Systems that rely on computer vision techniques to extract object detections and features from the source image, using these as input to an NLG stage (Kulkarni et al., 2011; Mitchell et al., 2012; Elliott and Keller, 2013). The latter is roughly akin to the microplanning and realisation modules in the well-known NLG pipeline architecture (Reiter and Dale, 2000).

2. Systems that frame the task as a retrieval problem, where a caption, or parts thereof, is identified by computing the proximity/relevance of strings in the training data to a given image. This is done by exploiting either a unimodal (Ordonez et al., 2011; Gupta et al., 2012; Mason and Charniak,) or multimodal (Hodosh et al., 2013; Socher et al., 2014) space. Many retrieval-based approaches rely on neural models to handle both image features and linguistic information (Ordonez et al., 2011; Socher et al., 2014).

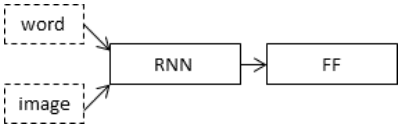
3. Systems that also rely on neural models, but rather than performing partial or wholesale caption retrieval, generate novel captions using a recurrent neural network (RNN), usually a long short-term memory (LSTM). Typically, such models use image features extracted from a pre-trained convolutional neural network (CNN) such as the VGG CNN (Simonyan and Zisserman, 2014) to bias the RNN towards sampling terms from the vocabulary in such a way that a sequence of such terms produces a caption that is relevant to the image (Kiros et al., 2014b; Kiros et al., 2014a; Vinyals et

al., 2015; Mao et al., 2015a; Hendricks et al., 2016).

This paper focuses on the third class. The key property of these models is that the CNN image features are used to condition the predictions of the best caption to describe the image. However, this can be done in different ways and the role of the RNN depends in large measure on the mode in which CNN and RNN are combined.

It is quite typical for RNNs to be viewed as 'generators'. For example, Bernardi et al. (2016) suggest that 'the RNN is trained to generate the next word [of a caption]', a view also expressed by LeCun et al. (2015). A similar position has also been taken in work focusing on the use of RNNs as language models for generation (Sutskever et al., ; Graves, 2013). However, an alternative view is possible, whereby the role of the RNN can be thought of as primarily to encode sequences, but not directly to generate them.

(a) Conditioning by injecting the image means injecting the image into the same RNN that processes the words.



(b) Conditioning by merging the image means merging the image with the final state of the RNN in a "multimodal layer" after processing the words.

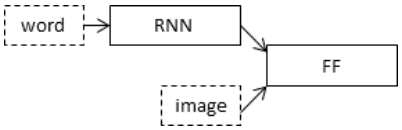


Figure 1: The inject and merge architectures for caption generation. The RNN's previous state going into the RNN is not shown. Legend: RNN - Recurrent Neural Network; FF - Feed Forward layer.

These two views can be associated with different architectures for neural caption generators, which we discuss below and illustrated in Figure 1. In one class of architectures, image features are directly incorporated into the RNN during the sequence encoding process (Figure 1a). In these models, it is natural to think of the RNN as the primary generation component of the image captioning system, making predictions conditioned by the image. A different architecture keeps the encoding of linguistic and perceptual features separate, merging them in a later multimodal layer, at which point predictions are made (Figure 1b). In this type of model, the RNN is functioning primarily as an encoder of sequences of word embeddings, with the visual features merged with the linguistic features in a later, multimodal layer. This multimodal layer is the one that drives the generation process since the RNN never sees the image and hence would not be able to direct the generation process.

While both architectural alternatives have been attested in the literature, their implications have not, to our knowledge, been systematically discussed and comparatively evaluated. In what follows, we first discuss the distinction between the two architectures (Section 2) and then present some experiments comparing the two (Sections 3 and 4). Our conclusion is that grounding language generation in image data is best conducted in an architecture that first encodes the two modalities separately, before merging them to predict captions.

2 Background: Neural Caption Generation Architectures

In a neural language model, an RNN encodes a prefix (for example, the caption generated so far) and either itself predicts the next item in the sequence with the help of a feed forward layer or else it passes the encoding to the next layer which will make the prediction itself. This new item is added to the prefix at the next iteration to predict another item, until an end-of-sequence symbol is reached. Typically, the prediction is carried out using a softmax function to sample the next item according to a probability distribution over the vocabulary items, based on their activation. This process is illustrated in Figure 2.

One way to condition the RNN to predict image captions is to inject both visual and linguistic features directly into the RNN, depicted in Figure 1a. We refer to this as 'conditioning-by-inject' (or *inject* for short). Different types of inject architectures have become the most widely attested among deep learning approaches to image captioning (Chen and Zitnick, 2015; Donahue et al., 2015; Hessel et al., 2015; Karpathy and Fei-Fei, 2015; Liu et al., 2016;

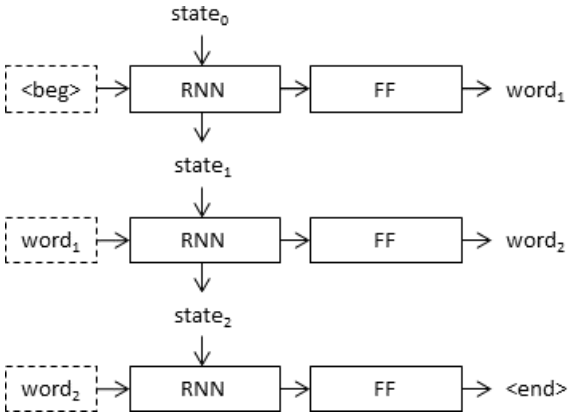


Figure 2: How RNNs work: each state of the RNN encodes a prefix, which incorporates the output word derived from the previous state. In practice the neural network does not output a single word but a probability distribution over all known words in the vocabulary. Legend: FF - feedforward layer; <beg> - the start-of-sentence token; <end> - the end-of-sentence token.

Yang et al., 2016; Zhou et al., 2016).[1] Given training pairs consisting of an image and a caption, the RNN component of such models is trained by exposure to prefixes of increasing length extracted from the caption, in tandem with the image.

An alternative architecture – which we refer to as 'conditioning-by-merge' (Figure 1b) – treats the RNN exclusively as a 'language model' to encode linguistic sequences of varying length. The linguistic vector resulting from this encoding is subsequently combined with the image features in a separate multimodal layer. This amounts to viewing the RNN as primarily an encoder of linguistic information. This type of architecture is also attested in the literature, albeit to a lesser extent than the inject architecture (Mao et al., 2014; Mao et al., 2015a; Mao et al., 2015b; Song and Yoo, 2016; Hendricks et al., 2016; You et al., 2016). A limited number of approaches have also been proposed in which both architectures are combined (Lu et al., 2016; Xu et al., 2015).

Notice that both architectures are compatible with the inclusion of attentional mechanisms (Xu et al., 2015). The effect of attention in the inject architecture is to combine a different representation of the image with each word. In the case of merge, a different representation of the image can be combined with the final RNN state before each prediction. Attentional mechanisms are however beyond the scope of the present work.

The main differences between inject and merge architectures can be summed up as follows: In an inject model, the RNN is trained to predict sequences based on histories consisting of both linguistic and perceptual features. Hence, in this model, the RNN is primarily responsible for image-conditioned language generation. By contrast, in the merge architecture, RNNs in effect encode linguistic *representations*, which themselves constitute the input to a later prediction stage that comes after a multimodal layer. It is only at this late stage that image features are used to condition predictions.

As a result, a model involving conditioning by inject is trained to learn linguistic representations directly conditioned by image data; a merge architecture maintains a distinction between the two representations, but brings them together in a later layer.

Put somewhat differently, it could be argued that at a given time step, the merge architecture predicts what to generate next by combining the RNN-encoded prefix of the string generated so far (the 'past' of the generation process) with non-linguistic information (the guide of the generation process). The inject architecture on the other hand uses the full image features with every word of the prefix during training, in effect learning a 'visuo-linguistic' representation of each word. One effect of this is that image features can serve to further specify or disambiguate the 'meaning' of words, by disambiguating tokens of the same word which are correlated with different image features (such as 'crane' as in the bird versus the construction equipment). This implies that inject models learn a larger vocabulary during training.

The two architectures also differ in the number of parameters they need to handle. As noted above, since an inject architecture combines the image with each word during training, it is effectively handling a larger vocabulary than merge. Assume that the image vectors are concatenated with the word embedding vectors (inject) or the final RNN state (merge). Then, in the inject architecture, the number

[1] See Tanti et al. (2017) for an overview of different versions of the inject architecture and a systematic comparison among models. In this paper we focus on parallel-inject.

of weights in the RNN is a function of both the caption embedding and the images, whereas in merge, it is only the word embeddings that contribute to the size of this layer of the network. Let e be the size of the word embedding, v the size of the vocabulary, i the image vector size and s the state size of the RNN. In the inject case, the number of weights in the RNN is $w \propto (e+i) \times s$, whereas it is $w \propto e \times s$ in merge. The smaller number of weights handled by the RNN in merge is offset by a larger number of weights at the final softmax layer, which has to take as input the RNN state and the image, having size $\propto (s+i) \times v$.

A systematic comparison of these two architectures would shed light on the best way to conceive of the role of RNNs in neural language generation. Apart from the theoretical implications concerning the stage at which language should be grounded in visual information, such a comparison also has practical implications. In particular, if it turns out that merge outperforms inject, this would imply that the linguistic representations encoded in an RNN could be pre-trained and re-used for a variety of tasks and/or image captioning datasets, with domain-specific training only required for the final feedforward layer, where the tuning required to make perceptually grounded predictions is carried out. We return to this point in Section 6.1.

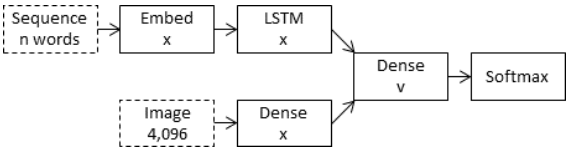
In the following sections, we describe some experiments to conduct such a comparison.

3 Experiments

To evaluate the performance of the inject and merge architectures, and thus the roles of the RNN, we trained and evaluated them on the Flickr8k (Hodosh et al., 2013) and Flickr30k (Young et al., 2014) datasets of image-caption pairs. For the purposes of these experiments, we used the version of the datasets distributed by Karpathy and Fei-Fei (2015)[2]. The dataset splits are identical to that used by Karpathy and Fei-Fei (2015): Flickr8k is split into 6,000 images for training, 1,000 for validation, and 1,000 for testing whilst Flickr30k is split into 29,000 images for training, 1,014 images for validation, and 1,000 images for testing. Each image

[2] http://cs.stanford.edu/people/karpathy/deepimagesent/

(a) The merge architecture.



(b) The inject architecture.

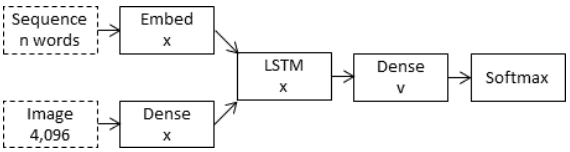


Figure 3: An illustration of the different architectures that are tested in this paper. The numbers or letters at the bottom of each box refer to the vector size output of a layer. 'x' is an arbitrary layer size that is varied in the experiments and 'v' is the vocabulary size which is also varied in the experiments. 'Dense' means fully connected layer with bias.

in both datasets has five different captions. 4,096-element image feature vectors that were extracted from the pre-trained VGG CNN (Simonyan and Zisserman, 2014) are also available in the distributed datasets. We normalised the image vectors to unit length during preprocessing.

Tokens with frequency lower than a threshold in the training set were replaced with the 'unknown' token. In our experiments we varied the threshold between 3 and 5 in order to measure the performance of each model as vocabulary size changes. For thresholds of 3, 4, and 5, this gives vocabulary sizes of 2,539, 2,918, and 3,478 for Flickr8k and 7,415, 8,275, 9,584 and for Flickr30k.

Since our purpose is to compare the performance of architectures, we used the 'barest' models possible, with the fewest number of hyperparameters. This means that complexities that are usually introduced in order to reach state-of-the-art performance, such as regularization, were avoided, since it is difficult to determine which combination of hyperparameters do not give an unfair advantage to one architecture over the other.

We constructed a basic neural language model consisting of a word embedding matrix, a basic LSTM (Hochreiter and Schmidhuber, 1997), and a

softmax layer. The LSTM is defined as follows:

$$i_n = sig(x_n W_{xi} + s_{n-1} W_{si} + b_i) \quad (1)$$
$$f_n = sig(x_n W_{xf} + s_{n-1} W_{sf} + b_f) \quad (2)$$
$$o_n = sig(x_n W_{xo} + s_{n-1} W_{so} + b_o) \quad (3)$$
$$g_n = \tanh(x_n W_{xc} + s_{n-1} W_{sc} + b_c) \quad (4)$$
$$c_n = f_n \odot c_{n-1} + i_n \odot g_n \quad (5)$$
$$s_n = o_n \odot \tanh(c_n) \quad (6)$$

where x_n is the n^{th} input, s_n is the hidden state after n inputs, s_0 is the all-zeros vector, c_n is the cell state after n inputs, c_0 is the all-zeros vector, i_n is the input gate after n inputs, f_n is the forget gate after n inputs, o_n is the output gate after n inputs, i_n is the input gate after n inputs, g_n is the modified input used to calculate c_n after n inputs, $W_{\alpha\beta}$ is the weight matrix between α and β, b_α is the bias vector for α, $\odot$ is the elementwise vector multiplication operator, and 'sig' refers to the sigmoid function. The hidden state and the cell state always have the same size.

In the experiments, this basic neural language model is used as a part of two different architectures: In the inject architecture, the image vector is concatenated with each of the word vectors in a caption. In the merge architecture, it is only concatenated with the final LSTM state. The layer sizes of the embedding, LSTM state, and projected image vector were also varied in the experiments in order to measure the effect of increasing the capacity of the networks. The layer sizes used are 128, 256, and 512. The details of the architectures used in the experiments are illustrated in Figure 3.

Training was performed using the Adam optimisation algorithm (Kingma and Ba, 2014) with default hyperparameters and a minibatch size of 50 captions. The cost function used was sum cross-entropy. Training was carried out with an early stopping criterion which terminated training as soon as performance on the validation data started to deteriorate (validation performance is measured after each training epoch). Initialization of weights was done using Xavier initialization (Glorot and Bengio, 2010) and biases were set to zero.

Each architecture was trained three separate times; the results reported below are averages over these three separate runs.

To evaluate the trained models we generated captions for images in the test set using beam search with a beam width of 3 and a clipped maximum length of 20 words. The MSCOCO evaluation code[3] was used to measure the quality of the captions by using the standard evaluation metrics BLEU-(1,2,3,4) (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), CIDEr (Vedantam et al., 2015), and ROUGE-L (Lin and Och, 2004). We also calculated the percentage of word types that were actually used in the generated captions out of the vocabulary of available word types. This measure indicates how well each architecture exploits the vocabulary it is trained on.

The code used for the experiments was implemented with TensorFlow and is available online[4].

4 Results

Table 1 reports means and standard deviations over the three runs of all the MSCOCO measures and the vocabulary usage. Since the point is to compare the effects of the architectures rather than to reach state-of-the-art performance, we do not include results from other published systems in our tables.

Across all experimental variables (dataset, vocabulary, and layer sizes), the performance of the merge architecture is generally superior to that of the inject architecture in all measures except for ROUGE-L and BLEU (ROUGE-L is designed for evaluating text summarization whilst BLEU is criticized for its lack of correlation with human-given scores). In what follows, we focus on the CIDEr measure for caption quality as it was specifically designed for captioning systems.

Although merge outperforms inject by a rather narrow margin, the low standard deviation over the three training runs suggests that this is a consistent performance advantage across train-and-test runs. In any case, there is clearly no disadvantage to the merge strategy with respect to injecting image features.

One peculiarity is that results on Flickr8k are better than those on Flickr30k. This could mean that Flickr8k captions contain less variation, hence are easier to perform well on. Preliminary results on the larger dataset MSCOCO (Lin et al., 2014) (currently in progress) show CIDEr results over 0.7

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

[4] https://github.com/mtanti/rnn-role

		% Vocabulary		CIDEr		METEOR		ROUGE-L	
Layer	**Vocab.**	**Merge**	Inject	**Merge**	**Inject**	**Merge**	**Inject**	**Merge**	**Inject**
128	2539	**14.730 (0.40)**	10.555 (0.34)	**0.460 (0.01)**	0.431 (0.01)	**0.192 (0.00)**	0.183 (0.00)	**0.445 (0.00)**	0.430 (0.00)
128	2918	**13.719 (0.49)**	8.876 (0.24)	**0.456 (0.00)**	0.431 (0.00)	**0.191 (0.00)**	0.185 (0.00)	**0.437 (0.00)**	0.434 (0.00)
128	3478	**11.223 (0.35)**	8.175 (0.31)	**0.458 (0.01)**	0.433 (0.01)	**0.192 (0.00)**	0.187 (0.00)	**0.442 (0.00)**	0.432 (0.00)
256	2539	**15.439 (0.84)**	11.448 (0.71)	**0.462 (0.01)**	0.456 (0.01)	**0.192 (0.00)**	0.189 (0.00)	**0.439 (0.00)**	0.436 (0.00)
256	2918	**13.697 (0.19)**	10.430 (0.34)	**0.456 (0.01)**	0.451 (0.01)	**0.190 (0.00)**	0.189 (0.00)	0.438 (0.00)	**0.440 (0.00)**
256	3478	**11.252 (0.51)**	8.405 (0.39)	**0.470 (0.01)**	0.449 (0.02)	**0.191 (0.00)**	0.189 (0.00)	**0.439 (0.00)**	0.437 (0.00)
512	2539	**15.741 (0.40)**	12.761 (0.81)	0.452 (0.01)	**0.464 (0.00)**	0.191 (0.00)	**0.192 (0.00)**	0.437 (0.00)	**0.442 (0.00)**
512	2918	**13.114 (0.75)**	10.155 (0.42)	**0.469 (0.01)**	0.457 (0.00)	**0.193 (0.00)**	0.189 (0.00)	**0.440 (0.00)**	0.437 (0.00)
512	3478	**11.501 (0.49)**	8.587 (0.50)	**0.458 (0.01)**	0.439 (0.01)	**0.192 (0.00)**	0.188 (0.00)	**0.439 (0.00)**	0.434 (0.00)

(a) Flickr8k: % of vocabulary used, CIDEr, METEOR and ROUGE-L results.

		BLEU-1		BLEU-2		BLEU-3		BLEU-4	
Layer	**Vocab.**	**Merge**	**Inject**	**Merge**	**Inject**	**Merge**	**Inject**	**Merge**	**Inject**
128	2539	**0.600 (0.00)**	0.592 (0.01)	**0.410 (0.00)**	0.405 (0.01)	**0.272 (0.00)**	0.270 (0.01)	**0.179 (0.00)**	0.177 (0.00)
128	2918	**0.595 (0.01)**	0.590 (0.00)	0.405 (0.01)	**0.406 (0.00)**	0.267 (0.01)	**0.271 (0.00)**	0.175 (0.00)	**0.178 (0.00)**
128	3478	**0.608 (0.01)**	0.586 (0.01)	**0.416 (0.01)**	0.401 (0.01)	**0.276 (0.01)**	0.268 (0.01)	**0.182 (0.01)**	0.178 (0.01)
256	2539	**0.594 (0.00)**	0.591 (0.00)	0.407 (0.01)	**0.408 (0.00)**	0.269 (0.01)	**0.276 (0.00)**	0.176 (0.01)	**0.184 (0.00)**
256	2918	**0.596 (0.01)**	0.596 (0.01)	0.405 (0.01)	**0.413 (0.01)**	0.265 (0.00)	**0.278 (0.01)**	0.172 (0.00)	**0.184 (0.00)**
256	3478	**0.601 (0.00)**	0.596 (0.01)	**0.411 (0.00)**	0.409 (0.01)	0.272 (0.01)	**0.274 (0.01)**	0.179 (0.01)	**0.181 (0.01)**
512	2539	0.597 (0.01)	**0.603 (0.00)**	0.406 (0.01)	**0.419 (0.00)**	0.267 (0.01)	**0.283 (0.00)**	0.176 (0.01)	**0.188 (0.00)**
512	2918	**0.593 (0.01)**	0.589 (0.01)	0.404 (0.01)	**0.409 (0.00)**	0.268 (0.00)	**0.277 (0.00)**	0.177 (0.00)	**0.185 (0.00)**
512	3478	**0.597 (0.01)**	0.587 (0.00)	**0.407 (0.01)**	0.405 (0.00)	0.270 (0.01)	**0.272 (0.00)**	0.178 (0.00)	**0.180 (0.01)**

(b) Flickr8k: BLEU-n scores.

		% Vocabulary		CIDEr		METEOR		ROUGE-L	
Layer	**Vocab.**	**Merge**	Inject	**Merge**	**Inject**	**Merge**	**Inject**	**Merge**	**Inject**
128	7415	**6.253 (0.06)**	5.255 (0.02)	**0.362 (0.01)**	0.339 (0.01)	**0.174 (0.00)**	0.169 (0.00)	**0.417 (0.00)**	0.415 (0.00)
128	8275	**5.402 (0.20)**	4.939 (0.08)	**0.376 (0.00)**	0.351 (0.00)	**0.174 (0.00)**	0.171 (0.00)	**0.420 (0.00)**	0.417 (0.00)
128	9584	**4.793 (0.01)**	4.090 (0.18)	**0.378 (0.00)**	0.355 (0.00)	**0.175 (0.00)**	0.171 (0.00)	**0.420 (0.00)**	0.419 (0.00)
256	7415	**6.150 (0.18)**	5.597 (0.11)	**0.363 (0.00)**	0.361 (0.01)	**0.174 (0.00)**	0.173 (0.00)	0.414 (0.00)	**0.420 (0.00)**
256	8275	**5.559 (0.08)**	5.410 (0.10)	**0.364 (0.01)**	0.359 (0.00)	**0.174 (0.00)**	0.173 (0.00)	0.416 (0.00)	**0.417 (0.00)**
256	9584	**4.873 (0.07)**	4.309 (0.18)	**0.364 (0.01)**	0.359 (0.01)	**0.175 (0.00)**	0.173 (0.00)	0.416 (0.00)	**0.420 (0.00)**
512	7415	**6.330 (0.56)**	5.732 (0.32)	0.365 (0.01)	**0.367 (0.01)**	**0.173 (0.00)**	0.173 (0.00)	0.416 (0.00)	**0.422 (0.01)**
512	8275	**5.619 (0.09)**	5.221 (0.49)	**0.370 (0.00)**	0.369 (0.01)	**0.174 (0.00)**	0.174 (0.00)	0.419 (0.00)	**0.422 (0.00)**
512	9584	**4.887 (0.16)**	4.309 (0.25)	0.357 (0.01)	**0.360 (0.01)**	0.172 (0.00)	**0.172 (0.00)**	0.414 (0.00)	**0.417 (0.00)**

(c) Flickr30k: % of vocabulary used, CIDEr, METEOR and ROUGE-L results.

		BLEU-1		BLEU-2		BLEU-3		BLEU-4	
Layer	**Vocab.**	**Merge**	**Inject**	**Merge**	**Inject**	**Merge**	**Inject**	**Merge**	**Inject**
128	7415	**0.601 (0.01)**	0.595 (0.01)	**0.403 (0.01)**	0.400 (0.01)	**0.268 (0.01)**	0.265 (0.01)	**0.179 (0.01)**	0.175 (0.01)
128	8275	**0.605 (0.01)**	0.604 (0.00)	**0.411 (0.01)**	0.409 (0.00)	**0.276 (0.01)**	0.275 (0.00)	**0.185 (0.00)**	0.183 (0.00)
128	9584	**0.610 (0.01)**	0.605 (0.00)	**0.414 (0.01)**	0.411 (0.00)	**0.278 (0.00)**	0.275 (0.01)	**0.186 (0.00)**	0.184 (0.01)
256	7415	0.593 (0.01)	**0.606 (0.00)**	0.400 (0.01)	**0.412 (0.00)**	0.268 (0.01)	**0.277 (0.00)**	0.179 (0.01)	**0.186 (0.01)**
256	8275	0.594 (0.01)	**0.603 (0.01)**	0.402 (0.01)	**0.409 (0.00)**	0.269 (0.01)	**0.275 (0.00)**	0.180 (0.00)	**0.183 (0.00)**
256	9584	0.596 (0.01)	**0.614 (0.01)**	0.404 (0.00)	**0.419 (0.01)**	0.270 (0.00)	**0.283 (0.00)**	0.181 (0.00)	**0.189 (0.00)**
512	7415	0.598 (0.02)	**0.617 (0.01)**	0.404 (0.02)	**0.422 (0.01)**	0.270 (0.01)	**0.285 (0.00)**	0.181 (0.01)	**0.191 (0.00)**
512	8275	0.603 (0.00)	**0.609 (0.01)**	0.406 (0.00)	**0.419 (0.01)**	0.271 (0.00)	**0.284 (0.01)**	0.181 (0.00)	**0.191 (0.00)**
512	9584	0.596 (0.00)	**0.609 (0.01)**	0.399 (0.00)	**0.414 (0.01)**	0.265 (0.00)	**0.278 (0.01)**	0.177 (0.00)	**0.185 (0.00)**

(d) Flickr30k: BLEU-n scores.

Table 1: Results on the captions generated using the inject and merge architectures. Values are means over three separately retrained models, together with the standard deviation in parentheses. Legend: Layer - the layer size used ('x' in Figure 3); Vocab. - the vocabulary size used.

which means that either Flickr8k is too easy or Flickr30k is too hard when compared to the much larger MSCOCO.

The best-performing models are merge with state size of 256 on Flickr8k, and merge with state size 128 on Flickr30k, both with minimum token fre-

quency threshold of 3. Inject models tend to improve with increasing state size, on both datasets, while the relationship between the performance of merge and the state size shows no discernible trend. Inject therefore does not seem to overfit as state size increases, even on the larger dataset. At the same time, inject only seems to be able to outperform the best scores achieved by merge if it has a much larger layer size. Therefore, in practical terms, inject models have to have larger capacity to be at par with merge. Put differently, merge has a higher performance to model size ratio and makes more efficient use of limited resources (this observation holds even when model size is defined in terms of number of parameters instead of layer sizes).

Given the same layer sizes and vocabulary, the number of parameters for merge is greater than for inject. The difference becomes greater as the vocabulary size is increased. For a vocabulary size of 2,539 and layer size of 512, merge has about 3% more parameters than inject whilst for a vocabulary size of 9,584 and layer size of 512, merge has about 20% more parameters. However, the foregoing remarks concerning over- and under-fitting also apply when the difference between the number of parameters is small. That is, the difference in performance is due at least in part to architectural differences, not just to differences in number of parameters.

Merge models use a greater proportion of the training vocabulary on test captions. However, the proportion of vocabulary used is generally quite small for both architectures: less than 16% for Flickr8k and less than 7% for Flickr30k. Overall, the trend is for smaller proportions of the overall training vocabulary to be used, as the vocabulary grows larger, suggesting that neural language models find it harder to use infrequent words (which are more numerous at larger vocabulary sizes, by definition). In practice, it means that reducing training vocabularies results in minimal performance loss.

Overall, the evidence suggests that delaying the merging of image features with linguistic encodings to a late stage in the architecture may be advantageous, at least as far as corpus-based evaluation measures are concerned. Furthermore, the results suggest that a merge architecture has a higher capacity than an inject architecture and can generate better quality captions with smaller layers.

5 Discussion

If the RNN had the primary role of generating captions, then it would need to have access to the image in order to know what to generate. This does not seem to be the case as including the image into the RNN is not generally beneficial to its performance as a caption generator.

When viewing RNNs as having the primary role of encoding rather than generating, it makes sense that the inject architecture generally suffers in performance when compared to the merge architecture. The most plausible explanation has to do with the handling of variation. Consider once more the task of the RNN in the image captioning task: During training, captions are broken down into prefixes of increasing length, with each prefix compressed to a fixed-size vector, as illustrated in Figure 2 above.

In the inject architecture, the encoding task is made more complex by the inclusion of image features. Indeed, in the version of inject used in our experiments – the most commonly used solution in the caption generation literature[5] – image features are concatenated with every word in the caption. The upshot is (a) a requirement to compress caption prefixes together with image data into a fixed-size vector and (b) a substantial growth in the vocabulary size the RNN has to handle, because each image+word is treated as a single 'word'. This problem is alleviated in merge, where the RNN encodes linguistic histories only, at the expense of more parameters in the softmax layer.

One practical consequence of these findings is that, while merge models can handle more variety with smaller layers, increasing the state size of the RNN in the merge architecture is potentially quite profitable, as the entire state will be used to remember a greater variety of previously generated words. By contrast, in the inject architecture, this increase in memory would be used to better accommodate information from two distinct, but combined, modalities.

[5] We are referring to architectures that inject image features in parallel with word embeddings in the RNN. In the literature, when this type of architecture is used, the image features might only be included with some of the words or are changed for different words (such as in attention models).

6 Conclusions

This paper has presented two views of the role of the RNN in an image caption generator. In the first, an RNN decides on which word is the most likely to be generated next, given what has been generated before. In multimodal generation, this view encourages architectures where the image is incorporated into the RNN along with the words that were generated in order to allow the RNN to make visually-informed predictions.

The second view is that the RNN's role is purely memory-based and is only there to encode the sequence of words that have been generated thus far. This representation informs caption prediction at a later layer of the network as a function of both the RNN encoding and perceptual features. This view encourages architectures where vision and langauge are brought together late, in a multimodal layer.

Caption generation turns out to perform worse, in general, when image features are injected into the RNN. Thus, the role of the RNN is better conceived in terms of the learning of linguistic representations, to be used to inform later layers in the neural network, where predictions are made based on what has been generated in the past together with the image that is guiding the generation. Had the RNN been the component primarily involved in generating the caption, it would need to be informed about the image in order to know what needs to be generated; however this line of reasoning seems to hurt performance when applied to an architecture. This suggests that it is not the case that the RNN is the main component of the caption generator that is involved in generation.

In short, given a neural network architecture that is expected to process input sequences from multiple modalities, arriving at a joint representation, it would be better to have a separate component to encode each input, bringing them together at a late stage, rather than to pass them all into the same RNN through separate input channels. With respect to the question of how language should be grounded in perceptual data, the tentative answer offered by these experiments is that the link between the symbolic and perceptual should be established late, once encoding has been performed. To this end, recurrent networks are best viewed as learning representations, not as generating sequences.

6.1 Future work

The experiments reported here were conducted on two separate datasets. One concern is that results on Flickr8k and Flickr30k are not entirely consistent, though the superiority of merge over inject is clear in both. We are currently extending our experiments to the larger MSCOCO dataset (Lin et al., 2014).

The insights discussed in this paper invite future research on how generally applicable the merge architecture is in different domains. We would like to investigate whether similar changes in architecture would work in sequence-to-sequence tasks such as machine translation, where instead of conditioning a language model on an image we are conditioning a target language model on sentences in a source language. A similar question arises in image processing. If a CNN were conditioned to be more sensitive to certain types of objects or saliency differences among regions of a complex image, should the conditioning vector be incorporated at the beginning, thereby conditioning the entire CNN, or would it be better to instead incorporate it in a final layer, where saliency differences would then be based on high-level visual features?

There are also more practical advantages to merge architectures, such as for transfer learning. Since merge keeps the image separate from the RNN, the RNN used for captioning can conceivably be transferred from a neural language model that has been trained on general text. This cannot be done with an inject architecture since the RNN would need to be trained to combine image and text in the input. In future work, we intend to see how the performance of a caption generator is affected when the weights of the RNN are initialized from those of a general neural language model, along lines explored in neural machine translation (Ramachandran et al., 2016).

Acknowledgments

This work was partially funded by the Endeavour Scholarship Scheme (Malta), part-financed by the European Social Fund (ESF).

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *Proc. ICCV'15*, pages 2425–2433, Santiago, Chile.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proc. Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, volume 29, pages 65–72.

Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank. 2016. Automatic Description Generation from Images: A Survey of Models, Datasets, and Evaluation Measures. *Journal of Artificial Intelligence Research*, 55:409–442.

Xinlei Chen and C. Lawrence Zitnick. 2015. Mind's eye: A recurrent visual representation for image caption generation. In *Proc. CVPR'15*.

Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term Recurrent Convolutional Networks for Visual Recognition and Description. In *Proc. CVPR'15*.

Desmond Elliott and Frank Keller. 2013. Image Description using Visual Dependency Representations. In *Proc. EMNLP'13*, pages 1292–1302, Seattle, WA. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proc. 13th International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Alex Graves. 2013. Generating Sequences with Recurrent Neural Networks. *CoRR*, abs/1308.0850:1–43.

Ankush Gupta, Yashaswi Verma, and C. V. Jawahar. 2012. Choosing Linguistics over Vision to Describe Images. In *Proc. AAAI'12*, pages 606–612.

Stevan Harnad. 1990. The symbol grounding problem. *Physica D*, 42:335–346.

Lisa Anne Hendricks, Subhashini Venugopalan, Marcus Rohrbach, Raymond Mooney, Kate Saenko, and Trevor Darrell. 2016. Deep Compositional Captioning: Describing Novel Object Categories without Paired Training Data. In *Proc. CVPR'16*.

Jack Hessel, Nicolas Savva, and Michael J. Wilber. 2015. Image Representations and New Domains in Neural Image Captioning. *CoRR*, abs/1508.02091.

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

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics. *Journal of Artificial Intelligence Research*, 47(1):853–899.

Ting-Hao Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, C Lawrence Zitnick, Devi Parikh, Lucy Vanderwende, Michel Galley, and Margaret Mitchell. 2016. Visual Storytelling. In *Proc. NAACL-HLT'16*, pages 1233–1239.

Andrej Karpathy and Li Fei-Fei. 2015. Deep Visual-Semantic Alignments for Generating Image Descriptions. In *Proc. CVPR'15*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. 2014a. Multimodal neural language models. In *Proc. ICML'14*, page 595603.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. 2014b. Unifying visual-semantic embeddings with multimodal neural language models. *CoRR*, abs/1411.2539.

Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C Berg, and Tamara L Berg. 2011. Baby Talk : Understanding and Generating Image Descriptions. In *Proc. CVPR'11*, pages 1601–1608, Colorado Springs.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436–444.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proc. ACL'04*. Association for Computational Linguistics (ACL).

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common objects in context. In *Proc. ECCV'14*, pages 740–755.

Siqi Liu, Zhenhai Zhu, Ning Ye, Sergio Guadarrama, and Kevin Murphy. 2016. Optimization of image description metrics using policy gradient methods. *CoRR*, abs/1612.00370.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2016. Knowing when to look: Adaptive attention via A visual sentinel for image captioning. *CoRR*, abs/1612.01887.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan L. Yuille. 2014. Explain images with multimodal recurrent neural networks. In *Proc. NIPS'14 Deep Learning Workshop*.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2015a. Deep Captioning with Multimodal Recurrent Neural Networks (m-RNN). In *Proc. ICLR'15*.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2015b. Learning like a Child: Fast Novel Visual Concept Learning from Sentence Descriptions of Images. In *Proc. ICCV'15*.

Rebecca Mason and Eugene Charniak. In *Proc. CONLL'14*, pages 11–20, Baltimore, MA.

Margaret Mitchell, Jesse Dodge, Amit Goyal, Kota Yamaguchi, Karl Stratos, Xufeng Han, Alyssa Mensch, Alex Berg, Xufeng Han, Tamara Berg, and Hal Daume III. 2012. Midge: Generating Image Descriptions From Computer Vision Detections. In *Proc. EACL'12*, pages 747–756, Avignon, France.

V Ordonez, G Kulkarni, and Tl Berg. 2011. Im2text: Describing images using 1 million captioned photographs. In *Proc. NIPS'11*, pages 1143–1151, Granada, Spain.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. ACL'02*, pages 311–318.

Prajit Ramachandran, Peter J. Liu, and Quoc V. Le. 2016. Unsupervised pretraining for sequence to sequence learning. *CoRR*, abs/1611.02683.

E. Reiter and R. Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge, UK.

Deb Roy and Ehud Reiter. 2005. Connecting language to the world. *Artificial Intelligence*, 167(1-2):1–12.

Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, abs/1409.1556.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded Compositional Semantics for Finding and Describing Images with Sentences. *Transactions of the Association for Computational Linguistics (TACL)*, 2(April):207–218.

Mingoo Song and Chang D. Yoo. 2016. Multimodal representation: Kneser-ney smoothing/skip-gram based neural language model. In *Proc. ICIP'16*.

Ilya Sutskever, James Martens, and Geoffrey Hinton. In *Proc. ICML'11*, pages 1017–1024, Bellevue, WA.

Marc Tanti, Albert Gatt, and Kenneth P. Camilleri. 2017. Where to put the image in an image caption generator. *CoRR*, abs/1703.09137.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. CIDEr: Consensus-based image description evaluation. In *Proc. CVPR'15*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proc. CVPR'15*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proc. ICML'15*.

Zhilin Yang, Ye Yuan, Yuexin Wu, Ruslan Salakhutdinov, and William W. Cohen. 2016. Encode, review, and decode: Reviewer module for caption generation. *CoRR*, abs/1605.07912.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *Proc. CVPR'16*.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

Luowei Zhou, Chenliang Xu, Parker Koch, and Jason J. Corso. 2016. Image caption generation with text-conditional semantic attention. *CoRR*, abs/1606.04621.