Where to put the Image in an Image Caption Generator[*]

Marc Tanti
marc.tanti.06@um.edu.mt
Institute of Linguistics
University of Malta

Albert Gatt
albert.gatt@um.edu.mt
Institute of Linguistics
University of Malta

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

Under review. Comments welcome.

Abstract

When a neural language model is used for caption generation, the image information can be fed to the neural network either by directly incorporating it in a recurrent neural network – conditioning the language model by injecting image features – or in a layer following the recurrent neural network – conditioning the language model by merging the image features. While merging implies that visual features are bound at the end of the caption generation process, injecting can bind the visual features at a variety stages. In this paper we empirically show that late binding is superior to early binding in terms of different evaluation metrics. This suggests that the different modalities (visual and linguistic) for caption generation should not be jointly encoded by the RNN; rather, the multimodal integration should be delayed to a subsequent stage. Furthermore, this suggests that recurrent neural networks should not be viewed as actually generating text, but only as encoding it for prediction in a subsequent layer.

[*]The research work disclosed in this publication is partially funded by the Endeavour Scholarship Scheme (Malta). Scholarships are part-financed by the European Union - European Social Fund (ESF) - Operational Programme II Cohesion Policy 2014-2020 Investing in human capital to create more opportunities and promote the well-being of society.

1 Introduction

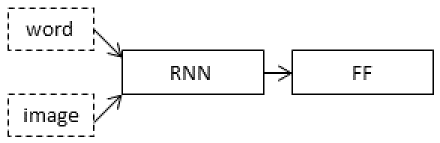
Image caption generation[1] is the task of generating a natural language description of the content of an image (Bernardi et al., 2016). One way to do this is to use a neural language model, a neural network that generates a sentence word by word. These work by using a recurrent neural network (RNN) that predicts the next word in the sentence based on its prefix or 'history'. This predicted word can then be appended to the previous prefix in order to predict the word after that, and so on, until a whole sentence is generated. A simple neural language model can be extended into an image caption generator by conditioning predictions on image features. In other words, the language model takes as input not only the prefix, but also the image being captioned. This raises the question: *At which stage should image information be introduced into the language model?*

Recent work on image captioning has answered this question in different ways, suggesting different views of the relationship between image and text in the caption generation task. To our knowledge, however, these different models and architectures have not been systematically compared. Yet, the question of where image information should feature in captioning is at the heart of a broader set of questions concerning how language can be grounded in perceptual information, questions which have been addressed by philosophers (Harnad, 1990) and AI practitioners (Roy, 2005).

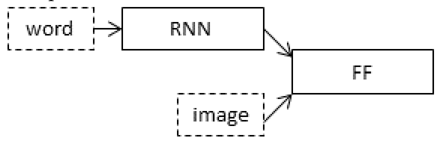
As we will show in more detail in Section 2, differences in the way caption generation architectures treat image features can be characterised in terms of two distinct sets of decisions:

Conditioning by injecting versus conditioning by merging : A language model can be conditioned by injecting the image (see Figure 1a). In these 'inject' architectures, the image vector (usually derived from the activation values of a hidden layer in a convolutional neural network) is injected into the RNN, for example by treating the image vector on a par with a 'word' and including it as part of the caption prefix. The RNN is trained to vectorise the image-caption mixture in such a way that this vector can be used to predict the next word. On the other hand, a language model can be conditioned by merging the image (see Figure 1b). In the case of 'merge' architectures, the image is left out of the RNN subnetwork, such that the RNN handles only the caption prefix, that is, handles only purely linguistic information. After the prefix has been vectorised, the image vector is then merged with the prefix vector in a separate 'multimodal layer' which comes after the RNN subnetwork. Merging can be done by, for example, adding the two vectors together elementwise. In this case, the RNN is trained to only vectorise the prefix and the mixture is handled in a subsequent feedforward layer.

[1]Throughout this paper we refer to textual descriptions of images as captions, although technically a caption is text that complements an image with extra information that is not available from the image. Specifically, the descriptions we talk about are 'concrete' and 'conceptual' image descriptions (Hodosh et al., 2013).



(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 output of the RNN after processing the words.

Figure 1: The inject and merge architectures for caption generation. Legend: RNN - Recurrent Neural Network; FF - Feed Forward layer.

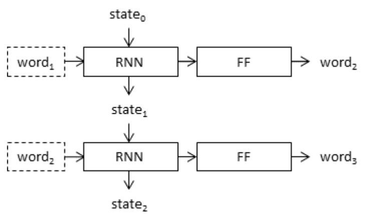
In short, if the image somehow influences the state of an RNN that is also processing words then it is being injected, otherwise it is being merged.

Early versus late inclusion of image features : As the foregoing description suggests, merge architectures tend to incorporate image features somewhat late in the generation process, that is, after processing the whole caption prefix. In the case of inject architectures, there is a broader range of possibilities, from injecting the image before processing any of the words in the caption prefix, to injecting the image after processing the last word of the prefix. Section 2 describes these options in more detail. The main point, however, is that it is possible to conceive of models in which visual information influences linguistic choices at different stages.

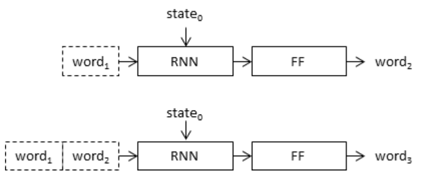
The main contribution of this paper is to present a systematic comparison of the different ways in which the 'conditioning' of linguistic choices based on visual information can be carried out, studying their implications for caption generator architectures. Thus, rather than seeking new results that improve on the state of the art, we seek to determine, based on an exhaustive evaluation of inject and merge architectures on a common dataset, where image features are best placed in the caption generation and image retrieval process.[2]

From a scientific perspective, such a comparison would be useful for shedding light on the way language can be grounded in vision. Should images and text be intermixed throughout the process, or should they initially be kept separate before being combined in some multimodal layer? Many papers speak of RNNs as 'generating' text. Is this the case or are RNNs better viewed as encoders which

[2]All the code used in our experiments is available at `https://github.com/mtanti/where-image`



(a) The continuous view of neural language models for generation. This is the common way RNNs are illustrated.



(b) The discontinuous view of neural languge models for generation. This is the way RNNs are illustrated in this paper.

Figure 2: Two views on neural language models. Legend: RNN - Recurrent Neural Network; FF - Feed Forward layer.

vectorise a prefix so that the next neural network layer can predict the next word? Answers to these questions would help inform theories of how caption generation can be performed.

From an engineering perspective, insights into the relative performance of different models could provide rules of thumb for selecting an architecture for image captioning, possibly for other tasks as well. This would make it easier to develop new architectures and new ways to perform caption generation.

The remainder of this paper is structured as follows. We first give an overview of related work, focusing in particular on the architectures used for caption generation. Section 3 discusses the architectures we compare, followed by a description of the data and experiments in Section 4. Results are presented and discussed in Section 5. We conclude with some general discussion and directions for future work.

2 Background

In this section we discuss a number of recent image caption generation models, with emphasis on how the image conditions the neural language model, based on the distinction between inject and merge architectures illustrated in Figure 1. Before we discuss these models, we first outline the view of RNNs as used for language modelling that will be taken in the remainder of this paper. This serves as useful background to understanding the purpose of the illustrations used throughout.

2.1 Ways of viewing recurrent neural networks

Traditionally, neural language models are depicted as in Figure 2a, where strings are thought of as being continuously generated. A new word is generated after each time step, with the RNN's state being combined with the last generated word in order to generate the next word. We refer to this as the 'continuous view'. In this paper, we adopt a slightly different – though functionally identical

– perspective on RNNs and their role in language models. This is illustrated in Figure 2b. We propose to think of the RNN in terms of a series of discontinuous snapshots over time, with each word being generated from the entire prefix of previous words and with the RNN's state being reinitialised each time. We refer to this as the 'discontinuous view'.

The discontinuous perspective brings to light some similarities between RNNs as used in language modelling and the encoding RNNs in sequence-to-sequence models, for example in neural machine translation systems (Sutskever et al., 2014). In order to translate a source sentence into a target sentence, the whole source sentence is first encoded into a fixed vector which is then mapped into a translation in the target language. In language modelling, instead of translating a sentence into another sentence, we're 'translating' the prefix of a sentence into a possible next word.

2.2 Types of architectures

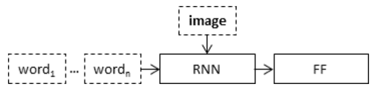
In Section 1, we made a high-level distinction between architectures that merge linguistic and image features in a multimodal layer, and those that inject image features directly into the caption prefix encoding process. As we noted, merge architectures tend to incorporate image features relatively late, after linguistic strings have been encoded. By contrast, the inject architecture has been instantiated in a number of different ways in the literature which bind image features at different times, illustrated in Figure 3 and described below:

- Init-inject: The RNN's initial state is set to be the image vector (or a vector derived from the image vector). This is an early binding architecture.

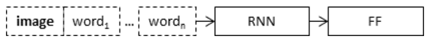
- Pre-inject: The first input to the RNN is the image vector (or a vector derived from the image vector). The word vectors of the caption prefix come later. The image vector is thus treated as a first word in the prefix. This is an early binding architecture.

- Par-inject: The image vector (or a vector derived from the image vector) serves as input to the RNN in parallel with the word vectors of the caption prefix, such that either (a) the RNN takes two separate inputs; or (b) the word vectors are combined with the image vector into a single input before being passed to the RNN. The image vector doesn't need to be exactly the same for each word, nor does it need to be included with every word. This is a mixed binding architecture.

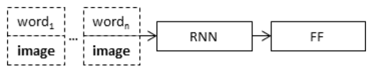
- Post-inject: This is a theoretical possibility, though no work actually adopts this architecture, to our knowledge. In this case, the last input to the RNN is the image vector (or a vector derived from the image vector) and is preceded by the word vectors of the caption prefix. The image vector is thus treated as the 'final word' in the prefix, a possibility that is probably easier to envisage under a discontinuous view of RNNs (Figure 2b). Post-inject architectures are mentioned here in view of our in-



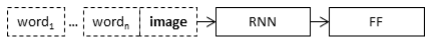
(a) Init-inject: The image vector is used as an initial state for the RNN.



(b) Pre-inject: The image vector is used as a first word in the prefix.



(c) Par-inject: The RNN accepts two inputs at once: the words and the image.



(d) Post-inject: The image vector is used as a last word in the prefix.

Figure 3: Different ways of conditioning a language model by inject.

clusion of them in the experiments reported below. This is a late binding architecture.

With these distinctions in mind, we next discuss a selection of recent contributions, placing them in the context of this classification. Table 1 provides a summary of a wider selection of published architectures.

2.2.1 Init-inject architectures

Architectures conforming to the init-inject model treat the image vector as the initial state of an RNN. These include (Devlin et al., 2015), who use a gated recurrent unit (GRU) model (Chung et al., 2014), that was initialised with an image vector. This was compared with a maximum entropy model (ME) that maps a bag of visual words extracted from an image onto the most likely complete caption.

In a related vein, (Ma and Han, 2016) first extract a sequence of attributes from an image, then map these to a caption using a long short-term memory (LSTM) network (Hochreiter and Schmidhuber, 1997), in an encoder-decoder framework á la (Sutskever et al., 2014).

(Wang et al., 2016) combine a simple RNN and an LSTM in one architecture, keeping them independent until their vectors are mixed together by weighted sum. This was motivated by the observation that the simple RNN and LSTM get better scores in different evaluation metrics.

(Liu et al., 2016) describe two systems, both of which are optimized by discrete optimisation on caption quality metrics directly which are CIDEr (Vedantam et al., 2015) and SPICE (Anderson et al., 2016). During learning, the system's error is measured by generating (sampling) a number of continuations of every prefix in the training set into a whole caption and measuring the average quality of the full generated captions. This quality is used to guide optimisation. One of the two systems described performs init-inject on an image vector.

Source	Init-inject	Pre-inject	Par-inject	Merge
(Chen and Zitnick, 2014)			✓	
(Chen and Zitnick, 2015)			✓	
(Devlin et al., 2015)	✓			
(Donahue et al., 2015)			✓	
(Hendricks et al., 2016)				✓
(Hessel et al., 2015)			✓	
(Karpathy and Fei-Fei, 2015)			✓	
(Kiros et al., 2014a)				✓
(Kiros et al., 2014b)				✓
(Krause et al., 2016)		✓		
(Liu et al., 2016)†	✓			
(Liu et al., 2016)†	✓		✓	
(Lu et al., 2016)			✓	✓
(Ma and Han, 2016)	✓			
(Mao et al., 2014)				✓
(Mao et al., 2015a)				✓
(Mao et al., 2015b)				✓
(Nina and Rodriguez, 2015)		✓		
(Oruganti et al., 2016)			✓	
(Rennie et al., 2016)†			✓	
(Rennie et al., 2016)†		✓		
(Song and Yoo, 2016)				✓
(Vinyals et al., 2015)		✓		
(Wang et al., 2016)	✓			
(Wu et al., 2015)		✓		
(Xu et al., 2015)	✓		✓	✓
(Yang et al., 2016)	✓		✓	
(Yao et al., 2016)†		✓		
(Yao et al., 2016)†		✓	✓	
(You et al., 2016)		✓	✓	✓
(Zhou et al., 2016)			✓	

Table 1: Summary of caption generators that use the different conditioning methods. Post-inject has never been used to our knowledge. † means that the publication describes multiple systems which use different conditioning methods.

2.2.2 Pre-inject architectures

Pre-inject models treat the image as though it were the first word in the prefix during training. (Vinyals et al., 2015) use an LSTM language model in this way, finding that pre-injection gives better results than par-injection (described below). Their hypothesis is that this is because par-injection makes the neural network overfit to image noise, which is always present with every word in the prefix.

Rather than injecting a low-level image vector, (Wu et al., 2015) opt for a higher-level approach by extracting attributes from an image and passing those as an input to the caption generator. This attributes vector is passed to the RNN by pre-injection.

(Krause et al., 2016) generate paragraph-length captions in two stages. First, an LSTM which incorporates the image at every time step generates a sequence of vectors representing sentence topics; these constitute the input to another LSTM which generates the caption, by injecting the sentence vector with the caption prefix to predict the next word in the caption.

(Rennie et al., 2016) describe two systems, an attention model and a non-attention model, both of which use discrete optimisation in order to optimize CIDEr directly. It was found that optimizing CIDEr directly is the best way to improve other metrics as well. During learning, the system's error is measured by generating (sampling) full captions and measuring the quality of the captions. This quality is used to guide optimisation. The non-attention model pre-injects the image vector.

(Yao et al., 2016) describe five systems which mix image vectors and image attributes in different ways. Three of the systems work as follows: (a) pre-injecting the attributes alone (b) injecting the image as a first word and the attributes as a second, (c) injecting the attributes as a first word and the image as a second. This last one gives the best performance of all five in terms of METEOR (Banerjee and Lavie, 2005).

2.2.3 Par-inject architectures

Par-injection combines image features and word features together when passing them to the RNN. For example, in two papers, Chen and colleagues (Chen and Zitnick, 2014, 2015) approach caption generation using two simple RNNs in parallel. One RNN is trained to predict the image vector given the caption prefix while the other RNN is trained to predict the next word given the caption prefix, the image vector, and the first RNN. The first RNN 'remembers' visual information about the caption prefix; this provides a more stable memory than a simple RNN's state.

Similarly, (Donahue et al., 2015) describes an image and video captioning system that puts two LSTMs in series after each other. However, it was found that injecting the image into the second (later) LSTM works better than injecting it into the first.

(Karpathy and Fei-Fei, 2015) generate captions for multiple subregions within an image. The caption generation works by using a simple RNN that uses par-injection, but only with the first word, that is, only the first word is combined with the image. Everywhere else the image vector is treated as if it is an all-zeros vector. It was found that this works better than par-injection with every word.

(Zhou et al., 2016) also use par-injection, but the first word that the image is combined with is an all-zeros vector, that is, they use both pre-injection and par-injection. The image vector's elements are attended to differently for each word in the prefix, which is then par-injected into the RNN.

One of the two systems of (Rennie et al., 2016) has already been described in Section 2.2.2. The attention model performs par-inject of the attended image vector into the cell memory of an LSTM. Attention was found to improve the quality of the system.

2.2.4 Merge architectures

Rather than combining image features together with linguistic features from within the RNN, merge architectures delay their combination until after the caption prefix has been vectorised. Among the exponents of this approach is the work of (Kiros et al., 2014a), who uses a log-bilinear language model (LBL) (Mnih and Hinton, 2007) to convert a caption prefix into a single vector which is later merged with the image vector in order to determine the next word in the caption.

In later work, (Kiros et al., 2014b) describe a caption generator that can be conditioned either on images or on full captions. This is because both image and caption vectors are embedded into a common multimodal space which maximises the similarity between corresponding images and captions. An image or caption vector is then merged with the output of a LBL in order to predict the next word in a prefix. To assist with the generation process, the expected parts of speech are passed to the LBL as well.

Similar to Kiros, (Mao et al., 2014, 2015a) use a multimodal layer, where an image vector is merged with a vector from a simple RNN. (Mao et al., 2015a) do a basic comparison between inject and merge architectures and find that merging works better than injecting. (Mao et al., 2015b) describe a subsequent development, where the vocabulary of a caption generator can be increased after training by only learning the new words' embeddings whilst leaving the rest of the network intact.

(Hendricks et al., 2016) also uses a multimodal layer that merges the image vector with the caption prefix vector produced using an LSTM. This architecture keeps the image out of the LSTM by necessity, as the system is designed to be trained on images and sentences separately, to enable generalisation to objects not found in the training set. Hence, an object recogniser and a language model are trained separately. Their outputs are combined into a single vector which is used to predict the next word in the caption prefix. This makes it possible to improve the performance of each module separately using non-captioned images and free text.

2.2.5 Mixed models

There is a growing amount of work that has explored a combination of different inject and/or merge strategies, most of which are attention-based models. This is the case in the work of (Xu et al., 2015), who describe an attention-based caption generator. Here, an LSTM is initialised using the full image. The attended image is passed to the LSTM by par-injection, giving rise to a new state. This new state is then merged with the attended image again in order to determine the next word. Thus, the architecture incorporates aspects of the init-inject, par-inject and merge strategies.

(Lu et al., 2016) describe an attention-based model that also determines the importance that should be given to an image during generation. If a caption needs to generate the compound word "stop sign", then only "stop" requires access to the image and "sign" can be deduced linguistically. The attended image is merged whilst the full image is par-injected.

(Yang et al., 2016) describe a generic attention-based encoder-decoder system that was used for image captioning and source code commenting. The attended image is par-injected whilst a mixture of the attended and full image is init-injected.

(You et al., 2016) describe an attention-based caption generator that init-injects the full image whilst using attributes from the image to perform attention both at the word level (par-inject) and at the RNN level (merge).

One of the two systems of (Liu et al., 2016) has already been described in Section 2.2.1. The other system performs init-inject of the image vector and par-inject of the image attributes. The addition of attributes did not significantly improve the system.

Three of the five systems of (Yao et al., 2016) have already been described in Section 2.2.2. The other two systems perform pre-inject of the image attributes and par-inject of the image vector or pre-inject of the image vector and par-inject of the image attributes. This last system performs the best of all five in terms of CIDEr.

2.3 Summary and outlook

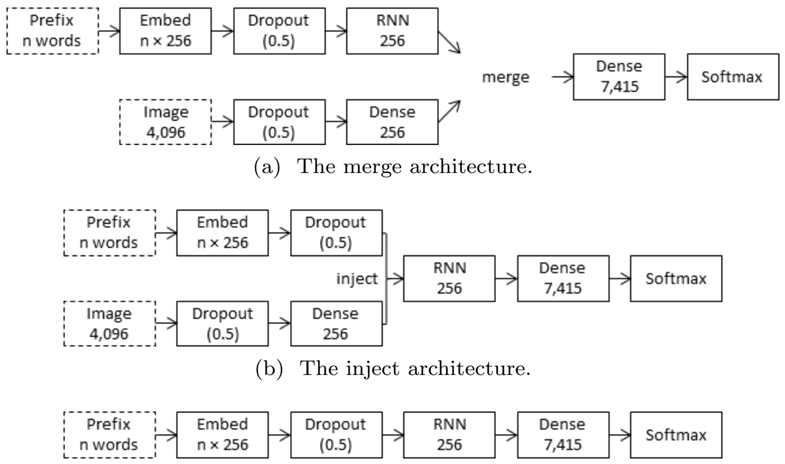
While the literature on caption generation now provides a rich range of models and comparative evaluations, there is as yet very little explicit systematic comparison between the performance of the architectures surveyed above, each of which represents a different way of conditioning the prediction of language sequences on visual information. Work that has tested both par-inject and pre-inject, such as (Vinyals et al., 2015), reports that pre-inject works better. The work of (Mao et al., 2015a) compares inject and merge architectures and concludes that merge is better than inject. However Mao *et al.*'s comparison between architectures is a relatively tangential part of their overall evaluation, and is based only on the BLEU metric (Papineni et al., 2002).

Answering the question of which architecture is best is difficult because different architectures perform differently on different evaluation measures, as shown for example by (Wang et al., 2016), who compared simple RNNs and LSTMs. Although the state of the art systems in caption generation all use inject type architectures, it is also the case that they are more complex systems than the published merge architectures and so it is not fair to conclude that inject is better than merge on a survey of the literature alone.

In what follows, we present a systematic comparison between all the different architectures discussed above, using both simple RNNs and LSTMs. We perform these evaluations using a common dataset and a variety of quality metrics, covering (a) the predictive capacity of the caption generator; (b) the quality of the generated captions; (c) the linguistic diversity of the generated captions; and (d) the networks' capabilities to find the most relevant image given a caption. This last metric is included by way of comparison: many caption generators have been shown to serve a dual function, for both image description and retrieval. By including retrieval among our evaluation methods, we seek to shed light on whether, from an architectural perspective, the same conclusions can be drawn as for captioning.

3 Architectures

In this section we go over the different architectures that are evaluated in this paper. A diagram illustrating the three main architecture types is shown in



(a) The merge architecture.

(b) The inject architecture.

(c) The imageless language model architecture.

Figure 4: An illustration of the different architectures that are tested in this paper. The numbers refer to the vector size output by a layer, except for dropout layers which refer to the dropout rate. Note that the softmax size of 7,415 refers to the number of possible words that can be chosen as next words (determined by the vocabulary size). 'Dense' means fully connected layer.

Figure 4. We start with a description of a general, schematic architecture in Section 3.1, followed by a description of how each evaluated architecture modifies this schema in Section 3.2. Finally, we give a formal description of the important details of the architectures in Section 3.3.

3.1 General architecture

This section describes the basis from which all evaluated architectures are derived. It is a scaled-down version of the architecture described in (Vinyals et al., 2015). This architecture was chosen for its simplicity whilst still being the best performing system in the 2015 MSCOCO image captioning challenge.[3]

Word embeddings Word embeddings, that is, the vectors that represent known words prior to being fed to the RNN, consist of 256-element vectors that have been randomly initialised. No precompiled vector embeddings such as word2vec (Mikolov et al., 2013) were used. Instead, the embeddings are trained as part of the neural network in order to learn the best representations of words.

Recurrent neural network The purpose of the RNN is to take a prefix of embedded words (with image vector in inject architectures) and produce a single vector that represents the sequence. The size of this vector is 256. Two types

[3]`http://mscoco.org/dataset/#captions-leaderboard`

of RNNs were used: the simple RNN described in (Mao et al., 2015a) and the LSTM described in (Vinyals et al., 2015). Apart from enabling a comparison of different RNN types overall, this strategy accounts for using an RNN that is reported to work well in a merge architecture (Mao et al., 2015a) and another RNN that is reported to work well in an inject architecture (Vinyals et al., 2015).

Image Prior to training, all images were vectorised using the activation values of the penultimate layer of the VGG OxfordNet 19-layer convolutional neural network (Simonyan and Zisserman, 2014), which is trained to perform object recognition and returns a 4,096-element vector. The convolutional neural network is not influenced by the caption generation training. During training, a layer of the neural network (with no activation function) compresses this vector into a 256 element vector (the same size as word vectors). It is this 256-element vector that is referred to as the 'image vector' in our experiments.

Output Once the image and the caption prefix have been vectorised and mixed into a single vector, the next step is to use them to predict the next word in the caption. This is done by passing the mixed vector through a softmax layer that predicts the probability of each possible 7,415 next word being the word that comes after the prefix.

Regularisation In order to reduce overfitting, dropout (Srivastava et al., 2014) with a dropout rate of 0.5 was applied on the 4,096-element image input and the word embeddings. Preliminary experiments showed that applying dropout to the input vectors works better than applying dropout to intermediate layers.

3.2 Evaluated architectures

This section describes each architecture that is evaluated in our experiments. There is one language model architecture (***langmodel***), three merge architectures (***merge***), and four inject architectures (***inject***), each of which can either use a simple RNN (***srnn***) or an LSTM (***lstm***) as a recurrent neural network. In total we therefore evaluate (8×2) 16 different architectures.

3.2.1 The language model architecture

The language model architecture (***langmodel***) is a simple unconditioned imageless language model that predicts the next word in a caption given a prefix only. It is used as a baseline against which to measure the effectiveness of different ways of including an image in an architecture.

3.2.2 The merge architectures

The merge architectures merge the prefix vector with the image vector before passing the result to the output layer. We consider three ways to merge the image vector with the prefix vector:

- ***merge-concat***: The image vector and prefix vector are concatenated into a single vector that has a length equal to the sum of the length of the two vectors. Whilst this method is intuitive, we are mindful of the extra parameters this architecture needs to have in order to process a larger layer.
- ***merge-add***: The image vector and prefix vector are added together elementwise into a single vector that has a length equal to the original vectors (image vector and prefix vector have an equal length).
- ***merge-mult***: The image vector and prefix vector are multiplied together elementwise into a single vector that has a length equal to the original vectors (image vector and prefix vector have an equal length).

3.2.3 The inject architectures

The inject architecture injects the image vector into the RNN as if it were part of the caption prefix. As noted in Section 2, there are four ways to inject the image into the RNN, which are realised in our experiments in the following way:

- ***inject-init***: The image vector is treated as an initial state for the RNN. After initialising the RNN, the vectors in the prefix are then fed to the RNN as usual. Every other architecture in our experiments uses the all-zeros vector as an initial state.
- ***inject-pre***: The image vector is used as the first 'word' in the caption prefix. This makes the image vector the first thing that the RNN will see.
- ***inject-par***: The image vector is added (elementwise) to every word vector in the caption prefix in order to make the RNN take a mixed word-image vector. Every word would have the exact same image vector added to it.
- ***inject-post***: The image vector is used as the last 'word' in the caption prefix. This makes the image vector the last thing that the RNN will see before predicting each next word.

3.3 Formal details

This section gives a more formal description of the evaluated architectures. As a matter of notation, we treat vectors as being horizontal.

The simple RNN, which is the same one used in (Mao et al., 2015a), is defined as

$$s_n = \mathrm{ReLU}(x_n + s_{n-1}W_{ss} + b_s) \tag{1}$$

where x_n is the n^{th} input, s_n is the hidden state after n inputs, s_0 is the initial state, W_{ss} is the state-to-state weight matrix (which is square), b_s is the state bias vector, and ReLU refers to the Rectified Linear Unit function, which is defined as

$$\text{ReLU}(x) = \max(0, x) \tag{2}$$

The LSTM model, which is the one used in (Vinyals et al., 2015), is defined as

$$i_n = \text{sig}(x_n W_{xi} + s_{n-1} W_{si} + b_i) \tag{3}$$
$$f_n = \text{sig}(x_n W_{xf} + s_{n-1} W_{sf} + b_f) \tag{4}$$
$$c_n = f_n \odot c_{n-1} + i_n \odot \tanh(x_n W_{xc} + s_{n-1} W_{sc} + b_c) \tag{5}$$
$$o_n = \text{sig}(x_n W_{xo} + s_{n-1} W_{so} + b_o) \tag{6}$$
$$s_n = o_n \odot c_n \tag{7}$$

where x_n is the n^{th} input, s_n is the hidden state after n inputs, s_0 is the initial state, c_n is the cell state after n inputs, c_0 is the all-zeros vector (even when init-inject is used), 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, $W_{\alpha\beta}$ is the weight matrix between α and β, b_α is the bias vector for α, and $\odot$ is the elementwise vector multiplication operator. In the above, 'sig' refers to the sigmoid function which is defined as:

$$\text{sig}(x) = \frac{1}{1 + e^{-x}} \tag{8}$$

Only the last hidden state of the RNN is used in the rest of the neural network. Both the state vector (in the simple RNN and LSTM) and the cell vector (in the LSTM) are of size 256.

The feedforward layers used for the image and output are normal dense layers which are defined as

$$z = xW + b \tag{9}$$

where z is the result vector, x is the input vector, W is the weight matrix, b is the bias vector.

The result vector can then be passed through an activation function, such as softmax, which is defined as

$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{10}$$

where $\text{softmax}(x)_i$ refers to the i^{th} element of the output vector.

Framework	Theano 0.9.0 (Theano Development Team, 2016)
Language	Python 2.7.6
Data set	Flickr30k (Young et al., 2014)
Vocabulary	all tokens in training set occurring 5 times or more
Cost function	sum cross-entropy
Optimisation	Adam (P. Kingma and Ba, 2014)
Minibatch size	500
Regularisation	dropout with dropout rate 0.5
Stopping criteria	early stopping patience of 2 epochs with maximum of 20 epochs
Gradient control	elementwise gradient clipping of ± 5
Initialisation	biases - zeros; feedforward weights - xavier (Glorot and Bengio, 2010); recurrent weights - orthogonal
Generation	beam search with beam width being 40 and a maximum length of 50 words

Table 2: Summary of parameters and configurations used in all experiments.

4 Experiments

This section describes the experiments conducted in order to compare the performance of the different architectures described in the previous section. A summary of all the parameters and configurations is given in Table 2.

4.1 Dataset

Images The dataset used for all experiments was the version of Flickr30k[4] (Young et al., 2014) distributed by (Karpathy and Fei-Fei, 2015)[5], a set of 31,014 images taken from Flickr combined with five manually written captions per image. The provided dataset is split into a training, validation, and test set of 29,000, 1,014, and 1,000 images respectively. The images are already vectorised into 4,096-element vectors via the activations of layer 'fc7' (the penultimate layer) of the VGG OxfordNet 19-layer convolutional neural network (Simonyan and Zisserman, 2014), which was trained for object recognition on the ImageNet dataset (Deng et al., 2009). These vectors were normalised to unit-length vectors prior to training.

Vocabulary The known vocabulary consists of all the words in the captions training set that occur at least 5 times (capital letters are normalised to lowercase before constructing the vocabulary). This gives us a total of 7,413 known words. These words are used both as inputs, which are embedded and fed to

[4]We used Flickr30k instead of the larger MSCOCO (Lin et al., 2014) in order to reduce training time as we had to evaluate many different architectures.

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

the RNN, and as outputs, which are given probabilities by the softmax. Apart from these content tokens, some other special tokens were used:

- PAD: ignored by the RNN and used to make all caption prefixes in the training set of equal length (used only with the input words);
- START: used to mark the beginning of the caption prefix and useful for predicting the first word in the caption (used only with the input words);
- END: used to mark the end of the caption and useful for predicting the end of the caption (used only with the output words);
- UNKNOWN: used to replace and represent any word which is not in the known vocabulary (used with both the input and output words).

4.2 Training set

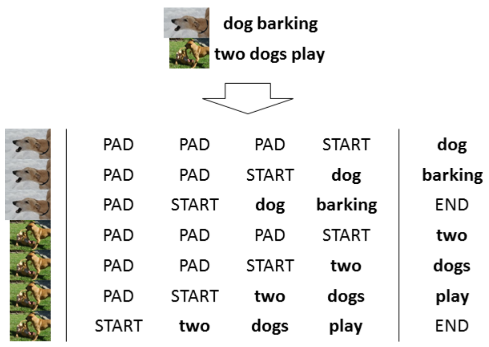
In Section 2, we noted that it is possible to adopt a continuous or a discontinuous perspective on an RNN. For the experiments reported here, adopting the latter perspective turns out to be beneficial, in that it naturally enables a unified treatment of all the models under consideration. In particular, training a post-inject model requires us to inject the image at every stage of training on a given prefix.

Therefore, throughout our experiments, captions are broken down into separate prefixes of increasing length and each prefix is treated as a separate entry in the training set. The training goal is to maximize the individual probability of each word in all the captions (including the END token), given the image and caption prefix.

The training set produced from the dataset consists of triples $\langle I, C_{0 \ldots n}, C_{n+1} \rangle$, where I is an image described by caption C, $C_{0 \ldots n}$ is the caption prefix with n words (plus START token), and C_{n+1} is the next word (which can be the END token). Each caption-image pair is broken down into all the caption's prefixes together with the word following the prefix and corresponding image. All prefixes start with a START token (including the empty prefix with the first word as a next word) and are padded so that all of the sequences are of the same length (equal to the longest caption in the training set plus the START token). An example is given in Figure 5. Before each training epoch, the rows are all shuffled and then divided into minibatches. We allow prefixes of the same caption to be placed in different minibatches.

4.3 Learning

We did not focus too much on optimizing hyperparameters as it is difficult to find a set of hyperparameters that benefits all architectures at once and we did not want one architecture to have an advantage over the others. For this reason we used default or generally recommended settings were possible.



	PAD	PAD	PAD	START	**dog**
	PAD	PAD	START	**dog**	**barking**
	PAD	START	**dog**	**barking**	END
	PAD	PAD	PAD	START	**two**
	PAD	PAD	START	**two**	**dogs**
	PAD	START	**two**	**dogs**	**play**
	START	**two**	**dogs**	**play**	END

Figure 5: An illustration of how the Flickr30k dataset is processed into a training set for training. First column in the training set is for the image, second is for the prefix, and third is for the next word after the prefix.

Cost function As a cost function, the sum of the crossentropy of all $\langle I, C_{0\ldots n}, C_{n+1}\rangle$ triples (symbols defined in Section 4.2) in the training set minibatch were used. This means that the neural networks are trained to minimise the crossentropy of their probability predictions on the training set captions. The crossentropy is defined as follows:

$$\text{crossentropy}(P, I, C_{0\ldots n}, C_{n+1}) = -\ln\left(P(C_{n+1}|C_{0\ldots n}, I)\right) \tag{11}$$

where P is the output of the neural network being trained which is the probability of a particular word being the next word in a caption prefix.

Optimisation Neural network learning is performed using Adam (P. Kingma and Ba, 2014) as an optimisation method. All hyperparameters were left as suggested in the original paper: $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$.

Gradient control In order to avoid gradient explosion (Pascanu et al., 2012), all gradients in the gradient descent algorithm were clipped to be within the range ± 5 as suggested by (Karpathy and Fei-Fei, 2015).

Minibatches Minibatches of size 500 were used. Minibatches consist of prefixes rather than sentences (see Figure 5).

Stopping criteria Each training session lasted at most 20 epochs. The same cost function that is used on the training set is also used on the validation set so that only the weights that gave the smallest validation cost will be evaluated on the test set. An early stopping patience of two was used such that if the cost stops decreasing for two epochs on the validation set then training is stopped.

Initialisation For weight initialisation, all biases were set to zero and all feed forward weights were randomly set using xavier initialisation (Glorot and Bengio, 2010), including the word embeddings. Xavier initialisation sets the weights of a layer to random numbers sampled from a normal distribution with mean equal to zero and standard deviation equal to $\sqrt{\frac{2}{n_i+n_o}}$ where n_i is the number of inputs to the layer and n_o is the number of outputs from the layer. The recurrent weights of the RNNs (square matrices that process the state) were initialised using orthogonal weights.

4.4 Evaluation metrics

To evaluate the different architectures, the test set captions (which are shared among all experiments) are used to measure the architectures' quality using metrics that fall into four classes, described below.

4.4.1 Probability metrics

These are metrics that quantify how well the architectures are at predicting probabilities of the caption words given their corresponding image and prefix. We report the perplexity of the model given the test captions. The predicted perplexity of a sentence is calculated as:

$$\text{perplexity}(P, C, I) = 2^{H(P,C,I)} \tag{12}$$

$$H(P, C, I) = -\frac{1}{|C|}\sum_{n=0}^{|C|}\log_2 P(C_{n+1}|C_{0\ldots n}, I) \tag{13}$$

where P is the trained neural network that gives the probability of a particular word being the next word in a caption prefix, C is a caption with $|C|$ words, I is an image described by caption C, and H is the entropy function. Note that C_n is the n^{th} word in C and $C_{0\ldots n}$ are the first n words in C (with START token).

In order to aggregate the caption perplexity of the entire test set of captions into a single number, we report the arithmetic mean, geometric mean, and median of all the caption's scores.

4.4.2 Generation metrics

These metrics quantify the quality of the generated captions by measuring the degree of overlap between generated captions and those in the test set. We use the MSCOCO evaluation code[6] which measures 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). Captions were generated using beam search with a beam width of 40 and a clipped maximum length of 50 words.

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

4.4.3 Diversity metrics

Apart from measuring the caption similarity to the ground truth we also measure the diversity of the vocabulary used in the generated captions. This is intended to shed light on the extent to which captions are 'stereotyped', that is, the extent to which a model re-uses (sub-)strings from case to case, irrespective of the input image.

As a limiting case, consider a caption generator which always outputs the same caption. This has the lowest possible diversity. In order to quantify this we measure the percentage of known vocabulary words used in all generated captions and the entropy of the unigram and bigram frequencies in all the generated captions together, which is calculated as:

$$\text{diversity}(P, F) = -\sum_{i=1}^{|F|} P_i(F) \log_2 P_i(F) \tag{14}$$

$$P_i(F) = \frac{F_i}{\sum_{j=1}^{|F|} F_j} \tag{15}$$

where F is the frequency distribution over unigrams or bigrams, with $|F|$ different unigrams or bigrams, and P_i is the maximum likelihood estimate probability of encountering unigram or bigram i. Note that F_n is the frequency of the n^{th} unigram or bigram.

Entropy gives a measure of how uniform the frequency distributions are (with higher entropy for more uniform distributions). The more uniform, the more likely that each unigram or bigram was used in equal proportion, rather than using the same few words for the majority of the time, hence the greater the variety of words used.

4.4.4 Retrieval metrics

As noted in Section 2, we include image retrieval performance among our evaluation measures in view of the frequent practice in the literature of evaluating image captioning models bidirectionally, for both generation and retrieval. In the present case, we are mainly interested in whether the various architectures under consideration are ranked the same in the two tasks.

Retrieval metrics are metrics that quantify how well the architectures are at retrieving the correct image given a caption. A conditioned language model can be used for retrieval by measuring the degree of relevance each image has to the given caption. Relevance is measured as the probability of the whole caption given the image (by multiplying together each word's probability). Different images will give different probabilities for the same caption. The more probable the caption is, the more relevant the image is.

We use the standard $R@n$ recall measures (Hodosh et al., 2013), and report recall at 1, 5, and 10. Recall at n is the percentage of captions whose correct image is ranked at position n or less in the list of images sorted by relevance.

5 Results

Three runs of each experiment were performed and the mean together with the standard deviation (reported in parentheses) of the different evaluation measures over the three runs is reported. For each run, the initial weight settings, minibatch selections, and dropout selections are different since these are randomly determined. Everything else is identical across runs.

For comparison, the architectures used in (Mao et al., 2015a) and (Vinyals et al., 2015) are also included in the tables below. Both of these approaches have been influential in the literature; furthermore, they are explicitly defined and hence can be easily reimplemented. For the purposes of the present experiments, they were reimplemented using the same layers and layer sizes as in the original; however, to ensure a fair comparison we have trained and tested them in the same way as the rest of the architectures considered here. This means that we have used the same initialisation, regularisation, training method, *etc.* For this reason, the results reported below for the Mao and Vinyals architectures occasionally differ from those originally reported by the authors.

5.1 Data

Table 3 shows the results of the evaluation using probability metrics, which consist of different ways to aggregate perplexities, against the captions in the test set.

Table 4 shows metrics that measure the quality of generated captions. Note that the imageless language model here generates one caption, which is the most probable caption overall, and uses it for all images.

Table 5 shows the extent to which models exploit a significant proportion of the available vocabulary. We estimate the proportion of the available vocabulary (unigram or bigram) used by a model. For comparison, we include the proportions for human captions in the test set, considering both an overall proportion based on all five captions per test set image (***human-all***) and a proportion based on only the first caption from the five available human captions (***human-one***).

Finally, Table 6 shows the results of evaluating the models based on a 'reversal' of the caption generation process, considering image retrieval based on a caption in the test set. Note that, since the imageless language model cannot be applied for retrieval, it is not included in this table.

5.2 Discussion

If we take the late binding architectures, merge and post-inject, and the early binding architectures, init-inject and pre-inject, as two groups, then there is a clearly discernible pattern for both the models using a simple RNN and those using an LSTM: given the same RNN type, late binding architectures perform better than early binding architectures with mixed binding architectures (par-inject) floating somewhere in the middle. This holds across the board, for all

	pplx geomean	pplx artmean	pplx median
vinyals	**18.405 (0.061)**	**36.507 (0.169)**	**16.029 (0.155)**
merge-concat-lstm	18.740 (0.038)	41.934 (0.324)	16.236 (0.149)
inject-post-lstm	18.868 (0.052)	45.652 (3.077)	16.143 (0.041)
merge-add-lstm	18.943 (0.057)	38.647 (0.479)	16.500 (0.151)
inject-init-lstm	19.108 (0.100)	43.358 (1.419)	16.435 (0.135)
inject-par-lstm	19.142 (0.043)	41.422 (1.080)	16.629 (0.058)
inject-pre-lstm	19.228 (0.053)	45.984 (3.867)	16.586 (0.138)
mao	19.404 (0.045)	40.023 (1.452)	16.961 (0.061)
merge-add-srnn	20.812 (0.027)	43.040 (1.030)	18.054 (0.154)
merge-concat-srnn	20.971 (0.077)	49.812 (1.812)	17.981 (0.073)
inject-post-srnn	21.810 (0.142)	102.489 (45.708)	18.402 (0.384)
inject-par-srnn	21.991 (0.148)	60.842 (6.368)	18.712 (0.165)
merge-mult-srnn	22.268 (0.170)	53.501 (1.498)	19.053 (0.158)
merge-mult-lstm	23.655 (0.861)	96.770 (6.263)	19.939 (0.810)
inject-init-srnn	24.220 (0.378)	83.403 (16.085)	20.456 (0.328)
inject-pre-srnn	24.279 (0.335)	88.050 (23.114)	20.695 (0.466)
langmodel-lstm	24.740 (0.122)	55.003 (1.016)	20.914 (0.081)
langmodel-srnn	28.822 (0.079)	84.552 (1.917)	24.292 (0.080)

Table 3: Results of the caption probability metrics. Legend: geomean - geometric mean; artmean - arithmetic mean; pplx - perplexity. Lower is better.

evaluation metrics, including retrieval.

This result is somewhat surprising, since the LSTM is a reimplementation of the one described by (Vinyals et al., 2015), which is optimized for a pre-inject architecture. In fact there doesn't seem to be any evaluation criterion where early binding architectures have an advantage over late binding ones. Even if we ignore merge-concat, which has more weights than other architectures, merge-add is always better than inject-par which is almost always better than inject-pre.

Merge by multiplication seems to suffer on perplexity, but performs well on the other measures. It also has the highest standard deviation in median perplexity (across the three runs). This suggests that one possible reason for its relatively poor ranking on perplexity is due to a high degree of variation in output probability across runs.

The diversity metrics show that all architectures have a similar word frequency distribution (although simple RNN early binding architectures are relatively more skewed). Nevertheless, the statistics on vocabulary usage are very telling. It seems that even though humans use at least 32% of the known vocabulary to describe the test set images, none of the evaluated systems used more than 7%. We interpret this as meaning that neural caption generators require seeing a word in the training set very often in order to learn how to use it. From a methodological perspective, this further implies that setting an even higher frequency threshold below which words are mapped to the UNKNOWN word (the current experiments set the threshold at five), would be feasible and would make relatively little difference to the results. Furthermore, this also explains why the standard deviation of the generation measures is so low compared to other measures: the caption generators were conservatively using a handful of words, resulting in a relatively low degree of variation in the captions.

In the concluding section, we turn to a more thorough interpretation of these results. Our conclusion, however, must be that models in which image features are included *early* in the generation process perform poorly, relative to those models based on injecting or merging image features *later*. It appears,

	CIDEr	METEOR	ROUGE-L
merge-add-srnn	**0.337 (0.009)**	0.157 (0.002)	0.397 (0.003)
merge-mult-lstm	0.337 (0.004)	**0.158 (0.002)**	**0.399 (0.004)**
inject-post-srnn	0.333 (0.008)	0.156 (0.002)	0.392 (0.003)
merge-add-lstm	0.331 (0.011)	0.156 (0.002)	0.394 (0.002)
mao	0.325 (0.012)	0.156 (0.003)	0.395 (0.006)
merge-concat-lstm	0.320 (0.013)	0.155 (0.001)	0.393 (0.003)
inject-post-lstm	0.320 (0.007)	0.152 (0.001)	0.386 (0.003)
merge-mult-srnn	0.319 (0.009)	0.155 (0.001)	0.393 (0.004)
inject-par-lstm	0.318 (0.006)	0.152 (0.001)	0.388 (0.003)
merge-concat-srnn	0.316 (0.006)	0.152 (0.001)	0.388 (0.001)
inject-par-srnn	0.297 (0.007)	0.148 (0.001)	0.381 (0.004)
inject-pre-lstm	0.291 (0.009)	0.150 (0.004)	0.383 (0.008)
vinyals	0.290 (0.005)	0.148 (0.002)	0.379 (0.002)
inject-init-lstm	0.281 (0.003)	0.146 (0.000)	0.379 (0.003)
inject-init-srnn	0.256 (0.007)	0.147 (0.001)	0.381 (0.003)
inject-pre-srnn	0.238 (0.005)	0.144 (0.003)	0.371 (0.008)
langmodel-srnn	0.085 (0.001)	0.097 (0.011)	0.260 (0.027)
langmodel-lstm	0.070 (0.010)	0.090 (0.001)	0.260 (0.028)

(a) CIDEr, METEOR, and ROUGE-L results.

	BLEU-1	BLEU-2	BLEU-3	BLEU-4
merge-add-srnn	**0.578 (0.008)**	0.385 (0.005)	0.254 (0.005)	0.164 (0.005)
merge-mult-lstm	0.575 (0.008)	**0.390 (0.007)**	**0.260 (0.005)**	**0.170 (0.005)**
merge-add-lstm	0.569 (0.005)	0.378 (0.003)	0.249 (0.004)	0.162 (0.005)
merge-concat-lstm	0.569 (0.004)	0.374 (0.001)	0.243 (0.002)	0.156 (0.003)
merge-mult-srnn	0.562 (0.004)	0.379 (0.005)	0.252 (0.006)	0.166 (0.004)
mao	0.561 (0.009)	0.377 (0.009)	0.249 (0.009)	0.162 (0.009)
inject-post-srnn	0.555 (0.006)	0.375 (0.003)	0.249 (0.003)	0.164 (0.003)
merge-concat-srnn	0.551 (0.004)	0.366 (0.004)	0.241 (0.003)	0.156 (0.002)
inject-init-srnn	0.546 (0.003)	0.356 (0.001)	0.229 (0.001)	0.147 (0.001)
inject-par-lstm	0.544 (0.004)	0.365 (0.005)	0.240 (0.006)	0.155 (0.006)
inject-post-lstm	0.542 (0.012)	0.366 (0.008)	0.242 (0.006)	0.158 (0.005)
inject-pre-lstm	0.540 (0.019)	0.357 (0.013)	0.230 (0.008)	0.147 (0.007)
inject-pre-srnn	0.535 (0.013)	0.347 (0.008)	0.224 (0.006)	0.144 (0.005)
vinyals	0.535 (0.007)	0.354 (0.006)	0.228 (0.005)	0.144 (0.004)
inject-init-lstm	0.530 (0.004)	0.350 (0.004)	0.223 (0.004)	0.141 (0.004)
inject-par-srnn	0.529 (0.008)	0.354 (0.006)	0.234 (0.003)	0.151 (0.003)
langmodel-srnn	0.415 (0.044)	0.192 (0.027)	0.106 (0.002)	0.064 (0.005)
langmodel-lstm	0.401 (0.026)	0.176 (0.039)	0.099 (0.031)	0.059 (0.019)

(b) BLEU-1, BLEU-2, BLEU-3, and BLEU-4 results.

Table 4: Results of the caption generation metrics. Higher is better.

then, that visual information disrupts the processing of linguistic information, as evidenced by the better performance of merge and post-inject models, compared to pre- and par-inject. A better strategy seems to be to encode visual and linguistic information separately prior to the creation of a mixed, or multimodal, representation.

6 Conclusion

This paper presented a systematic evaluation of a number of variations on architectures for image caption generation and retrieval. The primary focus was on the distinction between what we have termed 'inject' and 'merge' architectures. The former type of model mixes image and language information, training an RNN to vectorise an image-language prefix mixture. By contrast, merge architectures maintain a separation between an RNN subnetwork, which encodes a linguistic string, and the image vector, merging them late in the process, prior to a prediction module. These models are therefore compatible with approaches to image caption generation using a 'multimodal' layer (Kiros et al., 2014b,a, Mao et al., 2014, 2015a, Hendricks et al., 2016, Song and Yoo, 2016). A related, though distinct question we addressed concerns the stage at which image

	known vocab used	unigram entropy	bigram entropy
merge-concat-srnn	**6.560% (0.344%)**	5.756 (0.077)	7.596 (0.149)
mao	6.339% (0.124%)	5.815 (0.015)	7.754 (0.059)
merge-concat-lstm	6.321% (0.370%)	5.729 (0.057)	7.620 (0.141)
merge-add-lstm	6.231% (0.115%)	5.736 (0.012)	7.620 (0.034)
inject-post-lstm	6.169% (0.191%)	**5.931 (0.053)**	**7.894 (0.061)**
merge-add-srnn	5.926% (0.193%)	5.625 (0.047)	7.399 (0.100)
inject-par-lstm	5.795% (0.094%)	5.832 (0.035)	7.756 (0.070)
inject-par-srnn	5.737% (0.647%)	5.722 (0.166)	7.548 (0.211)
merge-mult-lstm	5.710% (0.234%)	5.591 (0.069)	7.402 (0.104)
inject-init-lstm	5.539% (0.154%)	5.798 (0.041)	7.725 (0.055)
merge-mult-srnn	5.391% (0.441%)	5.567 (0.076)	7.307 (0.115)
inject-pre-lstm	5.341% (0.173%)	5.768 (0.076)	7.689 (0.106)
vinyals	5.332% (0.464%)	5.787 (0.066)	7.689 (0.136)
inject-post-srnn	5.260% (0.585%)	5.646 (0.098)	7.465 (0.182)
inject-init-srnn	3.444% (0.199%)	5.413 (0.045)	7.092 (0.089)
inject-pre-srnn	3.242% (0.138%)	5.391 (0.011)	6.970 (0.098)
langmodel-srnn	0.121% (0.000%)	3.154 (0.023)	3.057 (0.080)
langmodel-lstm	0.117% (0.017%)	3.081 (0.242)	2.992 (0.148)
human-all	56.771%	8.210	12.428
human-one	32.520%	8.109	11.846

Table 5: Results of the caption diversity metrics. Higher is better.

	R@1	R@5	R@10
mao	**21.927% (0.390%)**	47.647% (0.577%)	**59.813% (0.294%)**
merge-add-srnn	21.913% (0.204%)	**47.740% (0.343%)**	59.420% (0.412%)
merge-concat-srnn	21.833% (0.141%)	47.420% (0.445%)	59.647% (0.446%)
merge-concat-lstm	21.573% (0.639%)	47.313% (0.262%)	59.480% (0.247%)
merge-add-lstm	21.233% (0.213%)	46.773% (0.710%)	58.793% (0.238%)
inject-post-srnn	20.987% (0.312%)	46.233% (0.236%)	58.653% (0.217%)
merge-mult-lstm	20.947% (0.542%)	46.520% (0.107%)	58.600% (0.482%)
merge-mult-srnn	18.900% (0.283%)	43.820% (0.242%)	56.033% (0.554%)
inject-post-lstm	18.447% (0.571%)	43.733% (0.766%)	55.913% (0.457%)
inject-par-srnn	18.040% (0.142%)	42.120% (0.261%)	54.993% (0.489%)
inject-par-lstm	17.247% (0.184%)	41.727% (0.407%)	54.333% (0.421%)
inject-pre-lstm	14.860% (0.212%)	37.620% (0.306%)	50.287% (0.520%)
vinyals	14.787% (0.360%)	37.773% (0.186%)	50.660% (0.157%)
inject-init-lstm	14.513% (0.148%)	37.447% (0.346%)	49.780% (0.748%)
inject-init-srnn	7.233% (0.431%)	22.093% (0.670%)	33.013% (0.591%)
inject-pre-srnn	7.100% (0.435%)	22.267% (0.451%)	32.980% (0.425%)

Table 6: Results of the image retrieval metrics. Language model architectures were left out as they cannot be used to retrieve. Legend: R@1,5,10 - recall at 1, 5, 10. Higher is better.

information is incorporated in the generation process, with inject architectures permitting a variety of strategies for early or late insertion.

While both types of architectures have been exploited in the literature, the inject architecture has been more popular. Yet, there has been little systematic evaluation of its advantages compared to merge. Our experiments show that the late binding architectures such as merge and post-inject are superior to early and mixed binding architectures such as init-inject, pre-inject, and par-inject, in any of the configurations used in the experiments. This is the case whether the models are evaluated on the basis of perplexity, n-gram overlap against the test set, or vocabulary diversity (though all models turn out to be highly conservative on the latter set of measures). Late binding architectures also perform better in image retrieval tasks.

Two main conclusions can be drawn from this work. First, our results show that treating image features on a par with linguistic information, and ‘mixing’ them in the sequence processing part of the model, is not optimal. Rather, as in the late binding models, it seems better to separate the linguistic encoding and image features, merging them at the end. In short, in multimodal tasks where language is being grounded in visual information, the latter should not be treated as 'another word'. Another explanation for why late binding should be better than early binding is because by treating the images as words, the RNN is effectively having to deal with a much larger vocabulary, since the vocabulary would include all the images in the training set apart from all the different words. This can compromise performance. In general it seems that introducing 'extra' information to the RNN disrupts the RNN's encoding process, which can be seen from the fact that par-inject architectures perform worse than post-inject.

A second set of conclusions is related to our view of the function of RNNs. As noted in Section 2, we postulate two perspectives, a continuous and a discontinuous view. The latter makes one important feature of RNNs explicit: at any time step, the RNN is encoding a prefix, and the result is used to predict the next element in the sequence. Thus, the RNN is not really 'generating text'; were the RNN truly generating the text, then it would need to know which image it was generating text for, as in an early binding model, but this seems to degrade performance compared to the late binding alternative. Against this background, it becomes clearer why late binding architectures outperform early binding ones: prediction in these architectures is based on a language prefix and an image, which are put together prior to the prediction stage. In sum, it is better to keep the RNN exclusively for encoding linguistic information, that is, interpreting sequences for other layers in the neural network. It would seem that making the RNN do more than simply encoding a linguistic sequence is not ideal.

The work presented here opens up some avenues for future research. As we noted at the outset, there are some similarities between our view of deep learning architectures and previous work on neural machine translation models (Sutskever et al., 2014, Bahdanau et al., 2014). In future work, we hope to investigate whether conditioning by merge is also the best method of conditioning the sentence generator in a language translation model or a model for other sequence-to-sequence tasks, such as question answering. This would also shed light on the similarities and differences between a range of NLP tasks, as shown by other work on sequence-to-sequence modelling (Sutskever et al., 2014).

Furthermore, by keeping language and image information separate, merge architectures lend themselves to potentially greater portability and ease of training. For example, it should be possible in principle to take the parameters of the RNN and embedding layers of a general text language model and transfer them to the corresponding layers in a caption generator. This would reduce training time as it would avoid learning the RNN weights and the embedding weights of the caption generator from scratch. As understanding of deep learning architectures evolves in the NLP community, one of our goals should be to maximise the degree of transferability among model components.

References

Anderson, P., Fernando, B., Johnson, M., and Gould, S. (2016). *SPICE: Semantic Propositional Image Caption Evaluation*, pages 382–398. Springer International Publishing, Cham.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, abs/1409.0473.

Banerjee, S. and Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, volume 29, pages 65–72.

Bernardi, R., Cakici, R., Elliott, D., Erdem, A., Erdem, E., Ikizler-Cinbis, N., Keller, F., Muscat, A., and Plank, B. (2016). Automatic Description Generation from Images: A Survey of Models, Datasets, and Evaluation Measures. *Journal of Artificial Intelligence Research*, 55:409–442.

Chen, X. and Zitnick, C. L. (2014). Learning a recurrent visual representation for image caption generation. *CoRR*, abs/1411.5654.

Chen, X. and Zitnick, C. L. (2015). Mind's eye: A recurrent visual representation for image caption generation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Institute of Electrical and Electronics Engineers (IEEE).

Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR*, abs/1412.3555.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. Institute of Electrical and Electronics Engineers (IEEE).

Devlin, J., Cheng, H., Fang, H., Gupta, S., Deng, L., He, X., Zweig, G., and Mitchell, M. (2015). Language Models for Image Captioning: The Quirks and What Works. *CoRR*, abs/1505.01809.

Donahue, J., Hendricks, L. A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2015). Long-term Recurrent Convolutional Networks for Visual Recognition and Description. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Institute of Electrical and Electronics Engineers (IEEE).

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256.

Harnad, S. (1990). The symbol grounding problem.

Hendricks, L. A., Venugopalan, S., Rohrbach, M., Mooney, R., Saenko, K., and Darrell, T. (2016). Deep Compositional Captioning: Describing Novel Object Categories without Paired Training Data. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Institute of Electrical and Electronics Engineers (IEEE).

Hessel, J., Savva, N., and Wilber, M. J. (2015). Image Representations and New Domains in Neural Image Captioning. *CoRR*, abs/1508.02091.

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

Hodosh, M., Young, P., and Hockenmaier, J. (2013). Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics. *Journal of Artificial Intelligence Research*, 47(1):853–899. Flickr8k.

Karpathy, A. and Fei-Fei, L. (2015). Deep Visual-Semantic Alignments for Generating Image Descriptions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Institute of Electrical and Electronics Engineers (IEEE).

Kiros, R., Salakhutdinov, R., and Zemel, R. S. (2014a). Multimodal neural language models. In *Proceedings of The 31st International Conference on Machine Learning*, page 595603.

Kiros, R., Salakhutdinov, R., and Zemel, R. S. (2014b). Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*.

Krause, J., Johnson, J., Krishna, R., and Fei-Fei, L. (2016). A hierarchical approach for generating descriptive image paragraphs. In *ArXiv*.

Lin, C.-Y. and Och, F. J. (2004). Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics - ACL '04*. Association for Computational Linguistics (ACL).

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755. Springer Nature.

Liu, S., Zhu, Z., Ye, N., Guadarrama, S., and Murphy, K. (2016). Optimization of image description metrics using policy gradient methods. *CoRR*, abs/1612.00370.

Lu, J., Xiong, C., Parikh, D., and Socher, R. (2016). Knowing when to look: Adaptive attention via A visual sentinel for image captioning. *CoRR*, abs/1612.01887.

Ma, S. and Han, Y. (2016). Describing images by feeding LSTM with structural words. In *2016 IEEE International Conference on Multimedia and Expo (ICME)*. Institute of Electrical and Electronics Engineers (IEEE).

Mao, J., Xu, W., Yang, Y., Wang, J., Huang, Z., and Yuille, A. (2015a). Deep Captioning with Multimodal Recurrent Neural Networks (m-RNN). *ICLR*.

Mao, J., Xu, W., Yang, Y., Wang, J., Huang, Z., and Yuille, A. (2015b). Learning like a Child: Fast Novel Visual Concept Learning from Sentence Descriptions of Images. In *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile. Institute of Electrical and Electronics Engineers (IEEE).

Mao, J., Xu, W., Yang, Y., Wang, J., and Yuille, A. L. (2014). Explain images with multimodal recurrent neural networks. *NIPS Deep Learning Workshop*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781.

Mnih, A. and Hinton, G. (2007). Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning - ICML '07*. Association for Computing Machinery (ACM).

Nina, O. and Rodriguez, A. (2015). Simplified LSTM unit and search space probability exploration for image description. In *2015 10th International Conference on Information, Communications and Signal Processing (ICICS)*. Institute of Electrical and Electronics Engineers (IEEE).

Oruganti, R. M., Sah, S., Pillai, S., and Ptucha, R. (2016). Image description through fusion based recurrent multi-modal learning. In *2016 IEEE International Conference on Image Processing (ICIP)*. Institute of Electrical and Electronics Engineers (IEEE).

P. Kingma, D. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Pascanu, R., Mikolov, T., and Bengio, Y. (2012). Understanding the exploding gradient problem. *Computing Research Repository (CoRR) abs/1211.5063*.

Rennie, S. J., Marcheret, E., Mroueh, Y., Ross, J., and Goel, V. (2016). Self-critical sequence training for image captioning. *CoRR*, abs/1612.00563.

Roy, D. (2005). Semiotic schemas: A framework for grounding language in action and perception. *Artificial Intelligence*, 167(1-2):170–205.

Simonyan, K. and Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, abs/1409.1556.

Song, M. and Yoo, C. D. (2016). Multimodal representation: Kneser-ney smoothing/skip-gram based neural language model. In *2016 IEEE International Conference on Image Processing (ICIP)*. Institute of Electrical and Electronics Engineers (IEEE).

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Vedantam, R., Zitnick, C. L., and Parikh, D. (2015). CIDEr: Consensus-based image description evaluation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Institute of Electrical and Electronics Engineers (IEEE).

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Institute of Electrical and Electronics Engineers (IEEE).

Wang, M., Song, L., Yang, X., and Luo, C. (2016). A parallel-fusion RNN-LSTM architecture for image caption generation. In *2016 IEEE International Conference on Image Processing (ICIP)*. Institute of Electrical and Electronics Engineers (IEEE).

Wu, Q., Shen, C., van den Hengel, A., Liu, L., and Dick, A. R. (2015). Image captioning with an intermediate attributes layer. *CoRR*, abs/1506.01144.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A. C., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of The 32nd International Conference on Machine Learning*, volume abs/1502.03044, page 20482057.

Yang, Z., Yuan, Y., Wu, Y., Salakhutdinov, R., and Cohen, W. W. (2016). Encode, review, and decode: Reviewer module for caption generation. *CoRR*, abs/1605.07912.

Yao, T., Pan, Y., Li, Y., Qiu, Z., and Mei, T. (2016). Boosting image captioning with attributes. *CoRR*, abs/1611.01646.

You, Q., Jin, H., Wang, Z., Fang, C., and Luo, J. (2016). Image captioning with semantic attention. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Institute of Electrical and Electronics Engineers (IEEE).

Young, P., Lai, A., Hodosh, M., and Hockenmaier, J. (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.

Zhou, L., Xu, C., Koch, P., and Corso, J. J. (2016). Image caption generation with text-conditional semantic attention. *CoRR*, abs/1606.04621.