Natural Language Engineering **24** (3): 467–489. © Cambridge University Press 2018
doi:10.1017/S1351324918000098

Where to put the image in an image caption generator†

MARC TANTI[1], ALBERT GATT[1]
and KENNETH P. CAMILLERI[2]
[1]*Institute of Linguistics and Language Technology*
University of Malta, Msida MSD, Malta
e-mail: marc.tanti.06@um.edu.mt, albert.gatt@um.edu.mt
[2]*Department of Systems and Control Engineering*
University of Malta, Msida MSD, Malta
e-mail: kenneth.camilleri@um.edu.mt

(*Received 19 March 2017; revised 20 February 2018; accepted 20 February 2018*)

Abstract

When a recurrent neural network (RNN) language model is used for caption generation, the image information can be fed to the neural network either by directly incorporating it in the RNN – conditioning the language model by 'injecting' image features – or in a layer following the RNN – conditioning the language model by 'merging' image features. While both options are attested in the literature, there is as yet no systematic comparison between the two. In this paper, we empirically show that it is not especially detrimental to performance whether one architecture is used or another. The merge architecture does have practical advantages, as conditioning by merging allows the RNN's hidden state vector to shrink in size by up to four times. Our results suggest that the visual and linguistic modalities for caption generation need not be jointly encoded by the RNN as that yields large, memory-intensive models with few tangible advantages in performance; rather, the multimodal integration should be delayed to a subsequent stage.

1 Introduction

Image caption generation is the task of generating a natural language description of the content of an image (Bernardi *et al.* 2016), also known as a caption. One way to do this is to use a neural language model, typically in the form of a recurrent neural network (RNN), which is used to generate text (illustrated in Figure 1). Given a sentence prefix, a neural language model will predict which words are likely to follow. With a small modification, this simple model can be extended into an image caption generator, that is, a language model whose predictions are conditioned on image features. To do this, the neural language model must somehow accept as

†The research in this paper 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".

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press

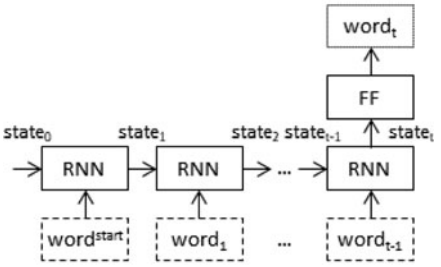


Fig. 1. How RNN-based neural language models work. Legend: RNN – Recurrent Neural Network; FF – Feed-Forward layer; word$_i$ – the ith generated word in the text; word^{start} – the START token which is an artificial word placed at the beginning of every sentence in order to still have a prefix when predicting the first word (likewise there is an END token to predict the end of a sentence). Note that state$_1$ represents the prefix 'word^{start}', state$_2$ represents the prefix 'word^{start} word$_1$', *etc.* After processing a prefix, the RNN passes its final state state$_t$ to a feed-forward layer, which then predicts how likely each known word is to be the next word in the prefix.

input not only the sentence prefix, but also the image being captioned. This raises the question: *At which stage should image information be introduced into a 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 cognitive scientists (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 three distinct sets of design choices:

Conditioning by injecting versus conditioning by merging: A neural language model can be conditioned by injecting the image (Figure 1a) or by merging the image (Figure 1b). In '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 it on a par with a 'word' and including it as part of the caption prefix. The RNN is trained to encode the image–language mixture into a single vector in such a way that this vector can be used to predict the next word in the prefix. On the other hand, 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 encoded, the image vector is then merged with the prefix vector in a separate 'multimodal layer' that comes after the RNN subnetwork. Merging can be done by, for example, concatenating the two vectors together. In this case, the RNN is trained to only encode the prefix and the mixture is handled in a subsequent feed-forward layer.

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press

In the terminology adopted in this paper: *if an RNN's hidden state vector is somehow influenced by both the image and the words then the image 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. On the other hand, some inject architectures tend to incorporate image features early in the generation process. Other inject architectures incorporate image features for the whole duration of the generation process. Different architectures can make visual information influence linguistic choices at different stages.

Fixed versus modifiable image features: For each word predicted, some form of visual information must be available to influence the likelihood of each word. Merge architectures typically use the exact same image representation for every word output. On the other hand, injecting the image features into the RNN allows the internal representation of the image inside the hidden state vector to be changed by the RNN's internal updates after each time step. Different architectures allow for different degrees of modification in the image features for each generated word.

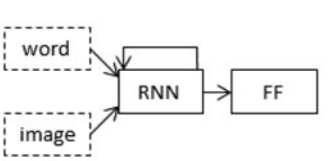
The main contribution of this paper is to present a systematic comparison of the different ways in which the 'conditioning' of linguistic choices based on visual information can be carried out, studying their implications for caption generator architectures. Thus, rather than seeking new results that 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.[1]

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 that vectorise a linguistic prefix so that the next feed-forward layer can predict the next word, conditioned on an image? Answers to these questions would help inform theories of how caption generation can be performed. The architectures we compare provide different answers to these questions. Hence, it is important to acquire some insights into their relative merits.

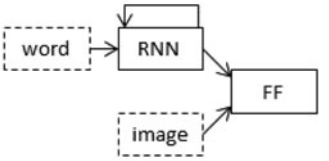
From an engineering perspective, insights into the relative performance of different models could provide rules of thumb for selecting an architecture for the task of image captioning, possibly for other tasks as well such as machine translation. This would make it easier to develop new architectures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

[1] All the code used in our experiments is available at `https://github.com/mtanti/where-image2`

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press



(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.

Fig. 2. The inject and merge architectures for caption generation. Legend: RNN – Recurrent Neural Network; FF – Feed-Forward layer.

The remainder of this paper is structured as follows. We first give an overview of published caption generators based on neural language models, focusing in particular on the architectures used. 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

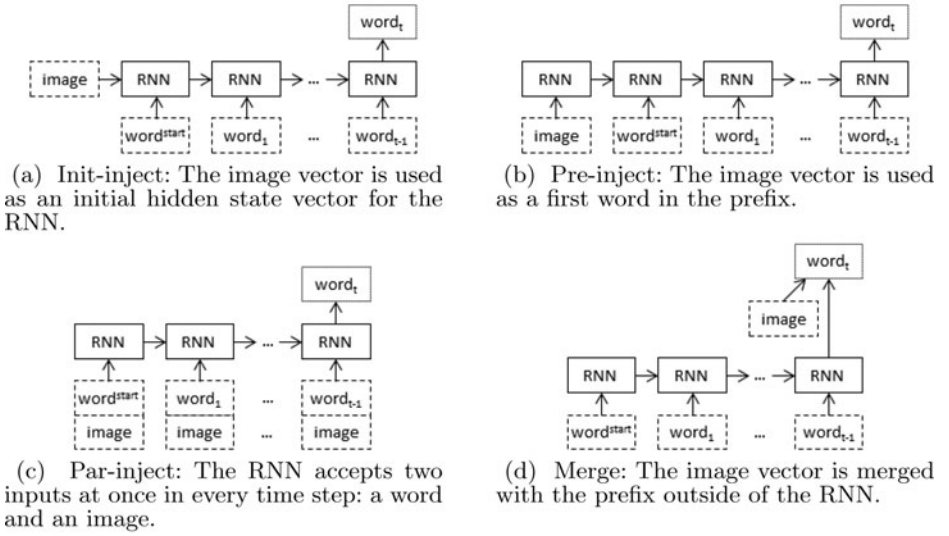
In this section, we discuss a number of recent image caption generation models with emphasis on how the image conditions the neural language model, based on the distinction between inject and merge architectures illustrated in Figure 2. Before we discuss these models, we first outline four broad sub-categories of architectures that we have identified in the literature.

2.1 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. We can in fact distinguish four theoretical possibilities arising from these, as illustrated in Figure 3 and described below.

- Init-inject: The RNN's initial hidden state vector is set to be the image vector (or a vector derived from the image vector). It requires the image vector to have the same size as the RNN hidden state vector. This is an early binding architecture and allows the image representation to be modified by the RNN.
- 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. It requires the image vector to have the same size as the word vectors. This too is an early binding architecture and allows the image representation to be modified by the RNN.

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press



(a) Init-inject: The image vector is used as an initial hidden state vector 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 in every time step: a word and an image.

(d) Merge: The image vector is merged with the prefix outside of the RNN.

Fig. 3. Different ways of conditioning a neural language model with an image. The feed-forward layer was left out to save space.

- 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 does not need to be exactly the same for each word (such as is the case with attention-based neural models); nor does it need to be included with every word. This is a mixed binding architecture and, whilst allowing some modification in the image representation, it will be harder for the RNN to do so if the same image is fed to the RNN at every time step due to its hidden state vector being refreshed with the original image each time.
- Merge: The RNN is not exposed to the image vector (or a vector derived from the image vector) at any point. Instead, the image is introduced into the language model after the prefix has been encoded by the RNN in its entirety. This is a late binding architecture and it does not modify the image representation with every time step.

With these distinctions in mind, we next discuss a selection of recent contributions, placing them in the context of this classification.

Init-inject architectures: Architectures conforming to the init-inject model treat the image vector as the initial hidden state vector of an RNN (Devlin *et al.* 2015; Liu *et al.* 2016). Wang *et al.* (2016) combine two RNNs in parallel, both initialized with the same image.

A similar architecture to init-inject is used in traditional deep learning machine translation systems (Sutskever, Vinyals and Le 2014) where a source sentence is

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press

encoded into a vector and used to condition a language model to generate a sentence in another language. This is the basis for the system described by Ma and Han (2016), who first extract a sequence of attributes from an image, then translate this sequence into a caption.

It is also used in attention mechanisms in order to provide a vector representing information about the whole image, whilst parts of the image that are attended differently during each time step are provided via par-injection. For example, Xu *et al.* (2015) initialize the RNN with the centroid of all image parts before attending to some parts as needed.

Pre-inject architectures: Pre-inject models treat the image as though it were the first word in the prefix (Nina and Rodriguez 2015; Vinyals *et al.* 2015; Rennie *et al.* 2017). Image attributes are sometimes used instead of image vectors (Wu *et al.* 2015; Yao *et al.* 2017). Yao *et al.* (2017) also try passing an image as the first two words instead of just one word by using the image vector as the first word and image attributes as a second, or vice versa.

Just like init-inject, pre-inject is also used to provide information about the whole image in attention mechanisms (You *et al.* 2016; Zhou *et al.* 2016).

Par-inject architectures: Par-injection inputs the image features into the RNN jointly with each word in the caption. It is by far the most common architecture used and has the largest variety of implementation. For example, Donahue *et al.* (2015) do this with two RNNs in series and find that it is better to inject the image in the second RNN than the first. Yao *et al.* (2017) par-inject the image whilst pre-injecting image attributes (or vice versa); and Liu *et al.* (2016) par-inject attributes from the image whilst init-injecting the image vector. Other, less common instantiations include par-injecting the image, but only with the first word (this is not pre-inject as the image is not injected on a separate time step) (Karpathy and Fei-Fei 2015; Hessel, Savva and Wilber 2015); and passing the words through a separate RNN, such that the resulting hidden state vectors are what is combined with the image vector (Oruganti *et al.* 2016).

Many times this architecture is used in order to pass a different representation of the same image with every word so that visual information changes for different parts of the sentence being generated. For example, Zhou *et al.* (2016) perform element-wise multiplication of the image vector with the last generated word's embedding vector in order to attend to different parts of the image vector. Oruganti *et al.* (2016) pass the image through its own RNN for as many times as there are words in order to use a different image vector for every word. Chen and Zitnick (2014; 2015) use a simple RNN to try to predict what the image vector looks like given a prefix. This predicted image is then used as a second image representation, which is par-injected together with the actual image vector.

More commonly, modified image representations come from attention mechanisms (Xu *et al.* 2015; You *et al.* 2016; Rennie *et al.* 2017). Rennie *et al.* (2017) inject the image not as an input to the RNN but use a modified long short-term memory network (Hochreiter and Schmidhuber 1997), or LSTM, which allows them to inject

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press

the attended image directly inside the input gated expression (the part of the LSTM which is multiplied by the input gate).

Like init-inject and pre-inject, par-inject is sometimes used to provide information about the whole image in attention mechanisms whilst the attended image regions are merged (Lu *et al.* 2017).

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 (Mao *et al.* 2014, 2015a, 2015b). Hendricks *et al.* (2016) use a merge architecture in order to keep the image out of the RNN and thus be able to train the part of the neural network that handles images and the part that handles language separately, using images and sentences from separate training sets.

Some work on attention mechanisms also uses merge architectures with attention mechanisms by merging a different image representation at every time step. You *et al.* (2016) and Xu *et al.* (2015) merge as well as par-inject the attended visual regions, whilst Lu *et al.* (2017) only merge the regions whilst par-injecting a fixed image representation.

Though they do not use an RNN and hence are not focussed on in this review, caption generators that use log-bilinear models (Mnih and Hinton 2007) usually merge the image with the prefix representation (Kiros, Salakhutdinov and Zemel 2014a, 2014b; Song and Yoo 2016).

2.2 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 performances 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 architectures with 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 based on a survey of the literature alone.

In what follows, we present a systematic comparison between all the different architectures discussed above. We perform these evaluations using a common dataset and a variety of quality metrics, covering (a) the quality of the generated captions; (b)

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press

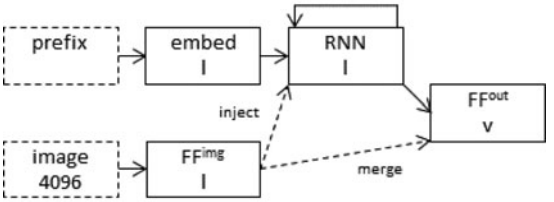


Fig. 4. An illustration of the main architecture schema that is instantiated in the four different architectures tested in this paper. Legend: 'FF' – fully connected layer with bias; 'FF^{img}' – layer projecting the image vector (may or may not have an activation function); 'FF^{out}' – layer projecting into the softmax output; 'I' – the layer size (which is the same for three different layers); 'v' – the vocabulary size (which is different for different datasets). Only one of the dashed arrows is used depending on whether the architecture is one of merge or inject.

the linguistic diversity of the generated captions; and (c) the networks' capabilities to determine the most relevant image given a caption.

3 Architectures

In this section, we go over the different architectures that are evaluated in this paper. A diagram illustrating the main architecture schema, which is the basis of every tested architecture, is shown in Figure 4. The schema is based on the architecture described by Vinyals *et al.* (2015), without the ensemble. This architecture was chosen for its simplicity, whilst still being the best performing system in the 2015 MSCOCO image captioning challenge.[2]

Word embeddings: Word embeddings, that is, the vectors that represent known words prior to being fed to the RNN, consist of 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 for the task.

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. A gated recurrent unit (Chung *et al.* 2014), or GRU, was used in our experiments for the simple reason that it is a powerful RNN that only has one hidden state vector. By contrast, an LSTM has two state vectors (hidden and cell states). This would make architecture comparisons more complex, as the presence of two state vectors raise the possibility of multiple versions of the init-inject architecture. By using an RNN with a single hidden state vector there is only one way to implement init-inject.

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

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press

Image: Prior to training, all images were vectorised using the activation values of the penultimate layer of the VGG OxfordNet nineteen-layer convolutional neural network (Simonyan and Zisserman 2014), which is trained to perform object recognition and returns a 4096-element vector. The convolutional neural network is not influenced by the caption generation training process. During training, a feed-forward layer of the neural network compresses this vector into a smaller vector.

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 feed-forward layer with a softmax activation function that outputs the probability of each possible next word in the vocabulary. Based on this distribution, the next word that comes after the prefix is selected.

The four architectures discussed in the previous section are evaluated in our experiments as follows:

- init-inject: The image vector is treated as an initial hidden state vector for the RNN. After initialising the RNN, the vectors in the caption prefix are then fed to the RNN as usual.
- pre-inject: The image vector is used as the first 'word' in the caption prefix. This makes the image vector the first input that the RNN will see.
- par-inject: The image vector is concatenated 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 concatenated to it.
- merge: The image vector and caption prefix vector (RNN state vector) are concatenated into a single vector before being fed to the output layer.

4 Experiments

This section describes the experiments conducted in order to compare the performance of the different architectures described in the previous section. Tensorflow[3] v1.2 was used to implement the neural networks.

4.1 Datasets

The datasets used for all experiments were the version of Flickr8K (Hodosh, Young and Hockenmaier 2013), Flickr30K (Young *et al.* 2014), and MSCOCO (Lin *et al.* 2014) distributed by Karpathy and Fei-Fei (2015).[4] All three datasets consist of images taken from Flickr combined with between five and seven manually written captions per image. The provided datasets are split into a training, validation, and test set using the following number of images respectively: Flickr8K – 6000, 1000, 1000; Flickr30K – 29000, 1014, 1000; MSCOCO – 82783, 5000, 5000. The images

[3] See: `https://www.tensorflow.org/`
[4] See: `http://cs.stanford.edu/people/karpathy/deepimagesent/`

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press

are already vectorised into 4096-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).

The known vocabulary consists of all the words in the captions of the training set that occur at least five times. This amounts to 2539 for Flickr8K, 7415 for Flickr30K, and 8792 for MSCOCO. These words are used both as inputs, which are embedded and fed to the RNN, and as outputs, which are assigned probabilities by the softmax function. Any other word which is not part of the vocabulary is replaced with an UNKNOWN token.

4.2 Hyperparameter tuning

For the results to be reliable, it is important to find the best (within practical limits) hyperparameters for each architecture so that we can judge the performance of the architectures when they are optimally tuned, rather than using one-size-fits-all hyperparameter settings that might cause some architectures to under-perform. For this reason, we used a multi-step process of hyperparameter tuning, which is described below. We optimized the hyperparameters in order to maximize caption quality on the Flickr8K validation set, using beam search as a generation method and CIDEr as the objective function. The optimal hyperparameters were then fixed across all datasets. Mao *et al.* (2015a) also used Flickr8K for hyperparameter tuning and CIDEr was shown by Rennie *et al.* (2017) to be a useful metric to optimise on, yielding an improvement on other quality metrics when used as the objective function.

The following hyperparameters were fixed across all architectures:

- Parameter optimization is performed using the Adam algorithm (Diederik and Ba 2014) with its hyperparameters kept as suggested in the original paper: $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$.
- Loss function is the mean of the cross-entropy of each word in each caption in a minibatch.
- An early stopping criterion is used, such that the geometric mean of the language model perplexity on the validation set is measured and as soon as one epoch results in a worse perplexity than the previous epoch, the training stops. A maximum number of epochs are still used to prevent training from going on for too long (more on this later).
- During caption generation, the caption must be between five and fifty words long. Beam search will not end a sentence before there are at least five words in it and will abruptly stop a sentence that is fifty words long.
- All biases are initialized to zeros.

The following are hyperparameters that were tuned (the ranges of values were minimized in order to keep the search space tractable):

- The weights initialization procedure (normal distribution or xavier (Glorot and Bengio 2010) with normal distribution).

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press

- The weights initialization range (−0.1 to 0.1 or −0.01 to 0.01).
- The size of the layers for embedding, image projection (FF^{img} in Figure 4), and RNN hidden state vector (64, 128, 256, or 512), all three of which are constrained to be equal.[5]
- Whether to normalize the image vector before passing it to the neural network.
- Whether to use ReLU after the image projection (FF^{img} in Figure 4) or to leave it linear.
- Whether to use an all-zeros vector as an initial RNN hidden state vector or to use a learnable vector (not applicable to init-inject since its initial hidden state vector is the image projection).
- Whether to use L2 weights regularization with a weighting constant of 10^{-8}.
- Whether to apply dropout regularisation at different points in the architecture (in Figure 4: after 'image', after 'FF^{img}', after 'embed', and/or after 'RNN'). Each application of dropout (if any) has a dropout rate of 0.5.
- The minibatch size (32, 64, or 128).

The following steps were followed in order to tune these hyperparameters, which were evaluated by training a neural network for a maximum of ten epochs, generating captions with a beam width of 2, and evaluating the captions using CIDEr:

(1) Randomly generate 100 unique hyperparameter combinations and record their performance.
(2) Use Baysian optimization via the library GPyOpt[6] for 100 iterations and record each generated candidate combination's performance. Use the combinations from step 1 to initialize the search.
(3) Use trees of Parzan estimators via the library hyperopt[7] for 100 iterations and record each generated candidate combination's performance.
(4) Take the best combination found in all of the previous steps and fine-tune it using greedy hill climbing and record each modified combination. This is to check if changing any one hyperparameter will improve the performance.

The previous steps do not have very reliable CIDEr scores associated with them as their score was produced using just one training and generation run and so might coincidentally be an unusual score (far from the mean score if we trained the same neural network several times). Ideally, we would have tested each hyperparameter combination three times and taken the mean of the resulting CIDEr scores. Ideally, we would have also tried different values for maximum number of epochs and beam width. This, however, would have been extremely time consuming. Thus, we

[5] Note that if we allowed each layer to change freely from the other layers, init-inject would still require that the image size and RNN size be equal and pre-inject would still require that the image size and the embedding size be equal, whilst par-inject and merge would have no such size restrictions. This would make the former two architectures have significantly less hyperparameter combinations to explore that would likely result in an unfair advantage after hyperparameter tuning.
[6] See: `http://sheffieldml.github.io/GPyOpt/`
[7] See: `https://jaberg.github.io/hyperopt/`

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press

only apply the procedure to a subset of the best performing combinations from the previous steps. We ensure that the subset is diverse by only choosing combinations that are dissimilar from each other, as follows:

(5) Take all duplicate combinations generated in all of the previous steps and replace them with a single combination with their average CIDEr score. Take the top ten scoring combinations.
(6) Out of the selected ten combinations take the three combinations that are most different from each other in terms of Hamming distance. Ensure that one of these three combinations is the best combination found in the previous step.
(7) Take the three combinations selected and try different maximum epochs (ten and 100) and beam widths (1, 2, 3, 4, 5, and 6) on them. Each evaluation is measured using the average CIDEr score of three independent training and generation runs.
(8) Return the best combination found in the previous step.

In Section 5.1, we will discuss the optimal hyperparameters found.

4.3 Evaluation metrics

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

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[8] that measures the standard evaluation metrics BLEU-(1,2,3,4) (Papineni *et al.* 2002), ROUGE-L (Lin and Och 2004), METEOR (Banerjee and Lavie 2005), and CIDEr (Vedantam, Zitnick and Parikh 2015).

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 the captions produced by models 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 that always outputs the same caption. Such a generator would have the lowest possible diversity score. 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

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

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press

the generated captions together, which is calculated as

$$\text{entropy}(F) = -\sum_{i=1}^{|F|} P_i(F) \log_2 P_i(F) \quad (1)$$

$$P_i(F) = \frac{F_i}{\sum_{j=1}^{|F|} F_j} \quad (2)$$

where F is the frequency distribution over generated unigrams or bigrams with $|F|$ different types of unigrams or bigrams and P_i is the maximum likelihood estimate probability of encountering unigram or bigram i. Note that F_i is the frequency of the unigram or bigram i.

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.

Finally, we also measure the percentage of generated captions that already exist in the training set, as an estimate of the extent to which a model evinces 'parroting', or wholesale caption reuse from the training set.

For these diversity metrics, we obtain a ceiling estimate by computing the same measures on the test set captions themselves. We take the first caption out of the group of human-written captions available for each image in the test set and apply these diversity metrics on them.

Retrieval metrics: Retrieval metrics are metrics that quantify how well the architectures perform when retrieving the correct image out of all the test set images in the test set given a corresponding 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.

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 among the top n most relevant images.

Since this process takes time proportional to the number of captions multiplied by the number of images, the pool of possible captions to consider during retrieval excluded all captions except the first out of the group of captions available for each image in order to reduce the evaluation time. For MSCOCO, we only used the first 1,000 test set images out of 5,000 for the same reason, similar to Flickr8K and Flickr30K that only have 1,000 images.

We also included the language model perplexity. In order to aggregate the caption perplexity of the entire test set of captions into a single number, we report the geometric mean of all the caption's scores.

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press

Table 1. *The optimal hyperparameters found for each architecture, tuned on Flickr8K with CIDEr as an objective function. Each row is explained in Section 4.2*

	Init-inject	Pre-inject	Par-inject	Merge
Init. method	Xavier	Normal	Normal	Normal
Init. weight range	−0.01 – 0.01	−0.1 – 0.1	−0.1 – 0.1	−0.1 – 0.1
Layer size	512	512	256	128
Normalize image	Yes	Yes	Yes	Yes
Image activation	None	None	None	None
Init. RNN hidden state	N/A	Zero	Learnable	Learnable
Regularize weights	No	No	Yes	No
Image dropout	No	No	Yes	No
Image proj. dropout	No	No	No	No
Embedding dropout	Yes	Yes	Yes	No
RNN dropout	Yes	Yes	Yes	Yes
Minibatch size	128	32	64	128
Max. epochs	100	100	100	100
Beam width	3	3	5	3

5 Results and discussion

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

5.1 Optimal hyperparameters

We start by discussing the optimal hyperparameters found for each architecture which are listed in Table 1.

It is interesting to note that, in every architecture's optimal hyperparameters, the RNN output needs to be regularized with dropout, the image vector should not have a non-linear activation function or be regularized with dropout, and the image input vector must be normalized before being fed to the neural network. Par-inject seems to need the most help in terms of regularization and even in terms of beam width, whilst the small size of merge means that it needs the least amount of regularization.

The most interesting observation is that the merge architecture is much 'leaner' overall. In terms of RNN size, it needs half of what par-inject needs, and only a quarter of what init-inject and pre-inject require for optimal performance. This makes sense, since merge only needs the RNN for storing linguistic information, whilst the other architectures need to additionally store visual information from the image. Using a larger RNN with the merge architecture would likely lead to overfitting.

The implication is that init-inject and pre-inject are much more memory-hungry architectures that require large RNN hidden state vectors in order to function well, whilst merge is more efficient. In fact, the number of parameters for merge is

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press

Table 2. *Results of caption quality metrics (CIDEr, METEOR, and ROUGE-L). Best values are in bold.*

	CIDEr	METEOR	ROUGE-L
Results for Flickr8K			
Init-inject	**0.481 (0.010)**	**0.194 (0.000)**	0.445 (0.002)
Par-inject	0.475 (0.004)	0.193 (0.002)	**0.448 (0.003)**
Pre-inject	0.469 (0.009)	0.191 (0.001)	0.444 (0.003)
Merge	0.469 (0.015)	0.193 (0.002)	0.443 (0.003)
Results for Flickr30K			
Merge	**0.385 (0.006)**	0.174 (0.000)	0.423 (0.001)
Init-inject	0.383 (0.005)	**0.177 (0.002)**	**0.425 (0.003)**
Pre-inject	0.380 (0.006)	0.174 (0.001)	0.420 (0.002)
Par-inject	0.361 (0.004)	0.170 (0.002)	0.418 (0.001)
Results for MSCOCO			
Init-inject	**0.818 (0.005)**	**0.226 (0.002)**	**0.499 (0.003)**
Pre-inject	0.807 (0.007)	0.224 (0.000)	0.498 (0.002)
Merge	0.791 (0.010)	0.222 (0.001)	0.494 (0.002)
Par-inject	0.774 (0.003)	0.219 (0.001)	0.493 (0.001)

between three and four times smaller than the number of parameters for init-inject and pre-inject. Merge is also about two or three times faster to train.

Of the inject architectures, par-inject has the smallest optimal RNN size. This is probably due to the fact that, in this model, the image is present at all-time steps, thereby necessitating less memory to be allocated to 'remember' visual information together with linguistic information, compared to early binding architectures. It's interesting to note that the par-inject RNN size is equal to the size of the concatenated image and RNN hidden state vector in the merge architecture.

5.2 Quality of generated captions

Tables 2 and 3 display the metrics that measure the quality of generated captions, calculated using the MSCOCO evaluation toolkit and averaged over the three experimental runs.

Does merge's small size impact its performance when generated captions are compared to corpora? The metrics reported here show considerable variability in ranking of the various architectures depending on dataset. For example, CIDEr scores place init-inject at the top for both Flickr8K and MSCOCO, but merge outperforms it on this measure on Flickr30K. Comparing ROUGE-L, METEOR, and CIDEr, init-inject seems to be ranked highest over most datasets (the situation is far more variable with the BLEU scores in Table 3, however). However, the

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press

Table 3. *Results of caption quality metrics (BLEU-1, BLEU-2, BLEU-3, and BLEU-4). Best values are in bold.*

	BLEU-4	BLEU-3	BLEU-2	BLEU-1
Results for Flickr8K				
Par-inject	**0.191 (0.003)**	**0.287 (0.003)**	**0.424 (0.002)**	**0.611 (0.001)**
Init-inject	0.191 (0.004)	0.285 (0.005)	0.424 (0.005)	0.611 (0.002)
Pre-inject	0.190 (0.003)	0.285 (0.004)	0.421 (0.005)	0.609 (0.007)
Merge	0.178 (0.004)	0.273 (0.005)	0.413 (0.006)	0.600 (0.007)
Results for Flickr30K				
	BLEU-4	BLEU-3	BLEU-2	BLEU-1
Pre-inject	**0.192 (0.002)**	**0.284 (0.001)**	**0.419 (0.003)**	0.613 (0.004)
Init-inject	0.191 (0.002)	0.283 (0.002)	0.419 (0.002)	0.613 (0.004)
Merge	0.187 (0.001)	0.280 (0.002)	0.419 (0.002)	**0.614 (0.002)**
Par-inject	0.183 (0.004)	0.275 (0.003)	0.410 (0.003)	0.605 (0.004)
Results for MSCOCO				
	BLEU-4	BLEU-3	BLEU-2	BLEU-1
Init-inject	**0.271 (0.002)**	**0.367 (0.002)**	**0.502 (0.002)**	**0.679 (0.003)**
Pre-inject	0.267 (0.002)	0.366 (0.003)	0.501 (0.003)	0.677 (0.002)
Par-inject	0.265 (0.003)	0.359 (0.004)	0.492 (0.004)	0.667 (0.003)
Merge	0.262 (0.003)	0.362 (0.003)	0.500 (0.003)	0.677 (0.003)

differences among architectures are very small. This is especially true for the larger MSCOCO dataset. Thus, though init-inject often comes out on top, the other architectures are not lagging behind by a wide margin.

5.3 Image retrieval

Image retrieval results across the three datasets are shown in Table 4.

When it comes to retrieving the most relevant image for a caption, we once again see merge ranked first on Flickr30K, while init-inject is at the top on Flickr8K and MSCOCO, on practically all $R@n$ measures, as well as median rank. Interestingly, in the two sets of cases where init-inject outperforms other architectures, merge is a close second, at least for $R@1$. In terms of perplexity, the general picture is in favour of inject models, with merge evincing marginally greater perplexity on all datasets. Overall, however, the outcomes mirror those of the previous sub-section: differences among architectures do not seem compelling and although the init-inject model outperforms merge in a number of instances, merge is a close second.

5.4 Caption diversity metrics

Next, we turn to the caption diversity metrics, shown in Table 5.

These diversity metrics evince the most dramatic performance differences. If we focus on the proportion of generated captions that were found in the training

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press

Table 4. *Results of the image retrieval metrics. Language model perplexity was also included here. Best values are in bold.*

	R@1 %	R@5 %	R@10 %	Med. rank	Pplx.
Results for Flickr8K					
Init-inject	**17.5 (0.6)**	**43.2 (1.1)**	**56.1 (0.4)**	**7.7 (0.5)**	13.70 (0.02)
Merge	17.3 (0.1)	41.1 (0.4)	54.6 (0.6)	8.5 (0.4)	13.96 (0.07)
Par-inject	15.8 (0.4)	39.7 (0.6)	53.3 (0.6)	9.0 (0.0)	**13.42 (0.08)**
Pre-inject	14.8 (0.2)	37.9 (0.8)	51.8 (1.1)	9.3 (0.5)	13.54 (0.13)
Results for Flickr30K					
	R@1 %	R@5 %	R@10 %	Med. rank	Pplx.
Merge	**24.2 (0.0)**	**51.0 (0.5)**	**61.5 (0.7)**	**5.0 (0.0)**	22.08 (0.03)
Par-inject	22.8 (0.8)	48.1 (0.4)	60.0 (0.9)	6.0 (0.0)	21.09 (0.46)
Init-inject	22.7 (0.7)	48.6 (0.3)	60.1 (0.1)	6.0 (0.0)	**19.50 (0.12)**
Pre-inject	21.5 (0.4)	48.1 (0.7)	60.0 (0.5)	6.0 (0.0)	20.18 (0.03)
Results for MSCOCO					
	R@1 %	R@5 %	R@10 %	Med. rank	Pplx.
Init-inject	**29.1 (0.9)**	**63.8 (0.9)**	**77.2 (0.8)**	**3.0 (0.0)**	**9.27 (0.03)**
Merge	28.7 (0.7)	62.2 (0.5)	74.8 (0.5)	**3.0 (0.0)**	10.40 (0.04)
Par-inject	27.2 (0.4)	58.4 (0.7)	73.3 (1.0)	4.0 (0.0)	10.07 (0.05)
Pre-inject	27.1 (0.9)	60.5 (0.6)	75.1 (0.2)	4.0 (0.0)	9.88 (0.04)

set, on MSCOCO, this figure ranges from just over forty percent for merge to over sixty percent for par-inject. With the exception of Flickr8K, merge has the lowest proportion of caption reuse overall. If these results are compared to those in preceding sub-sections, the fact that those models with the greatest tendency to reuse captions tend to perform well on corpus-based metrics such as CIDEr suggests that the datasets under consideration are highly stereotyped, perhaps with a significant amount of redundancy and lack of variety.

A similar observation has been made by Devlin *et al.* (2015). In a comparison of retrieval-based and neural architectures for image captioning, these authors found that corpus-based metrics (especially BLEU) tend to give higher scores on test instances where the images were very similar to training instances. Neural architectures performed better for more similar images overall.

The results obtained for the human captions (bottom rows of Table 5) suggest that the level of caption reuse by humans is extremely low compared to the models under consideration, though it stands at seven percent on MSCOCO.

Turning to the extent to which architectures use their training vocabulary, the picture that emerges is consistent with the above. While humans used between twenty-nine percent and forty-seven percent of the known vocabulary (taken from the training set) to describe the test set images, none of the evaluated systems used more than fourteen percent. The merge architecture tops the ranks for all datasets

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press

Table 5. *Results of the caption diversity metrics. The metrics were also applied to the first caption of each image in the human written test set captions. Best values are in bold.*

	Vocab. Used %	Unigram Ent.	Bigram Ent.	Existing Caps. %
Results for Flickr8K				
Merge	**13.52 (0.97)**	5.591 (0.091)	7.626 (0.159)	11.03 (0.54)
Pre-inject	12.46 (0.74)	**5.623 (0.064)**	**7.672 (0.066)**	**9.43 (0.31)**
Init-inject	12.01 (0.34)	5.617 (0.043)	7.649 (0.050)	11.13 (0.05)
Par-inject	10.67 (0.53)	5.507 (0.023)	7.421 (0.024)	13.30 (2.20)
Human	46.75 (0.00)	7.333 (0.000)	10.836 (0.000)	1.10 (0.00)
Results for Flickr30K				
Merge	**5.95 (0.13)**	5.368 (0.058)	7.215 (0.099)	**6.77 (0.45)**
Init-inject	5.70 (0.11)	5.509 (0.067)	7.407 (0.078)	8.70 (0.73)
Pre-inject	5.45 (0.36)	**5.511 (0.045)**	**7.438 (0.118)**	10.07 (2.34)
Par-inject	3.77 (0.19)	5.125 (0.028)	6.720 (0.076)	11.27 (1.10)
Human	29.40 (0.00)	8.011 (0.000)	11.786 (0.000)	0.00 (0.00)
Results for MSCOCO				
Merge	**7.91 (0.10)**	6.073 (0.023)	8.738 (0.025)	**40.93 (0.11)**
Init-inject	7.26 (0.05)	**6.128 (0.009)**	**8.768 (0.025)**	51.88 (0.20)
Pre-inject	6.59 (0.16)	6.064 (0.048)	8.657 (0.049)	51.92 (0.81)
Par-inject	5.00 (0.05)	5.863 (0.008)	8.192 (0.028)	62.73 (0.97)
Human	34.92 (0.00)	7.833 (0.000)	11.915 (0.000)	7.14 (0.00)

by a small margin, although unigram and bigram entropy is highest for pre-inject (Flickr8K and Flickr30K) and init-inject (MSCOCO).

We interpret these results as showing that neural caption generators require seeing a word in the training set very often in order to learn 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 token (the current experiments set the threshold at five), would be feasible and would make relatively little difference to the results.

5.5 Visual information retention

As noted in Section 1, one of the differences between the architectures under consideration is whether they incorporate the image features early or late. This raises the possibility of differences in the degree to which visual information is retained by each architecture in the multimodal vector, that is, the input to 'FF^{out}'

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press

in Figure 4. This is where information about visual and linguistic input is combined and is the information bottleneck that the output depends on. The question we want to answer is: Do (early-binding) inject architectures tend to 'forget' about the image as more words are input into the RNN? Given that the RNN's memory is finite, it should be difficult to retain information about all inputs as the length of the sequence increases, so information about the image might start fading away as the input sequence gets longer. Merge architectures do not have this problem with visual information as it is kept outside of the RNN and so is fully retained in the multimodal vector regardless of the number of time steps.

To measure how much visual information is retained as the number of time steps grows, we do the following:

(1) Take a trained neural network and input an image and a matching caption.
(2) Record the multimodal vector in the neural network at every time-step.
(3) Replace the image from the original neural network in step 1 with a randomly selected image, paired with the original caption, thus introducing an image-caption mismatch.
(4) Record the new, adulterated multimodal vector at every time-step for the new caption-image combination.
(5) Compare the original and adulterated vectors: if these converge as more words are fed to the model, it implies that the multimodal vector is losing image information, as it would be getting influenced less by the image and more by the prefix.

As a measure of distance between original and adulterated vectors, we use the mean absolute difference, that is, we take the absolute difference between each corresponding dimension in the two multimodal vectors and then take the mean of these differences. Mean absolute difference avoids giving a larger distance to larger vectors and is also intuitive as a measure of difference between vectors. It also keeps the distance between time steps exactly equal for merge, which is desirable since merge does not lose visual information across time steps.

For this set of experiments, we used all twenty-word captions in the MSCOCO test set and measured the mean distance over all twenty-one time steps (the twenty words plus the START token). Twenty-word captions are long enough to see a trend without ending up with too few captions (the mean caption length on the MSCOCO test set is about 10.4). To create a more reliable mean, we repeat this procedure 100 times so that the mean is over all images in the test set using 100 random images per instance. The results are shown in Figure 5.

None of the inject architectures maintained a consistent distance between the original and adulterated multimodal vectors. Crucially, the merge architecture also has the largest distance among all architectures, demonstrating that, in this architecture, the words in a caption exhibit a greater dependency on the image to which they pertain (hence, adulterating the multimodal vector with an irrelevant image alters the representation considerably). Par-inject comes in second place in terms of multimodal vector distance. This suggests that it retains more visual information than the other inject architectures, though not as much as merge. It seems that the

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press

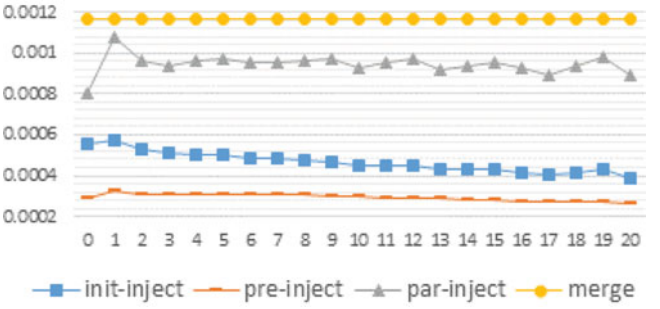


Fig. 5. (Colour online) Difference between multimodal vectors over time, when the caption is held constant but the image is changed. For a trained model, markers show mean absolute difference between multimodal vectors when a twenty-word caption is fed to the model with the original, versus a randomly chosen image. The marker at position 0 gives the multimodal vector difference after feeding in the START token whilst the marker at position twenty gives the multimodal vector difference after feeding in the last word in the caption.

amount of retention across time steps changes somewhat unpredictably, but tends to decrease overall, which means that information gets lost over time (though not to the extent of init-inject and pre-inject). Init-inject comes third in visual information retention followed by pre-inject, both of which decrease over time. It seems that, in a GRU trained for caption generation, the initial hidden state vector exerts more influence on the final hidden state vector than the first input.

These results predict that if the generated captions needed to be very long, late binding architectures will produce better captions as they will retain visual information over longer time steps, maintaining a tighter coupling between visual and linguistic information.

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 by training an RNN to encode an image-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 step. These models are therefore compatible with approaches to image caption generation using a 'multimodal' layer (Mao *et al.* 2014, 2015a, 2015b; Hendricks *et al.* 2016). While both types of architectures have been discussed 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 on standard corpus-based metrics such as CIDEr, the difference in performance between architectures is rather small. Init-inject tends to be better at generation and retrieval measures. Thus, from the perspective of corpus similarity, early binding of image features in models that view such features

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press

as 'modifiable' (in the sense outlined in the introduction) appear to be better than the alternatives.

Crucially, however, we also show that inject architectures are much more likely to re-generate captions wholesale from the training data and evince less vocabulary variation. Hence, from the perspective of variation, late-binding models that treat image features as fixed (i.e. not mixed with linguistic features) are better. While this is due in part to the nature of the available corpora, the superior performance of merge on this measure does suggest that, by encoding information from the two modalities separately, merge architectures might be producing less generic and stereotyped captions, exploiting their multimodal resources more effectively.

Our experiments on visual information retention show that, over time, inject architectures tend to loosen the coupling between visual and linguistic features, so that the difference between actual and adulterated multimodal vectors gets smaller. This too supports the view that inject models may, especially for longer captions, tend towards more generic and less image-specific captions, a finding that echoes the observations of Devlin *et al.* (2015), to some extent. In any case, late merging is, by definition, not susceptible to this problem.

From an engineering perspective, there is a significant difference between the required sizes of the RNN hidden state vectors. Whilst merge only requires a hidden state vector size sufficient to 'remember' caption prefixes, which depends on the length and complexity of the training set captions, inject architectures require additional memory to also store image information. This means that merge architectures make better use of their RNN memory. They also require less regularization whilst maintaining similar performance as other architectures.

The work presented here opens up some avenues for future research. In future work, we hope to investigate whether the results in this paper would remain similar when the experiments are repeated on other applications of conditioned neural language models such as neural machine translation or question answering.

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

Banerjee, S., and Lavie, A. 2005. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, vol.u 29, pp. 65–72.

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press

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. *JAIR* **55**: 409–42.

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

Chen, X., and Zitnick, C. L. 2015. Mind's eye: a recurrent visual representation for image caption generation. In *Proceedings of the CVPR'15*.

Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, 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 *Proceedings of the CVPR'09*.

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. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Volume 2: Short Papers*, Beijing, China, pp. 100–105.

Diederik, P. K., and Ba, J. 2014. Adam: a method for stochastic optimization. *CoRR*, 1412.6980.

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 *Proceedings of the CVPR'15*.

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. *Aistats* **9**: 249–56.

Harnad, S. 1990. The symbol grounding problem. *Physica D* **42**: 335–46.

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 *Proceedings of the CVPR'16*.

Hessel, J., Savva, N., and Wilber, M. J. 2015. Image representations and new domains in neural image captioning. *CoRR*, 1508.02091.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* **9**(8): 1735–80.

Hodosh, M., Young, P., and Hockenmaier, J. 2013. Framing image description as a ranking task: data, models and evaluation metrics. *JAIR* **47**(1): 853–99.

Karpathy, A., and Fei-Fei, L. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the CVPR'15*.

Kiros, R., Salakhutdinov, R., and Zemel, R. S. 2014a. Multimodal neural language models. In *Proceedings of the ICML'14*, pp. 595–603.

Kiros, R., Salakhutdinov, R., and Zemel, R. S. 2014b. Unifying visual-semantic embeddings with multimodal neural language models. *CoRR*, 1411.2539.

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 ACL'04*.

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 *Proceedings of the ECCV'14*, pp. 740–55.

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

Lu, J., Xiong, C., Parikh, D., and Socher, R. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 3242–3250.

Ma, S., and Han, Y. 2016. Describing images by feeding LSTM with structural words. In *Proceedings of the ICME'16*.

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press

Mao, J., Xu, W., Yang, Y., Wang, J., Huang, Z., and Yuille, A. 2015a. Deep captioning with multimodal recurrent neural networks (m-RNN). In *Proceedings of the ICLR'15*.

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 *Proceedings of the ICCV'15*.

Mao, J., Xu, W., Yang, Y., Wang, J., and Yuille, A. L. 2014. Explain images with multimodal recurrent neural networks. In *Proceedings of the NIPS'14*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. 2013. Efficient estimation of word representations in vector space. *CoRR*, 1301.3781.

Mnih, A., and Hinton, G. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the ICML'07*.

Nina, O., and Rodriguez, A. 2015. Simplified LSTM unit and search space probability exploration for image description. In *Proceedings of the ICICS'15*.

Oruganti, R. M., Sah, S., Pillai, S., and Ptucha, R. 2016. Image description through fusion based recurrent multi-modal learning. In *Proceedings of the ICIP'16*.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the ACL'02*, pp. 311–8.

Rennie, S. J., Marcheret, E., Mroueh, Y., Ross, J., and Goel, V. 2017. Self-critical sequence training for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, July 21-26, 2017*, Honolulu, HI, USA, pp. 1179–1195.

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*, 1409.1556.

Song, M., and Yoo, C. D. 2016. Multimodal representation: Kneser-Ney smoothing/skip-gram based neural language model. In *Proceedings of the ICIP'16*.

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

Vedantam, R., Zitnick, C. L., and Parikh, D. 2015. CIDEr: consensus-based image description evaluation. In *Proceedings of the CVPR'15*.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. 2015. Show and tell: A neural image caption generator. In *Proceedings of the CVPR'15*.

Wang, M., Song, L., Yang, X., and Luo, C. 2016. A parallel-fusion RNN-LSTM architecture for image caption generation. In *Proceedings of the ICIP'16*.

Wu, Q., Shen, C., van den Hengel, A., Liu, L., and Dick, A. R. 2015. Image captioning with an intermediate attributes layer. *CoRR*, 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 ICML'15*.

Yao, T., Pan, Y., Li, Y., Qiu, Z., and Mei, T. 2017. Boosting image captioning with attributes. In *IEEE International Conference on Computer Vision, ICCV 2017, October 22-29, 2017*, Venice, Italy, pp. 4904–4912.

You, Q., Jin, H., Wang, Z., Fang, C., and Luo, J. 2016. Image captioning with semantic attention. In *Proceedings of the CVPR'16*.

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. *TACL* **2**: 67–78.

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

https://doi.org/10.1017/S1351324918000098 Published online by Cambridge University Press