Picture It In Your Mind: Generating High Level Visual Representations From Textual Descriptions*

Fabio Carrara[1] Andrea Esuli[1] Tiziano Fagni[2]
Fabrizio Falchi[1] Alejandro Moreo Fernández[1]

[1]ISTI-CNR, via G. Moruzzi, 1, 56124 Pisa, Italy
[2]IIT-CNR, via G. Moruzzi, 1, 56124 Pisa, Italy

Abstract

In this paper we tackle the problem of image search when the query is a short textual description of the image the user is looking for. We choose to implement the actual search process as a similarity search in a visual feature space, by learning to translate a textual query into a visual representation. Searching in the visual feature space has the advantage that any update to the translation model does not require to reprocess the (typically huge) image collection on which the search is performed. We propose various neural network models of increasing complexity that learn to generate, from a short descriptive text, a high level visual representation in a visual feature space such as the pool5 layer of the ResNet-152 or the fc6-fc7 layers of an AlexNet trained on ILSVRC12 and Places databases. The Text2Vis models we explore include (i) a relatively simple regressor network relying on a bag-of-words representation for the textual descriptors, (ii) a deep recurrent network that is sensible to word order, and (iii) a wide and deep model that combines a stacked LSTM deep network with a wide regressor network. We compare the models we propose with other search strategies, also including textual search methods that exploit state-of-the-art caption generation models to index the image collection.

image retrieval; cross-media retrieval; text representation

1 Introduction

Using a textual query to retrieve images is a very common cross-media search task, as text is the most efficient media to describe the kind of image the user is

*This paper is a revised and largely extended version of a preliminary work presented at the Neu-IR '16 SIGIR Workshop on Neural Information Retrieval, uploaded to arXiv: `https://arxiv.org/abs/1606.07287v1`

searching for. Each media has its own representation space, which is modeled on a collection of representative content for that media. For example, text can be represented by means of a simple bag-of-words feature space, with the feature space being defined by a dictionary of observed words; or by means of more complex distributional semantic models, such as those based on neural networks, e.g., Word2Vec [33]. Similarly, a visual space can be modeled by identifying a set of relevant visual features in a collection on images, e.g., as those extracted by the deeper layers of Convolutional Neural Networks (CNN) [27].

In cross-media retrieval, the actual retrieval process can be implemented in a number of ways, depending on how the two feature spaces are joined. The cross-media search space can be a textual feature space, i.e., a space whose definition is determined exclusively by observing textual content; a visual feature space, i.e., a space whose definition is determined exclusively by observing visual content; or a common latent space in which textual and visual features are projected into.

Using textual features is the most common solution. Each image is associated with a set of textual features extracted from its context of use (e.g., the text surrounding the image in the Web page, description fields in metadata), and eventually enriched by means of classifiers that assign textual labels related to the presence of certain relevant entities or abstract properties in the image. The textual search space model can exploit the actual visual content of the image only when classifiers for the concepts of interest are available, thus requiring a relevant number of classifiers; this also requires to reprocess the entire image collection whenever a new classifier is made available.

Searching in a common latent space requires learning two projections (i.e., from text-to-latent and from image-to-latent). The main advantage of searching in a common latent space lies on the freedom the system has to jointly model reciprocal relations between the two media, while other strategies can only learn the relations from the source media to the target media, but not vice versa. However, as in the textual space, projecting into a common latent space also requires to reprocess all the images whenever the textual model is updated, since the latent space where images are projected into is also influenced by the textual model part. It also requires managing and storing the additional latent representations that are used only for the cross-media search.

A last, less explored, possibility is to use a visual space to convert any textual query into a visual representation. A key advantage of this model is that the representation of images remains unaltered regardless of the projection model being developed. This means that any improvement in the projection model, e.g., in the underlying language model, has immediate effects on the image retrieval process, without requiring to reprocess the (typically huge) whole image collection, and to rebuild the similarity search data structures required for efficient retrieval. Another advantage is that, since the visual space is language-independent, multiple models, e.g., for multiple languages or specialized on different domains, can be used independently on the same collection of images, without requiring multiple instances of representations for the images

and multiple instances of similarity search data structures.

In this paper we explore the use of a visual space for cross-media retrieval. Methods that use a common space projection may be able to produce better results because they can exploit cross-correlations between the two media, while the other two approaches are constrained to leverage on correlations that come from one single direction. However, we deem that the ability of using a single static collection of visual representations for images, irrespectively to how many text-to-visual projection models are used and how often they change, is a practical advantage of visual space-based methods that counters such possible loss of quality in results.

We present Text2Vis, a family of neural network models that convert textual descriptions into visual representations in the same space of those extracted from deep CNN such as the AlexNet [27] or ResNet-152 [17] trained on ILSVRC12 [38] and Places [47] datasets. We first offer an overview of relevant cross-media retrieval in section 2. We propose different neural network models of increasing complexity in section 3, including (i) S-Text2Vis, a simple regressor network relying on sparse representations (bag-of-words and bag-of-bigrams) for the textual descriptors; (ii) D-Text2Vis, a deep recurrent network relying on a continuous dense representations (word embeddings); and (iii) W&D-Text2Vis, a wide and deep architecture relying on both sparse and dense representations. We report experimental results in section 4, comparing with other methods that use different projection approaches. Section 5 concludes and outlines possible directions for future research.

2 Related Work

Deep Learning and Deep Convolutional Neural Networks (DCNNs) in particular, have recently shown impressive performance on a number of multimedia information retrieval tasks [27, 41, 17]. Deep Learning methods learn representations of data with multiple levels of abstraction. As a result, the activation of the deeper hidden layers has been used in the context of transfer learning and content-based image retrieval [9, 37] as high-level representations of the visual content. Somewhat similarly, distributional semantic models, such as those produced by Word2Vec [33], or GloVe [36], have been found useful in modeling semantic similarities among words by establishing a connection between *word meaning* and *position* in a vector space.

In order to perform cross-media retrieval, the two feature spaces (text and images in our case) should be made comparable, typically by learning how to properly map the different media. This problem has been attempted in different manners so far, which could be roughly grouped into three main variants, depending on whether the mapping is performed into a common latent space (Section 2.1), a textual space (Section 2.2), or a visual space (Section 2.3).

2.1 Mapping Into a Common Space

The idea of comparing texts and images in a common latent space has been investigated by means of Cross-modal Factor Analysis and (Kernel) Canonical Correlation Analysis in [7, 15]. In a similar vein, Corr-AE was proposed for cross-modal retrieval, allowing the search to be performed in both directions, i.e., from text-to-image and vice versa [12]. The idea is to train two autoencoders, one for the image domain and another for the textual domain, imposing restrictions between the two. Similarly, in [24] the authors propose an encoder-decoder architecture, in which the encoder part, formed by a LSTM (for textual input) and a CNN (for visual input), is trained to project both inputs into near points in a common multimodal space, and the decoder part generates new text from a point in this new space. As will be seen, one of the architectures we are presenting in the following (S-Text2Vis, Section 3.2) bears resemblance to one of the architectures investigated in [12], the so-called *Correspondence full-modal autoencoder* (which is inspired by the multimodal deep learning method [34]). However, the two networks have a fundamental difference, since the Correspondence full-modal autoencoder takes examples from both media as the inputs. The DeViSE [13] method jointly trains a pre-trained instance of the convolutional neural network of [27] (with its last layer replaced with a linear mapping into the final embedding space), and a textual embedding space pre-trained as a skip-gram model [33]. Even though DeViSE uses a final space which is of the same size of the textual space, the pre-trained word embeddings are only used as initial parameters and then they are adapted jointly with visual embeddings during the training. The training is made on image and label pairs, where the labels are not a full description of the scene, indicating only the presence of certain entities in the image.

2.2 Mapping Into the Textual Space

The BoWDNN method [1] trains a deep neural network to map images directly into a bag-of-words (BoW) space, where the cosine similarity between BoWs representations is used to generate the ranking. Somehow similarly, a dedicated area of related research is focused on generating captions describing the salient information of an image (see, e.g., [22, 11, 44]). The m-RNN method [31] trains a multimodal recurrent neural network to generate a caption description for a given image. The model consists of a recurrent sub-network (operating on text data) and a convolutional sub-network (operating on image data) which combine into a multimodal layer where the recurrent state interacts with the image representation. In [30], authors propose m-CNN, a multimodal architecture in which convolutions are used on both the image and textual inputs to directly output a match score between them. Models like m-CNN, which do not explicitly learn a projection but a distance function on a latent projection, are not fit for retrieval on large collections. Given a query, such models need to perform a forward pass through the network for every image in the collection in order to compute the distances. This entails a much higher cost with respect to

traditional metrics, such as the Euclidean distance or the cosine similarity. The ConSE [35] method adopts a very simple approach, inspired by DeViSE, that uses the classification labels of the convolutional neural network of [27] to select and combine, by their classification probability, the set of textual embeddings related to the top assigned labels.

2.3 Mapping Into the Visual Space

Our Text2Vis variants belong to this group where, to the best of our knowledge, the only other proposal up to now is a method dubbed *Word2VisualVec* [10], which was reported just very recently. There are some fundamental points where their method and ours differ, though. *Word2VisualVec* takes combinations of Word2Vec-like vectors as a starting point, thus reducing the dimensionality of the input space, whereas we directly take the bag-of-words vector encoding of the textual space as the input (S-Text2Vis), or learn the word embeddings (D-Text2Vis, Section 3.3) during the training process, as we did not observe any improvement in pre-training the textual part. Moreover, *Word2VisualVec* builds a deep regressor on top of the textual representation that are aggregations of word embeddings, which thus discard word order information. Contrarily, we observed that, when disregarding word order, yet a shallow regressor (S-Text2Vis) produces effective mappings of textual vectors into the visual space. We also observed that taking word order into account helps to improve results (D-Text2Vis and W&D-Text2Vis, Section 3.4).

3 Generating Visual Representations of Text

Our goal is to map textual descriptions to high-level visual representations. As the visual space we used the pool5 layer of the ResNet-152 [17] trained on ILSVRC12[1], and the fc6 and fc7 layers of the Hybrid network [47] (i.e., an AlexNet [27] trained on both ILSVRC12 and Places[2] datasets). Principal Component Analysis (PCA) and whitening are commonly used in retrieval processes based on vector similarity to reduce the dimensionality and to improve the retrieval effectiveness of visual features. Projecting the dataset onto the eigenvectors results in no correlation between the components, while whitening normalizes the vectors to have unit variance for all components. This is done by simply dividing each component by the square root of its eigenvalue. Originally proposed for local features aggregations such as VLAD [21], PCA and whitening are also largely used for processing the activation of neurons [40, 14, 16]. As reported in Section 4.6, we observed relevant improvement by applying PCA and whitening to the visual features.

In this section we describe the experimental activities we have carried out in order to achieve our goal. We take a simple feedforward regressor as a starting point (Section 3.1) to then propose three different architectures of increas-

[1] `http://image-net.org/challenges/LSVRC/2012`
[2] `http://places.csail.mit.edu/index.html`

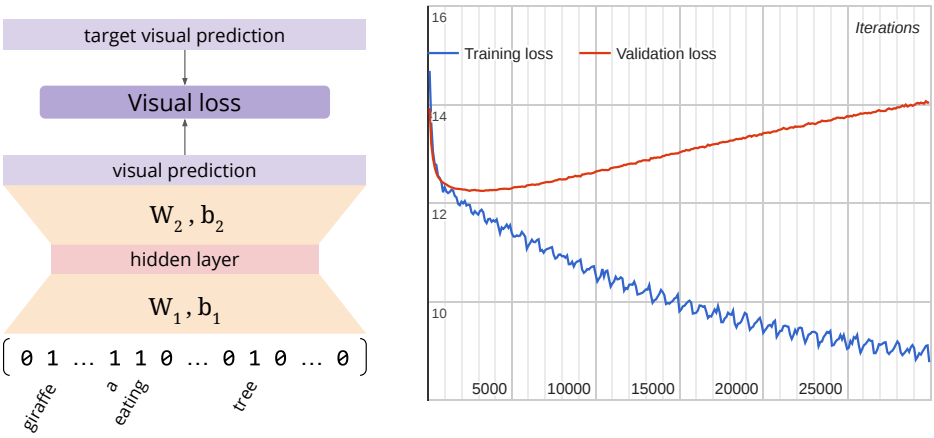

Figure 1: Left: Architecture of a simple regressor model with one hidden layer of size 1024. Right: The training and validation loss (on y-axis) in function of the training iteration (on x-axis). Notice the model overfits in the early phase of the training process.

ing complexity: a regressor learning from unordered sparse features, called S-Text2Vis (Section 3.2); a deep recurrent network learning from ordered dense features, called D-Text2Vis (Section 3.3); and a wide & deep neural network which jointly learns from both types of representations, called W&D-Text2Vis (Section 3.4).

3.1 VisReg

As a reference baseline we started with a simple feedforward regressor model with a hidden layer trained on the sparse one-hot representation of the textual input to directly predict the visual representation of the image (Figure 1, left). We observed a strong tendency to overfit (Figure 1, right), thus degrading the applicability of the method to unseen images.

We explain this overfitting with the fact that a visual representation keeps track of every element that appears in the image, regardless of their semantic relevance within the image, while a (short) textual description is more likely focused on the visually relevant information, disregarding the secondary content of the image. For example, the relevant images for the query "a person doing jogging" will likely share a subset of common features that denote the presence of a person with a posture that is associated to the action of gentle running, and then have many other features related the different compositions of colors, perspective, background elements each image may contain. As the learning iterations proceed, the simple regressor model starts capturing these secondary elements of the images that are not relevant for the main represented concept, but are somewhat characteristic to the specific set of images that compose the

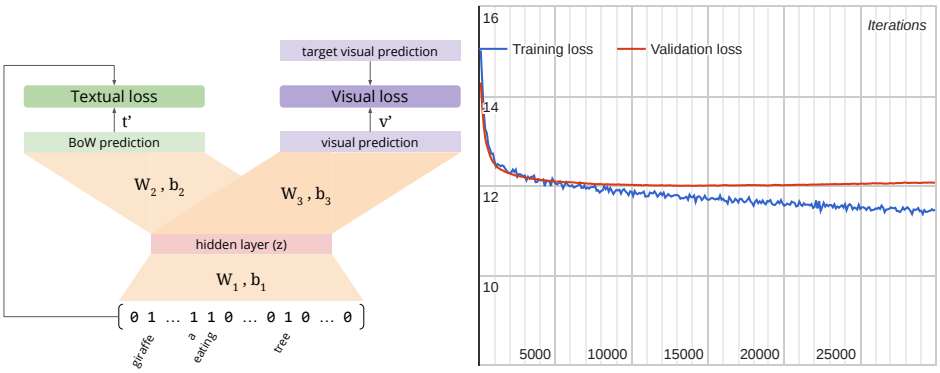

Figure 2: Left: Architecture of our proposed S-Text2Vis which controls overfitting by adding an autoencoding constraint on the hidden state. Right: The training and validation loss (on y-axis) in function of the training iteration (on x-axis).

training data.

This preliminary experiment suggests that text-to-image mapping must be somehow regularized. In the following we propose various strategies aiming at constraining the mapping to better model the textual part.

3.2 S-Text2Vis

The first model we propose, dubbed S-Text2Vis, is based on forcing the hidden representation to be representative not only for the visual reconstruction, but also for reconstructing the sparse textual signal.

S-Text2Vis thus contrasts the overfitting by adding a text-to-text autoencoding branch to the hidden layer (Figure 2, left), constraining the model to jointly satisfy two different losses: one visual (text-to-visual regression) and one linguistic (text-to-text autoencoder). The linguistic loss works at higher level of abstraction than the visual one, acting as an additional constraint on the model, and preventing (as confirmed by our experiments) overfitting on the visual loss (Figure 2, right).

S-Text2Vis consists of two overlapped feedforward neural nets with a shared hidden layer. The feedforward computation is described by the following equations:

$$z = ReLU(W_1 t_{in} + b_1) \tag{1}$$

$$t' = ReLU(W_2 z + b_2) \tag{2}$$

$$v' = ReLU(W_3 z + b_3) \tag{3}$$

where t_{in} represents the sparse one-hot encoding for the textual descriptor given as input to the net, z is the hidden representation, v' and t' are the visual and

textual predictions, respectively, obtained from the hidden representation z, $\Theta = \{W_i, b_i\}_{i \in \{1,2,3\}}$ are the model parameters to be learned, and $ReLU$ is the activation function, defined by $ReLU(x) = \max\{0, x\}$.

Both predictions v' and t' are then compared with the expected outputs, i.e., the visual embedding representation v, and a textual descriptor t_{out} that is either t_{in} or semantically equivalent to t_{in} (we expand on this below). We used the *mean squared error* (MSE – Equation 4) as the loss function both for the visual loss and the textual loss, denoted by $\mathcal{L}_v$ and $\mathcal{L}_t$, respectively.

$$MSE(y, y') = \frac{1}{n} \sum_{i=1}^{n} (y_i - y'_i)^2 \tag{4}$$

where y, y' are a pair of target description and prediction either in the textual (t, t', left part of the network in Figure 2) or in the visual (v, v', right part of the network in Figure 2) space. The model is thus multi-objective, and many alternative strategies could be followed at this point in order to set the Θ parameters so that both criteria are jointly minimized. A simple strategy to jointly optimize the two losses consists of defining a single loss as a parametrized aggregation (Equation 5), with typically one single parameter controlling the relative contribution of the losses [12]. We also add a regularization parameter to further counter overfitting.

$$\widehat{\Theta} \quad = \quad argmin_{\Theta} \left(\mathcal{L}_t(t_{out}, t') + \alpha \mathcal{L}_v(v, v') + \lambda ||\Theta||_2 \right) \tag{5}$$

Note that the net is fed with a triplet $\langle v, t_{in}, t_{out} \rangle$ at each iteration. When $t_{out} = t_{in}$ the text-to-text branch is an *autoencoder*. It is also possible to have $t_{in} \neq t_{out}$, with the two pieces of text being semantically equivalent (e.g., t_{in} =“`a woman cutting a pizza with a knife`”, t_{out} =“`a woman holds a knife to cut pizza`”). The text-to-image branch is, in any case, a regressor. Notwithstanding, since our final goal is to project the textual descriptor into the visual space, the text-to-text branch might be though as an additional constraint (of linguistic nature) to the visual reconstruction (and, more specifically, to its internal encoding).

The main strength of S-Text2Vis regards its simplicity, specially in the use of the most simple representation for the input (the sparse encoding); yet it produces effective results (as discussed bellow). That being said, the model presents some flaws too, i.e., (i) the sparse encoding results in a high dimensionality, thus constraining the net to optimize a large number of parameters, and (ii) the model is agnostic to word order, thus losing relevant information from text, e.g.: “`a white cat and a black dog`” vs “`a black cat and a white dog`”.

3.3 D-Text2Vis

The second model we propose, dubbed D-Text2Vis, is meant to overcome the limitations of S-Text2Vis.

In order to reduce the amount of parameters of the net, we resort to dense representations (i.e., word embeddings) for the terms in the description. Besides the mere reduction in the number of dimensions, the main reason that motivates operating in a dense embedding space concerns with the gain in generalization. Words with similar meanings end up being represented by similar vectors (in the sense of the inner product), which allows the model to better generalize, i.e., the patterns discovered become descriptive for an embedding region (and to the greater or lesser extent to words with nearby embeddings) rather than descriptive for a single word.

In order to make the model become sensible to word order, we adopt an LSTM [18] architecture, a special kind of recurrent neural network which is particularly robust to learn from sequential data (such as textual data). Concretely, we train an LSTM on the task of language modeling (that is, the task of predicting the most likely following term given the sequence of preceding terms – see e.g., [42]) with backpropagation through time [46]. We constrain the internal memory state of the last memory cell to be a good representation to predict the visual embedding (Figure 3).

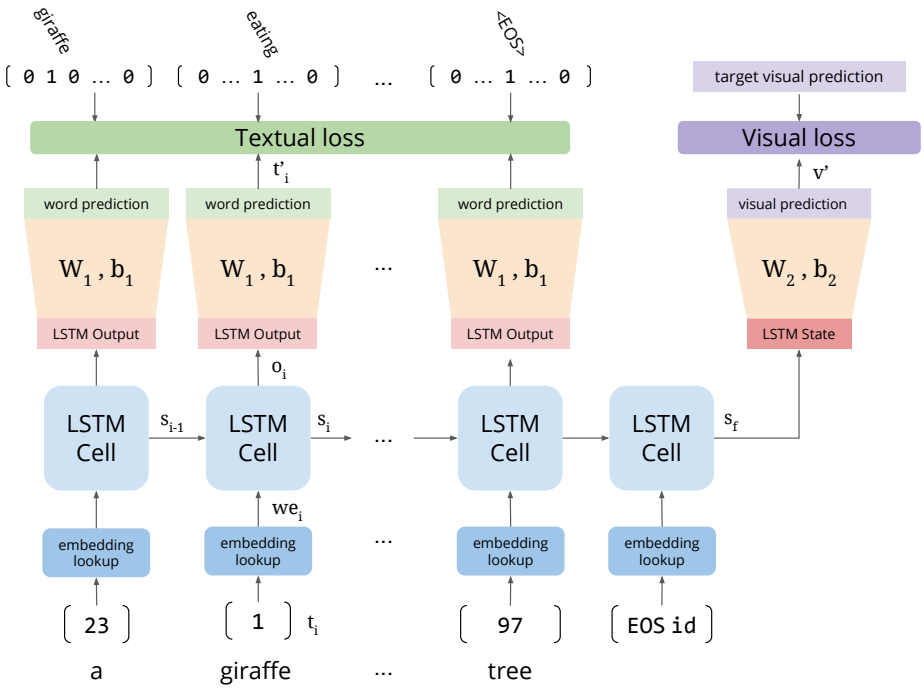

Figure 3: Architecture of D-Text2Vis.

The computation is described by the following equations:

$$we_{t_i} = lookup(WE, t_i) \tag{6}$$

$$o_i, s_i = LSTMcell(we_{t_i}, s_{i-1}) \tag{7}$$

$$t'_i = softmax(W_1 o_i + b_1) \tag{8}$$

$$v' = ReLU(W_2 s_f + b_2) \tag{9}$$

where *lookup*() returns the word-embedding we_{t_i} from the (trainable) matrix WE for the i*th* word in the textual descriptor with index t_i, $LSTMcell$ is the memory cell, o_i and s_i represent the *output* and *state* signals produced after processing we_{t_i} and s_{i-1} (the state signal produced in the precedent step), and s_f is the state of the last memory cell. The softmax function transforms the output signal into a probability distribution on the vocabulary-length space. Finally, v' and t'_i are the visual vector and term predictions, respectively.

Note that in addition to the parameters WE, $W_{\{1,2\}}$ and $b_{\{1,2\}}$, the LSTM-cell internally maintains an *input*, *output*, and *forget* gates with their own parameters; as the memory cell we used the implementation described in [39].

The sequence of term vectors predictions t'_i and the visual prediction v' are then compared to the expected textual and visual outputs. For the visual loss $\mathcal{L}_v$ we use the MSE (Equation 4), as before. Each predicted term t'_i is a $|V|$-dimensional vector that could be though as a probability distribution over the term indexes, where V is the vocabulary. Analogously, each term w can be codified as a *one-hot* vector, i.e., a $|V|$-dimensional vector with all zero values except the dedicated dimension indexing t_i, which is set to one. Note that a one-hot encoding could be interpreted as a probability distribution as well. (When not confusing, we use t_i both to refer to the term symbol and to its one-hot encoding.) The error between both distributions is compared via the cross-entropy error (Equation 10). Given the sequence t of expected terms t_i and the sequence t' of predicted signals t'_i outputted by the net, the textual loss is computed as the averaged cross-entropy (Equation 11).

$$CrossEntropy(y, y') = -\sum_{i=1}^{n} y'_i \log(y_i) \tag{10}$$

$$\mathcal{L}_t(t, t') = \frac{1}{n} \sum_{i=1}^{n} CrossEntropy(t_i, t'_i) \tag{11}$$

where y, y' represent any pair of true and predicted distributions. As before, the net is fed with a triple $\langle v, t_{in}, t_{out} \rangle$ where, given a caption $[t_0 \dots t_L]$, the input ant output textual sequences are defined as $t_{in} = [t_0 \dots t_{L-1}]$ and $t_{out} = [t_1 \dots t_L]$, being $t_L = EOS$ a special symbol delimiting end of the sequence. That is, the expected sequence corresponds to the input sequence shifted one position since the LSTM part is trained to predict the next term in the sequence. As the model is, again, multi-objective, we apply the weighted aggregation described by equation 5 to set the optimization problem.

3.4 W&D-Text2Vis

Our last proposal, dubbed W&D-Text2Vis, combines the sparse and dense representations by following the recently proposed *Wide & Deep Learning* strat-

egy [4].

W&D-Text2Vis combines the deep LSTM (borrowed from D-Text2Vis) with a wide regressor. Linear models with nonlinear feature transformations are known to be useful for large-scale regression problems with sparse inputs (as is the case for short text descriptions). This model emerged from the belief, discussed in [4], that the deep part contributes to model *generalization* while the wide part contributes to model *memory* and therefore their combination might be beneficial.

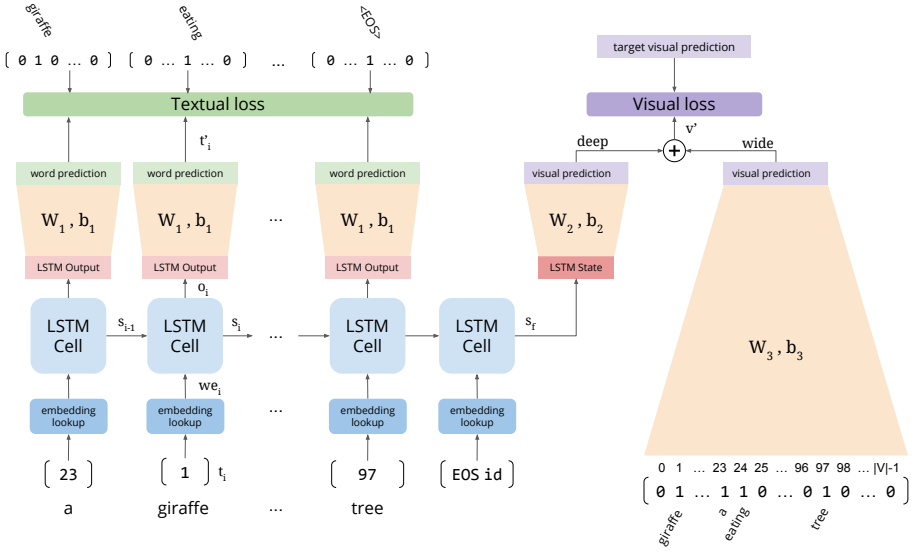

Figure 4: Architecture of W&D-Text2Vis.

The fundamental difference with respect to [4] is that we use a recurrent neural network as the deep part (instead of a feedforward network) since LSTMs are particularly fit to learn from sequential data such as our textual descriptions.

The computations reuse Equations 6–8 from D-Text2Vis and incorporate the following set of equations for the wide part:

$$deep = W_2 s_f + b_2 \tag{12}$$

$$wide = W_3 \left(\sum_{i=0}^{L} onehot(t_i) \right) + b_3 \tag{13}$$

$$v' = ReLU(wide + deep) \tag{14}$$

where $onehot(t_i)$ returns the one-hot encoding vector for term t_i. As in D-Text2Vis, we used the MSE (Equation 4) for the visual loss $\mathcal{L}_v$ and the averaged cross-entropy (Equation 11) for the textual loss $\mathcal{L}_t$. The optimization problem is set as in equation 5.

4 Experiments

In this section we describe the set of experiments we have carried out in order to test our methods.

4.1 Datasets

We used the *Microsoft COCO* dataset (MsCOCO[3] [29]). MsCOCO was originally proposed for image recognition, segmentation, and caption generation. Although other datasets for image retrieval exist (e.g., the one proposed in [19]), they are more oriented to keyword-based queries. We believe MsCOCO to be more fit to the scenario we want to explore, since the captions associated to the images are expressed in natural language, thus semantically richer than a short list of keywords composing a query.

MsCOCO contains 82.783 training images (*Train2014*), 40.504 validation images (*Val2014*), and about 40K and 80K test images corresponding to two different competitions [3] (*Test2014* and *Test2015*). Because MsCOCO was proposed for caption generation, the captions are only accessible in the *Train2014* and *Val2014* sets, while they are not yet released for *Test2014* and *Test2015*. We have thus taken the *Train2014* set for training, and randomly split the *Val2014* into two disjoint sets of 20K images each for validation and test.

Each image in MsCOCO has five different captions associated[4], each of which written by a different individual. Let $\langle I, C\rangle$ be any labeled instance in MsCOCO, where I is an image and $C = \{c_1..c_5\}$ is a set of captions describing the content of I. Given a $\langle I, C\rangle$ pair, we define a training labeled instance in our model as $\langle v, t_{in}\rangle$ where $v \in \mathbb{R}^{2048}$ is the visual representation of the image I taken from the pool5 layer, or $v \in \mathbb{R}^{4096}$ when the representation comes from the fc6 or fc7 layer (each representation has been tested in distinct experiments), and t_{in} is a textual descriptor randomly chosen from C representing the input descriptor for the model. In the exceptional case of S-Text2Vis a training label instance is defined as $\langle v, t_{in}, t_{out}\rangle$, where t_{out} is the output textual descriptor randomly chosen from C (the meanings for v and t_{in} remain untouched). Note that, in this case, t_{in} and t_{out} are not imposed to be different, thus leading to a total of 25 possible combinations of training instances one could extract from a single pair $\langle I, C\rangle$; this increases the variability of the training set a lot along the different epochs. The training triplet $\langle v, t_{in}, t_{out}\rangle$ for D-Text2Vis and W&D-Text2Vis models are extracted from the instance $\langle I, C\rangle$ by randomly choosing t_{in} from C and then defining t_{out} as t_{in} shifted one position (as explained above, see section 3.3).

[3] Publicly available at `http://mscoco.org/`

[4] Actually in the dataset there are few images with more than five captions available for processing. In such cases we took the first five listed.

4.2 Visual similarity search

We evaluated the visual similarity between any two images by comparing their visual descriptions obtained as described in Section 3. In particular, given the improvement in performance in Content-Based Image Retrieval task reported in [40, 14, 16], the Euclidean distance is used to compare the vectors obtained applying PCA and whitening [6] to the neurons activation. The resulting vectors have components which are both not correlated and have unit variance. In our experiments, we considered the first 256 components obtained after PCA (while the original dimension was 2,048 in the pool5 layer, and 4,096 in fc6-fc7 layers).

4.3 Training

We tackle the optimization problems using the Adam optimizer [23] with default parameters (learning rate 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e^{-0.8}$) in all cases.

We set the size of the training batch to 64 examples; each of which was extracted from a different image. Each training example in the batch corresponds to the visual features v of a different image I, and a textual descriptor t_{in} picked at random from the set C of captions associated to I in MsCOCO. As explained above, S-Text2Vis requires an additional t_{out} which is also picked at random from C during training. (During test, we consider all captions as different queries.) We set the maximum number of iterations to 300.000 in S-Text2Vis, and to 50.000 in D-Text2Vis and W&D-Text2Vis, but apply an early stop when the model starts overfitting (as reflected in the validation error). The training set is shuffled each time a complete pass over all images is performed.

The word embedding matrix for D-Text2Vis and W&D-Text2Vis has been initialized at random according to an uniform distribution ranging from -1.0 to 1.0. In preliminary experiments, we investigated on the use of pre-trained word embeddings, i.e., representing the textual description as the average of the embeddings of the words composing the description (see Equation 1 in [10]), but we have not observed any improvement. Pre-training the word embeddings is an additional cost, and the fitness of the embeddings for the task depends on the type of documents they are learned from. For example, an 11% improvement in MAP is reported in [2] from learning embeddings from Flickr tags compared to learning them from Wikipedia pages.

The rest of the Θ parameters for all models (with the sole exception of the word embedding matrix) have been initialized at random according to a truncated normal distribution centered in zero with standard deviation of $\frac{1}{\sqrt{n}}$, where n is the number of columns. The biases have all been initialized to 0.

Following previous approaches to multimodal learning [12, 34], we adopted an aggregated loss which depends on one single parameter α (Equation 5). In [12] it was found that unbalancing aggressively the loss pressure towards one or the other extremes tends to degrade the performance. For the α hyperparameter we have tried the values $\{0.01, 0.1, 1.0, 10.0, 100.0\}$, choosing the best

one for each visual embedding layer as reflected in the validation error. For the parameter λ determining the impact of the L_2 regularization we tried the values $10^i, i \in \{1, -2, -4, -6\}$.

For S-Text2Vis we tested two different vectorial representations of text: S-Text2Vis-U uses a simple bag-of-words vectors that marks with a value of one the positions that are relative to words that appear in the textual description and leave to zero all the others; S-Text2Vis-N adds a little bit of information on the text structure by considering also N-grams for a selection of part-of-speech patterns[5]. The resulting vocabulary size is 10,358 for S-Text2Vis-U after removing terms appearing in less than 5 captions. For S-Text2Vis-N we considered the 23,968 uni-grams and N-grams appearing at least in 10 captions. We set the number of nodes of the hidden layer to 1024 which was experimentally confirmed as the best value among the candidates $\{256, 512, 1024, 2048\}$; we omit those experiments for the sake of conciseness.

In order to efficiently train the LSTM part in D-Text2Vis and W&D-Text2Vis we make use of *padding* and *bucketing*. That is, to avoid constructing as many graphs as different caption lengths there are in the dataset, we fix a number of buckets (i.e., sequences of fixed length – we considered $\{15, 20, 40\}$ in our experiments) and apply padding to the captions (i.e., repeatedly adding the 'PAD' token at the beginning of the tokens sequence, and the special 'EOS' token announcing the end of the sequence) to fit in the corresponding bucket (the smallest one that could allocate the caption).

For D-Text2Vis and W&D-Text2Vis we have also considered stacking LSTM cells as a mean to give the model a greater expressive power. We denote those variations by the suffix '-$\langle n \rangle$' where n indicates the height of the stack. E.g., D-Text2Vis-1 corresponds to the vanilla model in Figure 3, while W&D-Text2Vis-4 is the wide & deep approach with 4 LSTM cells stacked. In all cases, we set the dimensionality of the embedding space to 100 and the size of the internal LSTM nodes to 512; again, those values were chosen during preliminary experiments run on the validation set.

A Tensorflow implementation of all our methods, and of all the compared methods described in the next section, is available at `https://github.com/AlexMoreo/tensorflow-Tex2Vis`.

4.4 Compared methods

We compare the performance of the various Text2Vis models against a selection of methods that perform search either in the visual space or in the textual space. We define as the trivial lower bound baseline the method that produces a random ranking of the images in the collection (dubbed *RRank*).

We define as *VisSim* the direct similarity method that computes the Euclidean distances using the original pool5 (from the ResNet-152 [17]), and fc6 or fc7 features (from the AlexNet [47]) for the image that is associated to the

[5]We considered the part-of-speech patterns: 'NOUN-VERB', 'NOUN-VERB-VERB', 'ADJ-NOUN', 'VERB-PRT', 'VERB-VERB', 'NUM-NOUN', and 'NOUN-NOUN'.

query caption in MsCOCO. *VisSim* models the scenario in which the user submits the query using an image that is representative of the original textual description. *VisSim* is thus not a real cross-media search model, but it allows us to measure how a search-by-representative-image process compares with the real cross-media search approaches. We also compare with VisReg, the text-to-image sparse regressor described in section 3.1.

We use the caption generation methods presented in [22] (dubbed *NeuralTalk*) and [44] (dubbed *Show&Tell*) to implement cross-media search methods based on textual search. Given a caption generation method, we generate captions for all the 20K images in the test collection, and then we implement the search process as a text similarity search process based on two retrieval models: one that used the same $ROUGE_L$ metric that is used for the evaluation (dubbed *CapRouge*), and one that uses a more classic ranking by L_2 norm of the vectors resulting from text indexing based on bag-of-words or characters 3- and 4-grams (respectively dubbed *CapBow* and *CapGrams*). We used two *Show&Tell* models, one trained for one million iterations (*Show&Tell-1M*), and another for two million iterations (*Show&Tell-2M*). It is important to stress that the *CapRouge* method is to be considered as a very strong but unrealistic baseline, added for the sake of a richer comparison, and not a viable retrieval method, for two reasons. First, it uses the same metric of the evaluation, so it improperly overfits on it. Second, it has a computational cost that is quadratic with the length of the compared string, making it not practically usable. For example, computing the ranking of 20K captions from the test set against a query caption required on average for all the queries, on the same hardware and using efficient implementation, 0.046 seconds for *CapBoW* and *CapGram* methods and 3.267 seconds for *CapRouge*, resulting two orders of magnitude slower than the other methods.

We also compare our Text2Vis variants against Word2VisualVec [10], a method that maps the text input into the visual space (as described in section 2.3). We have reimplemented Word2VisualVec by following [10]. We have pre-trained a 500-dimensional word embeddings space on the user tags associated to 100M images in the YFCC100M dataset [43] using the skip-gram model in Word2Vec [33]. We have experimented with two variants: *short* (Word2VisualVec-S), which trains a feedforward network with two hidden layers of $[1000, 2000]$; and *long* (Word2VisualVec-L), which considers three hidden layers of $[1000, 2000, 3000]$. We have only experimented with the variant that adopts the same MSE loss function as our model[6]. In order to carry out a fair comparison, we have implemented the exact same retrieval for Word2VisualVec as our method (i.e., euclidean distance after PCA and whitening – see section 3) instead of the originally proposed *cosine similarity*. The reason for doing so is that we have observed a consistent improvement of about 3% in our experiments, and verified similar improvements to be achieved in the case of

[6]They reported slightly better results with the Marginal Ranking Loss (MRL), a cost function that takes two visual vectors for each example, one considered relevant, and another irrelevant, to the textual description. However, relevance judgments to generate the training triplets relied on the user-click logs available in their dataset.

WORD2VISUALVEC as well; concretely, WORD2VISUALVEC improved its DCG average by 0.073 (with a standard deviation of ±4.8E-03) due to the use of this retrieval method in place of the cosine similarity.

4.5 Evaluation Measures

We measure the retrieval effectiveness of the various methods we compare by means of the *Discounted Cumulative Gain* (DCG [20]), defined as:

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{15}$$

where rel_i quantifies the *relevance* of the retrieved element at rank position i with respect to the query, and p is the rank at which the metric is computed; we set $p = 25$ in our experiments, as was done in related research [19, 10].

Given that some of the compared methods (e.g., text-based search) can produce rankings with ties, we actually use the *Ties-aware Discounted Cumulative Gain* (TDCG [32]), defined as:

$$TDCG_p = \sum_{i=1}^{m} \left(\left(\frac{1}{n_i} \sum_{j=t_i+1}^{t_{i+1}} 2^{rel_i} - 1 \right) \sum_{j=t_i+1}^{\min(t_{i+1},k)} \frac{1}{\log_2(i+1)} \right) \tag{16}$$

where m is the number of group of ties in the ranking of the first p results, n_i indicates the number of tied result in the i-th group, t_i indicates the starting position of each tied group. $TDCG_p$ is derived from DCG_p by observing that the average gain for a position in a group of tied results is the average of the gain of such tied results. $TDCG_p$ is obviously equivalent to DCG_p in the case there are no ties in the results.

Because the *rel* values are not provided in the MsCOCO, we estimate them by using the $ROUGE_L$ [28] metric, a measure often used for the evaluation of the results of text summarization algorithms and one of the evaluation measures for the MsCOCO caption generation competition[7] [3]. This is a metric based on finding the Longest Common Subsequence (LCS) between the two strings being compared[8] and then measuring a weighted harmonic mean (F_β, with $\beta = 1.2$) of the coverage ratios of the subsequence with the two strings. Using a β value greater than one ($\beta = 1.2$ is the default value in the MsCOCO evaluation software) gives a little more importance to producing a good coverage of the gold standard caption. We compute $rel_i = ROUGE_L(t_{in}, C_i)$, where t_{in} is the query caption, and C_i are the 5 captions associated to the retrieved image at rank i. This caption-to-caption relevance model is thus aimed at measuring how much the concepts expressed in the query appear as relevant parts of the retrieved images.

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

[8] LCS is a way to find a common exact sequence of words that is similar to matching word n-grams but less stringent (i.e., inside the LCS sequence other words may appear).

	1K samples					5K samples				
	R@1	R@5	R@10	medR	DCG	R@1	R@5	R@10	medR	DCG
mean	19.1	48.5	64.6	5.9	2.370	6.9	21.3	31.7	25.7	2.408
std	1.047	1.350	1.393	0.349	0.023	0.322	0.507	0.538	0.749	0.012
max	21.6	52.4	67.9	7	2.426	7.6	22.5	33.3	27	2.439
min	15.5	44.8	60.1	5	2.294	5.9	20.0	30.5	24	2.375

Table 1: Variance in evaluation metrics measured on 200 random test splits of 1K images vs. 5K images in MsCOCO. Results correspond to the S-TEXT2VIS-U model projecting into the pool5 layer. The rank-centered metrics R@K and medR present a higher dependency on the sampling size than the DCG.

As a final note on the evaluation, it is noteworthy that many related methods so far [31, 30, 26, 22, 8, 25, 45] have been tested in MsCOCO as well. In doing so, however, they have followed a different experimental protocol, involving one random split of 1K test items (not standard across the experiments) and relying on rank-based metrics, namely, *recall at K* (noted by $R@K$ – the proportion of queries whose expected image was found among the top-K retrieved items) and *medR* (the median of the rank distribution). Despite the fact that this protocol has become almost a standard practice in the literature, we argue it might fail to reflect the scenario we are concerned with here, i.e., the fact that the "prototypical" image one has in mind might be better described through a short textual description of it than through a specific sample image accommodating the textual description (and, unavoidably, much other irrelevant information). Therefore, the rank of the *specific* test image might not necessarily be a good estimator of the system's ability to generalize well in text-to-image retrieval task. That is, although it is clear that a well-performing system will deliver competitive rank-based metrics, it is also true that an overfitted system will rank a test image well whenever a very similar example is seen in the training phase. Contrarily, a text-centered metric (as the DCG with $ROUGE_L$) is not liable to be likewise cheated. Figure 5 reports some examples, taken from actual results from our methods, in which a good recall does not result in a good selection of top ranked images, and examples in which a zero R@5 is obtained on rankings that contain a good selection of relevant images.

Moreover, rank-based metrics are strongly biased towards the test set size (where only 1K images might fail to represent the web-scale scenario) and very unstable with respect to the particular split one could extract from MsCOCO. To show this issue, Table 1, reports the variation of R@1, R@5, R@10, medR and DCG at the variation of the test set size from 1K to 5K images. DCG is the only measures whose value remain stable, while the other measures have a significant drop. All other things being equal, we present additional results we have obtained by following this protocol for the sake of comparability (see below); we also report the MRR (meaning Mean Reciprocal Rank, i.e., the average of the inverse ranks) as a possibly more reliable rank-centered metric.

Figure 5: Evaluation metrics through examples retrieved by our W&D-Text2Vis-4 in the fc7 visual space. The first and second rows of results, despite including the specific query image (ranked 4 and 3, respectively), are mostly unrelated to the textual query; yet they would have obtained a maximum R@5 score. The rest of the examples do not include the query image in the top-5 rank, but are relevant to the query (and arguably, better prototypes of the description than the query image itself). DCG successes to capture this phenomenon.

4.6 Results

Table 2 reports the average DCG scores obtained by the compared methods after five runs with different seeds. These results show a significant improvement of our Text2Vis variants with respect to the compared methods. The best absolute result is obtained by W&D-Text2Vis when using 2 stacked LSTMs in the visual space pool5, which represents the 8% of relative improvement with respect to the baseline *VisSim*. In the fc6 and fc7 visual spaces, the W&D-Text2Vis-4 obtained the best performance, with a relative improvement to *VisSim* of 11.5% and 9.7%, respectively. The S-Text2Vis model also improved, yet by a smaller margin, over the *VisReg* model, showing that an auto-enconding branch in the network is useful to avoid overfitting on visual features. The best performing method in the textual space is *CapRouge*. As detailed in Section 4.4, it cannot be considered a realistic retrieval method, given its computational cost and also because it uses the same measure of the evaluation. We used it to have a strong baseline against which to compare the visual-space based methods, as we discuss in the following.

The Text2Vis methods all compare better by a large margin than the best *CapBoW* and *CapGram* results. The worst Text2Vis result (D-Text2Vis-1 in the fc6 layer) improves by 11.8% over the best *CapGram* result. The best *CapRouge* result is in line with the average Text2Vis results, yet it is affected by the computational issue mentioned before. The best Text2Vis

	Search space						# params
	Textual			Visual			
Method	CapRouge	CapBoW	CapGram	pool5	fc6	fc7	
RRank	1.524	1.524	1.524	1.524	1.524	1.524	-
NeuralTalk	2.016	1.979	1.813	-	-	-	148.3M
Show&Tell-1M	2.290	2.032	2.062	-	-	-	37.6M
Show&Tell-2M	**2.360**	**2.092**	**2.122**	-	-	-	37.6M
VisSim	-	-	-	2.266	2.150	2.180	-
VisReg	-	-	-	2.349	2.317	2.359	14.8M
Word2VisualVec-S$_{cos}$	-	-	-	2.394	2.316	2.317	10.7M
Word2VisualVec-L$_{cos}$	-	-	-	2.405	2.317	2.318	20.8M
Word2VisualVec-S	-	-	-	2.433	2.389	2.386	10.7M
Word2VisualVec-L	-	-	-	2.443	2.390	2.389	20.8M
S-Text2Vis-U	-	-	-	2.428	2.381	2.387	25.4M
S-Text2Vis-N	-	-	-	2.432	2.382	2.384	53.3M
D-Text2Vis-1	-	-	-	2.418	2.372	2.381	10.5M
D-Text2Vis-2	-	-	-	2.435	2.384	2.389	15.7M
D-Text2Vis-4	-	-	-	2.442	2.393	2.388	26.0M
W&D-Text2Vis-1	-	-	-	2.435	2.382	2.385	51.5M
W&D-Text2Vis-2	-	-	-	**2.447**	2.392	**2.391**	56.7M
W&D-Text2Vis-4	-	-	-	2.446	**2.397**	**2.391**	67.0M

Table 2: Performance comparison of the different methods in terms of average DCG. The bold value highlights the best result in each of the search spaces.

result (W&D-Text2Vis-2 on pool5) shows a relative improvement of 3.7% over the best *CapRouge* result, and of 15.3% of the best *CapGram* result.

When comparing the Text2Vis results among themselves it is not obvious whether the use of sparse features leads to better or worse results than the use of dense features. For example, the dense-based models (i.e., the D-Text2Vis variants) improve over the sparse-based models (i.e., the S-Text2Vis variants) only when resorting to stacking LSTM cells. In strict terms of effectiveness (as measured by DCG), this blurs any conclusive remark on the preference of either sparse or dense representations. Notwithstanding, this seemingly contradictory result serves to reinforce another interesting insight, i.e., the fact that, despite being unclear which representation mechanism is preferable, the wide & deep architecture effectively takes advantage of the combination, consistently producing better results.

In the table we report the results obtained by the implementation of Word2VisualVec that uses the cosine similarity model originally adopted in [10] (dubbed Word2VisualVec-S$_{cos}$ and Word2VisualVec-L$_{cos}$). Our implementation using PCA and whitening for the similarity search (dubbed Word2VisualVec-S and Word2VisualVec-L) obtains an average 4.5% of improvement.

We found the following differences in performance between the best configuration of each variant to be statistically significant (two-tailored t-test): both D-Text2Vis and W&D-Text2Vis are significantly better than S-Text2Vis with a confidence $p < 0.005$, while W&D-Text2Vis could only be considered better than Word2VisualVec at the smaller confidence of $p < 0.05$. Further-

more, there are no statistically significant differences between W&D-Text2Vis and D-Text2Vis, nor between D-Text2Vis and Word2VisualVec performances.

In Table 3 we compare our methods against the results reported for other state-of-the-art methods, using their evaluation measures (see section 4.5 for a vaster discussion). The Text2Vis methods perform worse than the other methods yet by a margin that we deem acceptable (a loss of 1-2 ranks in $medR$) considering that all the other methods use a joint space projection, thus they have the drawbacks on the image collection processing we discussed in Section 1. The best performing method, m-CNN, is not even really suited for fast retrieval on large collections, since its network models a distance function, not an explicit projection model, and thus every time a query is given all the collection must be processed by the network to compute the distances between the query and each image.

Method	DCG	MRR	R@1	R@5	R@10	medR
S-Text2Vis-U	2.341	0.339	20.1	48.9	64.2	6
D-Text2Vis-4	2.356	0.355	21.2	51.5	67.4	5
W&D-Text2Vis-4	2.367	0.367	22.8	52.4	67.6	5
W&D-Text2Vis-8	2.370	0.372	22.8	53.2	68.5	5
m-RNN [31]	-	-	29.0	42.2	77.0	3
m-CNN [30]	-	-	32.6	68.6	82.8	3
CCA [FV-HGLMM] [26]	-	-	25.6	60.4	76.8	4
DVSA / BRNN [22]	-	-	27.4	60.2	74.8	3
LRCN [8]	-	-	29.0	61.6	74.8	3
STV [25]	-	-	25.9	60.0	74.6	4

Table 3: Comparison of results on MsCOCO 1K test set for a selection of Text2Vis methods projecting into pool5.

An interesting aspect that deserves attention concerns with the models complexity, as measured by the amount of parameters to train. We have investigated the trade-off between the model complexity and the results delivered (Figure 6). The plot shows that the D-Text2Vis variants require a significantly reduced number of parameters while still being competitive in performance, followed by the S-Text2Vis variants, that however produce fluctuating results in terms of DCG, and finally followed by the W&D-Text2Vis variants, that despite requiring many parameters to train, consistently deliver good results in all visual spaces.

As a final remark, we have investigated the convergence trends on the training loss of the different methods. We found the best performing Text2Vis variants to also converge faster to their better solutions. Word2VisualVec requires instead a much larger number of iterations to converge. Figure 7 shows some selected representative trends; D-Text2Vis-1 converges approximately as fast as S-Text2Vis; the error decreases faster when stacking LSTM cells, and even faster when combining the wide and deep approach. We have also kept

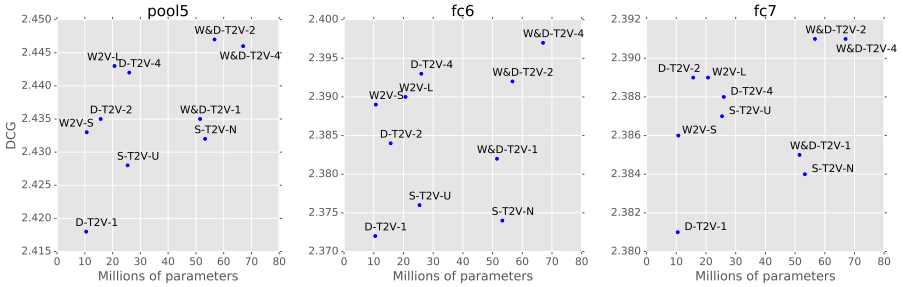

Figure 6: Performance vs number of parameters to learn per model for all feature spaces (pool5, fc6, and fc7 from left to right). Prefix 'S-' denotes S-Text2Vis, 'D-' denotes D-Text2Vis, 'W&D-' denotes W&D-Text2Vis, and 'W2V-' denotes Word2VisualVec.

track of the iteration in which the best solution (estimated in the validation error) was found for each model. In average, the W&D-Text2Vis variants required 18K steps, followed by D-Text2Vis with 22K steps, S-Text2Vis with 25K steps, and finally by Word2VisualVec with 104K steps.

5 Conclusions

We have investigated various neural network model designed to learn a projection from a textual space to a visual space, in order to enable cross-media similarity search without reprocessing the representation of the image collection and the relative data structures one may have already produced to perform image similarity search.

The experiments we conducted indicate that our methods produce better results than those produced by performing similarity search directly on the visual features of a query image. This is an indication that our text-to-image mappings produce better prototypical representations of the desired scene than the representation of a sample image itself. A simple explanation of this result is that textual descriptions strictly emphasize the relevant aspects of the scene the user has in mind, whereas the visual features, directly extracted from the query image, are keeping track of all the information that is contained in that image, causing the similarity search to be potentially confused by secondary elements of the scene.

Our results also indicate that our methods produce better results than those obtained by similarity search methods on the textual space where the images are indexed by means of automatically generated captions. The better results that visual-space based methods have produced over textual-space based ones are not the only argument in favor of the former. We deem that a stronger argument in favor of visual-space methods is the fact the any improvement to the projection method does not require to reprocess the entire image collection,

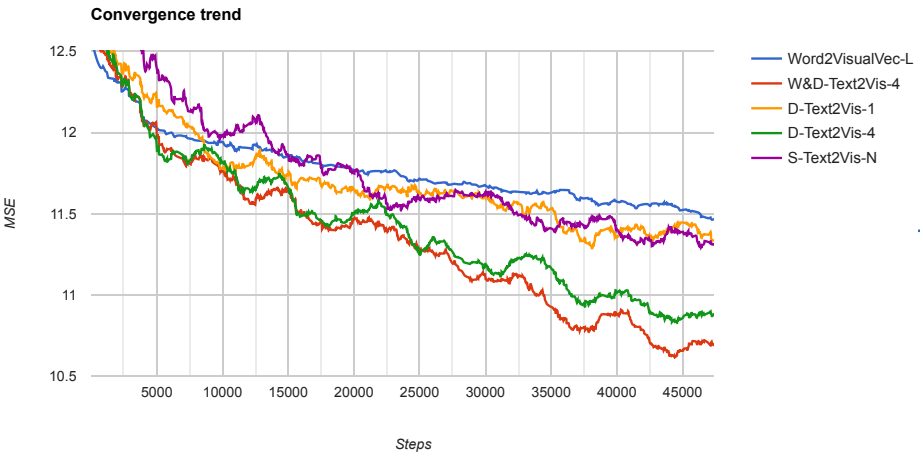

Figure 7: Convergence error loss trends on the fc6 layer of some selected models.

affecting it only the query processing pipeline. A web scale image collection can thus immediately benefit from a model update without requiring any processing. Moreover, a single image similarity search data structure can serve multiple cross-media search models, e.g., built for different languages or specialized on different domains.

We have compared against Word2VisualVec, a recently proposed method that, like ours, uses the visual space as the search space. In our experimental setup we improved the performance obtained by the original Word2VisualVec$_{cos}$ by switching from cosine similarity to euclidean similarity that uses PCA and whitening, following the state of the art in similarity search literature. The improved Word2VisualVec model obtained among the best results, together with D-Text2Vis and W&D-Text2Vis. Our W&D-Text2Vis model improved over Word2VisualVec by a statistical significant margin. W&D-Text2Vis has more parameters than Word2VisualVec but converges much faster.

One interesting aspect that proved to be effective in our experiments is the use of a different t_{out} as a constraint for the hidden representation. When t_{in} and t_{out} are different, though semantically similar, the autoencoder branch becomes semantically constrained. We have investigated this idea in the S-Text2Vis and we believe the same principle could also bring similar benefits for the D-Text2Vis and W&D-Text2Vis models. We thus plan to investigate the effects of such "semantic-autoencoding" principle by adopting a *Seq2Seq* [5] architecture, i.e., by constraining the final memory state from an encoding LSTM after processing the input t_{in} to be a good representation to generate a different, but semantically similar, t_{out} with a decoder LSTM. We believe such an intermediate state to be able to produce better projections to the visual

features.

References

[1] Y. Bai, W. Yu, T. Xiao, C. Xu, K. Yang, W.-Y. Ma, and T. Zhao. Bag-of-words based deep neural network for image retrieval. In *Proceedings of the ACM International Conference on Multimedia*, pages 229–232. ACM, 2014.

[2] S. Cappallo, T. Mensink, and C. G. Snoek. Image2emoji: Zero-shot emoji prediction for visual media. In *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, pages 1311–1314, New York, NY, USA, 2015. ACM.

[3] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

[4] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10. ACM, 2016.

[5] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[6] P. Comon. Independent component analysis, a new concept? *Signal Process.*, 36(3):287–314, Apr. 1994.

[7] J. Costa Pereira, E. Coviello, G. Doyle, N. Rasiwasia, G. R. Lanckriet, R. Levy, and N. Vasconcelos. On the role of correlation and abstraction in cross-modal multimedia retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(3):521–535, 2014.

[8] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[9] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.

[10] J. Dong, X. Li, and C. G. M. Snoek. Word2VisualVec: Cross-Media Retrieval by Visual Feature Prediction. *ArXiv e-prints*, Apr. 2016.

[11] H. Fang, S. Gupta, F. Iandola, R. K. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. C. Platt, et al. From captions to visual concepts and back. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1473–1482, 2015.

[12] F. Feng, X. Wang, and R. Li. Cross-modal retrieval with correspondence autoencoder. In *Proceedings of the ACM International Conference on Multimedia*, pages 7–16. ACM, 2014.

[13] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov, et al. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems*, pages 2121–2129, 2013.

[14] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *European Conference on Computer Vision*, pages 392–407. Springer, 2014.

[15] Y. Gong, L. Wang, M. Hodosh, J. Hockenmaier, and S. Lazebnik. Improving image-sentence embeddings using large weakly annotated photo collections. In *Computer Vision–ECCV 2014*, pages 529–545. Springer, 2014.

[16] A. Gordo, J. Almazán, J. Revaud, and D. Larlus. *Deep Image Retrieval: Learning Global Representations for Image Search*, pages 241–257. Springer International Publishing, Cham, 2016.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

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

[19] X.-S. Hua, L. Yang, J. Wang, J. Wang, M. Ye, K. Wang, Y. Rui, and J. Li. Clickage: Towards bridging semantic and intent gaps via mining click logs of search engines. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 243–252. ACM, 2013.

[20] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[21] H. Jégou and O. Chum. Negative evidences and co-occurences in image retrieval: The benefit of PCA and whitening. In *Computer Vision–ECCV 2012*, pages 774–787. Springer, 2012.

[22] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

[23] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[25] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

[26] B. Klein, G. Lev, G. Sadeh, and L. Wolf. Fisher vectors derived from hybrid gaussian-laplacian mixture models for image annotation. *arXiv preprint arXiv:1411.7399*, 2014.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[28] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In S. S. Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

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

[30] L. Ma, Z. Lu, L. Shang, and H. Li. Multimodal convolutional neural networks for matching image and sentence. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2623–2631, 2015.

[31] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*, 2014.

[32] F. McSherry and M. Najork. *Computing Information Retrieval Performance Measures Efficiently in the Presence of Tied Scores*, pages 414–421. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[33] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[34] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011.

[35] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean. Zero-shot learning by convex combination of semantic embeddings. *arXiv preprint arXiv:1312.5650*, 2013.

[36] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[37] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813, 2014.

[38] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[39] H. Sak, A. W. Senior, and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH*, pages 338–342, 2014.

[40] A. Sharif, R. Hossein, A. Josephine, S. Stefan, K. T. H. Royal, A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014.

[41] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[42] M. Sundermeyer, R. Schlüter, and H. Ney. Lstm neural networks for language modeling. In *Interspeech*, pages 194–197, 2012.

[43] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.

[44] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2015.

[45] L. Wang, Y. Li, and S. Lazebnik. Learning deep structure-preserving image-text embeddings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5005–5013, 2016.

[46] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[47] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.