Language Features Matter: Effective Language Representations for Vision-Language Tasks

Andrea Burns Reuben Tan Kate Saenko Stan Sclaroff Bryan A. Plummer
Boston University
{aburns4, rxtan, saenko, sclaroff, bplum}@bu.edu

Abstract

Shouldn't language and vision features be treated equally in vision-language (VL) tasks? Many VL approaches treat the language component as an afterthought, using simple language models that are either built upon fixed word embeddings trained on text-only data or are learned from scratch. We believe that language features deserve more attention, and conduct experiments which compare different word embeddings, language models, and embedding augmentation steps on five common VL tasks: image-sentence retrieval, image captioning, visual question answering, phrase grounding, and text-to-clip retrieval. Our experiments provide some striking results; an average embedding language model outperforms an LSTM on retrieval-style tasks; state-of-the-art representations such as BERT perform relatively poorly on vision-language tasks. From this comprehensive set of experiments we propose a set of best practices for incorporating the language component of VL tasks. To further elevate language features, we also show that knowledge in vision-language problems can be transferred across tasks to gain performance with multi-task training. This multi-task training is applied to a new Graph Oriented Vision-Language Embedding (GrOVLE), which we adapt from Word2Vec using WordNet and an original visual-language graph built from Visual Genome, providing a ready-to-use vision-language embedding: http://ai.bu.edu/grovle.

1. Introduction

In recent years many methods have been proposed for vision-language tasks such as image and video captioning [12, 27, 47, 48, 52], multimodal retrieval [16, 24, 20, 49, 37, 46, 51], phrase grounding [42, 19, 41, 43], and visual question answering [14, 2, 56, 44, 54]. Language representations for these models tend to be obtained by averaging word embeddings (*e.g.* [49, 41, 40, 24]), feeding features representing each word into a LSTM (*e.g.* [43, 52, 51]), and using word-level or phrase-level attention models (*e.g.* [1, 11, 33, 5, 30]). The word embeddings used in these tasks include a simple one-hot encoding of each word in a vocabulary (*e.g.* [14, 48, 49]), pretrained dense vector representations like Word2Vec [35] or GloVe [38], and Fisher vectors built on top of these dense representations (*e.g.* [24, 40, 49]). Although there are more modern embeddings such as FastText [4], ELMo [39] and BERT [9] that have shown significant performance improvements on language tasks such as sentiment analysis and question answering, many vision-language approaches still use the more dated feature representations.

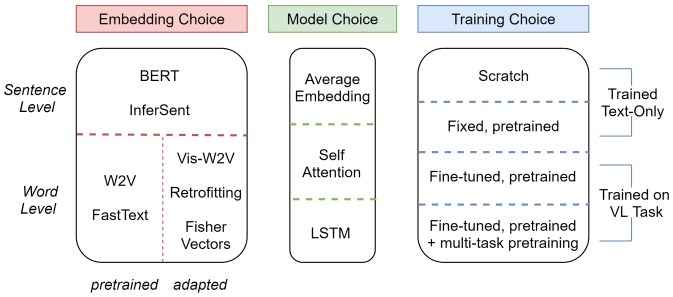

Figure 1. How should language features be constructed for a vision-language task? We provide a side by side comparison of how word-level and sentence-level embeddings, simple and more complex language models, and fine-tuning and post-processing vectors impact performance.

While there are isolated cases where these language model and feature choices are compared for the same task model (*e.g.* [49, 17]), to our knowledge there exists no comprehensive comparison. To address this neglect of language feature exploration, we provide an all-inclusive experimental survey of embedding, language model, and training choice. We perform experiments using from-scratch, Word2Vec [35], WordNet retrofitted Word2Vec [13], FastText [4], Visual Word2Vec [26], HGLMM (300-D, 6K-D) [24], InferSent [8], and BERT [9] representations in addition to a new embedding, GrOVLE, on five vision-language tasks: image-sentence retrieval, visual question answering, phrase grounding, image captioning, and text-to-clip retrieval.

Our goal is to provide insight for vision-language appli-

cations based on extensive experiments varying choices illustrated in Figure 1. Our findings show how to make these choices to take advantage of language features in vision-language work. For example, we find that using an Average Embedding language model, which ignores word ordering, tends to perform better than a LSTM. This suggests that the LSTM overfits to the task it is trained on. However, when training a word embedding from scratch a LSTM performs best. This result is mostly likely a product of the LSTM learning to predict the next word given previous words, learning context. Pretrained word vectors likely already provide some semblance of this context information since that is how they are typically trained. The take-aways from all experimental results are summarized in Figure 2.

Relying on word embeddings trained solely on large text corpora can have important consequences. For example, in Word2Vec the words "boy" and "girl" have higher cosine similarity than either have to the word "child." While this is a subtle difference, it can impact tasks such as image captioning where "girl" can be replaced by "child" when describing a visual scene, but not by "boy." These nuances are not well captured when using text-only information. To address this, we introduce the Graph Oriented Vision-Language Embedding, GrOVLE, which has been learned for vision-language tasks specifically.

When building GrOVLE, we take into account the differences in the relationships between words when used to describe visual data. We introduce a new relational graph by extracting semantic relationships between words using the Visual Genome dataset [28], which is annotated with dense descriptions of entities, their attributes, and their relationships to other entities within an image. We use both WordNet and Visual Genome graphs to adapt Word2Vec, through the retrofitting process defined by Faruqui *et al*. [13].

Finally, in addition to viewing embedding performance for each individual task, we asked: Can an embedding generalize across vision-language tasks? Inspired by multi-task training strategies like PackNet [34], we train the GrOVLE embedding on all the vision-language tasks in our experiments. The word representation becomes more powerful with task specific knowledge, as the multi-task GrOVLE ultimately outperforms its single-task trained version, becoming a leading embedding amongst the five tasks. Note that unlike PackNet, GrOVLE operates directly on the word embeddings rather than model weights.

Below we summarize our primary contributions:

- Comprehensive experiments exhaustively comparing different word representations, language models, and pretraining and adaptation steps across five common vision-language tasks, providing best practices for future work. See Figure 2 for a summary of our findings.
- GrOVLE, a publicly available word embedding which has been specially trained for vision-language tasks[1].
- Key insight into the transferability of word embeddings across the five vision-language tasks through the use of multi-task training.

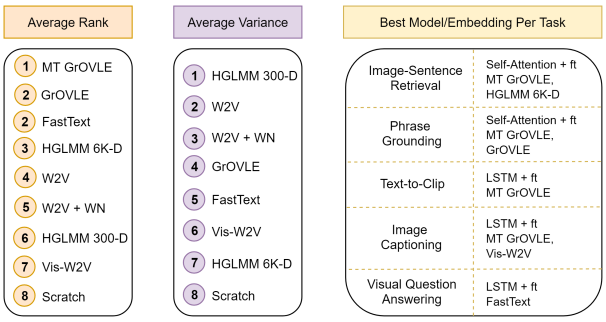

Figure 2. Average rank is defined using each tasks' best performing model. Variance is defined as the average difference between the best and worst performance of the fine-tuned language model options (*e.g*. Average Embedding + ft, Self-Attention + ft, LSTM + ft). Note that variance rank is listed from lowest to highest, *e.g*. from-scratch embeddings have highest variance. If the top embedding per task is a tie, both are provided in the right most column. For the tasks InferSent and BERT operate on, they would land between 7th and 8th place for average rank; average variance is N/A. Note that average variance is not provided for multi-task trained GrOVLE as it was created with the best model for each task.

2. Related Work

To the best of our knowledge, the effect of pretrained embeddings in VL tasks has never before been systematically compared. Visual information has been used in limited ways to improve word embeddings such as simply concatenating visual features [22] or focusing on abstract scenes [26]. Lazaridou *et al*. [29] focuses on leveraging first order semantic relationships by encouraging alignment between the visual and language embeddings for a predefined set of nouns describing objects. Word embeddings have also been improved by including additional constraints on the learning process [55] or as a post-processing step [13]. These models focus on improving some general sense of word similarity. GrOVLE is different in that it is directly optimized to work well on a variety of vision-language tasks. We focus on how 10 representations compare amongst model and training choices, some of which are considered state-of-the-art for language tasks such as the recently introduced BERT [9].

Several vision-language approaches have also tried to improve their language model, rather than the word embeddings, as a way to improve performance. These have included building Fisher vectors on top of pretrained word embeddings [24, 31], constraining a coarse-to-fine word ordering [10, 46], or performing co-reference resolution to

[1] http://ai.bu.edu/grovle

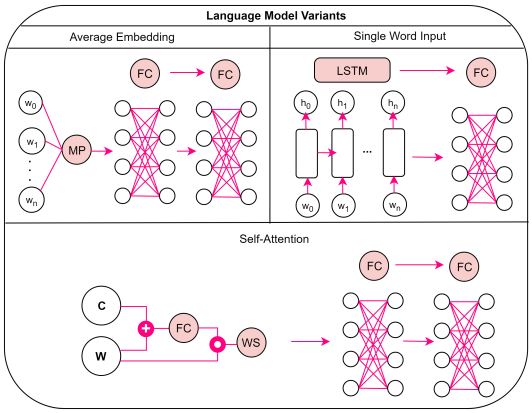

Figure 3. The language model variants used in our experiments include: mean pooling of embeddings (MP) which is then passed to fully connected layers (FC), a LSTM fed a single embedding at a time followed by a fully connected layer, or a self-attention model which builds a weighted context sum (WS) before being passed to a pair of fully connected layers.

identify additional constraints between entities ([50, 41, 25, 6]). Attention mechanisms have also become a popular way to improve performance: word-level attention has been used in image captioning by learning the weights of words using a LSTM [1] or a multi-layered perceptron [52, 11] before being passed to a language generation model. Dual attention [37] has also been used to attend to the question in VQA using feed-forward neural networks. These approaches could be used in conjunction with this work to further improve performance.

3. Language Models

We present three language model options for which we provide experimental results for 8 of 10 different embeddings to determine which language model is best for each task and each embedding (sentence level embeddings cannot be incorporated into some of these architectures).

In Figure 3 an Average Embedding, Self-Attention, and LSTM language architecture are shown. The Average Embedding model consists of mean pooling the embeddings, forming a single representation of all words w_i (with n words in total) in a given sentence or phrase. A sample's pooled vector is then passed through a pair of fully connected layers as shown in the upper left corner of Figure 3.

A more complex language architecture is a LSTM; word representations are individually passed through a LSTM cell, each producing their own hidden state. LSTMs are typically thought of as a "better" architecture choice, modeling the relationship between words in a sentence, as it maintains word ordering. We later show this assumption does not hold true across all vision-language tasks.

Lastly, we compare a Self-Attention model that is closely related to the Average Embedding architecture. The primary difference is the pooling layer, which now consists of two steps. First, a context vector $\mathbf{C}$ is concatenated with all word embeddings in $\mathbf{W}$ of a given sample. Our experiments use the average embedding as context. It is passed through a fully connected layer which applies Softmax to give context "scores" for each word in a sentence. Next, the inner product is taken of these weights and the original word embeddings from $\mathbf{W}$ to produce a context weighted sum which is then passed to a pair of fully connected layers.

4. Experimental Setup

In this section we provide details of each vision-language task. The datasets and vision-language task models are described in supplementary material, but are referenced in Table 1. We split our experiments into three parts: Pretrained Embeddings (Section 5), Adapted Embeddings (Section 6), and Multi-task Trained Embeddings (Section 7).

4.1. Compared Tasks and Metrics

Image-Sentence Retrieval. The goal is to retrieve relevant sentences given an image, or to retrieve relevant images given a sentence. It is evaluated using Recall@K where $K = [1, 5, 10]$, resulting in six numbers which measure the performance of the model (three for image-to-sentence and three for sentence-to-image). We report the average of these six numbers as a measure of overall performance. All six numbers can be found in supplementary material.

Phrase Grounding. In phrase grounding the task is to find the location of a phrase given an image it is known to exist in. Performance is measured using accuracy, where a box is deemed to be successfully localized if it has at least 0.5 intersection over union (IOU) with the ground truth box.

Text-to-Clip. For text-to-clip, the goal is to locate the temporal region (*i.e.* the video clip) that is described by a query. Performance is measured using a mix of Recall@K, where $K = [1, 5]$, and the average IOU the predicted temporal location of a query phrase has with its ground truth temporal segments. We use the evaluation code provided by Hendricks *et al.* [16] in our experiments. We report the average of these three metrics as an overall score; all metrics are reported in supplementary material.

Image Captioning. The goal of image captioning is to produce natural language which describes an image scene with a well formed sentence. The produced captions are evaluated against a set of reference sentences for each image. We report the commonly used evaluation metric BLEU-4, with CIDEr and METEOR results available in the supplementary material.

Visual Question Answering. In VQA [2], the goal is to produce a free-form natural language answer given an image and question. This open-ended task consists of three types of questions: yes/no, number and other. The accuracy of the model is determined by the number of correctly answered questions. We evaluate on the test-dev set.

5. Pretrained Word Embeddings

We begin our exhaustive search across language feature choices with pretrained word embeddings. These offer an initial comparison across techniques that do not use forms of post-processing to adapt embeddings, but rather learn vectors with different model architectures and training objectives. Word2Vec, FastText, InferSent, and BERT are reviewed before results are discussed.

5.1. Word Level Representations

Word2Vec [35] is one of the most widespread word embeddings in use since its release. It builds off of the probabilistic feed forward Neural Network Language Model (NNLM) introduced in [3], which is composed of input, projection, hidden, and output layers. The input is defined by a 1-out-of-V vector where V is the vocabulary size. The projection matrix is shared amongst all words and the computational complexity between hidden and output layers is reduced using a hierarchical Softmax where the vocabulary is represented as a Huffman binary tree.

Word2Vec introduced two variations of the NNLM model, with the primary distinction being that the non-linear hidden layer is removed and the projection layer is shared amongst all words, *i.e*. the words are averaged. This leads to the first model, Continuous Bag of Words (CBOW), in which given four previous and four future words, the current word is predicted. The second model, Skip-Gram, instead predicts the context words given the current word. This results in maximizing the classification of a word given the words it is surrounded by. Skip-Gram tends to perform better with a larger range of context words, but this also results in greater computational complexity.

FastText [4] is an extension of the Word2Vec model in which the atomic entities of the embeddings are no longer words, but are instead character n-grams. N can be decided given the task and time or space constraints. A word is represented as the sum of its character n-gram vectors in addition to the word vector itself. This change of reference can improve performance due to better representation of rare, misspelled, and out of vocabulary words, as the n-grams create more neighbors for use during training.

5.2. Sentence Level Representations

InferSent [8] uses a bi-directional LSTM with max-pooling to create a sentence-level embedding. It is trained using the Natural Language Inference (NLI) task, in which the goal is to categorize natural language English sentence (premise, hypothesis) pairs into three classes: entailment, contradiction, and neutral. The NLI model architecture separately encodes each sentence of the input pair using a BiLSTM. After, the pair's sentences form a shared representation composed of the concatenation of the vectors, the element-wise product, and the absolute element-wise difference. This vector is then fed into a three-class classifier, defined by several FC layers and a Softmax.

BERT [9] is currently the state-of-the-art word embedding model. Its language encoder is a bi-directional multi-layered Transformer which directly follows the architecture described in [45]. The embedding is trained on two tasks: Masked Language Modeling (MLM) and Next Sentence Prediction. The goal of MLM is to predict the original vocabulary ID of a masked word given its context words. Next Sentence Prediction is the binary classification task of determining if the second sentence is the true next sentence.

5.3. Results

We start with an embedding learned from scratch with random initialization as our first baseline. Results demonstrate that while many previous works use scratch embeddings, this greatly impacts performance in vision-language tasks. Unsurprisingly, when comparing the first lines of Table 1(a,b), we find that using Word2Vec rather than an embedding trained from scratch tends to improve performance. This is more important when considering a larger vocabulary as seen comparing phrase grounding experiments on DiDeMo and ReferIt, whose embeddings trained from scratch using their smaller vocabulary compare favorably to Word2Vec.

The original Word2Vec embedding pretrained on Google News can be considered a second baseline. While FastText is a more modern embedding, Word2Vec only falls behind within a point or two across all tasks, and even outperforms or performs equally as well as FastText for certain tasks (*e.g*. text-to-clip, image captioning). This validates works which extend Word2Vec such as Retrofitting, HGLMM Fisher Vectors, and GrOVLE, as Word2Vec may still provide advantages with additional adaptations; results for adapted embeddings follow in Section 6.

Table 1 also contains a comparison of language model variants across the five vision-language tasks we evaluate on. We see that fine-tuning a word embedding on a vision-language task can have dramatic effects on the performance of the language model (*e.g*. 5-10% increase to mean recall on image-sentence retrieval).

When comparing the architecture choices from Figure 3 we see that for retrieval-based tasks (*i.e*. where the output is not free-form text) the Average Embedding and Self-Attention models perform better than a simple LSTM-based approach, with Self-Attention being best on average. This is especially notable since these two models have fewer parameters and are faster to compute than a LSTM. Choosing to use a Self-Attention language model in future vision-language work will not only boost metrics, but will also be a more time efficient option. The only apparent exception to this is the text-to-clip task. This may be because it is a

Task	Image-Sentence Retrieval		Phrase Grounding		Text-to-Clip	Image Captioning		VQA
Dataset	Flickr30K [53]	MSCOCO [32]	Flickr30K Entities [42]	ReferIt [21]	DiDeMo [16]	MSCOCO [32]		VQA [15]
Method	Embedding Network [49]		CITE [40]			ARNet [7]		EtEMN [18]
Metric	Mean Recall		Accuracy		Average	BLEU-4	CIDEr	Accuracy
(a) Training from scratch								
Average Embedding	44.3	73.7	70.46	51.70	33.02	–	–	–
Self-Attention	44.6	77.6	70.68	52.39	33.48	–	–	–
LSTM	60.0	77.5	70.47	51.57	32.83	26.7	89.7	60.95
(b) Word2Vec [35]								
Average Embedding	62.5	75.0	70.03	52.51	32.95	–	–	–
Average Embedding + ft	71.5	78.2	70.85	53.29	32.58	–	–	–
Self-Attention	63.6	75.6	70.19	52.41	33.23	–	–	–
Self-Attention + ft	71.9	79.9	70.94	53.54	33.26	–	–	–
LSTM	68.5	72.5	69.83	52.86	33.73	**28.5**	92.7	61.40
LSTM + ft	69.0	78.2	70.55	53.58	**33.94**	**28.5**	**94.0**	61.35
(c) FastText [4]								
Average Embedding	69.2	78.5	69.75	51.27	32.45	–	–	–
Average Embedding + ft	73.0	**80.7**	70.62	53.24	32.01	–	–	–
Self-Attention	69.5	78.6	69.87	52.49	33.31	–	–	–
Self-Attention + ft	**73.1**	80.6	**71.23**	53.87	33.17	–	–	–
LSTM	69.1	76.9	69.76	52.21	33.06	**28.5**	92.7	**61.86**
LSTM + ft	68.5	80.1	71.09	**53.95**	32.51	28.3	93.2	61.66
(d) Sentence-Level								
InferSent [8]	71.2	76.4	57.83	52.29	31.87	–	–	–
BERT [9]	71.8	75.4	69.38	50.37	32.46	–	–	–

Table 1. Word Embedding Comparison Across Vision Language Tasks. **(a)** contains the results of learning an embedding from scratch *i.e.* random initialization with fine-tuning during training. The remaining sections compare **(b)** Word2Vec, **(c)** FastText, and **(d)** sentence level embeddings InferSent and BERT. All experiments show three model variants: Average Embedding, Self-Attention, and LSTM, with and without fine-tuning during training. Average Embedding and Self-Attention are not used in generation tasks for Image Captioning and VQA as they are known to show worse performance; sentence level embeddings are not applicable for these tasks. See text for discussion.

video-based task which contains some temporal language in its queries [16], so the ordering of words may be especially important to identifying which video clip to select compared to other retrieval-based tasks. While all language models perform closely on ReferIt phrase grounding, this still suggests that there is no need to use the more complex LSTM language model without additional modification.

Lastly, sentence level embeddings InferSent and BERT are compared in Table 1(d); results are without fine-tuning. Fine-tuning would likely improve performance, but is difficult to incorporate due to size (*e.g.* the larger BERT model contains a total of 340M parameters while the well-known VGG-16 network uses 138M; fine-tuning the top layers of BERT still requires loading the full model). The two are comparable to each other with the exception of phrase grounding accuracy on Flickr30K Entities; BERT surprisingly outperforms InferSent by 11.55%. Both InferSent and BERT do not provide the best results across any task, and thus are not a leading option for vision-language tasks.

InferSent and BERT reach comparable values to the best Word2Vec models for image-sentence retrieval on Flickr30K, performing more poorly for the MSCOCO dataset. For the remaining retrieval tasks, metrics are below the best performing model and embedding combination within 1-3 points, again noting the unusual exception of InferSent on phrase grounding of Flickr30K Entities, which significantly drops below scratch performance.

6. Adapted Word Embeddings

Since the introduction of Word2Vec, several enhancement techniques have been proposed. In this section we explore adaptations of Word2Vec which use different methods to post-process embeddings. Extensions either use language enhancements, visual enhancements, or both (*e.g.* WordNet retrofitting, HGLMM vs. Visual Word2Vec vs. GrOVLE, respectively). We shall now briefly discuss these enhancements.

6.1. Visual Word2Vec

Visual Word2Vec [26] is a neural model designed to ground the original Word2Vec representation with visual semantics. Its goal is to maximize the likelihood of a visual context given the set of words used to describe it, thus pushing word representations used to describe the same visual scene closer together. Clusters are first learned offline using

features from abstract clip-art scenes such as the locations of objects, pose, expressions, and gaze to provide surrogate class labels. Word vectors initialized with Word2Vec are then passed through a single hidden layer network. After, a learned output weight matrix and Softmax are applied to predict the visual semantic class the words belong to.

6.2. HGLMM Fisher Vectors

Another post-processed embedding we use for this set of experiments is the Hybrid Gaussian-Laplacian Mixture Model (HGLMM) representation built off of Fisher vectors for Word2Vec [24]. While bag-of-words pooling is simple and commonly applied, Fisher vectors change this pooling technique and achieve state-of-the-art results on many applications. Fisher vectors instead concatenate the gradients of the log-likelihood of local descriptors (which in this case are the Word2Vec vectors) with respect to the HGLMM parameters. HGLMM is a weighted geometric mean of the Gaussian and Laplacian distributions and is fit using Expectation Maximization. Following [49, 40], we reduce the dimensions of the original encodings (18K-D) to 6K-D or 300-D using PCA, as it has been found to improve numerical stability on VL tasks (except for experiments on ReferIt which we reduce to 2K-D due to its small vocabulary size).

6.3. GrOVLE: Graph Oriented Vision-Language Embedding

We provide a new embedding, GrOVLE, which adapts Word2Vec using two knowledge bases: WordNet and Visual Genome. This builds off of the retrofitting work of [13] in which WordNet was one of the lexicon options. The Visual Genome relational graph is novel, as it creates a language graph that captures how words are used in visual contexts, unlike any of the language databases used in [13]. We briefly review retrofitting and then detail the construction of our original Visual Genome word relation graph. GrOVLE provides a vision-language enhanced embedding and outperforms Visual Word2Vec across many tasks. The released version of GrOVLE is multi-task trained, creating an additional level of VL knowledge, later described in Section 7.

6.3.1 Retrofitting Word Embeddings

In this section we review the approach of Faruqui *et al*. [13], which proposed a graph based learning technique to "retrofit" additional semantic knowledge onto pretrained word embeddings.

Given a vocabulary V with words $\{w_1, w_2, ..., w_n\}$ and its corresponding word embedding $\hat{Q}$, where $\hat{q}_i$ is the embedding for w_i, belief propagation is performed to obtain a new embedding Q which minimizes the distances between the embedding representing each word and its neighbors. These neighbors are defined as edges E between words in a graph. $L2$ regularization is performed between the original and new word embeddings to help prevent overfitting. We find that this $L2$ regularization is necessary whenever we are updating the word embeddings (*i.e*. we also use it during multi-task training described in Section 7). We use the same regularization parameters as Faruqui *et al*. and refer the reader to their work to view the final objective function.

6.3.2 Word Relation Graph Construction

Below we describe the methods we use to create the edges between words which share some semantic relation. We use these edges to retrofit the word embeddings with the process described in Section 6.3.1. Of the lexicons provided by Faruqui *et al*. [13], we used only the WordNet graph, as it contains the largest vocabulary with the most edges. A joint lexicon is built with WordNet and Visual Genome as opposed to successively retrofitting the two; this minimized forgetting of the first and thus improved performance.

WordNet [36] is a hierarchical lexical database which organizes nouns, adjectives, verbs and adverbs into sets of synonyms (*synsets*) and uses semantic relations to associate them. As in Faruqui *et al*. [13], we construct a graph by creating links between words if they have a synonym, hypernym, or hyponym relationship.

Visual Genome [28] contains a wealth of language annotations for 108K images: descriptions of entities in an image, their attributes, relationships between multiple entities, and whole image and region-based QA pairs. Each instance in these annotations is considered a sample which we tokenize and remove stopwords from. We compute co-occurrence statistics over pairs of words within the sample for pairs that occur more than 50 times, resulting in 322,928 pairs for 12,849 words. For each word we compute a pointwise mutual information (PMI) score for all pairs it occurs in, and create links between the top ten words. This creates a graph where words that occur frequently together when describing visual data are linked.

6.4. Results

We see a small, but consistent improvement across most of the vision-language tasks using GrOVLE as seen in Table 2(b). These changes result in an embedding with comparable performance to the HGLMM 6K-D features, which are reported in Table 2(e). However, our word embedding tends to perform better when embeddings are the same size (*i.e*. 300-D). For the generation-based tasks (*i.e*. captioning and VQA), the benefits of using adapted embeddings are less clear. This may simply be an artifact of the challenges in evaluating these tasks (*i.e*., the captions are improving in a way the metrics don't capture). Also, models that more carefully consider the effect of each word in a caption may benefit more from our improved features (*e.g*. [37, 51]).

While Visual Word2Vec is an established visually-enhanced embedding, its published results did not include

	Task	Image-Sentence Retrieval		Phrase Grounding		Text-to-Clip	Image Captioning		VQA
	Dataset	Flickr30K	MSCOCO	Flickr30K Entities	ReferIt	DiDeMo	MSCOCO		VQA
	Metric	Mean Recall		Accuracy		Average	BLEU-4	CIDEr	Accuracy
(a)	**Word2Vec + wn** [13]								
	Average Embedding + ft	72.0	79.2	70.51	53.93	33.24	–	–	–
	Self-Attention + ft	72.4	80.0	70.70	53.81	33.65	–	–	–
	LSTM + ft	69.3	78.9	70.80	53.67	34.16	28.6	93.3	61.06
(b)	**GrOVLE**								
	Average Embedding + ft	72.3	80.2	70.77	**53.99**	33.71	–	–	–
	Self-Attention + ft	72.1	80.5	70.95	53.75	33.14	–	–	–
	LSTM + ft	69.7	78.8	70.18	**53.99**	34.47	28.3	92.5	61.22
(c)	**Visual Word2Vec** [26]								
	Average Embedding + ft	66.8	78.7	70.61	53.14	31.73	–	–	–
	Self-Attention + ft	68.8	79.2	**71.07**	53.26	31.15	–	–	–
	LSTM + ft	66.7	74.5	70.70	53.19	32.29	**28.8**	**94.0**	61.15
(d)	**HGLMM (300-D)** [24]								
	Average Embedding + ft	71.0	79.8	70.64	53.71	32.62	–	–	–
	Self-Attention + ft	71.8	80.4	70.51	53.83	33.44	–	–	–
	LSTM + ft	69.5	77.9	70.37	53.10	33.85	28.7	**94.0**	**61.44**
(e)	**HGLMM (6K-D)** [24]								
	Average Embedding + ft	73.5	**80.9**	70.83	53.36	32.66	–	–	–
	Self-Attention + ft	**75.1**	80.6	71.02	53.43	33.57	–	–	–
	LSTM + ft	68.0	79.4	70.38	53.89	**34.62**	28.0	92.8	60.58

Table 2. Modifications of Word2Vec. **(a)** contains Word2Vec retrofitted results using only the WordNet (wn) lexicon from [13]. Next, **(b)** is our baseline embedding which includes the new Visual Genome relational graph. Visual Word2Vec results are provided in **(c)**, and **(d)**, **(e)** are Fisher vectors on top of Word2Vec. See text for discussion.

Task	Image-Sentence Retrieval		Phrase Grounding		Text-to-Clip	Image Captioning		VQA
Metric	Mean Recall		Accuracy		Average	BLEU-4	CIDEr	Accuracy
GrOVLE w/o multi-task pretraining	64.7	75.0	70.53	52.15	34.45	28.5	92.7	61.46
+ multi-task pretraining w/o target task	65.8	76.4	70.82	52.21	34.57	**28.8**	**93.3**	61.47
+ multi-task pretraining w/ target task	66.2	80.2	70.87	52.64	34.82	28.5	92.7	**61.53**
+ multi-task pretraining w/ target task + ft	**72.6**	**81.3**	**71.57**	**54.51**	**35.09**	28.7	93.2	61.46

Table 3. Comparison of training our word embeddings on four tasks and testing on the fifth, as well as training on all five tasks.

these vision-language tasks. Visual Word2Vec performs comparably amongst results for generation tasks (*i.e.* image captioning and VQA), but these tasks have little variance in results, with less than a point of difference across the adapted embeddings. The small gain provided in generation tasks by Visual Word2Vec does not out-weight the drops in performance across other tasks such as the significant mean recall drop of 6.3 compared to HGLMM's 6K-D Self-Attention result in line two of Table 2(c) and Table 2(e) for image-sentence retrieval of Flickr30K. For comparison, GrOVLE's Self-Attention result in Table 2(b) is only 3 points lower.

Finally, we report results using HGLMM of different dimension. HGLMM 300-D features are used for a more fair comparison to other embeddings. While the HGLMM 6K-D representation primarily results in the highest performance, it performs more poorly on generation tasks and also results in high variance. For example, column one in Table 2(e) shows a range of 7.1 in mean recall, unlike GrOVLE which has a range of 2.6.

7. Multi-task Training

A drawback of using pretrained word embeddings like Word2Vec or the retrofitting process is that they are trained solely on text data. While our Visual Genome Graph provides some general information on how words in our vocabulary are used for visual data, it doesn't provide any sense of visual similarity between semantically different words that may be necessary to perform a particular vision-language task. To address this, we fine-tune GrOVLE across the five VL tasks.

We provide results for a four and five multi-task trained embedding. The four task experiments are performed with the final task embedding fixed to demonstrate how well the embeddings would generalize to new tasks. We also provide results for pretraining on five tasks with and without fine-tuning during the last task. Similarly to PackNet [34], for each dataset/task in the four and five task experiments, we keep the K most informative features frozen when training any subsequent task, diminishing the effect of catastrophic

Task	Image-Sentence Retrieval		Phrase Grounding		Text-to-Clip	Image Captioning		VQA
Additional Models	SCAN [30]		QA R-CNN [17]		TGN [5]	BUTD [1]		BAN[23]
Metric	Mean Recall		Accuracy		Average	BLEU-4	CIDEr	Accuracy
Training from scratch	72.8	83.2	68.56	50.23	43.91	35.2	109.8	68.98
FastText + ft	72.5	83.8	69.27	53.01	44.21	35.2	110.3	69.91
GrOVLE (w/o multi-task pretraining) + ft	72.7	84.1	70.03	53.88	**45.26**	35.1	110.4	69.36
+ multi-task pretraining w/ target task + ft	**76.2**	**84.7**	**71.08**	**54.10**	43.61	**35.7**	**111.6**	**69.97**

Table 4. We include results with additional models to verify trends. See text for discussion and supplementary material for more.

forgetting when fine-tuning on a new task. For an embedding of size D and T tasks, $K = \frac{D}{T}$, *i.e.* $K = 60$ in our experiments. We evenly split the K features for tasks with multiple datasets. Features that were tuned on a task are ranked according to variance and frozen before training on the next dataset/task. The end result is a pretrained word embedding which can be "dropped in" to existing models to improve performance across many vision-language tasks.

To verify that the multi-task GrOVLE performance improvements generalize across task model architecture, we provide results using additional task models in Table 4. More results can be found in the supplementary material.

7.1. Results

Table 3 reports results of the multi-task training procedure described above. We use the best performing language model in our comparisons for each task, *i.e.* Self-Attention for image-sentence retrieval and phrase grounding, and the LSTM language model for text-to-clip, image captioning, and VQA. The first lines of Table 3 report the results of the original fixed GrOVLE embedding, which should be considered the baseline. The second line of Table 3 reports performance when the four-task pretrained GrOVLE is fixed when used in the target task, *i.e.* the task currently being run. The third and fourth line of Table 3 report the results of our embedding when they were trained on all five tasks, and kept fixed or fine-tuned for the target task, respectively.

The results of line three and four demonstrate that our improved embedding tends to transfer better when applied with fine-tuning during the target task. We find similar trends in performance improvements across tasks: larger gains occur for image-sentence retrieval with +7.9 mean recall for the Flickr30K dataset and +6.3 for MSCOCO. All other tasks have performance improvements under one point, showing that while the vision-language tasks appear to transfer well without harming performance, they are leveraged most in image-sentence retrieval, with an exception of phrase grounding accuracy on ReferIt (+2.36%).

Table 4 provides more models per task and demonstrates consistent results: embeddings can significantly affect performance and GrOVLE variants are still the best embedding overall. As we move down the table we find even larger performance improvements made by using the five-task pretrained GrOVLE with fine-tuning than in Table 3. This multi-task variant is the best performing across all tasks, thus we release this embedding for public use.

8. Conclusion

We believe there are five major findings in our experiments that researchers should keep in mind when considering the language component for vision-language tasks:

1. On retrieval-style tasks, the Average Embedding and Self-Attention language model tend to outperform a simple LSTM.

2. Fine-tuning a word embedding for a task can significantly impact performance.

3. For standard vision-language metrics, language features matter most on retrieval and grounding tasks, and less on text-to-clip and generation tasks.

4. Word embeddings trained on outside vision-language datasets and tasks generalize to other applications.

5. Multi-task trained GrOVLE is the leading embedding option for four of the five vision-language tasks when used with the best corresponding language model.

We have provided evidence that language and vision features should be treated equally when used in vision-language tasks. When using the best embedding, language model, and training choices, performance for tasks with more variance can greatly improve, and tasks with more stubborn performance metrics can be nudged further. These insights are proposed to benefit future vision-language work. Along with these findings, we have introduced GrOVLE, which incorporates hierarchical language relations from WordNet as well as language with visual context from Visual Genome. In addition to these adaptations, we perform multi-task training with five common vision-language tasks to further incorporate nuanced visual information. This provides a 300-D embedding with vision-language enhancements that is comparable to current embeddings and provides low variance results.

Acknowledgements

We would like to thank the reviewers for their helpful suggestions. This work is supported in part by DARPA and NSF awards IIS-1724237, CNS-1629700, CCF-1723379.

References

[1] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*, 2018. 1, 3, 8

[2] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: Visual Question Answering. In *ICCV*, 2015. 1, 3

[3] Yoshua Bengio, Rjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. In *Journal of Machine Learning Research, 3:1137-1155*, 2003. 4

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. 1, 4, 5

[5] Jingyuan Chen, Xinpeng Chen, Lin Ma, Zequn Jie, and Tat-Seng Chua. Temporally grounding natural sentence in video. In *EMNLP*, 2018. 1, 8

[6] Kan Chen, Rama Kovvuri, and Ram Nevatia. Query-guided regression network with context policy for phrase grounding. In *ICCV*, 2017. 3

[7] Xinpeng Chen, Lin Ma, Wenhao Jiang, Jian Yao, and Wei Liu. Regularizing rnns for caption generation by reconstructing the past with the present. In *arXiv:1803.11439v2*, 2018. 5

[8] Alexis Conneau, Douwe Kiela, Holger Schwenk, and Loc Barraultand Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. 2017. 1, 4, 5

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *arXiv:1810.04805v1*, 2018. 1, 2, 4, 5

[10] Fartash Faghri, David J. Fleet, Jamie Ryan Kiros, and Sanja Fidler. VSE++: improving visual-semantic embeddings with hard negatives. In *BMVC*, 2018. 2

[11] Fang Fang, Hanli Wang, and Pengjie Tang. Image captioning with word level attention. *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 1278–1282, 2018. 1, 3

[12] Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John Platt, et al. From captions to visual concepts and back. *arXiv:1411.4952*, 2014. 1

[13] Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. Retrofitting word vectors to semantic lexicons. In *NAACL*, 2015. 1, 2, 6, 7

[14] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *EMNLP*, 2016. 1

[15] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 5

[16] Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. Localizing moments in video with natural language. In *ICCV*, 2017. 1, 3, 5

[17] Ryota Hinami and Shin'ichi Satoh. Discriminative learning of open-vocabulary object retrieval and localization by negative phrase augmentation. In *EMNLP*, 2018. 1, 8

[18] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. *CoRR, abs/1704.05526*, 3, 2017. 5

[19] Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. Natural language object retrieval. In *CVPR*, 2016. 1

[20] Yan Huang, Qi Wu, and Liang Wang. Learning semantic concepts and order for image and sentence matching. In *CVPR*, 2018. 1

[21] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. ReferItGame: Referring to objects in photographs of natural scenes. In *EMNLP*, 2014. 5

[22] Douwe Kiela and Lon Bottou. Learning image embeddings using convolutional neural networks for improved multimodal semantics. In *EMNLP*, 2014. 2

[23] Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. Bilinear attention networks. In *NeurIPS*, 2018. 8

[24] Benjamin Klein, Guy Lev, Gil Sadeh, and Lior Wolf. Fisher vectors derived from hybrid gaussian-laplacian mixture models for image annotation. In *CVPR*, 2015. 1, 2, 6, 7

[25] Chen Kong, Dahua Lin, Mohit Bansal, Raquel Urtasun, and Sanja Fidler. What are you talking about? text-to-image coreference. In *CVPR*, 2014. 3

[26] Satwik Kottur, Ramakrishna Vedantam, Jose M. F. Moura, and Devi Parikh. Visual word2vec (vis-w2v): Learning visually grounded word embeddings using abstract scenes. In *CVPR*, 2016. 1, 2, 5, 7

[27] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *ICCV*, 2017. 1

[28] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. 2016. 2, 6

[29] Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. Combining language and vision with a multimodal skip-gram model. In *NAACL*, 2015. 2

[30] Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. Stacked cross attention for image-text matching. In *ECCV*, 2018. 1, 8

[31] Guy Lev, Gil Sadeh, Benjamin Klein, and Lior Wolf. RNN fisher vectors for action recognition and image annotation. In *ECCV*, 2016. 2

[32] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 5

[33] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 289–297, USA, 2016. Curran Associates Inc. 1

[34] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *CVPR*, 2018. 2, 7

[35] Tom Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *NAACL*, 2013. 1, 4, 5

[36] George A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 1995. 6

[37] Hyeonseob Nam, Jung-Woo Ha, and Jeonghee Kim. Dual attention networks for multimodal reasoning and matching. In *CVPR*, 2017. 1, 3, 6

[38] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014. 1

[39] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *NAACL*, 2018. 1

[40] Bryan A. Plummer, Paige Kordas, M. Hadi Kiapour, Shuai Zheng, Robinson Piramuthu, and Svetlana Lazebnik. Conditional image-text embedding networks. In *ECCV*, 2018. 1, 5, 6

[41] Bryan A. Plummer, Arun Mallya, Christopher M. Cervantes, Julia Hockenmaier, and Svetlana Lazebnik. Phrase localization and visual relationship detection with comprehensive image-language cues. In *ICCV*, 2017. 1, 3

[42] Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30K Entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. *IJCV*, 123(1):74–93, May 2017. 1, 5

[43] Anna Rohrbach, Marcus Rohrbach, Ronghang Hu, Trevor Darrell, and Bernt Schiele. Grounding of textual phrases in images by reconstruction. In *ECCV*, 2016. 1

[44] Tatiana Tommasi, Arun Mallya, Bryan A. Plummer, Svetlana Lazebnik, Alex C. Berg, and Tamara L. Berg. Solving Visual Madlibs with Multiple Cues. In *BMVC*, 2016. 1

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010, 2017. 4

[46] Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. Order embeddings of images and language. In *ICLR*, 2016. 1, 2

[47] Subhashini Venugopalan, Marcus Rohrbach, Jeff Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. Sequence to sequence – video to text. In *ICCV*, 2015. 1

[48] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015. 1

[49] Liwei Wang, Yin Li, Jing Huang, and Svetlana Lazebnik. Learning two-branch neural networks for image-text matching tasks. *arXiv:1704.03470*, 2017. 1, 5, 6

[50] Mingzhe Wang, Mahmoud Azab, Noriyuki Kojima, Rada Mihalcea, and Jia Deng. Structured matching for phrase localization. In *ECCV*, 2016. 3

[51] Huijuan Xu, Kun He, Bryan A. Plummer, Leonid Sigal, Stan Sclaroff, and Kate Saenko. Multilevel language and vision integration for text-to-clip retrieval. In *AAAI*, 2019. 1, 6

[52] Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv:1502.03044*, 2015. 1, 3

[53] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *TACL*, 2:67–78, 2014. 5

[54] Licheng Yu, Eunbyung Park, Alexander C. Berg, and Tamara L. Berg. Visual Madlibs: Fill in the blank Image Generation and Question Answering. *ICCV*, 2015. 1

[55] Mo Yu and Mark Dredze. Improving lexical embeddings with semantic knowledge. In *ACL*, 2014. 2

[56] Peng Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Yin and Yang: Balancing and answering binary visual questions. In *CVPR*, 2016. 1