A neural architecture to learn image-text joint embedding

Sethu Hareesh Kolluru
hareesh@stanford.edu

Abstract

In this project, we explore the task of measuring semantic similarity between visual data and text data. We build two-branch neural networks for learning the similarity and train and validate on Flickr30K and MSCOCO datasets for image-sentence retrieval task i.e given an input image, the goal is to find the best matching sentences from a database. We conduct numerical investigations to quantify and understand the impact of several components of the proposed architecture on the image-sentence retrieval task.

1 Background and Motivation

The advent of deep learning has made great strides towards better visual understanding [1], and generating rich descriptions of visual data, in particular, in the form of natural language. A critical task for the applications such as bi-directional image and text retrieval [2, 3], image captioning [4] and visual question answering [5] is to learn joint embedding that entails mapping from two or more domains into common latent vector space in which semantically associated inputs are mapped to similar locations.

In this project, we build and study two-branch neural architectures to learn image-text joint embedding for the image-sentence retrieval task i.e given an input image, the goal is to find the best matching sentences from a list of given sentences. The formulation and model architecture of our work is similar to [6, 3], employing triplet ranking loss as the training objective. We validate the effectiveness of our approach on image-sentence retrieval task on Flickr30k [7] and MSCOCO [8] datasets.

Triplet (i.e query, matching and non-matching) selection or negative mining has been shown to have an impact on the representational efficiency of image embeddings [9] and cross-modal embeddings [3]. We complement this work by performing numerical experiments to quantify the impact of image encoder i.e VGG19 [10] and ResNet50 [11] and negative mining strategies i.e selecting $K = 1, 10$ hard negatives on the evaluation metrics.

2 Related Work

In this section, literature work related to learning cross-modal joint embedding, its application to the image-sentence retrieval task and the enablers (datasets) in this context are outlined.

Most popular approaches for obtaining image-text joint embeddings [12, 13] are based on Canonical Correlation Analysis, which seeks pairs of linear projections that maximizes the correlation of the two views. These methods have shown state-of-the-art results with image and text features, but have a high memory cost as they involve loading all data into memory and computing covariance between image and text data. To scale CCA to larger datasets, [14] and [15] proposed to cast CCA into a deep learning framework, but this approach suffered from stability issues.

DeVISE [16] applied a margin-based ranking loss to learn linear transformations of visual and text features into latent shared space. Wang et al. [2] extended their work using two-branch neural

networks and bi-directional ranking loss as training objective, which significantly outperformed CCA-based methods in both stability and scale.

FaceNet [9] has shown that the method of mining triplets of matching and non-matching pairs during training significantly improves representational efficiency of the embedding. VSE++[3] has successfully employed a similar hard negative mining method to learn embedding space across two different modalities using asymmetric branches and achieved state-of-the-art results. Our work draws inspiration from these approaches.

A major contributing factor to the progress of deep learning, especially to the problem of image classification is the availability of large-scale, publicly-available datasets such as ImageNet[17]. Similarly, research progress in the application of image-text matching tasks can be related to the existence of datasets such as Flickr30K[18] and MSCOCO[8]. A more recent effort to build a much larger dataset resulted in Google's Conceptual Captions dataset [19] which has more than 3 million images, paired with natural-language captions. It will be interesting to explore the task of learning joint embedding on such a large dataset.

3 Approach

In this section, we first describe the two branch network structure as well as image and text embedding (section 3.1).Then, we present the training objective for learning image text mapping using triplet ranking loss (section 3.2). Finally, we present our triplet selection strategy and negative mining techniques to learn an improved embedding (section 3.3).

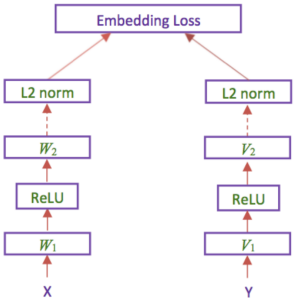


Figure 1: The architecture of the two-branch embedding network [2]

3.1 Network Architecture

The *embedding network* consists of two branches, each of which takes either an image embedding or text embedding, pass them through one or more layers of transformations, fuse them, and eventually output a learned similarity score as shown in Figure 1. Each branch is composed of a series of fully connected (FC) layers, separated by Rectified Linear Unit (ReLU) nonlinearities. We apply batch normalization [20] right after the last FC layer (without ReLU) to improve the convergence during training. The output vectors are further normalized by their L2 norm for efficient computation of similarity score.

The inputs to this network can be either pre-computed image and text features or outputs of other networks (e.g. CNNs or RNNs). The idea here is that, by feeding each branch with a different encoder network for a specific modality, the encoder networks will focus on identifying modality-specific features at first and the embedding network will convert the modality-specific features to modality-robust features. The network architecture is flexible in that embedding network can accommodate additional layers as well as can be fine-tuned together with the encoder network.

In our work, we focus on investigating the behavior of the two-branch networks with inputs as pre-computed image and text embedding as discussed below.

Image Embedding: We adopt a deep CNN model trained on ImageNet dataset as the image encoder. Specifically, we experiment with state-of-the-art 50 layer ResNet model [11] and 19 layer VGG

model [10] in this work. We feed the images into CNNs as inputs and extract image features directly from the penultimate FC layer. The dimension of the image embedding thus obtained is 512 for ResNet50 and 4096 for VGG19.

Text Embedding: We generate the feature representation of text by summing over the 300-dimensional GloVe [21] embedding of all words associated with an image and then normalizing it by the number of words.

3.2 Triplet ranking loss

The training objective is to minimize a triplet loss function, which applies a margin-based penalty to an incorrect annotation when it gets ranked higher than a correct one for describing an image as well as ensuring that for each annotation, the corresponding image gets ranked higher than unrelated images.

Given a training image x_i, let Y_p and Y_n denote its sets of matching (positive) and non-matching (negative) sentences, respectively. We want the distance between x_i and each positive sentence y_p to be smaller than the distance between x_i and each negative example y_n by some enforced margin m.

$$d(x_i, y_p) + m < d(x_i, y_n) \quad \forall y_p \in Y_p \quad and \quad \forall y_n \in Y_n \tag{1}$$

Similarly, given a sentence $y_{i'}$, we have

$$d(x_{p'}, y_{i'}) + m < d(x_{n'}, y_{i'}) \quad \forall x_{p'} \in X_{p'} \quad and \quad \forall x_{n'} \in X_{n'} \tag{2}$$

where X_p and X_n denote its sets of matching (positive) and non-matching (negative) images for $y_{i'}$.

To formalize this requirement, the bi-directional ranking loss will be defined over triplets of embeddings (x_i, y_p, y_n) and $(x_{p'}, x_{n'}, y_{i'})$,

$$\begin{aligned} L(X, Y) \quad = \quad & \sum_{i,j,k} max[0, m + d(x_i, y_p) - d(x_i, y_n)]_+ \\ & + \lambda \sum_{i',j',k'} max[0, m + d(x_{p'}, y_{i'}) - d(x_{n'}, y_{i'})]_+ \end{aligned} \tag{3}$$

where $[t]_+ = max(0, t)$ and parameter λ controls the strength of ranking loss in either direction.

3.3 Triplet Selection

Identifying the triplets of examples that violate the constraints in eqn (1) and eqn (2) during training is crucial for achieving the best performance [9]. Based on the definition of the loss, there are three categories of triplets:

- *easy triplets*: triplets which have a loss of 0, because $d(x_i, y_p) + m < d(x_i, y_n)$
- *hard triplets*: triplets where the negative is closer to the anchor than the positive, i.e. $d(x_i, y_n) < d(x_i, y_p)$
- *semi-hard triplets*: triplets where the negative is not closer to the anchor than the positive, but which still have positive loss: $d(x_i, y_p) < d(x_i, y_n) < d(x_i, y_p) + m$

Each of these definitions depend on where the negative is, relatively to the anchor and positive. We can therefore extend these three categories to the negatives: hard negatives, semi-hard negatives or easy negatives. Figure 2 shows the three corresponding regions of the embedding space for the negative. Ideally, for a given image and text pair, we would like to mine the entire training set for negative examples. But such a task is computationally not feasible. To avoid this issue, we select the triplets by mining only the mini-batch in an online learning fashion [9]. Suppose we have a mini-batch of image and text pair inputs of size B. Several strategies to pick triplets among the valid ones, to be used in loss computation are

- **selecting all hard and semi-hard negatives**: select all the valid triplets, and average the loss on the hard and semi-hard triplets not taking into account the easy triplets (those with loss 0), as averaging them would make the overall loss very small. This produces a total of B^2 triplets (B (anchors, positive) pairs and B possible negatives)

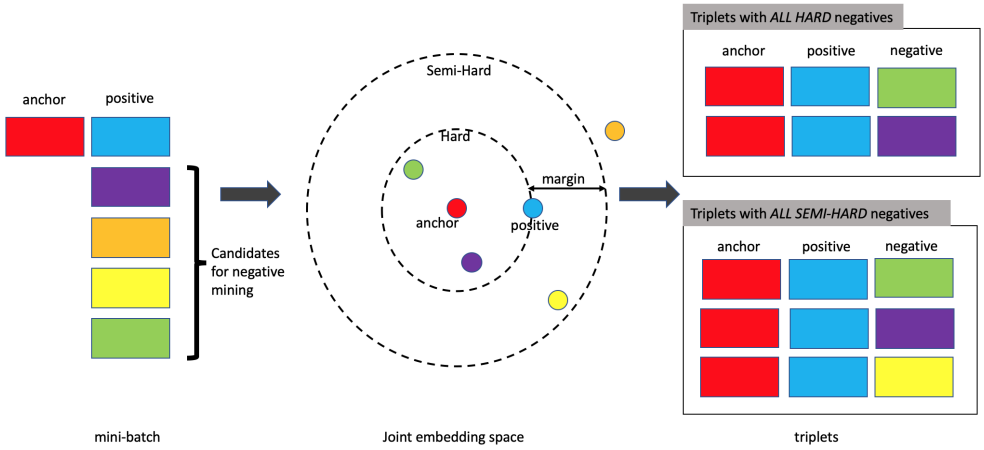


Figure 2: Negative Mining

- **selecting all hard negatives only**: narrowing the selection to only hard negatives.
- **selecting all semi-hard negatives only**: narrowing the selection to only semi-hard negatives.
- **selecting K hardest negatives**: for each anchor, select the hardest K among the mini-batch, this produces BK triplets which are the nearest among the mini-batch and therefore contribute most to the loss function.

Though using the hardest negative results only in faster convergence [3], [9] suggested that selecting hardest negatives can in practice lead to bad local minima early on in training and using semi-hard negative instead. We explore *batch hard* mining in our work, employing one hardest example per mini-batch, $(K = 1)$, multiple hard negative examples $(K = 10)$ per mini-batch and only semi-hard examples for loss computation.

4 Experiments

4.1 Datasets

We evaluate our proposed model on the Flickr30K [18] dataset and the MSCOCO [8] dataset. Flickr30K dataset includes $31,783$ images, while the MSCOCO dataset consists of about $123,000$ images and each image is annotated with 5 sentences in both datasets. Following [2], we use 1000 images for validation and 1000 images for testing for Flickr30K dataset and for MSCOCO, we use 1000 images for both validation and testing.

4.2 Experimental setup

We implemented our model using Tensorflow[6] and Keras[22]. We use Adam optimizer with learning rate of 0.0002. We set batch size to be 64. We apply dropout with keep-probability 0.5 after ReLU layer and use $\lambda = 0.1$ to control the direction of the loss. We use margin $m = 0.1$ in the triplet ranking loss.

4.3 Evaluation Metrics

Recall@K (K=1, 5, 10) [4], which indicates the percentage of the queries where at least one ground-truth is retrieved among the top-K results, is used as quantitative metric in this evaluation.

4.4 Results

Numerical results i.e Recall@1, Recall@5 and Recall@10, when our models are employed on the image-sentence retrieval task on Flickr30K and MSCOCO datasets are presented in Table 1 and Table

Table 1: Image-to-Text Retrieval Results on Flickr30K Dataset

#		Image-to-Sentence			Sentence-to-Image		
		R@1	R@5	R@10	R@1	R@5	R@10
state-of-art	Embedding Network [2]	43.2	71.6	79.8	31.7	61.3	72.4
	VSE++[3]	51.3	82.2	91.0	40.1	75.3	86.1
Image Encoder	Our model with VGG19 image encoder and $K = 1$ hard negative	19.6	43.0	56.2	15.3	36.7	48.7
	Our model with ResNet50 image encoder and $K = 1$ hard negative	13.6	24.4	36.1	11.8	23.6	32.4
Triplet Sampling	Our model with VGG19 image encoder and $K = 10$ hard negative	3.5	12.5	16.7	2.9	10.9	14.5
	Our model with ResNet50 image encoder and $K = 10$ hard negative	3.6	10.2	18.6	3.4	8.9	15.3

Table 2: Image-to-Text Retrieval Results on MSCOCO Dataset

#		Image-to-Sentence			Sentence-to-Image		
		R@1	R@5	R@10	R@1	R@5	R@10
state-of-art	Embedding Network [2]	54.9	84.0	92.2	43.3	76.4	87.5
	VSE++[3]	64.6	90.0	95.7	52.0	84.3	92.3
Image Encoder	Our model with VGG19 image encoder and $K = 1$ hard negative	30.5	59.5	72.5	21.8	53.0	69.2
	Our model with ResNet50 image encoder and $K = 1$ hard negative	10.7	35.5	51.3	8.9	29.7	45.1
Triplet Sampling	Our model with VGG19 image encoder and $K = 10$ hard negative	2.9	12.9	20.9	3.5	11.4	18.9
	Our model with ResNet50 image encoder and $K = 10$ hard negative	5.6	19.4	30.1	6.1	19.3	30.9

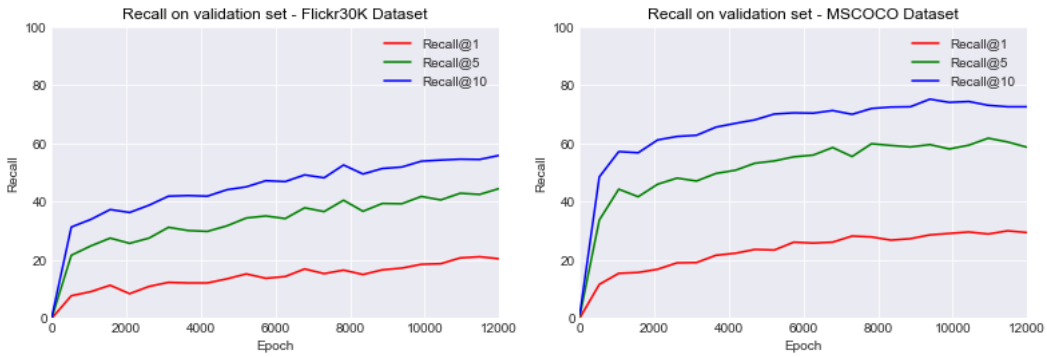
2 respectively. We also present the results from [2] and [3], which are state-of-art in this regard and thus serve as the baseline for the current study. We plot the Recall@1, Recall@5 and Recall@10 on validation split as a function of training epoch on both Flickr30K and MSCOCO datasets in Figure 3a and Figure 3b respectively.

Our model with best performance $i.e$ VGG19 image encoder and using $K = 1$ hard negative triplet selection has shown good results on both image-to-sentence and sentence-to-image tasks. Relative to state-of-art, however, the performance is lower by about $20 - 25\%$ on both Flickr30K and MSCOCO datasets. One possible reason for this could be due to the fact that we are using GloVe based word embeddings to represent sentences, which does not take dependency information into consideration. (We expand on thus further in section 4.5). Another possible reason could just be that we stopped training process prematurely due to time constraints as seen in Figure 3a and Figure 3b, where the Recall on validation split does not seem to saturate during training process.

It is to be noted that the term "epoch" in our study should really be inferred as to the number of minibatches employed during training. For example, Flickr30K dataset which has about $30,000$ training examples takes about 500 minibatches of size 64 to go through the entire training set once. In our work, we refer to this epoch as 500, instead of common practice as 1. This is just an unfortunate artifact of the way we are implementing the triplet selection, which happens at each training step for each minibatch.

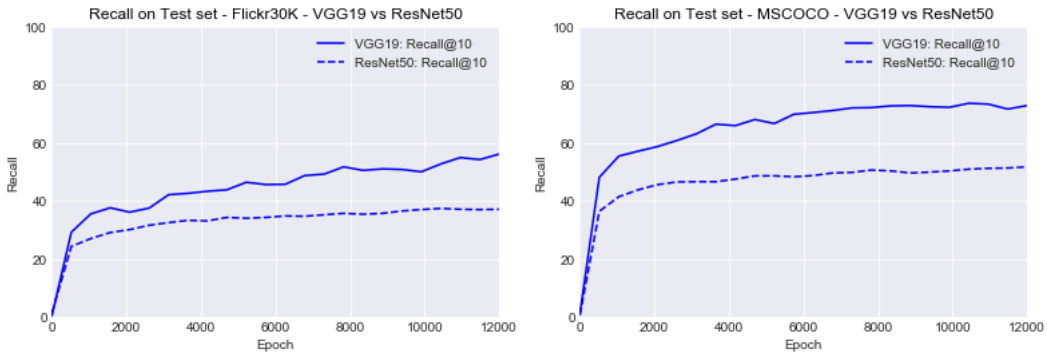
4.4.1 Effect of image encoding

For evaluating the impact of different image encoders on our model, we compare VGG19 feature (4096 dimensional) based results to ResNet50 feature (512 dimensional) based results. Figure 4a and Figure 4b show how the Recall@10 on test split varies during training using both image encoders on Flickr30K and MSCOCO respectively. On the Image-Sentence retrieval task, the average performance gap in Recall@10 using VGG19 features instead of ResNet50 is about 20%. We observe a similar improvement on Sentence-Image retrieval task as well.



(a) Flickr30k (b) MSCOCO

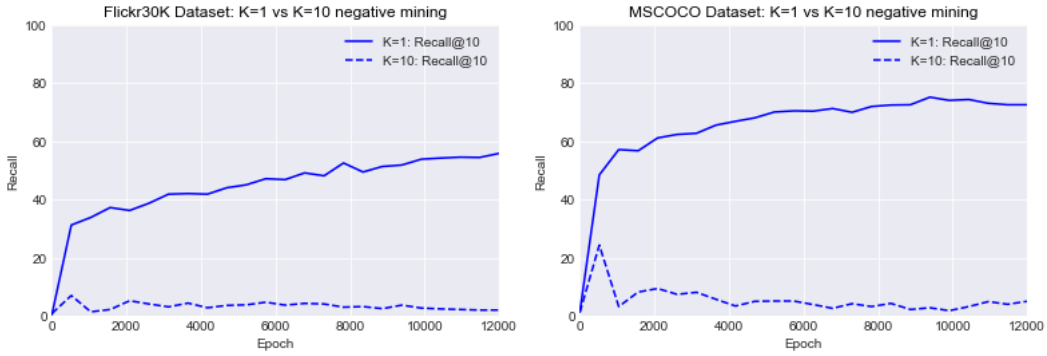
Figure 3: Recall on validation split during training



(a) Flickr30k (b) MSCOCO

Figure 4: Recall during training for VGG19 and ResNet50 image encoder

This improvement in the performance shows how the representational capacity of joint embeddings increases, when a more powerful image encoder is used. Though we have shown how the performance varies on test split during training, we have selected snapshot of the model based only on performance on validation split and reported results on test split using the same snapshot. Besides this performance gap, we find similar trends using either image encoder on how model performance varies due to variations in either training dataset(size) or triplet sampling strategies.



(a) Flickr30k (b) MSCOCO

Figure 5: Recall during training for $K = 1$ and $K = 10$ negative mining

4.4.2 Effect of triplet sampling

To reiterate our approach of selecting triplets, first we randomly select a minibatch of pairs of query image and matching text. Then for each pair, we calculate the similarity score of the non-matching text in the same minibatch (or negative examples) to the pair and rank them in the order of decreasing similarity. We select either one $K = 1$ negative or multiple $K = 10$ negatives and compute triplet ranking loss, that provides gradient update information during training.

In our experiments, we use a minibatch of size $B = 64$ and margin, $m = 0.1$. So, for $K = 1$ negative mining, the margin based loss computation depends on 64 triplets and the loss value varies from 0 to 6.4. When using $K = 10$ negative samples, the loss computation depends on 640 samples for minibatch and its value is in the range of 0 to 64. Figure 5a and Figure 5b shows how the Recall@10 varies on the test split during training using both $K = 1$ and $K = 10$ hard negative samples.

We notice that using $K = 10$, the training process starts off strong, but performs significantly poorly as training progress. Our interpretation is that, since we have multiple negatives at each training step; First the model might change to a "new" mapping that pushes these "old" negatives out, but ends up with "new" negatives. In the next step while pushing these "new" negatives out, the model might restore to the "old" mapping and oscillate between these two mappings, thereby creating a local minima in the loss function. Since we did not spend considerable time tuning hyper parameters for multiple negative sampling, it is entirely possible that the results using $K = 10$ negative samples could be much better than what we report here.

4.5 Error Analysis

To understand the merits and shortcomings of the model, we perform qualitative analysis of our results. Let us first restate the task we are solving in this work in the context of test set: We have a test set of 1000 images and each image has exactly 5 matching sentences. For a given query image among the test set, we would like to retrieve the matching sentence among a total of 5000 candidate sentences. This is done, first by computing feature representation of sentences using the 300-dimensional GloVe [21] embedding of all words of each sentence and then mapping these text embeddings to joint embedding space. It is to be noted that our model just views the sentences as bag of words without any dependency information.

The query image is encoded to an image embedding and then mapped into joint embedding space using our model. In the joint embedding space, we search for the nearest neighbor to this query image among the candidate text embeddings, which in this case is a total of 5000. If the text retrieved is one of the corresponding 5 matching sentences of the query image, then our task is successful.

Figure 6: An example of image-to-text retrieval error: query image, its 5 matching sentences and 2 sentences retrieved by our model that correspond to nearest embeddings in joint embedding space. Semantically similar words in ground truth and in retrieved results is highlighted in color.

Two examples of query image for which the model was not able to retrieve the matching result among the top 2 results are shown in Figure 6 and 7. For the query image in Figure 6, the model is trying to

Ground Truth

Five men , uniformly dressed in white shirts , tie and black slacks converse at the back of an open van .
small group of 5 white males in white suits hanging out by the back of a van in a parking lot talking .
Five men in white short-sleeved shirts and ties stand in a parking lot .
A group of men wearing ties talk while standing on the street .
Colleges stop to take a break from there business meetings .

Result # 1: Two men , one in a green shirt , one in a white shirt , are building and stacking wooden crates .

Result # 2: A man with a white beard is writing in a book while two women look on .

Figure 7: An example of image-to-text retrieval error: the query image, its 5 matching sentences and 2 sentences retrieved by our model that correspond to nearest embeddings in joint embedding space. Semantically similar words in ground truth and in retrieved results is highlighted in color.

find a similar sentence to the ground truth i.e **A band is playing in front of an audience and the singer is wearing an orange shirt and has tattoos on his arm.** Given how we are encoding text embeddings, the model only views this sentence as Bag-of-Words with no dependency information i.e *a, band, is, playing, in, front, of, an, audience, and, the, singer, is, wearing, an, orange, shirt, and has, tattoos, on, his, arm*

Due to the representation capacity of Glove embeddings, visually discriminate words such as *band, playing, front, audience, singer, orange, wearing, shirt, tattoo, arm* might have higher magnitude at the expense of more common words such as *a, is, in, of, an, and, the, is, an, and, has, on, his* and hence the model is looking for sentences that have essentially these words. If we look at the top match the model retrieved i.e **A small boy raising his hand and standing among a lot of orange pumpkins.**, we observe a similarity between the words in the ground truth and words in the retrieved result as following:

$$orange \iff orange, \quad arm \iff hand, \quad standing \iff audience,$$

Though the model notices the *orange* in the target sentence, it failed to understand the fact that *orange* is dependent on *shirt* in the ground truth, while *orange* is dependent on *pumpkin* in the retrieved result.

Similarly, for the query image in Figure 7, we observe a semantic similarity between the ground truth i.e. **Five men in white short-sleeved shirts and ties stand in a parking lot.** and in the top result i.e **Two men , one in a green shirt , one in a white shirt , are building and stacking wooden crates.** in the following words:

$$five \iff two, \quad men \iff men, \quad white \iff white, \quad shirts \iff shirt$$

Clearly, using Bag-of-words approach, which does not take structure into account is where the model is severely limited and might explain the relatively poor performance to state-of-art methods. But even with this limited representation of text features, the model has performed well on the task of finding sentences that have words which are semantically similar to the words in ground truth. Adding dependency parsing features or using networks that encode temporal information and thus encode sentences better, is going to help immensely making the model's performance better.

5 Conclusion and Future work

In this work, we implemented two-branch neural network based architectures for learning the semantic similarity between visual data and text data and validated them on Flickr30K and MSCOCO datasets for image-sentence retrieval task. We conducted numerical experiments to quantify the effect of employing different image encoder and several negative mining strategies on the retrieval task. Future efforts could explore application of RNN/LSTM based text encoding to capture the dependency information on learning image-text joint embedding as demonstrated in section 4.5 on where our model is limited. Due to resources and time constraints, we were only able to explore limited set of parameters and quantify their impact on the model performance, which could be the focus of future work as well.

Acknowledgments

We would like to thank Prof. Christopher Manning as well as Abigail See for their excellent instruction and our mentor Sahil Chopra for providing valuable feedback and help in the completion of this project.

Code

The code is posted here. link. Other than using pre-computed VGG19/ResNet50/GloVe features, we have implemented the architecture as well as loss function with several triplet selection strategies ourselves. The code is structured such that it is easy to extend our work to experiment with all triplet selection techniques and conduct parametric studies.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[2] Liwei Wang, Yin Li, and Svetlana Lazebnik. Learning two-branch neural networks for image-text matching tasks. *CoRR*, abs/1704.03470, 2017.

[3] Fartash Faghri, David J. Fleet, Ryan Kiros, and Sanja Fidler. VSE++: improved visual-semantic embeddings. *CoRR*, abs/1707.05612, 2017.

[4] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):664–676, April 2017.

[5] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: visual question answering. *CoRR*, abs/1505.00468, 2015.

[6] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[7] B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2641–2649, Dec 2015.

[8] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO captions: Data collection and evaluation server. *CoRR*, abs/1504.00325, 2015.

[9] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering - 1A_089.pdf.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[12] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12):2639–2664, Dec 2004.

[13] Yunchao Gong, Qifa Ke, Michael Isard, and Svetlana Lazebnik. A multi-view embedding space for modeling internet images, tags, and their semantics. *CoRR*, abs/1212.4522, 2012.

[14] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *International conference on machine learning*, pages 1247–1255, 2013.

[15] Fei Yan and Krystian Mikolajczyk. Deep correlation for matching images and text. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3441–3450, 2015.

[16] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*, pages 2121–2129, 2013.

[17] Li Fei-Fei, Jia Deng, and Kai Li. Imagenet: Constructing a large-scale image database. *Journal of Vision*, 9(8):1037–1037, 2009.

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

[19] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of ACL*, 2018.

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org, 2015.

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

[22] François Chollet et al. Keras. `https://keras.io`, 2015.