UNIVERSITY OF CALIFORNIA SAN DIEGO

Relational Inductive Biases for Visual-Semantic Embeddings

A thesis submitted in partial satisfaction of the
requirements for the degree of Master of Science

in

Computer Science

by

Zhiyao Yan

Committee in charge:

Professor Zhuowen Tu, Chair
Professor David Kriegman
Professor Hao Su

2019

The Thesis of Zhiyao Yan is approved and is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California San Diego

2019

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT OF THE THESIS

Relational Inductive Biases for Visual-Semantic Embeddings

by

Zhiyao Yan

Master of Science in Computer Science

University of California San Diego, 2019

Professor Zhuowen Tu, Chair

Neural networks have been shown effective at learning rich low-dimensional representations of high-dimensional data such as images and text. There has also been many recent works using neural networks to learn a common embedding between data of different modes, specifically between images and textual descriptions, a task commonly referred to as learning visual-semantic embeddings. This is typically achieved using a separate encoder for images and text and a contrastive loss. Inspired by recent works in relational reasoning and graph neural networks, this work studies the effects of using a relational inductive bias on the quality of learned visual-semantic embeddings. Training and evaluation is done using caption-to-image and image-to-caption retrieval on the MS-COCO dataset.

Introduction

The recent progress in deep learning has led to state of the art result across tasks in multiple domains including but not limited to image understanding, image generation, machine translation, speech synthesis, and speech recognition. A common approach in many of these tasks is to use a neural network to map high-dimensional data to a low-dimensional representation in the form of an embedding. This work studies the the application of using neural networks to embed data from different modes, specifically those of image and text domains, to a common space. The idea here is that an image and text with a common underlying semantic meaning should each be mapped to an embedding such that they are similar according to a metric, such as cosine similarity or Euclidean distance. An example would be that a image of a dog running in a park and the textual description a dog is running in a park should be mapped in such a way their embeddings have a high similarity as measured by cosine distance.

A shared embedding space has applications in many multimodal tasks, including image and caption retrieval, caption generation, image synthesis, and visual question answering. While typical visual-semantic embedding models rely on using a convolutional neural network (CNN) to embed images and a recurrent neural network (RNN) to embed captions, we explore the effects of using a structural prior that more explicitly captures relational information, specifically the Transformer encoder architecture. In this work we analyze the effects of a relational inductive bias and evaluate on the MS-COCO dataset for the image and caption retrieval tasks. Experimental results show that this choice of architecture results in small improvement over the baseline architecture.

Chapter 1

Background

1.1 Convolutional Neural Networks

Convolutional neural networks (also known as ConvNets or CNNs) [13] are currently the foundation of modern computer vision, achieving state of the art results on multitude of vision tasks such as object recognition [1, 2], object detection [14, 15], instance segmentation [16], and image generation [17, 18]. In the case of object recognition, the input to the network is an RGB image and the output is a probability distribution over class labels. Convolutional neural networks, like most neural networks, are trained using gradient descent and mini-batching.

Convolutional neural networks, at a high level, work by stacking multiple convolutional layers where each layer searches for specific patterns in the input and outputs a higher level, and generally lower dimension, representation. Each layer has a number of kernels (sometimes referred to as filters) which act as a pattern matching sliding window over the input. Kernels are typically of dimension $n \times n$ where n is odd and much smaller than the size of the input image, such as 3×3 or 7×7. The dimension of the output can be further reduced using pooling methods such as max-pooling, or by using strided convolutions. An analogy is often made convolutional neural networks and the ventral stream of the visual cortex system.

Here, "low" level features refer to the features in the pipeline closer to the input, and more "raw." Correspondingly, "high" level features refer to those features closer to the output, and therefore the most "abstract." In the case of object recognition, the lowest level representation

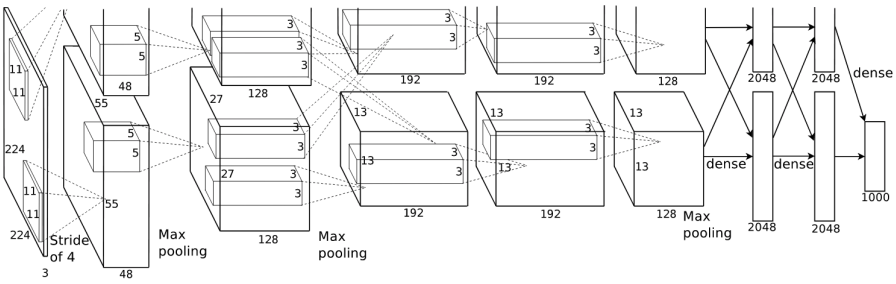

Figure 1.1. The AlexNet convolutional neural network architecture. [1].

would be the raw pixel values of the input image and the highest level representation would be the class distribution output.

It has been observed that the learned features from CNNs trained for object recognition on ImageNet [19] are rich and provide good representations for downstream tasks, enabling transfer learning for vision based tasks. While training a model on ImageNet is computationally expensive and can take multiple days to train, the final learned weights of the network can be saved and fine-tuned for downstream tasks. For downstream tasks that don't require fine-tuning, a trained CNN can be used to precompute image features which are saved and then loaded for downstream tasks. This approach does not require the downstream models to load a CNN during training, resulting in lower memory usage and faster training times.

1.1.1 Residual Networks

Many variants of convolutional neural networks have been proposed over the years, however Residual Networks (ResNet) [2] have proven to be one of the most successful. As networks became deeper (more layers), they also became harder to train. Residual networks addressed this by introducing residual connections which are essentially identity shortcut connections in the form of $\mathcal{F}(x)+x$ where x are the input features and $\mathcal{F}(\cdot)$ is some weight layers such as a convolutional layer (see Figure 1.2).

While CNN architectures have been proposed since ResNet which perform better, ResNet still remain a standard architecture used in many vision tasks. Pretrained ResNet weights are

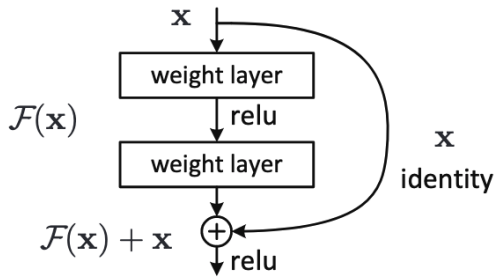

Figure 1.2. Residual connection [2].

readily available in all standard deep learning frameworks which makes for ease of integration and experimentation with new models.

1.2 Recurrent Neural Networks

While convolutional neural networks are the go-to architecture for vision tasks, recurrent neural networks (RNNs) have been vastly popular for modeling sequential data, such as text or time-series data. At a high level, recurrent neural networks operate by taking inputs from a sequence over multiple time steps while updating a hidden state over multiple time steps. Recurrent neural networks may also output some value at every time step based on the hidden state at that time step. More concretely, for a sequence of inputs $\mathbf{x}_1, ..., \mathbf{x}_T$, the hidden state update and output is defined as

$$\mathbf{h}_t = \sigma_h(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1})$$

$$\mathbf{y_t} = \sigma_y(\mathbf{W}_y \mathbf{h}_t)$$

where $\mathbf{h}_t$ is the hidden state at step t, $\mathbf{y}_t$ is the output at step t, $\mathbf{W}_h, \mathbf{U}, \mathbf{W}_y$ are learnable weights, and σ_h, σ_y are activation functions. Unlike the hidden state, the weights are shared across all time steps. Recurrent neural networks are trained using backpropagation through time, where the network is "unrolled" and standard backpropagation is applied.

One issue that standard recurrent neural networks face is that the network fails to capture

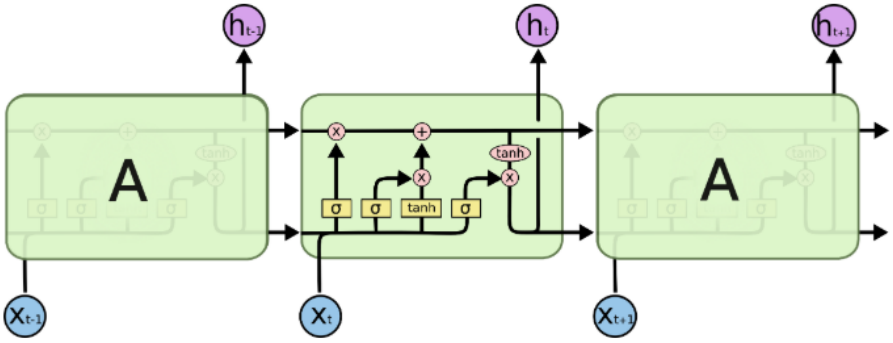

Figure 1.3. Diagram of an LSTM cell [3].

long range dependencies due to the vanishing gradient problem [20]. Variants of recurrent neural networks, such as long short-term memory (LSTM) [21] networks and gated recurrent units (GRU) [22], address this by adding a gating mechanism. The update rules for a GRU are defined as

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1})$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1})$$

$$\mathbf{h}'_t = \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{r}_t \odot \mathbf{U}\mathbf{h}_{t-1})$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \mathbf{h}'_t$$

where $\mathbf{z}_t$ is the update gate, $\mathbf{r}_t$ is the reset gate, $\mathbf{h}'_t$ is the candidate hidden state, $\mathbf{h}_t$ is the hidden state at step t, σ is the sigmoid function, and $\odot$ denotes element-wise multiplication. Also like in a standard recurrent neural network, $\mathbf{W}_z, \mathbf{W}_r, \mathbf{W}, \mathbf{U}_z, \mathbf{U}_r, \mathbf{U}$ are learnable weights. The update and reset gate allows the network to "store" information over more time steps, addressing the vanishing gradient problem.

As such, LSTMs and GRUs are standard architectures used for embedding. For example, sentence embeddings can be generated using the final hidden state of an recurrent model. Further extensions can be made, such feeding in data bidirectionally and maintaining a forward and backward hidden state.

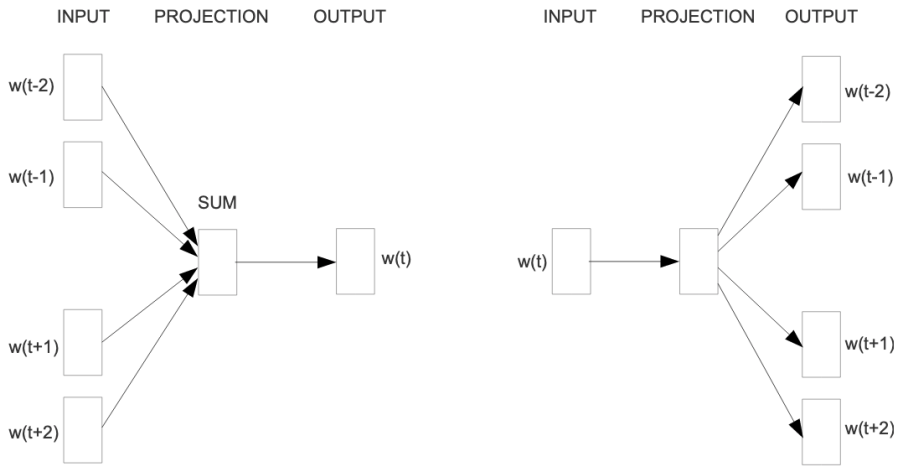

Figure 1.4. Word2vec continuous-bag-of-words (left) and continuous skip-gram (right) architectures [4].

1.3 Word Embeddings

Unlike in vision, where image features for transfer learning are typically learned in a supervised manner, rich word embeddings can be learned in an unsupervised manner. Over the years many approaches to unsupervised, and more recently referred to as self-supervised, learning of word embeddings have been proposed directly from text [4, 23, 24, 25, 26].

The goal is to learn a d-dimensional embedding vector for each word in a corpus. The approach taken is to mask out words in text and have a neural network predict the masked word using the surrounding words. The idea here is that words in language are highly contextual and a lot of information of the words can be extracted from the context. For example, given the phrase "the _____ is barking" a person is likely to guess the masked word is "dog" or some dog-related word. By learning to predict words, neural networks also learn embeddings that capture semantic meaning in relation to each other. One example of this is where Mikolov et al. [4] demonstrate, using their embedding function ϕ, that $\phi(\text{“}king\text{”}) - \phi(\text{“}man\text{”}) + \phi(\text{“}woman\text{”})$ is closest in distance to $\phi(\text{“}queen\text{”})$.

A woman and a dog sit on a park bench.
A woman is sitting on a bench with a dog.
The dog and her owner are sitting on the bench.
This is a woman sitting on a bench with a dog.
A woman pets a dog on a white bench.

Figure 1.5. An image and its corresponding captions from MS-COCO used to train visual-semantic embedding models. [5]

1.3.1 word2vec

Mikolov et al. [4] were the first to demonstrate word embeddings can be learned in a self-supervised manner. They introduced two approaches, 1) continuous bag-of-words (CBOW) and 2) continuous skip-gram. The continuous bag-of-words model takes the embeddings of neighboring words as input, averages them, and predicts the masked word. The skip-gram model takes a word as input and tries to predict the surrounding words. Mikolov et al. [4] found that, empirically, the skip-gram model resulted in better embeddings when trained on large amounts of data.

1.4 Visual-semantic embeddings

1.4.1 Problem Statement

As stated previously, convolutional neural networks are currently the foundation of computer vision. It has also been observed that the intermediate from CNNs trained for image classification are rich and provide good representations for downstream tasks, enabling transfer learning for vision based tasks [27].

Similarly, in natural language processing, unsupervised methods for learning word embeddings such as word2vec [4] and GloVe [23] have shown to be useful for downstream language tasks. These embeddings provide good token-level initializations for inputs to recurrent neural networks such as LSTMs and GRUs, which have been successfully utilized for modeling

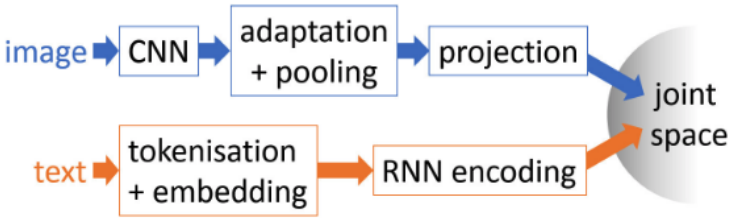

Figure 1.6. An example two-path network for visual-semantic embeddings [6].

sequential data such as text.

There has also been growing interest in building models which can learned a common representation for these two domains. This can take the form of many different tasks, such as caption generation from images [28] or visual question answering [29]. In this work we focus on learning visual-semantic embeddings [30], where our model should be able to embed semantically similar images and text close together in their embedding space.

As such, methods for learning visual-semantic embeddings typically take the form of a two-path network, one path for embedding an image and one path for embedding text, joined by a loss function. These approaches use a pretrained CNN to embed images and an RNN with pretrained word embeddings to embed text. Because paired images and captions are composed of multiple entities with no explicit mappings between them, learning visual-semantic embeddings has been posed as a weakly supervised problem [31]. Methods have also been proposed [7, 8] to add intermediate loss functions which help learn an alignment between the entities of the two modes which leads to most robust embeddings and prevent overfitting.

Let $X = \{(x_{v,i}, x_{u,i}) \mid i = 1, \ldots, N\}$ be the given data consisting of matching image x_v and caption x_u pairs. Then

$$\mathbf{v} = f_v(x_v; \theta_v)$$

$$\mathbf{u} = f_u(x_u; \theta_u)$$

where $\mathbf{v} \in \mathbb{R}^d$ and $\mathbf{u} \in \mathbb{R}^d$ are d-dimensional visual and caption embeddings, respectively. f_v and

f_u are the embedding models parameterized by θ_v and θ_u respectively.

Because the data comes in matching image-caption pairs, models are also typically trained using a triplet loss [30] which pulls matching (positive) image-caption embedding pairs closer together and pushes mismatched (negative) image-caption embedding pairs away from each other. With some abuse of notation, for every image embedding $\mathbf{v}$ there is a positive caption embeddings $\mathbf{u}^+$ and one or multiple negative caption embeddings $\mathbf{u}^-$. Similarly, this applies for caption embeddings. This loss is in the form of

$$\sum_{\mathbf{v}} |\alpha + s(\mathbf{v},\mathbf{u}^-) - s(\mathbf{v},\mathbf{u}^+)|_+ \; + \sum_{\mathbf{u}} |\alpha + s(\mathbf{v}^-,\mathbf{u}) - s(\mathbf{v}^+,\mathbf{u})|_+$$

where hyperparameter α is the margin, $s(\cdot)$ is some similarity measure *e.g.* cosine similarity, and $|x|_+ = \max(x,0)$. Intuitively, the margin α encourages the model to learn to embed matching image-caption pairs to more similar embeddings than mismatching pairs, while $|x|_+$ allows the model to ignore embeddings already with good embeddings and instead focus on poorly performing embeddings.

The quality of the learned embeddings is then evaluated on image and caption retrieval tasks. For a given set of image and caption pairs, retrieval is performed by first embedding either an image or caption, measuring its similarity with every embedding of the other mode, and checking if its pair is among the top K embeddings. The performance is measured as recall within the top K, or R@K.

It is important to note that visual-semantic embedding models are capable of embedding images or captions independently, that is, without "peeking" at the opposite domain. This is different than having a model which takes an image-caption pair and simply outputs a similarity score. Also, because retrieval can potentially be performed across a large database, the similarity score between embeddings should be able to be computed efficiently.

Many approaches to visual-semantic embeddings have been proposed, several of which are introduced below.

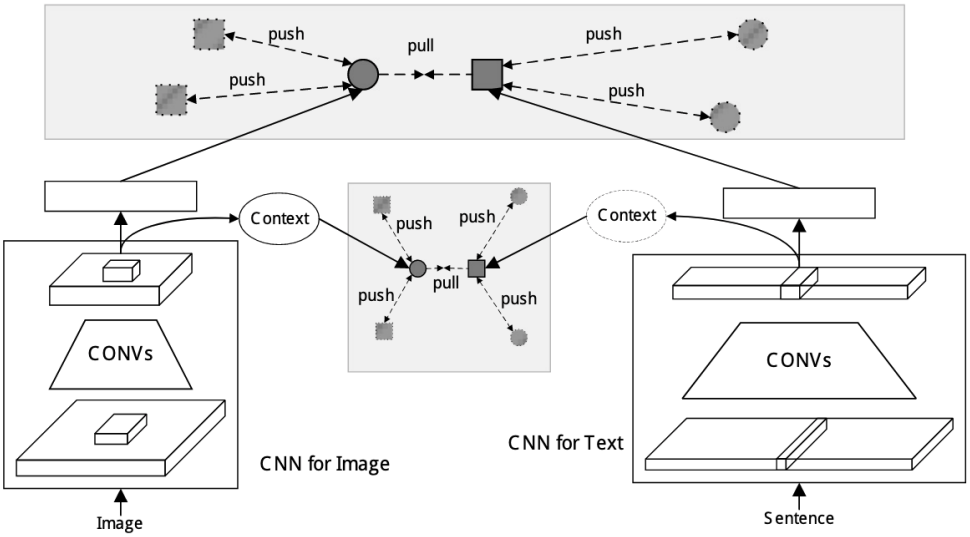

Figure 1.7. Architecture of the convolutional semantic embedding model [7].

1.4.2 VSE++: Hard Negative Mining

Faghri et al. [32] modifies the triplet loss to only sample hard negatives as negative embeddings, a technique inspired by hard negative mining for classification [33]. For a image embeddings $\mathbf{v}$, the hardest negative caption embedding would be $\mathbf{u}^- = \text{argmax}_{\mathbf{u} \neq \mathbf{u}^+} s(\mathbf{v}, \mathbf{u})$, and vice versa for caption embeddings. In order to reduce the search time, hard negative samples are computed only from each batch rather than from the entire dataset.

Faghri et al. [32] claim that the summing over many easier negative samples can result in the easier samples drowning out the hard samples in the loss. This would negatively impact the R@K retrieval score which is focused on being able to distinguish the positive sample from hard negative samples.

The authors experimented with using VGG19 [34] and ResNet-152 [2] architectures pretrained on ImageNet to embed images, and a GRU with random token initializations to embed captions. They find that using ResNet-152 with fine-tuning results in best performance. Their simple architecture using hard negative mining was able to beat state-of-the-art models at the time by a substantial amount.

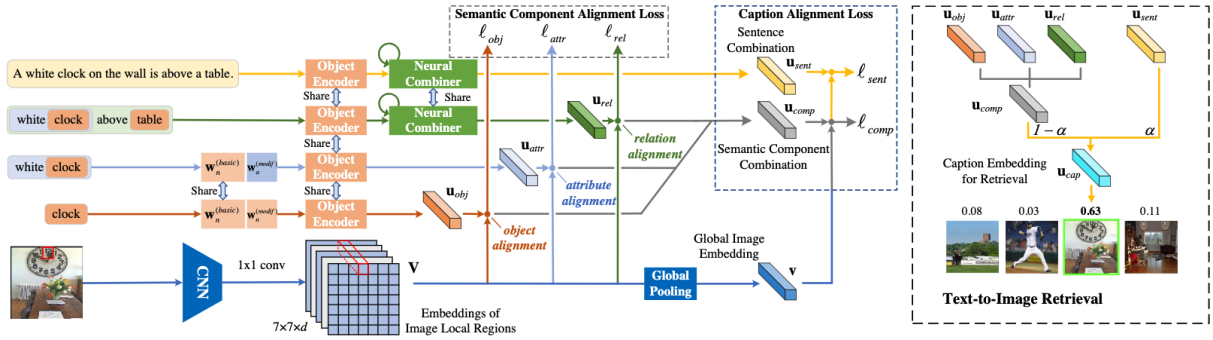

Figure 1.8. (Left) Architecture of UniVSE. (Right) Final embedding used for retrieval [8].

1.4.3 CSE: Convolutional Semantic Embeddings

You et al. [7] replaced the standard recurrent model used for caption embedding with a convolutional highway network [35]. They and also introduces an intermediate loss function to help guide the learning of intermediate convolutional features by aligning them with a global representation of the opposite domain. You et al. [7] also trained using the triplet loss across all negative samples in a batch rather than use hard negative mining. The authors did not fine-tune the ResNet during training.

1.4.4 UniVSE: Unified Visual Semantic Embeddings

Wu et al. [8] improved upon VSE++ by enforcing coverage of semantically significant components of the caption. They do this by first extracting semantic components (nouns, adjectives, and relationship pairs) from the captions using a syntax parser [36]. They then introduce loss term enforce similarity of the image embeddings with embeddings of the semantic components in addition the caption embedding. This approach leads to not only increases in image and caption retrieval, but also improved robustness against adversarial attacks.

This approach addresses the issue where human-labeled captions are usually biased [37] which models learn to exploit and result in overfitting, an issue that is also prevalent in visual question answering datasets [29]. Enforcing semantic coverage here acts as a form of regularization which reduces the amount the model can exploit these biases. The semantic

components represent the most relevant components of both the image and text, so enforcing coverage of those objects in the image features acts as additional supervision.

In a way, extracting semantic components using a syntax parser can be thought of as analogous to generating bounding boxes using an object detection network. One thing to note, however, is that the syntax parser is rule based [36] and designed to generate scene graphs [38].

1.4.5 SCAN: Stacked Cross Attention Network

Lee et al. [39] propose a cross-modal attention mechanism which aligns entities in the image with entities in the caption, and entities in the caption with entities in the image. They take a bottom-up approach [40] to generate entities by using a Faster R-CNN [15] object detection network pretrained on visual genome [38], and generate caption entities using a bidirectional GRU.

Given a set of visual features $\{v_1, ..., v_k\}, v_i \in \mathbb{R}^d$ and textual features $\{u_1, ..., u_n\}, u_i \in \mathbb{R}^d$, image-text stacked cross attention is computed as

$$
\begin{aligned}
s_{ij} &= \frac{v_i^T u_j}{\|v_i\|\|u_j\|} \\
\bar{s}_{ij} &= \frac{\max(s_{ij}, 0)}{\sqrt{\sum_i \max(s_{ij}, 0)^2}} \\
\alpha_{ij} &= \frac{\exp(\lambda_1 \bar{s}_{ij})}{\sum_m \exp(\lambda_1 \bar{s}_{im})} \\
a_i &= \sum_j \alpha_{ij} u_j \\
r_i &= \frac{v_i^T a_i}{\|v_i\|\|a_i\|}
\end{aligned}
$$

where s_{ij} is a similarity between visual and textual features, $\bar{s}_{ij}$ is normalized s_{ij}, a_i is an attention-based aggregation of relevant textual features, λ_1 is inversed softmax temperature and r_i is the similarity between the aggregated textual features. Finally the image-test similarity for

image I and caption T is computed using Log-SumExp (LSE) or averaging.

$$S_{LSE}(I,T) = \log(\sum_i \exp(\lambda_2 r_i))^{1/\lambda_2}$$
$$S_{AVG}(I,T) = \frac{1}{k}\sum_i r_i$$

A similar process is used calculate the text-image similarity. Lee et al. [39] use a triplet loss with hard negatives

$$|\gamma - S(I,T) + S(I,T^-)|_+ + |\gamma - S(I,T) + S(I^-,T)|_+$$

where γ is the margin, T^-, I^- are hard negatives, and $|x|_+ = \max(x,0)$, and $S(\cdot,\cdot)$ is some choice of similarity function.

One important distinction in this work is that it assigns images and captions a *set* of embeddings rather than learn a mapping to a single $\mathbb{R}^d$ embedding. Therefore the learned representations are much higher dimensional and calculating similarity scores is computationally expensive. It is also important to note that this approach relies on additional pretrained object detection network which requires training on a supervised dataset.

1.5 Relational Architectures

Recently there has been growing interest in relational inductive biases when designing neural network architectures [41]. We have seen that the local spatial inductive bias in convolutional neural networks, in the form of kernels, has been effective for learning visual features. Similarly, the recurrent inductive bias of recurrent neural networks has been successful in modeling sequential data. In this section we introduce some use cases and relational inductive biases.

In this work we apply the principle of explicitly modeling relationships between different entities for visual-semantic embeddings. Images can be thought of as a set of visual objects interacting with one another. These objects can be determined using a object detector [15] or by

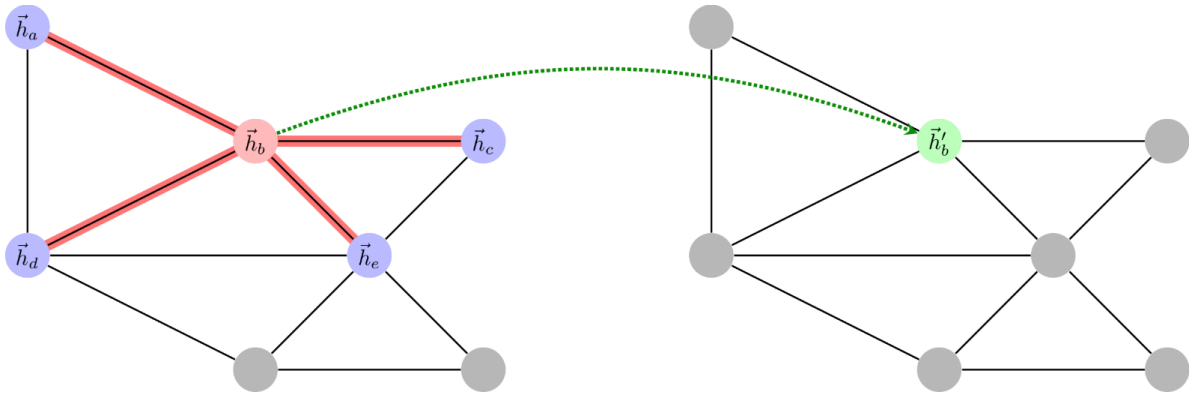

Figure 1.9. Convolution on a node in a graph by considering the features of its neighbors [9].

dividing the image into a regular grid and treating each region as an object [11]. Likewise, the individual words in a caption can be thought of as different objects, with relationships between words forming the overall semantic meaning. From this perspective, we hope that employing an architecture that more explicitly models relationships between entities could improve the quality of embeddings.

1.5.1 Graph Neural Networks

Neural network architectures like convolutional neural networks and recurrent neural networks are designed take advantage of the inherent structure present in the data they take in. Images are a regular 2D grid and CNN kernels take advantage of this spatial structure. Text and time-series data are sequential which RNNs are designed to process. However there are types of data, specifically graphs, that lack such structure which can be exploited by these architectures. Unlike in images or text, there is no inherent ordering to the nodes or edges of a graph. As such, graph neural networks (GNN) [42] such as graph convolutional networks (GCN) [43] and graph attention networks (GAT) [44] have been proposed to tackle these sorts of data.

Graph data, such as scene graphs [38], molecular structures, and social networks come in the general format $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{A})$ where $\mathbf{V}$ is a set of N nodes v_i, $\mathbf{E}$ is a set of edges $e_{i,j} = (v_i, v_j)$, and $\mathbf{A}$ is an $N \times N$ adjacency matrix. The graph may also have features $\mathbf{X} \in \mathbb{R}^{N \times D}$ where each

node v_i has an associated feature $\mathbf{x}_i \in \mathbb{R}^D$. Generally, layers in graph neural networks operate by aggregating the neighboring node features for each node

$$\mathbf{x}'_i = g(\{f(\mathbf{x}_j) \mid j \in \mathcal{N}_i\})$$

where $\mathcal{N}_i$ denotes the neighbors of v_i, $f(\cdot)$ is a mapping function, and $g(\cdot)$ is a symmetric aggregation function such as mean or max. In a sense, each layer is propagating a node's information one "hop" away to neighboring nodes. Using this very general formulation, it can be argued that popular architectures are specific cases of graph neural networks. Convolutional neural networks are graph networks operating on graphs with a regular spatial structure, PointNet [45] is a graph network operating on graphs with no edges, and Transformers are graph networks operating on a fully connected graph.

Tasks involving graph data include semi-supervised classification of unlabeled nodes, graph classification, and graph matching. For a more complete survey of graph neural networks, please refer to Wu et al. [42].

Graph neural networks have been shown to be effecting at modeling relational data [41], a property this work attempts to leverage for improving visual semantic embeddings. Some examples include using a object detection network to create a set of objects which are treated as nodes of graph [46] as input to a graph neural network.

However, despite formulating images and textual data as graphs by using image regions and words as nodes, this approach still lacks a formulation for an adjacency matrix. The solution we use in this work is to simply treat the nodes as if they were connected to every other node; so an edge e_{ij} exists for every pair of nodes (v_i, v_j). We hope that even without explicitly specifying the which node pairs are more important, our choice in architecture can implicitly find the connections between nodes.

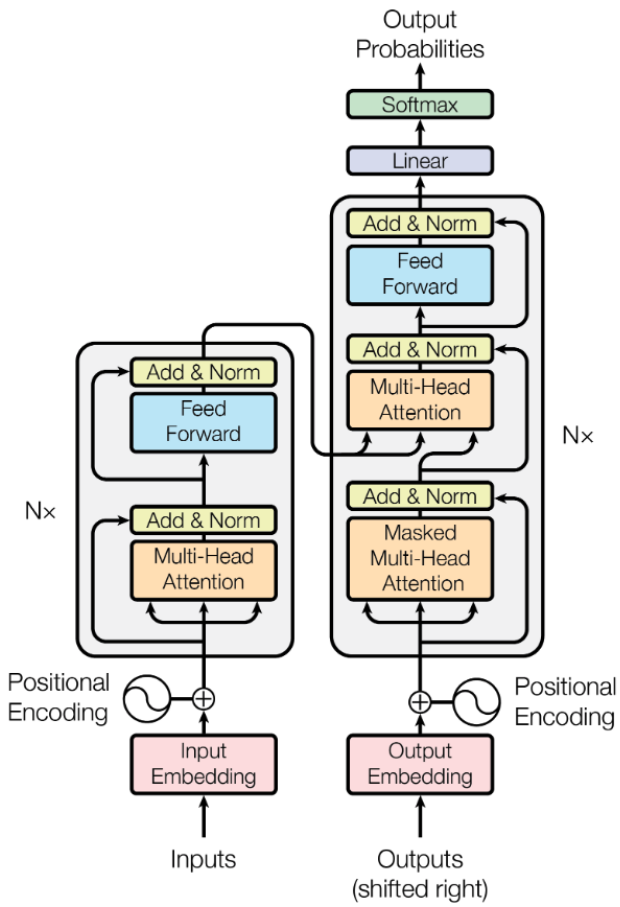

Figure 1.10. The transformer architecture [10].

1.5.2 Transformers

Vaswani et al. [10] proposed Transformers, an attention-based feedforward architecture that has proven to be hugely successful in natural language processing [47, 26]. Rather than process tokens sequentially like in recurrent models, Transformers are able to process tokens parallel manner. Transformers have been shown to outperform recurrent models on a multitude of natural language processing benchmarks.

Transformers operate on each token by "looking" at all tokens from the input, essentially contextualizing the tokens. In a way, it works like a graph neural network operating on a graph with a fully connected adjacency matrix. In this project we attempt to leverage the relational nature of the Transformer encoder for visual-semantic embeddings by applying it to both image

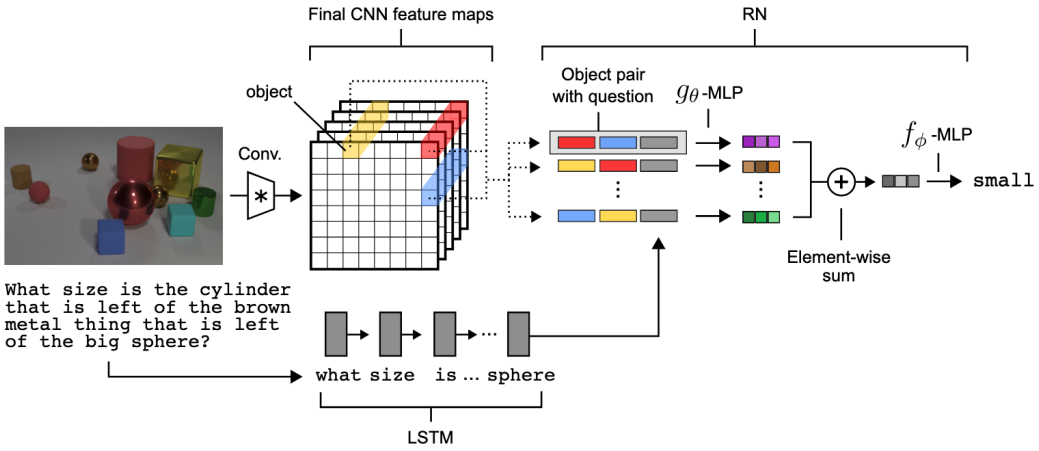

Figure 1.11. Architecture of a Relation Network [11] for visual question answering on the CLEVR [12] dataset. The CNN feature map is used as a grid of objects.

and textual data. We present more details of the implementation in the methods section.

1.5.3 Relation Networks

Relation networks [11] have been shown to be an simple yet effective approach to tackling relational reasoning in visual question answering. Santoro et al. [11] first divide the output feature map $\mathbf{V} \in \mathbb{R}^{h \times w \times d}$ as a set of $h \times w$ objects $\{v_1, ..., v_{h \times w}\}, v_i \in \mathbb{R}^d$. They then consider every pair of object relationships, in a simplified format, as

$$\mathrm{RN}(\mathbf{V}) = f_\phi \left(\sum_{i,j} g_\theta(v_i, v_j) \right)$$

where f_ϕ, g_θ are feedforward neural networks. Intuitively, g_θ is a function which outputs a relationship between two objects v_i and v_j and all relationships are aggregated using summation, which is a symmetric function.

This simple approach is able to learn various relational reasoning tasks such as positional reasoning and counting objects on the CLEVR [12] dataset whereas previous VQA models had poor performance. It is important to note that the CLEVR consists of renders of 3D objects, as opposed to natural images. Also calculating the relationships for all object pairs is

computationally expensive and has high memory usage. Later techniques were introduced to reduce the number of object pairs to consider by utilizing hard attention [48].

Chapter 2

Methods

We adopt a two-path network in this work. Our visual-semantic embedding model takes an image x_v and caption x_u pair and embeds them as $\mathbf{v} = f_v(x_v; \theta_v)$ and $\mathbf{u} = f_u(x_u; \theta_u)$, where f_v and f_u are differentiable functions that are trained using a triplet loss function with hard negative samples. The image embedding path consists of a pretrained ResNet-152 followed by a Transformer encoder. The caption embedding path consists of word representations initialized from pretrained word embeddings and a Transformer encoder. We follow the approaches of Wu et al. [8] where the ResNet-152 component is not fine-tuned on the visual-semantic embedding task. We acknowledge that fine-tuning the ResNet does result in substantial improvements in retrieval performance, but instead we want the experiments to focus on the effects of the relational architecture.

This main contributions of this work is that 1) we explore the effects of a relational architecture, the Transformer encoder, for embeddings and 2) intermediate losses that encourage relational alignment between images and captions.

2.1 Transformer Encoder

While Transformers were originally proposed to handle textual data, we repurpose it in this work to be domain-agnostic and operate on sets of entities [49, 50] from both images and text. Transformers consist of multiple encoder layers and decoder layers, however we only use

the encoder layers in this work.

We first present the Transformer encoder architecture, which is used in both the image and caption encoders. The Transformer encoder consists of layers which can be stacked. A layer consists of two sublayers: 1) a self-attention layer and 2) a feedforward layer.

The intuition behind self-attention is that each input entity can attend to all other entities by computing an attention map over all entities followed by an aggregation step. This idea can be further extended by introducing multi-head attention, where each head has its own set of randomly initialized weights in hope the heads will focus on different relational aspects.

Let the input to the self-attention layer be a set of entities $\mathbf{x}_1, ..., \mathbf{x}_n$ where $\mathbf{x}_i \in \mathbb{R}^d$ which can be represented as $\mathbf{X} \in \mathbb{R}^{n \times d}$. We first apply layer normalization [51] to the inputs of each layer. Then we compute query, key, and value components for the h-th attention head as

$$\mathbf{Q}_h = \mathbf{X}\mathbf{W}_h^Q,\ \mathbf{K}_h = \mathbf{X}\mathbf{W}_h^K,\ \mathbf{V}_h = \mathbf{X}\mathbf{W}_h^V$$

where $\mathbf{W}_h^Q, \mathbf{W}_h^K, \mathbf{W}_h^V \in \mathbb{R}^{d \times d_k}$ and $\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h \in \mathbb{R}^{n \times d_k}$. Intuitively, each input entity $\mathbf{x}_i$ has a corresponding query vector $\mathbf{q}_i$, which is used to "query" all other entities based on their key vector $\mathbf{k}_j$ using dot product similarity $\mathbf{q}_i \cdot \mathbf{k}_j$. This is then used to calculate an attention score a_{ij} used to aggregate values $\mathbf{z}_i = \sum_j a_{ij}\mathbf{v}_j$. Self-attention can be computed for all entities as

$$\mathbf{Z}_h = \text{softmax}(\frac{\mathbf{Q}_h\mathbf{K}_h^T}{\sqrt{d_k}})\mathbf{V}_h$$

where $\frac{1}{\sqrt{d_k}}$ is a scaling factor to prevent small gradients. The full multi-head attention is then computed as

$$\mathbf{Z} = \mathbf{X} + \text{Concat}(\mathbf{Z}_1, ..., \mathbf{Z}_H)\mathbf{W}^O$$

where $\mathbf{W}^O \in \mathbb{R}^{d \times d}$ and H is the number of attention heads. The final output of a layer is then computed as

$$\mathbf{X}' = \mathbf{X} + \text{Feedforward}(\text{LayerNorm}(\mathbf{Z})).$$

So far all computation steps are order invariant, that is, the order of input entities does not effect their respective outputs. We can permute the ordering of input entities $\mathbf{x}_1, ..., \mathbf{x}_n$ in any manner and their respective outputs will remain the same. This is due to the symmetric nature of the softmax aggregation step. Invariance to the order of inputs is a key property when processing graph and set inputs [49] where there is no inherent order to the inputs, which is why Transformers also work as a graph neural network.

However, we don't want to completely discard ordering information when dealing with image and textual inputs. Spatial information in images (trees *below* a sky) and ordering of words in text (green trees and a blue sky) determine their overall semantic meaning. Otherwise our final embedding could reduce to a bag-of-words approach which only considers the presence of entities and not their overall arrangement *e.g.* an image of trees below a sky and an image of trees above a sky could have the same embedding.

To address this, Vaswani et al. [10] add a positional encoding to input tokens which helps model sequential data. These positional embeddings can be either hand-crafted or learned. In our work we allow the model to learn these positional encodings rather than hand-crafting them. We combine the input entities $\mathbf{x}_i$ with the positional embeddings $\mathbf{p}_i \in \mathbb{R}^d$ using element-wise addition $\mathbf{x}_i + \mathbf{p}_i$.

2.2 Image Encoder

While images are composed of pixels at the low level, they can be understood at a higher level as an arrangement of discrete object with various inter-object relationships. Methods such as object detection and object segmentation have been proposed to extract these objects [39, 31, 46]. Various approaches have been proposed to utilize object detection for state-of-the-art results on visual-semantic embeddings. However, object detection networks require supervised training on specialized datasets, and the detected objects are limited to the ones present in training datasets.

Instead, we take approach used by Santoro et al. [11] in their Relation Networks for

visual question answering. Rather than use explicit bounding boxes from an object detection network, we divide the image using a regular grid and treat each region as an object. While this approach results in noisier representation of objects, it does not require an additional object detection network and is not limited to the object classes present in object detection training data.

The image encoder f_u consists of a pretrained ResNet-152 and a Transformer encoder [10]. We choose ResNet-152 as our CNN architecture because, as observed by Faghri et al. [32], it results in higher quality embeddings than alternative networks like VGG19. Like in previous approaches, we use ResNet-152 network pretrained on ImageNet and subsequently has its weights frozen so there is no fine tuning on the visual-semantic embedding task. All input images are scaled to 224×224 and normalized before feeding into network. No data augmentation such as random cropping is applied to the images. We then use the feature map $\mathbf{V}_{cnn} \in \mathbb{R}^{7\times7\times2048}$ from the final convolutional layer of the ResNet-152 as input to the Transformer encoder. The feature map $\mathbf{V}_{cnn}$ is treated as 7×7 objects represented by 2048-dimensional features, which can be interpreted as either a set of objects or nodes in a graph. The feature map is reshaped and further processed by a linear mapping resulting in $\mathbf{V}_{obj} \in \mathbb{R}^{(7\times7)\times d}$, which is then passed as the input to a single Transformer encoder layer. Finally, we can then obtain the final image embedding $\mathbf{v}$ by pooling the outputs of the Transformer encoder using a symmetric pooling function, such as mean or max.

Images are composed of objects, object attributes, and relationships between different objects. [8] demonstrated that semantic coverage of relational object pairs improves the quality of image embeddings. By using a Transformer encoder, as opposed to additional convolutional layers or pooling, we hope the pairwise attention mechanism is better able to capture the relational nature of the image in the embedding. Convolutions are able to capture relationships to an extent, but are restricted by the size of its receptive field, while simple pooling methods such may simply discard relational information.

This approach is similar to the approaches seen in Lee et al. [39] and Datta et al. [31] except here we do not use a object detector such as Faster R-CNN [15] which needs to be trained

in a supervised manner. We use the feature map as objects rather than bounding boxes from a region proposal network.

Following Faghri et al. [32], we also normalize all embeddings as $\frac{\mathbf{v}}{\|\mathbf{v}\|}$ to be an unit vector for efficient calculation of cosine similarity.

2.3 Caption Encoder

We start by initializing the word embedding weights from 300-dimensional fastText word embeddings [25] trained on an English Wikipedia corpus [52]. We pass treat each word as an entity. Because the ordering of words in text plays a crucial role in its overall semantic meaning, we also add positional encodings to the input (otherwise "blue sky and green grass" and "green sky and blue grass" would be semantically the same). The input to the Transformer encoder is then $\mathbf{U} \in \mathbb{R}^{l \times 300}$ where l is the length of the caption. We use two Transformer encoder layers here followed by a symmetric pooling function. One thing to note is that the we do not use a residual connection for the first self-attention sublayer due to mismatching dimensions. Therefore the first sublayer is instead

$$\mathbf{Z} = \text{Concat}(\mathbf{Z}_1, ..., \mathbf{Z}_k)\mathbf{W}^O$$

where $\mathbf{Z} \in \mathbb{R}^{l \times 1024}$.

Because our model is trained using batches and captions are variable length, we have the pad the inputs such that they contain the same number of words as the max sequence length of the batch. In order to prevent our model from overfitting by exploiting caption length and to learn more robust positional encodings, we randomly add padding to the beginning of the shorter captions. We also truncate the captions to contain at most 48 characters.

Like with the image encoder, we hope that the attention mechanism of the Transformer encoder can better capture inter-token dependencies than a recurrent model.

Also like with the image encoder, we normalize all embeddings as $\frac{\mathbf{u}}{\|\mathbf{u}\|}$ to be an unit

vector for efficient calculation of cosine similarity.

2.4 Hard Negative Mining

We employ hard negative mining as our global loss function. Following Faghri et al. [32], which achieved very impressive performance using a very simple image and caption embedding architecture, we use hardest negatives in our triplet loss where the hardest negative samples are found in each batch.

Because the image and caption embeddings are already normalized to be unit vectors, we can compute cosine similarity between vectors by their inner product. So $s(\mathbf{v},\mathbf{u}) = \mathbf{v}^T\mathbf{u}$. Our global loss is then

$$\mathcal{L}_{global} = \sum_{\mathbf{v}} |\alpha + s(\mathbf{v},\mathbf{u}^-) - s(\mathbf{v},\mathbf{u}^+)|_+ \; + \; \sum_{\mathbf{u}} |\alpha + s(\mathbf{v}^-,\mathbf{u}) - s(\mathbf{v}^+,\mathbf{u})|_+$$

where $\mathbf{u}^- = \text{argmax}_{\mathbf{u}\neq\mathbf{u}^+} s(\mathbf{v},\mathbf{u})$ and $\mathbf{v}^- = \text{argmax}_{\mathbf{v}\neq\mathbf{v}^+} s(\mathbf{v},\mathbf{u})$ within the embeddings of each batch. Because the similarity $s(\mathbf{v},\mathbf{u})$ is inner product, the similarity scores for the entire batch can be computed efficiently using matrix multiplication as

$$\mathbf{S} = \mathbf{V}\mathbf{U}^T$$

where $\mathbf{V} = [\mathbf{v}_1,...,\mathbf{v}_{49}]^T$, $\mathbf{U} = [\mathbf{u}_1,...,\mathbf{v}_l]^T$, and $\mathbf{S}$ is a matrix of all pairwise scores $\mathbf{S}_{ij} = s(\mathbf{v}_i,\mathbf{u}_j)$.

2.5 Structural Alignment

You et al. [7], Wu et al. [8] have demonstrated that an additional loss term which enforces alignment of the intermediate features between the two domains can lead to better embeddings. Similarly, Zanfir and Sminchisescu [53] compare intermediate nodes between graphs to guide the learning of graph embeddings.

Inspired by these works, we experiment with multiple intermediate loss terms that enforce

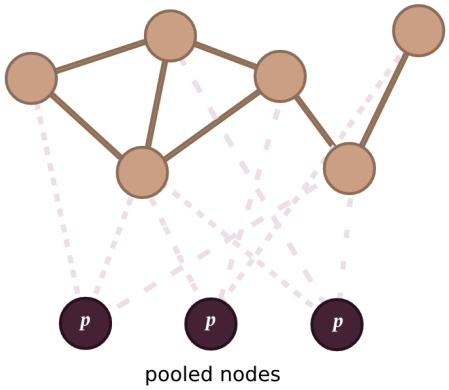

Figure 2.1. Node pooling.

structural alignment of between the image and caption intermediate features. Because the visual and caption inputs to the Transformer encoder can be interpreted as a graph, our approach to learning visual-semantic embeddings can also be framed as a graph embedding task. We treat the output of the image Transformer encoder $\{\mathbf{v}'_1, ..., \mathbf{v}'_I\}$ and caption Transformer encoder $\{\mathbf{u}'_1, ..., \mathbf{u}'_{49}\}$ as nodes of a graph.

We experiment with two different methods 1) node pooling and 2) node cross attention.

2.5.1 Node pooling

Because the number of entities in an image (fixed at 49) and caption (the number of words in the caption) are different and in particular order, there is no obvious way to directly compare the nodes of the two graphs. Here we take the approach of pooling the nodes into a fixed number P of nodes $\{\mathbf{p}_1, ..., \mathbf{p}_P\}$, an approach similar to the Induced Set Attention Block proposed by Lee et al. [50] and node pooling layer proposed by Gao and Ji [54]. We will refer to these new nodes $\mathbf{p}_i$ as *pools*. Each pool has learnable weights $\mathbf{w}_i$ and aggregates the input nodes using an attention mechanism. For an input set of nodes $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$ we can compute the pool

node as

$$\alpha_{ij} = \frac{\exp(\mathbf{w_i}^T \mathbf{x_j})}{\sum_k \exp(\mathbf{w_i}^T \mathbf{x_k})}$$
$$\mathbf{p}_i = \sum_j \alpha_{ij} \mathbf{x}_j$$

where α is softmax attention which ensures the magnitude of the pooled vector is similar to the magnitude of the input vectors.

We now enforce alignment between the corresponding pooled image nodes $\mathbf{p}_{vi}$ and pooled caption nodes $\mathbf{p}_{ui}$ using an intermediate hard negative mining triplet loss

$$\mathcal{L}_{pool} = \sum_i |\gamma_{pool} + s(\mathbf{p}_{vi}, \mathbf{p}_{ui}^-) - s(\mathbf{v}_{vi}, \mathbf{p}_{ui}^+)|_+ \; + \sum_{\mathbf{u}} |\gamma + s(\mathbf{p}_{vi}^-, \mathbf{p}_{ui}) - s(\mathbf{p}_{vi}^+, \mathbf{p}_{ui})|_+$$

where γ_{pool} is the margin and the corresponding negative pool nodes $\mathbf{p}_{vi}^-, \mathbf{p}_{ui}^-$ are those of the hardest negative determined by the global loss. Like with the global loss, we use cosine similarity as our similarity function s.

2.5.2 Node cross attention

Rather than pool nodes to build a correspondence between the image and text nodes, we can use cross attention (CA) to match nodes across domain in a very similar manner to what was proposed by Lee et al. [39].

Given a set of visual nodes $\{\mathbf{v}_1, ..., \mathbf{v}_m\}$ and caption nodes $\{\mathbf{u}_1, ..., \mathbf{u}_n\}, u_i \in \mathbb{R}^d$, we can normalize them and find corresponding caption nodes $\{\mathbf{v}'_1, ..., \mathbf{v}'_m\}$ as

$$s(\mathbf{v}_i, \mathbf{u}_j) = \mathbf{v}_i^T \mathbf{u}_j$$
$$\alpha_{ij} = \frac{\exp(s(\mathbf{v}_i, \mathbf{u}_j))}{\sum_k \exp(s(\mathbf{v}_i, \mathbf{u}_k))}$$
$$\mathbf{v}'_i = \sum_j \alpha_{ij} \mathbf{u}_j$$

where $s(\cdot,\cdot)$ measures similarity between two vectors, α_{ij} is softmax attention, and $\mathbf{v}'_i$ is an attention-based aggregation of relevant textual features. Similarly, we can find corresponding visual nodes as

$$\begin{aligned} s(\mathbf{u}_i, \mathbf{v}_j) &= \mathbf{u}_i^T \mathbf{v}_j \\ \alpha_{ij} &= \frac{\exp(s(\mathbf{u}_i, \mathbf{v}_j))}{\sum_k \exp(s(\mathbf{u}_i, \mathbf{v}_k))} \\ \mathbf{u}'_i &= \sum_j \alpha_{ij} \mathbf{v}_j \ . \end{aligned}$$

Now we can employ a intermediate loss to guide the learning of corresponding features between the graphs as

$$\mathcal{L}_{ca} = \sum_i |\gamma_{ca} + s(\mathbf{v}_i, \mathbf{v}'^{-}_i) - s(\mathbf{v}_i, \mathbf{v}'_i)|_+ \ + \ \sum_{\mathbf{u}} |\gamma + s(\mathbf{u}_i, \mathbf{u}'^{-}_i) - s(\mathbf{u}_i, \mathbf{u}'_i)|_+$$

where γ_{ca} is the margin, $s(\cdot,\cdot)$ is inner product, and $\mathbf{v}'^{-}_i, \mathbf{u}'^{-}_i$ are negative corresponding nodes determined by the hard negatives of the global loss.

Chapter 3

Experiments

In this section we describe our implementation details and experimental results.

3.1 Implementation

We implement our model in PyTorch [55]. We use a ResNet-152 model pretrained on ImageNet from the torchvision [56] package. The pretrained model is reported to achieve 94.06% top-5 accuracy and 78.31% top-1 accuracy on ImageNet. We also use fastText word embeddings [25] trained on an English Wikipedia corpus [52].

3.2 Dataset

We evaluate the quality of our learned embeddings through image retrieval and caption retrieval on the MS-COCO dataset [5]. The data is split into 118,287 training images and 5000 testing images. Karpathy and Li [30] and Faghri et al. [32] then split this into 113,287 training images, 5000 validation images, and 5000 testing images. Each image has at least 5 corresponding captions, so for images with more than 5 captions we randomly select only 5 to use. Evaluation on the validation and testing is done on a 1000 image subset or the full 5000 images. So a 1000 image subset would contain 5000 captions, and the full 5000 image split contains 25000 captions. We train on the original 118,287 training image split of MS-COCO and validate using the 5000 testing images.

Table 3.1. MS-COCO retrieval results.

Model	Caption Retrieval				Image Retrieval				Sum
	R@1	R@5	R@10	Med. R	R@1	R@5	R@10	Med. R	
1K testing split									
VSE-C [57]	48.0	81.0	89.2	2.0	39.7	72.9	83.2	2	414
sm-LSTM[58]	53.2	83.1	91.5	1.0	40.7	75.8	87.4	2	431.8
RRF-Net [59]	56.4	85.3	91.5	-	43.9	78.1	88.6	-	443.8
VSE++ [32]	58.3	86.1	93.3	1.0	43.6	77.6	87.8	2.0	446.7
CSE [7]	56.3	84.4	92.2	1.0	45.7	81.2	90.6	2.0	450.4
UniVSE [8]	64.3	89.2	94.8	1.0	48.3	81.7	91.2	2.0	469.5
Ours (basic)	53.1	84.7	92.5	1.0	44.4	77.7	87.9	2.0	440.3
Ours (pooling)	56.9	84.3	92.6	1.0	44.3	77.5	87.8	2.0	443.4
Ours (CA)	55.3	83.3	91.9	1.0	44.3	77.0	87.4	2.0	439.2

Performance is evaluated using the recall using the top K scoring retrievals R@K, where $K = 1, 5, 10$. So for caption-to-image retrieval with 1000 images, a caption embedding is used to query the 1000 image embeddings and a "hit" is considered if the corresponding image embedding is within the top K retrieved. For image-to-caption retrieval with 1000 images, an image embedding is used to query the 5000 caption embeddings and a "hit" is considered if any corresponding caption embedding is within the top K retrieved.

Also note that MS-COCO is designed around 91 object categories, therefore all images and captions contain objects of at least one of these categories.

3.3 Preprocessing Data

Because use pretrained weights for the ResNet-152 component and do not perform any fine-tuning, we can generate and save the features maps $\mathbf{V}_{cnn}$ for all images as a preprocessing step. This means we do not have to load the ResNet-152 during training, thus speeding up training and reduce memory usage.

We also preprocess all captions and remove words which do not occur in the fastTest dictionary. Most of the removed words are misspellings. We also truncate any descriptions with more than 48 words.

3.4 Training

We train on the full 118,287 training images for 20 epochs and evaluate on 1000 samples of the test set using the final learned weights.

Following previous works, we train using batch size of 128 samples and embedding size of 1024. We train all models using Adam optimizer [60] with $\beta_1 = 0.9, \beta_2 = 0.999$. We use a learning rate schedule of 0.00002 at epoch 1, 0.00001 at epoch 9, and 0.000002 at epoch 15.

We set the global margin α to 0.2. For the node pooling experiment we set the intermediate margin γ_{pool} to 0.2 and loss to be $0.2\mathcal{L}_{global} + 0.8\mathcal{L}_{p}ool$. For the the cross attention experiment we set the intermediate margin γ_{ca} to be 0.2 and loss to be $\mathcal{L}_{global} + 0.15\mathcal{L}_{ca}$.

3.5 Results

Overall, we observe that additional transformer encoders do result in a small increase in image retrieval recall over VSE++ [32] but lower caption retrieval recall. This indicates that recurrent neural networks may result in better performance over transformer encoders for caption embeddings in this task setup.

We also noticed that the intermediate losses did not result in higher performance, unlike that observed by You et al. [7] and Wu et al. [8]. However, this could be due to incorrectly tuned hyperparameters.

Street lights and a lit up street sign at an intersection

A bicycle is standing next to a bed in a room

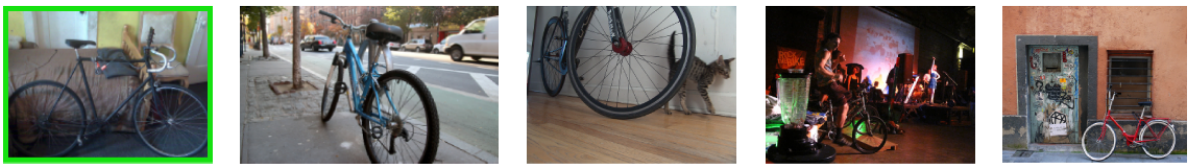

Figure 3.1. Top-5 retrieved images in the test set. The green border indicates the correct image.

Zebras and rhinos out in the wild on a sunny day

A woman taking pictures on a busy street

Figure 3.2. Failure cases for image retrieval in the test set. In the first row we see many semantically similar images. In the second row the correct image was actually rank 15.

The new motorcycle on display is very shiny
The shiny motorcycle has been put on display
A brand new motorcycle on display at a show
A blue motorcycle is displayed in a showroom
A motorcycle is parked inside of a building

Figure 3.3. Top-5 retrieved captions for an image in the test set. Bold indicates correct caption.

A living room with a couch and chair
A living room with a couch and large television
A comfortable living room with brown leather couches and a fireplace
A brightly decorated living room with a stylish feel
The dark wood and marble pillars give this living space a warm feel

Figure 3.4. Top-5 retrieved captions for an image in the test set. Bold indicates correct caption.

Several people crossing a road with one person having food in their hand
A person walking down a street past a teddy bear sign
Many people are sitting on two benches next to a light pole
A man walking down a street smoking a cigarette
A man walking down a street smoking a cigarette

Figure 3.5. Failure case example for caption retrieval in the test set. The correct caption was rank 309.

Chapter 4

Conclusion

Overall we noticed a small improvement in image retrieval recall performance through the use of transformer encoders. This could be due to the fact previous approaches use simple aggregation of image features while transformer encoders use pairwise attention. However, the use of intermediate objectives did resulted in improved caption retrieval, but no improvement in image retrieval.

4.1 Future Work

While the relational priors used in this work resulted in rather small improvements in the quality of visual-semantic embeddings, further modifications and improvements can still be made. One possible avenue is to see the affects of fine-tuning the ResNet-152 component, which would relate our work very close with Lee et al. [39].

Another question that has not been well studies is what aspects of the image and captions the embeddings in state-of-the-art models are capturing. Currently visual-semantic embedding models are evaluated almost exclusively on the MS-COCO and Flickr30k datasets [61]. MS-COCO is limited to its 91 object categories while Flickr30k has 44,518 distinct object categories. However these datasets still likely suffer from biased human-generated captions, a widespread issue observed and still being addressed in visual question answering datasets [37]. It is possible that the current visual-semantic embedding models are able to overfit to these human biases.

It is also not clear how much of the relational information is being captured in the embeddings. For example, to the best of our knowledge, no widespread study have been conducted which analyzes the extent to which embeddings capture *presence* of objects versus the *relations* between objects, and how many relations can be captured within a single embedding with respect to the dimension of the embedding. MS-COCO and Flickr30k images and captions generally focus on only a few objects each. Datasets such as Visual Genome [38] have been proposed which contains images with more densely labeled relations between objects. One possible approach is to use a dataset which explicitly tests for the relational properties of embeddings, such as the synthetic CLEVR [12] dataset which was proposed to test relational reasoning in visual question answering models.

Once again, we note that learning visual semantic embeddings is closely related to learning graph embeddings [62] with graph neural networks, a task recently gaining popularity. Therefore emerging techniques and theory for learning graph embeddings might be applicable in learning visual-semantic embeddings.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[3] Christopher Olah. Understanding lstm networks, Aug 2015. URL http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[4] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

[5] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

[6] Martin Engilberge, Louis Chevallier, Patrick Pérez, and Matthieu Cord. Finding beans in burgers: Deep semantic-visual embedding with localization. *CoRR*, abs/1804.01720, 2018. URL http://arxiv.org/abs/1804.01720.

[7] Quanzeng You, Zhengyou Zhang, and Jiebo Luo. End-to-end convolutional semantic embeddings. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[8] Hao Wu, Jiayuan Mao, Yufeng Zhang, Yuning Jiang, Lei Li, Weiwei Sun, and Wei-Ying Ma. Univse: Robust visual semantic embeddings via structured semantic representations.

CoRR, abs/1904.05521, 2019. URL http://arxiv.org/abs/1904.05521.

[9] Graph attention networks, 2017. URL http://petar-v.com/GAT/.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

[11] Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy P. Lillicrap. A simple neural network module for relational reasoning. *CoRR*, abs/1706.01427, 2017. URL http://arxiv.org/abs/1706.01427.

[12] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *CoRR*, abs/1612.06890, 2016. URL http://arxiv.org/abs/1612.06890.

[13] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio. Object recognition with gradient-based learning. In D. Forsyth, editor, *Feature Grouping*. Springer, 1999.

[14] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013. URL http://arxiv.org/abs/1311.2524.

[15] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL http://arxiv.org/abs/1506.01497.

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. URL http://arxiv.org/abs/1703.06870.

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[18] Justin Lazarow, Long Jin, and Zhuowen Tu. Introspective neural networks for generative modeling. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2793–2802, 2017.

[19] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575,

2014. URL http://arxiv.org/abs/1409.0575.

[20] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2):157–166, March 1994. ISSN 1045-9227. doi: 10.1109/72.279181. URL http://dx.doi.org/10.1109/72.279181.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8): 1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[22] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014. URL http://arxiv.org/abs/1412.3555.

[23] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

[24] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. *CoRR*, abs/1506.06726, 2015. URL http://arxiv.org/abs/1506.06726.

[25] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[26] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

[27] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. *CoRR*, abs/1403.6382, 2014. URL http://arxiv.org/abs/1403.6382.

[28] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555, 2014. URL http://arxiv.org/abs/1411.4555.

[29] 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. URL http://arxiv.org/abs/1505.00468.

[30] Andrej Karpathy and Fei-Fei Li. Deep visual-semantic alignments for generating image descriptions. *CoRR*, abs/1412.2306, 2014. URL http://arxiv.org/abs/1412.2306.

[31] Samyak Datta, Karan Sikka, Anirban Roy, Karuna Ahuja, Devi Parikh, and Ajay Divakaran. Align2ground: Weakly supervised phrase grounding guided by image-caption alignment. *CoRR*, abs/1903.11649, 2019. URL http://arxiv.org/abs/1903.11649.

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

[33] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 886–893, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2372-2. doi: 10.1109/CVPR.2005.177. URL http://dx.doi.org/10.1109/CVPR.2005.177.

[34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[35] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015. URL http://arxiv.org/abs/1505.00387.

[36] Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D. Manning. Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *Workshop on Vision and Language (VL15)*, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[37] Kushal Kafle and Christopher Kanan. Visual question answering: Datasets, algorithms, and future challenges. *CoRR*, abs/1610.01465, 2016. URL http://arxiv.org/abs/1610.01465.

[38] 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. URL https://arxiv.org/abs/1602.07332.

[39] Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. Stacked cross attention for image-text matching. *CoRR*, abs/1803.08024, 2018. URL http://arxiv.org/abs/1803.08024.

[40] 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 VQA. *CoRR*, abs/1707.07998, 2017. URL http://arxiv.org/abs/1707.07998.

[41] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris

Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018. URL http://arxiv.org/abs/1806.01261.

[42] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *CoRR*, abs/1901.00596, 2019. URL http://arxiv.org/abs/1901.00596.

[43] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. URL http://arxiv.org/abs/1609.02907.

[44] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

[45] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016. URL http://arxiv.org/abs/1612.00593.

[46] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph R-CNN for scene graph generation. *CoRR*, abs/1808.00191, 2018. URL http://arxiv.org/abs/1808.00191.

[47] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[48] Mateusz Malinowski, Carl Doersch, Adam Santoro, and Peter Battaglia. Learning visual question answering by bootstrapping hard attention. *CoRR*, abs/1808.00300, 2018. URL http://arxiv.org/abs/1808.00300.

[49] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep sets. *CoRR*, abs/1703.06114, 2017. URL http://arxiv.org/abs/1703.06114.

[50] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer. *CoRR*, abs/1810.00825, 2018. URL http://arxiv.org/abs/1810.00825.

[51] Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.

[52] Tim Finin James Mayfield Lushan Han, Abhay L. Kashyap and Johnathan Weese. UMBC_EBIQUITY-CORE: Semantic Textual Similarity Systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, June 2013.

[53] Andrei Zanfir and Cristian Sminchisescu. Deep learning of graph matching. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[54] Hongyang Gao and Shuiwang Ji. Graph u-net, 2019. URL https://openreview.net/forum?id=HJePRoAct7.

[55] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

[56] Pytorch. torchvision. https://github.com/pytorch/vision, 2019.

[57] Haoyue Shi, Jiayuan Mao, Tete Xiao, Yuning Jiang, and Jian Sun. Learning visually-grounded semantics from contrastive adversarial samples. *CoRR*, abs/1806.10348, 2018. URL http://arxiv.org/abs/1806.10348.

[58] Yan Huang, Wei Wang, and Liang Wang. Instance-aware image and sentence matching with selective multimodal LSTM. *CoRR*, abs/1611.05588, 2016. URL http://arxiv.org/abs/1611.05588.

[59] Yu Liu, Yanming Guo, Erwin M. Bakker, and Michael S. Lew. Learning a recurrent residual fusion network for multimodal matching. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[60] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[61] 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. *CoRR*, abs/1505.04870, 2015. URL http://arxiv.org/abs/1505.04870.

[62] HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques and applications. *CoRR*, abs/1709.07604, 2017. URL http://arxiv.org/abs/1709.07604.