# Graph-enhanced Transformers for Referring Expressions Comprehension

**Aditya Chinchure**
21811161
aditya10@cs.ubc.ca

## Abstract

Recent work in vision-and-language has shown the effectiveness of large-scale Transformer models for learning relationships between features in the two modalities. While this approach is promising, a major shortcoming of such models is the need for large datasets and access to expensive computational resources. The self-attention mechanism in Transformer models is tasked to learn relationships between all input tokens from scratch. However, the input tokens are rarely independent of each other; for example, two word tokens may be the same or closely related to each other semantically. In our project, we explore a simple method to incorporate these relationships in a Transformer before performing any training, using graphs with edge features. In VL-BERT-Graph, we generate a fully-connected graph of input tokens where the edges represent similarity between the tokens, obtained using GloVE and CLIP. We then use a message-passing GNN to incorporate these features into the input tokens or the output encoding of the model, and train the Transformer with edge-feature attention masks. The VL-BERT-Graph model is evaluated on the RefCOCO+ dataset, and preliminary results show that the best configuration of VL-BERT-Graph outperforms our baseline by a margin of 2% on one of the test sets. We believe that this technique can be applied, in future, to train Transformer models more efficiently. *Code:* `https://github.com/aditya10/VL-BERT-Graph`

## 1 Introduction

Learning relationships between data from different modalities, such as vision and text, is a fundamental problem in machine learning. In the recent years, Transformer [18] based models have shown impressive performance on visual-linguistic tasks such as Referring Expression Comprehension [11, 16, 9], Visual Question Answering [11, 16, 9], and image or text retrieval [11]. Moreover, CLIP [13], which utilizes contrastive learning applied to Transformer-encoded image and text data, has shown impressive zero-shot performance on several vision and visual-linguistic tasks. It is clear that massive datasets paired with extensive computational resources is required to train such state-of-the-art models.

The flexibility of Transformers in learning relationships between different data points driven by the attention mechanism that it employs. Given an input sequence, the self-attention mechanism learns to relate each input token with every other input token as well as to itself. Fundamentally, this is analogous to treating the input tokens as the nodes of a fully connected graph, and learning how the nodes interact with each other over several layers. However, unlike in graphs, a Transformer model does not take into account any predefined relationships between the input tokens, and is required to learn these relationships during training.

Motivated by the challenge of training large, data inefficient transformer models, as well as the prevalence of zero-shot visual-linguistic models like CLIP, we incorporate predefined inter-token relationships thorough a graph neural network (GNN) that assists with training an existing Transformer-based

**Referring Expression:**
guy in yellow dribbling ball

Figure 1: An example of Referring Expression Comprehension (REC) task from the RefCOCO+ dataset [7]. Given the image and the text expression, the goal is to identify the entity in the image (green box) that is described by the expression.

model, VL-BERT [16], on the Referring Expression Comprehension (REC) task [7]. REC involves identifying an object in the image that is described by a text expression. This requires a model to perform visual grounding on the words in the expression, which makes it an excellent candidate for incorporating useful visual-linguistic edge features across tokens in the two modalities. Our contributions are as follows:

1. A method for graph construction that accounts for relationships between visual and linguistic tokens.
2. A message passing GNN [4] to incorporate the graph information in the node features, which are subsequently used to enhance the input or output representations of the VL-BERT model.
3. Finally, the use of edge features to generate an attention mask that can be applied to the transformer encoder layers.

## 2 Related Work

**Efficiently training Transformers.** Several recent works have addressed the issue of training Transformer models more efficiently [17]. Notable methods include reducing the complexity of calculating attention (Reformer [8], Cluster-former [19]), using kernels (Performer [3]), and using sparse or hierarchical attention (Sparse Transformer [2], Swin Transformer [10]). While these methods are effective, our method approaches the problem differently, by assuming pre-defined relationships between input tokens, and expecting them to be useful priors for training a Transformer model. Moreover, in this project, we do show that our method is more effective than any prior methods, but rather focus on the ground-work required for future research in this area.

**Transformers for vision-language tasks.** Many state-of-the-art models for visual question answering, referring expressions, and similar tasks use the Transformer encoder. In ViLBERT [11], the authors propose to use separate encoder layers for the image and the text domains, and use co-attention layers for grounding the text in the image. Subsequently, VL-BERT [16] and Oscar [9] show that a single-stream transformer encoder can be used on both image and text tokens, with special embeddings to differentiate between the two domains. VL-BERT uses visual geometric embeddings, described in [6], to incorporate some spatial information in image region inputs to the model. In our work, we use the VL-BERT model as our baseline, but modify it significantly by introducing a GNN model to generate enhanced input and output token representations and an edge-feature based attention mask (edge mask).

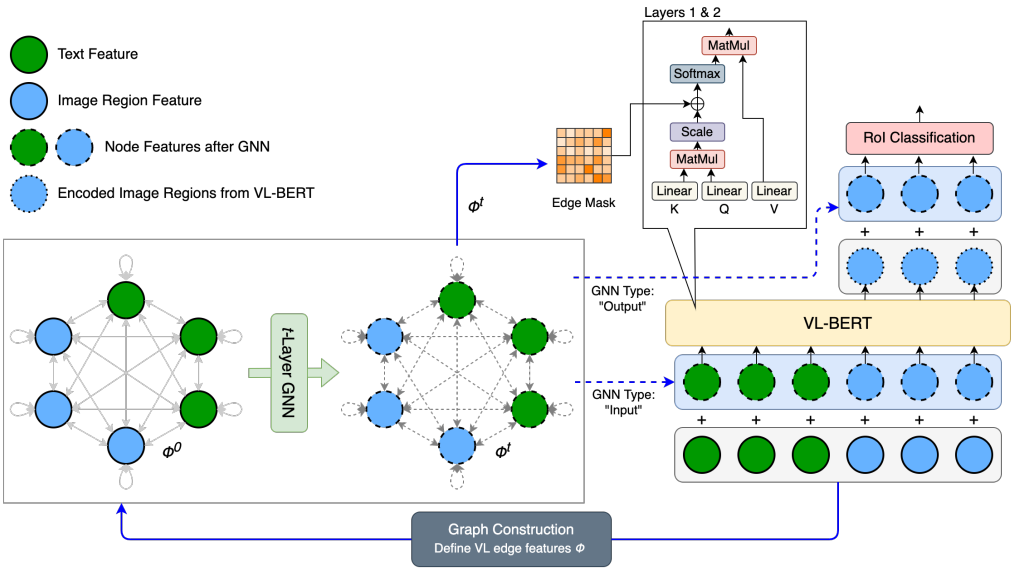


Figure 2: The VL-BERT-Graph model. First, the edge features between the input tokens is used to construct a fully connected graph. This graph undergoes GNN updates, and the final nodes are either used to enhance the input tokens (GNN Type: *input*), or as additional information for the RoI classifier head (GNN Type: *output*). Finally, the edge mask is generated using the final edge features from the GNN, and is incorporated as an attention mask in the first two layers of VL-BERT.

**Transformers for graphs.** Several efforts have been made to adapt the Transformer model to graphs. Most notably, Graphnormer [20] adapts the transformer model to graph data using simple modifications to the model, by introducing centrality, spatial, and edge encodings. The centrality encoding encapsulates the degree of each node in the input sequence to the Transformer, whereas the spatial and edge encodings mask the attention weights to assert the graph structure. Our model uses a similar approach as the spatial encoding based attention mask, but it differs from the rest of the model because we define edges using GloVE and CLIP-based relationships between input tokens, and enhance them using a GNN.

**Graph networks.** We use methods from Graph Neural Networks (GNNs) [15] and MPNN [4] in our work, for building the graph representation and performing message passing to update the node features. We incorporate aspects of the above models in our work.

## 3 Model

In this section, we briefly describe the VL-BERT model, and then explain the three main parts of the VL-BERT-Graph architecture. We walk through the construction of graphs using the edge features of choice. Then we elaborate on the GNN message passing layers. Finally, we discuss the edge mask generation component of our model.

### 3.1 VL-BERT Recap

VL-BERT [16] is a BERT-based transformer model for learning visual-linguistic representations, and is fine-tuned on Referring Expression Comprehension, Visual Question Answering, and Visual Commonsense Reasoning tasks.

We are interested in the REC task. VL-BERT consumes a sequence of input tokens consisting of text and image data. First, words from the referring expression are tokenized and embedded into a 768-dimensional feature space. For the image, a backbone Fast-RCNN [5] model is used to extract object features of 768-dimension for annotated bounding box regions from the COCO 2014 dataset. In addition to each image region (RoI), the full image is also treated as a separate

RoI and is also appended to the input token sequence. Finally, VL-BERT utilizes a combination of BERT-like positional embeddings, token type embeddings to differentiate between text and image tokens, and visual geometric embeddings, to enhance the the input sequence. In order to select the region described by the referring expression, an RoI classification MLP is used over the encoded image region features from the transformer encoder. A simplified version of VL-BERT is shown in Figure 2.

### 3.2 Constructing a Graph Representation

We construct a graph, $G = (V, E)$, where $V$ be the set of vertices and $E$ be the set of edges. Following common notations used in GNN, we denote $h_i$ as the node feature of node $v_i \in V$, and $e_{ij}$ as the edge feature of the edge between nodes $v_i$ and $v_j$, where $v_i, v_j \in V$. We assume that the graph is fully connected, and for the edge between $v_i, v_j \in V$, we assign the edge feature to be

$$e_{ij} = \phi(h_i, h_j).$$

We define $\phi$ in the following ways:

**Difference between GloVe Embeddings ($\phi_{glove}$):** Figure 3(a) shows an example of this. First, we obtain ground truth category labels from COCO for all image RoIs in the input sequence. We then use GloVe embeddings [12] to embed each word in the text expression and each word ($v$) in the category labels for the image. Accordingly, the edge feature for any pair of input tokens $h_i$ and $h_j$ is the vector difference between GloVe embeddings of $v_j$ and $v_i$.

**CLIP cosine similarity ($\phi_{clip}$):** Figure 2(a) also shows an example of this. Instead of using GloVe embeddings, we use CLIP [13] to obtain embeddings for the words in the expression, as well as for all the category labels for the RoIs. We then calculate the cosine similarity between every pair of CLIP word embeddings.[1]

In addition to the above methods for producing edge features, we also experimented with image-only methods such as RoI overlap between two bounding boxes or Euclidean distance between two RoIs. These methods were ineffective for the task, so we do not report results using them.

### 3.3 Message-Passing GNN Layers

To incorporate the information introduced in the edge encoding process into the input tokens, which form the nodes of the graph, we propose to use a GNN with message-passing layers. Concretely, for a specific node, messages are generated for all its neighbors according to:

$$\mathbf{m}_{ji}^t = MLP_1([h_j^t, h_i^t, e_{ji}]). \tag{1}$$

Where $MLP$ is short for a 2-layer multi-layer perceptron. Then the message is aggregated according to

$$\bar{\mathbf{m}}_i^t = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mathbf{m}_{ji}^t. \tag{2}$$

Finally, the node embedding is update with

$$h_i^{t+1} = MLP_2([h_i^t, \bar{\mathbf{m}}_i^t]). \tag{3}$$

For each message-passing layer, the node embeddings of all nodes are updated. Subsequently, we also update the edge embeddings in each layer as follows:

$$e_{ij}^{t+1} = MLP_3([W_s \cdot h_j^t, W_s \cdot h_i^t, e_{ij}]). \tag{4}$$

Here, $W_s$ is a learnable projection layer that shrinks the node embedding dimension so that it is comparable in size to the edge feature dimension.

After the message-passing layers, the updated node embeddings may be added to the input token embeddings (described as GNN Type: *input*), or, the nodes representing image RoIs may be added to the output image RoI tokens, right before the classification layer (described as GNN Type: *output*).

[1] Our initial goal was to implement multi-modal embeddings where CLIP can encode words in the expression, as well as each image RoI directly. However, this is extremely slow to do during training, and requires at least two GPUs to perform successfully. As this is out of the scope of our project, we leave as future work.

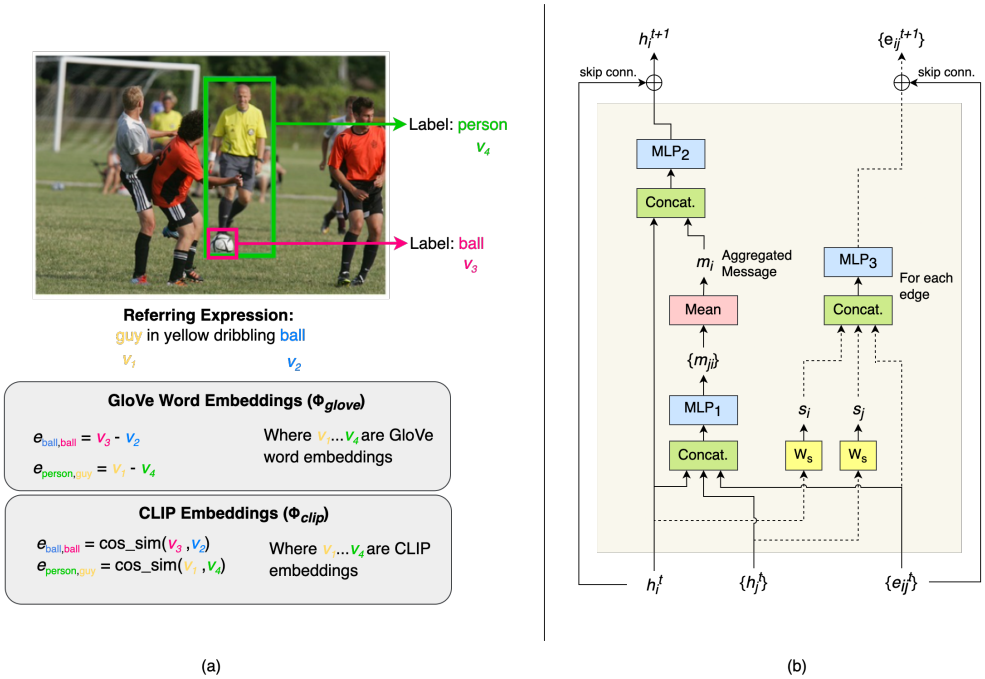


Figure 3: (a) describes the two methods by which edge features are constructed. (b) explains the workings of our Message-Passing GNN model, where, given nodes $h$ and edges $e$ at iteration $t$, we update node features using message passing, and edge features using an $MLP$.

### 3.4 Edge Mask Generation

Inspired by [20], we augment the attention scores in the first two layers of the VL-BERT transformer by adding an edge-feature attention mask to incorporate relationships between input tokens. With $A_{ij}$ as the $(i, j)$-th element of the attention matrix $A$, we compute

$$A_{ij} = \frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d}} + b_{ij}. \tag{5}$$

where

$$b_{ij} = MLP_4(e_{ij}^t). \tag{6}$$

The computation of $b_{ij}$ follows edge readout scheme in GNN. Specifically, a neural network, $MLP_3$ is used to project the final edge features to a 12-dimension feature corresponding to the 12 attention heads in the transformer.

## 4 Dataset

We use the RefCOCO+ dataset [7] in our work. RefCOCO+ differs from the original RefCOCO dataset in the sense that no location words (e.g. "left", "top of") are used in the expressions that describe objects in the image. Rather, these descriptions only contain appearance ("blue shirt") or activity words ("running") for the entities described. This is an important property of the dataset that we exploit in our work.

RefCOCO+ contains 141k textual descriptions ("referring expressions") for 50k objects in images from the COCO 2014 dataset. The dataset is split into four independent parts, a training set, a validation set, and two test sets, testA and testB. TestA contains examples where the referred entity is a person, whereas TestB contains examples where the referred entity is an object. The dataset is crowdsourced using the ReferIt Game [7].

For ease of experimentation, given limited time and resources for this project, we only use a smaller training set with 28k expressions from the training set. However, we utilize the full validation and test sets for evaluation.

Table 1: We train a new baseline with the mini dataset and fewer layers compared to the original VL-BERT paper. Results show that the *output* GNN type is more performant, and this used alongside CLIP embeddings and the edge mask give a significant 2% improvement on the testB set.

| Model | | | | | Accuracy (RefCOCO+) | | |
|---|---|---|---|---|---|---|---|
| Model Name | Param. | Edge Feat. | GNN Type | E. Mask | val | testA | testB |
| VL-BERT [16] | 155.14M | - | - | - | **74.41** | **77.28** | **67.52** |
| VL-BERT (Baseline) | 70.11M | - | - | - | 67.47 | 73.63 | 59.90 |
| VL-BERT-Graph | 71.35M | GloVe | input | - | 64.33 | 70.12 | 55.34 |
| VL-BERT-Graph | 71.35M | GloVe | input | ✓ | 64.77 | 71.49 | 56.76 |
| VL-BERT-Graph | 71.35M | GloVe | output | ✓ | **68.76** | 74.38 | 60.50 |
| VL-BERT-Graph | 71.35M | CLIP | input | - | 65.69 | 70.90 | 56.60 |
| VL-BERT-Graph | 71.35M | CLIP | input | ✓ | 66.22 | 72.25 | 58.46 |
| **VL-BERT-Graph** | 71.35M | CLIP | output | ✓ | 68.72 | **74.48** | **62.03** |

## 5 Implementation Details

As described in the method section, we extend the VL-BERT [16] model to incorporate edge features using a GNN. We utilize the provided code for VL-BERT as a starting point for our project. In our implementation, we use 4 GNN layers and 6 BERT layers, with an edge feature size of 64. Detailed hyperparamter values are provided in the GitHub repository of our project. For GloVe edge features, we use glove50D vectors. For CLIP-based edge features, we use the S-BERT wrapper for CLIP [14] because it is easy to implement and is capable of running independently of the GPU. We implement a simple caching mechanism to store generated word embeddings in a cache dictionary so that it can be referenced quickly at any later time during training. The edge mask is generated for all 12 attention heads of a transformer layer, and is applied to the first two layers of the encoder.

During training, the label is obtained by selecting the RoI bounding box that has the maximum IoU score against the ground truth bounding box. Standard binary cross-entropy loss is used on the RoI classification head in VL-BERT.

## 6 Results

Our main results are shown in Table 1. To establish a baseline model, we first train a 6-layer VL-BERT model without pre-training (only use BERT weights to initialize) on the mini dataset. Following that, we trained six VL-BERT-Graph models with the two types of edge features, changing the GNN Type, and whether or not the edge mask is used in the model.

### 6.1 Performance against baseline

Our results show that four of our models with GNN Type set to *input* do not outperform the baseline. This is likely because adding graph node embeddings from the GNN to the input sequence is effectively modifying the input token embeddings quite drastically, causing the model initialized with BERT weights to fail to learn effectively. This also highlights an important property of GNNs: over several iterations, GNNs have a smoothing effect over all the node features. This smoothing effect may make it harder for the transformer to differentiate between the input tokens. Using edge masks improves the model performance slightly, likely because the edge masks act as a stronger pathway for backward gradient flow when training the model.

In the case of *output*, we see a clear increase in performance for both edge features. In fact, this approach with CLIP edge features and the edge mask outperforms all other methods on both test sets. The advantage to this approach is two fold. One, because the output node features of the GNN are used closer to the loss calculation during training, it improves the backward gradient flow to the GNN model. Two, the input node features remain unmodified, allowing existing pre-trained weights (from BERT) to be useful from the get-go.

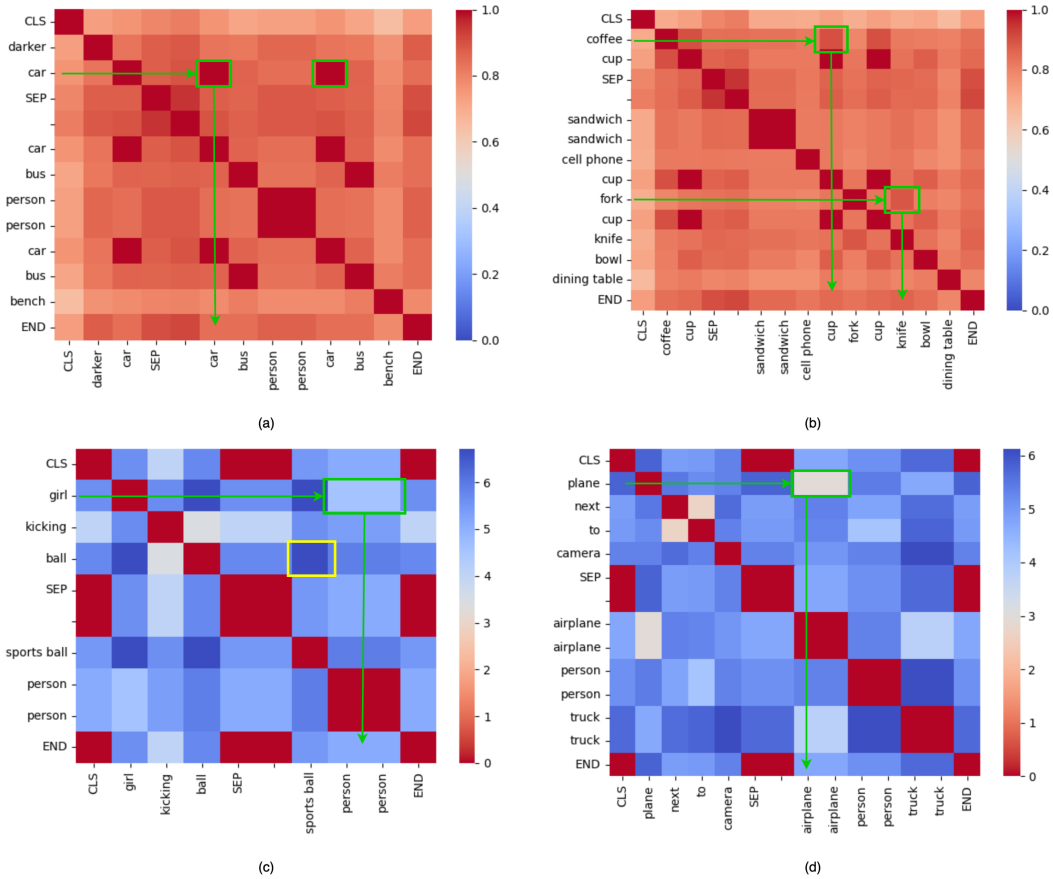


Figure 4: Edge features of our model. (a) and (b) are the cosine similarity scores using CLIP (larger value = more related), whereas (c) and (d) are euclidean norm distance between GloVe embeddings (smaller value = more related)

### 6.2 Quality of Edge Features

As described in the method section, we use two multi-modal edge features, the difference between two GloVe embeddings, and the cosine similarity between CLIP embeddings. To visualize these edge features, we plotted the CLIP cosine similarities in Figure 4 (a) and (b), and the Euclidean norm distance between GloVe embeddings, in Figure 4 (c) and (d). In (b) we note how CLIP is effective at matching the word "coffee" in the referring expression to the bounding boxes labelled "cup". In (c), we show that the distance between GloVe embeddings of the word "girl" and "person" is small, and in (d), the distance between the words "plane" and "airplane" is small. However, we also note that in (c), the distance from "ball" to "sports ball" is relatively large, and is a failure case. Nonetheless, it is important to note that during training, we consider the difference between GloVe word vectors directly rather than the euclidean distance.

In Table 1, we see that CLIP-based models outperform the GloVe based models in almost all cases. This is expected, as CLIP is far more recent, trained on much larger datasets, and is capable of generalizing to new words in a zero-shot manner.

In addition to these edge features, we initially experimented with image-only edge features such as IoU score between bounding boxes, or the union box feature between two boxes. However, these methods do not serve the task of REC well, as the goal is to learn relationships between text and image features, rather than image features themselves.

### 6.3 Other Architecture Choices

**Number of BERT Layers.** The original VL-BERT model from [16] is a 12-layer BERT transformer with over 120M parameters by itself. We train or model on the mini training set, with significantly fewer training samples. Therefore, we compared the performance of VL-BERT with 12, 8 and 6 layers on the mini set, and got validation scores of 67.9, 68.1, 67.5 respectively. Given that the training time required for the 12-layer model was 1.7x that of the 6-layer model, with only a minimal increase in performance, we chose to run experiments using the 6-layer model.

**Number of GNN Layers.** We evaluated the performance of one of our initial experimental models with 2 and 4 GNN layers and noted a performance increase of +0.6% on the validation score. From that point on, we only trained with the 4-layer GNN model. Given the time constraints of this project, we were unable to re-run our experiments with the 2-layer GNN model.

**Use of Batch Normalization.** Previous works ([1]) have shown the effectiveness of batch normalization when training graph neural networks. We tested a few models with and without batch normalization applied at the end of $MLP_2$ for generating the message, but found no performance advantage when using it.

## 7 Future Work

While we show performance improvements using the edge mask with the *output* GNN type method, we were limited by the scope of this project and the time we had in hand, therefore forcing us to use a smaller training set. We would like to run our model with VL-BERT at its full scale, with better hyperparameter tuning, giving us a more complete picture about the effectiveness of our approach. We can also incorporate our method in the pre-training process. Moreover, we would like to experiment with utilizing CLIP image embeddings, and using other methods to capture inter-token relationships.

There are several small challenges we faced in our project. One such challenge is that our node feature dimension is far larger than our edge feature dimension, leading to the edge features not being very effective. One way to tackle this challenge would be to use a different method for node and edge updates, such as the one in GatedGCN [1], where the edge features are used as gates for node updates.

Finally, we want to work towards applying this method for training more efficient transformer models. As future work, we can investigate if edge masks in the self-attention mechanism can allow us to use fewer transformer layers, while maintaining the performance against a larger model.

## 8 Conclusion

In this work, we propose a graph-enhanced transformer model, VL-BERT-Graph, to incorporate pre-existing relationships between input tokens to improve performance, and take a step towards building more efficient transformer models. We use GloVe and CLIP embeddings to define edge features, and use a message-passing GNN to enhance our input embeddings or output RoI encoding, as well as produce an edge mask for the VL-BERT transformer. Through quantitative results, we show that our method is effective at improving performance of VL-BERT on when trained on a mini subset of the RefCOCO+ dataset. Finally, through qualitative methods, we explore the quality of our edge features and explain that CLIP embeddings generally perform better.

## 9 Acknowledgement

I would like to acknowledge Xianda Sun for his contributions in the planning stage of this project, and his feedback in the later stages. Xianda contributed to the project proposal and helped with the initial implementation of our GNN model. However, due to other commitments, he could not participate in the project presentation and the final project report, and was not involved in running any experiments. Xianda is not a registered student in this class, so you may grade this as a one-person project.

## References


[1] Xavier Bresson and Thomas Laurent. *Residual Gated Graph ConvNets*. 2017. DOI: 10.48550/ARXIV.1711.07553. URL: https://arxiv.org/abs/1711.07553.

[2] Rewon Child et al. *Generating Long Sequences with Sparse Transformers*. 2019. DOI: 10.48550/ARXIV.1904.10509. URL: https://arxiv.org/abs/1904.10509.

[3] Krzysztof Choromanski et al. *Rethinking Attention with Performers*. 2020. DOI: 10.48550/ARXIV.2009.14794. URL: https://arxiv.org/abs/2009.14794.

[4] Justin Gilmer et al. *Neural Message Passing for Quantum Chemistry*. 2017. arXiv: 1704.01212 [cs.LG].

[5] Ross Girshick. *Fast R-CNN*. 2015. DOI: 10.48550/ARXIV.1504.08083. URL: https://arxiv.org/abs/1504.08083.

[6] Han Hu et al. "Relation networks for object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 3588–3597.

[7] Sahar Kazemzadeh et al. "ReferItGame: Referring to Objects in Photographs of Natural Scenes". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 787–798. DOI: 10.3115/v1/D14-1086. URL: https://aclanthology.org/D14-1086.

[8] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. *Reformer: The Efficient Transformer*. 2020. DOI: 10.48550/ARXIV.2001.04451. URL: https://arxiv.org/abs/2001.04451.

[9] Xiujun Li et al. "Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks". In: *ECCV 2020* (2020).

[10] Ze Liu et al. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. 2021. arXiv: 2103.14030 [cs.CV].

[11] Jiasen Lu et al. "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks". In: *Advances in Neural Information Processing Systems*. 2019, pp. 13–23.

[12] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[13] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. DOI: 10.48550/ARXIV.2103.00020. URL: https://arxiv.org/abs/2103.00020.

[14] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Nov. 2019. URL: https://arxiv.org/abs/1908.10084.

[15] Franco Scarselli et al. "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605.

[16] Weijie Su et al. *VL-BERT: Pre-training of Generic Visual-Linguistic Representations*. 2020. arXiv: 1908.08530 [cs.CV].

[17] Yi Tay et al. *Efficient Transformers: A Survey*. 2020. DOI: 10.48550/ARXIV.2009.06732. URL: https://arxiv.org/abs/2009.06732.

[18] Ashish Vaswani et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[19] Shuohang Wang et al. *Cluster-Former: Clustering-based Sparse Transformer for Long-Range Dependency Encoding*. 2020. DOI: 10.48550/ARXIV.2009.06097. URL: https://arxiv.org/abs/2009.06097.

[20] Chengxuan Ying et al. *Do Transformers Really Perform Bad for Graph Representation?* 2021. arXiv: 2106.05234 [cs.LG].