



Contrastive Visual and Language Learning for Visual Relationship Detection

By

Thanh Gia Tran

Thesis
Submitted to Flinders University
for the degree of Master of Science in Computer Science

College of Science and Engineering
July 13th, 2023

Table of Contents

TABLE OF CONTENTS	I
ABSTRACT	III
LIST OF FIGURES	V
LIST OF TABLES	VII
CHAPTER 1 INTRODUCTION	1
1.1 Visual understanding	3
1.2 Thesis statement	4
1.3 Thesis outlines	5
CHAPTER 2 LITERATURE REVIEW	6
2.1 Deep learning architectures	7
2.1.1 Convolutional Neural Network.....	7
2.1.2 Transformer	8
2.2 Representational learning for image and text.....	13
2.2.1 Language representational learning	13
2.2.2 Image representational learning.....	15
2.2.3 Joint language and image representational learning	17
2.3 Contrastive Representational Learning	19
2.3.1 Statistical models and Energy-based models.....	19
2.3.2 Loss Functions for Energy-Based model.....	21
2.3.3 Noise Contrastive Estimation Loss.....	23
2.4 Structured representation of a scene	28
2.4.1 Scene Graph and Visual Genome	29
2.4.2 Visual Relationship Detection	31
2.4.3 Engineering Loss Objectives for VRD	32
2.4.4 Message Passing Approaches for VRD.....	34
2.4.5 A General CNN-based Architecture for VRD.....	37
2.5 Discussion	38
CHAPTER 3 DATASETS AND EVALUATION METRIC FOR VISUAL RELATIONSHIP DETECTION	40
3.1 The long-tail problem of Visual Genome dataset.....	40
3.1.1 From Unstructured to Structured Data	41
3.1.2 Statistics of VRD and VG150.....	42
3.1.3 Tackling the long-tail problem	44
3.2 Evaluation Metrics	45
3.2.1 Precision, Recall, and F-measure.....	45
3.2.2 Precision-Recall Curve and ROC Curve	47

3.3 Discussion	50
CHAPTER 4 CONTRASTIVE VISUAL AND LANGUAGE LEARNING FOR VISUAL RELATIONSHIP DETECTION.....	51
4.1 Visual Relationship Detection	52
4.2 The VLTransE Architecture	54
4.2.1 Visual and Spatial Module	55
4.2.2 Language Module	56
4.2.3 Loss Functions	57
4.3 A Simple Contrastive Learning Architecture	61
4.3.1 Visual Module	62
4.3.2 Language Module	63
4.3.3 Contrastive Loss Module	63
4.5 Discussion	64
CHAPTER 5 EXPERIMENTAL RESULTS AND ANALYSIS	66
5.1 Implementation	66
5.1.1 Implementation details of the VLTransE architecture.....	67
5.1.2 Implementation details for a Simple Contrastive Learning Architecture	69
5.1.3 Replication details for Scene Graph Benchmarks models.....	72
5.2 Experimental Results and Analysis	72
5.2.1 VLTransE Experimental Results	72
5.3.2 Simple Contrastive Learning Architecture Experimental Results	76
5.3 Discussion	86
CHAPTER 6 CONCLUSIONS AND FUTURE WORK.....	89
BIBLIOGRAPHY	92
APPENDICES	112
Appendix A: Biased Object Detector for VRD	112
Appendix B: ROC Curves for top sixteen predicate classes	114

Abstract

The visual world is highly structured, and real-world scenes are often decomposed into multiple objects and parts of objects that interact with one another. While deep learning models have demonstrated their abilities to detect visual objects from images, they remain unable to detect higher level visual relationships that exist between these object pairs. In this work, we focus on building visual relationship detection (VRD) systems that can recognize visual relationships between objects in images. Here, we use the scene graph representations from the Visual Genome dataset, which contains objects and relationships grounded to image regions in the form of (subject, predicate, object) triples. Different from existing work which used supervised classification techniques to build VRD systems, we interpret VRD as a representational learning task and apply visual-language contrastive learning in conjunction with knowledge graph representational learning techniques to build joint visual and language embedding spaces for the VRD task. The results show that contrastive visual and language learning can improve the model’s performance on the *Recall@n* metric while penalizing its ability to generalize to rare visual relationship classes. We also show that translational knowledge graph embedding techniques can be applied to preserve the first-order hierarchical structure without penalizing the model’s overall performance on VRD. With these results, we argue that deep learning models have extra capacity to learn visual relationship concepts and structures through additional contrastive loss constraints, and further categorization of visual relationship labels can improve the final representational spaces.

Keywords: scene graph, deep learning, contextualized representational learning, contrastive learning, transfer learning

Acknowledgements

Throughout this work, I have received a great deal of support and assistance. I would first like to thank my supervisors, Prof. Paulo Santos and Prof. David Powers, whose expertise and advice were invaluable. Your insightful feedback pushed me to sharpen my writing and brought my work to a higher level.

I am also grateful for the evaluation committee, Prof. Greg Falzon and Prof. Richard Leibbrandt, for giving me feedbacks on my work and how to improve upon them.

Finally, I want to thank Maelic Neau and my colleagues for insightful discussions throughout the candidature.

List of Figures

Figure 1.1: An example of a scene graph of a man riding a horse.....	1
Figure 1.2: Left) black bear in a forest. Middle) polar bear on snow. Right) polar bear in a forest.....	3
Figure 2.1: a visualization of <i>scaled dot product attention</i> . Adapted from Vaswani et al., 2017.....	10
Figure 2.2: Visualizing the building blocks for the Transformer model. Adapted from Vaswani et al., 2017.....	11
Figure 2.3: a visualization for <i>causal self-attention layer</i> vs <i>self-attention layer</i>	12
Figure 2.4: visualization for <i>residual connection</i> . Adapted from He et al., 2015.....	12
Figure 2.5: a visualization for <i>triplet margin loss</i>	22
Figure 2.6: a visualization of the labelled visual relationship triples from an image. The above image is retrieved from the Visual Genome dataset.....	29
Figure 2.7: The general Faster R-CNN architecture for scene graph generation.....	37
Figure 3.1: Frequency counts of different objects in the (a) test and (b) train set in VRD dataset (C. Lu et al., 2016).	43
Figure 3.2: The long tail distribution of the number of subjects, predicates, and objects in the VG150 test dataset.	43
Figure 3.3: The long tail distribution of the visual relationship triples in the VG150 dataset.....	43
Figure 4.1: Overview of the proposed VLTransE architecture. Red, black, and blue colors represent subject, predicate, and object respectively.	54
Figure 4.2: A visualization for test time scoring functions for VLTransE.	59
Figure 4.3: Overview of the proposed contrastive architecture. Visualization is partially adapted from pairs Radford et al. (2021).	61
Figure 5.1: A visualization of TransE vs TransR architectural differences, where $fc(\cdot)$ is a MLP with Leaky ReLU.	67
Figure 5.2: A visualization of a Vision Transformer encoder (Adapted from Dosovitskiy et al., 2020).....	70

Figure 5.3: Learning rate schedule that warmups for 1000 steps and decays every 15,000 steps.....	71
Figure 5.4: Predicate Prediction Results on Visual Genome dataset by individual relationship class. The blue line represents the prevalence of the class labels in the dataset.....	79
Figure 5.5: Receiver operating characteristic curves for the predicate class ‘on’.	82
Figure 5.6: Receiver operating characteristic curves for the predicate class ‘lying on’.	83
Figure 5.7: A visualization of misclassified visual relationships made by the OpenCLIP model with ViT-B/32 pretrained LAION_400M.....	85

List of Tables

Table 3.1: Confusion matrix for binary classification	46
Table 4.1: Number of unique image-text pairs for two different datasets used in pre-training.	61
Table 4.2: Number of parameters for the visual encoder used in CLIP and OpenCLIP.	62
Table 5.1: The train and test split for the smaller VRD dataset and larger VG dataset ..	66
Table 5.2: comparison with state-of-the-art results on VRD test set (– means unavailable or unknown). None of the other models are replicated.	73
Table 5.3: Predicate prediction results on VRD test set for different loss functions using the Recall@n metrics.	74
Table 5.4: Tail entity prediction results on VRD test set.	75
Table 5.5: Statistics of the ViT models (Dosovitskiy et al., 2020).	76
Table 5.6: Predicate Classification Results using the Recall@1 metrics for different contrastive models on the Visual Genome dataset.	77
Table 5.7: Comparing Predicate Prediction Results on Visual Genome dataset using the recall metrics. The replicated results are slightly different from those in Han et al., (2021).	78
Table 5.8: Comparing Predicate Prediction Results on VG150 dataset with other work using the mean Recall and Informedness metric.	80
Table 5.9: Top 30 misclassified visual relationships made by the OpenCLIP model with ViT-B/32 pretrained LAION_400M.	84

Chapter 1

Introduction

Scene understanding aims not only to detect objects in real-world scenes but also to form an interpretation of how these objects are related to one another. Deep learning, when combined with recent advances in computing hardware and large-scale image-text datasets, offers a powerful set of tools that can help us tackle the perception tasks through its representational ability. Yet, using deep neural networks' representations learned from simple perception tasks such as object recognition (Dosovitskiy et al., 2020; He et al., 2015) and object detection (Lin et al., 2017; Ren et al., 2015) alone is not enough to achieve scene understanding. Instead, the problem calls for a more structural approach in representing the visual scene.

The visual world is highly compositional and structured, and these structures often manifest themselves in our daily usage of language when describing an image or a scene. For example, we can describe the image in Figure 1.1 with a natural language phrase: “the person is riding a horse”, forming a (*subject, verb, object*) semantic structure. In this work, we use such structures in visual scenes' descriptions to build systems that generate interpretable and structured representations called scene graphs. More specifically, we aim to tackle a specific subtask of scene graph generation called visual relationship detection (VRD), where the goal is to detect not only objects in the visual image but also the visual relationships that exist between these object pairs, forming multiple (*subject, predicate, object*) triples (Lu et al., 2016).

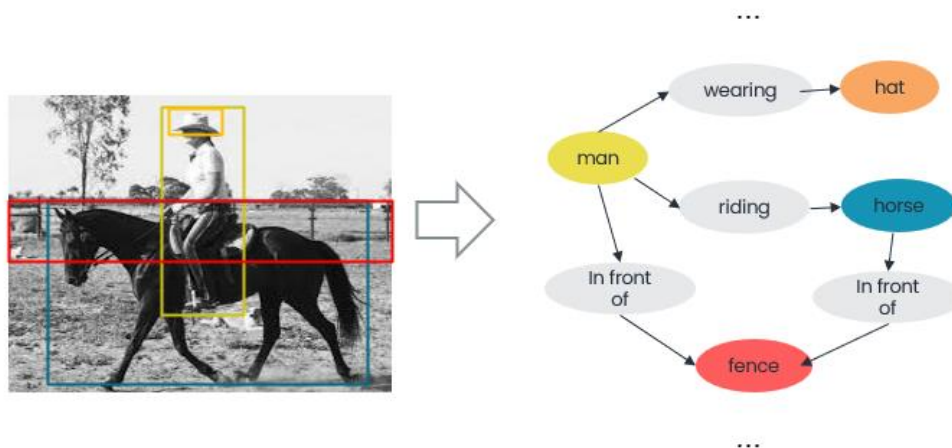


Figure 1.1: An example of a scene graph of a man riding a horse

A common approach for VRD is to build a deep neural classifier that directly predicts the *predicate* from the input image and object’s labels (Yu et al., 2017; J. Zhang et al., 2019). While this approach has shown impressive performance on benchmarking datasets (Krishna et al., 2017; Lu et al., 2016), such models are highly susceptible to dataset biases and often learn spurious correlations that do not generalize well to out-of-domain compositions. For example, a classification model may detect common relationships like (*person, ride, horse*), while struggle to detect (*person, ride, dog*) or (*dog, ride, bike*). This thesis aims to tackle part of these issues for VRD by exploiting different types of priors, including the natural language priors of the visual relation triples.

To exploit the priors from the benchmark dataset, we started by exploring the language aspect of the visual relationship datasets in Chapter 3 to analyse the labelled visual relationships in the VRD dataset benchmarks. While examining the visual relationship dataset, we highlighted how the use of English descriptions in the original data curation process gives rise to *polysemy* (Rodrigues et al., 2020), resulting in prepositions and relationships with different senses or meaning. For example, the preposition ‘*on*’ can be used in both the ‘*attach to object*’ sense such as (*wheel, on, car*) and the ‘*on top of*’ sense such as (*skateboard, on, ground*). This polysemous nature of the visual relation triples makes it difficult to evaluate the model’s performance with the commonly used Recall@n metric where only one *predicate* can be assigned to every (*subject, object*) pair. Thus, we also propose the use of one-vs-all evaluation metrics based on ROC analysis (D. Powers, 2015) for each class in Chapter 3, allowing more than one predictions for every (*subject, object*) pair.

After the analysis of the visual relationship dataset and the evaluation metrics, we apply contrastive learning to build different visual relationship detection systems, while exploiting the recent advances in contextualized encoders called the Transformer (Vaswani et al., 2017). Through the *visual-language consistency* contrastive loss function, we aim to encode both image and text onto a joint vision-language space by maximizing mutual information between the two modalities. We also incorporated a knowledge graph embedding loss called *translational loss* which enforce the first-order hierarchical structure of the (subject, predicate, object) triple ($EMB<subject> + EMB<predicate> = EMB<object>$). At test time, the model encodes the relevant image regions and text pairs to rank them using scoring distance metrics. With engineered negative sampling strategy, the model can achieve comparable Recall@n results (Chapter 5) on the VRD classification task when compared against other benchmark models.

The VRD problem can also be treated as a subset of the overarching challenge of visual understanding, described in the Section 1.1 below.

1.1 Visual understanding

Understanding visual contents from images through means of image decomposition and parsing has dated back to the 1960s-1970s (S.-C. Zhu & Mumford, 2006). In these earlier works, the task of image parsing aims to bridge the semantic gap between low-level noisy visual pixels and high-level *parse graphs*. Here, a parse graph is represented as a hierarchical tree structure that consist of scene labels, objects, parts, primitives, pixels, and functional relations (S.-C. Zhu & Mumford, 2006).

For object-level structure parsing, traditional approaches prior to deep learning era leverage human's domain knowledge and build local feature descriptors for different objects such as HOG (Dalal & Triggs, 2005), SIFT (Lowe, 2004), and Shape Context (Belongie et al., 2000). After obtaining these hand-crafted feature representations, classifiers such as the support vector machines (SVM) are often used to make the final predictions (Cortes & Vapnik, 1995). However, an object's appearance can be diverse and wide ranging, making it difficult to build local descriptors that can generalize beyond the assumed domain and fit all plausible configurations.



Removed due to
copyright restriction.

Figure 1.2: Left¹) black bear in a forest. Middle²) polar bear on snow. Right³) polar bear in a forest.

Recent innovations in deep learning have allowed us to bypass such manual feature engineering process while being successful in detecting and classifying objects

¹ Left black bear image: <https://www.ysnp.gov.tw/En/StaticPage/BSE01Copy>

² Middle polar bear image: https://en.wikipedia.org/wiki/Polar_bear

³ Right polar bear image: <https://panthertown.org/2016/04/01/polar-bear-in-panthertown-usfs-closes-hiking-trails/>

(LeCun et al., 1999). However, deep learning is not without its problems. While deep learning models are often treated as black boxes that can automatically learn structures from data and map any input to output, it is more often the case that these models learn spurious correlations and not causations (Shekhar et al., 2017; Szyk et al., 2021). For example, in Figure 1.2, if we train a black bear vs polar bear classifier using a set of images similar to the left and middle ones and test the model on the right image, the model can still make a mistake and classify the polar bear (Figure 1.2, right) as a black bear by looking at the green forest background. Therefore, applying deep learning as a universal black box without human’s involvement is not a solution either. Human’s domain knowledge is still required to introduce different forms of priors to help the model to learn the relevant features.

1.2 Thesis statement

In this work, we propose that contrastive learning when applied to domains represented as scene graphs can reduce the need for more labelled data for the representational learning problem, thus help us tackle visual relationship detection in a resource-constraint setting. However, given the unbalanced nature of the dataset, using the recall metric for evaluation gives a biased view towards common predicates. Thus, to have a more holistic view of the models’ performances, we also propose the use case of one-vs-all evaluation metrics and ROC curve for each individual relationship class. Our main contributions in this work are:

1. We applied translational loss in conjunction with contrastive vision and language loss for scene graph representational learning and showed that translational loss can improve the expressivity of the final embedding space without penalizing the performance of the VRD task.
2. We explored a new use case of the recent contrastive contextualized encoder-encoder architecture called CLIP for the visual relationship detection task, showing that existing pre-trained contrastive vision and language model can be applied to the VRD task.
3. We proposed an adaptive negative sampling strategy that drew negative samples from the inverse distribution of the drawn samples. By applying this sampling strategy over the naïve random sampling approach, our model achieves a consistent improvement on the Recall@n metric as we scale up its size.

1.3 Thesis outlines

This thesis is structured as follows. In Chapter 2, we start by reviewing the foundational building blocks of deep learning and the relevant literature on scene graph generation (D. Xu et al., 2017) and visual relationship detection (C. Lu et al., 2016). We also describe the different contrastive learning approaches used in this work.

In Chapter 3, we examine the origin of the Visual Genome dataset and discuss the different priors and limitations of the benchmarked data. We also examine the commonly used evaluation metrics for the VRD benchmarks and proposed the use of Informedness and ROC curves to evaluate the performance of the model.

Chapter 4 begins by describing how multi-modal representational learning via contrastive method can benefit the VRD task and propose two contrastive learning architectures to tackle the problem. The first VLTransE architecture interprets a scene graph as a knowledge graph and focuses on preserving the first-order hierarchical structure of the graph. The second architecture focused on reducing the memory footprint used during the contrastive training process and employs an existing contrastive learning architecture to tackle the VRD task.

In Chapter 5, we report the experimental results for our models along with other benchmarked results, identifying strengths and weaknesses in our proposed architectures. Here, we also outline the implementation details and design decisions, and analyse the outcomes of the proposed contrastive losses.

Finally, we conclude the thesis with a general discussion and future work section on VRD in Chapter 6.

Parts of this thesis were presented at the *AAAI-MAKE 2022 Spring Symposium, Stanford, US* (Tran, Santos, et al., 2022) and *ALTA 2022, Flinders University, AU* (Tran, Neau, et al., 2022).

Chapter 2

Literature Review

In recent years, deep learning systems have achieved considerable success in computer vision and natural language processing. This is especially true when the task involves perception or pattern recognition. However, these deep learning methods often exhibit brittleness and unpredictable errors when attempting to generalize beyond the given datasets (Geirhos et al., 2020). These errors happen because deep learning models are susceptible to *shortcut learning*, where the systems learn correlations that give the correct answers but for the wrong reasons (Geirhos et al., 2020). Over the years, there were three prevalent approaches that aimed to tackle this problem, including the evolution of hardware accelerator and computing strategies, the development of quality datasets and data processing pipelines, and the innovation in the deep learning architecture designs. While the improvements of the dataset and computer architecture are outside of the scope of this work, we aim to provide an overview of different neural architectures and give an interpretation on how these design choices can impact the performance of visual relationship detection models.

In this chapter, we describe the recent innovations in the deep learning architectural design space and show how introducing different types of human-engineered *inductive biases* or *priors* to the system can lead to better generalization in a wide variety of tasks. Building on top of these neural network architectures, we then review different methods of learning numeric vector representations called *embeddings* for images and text, with an emphasis on contrastive learning approaches. Finally, we explore the development of a scene graph dataset called the *Visual Genome* (VG) (Krishna et al., 2017), understand the implications of such dataset in recent years, and give an intuition on how the architectural choices can blend itself to improve the VRD task. Here, the VG dataset contains 108K pairs of image and scene graphs crowdsourced from Amazon Mechanical Turk workers (Keith et al., 2017), where each scene graph is represented by a set of visual relationship triples in RDF format.

2.1 Deep learning architectures

Artificial neurons are the foundational building blocks of any modern deep learning architecture, where each *neuron* is a linear function that transforms a set of input signals into an output signal. Here, an additional activation function is often applied to get the final output signal for other neurons to process, and many of these neurons can be aggregated into multiple interconnected layers, creating an *Artificial Neural Network* (ANN) or *Multi-Layer Perceptron* (MLP).

$$AN(\vec{x}) = W\vec{x} + b \quad \text{Eq. 2.1}$$

While Hornik et al. (1989) have theoretically proved that ANNs with at least one hidden layer are universal approximators for any functions, practical applications of ANNs show that one hidden layer is far from enough. In practice, deeper networks are required in order to learn better representations of a complex set of data.

Still, labelled observations and datapoints are limited, and ANN, when trained on a small set of data, often leads to biases and poor generalization. Therefore, different types of *inductive biases* and architectures were introduced to help the model better generalize to new situations. While we cannot cover all possible inductive biases and architectures here, we will describe some contemporary architectural choices for visual relationship detection such as *Convolutional Neural Network* (CNN) and the *Transformer* as they are most relevant to this work.

2.1.1 Convolutional Neural Network

Convolutional Neural Network (CNN) (Lecun et al., 1998) is one of the most popular deep learning architectures used to tackle many of the perception problems, including object recognition (LeCun et al., 1999), object detection (Ren et al., 2015), and instance segmentation (He et al., 2018). Its success can be attributed to the design choices based on two observable properties in visual images (LeCun et al., 1999). First is the *locality property* where nearby pixels are often related to the same object, and second is the *spatially invariant* property which states that the same object can occur in different locations in an image. The CNN architecture, designed with constraints surrounding these two observations, consists of three main types of layers: the *convolutional layer*, the *pooling layer*, and the *fully connected layer*.

- The *convolutional layer* consists of multiple weighted kernels or filters, where each filter aggregates local information or pixels from the previous layer to

produce a new pixel in the next layer. To fully form a new layer, the filter can be thought as a sliding window that produces one new pixel with every slide.

- The *pooling layer* focuses on regularizing the network by performing a down sampling operation, usually in the form of a MAX or AVERAGE pooling. Like the convolutional layer, we can think of the pooling layer as a sliding rectangular window that performs the selected operation at every slide.
- The *fully connected layer* or multi-layer perceptron (MLP) is used to perform a matrix multiplication followed by a bias offset that transforms the previous layer into a new layer. Usually, a non-linearity activation function such as ReLU (Eq. 2.2) is applied after the fully connected layers and the convolutional layers in order to enables the depth of the network.

$$f_{ReLU}(x) = \max(0, x) \quad \text{Eq. 2.2}$$

With these three operations, a CNN trained on image classification task can capture general knowledge about the objects, including the hierarchical structure of pixels where lower layers learn local information like edges and higher layers learn global information like objects (Krizhevsky et al., 2017). Despite CNN's original design for pattern recognition, its scope has broadened as subsequent works demonstrating the effective use of pre-trained CNN models to extract visual embeddings for more complex downstream tasks. Selvaraju et al. (2017) adds this point by showing through activation maps how convolutional layers learn to localize discriminative parts and features, which has been crucial for downstream image classification, object detection, and visual question answering tasks. In this work, we use these pre-trained CNN models to extract visual embeddings for downstream VRD task. Besides the CNN architecture, we also use the Transformer architecture for extracting both visual and textual embeddings as described in Section 2.1.2 below.

2.1.2 Transformer

While the CNN has led to breakthroughs in many computer vision tasks, the strong local spatial assumption that made the CNN succeed in the image domain have prevented its application to the language domain which requires a holistic understanding of the global context. The Transformer architecture (Vaswani et al., 2017), originally designed for machine translation (Sutskever et al., 2014), solves such limitation by relaxing the spatial assumptions inherent to the CNN design. As a result of these relaxed

assumptions, the architecture can be applied to multiple domains including language (Devlin et al., 2019), speech (Chorowski et al., 2015; Radford et al., 2022), and image (Dosovitskiy et al., 2020), although the latter required a somewhat unnatural representation of images as sequence. Unlike the CNN which assumes the locality and spatially invariant property of the input, the Transformer makes no assumptions about the temporal and spatial relationships across data (Devlin et al., 2019; Radford et al., 2018). Instead, along with the widely used MLP and skip connection, the Transformer encodes the position of its token and mainly operates using the *scaled dot product attention* mechanism where it learns to retrieve the most relevant information from the previous layer based on a learnable query (Bahdanau et al., 2016). While there are other methods to compute attention, this following section focuses solely on the *scaled dot product attention* operation and calls it as *attention* for short.

Formally, given input $X \in \mathcal{R}^{batch \times tokens \times d_{model}}$ where *tokens* is the length of the input sequence and d_{model} is the vector length of each token embedding, the attention mechanism learns three weight matrices $W_Q, W_K, W_V \in \mathcal{R}^{d_{model} \times d_k}$ that transform the tokens' vectors into queries, keys, and values with arbitrary dimension d_k . The attention operation is defined as follows:

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad Eq. 2.3$$

$$Q = XW_Q, K = XW_K, V = XW_V$$

A simple and hypothetical way to visualise how attention works is to look at the example of retrieving values from a hash table using a given set of queries and keys (Vaswani et al., 2017). Of course, in the attention mechanism, the hash tables are the output embeddings of the previous fully connected layers, and the hash functions are learnable linear projections or weights. Here, the query-key matching operation generates a probability distribution over all possible keys and retrieves the weighted sum of the associated values based on the generated distribution. This attention operation can also be visualized in *Figure 2.1*.

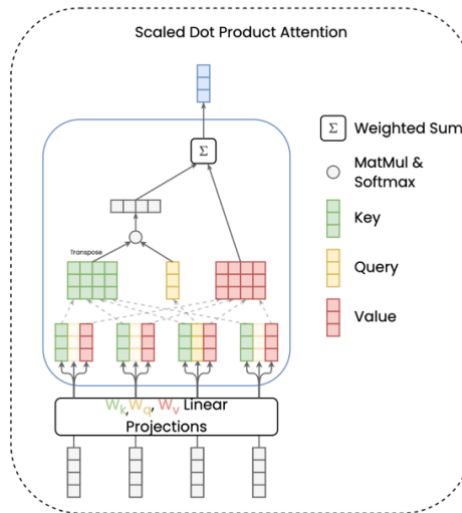


Figure 2.1: a visualization of *scaled dot product attention*. Adapted from Vaswani et al., 2017.

Under this hash table analogy, a layer with only one attention operation can route information in a single way. However, different words in a sentence can relate to each other in many ways, and routing information from only a limited set of input tokens prevents the model from learning the distinct syntactic, semantic, and discourse relationships (Voita et al., 2019). To overcome this limitation, the Transformer performs multiple attention operations with distinct learnable weights in parallel, creating multiple information routes at each layer. This process is also called *multi-head attention* where each head consists of an attention operation:

$$MultiHead(Q, K, V) = Concatenate(head_1, \dots, head_h)W_O \quad Eq. 2.4$$

$$\text{where } head_i = Attention(QW_Q^i, KW_K^i, VW_V^i), i = 1, \dots, h$$

Originally designed for the machine translation task, the Transformer's goal is to transform the original language input into a sequence of probability distribution of targeted language tokens (Vaswani et al., 2017). To facilitate the transformation of language, the architecture encodes the original language using the encoder module (Figure 2.2, left) and transforms the encoded tokens to the target language tokens using the decoder module (Figure 2.2, right). This encoder-decoder architecture applies the described multi-head attention in three different types of layers: the *self-attention layer*, the *causal self-attention layer*, and the *cross-attention layer*.

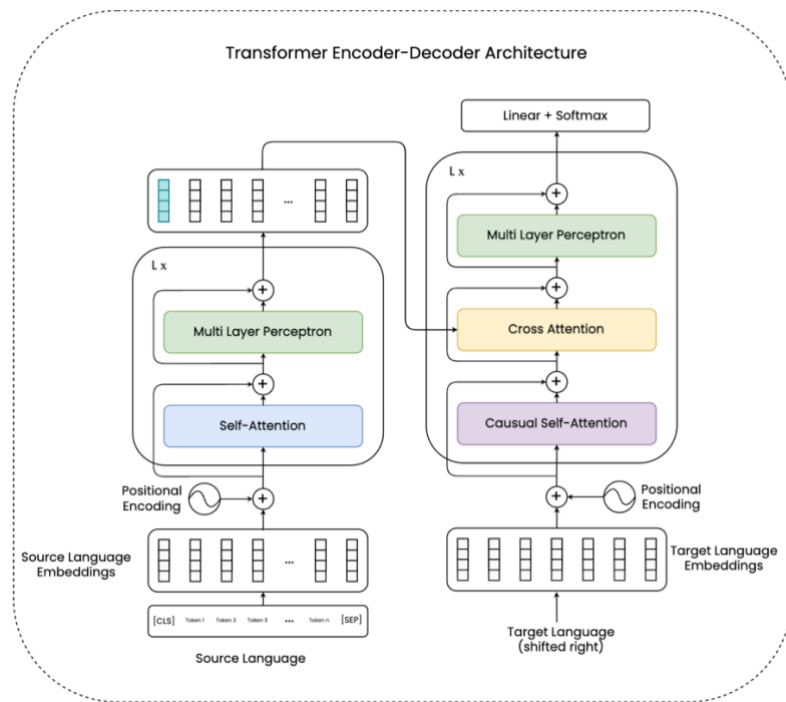


Figure 2.2: Visualizing the building blocks for the Transformer model. Adapted from Vaswani et al., 2017.

In the self-attention layer, the keys, queries, and values are learned directly from the input. The described multi-head attention operation is then applied to aggregate information from the previous input layer to generate an output vector for the next layer. The *causal self-attention layer* also performs the multi-head attention operation to generate the output vectors. However, instead of aggregating information from every input vector in the entire sequence, the causal self-attention mechanism only looks at input vectors that occur in the previous positions in the sequence (*Figure 2.3*). In practice, this is done by multiplying the input sequence by a binary mask. Due to its backward-looking property, this layer is only used in the decoder block to perform inference for the next token. The cross-attention layer is also used in the decoder like the causal self-attention layer. However, different from the above two layers, the layer aims to retrieve information or values from the encoder output embeddings by matching the encoder generated keys with the decoder generated queries. This facilitates information flow between the encoder and decoder module as shown in *Figure 2.2*.

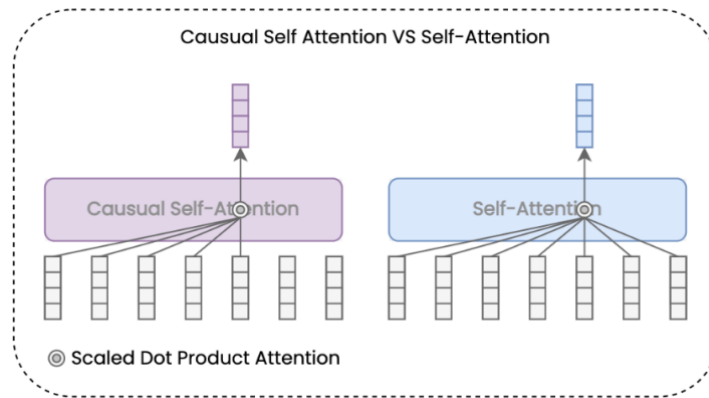


Figure 2.3: a visualization for *causal self-attention layer vs self-attention layer*.

Besides the described attention layers, the architecture also uses the *residual connection* or *skip connection* (Figure 2.4) to prevent information loss as the depth of the network increases (He et al., 2015). This residual connection when coupled with MLP has shown to be an important structural prior that enables the depth and scalability of the model. Here, the residual connection serves two purposes: (1) it prevents the gradient vanishing problem that often occurs with deeper neural networks (He et al., 2015), and (2) it creates additional pathways for information flow from the lower layers to the higher layers (Dong et al., 2021). While Transformer-based models demonstrated impressive performance on a wide variety of tasks, it is still an open question on how to effectively interpret and explain the results generated by the model. Such topic involves the subfield of explainability and responsible AI (Arrieta et al., 2019; Ilinykh & Dobnik, 2021) and is not covered under the scope of this work.

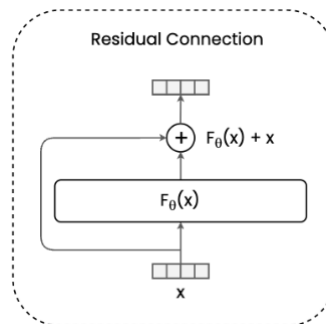


Figure 2.4: visualization for *residual connection*. Adapted from He et al., 2015.

In *Section 2.2* below, we further describe the applications of Transformer-based and CNN-based architectures in the image and text representational learning problem, highlighting their strengths and weaknesses when compared with alternative designs. In *Section 2.4.4*, we also highlight how Transformer can be used to learn different topological structures from knowledge graphs and the scene graph. While such approach described in *Section 2.4.4* is not directly applied in this work, understanding of how neural

network architectures can be engineered for scene graph can help us build better models in future work.

2.2 Representational learning for image and text

Deep learning main strength stems from its powerful representational ability of large-scale datasets, where the learned numeric representation can be transferred to other downstream tasks through a process called *transfer learning* (F. Zhuang et al., 2020). Here, transfer learning is the process of learning and storing general knowledge from a set of pre-defined tasks and applying this knowledge towards a different or related machine learning task (Bengio et al., 2012). With ANNs, this general knowledge is stored in a distributed manner using continuous and low-dimensional vector embeddings (Bengio et al., 2012). For the visual relationship detection task, these vector embeddings should capture knowledge from both the image and text modalities because the objects' interactions rely not only on visual appearances but also the labels and use cases of the given objects. This multimodal representational learning is particularly challenging because images are considered as signal with redundant information whereas textual triples are symbolic.

In *Section 2.2.1*, we introduce different techniques for learning language representations from large textual corpora in the form of dense vector embeddings. Similarly, in *Section 2.2.2*, we introduce different state-of-the-art techniques used to capture visual features from images. Finally, in *Section 2.2.3*, we describe different approaches to jointly learn image and text representations that incorporate information from both modalities.

2.2.1 Language representational learning

In language, words that have similar meanings often have similar surrounding context and syntactic structure, and embedding models make use of such observation to generate words' vector representations. For example, Mikolov et al. (2013) introduce Word2Vec, which consists of two methods of learning word vector representations: the continuous bag of words (CBOW) method and the Skip-Gram method. The CBOW model finds word representations based on their surrounding words or context, while the Skip-Gram model finds word representations by predicting the surrounding words in the sentence or document (Mikolov et al., 2013). After training, these dense vectors can capture not only the syntactic relationships but also the surface-level correlations between

words' meanings. For example, $\text{vector}(\text{"King"}) - \text{vector}(\text{"Man"}) + \text{vector}(\text{"Woman"})$ results in the vector("Queen"), where $\text{vector}(w)$ is the embeddings for the word w .

While these distributed representations or embeddings can capture general words' meaning, the trained embedding vectors for individual words are fixed no matter what the surrounding word or context is (McCann et al., 2018; Peters et al., 2017). This drawback means that these vectors cannot fully represent the polysemous nature of language, where one word can have a different meaning in a different context (Herskovits, 1986). To resolve this problem, Peters et al. (2018) introduced a new type of deep contextualized word embedding model called ELMo. Unlike traditional word embeddings that are fixed for every input word, contextualized models can generate word embeddings that vary according to the sequence context. For example, the same token 'bank' in the sentence 'he withdraws money from the bank' and the sentence 'he is sitting on the bank next to the river' would have two different contextualized embeddings. To generate these contextualized word embeddings, the model takes the entire sequence of words as input and learns to predict the next word from the previous words in a task called *language modeling*. Here, ELMo uses a recurrent neural network (RNN) or more specifically long short term memory (LSTM) network (Sutskever et al., 2014), to learn the contextualized embedding for each word (Peters et al., 2018).

However, using recurrent neural networks often exhibits long-term dependency issues due to their sequential nature and the information bottleneck problem (Cheng et al., 2016). To counteract these problems, RNN is often trained with the attention mechanism which aims to aggregate information from all the hidden states weighted by their importance (Bahdanau, 2014). This attention mechanism is later extended to create the Transformer architecture described in *Section 2.1.2* above (Vaswani et al., 2017). In a way, the Transformer trades away the sequential nature of the RNN architecture in preference for parallelism, where the model processes the entire sequence at once instead on a token-by-token basis. The downside to this parallelism is that the length of the input sequence is limited by the amount of physical resource available, with GPU's memory being the most important constraint. To preserve the position of the input, the model introduces an additional input refinement step that adds sinusoidal positional encoding to the input embeddings. Every layer in the Transformer can be thought of as a sequence of states, and a single state in the sequence is formed by aggregating information from all states in the previous timestep using the attention operation (Vaswani et al., 2017). Thus,

zothe final output embedding of a position encodes information not just from input token but also from other tokens in the input sequence.

Given the success of Transformer in the machine translation task, different contextualized word embedding models emerged under the same design choices and building blocks. For example, Devlin et al. (2019) introduced BERT, a Transformer encoder pre-trained on the *masked language modeling* and the *next sentence prediction* tasks (Devlin et al., 2019). Here, BERT takes a sequence of encoded tokens as input and propagates them through multiple layers of self-attention and MLP blocks to output a sequence of contextualized word embeddings. Similarly, GPT-2 uses only the decoder blocks of the Transformer and was pre-trained using the *language model* task. Unlike BERT, GPT-2’s embeddings are often not directly used for downstream tasks. Instead, GPT-2 is often fine-tuned using discriminative losses that fit the needs of the experimenter’s goal (Radford, Narasimhan, et al., 2019).

In this thesis, we apply Transformed-based encoders like BERT to learn contextualized word embeddings for visual relationship triples. Still, these word embedding techniques are not enough. While word embedding models can capture general language prior knowledge, VRD also requires visual features extracted from the visual entities or objects in the image. The next section describes the image representational learning process and introduces the application of CNN and Transformer to generate visual embeddings.

2.2.2 Image representational learning

As described in *Section 2.1.1*, a CNN is designed with two main assumptions: (1) the locality property which indicates that nearby pixels are related, and (2) the spatially invariant property which states that different portions of the image should be processed identically despite their location by sharing the filter weights (LeCun et al., 1999). Under these design decisions, a CNN trained on image classification task can capture the hierarchical structure of pixels where the lower layers learn local information like edges and the higher layers learn global information like objects (Krizhevsky et al., 2017).

As image datasets become larger and computational hardware becomes exponentially faster over the year, pre-trained CNN models such as AlexNet (Krizhevsky et al., 2017), VGGNet (Simonyan & Zisserman, 2015), GoogLeNet (Szegedy et al., 2014), and ResNet (He et al., 2015) for image representation have also become larger and deeper. As a practical result of the increased scale of these models, the final layer output

of the CNN pre-trained on the image classification task is often used as a feature map for visual embeddings that can be transferred to other relevant downstream tasks. Beyond image classification for image representation, CNN is also used in object detection, which aims to detect and classify different objects by predicting their corresponding bounding boxes in conjunction with the object's class. One specific object detection model is Faster R-CNN, which consists of a shared backbone convolution network, a region proposal network (RPN), and a region of interest detector (Ren et al., 2015). Built on top of the backbone network, the RPN proposes potential bounding boxes by predicting the probability that an object exists while minimizing the difference between the ground truth bounding boxes and the proposed anchor boxes, which are then refined to get the final proposals (Ren et al., 2015). Still, the original Faster R-CNN architecture uses only the final output layer of the backbone CNN to perform both RPN and feature extraction, creating a bottleneck and limiting its capability to detect smaller object sizes because the smaller objects' information is often lost in higher layers due to the pooling procedure.

To resolve this problem, the Feature Pyramid Network (FPN) is often used in conjunction with Faster R-CNN model to incorporate features from different levels in the backbone network (T.-Y. Lin et al., 2017). The FPN constructs multiple feature maps with different resolutions based on convolutions, allowing the detection of large objects to take place in lower-resolution feature maps and small objects to take place in higher-resolution feature maps. As a result, the model does not only depend on the final layer output of the CNN backbone but also on the intermediate layers to extract its features, increasing the model's expressivity and reducing the amount of information loss through multiple pooling layers. In Chapter 4, we train the Faster R-CNN with FPN for detecting objects in the visual embedding module to facilitate fair comparisons with existing work. We also use the multi-resolution feature maps extracted from the trained CNN backbone to extract visual features for downstream tasks.

While CNN has been the dominant architecture for visual encoding, Transformer-based models for vision have also gained popularity in recent years due to their flexibility and scalability. More specifically, Transformer variants called the Visual Transformer (ViT) (Dosovitskiy et al., 2020) have shown competitive results in computer vision tasks such as image recognition (Touvron et al., 2021), object detection (Carion et al., 2020; X. Zhu et al., 2021), and image segmentation (J. Chen et al., 2021). However, the ViT's architecture does not contain any spatial inductive biases like the CNN's convolutional layer. To solve this issue, the model subdivides the image into smaller patches to form

2D grids of different sizes (14x14, 16x16, 32x32) and incorporates spatial information by assigning an additional positional embedding for each patch. These positional embeddings can be automatically learned during training via projection matrices. Interestingly, empirical experiments show that the different initialization techniques for positional embeddings do not affect the final performance of the model (Dosovitskiy et al., 2020). Even with randomly initialized positional embeddings, a ViT model trained the image classification task can automatically learn structures that capture local information at lower layers and global information at higher layers (Raghu et al., 2022). While this observation is similar to that of a CNN, ViT models require more training data and scale to achieve the same level of performance as that of CNN-based models (Raghu et al., 2022). In a way, deep neural networks with relaxed assumptions like ViT can still implicitly learn object-level patterns and structures from images. However, there is still value in introducing design biases to the neural model to increase learning and inference efficiency (C.-Y. Wang et al., 2022).

In Section 4.3, we compare the performance of ViT of different scale on the visual relationship detection benchmarks in the context of joint visual and language representational learning. Section 2.2.3 briefly expands upon the task of joint language and image representational learning and introduces contrastive representational learning within the wider context of self-supervised learning framework.

2.2.3 Joint language and image representational learning

The task of joint vision and language representational learning can be seen as a subtask of the multi-modal machine learning problem, where the goal is to process and understand information from different sensors. Here, we focus on the challenge of learning how to represent both image and text in a way where each modality can complement and not interfere with the other. This motivates further research in developing a joint representation for both language and vision where each modality contributes positively to the other, facilitating a more complete set of concepts.

Most techniques for learning joint representations depend on deep learning architecture automatically learn relevant information from the language and image modalities. Given the successes of Transformer in unimodal representational learning like BERT (Devlin et al., 2019) and ViT (Dosovitskiy et al., 2020), the same Transformer architecture was also applied to learn joint vector representations that capture not only image and text features but also their dependencies. There are two main approaches to

adapt the existing Transformer architecture to both image and text: (i) the single-stream method and (ii) the dual-stream method. In the single-stream approach, both image and text are encoded using a single encoder module with self-attention operations, and their interactions are automatically learned during optimization (Y.-C. Chen et al., 2020; Li et al., 2019; Su et al., 2020). This is usually done by concatenating image region embeddings and word embeddings into an input sequence and encoding the sequence using the encoder network and loss functions. Different from the single-stream method, the dual-stream approach uses two encoder modules, one for image and one for text (J. Lu et al., 2019). The encoded image embeddings and the text embeddings are exchanged through the co-attention transformer layer (J. Lu et al., 2019). In either case, we can observe that information from both modalities is exchanged through the attention mechanism, and the flow of information is determined by the architectural choice and a set of pre-defined objectives.

While the simplicity of the Transformer architecture made it popular in recent work, a well-designed ANN architecture is not enough. Another critical determinant of success for these networks are the *self-supervising* objectives or loss functions. Here, self-supervised learning (SSL) encompasses both supervised and unsupervised learning, and the goal of the SSL is to help the model to learn useful representation that can be transferred to downstream supervision task. Two of the most common SSL goals involve: (i) predicting the future from the past, or (ii) predicting the masked parts from the visible parts (Ericsson et al., 2022). For example, UNITER (Y.-C. Chen et al., 2020) applied four different SSL pretext objectives: (i) the *masked language modeling* aims to predict the missing words based on the surrounding input context, (ii) *masked region modeling* that aims to reconstruct the image region based on the surrounding context, (iii) *image-text matching* that aims to predict whether the image and text are related, and (iv) *word-region alignment* that aims to minimize a defined distance between matching words and image regions. In practice, selecting a good neural architecture along with relevant SSL loss objectives leads to a consistent improvement in benchmark performance. However, the theory as to why certain combinations of architectural choices and loss functions improve the model’s representations remains open. This topic involves the subfield of multitask learning and meta learning and is covered by Hospedales et al. (2022)’s survey.

While our approach presented in Chapter 4 also leverages SSL to learn joint visual and language representations, the architecture mostly comprises of contrastive loss functions as described in *Section 2.3* below.

2.3 Contrastive Representational Learning

Contrastive Representational Learning (CRL) (Le-Khac et al., 2020) is a subtask of the self-supervised learning framework which aims to enforce a set of loss functions that minimize the distance between the embeddings of positive pairs while maximising the distance between the embeddings of negative pairs. In the visual relationship detection context where data annotations are scarce, we believe that CRL can serve as an efficient method for aligning the image representation with the corresponding visual relation triples. To fully understand the inner workings of contrastive methods and compare them with other self-supervised learning objectives, we first describe both the statistical modeling framework and the energy-based modeling framework (Lecun et al., 2006). Afterwards, we explore the different approaches for learning the Energy-based model, with an emphasis on *noise contrastive estimation* (NCE) and its variants. Here, we give examples for the classification task and ranking task because most of the current literature frames the VRD task as a discriminative problem.

2.3.1 Statistical models and Energy-based models

For the classification or prediction task, the goal of estimating a statistical model is to capture the dependencies between the input features and the label. In the ideal case, the model encodes the relevant dependencies from the observed samples, giving it the ability to make predictions about unknown samples drawn from the population.

Given that we have m samples with labels $D = \{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$ where each pair is drawn from an unknown data distribution $p_{data}(\cdot, \cdot)$ over $\mathbb{R}^n \times \mathcal{Y}$, we create a statistical model parametrized by θ with a distribution $p_{\theta}(Y|X)$ that aims to predict the label from the observable data. Using maximum likelihood estimation (MLE), we can then estimate θ by maximizing an objective function such as the log likelihood function. Under the stochastic modeling constraint, any estimated solution $\hat{\theta}$ must satisfy:

$$\int p_{\hat{\theta}}(Y|X) dx = 1$$

To satisfy the above constraint where the probability distribution is integrated to one, we can redefine the PDF using a normalizing constant Z_{θ} and a new function $\varphi_{\theta}(Y|X)$.

$$p_{\theta}(Y|X) = \frac{\varphi_{\theta}(Y|X)}{Z_{\theta}} \tag{Eq. 2.5}$$

$$\text{where } Z_\theta = \int_{y \in \mathcal{Y}} \varphi_\theta(y|X)$$

Here, $\varphi_\theta(Y|X)$ can be an unnormalized model parametrized by θ and $\varphi_\theta(Y|X) \propto p_\theta(Y|X)$. The normalizing constant Z_θ , also known as *the partition function*, ensures that the above constraint for $p_\theta(Y|X)$ holds. One problem arises is that the integral for the partition function Z_θ is intractable and cannot be easily evaluated under closed-form solution when the labels \mathcal{Y} are high dimensional or have high cardinality (Lecun et al., 2006). Thus, it becomes difficult to directly compute the gradient for the log likelihood term $\log Z_\theta$ with respect to its parameters θ (see Eq. 2.7).

$$\begin{aligned} \theta_{MLE} &= \underset{\theta}{\operatorname{argmax}} \log \prod_{i=1}^m p_\theta(y_i|x_i) \\ &= \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^m \log p_\theta(y_i|x_i) \end{aligned} \tag{Eq. 2.6}$$

$$\nabla_\theta \sum_{i=1}^m \log p_\theta(y_i|x_i) = \sum_{i=1}^m \nabla_\theta \log \varphi_\theta(y_i|x_i) - \nabla_\theta \log Z_\theta \tag{Eq. 2.7}$$

To avoid the intractable integral problem, the energy-based model (EBM) relaxes the normalization constraint and propose the use of an *energy function* for gradient optimization (Lecun et al., 2006). Here, instead of optimizing the log likelihood of a probability density function, an energy-based model optimizes the *energy function*, which outputs a number that measures the dependency between the input x and the label y . Following convention, the energy function parameterized by θ is denoted as $E_\theta(X, Y)$, and the optimization procedure aims to find the optimal solution for a *loss function*, which measures the quality of the energy function. In our context which involves real-world visual information, the energy function can also be interpreted as a nonlinear neural network model with parameters θ , and a loss function is an objective function that guides the optimization process. Here, we select neural network models due to their success in representing visual images in recent years as described in Section 2.2.2. In Section 2.3.2 below, we describe the loss functions used in this work under the energy modeling framework.

2.3.2 Loss Functions for Energy-Based model

With relaxed assumptions, there are many ways to design a loss function for an energy-based model (H. Li et al., 2022). However, not all loss functions can yield an optimized EBM that can perform the classification task. While the search space for a loss function is large, we can apply two intuitive conditions to design a good loss function. First, by convention, a model optimized on a good loss function should yield the lowest energy for a correct sample pair (Lecun et al., 2006). More formally, given matching (x^i, y^i) pair, the energy function gives the correct answer for x^i if $E_\theta(x^i, y^i) < E_\theta(x^i, y)$, $\forall y \in \mathcal{Y}$ and $y \neq y^i$. Second, for an optimized model, the energy score for the matching pairs should be lower than that of non-matching pairs by a margin of m (Rosasco et al., 2004). More formally, given an additional incorrect pair (x^i, \bar{y}^i) and a positive margin m , the model should give the correct answer for x^i if $E_\theta(x^i, y^i) < E_\theta(x^i, \bar{y}^i) - m$. In the current literature (Gentile & Warmuth, 1998; Schroff et al., 2015; Wei et al., 2020), these two conditions combined form the foundation of the *margin loss* functions.

One of the most popular margin loss functions for the classification problem is the *hinge loss* (Rosasco et al., 2004). Given a positive margin m and a parameterized energy function, the *per-sample* hinge loss is defined in Equation 2.8:

$$L_{hinge}(x^i, y^i) = \max(0, m + E_\theta(x^i, y^i) - E_\theta(x^i, \bar{y}^i)) \quad Eq. 2.8$$

$$\mathcal{L}_{hinge} = \frac{1}{n} \sum_{i=1}^m L_{hinge}(x^i, y^i) + \lambda R_\theta \quad Eq. 2.9$$

Here, \mathcal{L} is the loss function averaged over all observed samples and R_θ is a *regularizer* that prevents the energy function from overfitting or underfitting. In the hinge loss, the parameter is only optimized if the difference between the energy of the incorrect answer and energy of the correct answer is larger than m . Still, because there is no constraint to the output energy, the model can be overfitted to the data samples and perform poorly on unseen pairs. Thus, an additional regularizer function is added to regularize the parameters, forcing the model to explore relevant features.

While the hinge loss was traditionally used for the classification task, it can also be applied to the ranking task. Different from the classification task that aims to select only the best label y given the input x , the ranking task produces a complete ranking of

all possible labels $y \in \mathcal{Y}$ when given input x . This application is more suitable to the VRD task since the *(subject, object)* pair can have multiple relations. In the current literature (Balntas et al., 2016; Ge et al., 2018; B. Yu & Tao, 2019), the hinge loss when applied to the ranking task is also known as the *triplet margin loss* (Balntas et al., 2016), and the energy function in the triplet margin loss outputs a defined distance of the input pair. Figure 2.5 below shows how the triplet margin loss pushes the positive and anchor vectors closer together while pushing apart the negative and anchor vectors.

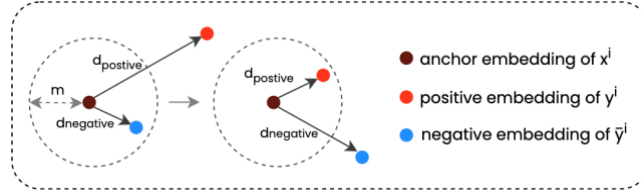


Figure 2.5: a visualization for *triplet margin loss*. In this figure, \mathbf{m} is the margin, $\mathbf{d}_{positive}$ is the distance anchor embedding and positive embedding, and $\mathbf{d}_{negative}$ is the distance anchor embedding and negative embedding.

While the above margin loss functions are not restricted to the conditional probabilistic form under the EBM framework, following the probabilistic interpretation when formulating a loss function is still useful for the classification task. The *negative log-likelihood loss* (NLL) (Bosman & Thierens, 2000), also known as the *cross-entropy loss*, is one of such loss functions that can be used to train a model to produce conditional probability estimates. Due to the convention where we want to minimize the energy, the negative sign is added to the log likelihood function, hence the name *negative log-likelihood*. Under the EBM framework, we can modify Equation 2.5 and replace $\varphi_{\theta}(Y|X)$ with $e^{-\frac{1}{t}E_{\theta}(X,Y)}$ to yield:

$$p_{\theta}(Y|X) = \frac{e^{-\frac{1}{t}E_{\theta}(X,Y)}}{\int_{y \in \mathcal{Y}} e^{-\frac{1}{t}E_{\theta}(X,y)}} \quad \text{Eq. 2.10}$$

where the temperature t is an arbitrary constant used to regulate the model's representational space and $\int_{y \in \mathcal{Y}} e^{-\frac{1}{t}E_{\theta}(X,y)}$ is the partition function. In machine learning, the above model is also known as the SoftMax function of $-\frac{1}{t}E_{\theta}(X,Y)$, and the exponential function is a monotonic function applied for mathematical convenience. Applying negative log to the above likelihood, we get the following NLL:

$$L_{nll}(x^i, y^i) = -\log \prod_{i=1}^m p_{\theta}(y_i|x_i) = \sum_{i=1}^m -\log \left(\frac{e^{-\frac{1}{t}E_{\theta}(X,Y)}}{\int_{y \in \mathcal{Y}} e^{-\frac{1}{t}E_{\theta}(X,Y)}} \right) \quad Eq. 2.11$$

Because the partition function is parameterized and consists of all labels $y \in \mathcal{Y}$, the NLL does not only push down the energy of the matching pair but also pulls up the energy of all answers in proportion to the amount of energy that gets pushed down (Lecun et al., 2006). In the probabilistic framework, minimizing the NLL is equivalent to minimizing the KL divergence between the estimated distribution $p_{\theta}(y_i|x_i)$ and the underlying data distribution of the dataset $p_{data}(y_i|x_i)$ (Song & Kingma, 2021).

$$\begin{aligned} \sum_{i=1}^m -\frac{1}{m} \log p_{\theta}(y_i|x_i) &= -\mathbb{E}_{y_i|x_i \sim p_{data}(y_i|x_i)} [\log p_{\theta}(y_i|x_i)] \quad Eq. 2.12 \\ &= D_{KL}(p_{data}(y_i|x_i) \parallel p_{\theta}(y_i|x_i)) - constant \end{aligned}$$

Here, minimizing the NLL is also equivalent with maximizing the mutual information between the matching observed samples and labels (Bengio et al., 1993; Wu et al., 2020). Despite its popularity, taking the derivative of the NLL still requires the evaluation of the partition function and its derivatives which can be computationally demanding (Gutmann & Hyvärinen, 2010). To tackle this problem, multiple methods were proposed including Markov chain Monte Carlo (MCMC) sampling (van Ravenzwaaij et al., 2018), Score Matching (Hyvärinen, 2005), and Noise Contrastive Estimation (Gutmann & Hyvärinen, 2010). Here, we focus on Noise Contrastive Estimation because it forms the foundation of the contrastive learning approach with negative sampling used in this work.

2.3.3 Noise Contrastive Estimation Loss

The Noise Contrastive Estimation function aims to estimate the parameter of an EBM without the calculation of the partition function (Gutmann & Hyvärinen, 2010). Instead of learning to predict the label y given the observed feature x , NCE aims to learn a binary classifier $p_{\theta}(D|Y, X)$ that predicts whether the sample pair comes from the real distribution $p_{real}(Y|X)$ or the noise distribution $q_{noise}(Y)$. These conditional probability terms can be written as:

$$\begin{aligned} p(D = 0 | Y, X) &= \frac{kq_{noise}(X)}{p_{real}(Y|X) + kq_{noise}(Y)} \\ p(D = 1 | Y, X) &= \frac{p_{real}(Y|X)}{p_{real}(Y|X) + kq_{noise}(Y)} \end{aligned}$$

To gather the data for the above conditional terms, we sample k negative samples with label $D = 0$ from the noise distribution $q_{noise}(Y)$ for every one positive sample with label $D = 1$ from $p_{real}(Y|X)$. In the original paper (Gutmann & Hyvärinen, 2010), as the number of negative examples k approach infinity, the gradient of $L_{binaryNCE}$ becomes zero once $E_{\theta}(Y, X) = p_{real}(Y|X)$. By the weak law of large numbers, this *self-normalization* assumption allows us to replace $p_{real}(Y|X)$ with the energy function $E_{\theta}(Y, X)$:

$$p_{\theta}(D = 0 | Y, X) = \frac{kq_{noise}(X)}{E_{\theta}(Y, X) + kq_{noise}(Y)}$$

$$p_{\theta}(D = 1 | Y, X) = \frac{E_{\theta}(Y, X)}{E_{\theta}(Y, X) + kq_{noise}(Y)}$$

We can rewrite the log likelihood of the above conditional probabilities as the binary classification loss:

$$L_{binaryNCE} = \sum_{(X,Y) \in D} \log p_{\theta}(D = 1 | Y, X) + k \mathbb{E}_{Y' \sim q_{noise}} [\log p_{\theta}(D = 1 | Y', X)] \quad Eq. 2.13$$

One observation is that the NCE loss is highly dependent on the choice of the noise distribution, and a poorly selected noise distribution can interfere with the learning process (Gutmann & Hyvärinen, 2010). Ideally, the noise distribution q_{noise} is close to the real data distribution and it is easy to sample from. However, this can quickly become difficult without additional labels. In practice, the process of identifying noise requires a closer manual examination of the task and the dataset at hand. An alternative way to interpret the selection of noise distribution is through the process of *negative sampling* proposed by Mikolov et al. (2013). Here, negative sampling is the process of generating negative examples that are beneficial to the given task. It can be viewed as a simplified version of NCE since it is not aiming to model $p(y|x)$ but to model a different quantity related to joint distribution $p(x, y)$ (Mikolov et al., 2013). In this work, we examine two different negative sampling techniques for VRD, showing that random selection of negative samples can impede the learning process and yield subpar performance.

More recent work extends NCE to the multiclass ranking case called InfoNCE by creating a classifier that identifies the positive samples from the negative samples (Oord et al., 2019). Given the labels y_1, \dots, y_m that consist of one real sample $y_i \sim p_{real}(Y|X)$, and $m - 1$ noisy samples $y_{j \neq i} \sim q_{noise}(Y)$, we create a categorical indicator d where $[d =$

i] means that y_i is a positive example. Using d , we can construct the probability for classifying the positive example correctly using in the equation below (Oord et al., 2019):

$$p(d = i | Y, X) = \frac{\frac{p(y_i|X)}{p(y_i)}}{\sum_{j=1}^k \frac{p(y_j|X)}{p(y_j)}}$$

Under the assumption that a neural network model is expressive enough to represent the density ratio $\frac{p(Y|X)}{p(Y)}$, we can replace it with $\phi_\theta(X, Y) \propto \frac{p(Y|X)}{p(Y)}$.

$$p(d = i | Y, X) \approx \frac{\phi_\theta(X, y_i)}{\sum_{j=1}^k \phi_\theta(X, y_j)} \quad \text{Eq. 2.14}$$

While the above term looks like Equation 2.5, the main difference is that we are estimating for $\frac{p(Y|X)}{p(X)}$ instead of $p(Y|X)$. We can then compute the negative log likelihood over all samples in the dataset to get the infoNCE loss in Equation 2.15.

$$L_{\text{InfoNCE}} = -\frac{1}{m} \sum_{i=1}^m \log \frac{\phi_\theta(X, y_i)}{\sum_{j=1}^m \phi_\theta(X, y_j)} \quad \text{Eq. 2.15}$$

Under this construct, the InfoNCE paper (Oord et al., 2019) also shows that by approximating the density ratio $\frac{p(Y|X)}{p(Y)}$ with an EBM model, the above loss maximizes the mutual information between the datapoint y_i and its context X , $I(y_i, X)$, while minimizing the mutual information between the $y_{j \neq i}$ and the context X , $I(y_{i \neq i}; X)$. Here, *mutual information* between two random variables measures how dependent they are to one another, and is defined as:

$$I(Y; X) = \sum_{Y, X} p(Y, X) \log \left(\frac{p(Y, X)}{p(Y)p(X)} \right) = \sum_{y, X} p(Y, X) \log \left(\frac{p(Y|X)}{p(Y)} \right) \quad \text{Eq. 2.16}$$

where $\frac{p(Y|X)}{p(Y)}$ is approximated by $\phi_\theta(X, Y)$.

Soft nearest neighbour (SNN) loss (Frosst et al., 2019) takes a step further to include multiple positive samples for the ranking task. Given a batch of samples $\{(x_1, y_1), \dots, (x_m, y_m)\}$ where y_i is the class label of x_i and $d(\cdot, \cdot)$ measures the distance between two inputs, the soft nearest neighbour loss is defined as:

$$L_{snn} = -\frac{1}{m} \sum_{i=1}^m \log \frac{\sum_{i \neq j, y_i = y_j, j=1 \dots m} e^{-\frac{1}{t}d(x_i, x_j)}}{\sum_{i \neq k, k=1 \dots m} e^{-\frac{1}{t}d(x_i, x_k)}} \quad Eq. 2.17$$

where the temperature t is a hyperparameter used to tune how concentrated the features are in the final representational space. In practice, the samples (x_i, y_i) do not have to be defined as (*feature, label*) pairs. Instead, they can be interpreted as positive pairs or negative pairs as shown in T. Wang & Isola, (2020). Here, positive pairs are often obtained by augmenting the same sample with random pre-processing techniques such as cropping, changing colours, etc. Let $p_{data}(\cdot)$ be the data distribution over \mathbb{R}^n and $p_{positive}(\cdot, \cdot)$ be the distribution of the positive pairs over $\mathbb{R}^n \times \mathbb{R}^n$ with following two assumptions: (i) the symmetry assumption where $p_{positive}(x, y) = p_{positive}(y, x)$ for all x and y pairs, and (ii) the matching marginal assumption where $\int p_{positive}(x, y) dy = p_{data}(x)$. The contrastive loss is defined as:

$$L_{contrastive} = \mathbb{E}_{\substack{(x, y) \sim p_{positive}, \\ \{x_i^-\}_{i=1}^M \sim p_{data}}} \left[-\log \frac{e^{\frac{1}{t}d(x, y)}}{e^{\frac{1}{t}d(x, y)} + \sum_i e^{\frac{1}{t}d(x_i^-, y)}} \right] \quad Eq. 2.18$$

where M is a fixed number of negative samples drawn from the data distribution.

Due to the relaxed definition of what contributes to positive or negative examples, this loss has been used in both supervised and unsupervised settings (T. Chen et al., 2020). In the unsupervised learning case, contrastive learning can help the model to learn to ignore different types of feature invariants such as color or noise through positive sampling of different data augmentations (Jaiswal et al., 2021). In the image representational learning case, a pre-trained supervised contrastive learning model has shown to produce better representations than a pre-trained unsupervised contrastive learning model (T. Chen et al., 2020). Instead of using only random augmentation, the supervised learning approach used ground truth labels in conjunction with augmentations to guide the positive and negative sampling process, leading to a more accurate clustering of different features. Under the mutual information interpretation, the contrastive loss function can also be extended to learn joint text and image representations by maximizing the mutual information between both modalities (Radford et al., 2021).

This task is also known as the image and text alignment task, and the loss function is known as image-text consistency loss that pushes the same visual and language concepts together (J. Zhang et al., 2019). For example, the Contrastive Language-Image Pretraining (CLIP) model trained contrastively on social media positive and negative

(*image, caption*) pairs can identify real pairs from false pairs (Radford et al., 2021). Here, the model essentially uses the InfoNCE loss function to map similar images and text close to each other in the joint embedding space, while pushing apart dissimilar image and text. In the VRD context, contrastive image-text learning can also be applied to visual relationship triples where the visual features of objects are aligned with the textual features of the triple (*subject, predicate, object*) labels. Peyre et al. (2019) also used this contrastive approach and proposed a two-stage model. In the first stage, a contrastive visual-language model is used to learn joint object, subject, and relationship embeddings. These embeddings are then used in the second stage model, dubbed as the Analogy Transfer (AT) network, to compose novel embeddings of a target relationship from existing relationship embeddings. For example, if the goal is to estimate the target embedding for “person ride cow”, the downstream analogy transformation network would aggregate all “Analogy Transformation” of similar pre-trained word embeddings such as “person ride horse” or “person ride bike” weighted by their relevance to the target embedding using language cosine distance score (Peyre et al., 2019). This aggregation process can also be thought of as a variant of the attention mechanism described in Section 2.1.2. While this application shows a potential application of contrastively learned embeddings to form novel compositions, it uses a second analogy transformation neural network and is highly dependent on the upstream contrastive visual and language network.

Unlike Analogy Transfer (Peyre et al., 2019), which trains two networks on the Visual Genome dataset (Krishna et al., 2017), our work uses BERT with self-attention operation alongside contrastive learning to learn joint visual and language embeddings in a single pipeline. For the VRD ranking task in Section 4.2, we applied two types of contrastive loss functions: (1) the triplet loss (hinge loss) and (2) the InfoNCE loss (negative log likelihood loss). In Section 4.3, we use the CLIP architecture with only the InfoNCE loss to measure the impact of large-scale pre-trained image and text models on visual relationship detection. Still, without the knowledge of how the scene is structured, the negative sampling process proved to be a challenging subproblem. This also raised another question on the visual relationship representational learning task: how can we preserve the structure of the final learned representation using contrastive learning? We believe that one potential answer lies in the manual knowledge engineering task of the input graph data.

2.4 Structured representation of a scene

One common criticism against deep learning models is that they lack the ability to represent compositionality. In a way, these models still learn spurious correlations that stem from dataset biases. We can observe these inherent biases within CNN’s activation map. For example, to distinguish “nurses” from “doctors”, the model uses the person’s hairstyle as a discriminative feature, showing that the model suffers from gender biases and still focuses on surface-level features to make its predictions (Selvaraju et al., 2017). Thus, simply making architectural changes and enforcing correlations via learning objectives such as object classification or missing part prediction cannot fully prevent the model from learning false correlates. Instead, the model needs to be pre-conditioned on a set of background knowledge, and the problem becomes how can a deep learning model acquire and use such background knowledge.

The resurgence of engineered structured representations such as Visual Genome (Krishna et al., 2017), Concept Net (Speer et al., 2018), and ATOMIC (Sap et al., 2019) aims to fulfil this need for background knowledge. Instead of using object labels as learning signals, these structure datasets and ontologies serve as a holistic guiding signal to the learning model. In a way, we believe the choice of knowledge, whether it be in the form of knowledge graphs or other structured representations, will become increasingly important. In this thesis, we focus on scene graph structures from the Visual Genome (VG) dataset to perform the visual relationship detection (VRD) task. It should be noted that the scene graphs in VG, created through crowd-sourced annotations, lack the systematic and formal structure typically associated with a factual ontology. Still, its creation process introduced and enforced a wider range of perspectives across objects (See Section 3.1.1 for more details), making it potentially less biased than unstructured image caption datasets. While focusing solely on one dataset puts a limit on what the trained model can achieve, we believe that scene-level sub-symbolic structures provided by these scene graphs is sufficient to test the limit of these deep learning models on their ability to learn compositions.

Section 2.4.1 describes the VG dataset in more detail and formalizes the scene graph definition. *Section 2.4.2* describes the visual relationship detection (VRD) task and motivates the use of graph representational learning to tackle the problem. *Section 2.4.3* and *Section 2.4.4* gives an overview of common approaches for VRD and SGG, with an emphasis on structural-preserving techniques. *Section 2.4.5* describes the general architecture used to extract visual features for the downstream VRD task. These five

sections serve to describe the VRD problem statement in more details and highlight existing approaches used to tackle the problem.

2.4.1 Scene Graph and Visual Genome

An image alone cannot fully represent all the syntactic relationships between entities. For the computer to understand “a man swinging a baseball bat” from an image, it needs not only to recognize the “baseball bat” and the “man” but also the direction of motion, the function of the bat, or the relationship between the man and the bat. Thus, to build such semantic description of one image without an exceeding amount of labeled data, background knowledge such as physics, motion, or a symbolic representation can be used. In this section, we limit the scope to static images and the *sub-symbolic* scene graph structure. Here, the *sub-symbolic* representation is defined as the visual connections between objects in the image through (subject, predicate, object) labels rather than well-defined rules and logic.

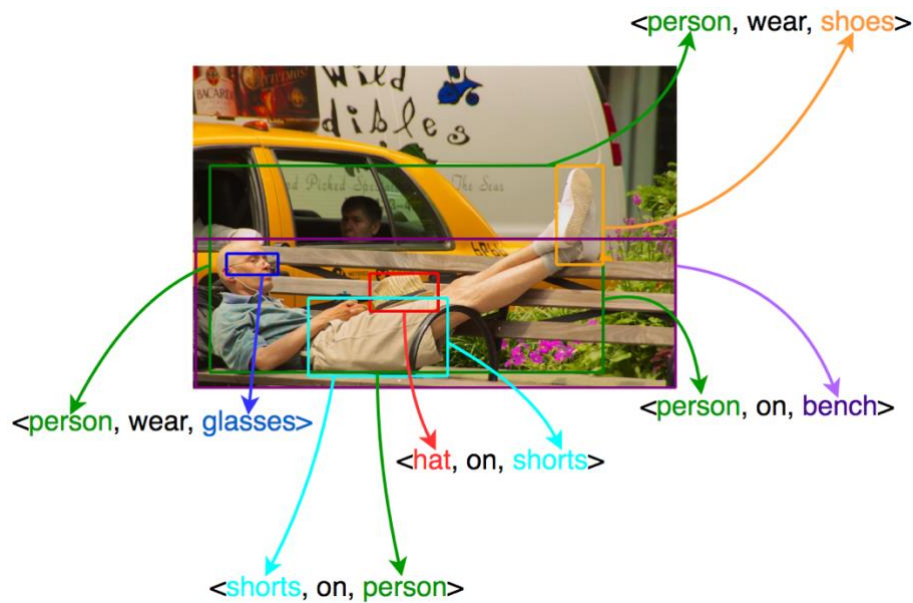


Figure 2.6: a visualization of the labelled visual relationship triples from an image. The above image is retrieved from the Visual Genome dataset.

A scene graph uses a graph-based formulation that explicitly models objects, attributes of objects, and relationships between objects (Johnson et al., 2015). This graphic representation does not only describe an image in greater details when compared to object classification labels, but also contains structured textual descriptions of the scenes in the image. Scene graph has been shown to improve the image retrieval task by comparing the structure of query sentences against the scene graph representations of images (Schuster et al., 2015). Moreover, it has been shown that scene graphs can enhance

computer vision tasks such as object localization because they can better represent relationships between entities (Krishna et al., 2018). Finally, scene graphs can model compositionality which allows the model to combine different objects and relationships to form a novel composition. As a result, when applied to downstream tasks such as VQA or VCR, a scene graph can also act as a knowledge base representation that can be queried upon (Hudson & Manning, 2019). Here, the visual relationships are represented in the RDF triples (*subject, predicate, object*) format, where *subject* and *object* are visual objects grounded to the image through 2D bounding boxes and *predicate* can be of a relation about distinct categories including action, spatial, preposition, comparative, and verb. Still, the VG dataset, being crowdsourced, has multiple drawbacks including inconsistent labels, missing annotations, or incorrect relationships. We will further explore the data-related issues Chapter 3.

More formally, given a dataset $\mathcal{D} = \{(I_1, G_1), \dots, (I_m, G_m)\}$ where $I_{n \in (1,m)}$ are the images and $G_{n \in (1,m)}$ are the corresponding scene graphs. Here, a scene graph $G_i = (O, E)$ consists of a set of objects O and edges E , where each element of $O = \{(o_1, \dots, o_k)\}$ is an object label grounded to image via a bounding box, and each element of $E \subseteq \{(o_i, r_{ij}, o_j) \mid o_i, o_j \in O, o_i \neq o_j\}$ is a triple. Note that each object $o_{i \in (1,k)}$ can be further decomposed into (b_i, y_i) where b_i is the bounding box coordinates and $y_i \in \mathcal{Y}$ is the class label of the object. A scene graph is an incomplete graph with missing labels, and each relationship $r_{ij} \in \mathcal{R}$ is a predicate category between the objects. There are three scene graph generation tasks with increasing levels of difficulty:

1. The visual relationship detection task, also known as the predicate classification task (PredCls), aims to find the missing relationship when are given the object pairs' ground truth bounding boxes and labels.
2. The scene graph classification task (SGCls) aims to predict the relationship as well as the object categories of the subject and the object in every pairwise relationship when given only the localized bounding boxes without the labels.
3. The scene graph generation task (SGGen) aims to construct the entire scene graph when given only the image.

In Chapter 4, we assume that the object detection task is a solved problem given the current state of the art and focus solely on visual relationship detection or PredCls. This assumption allowed us to effectively evaluate the model's ability to represent and predict visual relationships rather than object detection. In practice, given the small and

unbalance VG dataset, object detection remains a large bottleneck for scene graph generation systems. Section 2.4.2 below summarizes the common approaches for tackling the VRD task.

2.4.2 Visual Relationship Detection

Given the success of deep neural network architectures for the multiclass object classification task, multiple works (C. Lu et al., 2016; R. Yu et al., 2017; Zellers et al., 2018) have extended ANN model to build visual relationships classifier $p(r_{ij}|o_i, o_j)$. However, different from the multi-class object classification task, visual relationships depend not only on the visual features localized by the objects' bounding boxes (b_i, b_j) but also on the textual features given by the objects' labels (y_i, y_j). Thus, it is a common strategy to construct not only a CNN-based model to estimate visual likelihood $p^{visual}(r_{ij}|I, b_i, b_j)$ but also a separate network to estimate textual likelihood $p^{language}(r_{ij}|y_i, y_j)$ (C. Lu et al., 2016). These two models are then combined to get the final predictive model, $p(r_{ij}|I, o_i, o_j)$. To simplify the modeling framework for the classification case, we combine both visual and textual models into a single EBM model parameterized by θ . We can construct the general likelihood for the relationship class as:

$$p_{\theta}(r_{ij}|I, o_i, o_j) \approx \frac{e^{-\frac{1}{t}E_{\theta}(r_{ij}, I, o_i, o_j)}}{\sum_{r \in \mathcal{R}} e^{-\frac{1}{t}E_{\theta}(r, I, o_i, o_j)}} \quad Eq. 2.19$$

which can be learned through the following negative likelihood loss:

$$L_{VRD} = \sum_{i=1}^m -\log \left(\frac{e^{-\frac{1}{t}E_{\theta}(r_{ij}, I, o_i, o_j)}}{\sum_{r \in \mathcal{R}} e^{-\frac{1}{t}E_{\theta}(r, I, o_i, o_j)}} \right) \quad Eq. 2.20$$

While this simplified model and loss function underpin multiple works (Y. Li et al., 2017; Zellers et al., 2018; B. Zhuang et al., 2017), the definition comes with a few assumptions. First, the above loss function does not explicitly account for the underlying RDF structure of the scene graph and other known priors, which has proven crucial for VRD (Liang et al., 2018). Second, the parameterized energy model does not reflect the diverse architectural choices that can be customized for the given task. Third, the above loss function is applied under the classification setting, which assumes that each object pair can only have one decision. However, this assumption highly depends on the downstream use case for the estimated model. If the scene graph were to be designed for a well-defined and specific use case (e.g., spatial relationship classification under a

specific frame of reference for robot navigation), then making one final decision for each pair would be valid. However, visual relationships from the VG dataset are diverse and wide-ranging, making it difficult to predict only one predicate without additional context. Thus, with a fixed dataset, we argue that a better approach to tackle the VRD problem is to apply the EBM framework to learn a general representation that can be transferred to other downstream tasks.

To tackle the lack of structuring with the multi-labelled classification objective, multiple works have interpreted the VRD task as a ranking problem and aim to learn a representational space that preserve the structure of the graph. In a way, the observation made by these approaches is that the visual relationship prediction task is dependent on the surrounding context, and choice of the context is influenced by the choice of the graph structure. The work in Chapter 4 also uses this observation and incorporates local contextual information via a knowledge graph embedding technique to predict visual relationships. In general, the approaches taken by these works can be organized into two categories: (1) the engineering of loss objectives, and (2) the customization of the neural network architecture.

2.4.3 Engineering Loss Objectives for VRD

Since the VRD problem is framed as a ranking problem with an emphasis on representational learning, the final loss objective used to train the model has to be modified to accommodate the goal. One approach is to use additional auxiliary metric learning objectives (e.g., the triplet margin loss) that preserve a certain assumed structure. For example, Lu et al. (2016) assumed the word2vec embedding space contains visual triple knowledge via its individual component and added an auxiliary clustering loss that projects similar word relationships (*subject-predicate-object*) closer to one another. However, the assumed loss is highly dependent on the underlying word2vec embeddings or the language prior that exists within larger pre-trained word2vec’s corpora. Other work, including ours, assumed that the labelled scene graph structure is similar to that of a knowledge base and aimed to preserve the first-order structure of the knowledge base using the *translational constraint* (Hung et al., 2020; Liang et al., 2018; H. Zhang et al., 2017). Originally used for knowledge base completion, the *translational constraint* aimed to preserve the first-order hierarchical structure of the RDF triple (h, r, t) in a set S by enforcing the constraint $EMB(h) + EMB(r) \approx EMB(t)$ (Bordes et al., 2013). To do this, Bordes et al. (2013) used the hinge loss (or triplet loss) function with negative sampling. Here, negative sampling aims to curate a set of corrupt triples S' by replacing

the head and tail entities with random entities, annotated as \bar{h} and \bar{t} . Given that the embeddings are learned using an EBM model, the TransE loss over the entire knowledge base is defined as:

$$L_{hinge} = \max(0, m + d(l_h + l_r, l_t) - d(l_{\bar{h}} + l_r, l_{\bar{t}})) \quad Eq. 2.21$$

$$\mathcal{L}_{TransE} = \sum_{(h,r,t) \in S} \sum_{(\bar{h},r,\bar{t}) \in S'} L_{hinge} \quad Eq. 2.22$$

where m is a fixed margin, l_x is the embedding of x , and $d(\cdot, \cdot)$ outputs the distance between its inputs. Inspired by the TransE knowledge base completion approach, Zhang et al. (2017) also enforced the TransE constraint on the (*subject, predicate, object*) visual relationship triples. However, unlike the factual triples in the knowledge base, visual relationship triples consist of labelled 2D image regions, making it difficult to execute the same random negative sampling strategy. Thus, under the observation that most labelled triples have only one relationship, VTransE (Zhang et al., 2017) used the negative likelihood loss with a distance-based EBM model:

$$\mathcal{L}_{VTransE} = \sum_{(h,r,t) \in S} -\log \left(\frac{e^{-d(l_r, l_h - l_t)}}{\sum_{(h',r',t') \in S} e^{-d(l_{r'}, l_{h'} - l_{t'})}} \right) \quad Eq. 2.23$$

where $d(l_r, l_h - l_t)$ computes the cosine similarity between the EMB(predicate) and EMB(subject) - EMB(object). However, using such classification objective ignores the co-occurrence of predicates. Thus, Liang et al. (2018) instead proposed the use of adaptive margin hinge loss with a simple negative sampling approach. Here, the negative sampling process selects all non-labelled triples for every labelled triple.

$$\mathcal{L}_{DSR} = \sum_{(h,r,t) \in S} \sum_{(\bar{h},r,\bar{t}) \in S'} \max(0, \Delta + d(l_{\bar{r}}, \phi_{\theta}(\bar{h}, \bar{t})) - d(l_r, \phi_{\theta}(h, t))) \quad Eq. 2.24$$

where Δ is the adaptive margin and $\phi_{\theta}(\bar{h}, \bar{t})$ calculates the joint visual and spatial features of the head and tail entity. Under the observation that the labelled relationships set is incomplete and the negative sampling technique can select wrong negative examples, the DSR loss applied an adaptive margin, Δ , that is more lenient to the incomplete annotation with a high counting prior probability (Liang et al., 2018). While we also use the hinge loss and translational loss in Chapter 4, we do not apply the described adaptive margin. Instead, we specified a static margin m and propose the use of an adaptive negative

sampling technique that directly tackle the wrong negative samples selection problem. This technique is further described in the implementation section in Chapter 5.

2.4.4 Message Passing Approaches for VRD

Another approach to preserve the structure of the scene graph in the final learned representation is to modify the deep neural network architecture. The most direct way is to control the direction of information flow between the embedded features using a graph structure layout and a process called *message passing*, which decides how to aggregate information from the input nodes to get a refined output feature representation. Here, a graph-based neural network model is often used, and the model’s design depends on the choice of vertices, edges, and a message passing algorithm. The entire design process can be summarized into three steps. The first step involves the construction of a graphical structure or layout, which includes the choice of vertices, edges, and their interactions. The second step is the feature extraction step which aims to transform a set of image and text inputs into a distributed embedding. The third and final step involves the choice of the message passing algorithm. In general, the messages are the learned input features and the choice of message passing technique highly depends on the layout structure of the graph.

Two of the most popular layouts for scene graphs are the bipartite graph and the hierarchical tree structure. In Xu et al. (2017), the scene graph is interpreted as a directed bipartite graph that consists of two sets of vertices: *object vertices* and *relationship vertices*, where the vertices are the features extracted from the objects and labelled triples. To facilitate the interaction between the two sets of object and relationship vertices, Xu et al. (2017) used a RNN-based model called the gated recurrent unit (GRU) to iteratively aggregate and refine information from neighbour vertices, forming a chain-like structure. Different from the bipartite graph, a tree structure represents the visual objects in a hierarchical manner. Tang et al. (2018) proposed VCTREE, the first architecture that applies a tree layout to the scene graph generation and VRD tasks. Here, Tang et al. (2018) add an additional step to construct a maximum spanning tree using Prim’s algorithm based on the input object labels and scene graph. However, Tang et al. (2018) observed that the resulting spanning tree consists of only a “hard” hierarchical layout, which added an additional constraint to the direction of the message passing algorithm and prevented the model from gathering relevant contextual information from parallel nodes when needed. Thus, the generated multi-branch tree is further refined into a binary tree by changing non-leftmost edges into right branches, allowing nodes from the same level to

interact with one another. The final representation is then learned using a bi-directional TreeLSTM to encode the visual contexts (Tang et al., 2018).

Still, these two structured layouts or inductive biases are simply heuristics engineered under the assumption that the given layout is beneficial to downstream visual relationship detection or visual reasoning tasks. In practice, due to the diverse appearances and interactions between visual objects, quantifying such an assumption remains difficult. Thus, instead of constructing an alternative structured layout, other work directly interpreted a scene graph as a heterogeneous graph with multiple types of vertices and edges, where vertices can be larger visual composites instead of individual subject, predicate, and object parts. For example, Li et al. (2018) proposed Factorizable Net, which factorized the scene graph into smaller subgraphs and learned individual subgraph embeddings before refining them into a unified representation for the given task. Similarly, Zellers et al. (2018) observed that there exist recurring patterns within the larger subgraphs (or motifs) in the labelled scene graph, and proposed the use of contextualized encoders (e.g. bi-directional LSTM) to learn representations from not only the local nodes but also the global nodes.

Other work has also extended the graph representational learning process using other neural network architectures such as the Transformer or Graph Neural Network (GNN) (S. Ji et al., 2021; Scarselli et al., 2009). While we do not explore the details of GNN architecture in this work, the idea of GNN is based on that of message passing, which aggregates features from incoming neighbouring nodes to learn a refined feature for the target node. This principle is then recursively applied from the first-level nodes, second-level nodes, or the entire subgraph depending on the specification of the graph algorithm, allowing a more expressive global representation of the target node. For example, Mi & Chen (2020) applied the Graph Attention Network, a variant of the Graph Neural Network, to learn the hierarchical embeddings of subgraph structures starting from object-level features to triplet-level features to scene-level features. Similarly, Lin et al. (2022) further decomposed the *(subject, predicate, object)* triplet into four types of compositional pairs: *(subject, predicate)*, *(subject, object)*, *(predicate, predicate)*, *(predicate, object)*, and proceeded to learn pair-level features before performing message passing between these features using a GNN.

The Transformer’s self-attention layer can also be thought of as a special case of the message passing algorithm, where the structured layout of the input embeddings is a

fully connected graph and the message passing algorithm is the self-attention operation. As the number of self-attention layers increases, the amount of contextual information aggregated from neighbouring features also increases for the target feature. However, the use of a fully connected graph can be redundant. While an out-of-the-box Transformer architecture has fixed structure and connections, the input or output embeddings can be refined using an upstream or downstream neural network to make the learning more efficient. For example, inspired by the bipartite graph interpretation and motivated by the Transformer-based detector (X. Zhu et al., 2021), Li et al. (2022) used CNN-Transformer sub-networks to capture the joint visual and language features of the set of entity and predicate nodes. These entity and predicate features are then further refined in a downstream graph assembling module, which learns two adjacency (distance) matrices for the (subject, predicate) and (predicate, object) pairs, and the top K relationships are selected from each matrix to construct multiple (subject, predicate, object) triples.

In general, the engineering of neural network architectures for VRD task makes use of an assumed graph structure and incorporates contextual information from both the local atoms (subject, predicate, or object) and the global graph through a message passing technique. This engineering of the graph structure and learning algorithm yield overall improvement to the model’s performance. However, the majority of these assumptions rely on biases observed in the scene graph benchmark and the learned representation cannot be easily transferred to another downstream task. In this work, we only incorporate the first-order information from the object’s immediate neighbour via the translational embedding. While such approach does not capture the full contextual information that exists between the visual entities in the image, it provides a good starting point for exploring the application of knowledge graph embedding to visual relationship triples from the contrastive learning standpoint.

While we only summarized the most popular approaches to the VRD task in this section, there are alternative ways to reconfigure the neural network model to make it more suitable for the given task and achieve a higher benchmark result. So far, the discovery of neural architectures for VRD has mostly been done manually. However, there exists a subfield called Neural Architectural Search which aims to *learn how to learn* by searching for more efficient architectures and loss functions (Pengzhen et al., 2021). While this topic is not covered under our work, we believe that future work can explore the potential of automating neural network architecture design for a more efficient model under the benchmark data.

2.4.5 A General CNN-based Architecture for VRD

While there are many proposed structured layouts with various message passing architectures over the years, the feature extraction step from the given image and labelled scene graph remains consistent. One of the most popular architecture designs for VRD and SGG is to couple an object detection network such as the Faster-RCNN with a customized visual relationship detection network. The general architecture diagram for this design is shown in Figure 2.7. Here, visual representations are extracted from the Faster-RCNN’s backbone pre-trained on ImageNet or COCO datasets. There are two approaches to constructing the general architecture: (a) coupled CNN backbone network and (b) decoupled CNN backbone network.

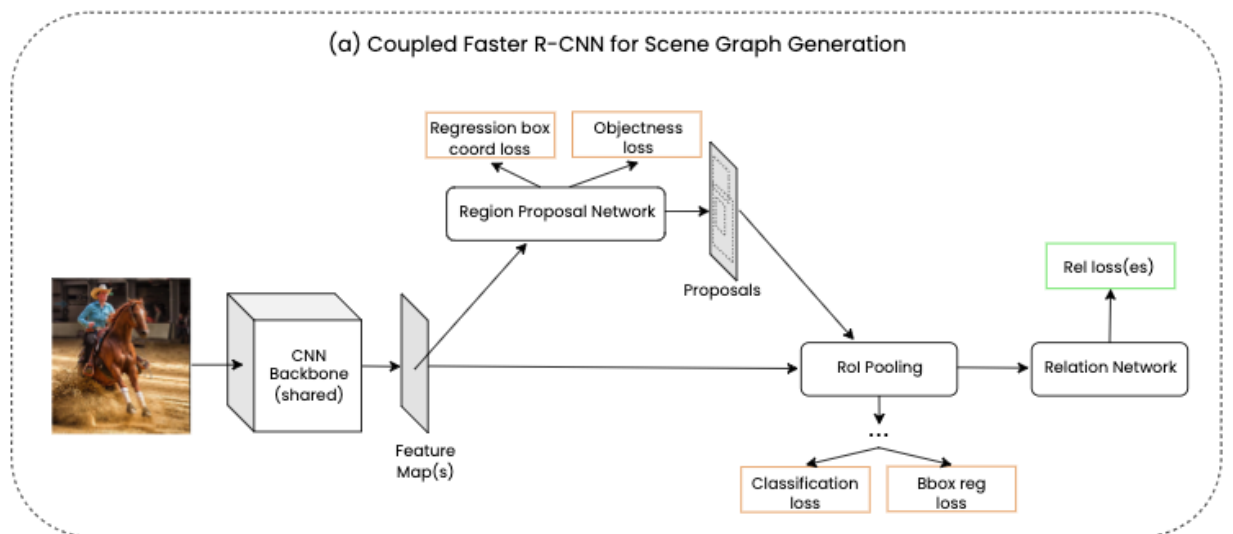


Figure 2.7a: The general coupled Faster R-CNN architecture for scene graph generation.

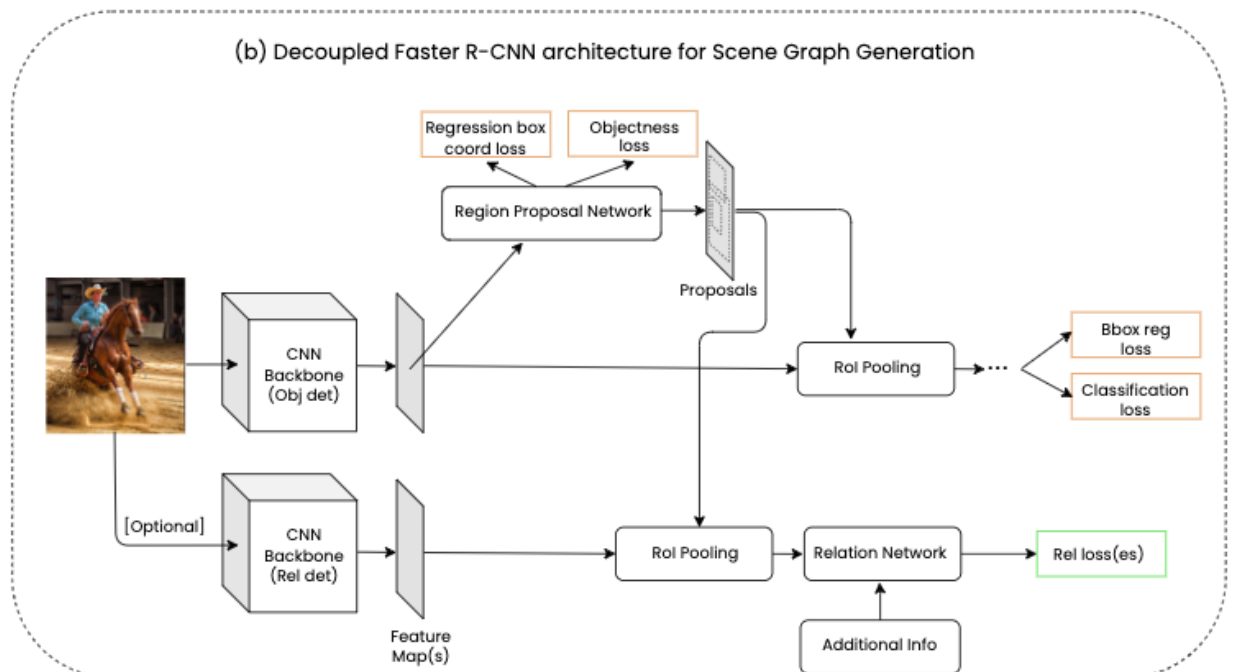


Figure 2.7b: The general decoupled Faster R-CNN architecture for scene graph generation.

Through observations, Han et al. (2021) has shown that the decoupled approach performs better in the VRD and SGG tasks. One possible reason for these observations is that each of the two CNN backbones in the decoupled network serves only one functionality: one backbone network fine-tuned on the Scene Graph dataset for the object detection task, and one backbone network for the visual relationship detection network trained on VRD or SGG tasks (Han et al., 2021). By separating the functionality for each network, the decoupled backbone approach allows gradients to backpropagate more effectively since VRD and object detection task does not necessarily require the same underlying features.

Since both of these general architectures are highly inspired by the design of Faster-RCNN, we can apply variants and other engineering priors to its individual component. For example, instead of extracting only one feature map from the CNN backbone, a feature pyramid network (FPN) can also be used to extract multi-resolution feature maps. Similarly, different regions of interest pooling techniques such as RoiPool or RoiAlign can also be applied. The Region Proposal Network (RPN) can also be adapted to suit the need of VRD. For example, J. Zhang et al. (2017) introduced a relationship proposal network (RelPN), which combined the union proposals from three different subject, predicate, and object RPNs to predict the “relationshipness” of the proposed bounding regions. Finally, the relation network and loss objectives can be customized for the representational learning task as introduced in the previous sections. The simplest relation network for VRD is to add an ANN that inputs the visual features to classify the predicate label, while a more complex approach incorporates an additional feature refinement step, which refines the visual features in correspondence with other engineered priors or external information. While we do not apply all the shown techniques in this work, exploring these engineering priors help us understand how neural networks can be designed to better fit the task at hand.

In Section 4.2, we first fine-tuned the object detection network in the decoupled Faster-RCNN model on the given benchmark dataset, and then train a duplicated backbone network to extract visual features from the image regions for the VRD task.

2.5 Discussion

In this chapter, we explore two contemporary deep learning architectures called the Convolutional Neural Network (CNN) and the Transformer, identifying assumptions and motivations behind the design of each architecture. We then described the

applications of these architectures in both image and text modalities and give an overview of how to apply them to learn joint visual and language representation. Afterwards, we explored the motivations and use cases of contrastive learning under the energy-based model framework. Finally, we briefly highlighted the importance of background knowledge with an emphasis on a sub-symbolic structure called the scene graph and described different existing techniques to get a better intuition on how to build a visual relationship detection system.

In Chapter 4, we combine the described neural network and loss engineering techniques to build a complete VRD system. Through a clear understanding of the design principle of these neural networks, we can select appropriate neural network architectures and engineer additional pathways to perform VRD. Through contrastive learning, we can guide the model representational learning process for both visual and language while incorporating relevant knowledge graph embedding loss to preserve the hierarchy of the visual relationship triples. In the next chapter, we explore the Visual Genome dataset in more detail and look at the VRD through the data-centric standpoint. We also describe the common issues highlighted by the literature and give a qualitative overview of the data collection issues which have demonstrated to be the one of the key bottlenecks for VRD systems.

Chapter 3

Datasets and Evaluation Metric for Visual Relationship Detection

In this chapter we describe the common problems that affect the visual relationship detection task through a data-centric lens. Instead of focusing on the architectural and loss design like the previous chapter, we redirect our focus to the data collection process of the Visual Genome (VG) dataset and take a closer look at the data samples. In *Section 3.1*, we describe the data collection process for the VG dataset, its properties, and the common long-tail problem. In *Section 3.2*, we describe existing evaluation metrics that come with the VG-derived benchmarks and propose an alternative metric using Youden’s J Statistics (Youden, 1950).

3.1 The long-tail problem of Visual Genome dataset

In statistics, a *long-tailed distribution* is a distribution that has a long tail that tapers off at the end. Here, a dataset is said to have a long-tailed distribution if only a small number of class labels have a massive number of samples while others are associated with only a small number of samples. Such class imbalance poses a huge challenge to the model training process, and in many real-world scenarios, it is also the quality and not just the quantity of data labels that determines the success of the model. The VG dataset, containing 3.8 million object labels and 2.3 million relationship labels, also suffers from this long tail problem. While it is easy to identify and dismiss the problem by assuming that the dataset is static, it is still important to revisit the original motivation for the Visual Genome dataset, analyse it, and get a better intuition for the root cause. In a hindsight, this understanding allows us to find novel and appropriate solutions for not only the neural network engineering problem but also the data collection problem in future work.

3.1.1 From Unstructured to Structured Data

Deep learning’s rise is intrinsically linked to the scale and complexity of the datasets, starting from the smaller hand-labelled datasets for image classification such as MNIST (Cireşan et al., 2011) and Caltech 101 (Fei-Fei et al., 2006) to the larger and more complex datasets for object detection such as Pascal VOC (Everingham et al., 2010) and ImageNet (Deng et al., 2009). However, as deep learning models become larger and more sophisticated, object-centric recognition tasks are no longer enough to test the limits of the technology. To push the boundary of these models, newer “AI-complete” tasks and benchmarks with an emphasis on multimodal learning were proposed, including image captioning and visual question answering (VQA). To this end, newer image and text datasets such as MS-COCO (Lin et al., 2015) and VQA 1.0 (Antol et al., 2015) were developed. Still, these benchmark datasets, for the most part, remain unstructured, free-form and open-ended. For example, image captioning aims to generate free-form sentences from the input image, and VQA aims to give an answer to an open-ended question with respect to the input image. However, humans’ natural language, in free-formed format, has its own biases stemming from their beliefs and background. For example, annotators tend to describe salient features and provide high-level summaries relevant to the given task, leaving out implicit details.

To tackle the biased annotation problem, Krishna et al. (2017) focused on curating a general-purpose representation of the visual scene without bias toward a particular task. To this end, a more sophisticated data collection pipeline was introduced to generate the VG dataset using a combination of human annotators and automation techniques. To facilitate the curation of coarse-to-fine annotations, the data collection process required human annotators to provide descriptions of numerous image regions. However, human annotators are biased and tend to describe the most contextually relevant relationships, which led to overlapping or duplicating sentences. The pipeline subsequently removed uninteresting or duplicated sentences by comparing them with existing sentences using the BLEU metric or the n-gram scores (Papineni et al., 2002). These image region descriptions are then parsed using a semantic parser to get the region graphs, which are then combined to get the final scene graph representation. While the proposed scene graph dataset gives a more detailed view of the image than its predecessors, the free-formed region descriptions are biased and still largely depend on natural language which leads to ambiguity in the final parsed visual relationships triples.

This ambiguity is an artifact of natural language and is partially caused by polysemy, which is pervasive in natural language and often affects both the content and function of the word (Quilty-Dunn, 2021). For example, the word ‘on’ in the following relationships (*wheel, on, train*) and (*person, on, sidewalk*) has two distinct senses: a part-whole sense and a spatial preposition sense (Rodrigues et al., 2020). To tackle this problem, Krishna et al. (2017) added the final canonization process which maps individual scene graph elements such as the objects and relationships to their corresponding WordNet’s synonym sets (synsets). Here, these synsets do not explain what the concepts are, but merely signify that the concepts exist (Miller, 1995). However, the automated process still results in noise in the final representation as a result of the noisy WordNet’s fine-grained sense distinctions (Krishna et al., 2017). This challenge when combined with the sparse nature of most scene graph labels prompted the creation of smaller datasets with denser classes. To this end, C. Lu et al., (2016) proposed a smaller subset of the VG dataset called VRD, containing 5000 images and 70 predicate classes. Similarly, Xu et al., (2017) proposed a distilled version of the Visual Genome dataset called the VG150 with 150 object classes and 50 predicate classes. Finally, Peyre et al., (2017) proposed a version of the Visual Genome dataset that emphasizes rare visual relationships called UnRel. Still, these smaller benchmarks are often automated using a small number of hand-made rules or taxonomies, trading off accuracy for efficiency. While most benchmarks contain an acceptable level of inaccuracies, tackling the long-tail problem remains a huge challenge and the problem of how to collect high quality structured knowledge in an efficient manner remains open, and this data problem is one of the limiting factors of the contemporary deep learning approach in this work.

In the *Section 3.1.2* below, we will give a brief analysis of the VRD dataset and the VG150 dataset, which were used to train and test our models in Chapter 4.

3.1.2 Statistics of VRD and VG150

While the VRD and VG150 have reduced the number of relationships in the original Visual Genome dataset from over 40,000 unique predicates to 70 unique predicates and 50 unique predicates respectively, the predicate class distributions for both datasets still suffer from the long tail problem due to the nature of the task. A similar phenomenon is observed for the subject and object classes, which have been reduced from 76,000 unique classes to 100 unique classes in VRD and 150 unique classes in VG150. This long tail problem is further amplified when the individual parts are combined to form the (*subject, predicate, object*) triple classes (See Figures 3.1, 3.2, and 3.3). Through

counting, we also observed that the top 16 visual relationships in VG150 account for 93.56% of all the predicate labels.

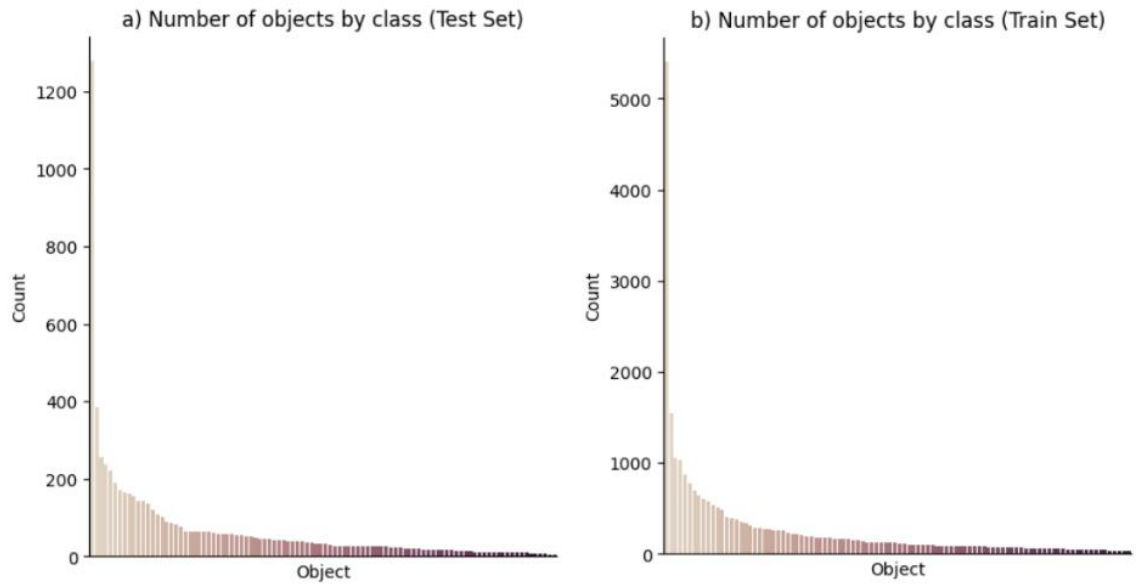


Figure 3.1: Frequency counts of different objects in the (a) test and (b) train set in VRD dataset (C. Lu et al., 2016).

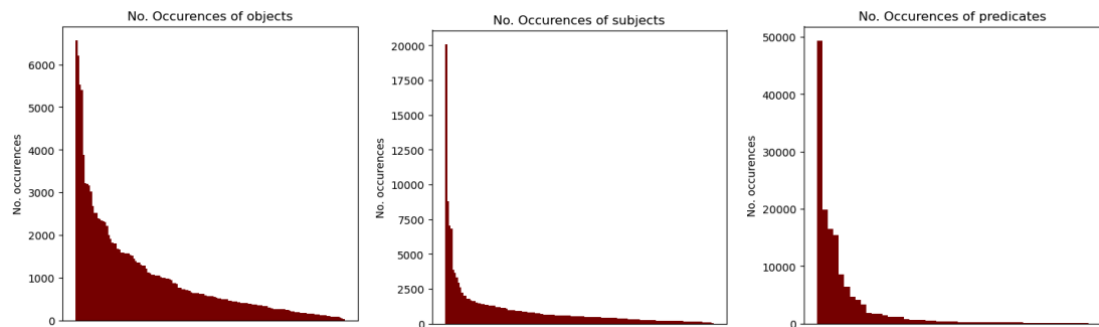


Figure 3.2: The long tail distribution of the number of subjects, predicates, and objects in the VG150 test dataset.

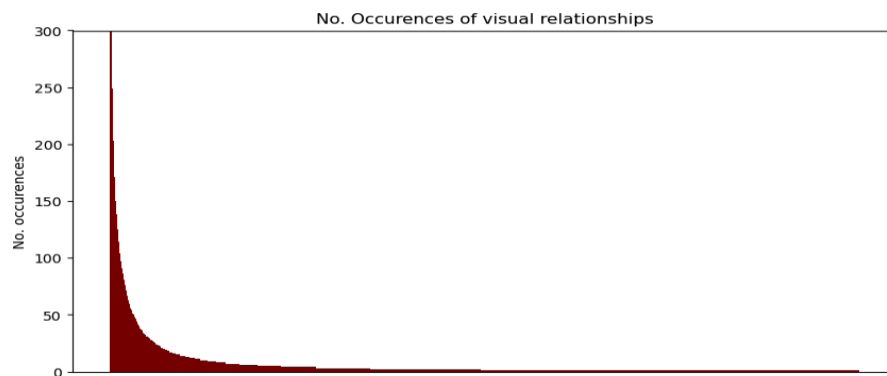


Figure 3.3: The long tail distribution of the visual relationship triples in the VG150 dataset.

Beyond the long tail distributions, the datasets contain relationships that cannot be predicted using solely visual and textual information from the dataset. For example, the predicate ‘watching’ represents an action that requires background information such as the function of the human’s eye. Similarly, the predicate ‘between’ expresses the spatial position of an object relative to two or more objects, which cannot be labelled using a single (*subject, predicate, object*) triple. Another aspect of the curated datasets is that the predicates with similar meanings are duplicated such as ‘on’, ‘over’, and ‘above’. While the listed predicates do not necessarily represent the same spatial relationship and the use case of each predicate depends on the context, any of the above predicates can be interpreted as the correct answer in many cases, making it difficult for the learned model to select one over the other using visual information and visual relationship triples alone. Finally, without a specific frame of reference when labeling the images, the labelled preposition predicates can be inconsistent across the labelled data, making it difficult for the model’s loss function to converge. This inconsistency in data labels is observed in our qualitative evaluation results in Section 5.3.

While these problems can be viewed as characteristics of language, they are often ignored in the analysis of the VRD results. Here, we identify the flaws stemmed from the nature of the VRD task, which helped us understand the weaknesses of the common benchmarks while highlighting a need for a better and well-defined benchmark for visual relationships. While a combination of human labels and automation techniques are still needed, perhaps the uses of annotated English descriptions and semantic parser are no longer enough. Understanding the limitations of the VG-related benchmarks also allows us to evaluate the true performance and limitations of the visual relationship detection model in the context of real-world photos. Section 3.1.3 below briefly describes the techniques used to tackle the long-tail problem.

3.1.3 Tackling the long-tail problem

One way to tackle the long-tail problem in VRD is to directly address the unbalanced distribution of the predicate classes through *re-weighting* or *re-sampling* approaches. In the re-weighting approach, the final loss function assigns adaptive weights or margins (Liang et al., 2018) for different predicate classes depending on the number of occurrences of the classes (Cao et al., 2019). Different from the re-weighting approach which focuses on engineering the loss function, the re-sampling approach focuses on smoothing the data distribution by performing a biased selection of the predicate classes. While both approaches can be beneficial to the VRD task when evaluated under

the mean Recall metric (see Section 3.2), these approaches introduce biases to the final learned model, making the model’s predictions deviate from the natural data distribution and penalizing the model’s ability to generalize when given a test set with similar distribution to the train set. In the context of our work, where the goal is to build a general representation of the visual relationships using contrastive learning, we aim to model the original data distribution while allowing the model to perform the VRD task. Thus, we did not apply any of the above re-balancing techniques which improve the model’s performance on one evaluation metric but penalize its performance on another evaluation metric (Section 3.2). Instead, we used a supervised negative sampling technique that depends on the original dataset distribution. We described this negative sample approach in Section 5.1.2 in more detail.

Section 3.2 below describes the benchmark evaluation metrics, allowing us to understand the trade-off between different evaluation techniques and how they affect the model’s design decision for the VRD benchmark.

3.2 Evaluation Metrics

Given the unbalanced nature of the datasets for the VRD task, a balance evaluation metric is needed to have a more holistic overview of the model’s performance. While the evaluation metric should not be biased towards the head classes of the long tail distributions, it should not over penalize the models for selecting the most prevalent class that follows the distribution. The sections below give an overview of the commonly used evaluation metrics for the machine learning tasks with an emphasis on object detection and visual relationship detection. In Section 3.2.1, we provide a comprehensive overview of the widely used Recall, Precision, and F-measure, highlighting common issues for each metric. In section 3.2.2, we describe the visualization techniques such as Precision-Recall Curve and ROC Curve and propose the use of an ROC-based metric called *Informedness* as a measure.

3.2.1 Precision, Recall, and F-measure

One of the most used metrics for the binary classification task in machine learning is *Recall* (Eq 3.1) and *Precision* (Eq 3.2). Here, *Precision* computes the fraction of the number of correctly predicted positives over the set of all positive predictions, and *Recall* (also known as *Sensitivity*) computes the fraction of the number of correctly predicted positives over the set of ground truth labels. A simple way to conceptualize *Precision and*

Recall is to use a confusion matrix (See Table 3.2), which subdivides a prediction made by the model into four quadrants: (1) the *true positive* (TP) quadrant that counts the number correctly predicted positive, (2) the *true negative* (TN) quadrant that counts the number of correctly predicted negatives, (3) the *false positive* (FP) quadrant that counts the number of incorrectly predicted positives, and (4) the *false negative* (FN) quadrant that counts the number of incorrectly predicted negatives.

		Prediction	
		Positive	Negative
Ground Truth	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Table 3.1: **Confusion matrix for binary classification**

In the VRD multiclass classification context, *Recall@n* (short as $R@n$) is the most used metric for evaluation, which computes the number of top n correctly predicted predicates over all relations with that label. In the original VRD work, $R@n$ is computed using the *micro-averaging* approach which aggregates the TP and FN counts for all classes prior to computing Recall. This may lead to confusion since such micro-averaging Recall is equivalent to *Accuracy* in the multiclass classification context (Eq. 3.3). Here, a common argument for using such $R@n$ metric is that it is more lenient to missing relationships that were not annotated. However, given that both the VRD and VG datasets are dominant by a few predicate classes, using the $R@n$ metric alone does not reveal how performant the model is on rare classes. As a hypothetical scenario, if a test set with fifty predicate classes were to contain one-hundred examples and eighty of those examples were labelled as ‘on’, a classifier that always guesses ‘on’ would have an overly high Accuracy of eighty percent while yielding zero Recall for the remaining forty-nine classes.

Besides $R@n$, another popular metric used to evaluate VRD is *mean Recall@n* (short as $mR@n$). Different from $R@n$ which uses the micro-averaging approach, $mR@n$ computes one-vs-all *Recall* for each predicate class and takes their average in a process called *macro-averaging*. While $mR@n$ tones down the optimism reflected by the $R@n$ metric, it still uses binary Recall at its core which can still be misleading. For example, the experimenter can optimize for Recall by lowering the model’s threshold for each class to reduce False Negatives at the cost of increased False

Positives and lower Precision. Moreover, given that the experimental dataset is static, mR@n can be considered as an overly pessimistic evaluation metric since it gives equal importance weight to every predicate class, ignoring both the skewed dataset distribution and the intentionally introduced bias by the experimenter.

$$Recall = \frac{TP}{TP + FN} \quad Eq\ 3.1$$

$$mean\ Recall = \sum_{i \in (0,n)} \frac{TP_i}{TP_i + FN_i}$$

$$Precision = \frac{TP}{TP + FP} \quad Eq\ 3.2$$

$$Recall@n = Accuracy@n_{multiclass} = \frac{correct\ classifications}{all\ classifications} \quad Eq\ 3.3$$

The alternative is the Precision metric, which measures the ratio of correct positive predictions to the total number of positively predicted classes. Here, optimizing the model for high precision reduces the number of false positives and makes the classifier more consistent across different test sets. However, as a consequence of this optimization, the classifier often becomes more conservative in its prediction, penalizing Recall in the process. For example, an experimenter can increase the threshold of the classifier to optimize for Precision, reducing the number of false positives while increasing the number of false negatives, resulting in a lower Recall. One of the most popular unified metrics used to tackle such trade-offs is F-measure, which computes the harmonic mean of both Recall and Precision. Although the F-measure shown in Equation 3.3 gives equal weight or importance to both Recall and Precision, practical applications of F-measure often assign different weights to each depending on the goal of the targeted experiment.

$$F_1 = \frac{2}{Recall^{-1} + Precision^{-1}} = \frac{2TP}{2TP + FP + FN} \quad Eq\ 3.3$$

3.2.2 Precision-Recall Curve and ROC Curve

Still, observing the model's performance using a single F-measure number can be misleading as it does not indicate whether the model is performing well on either Recall or Precision. One approach to tackle this issue is by plotting the *Precision-Recall Curve* (PRC) from information retrieval, which shows the trade-offs between the two metrics at

different thresholds. A complementary metric that is often used with PRC is *Average Precision (AP)*, which computes the average of Precisions across the thresholds. In the multi-class object detection context, *mean Average Precision (mAP)* first computes the Average Precision for each object class and takes their mean. While such a metric makes sense in the object detection context, it is still considered as a pessimistic evaluation metric for visual relationship detection due to the missing annotations that stemmed from human biases (C. Lu et al., 2016).

Another common trade-off metric pair that experimenters use to measure the model's performance is the *True Positive Rate (TPR)* and *False Positive Rate (FPR)*. Here, TPR is equal to Recall and is calculated by computing the fraction of correctly predicted positives over the number of ground truth positives. Similarly, FPR is calculated by computing the fraction of incorrectly predicted positives over the number of ground truth negatives. Unlike the PRC which originates from information retrieval and helps visualise the trade-offs between Precision and Recall, the *Receiver Operating Characteristic Curve (ROC)* originates from information theory and plots the trade-off between TPR and FPR which shows us the classifier ability to distinguish between positive and negative classes. Different from the PRC which aims to maximize both Precision and Recall, ROC aims to maximize TPR while minimizing FPR. With ROC, an experimenter can compare the model's performance to chance or random guessing, indicated by the TPR equals FPR diagonal line. Such measurement is called *Youden J's Statistic* in the binary case or *Informedness* in the multi-classes case (D. M. W. Powers, 2020), which specifies the probability that a prediction is informed in relation to the condition versus chance (see Eq 3.6).

$$TPR = \frac{TP}{TP + FN} \quad \text{Eq 3.4}$$

$$FPR = \frac{FP}{FP + TN} \quad \text{Eq 3.5}$$

$$\text{Youden J's Statistic} = TPR - FPR \quad \text{Eq 3.6}$$

Under certain conditions, both the Precision-Recall Curve and the ROC Curve can be considered equal, and one curve *dominates* a second curve in ROC space if and only if the first dominates the second in the Precision-Recall space (Davis & Goadrich, 2006). However, optimizing a model using a ROC-based metric such as *Area Under the Curve (AUC)* does not necessarily optimize the model for the PRC-based metric. To explore

why using ROC is more suitable than the Precision-Recall Curve, we must first avoid the assumption that the distribution of the model’s prediction is predetermined by the distribution of the test samples or that *bias* tracks *prevalence* (D. M. W. Powers, 2020).

Here, *prevalence* is determined by the test set we collected, and *bias* is determined by the model’s predictions which can be controlled by the experimenter. Comparing the metrics used in ROC to PRC, we can observe that TPR and FPR’s denominators are both constants defined by the static dataset whereas Precision’s denominator can vary across the model. Thus, ROC is highly dependent on prevalence and can show how likely the classifier is influenced by different test samples or conditions rather than chance, where expected positive predictions turn up at the same rate as negative predictions. This ability to optimize the model to perform well under different conditions is highly beneficial to the VRD task since many visual relationships are missing. On the contrary, there is no easy way to visualize this chance line using the PRC.

Moreover, by observing the denominators of FPR when compared to Precision, we can also see that ROC aims to find a balance threshold between lower FN predictions and higher TN predictions, while PRC aims to find a balance threshold between lower FN and lower FP. Thus, optimising prediction thresholds based on ROC puts more emphasis on the negative examples that are often ignored by PRC analysis (D. M. W. Powers, 2020). In the context of our work, which emphasizes the use of contrastive learning with positive and negative examples, the ability to distinguish between positive and negative classes by ROC is more relevant. However, in the VRD case, where the distribution of the predicate classes is highly skewed, optimising the threshold for individual class using ROC-AUC can give a more optimistic view than optimizing the model based on the PRC-AUC.

Given the above reasons, we opted for ROC in this work and applied *Informedness* (see Eq. 3.7) as a measurement, which is calculated using bias-weighted Youden’s J Statistic (D. M. W. Powers, 2020). Here, weighting Youden’s J Statistic by bias respects the experimenter’s model choices by taking the intentionally introduced biases into account. Arguably, this is better than assigning equal weights to every predicate class such as *mAP* or *mR@n* because equal weight assignment does not account for the various de-biasing techniques or priors that are incorporated into VRD models. Here, we emphasize the use of bias-weighted measure over prevalence-weighted measure since the number of visual relationships is exponentially large and the labelled dataset is incomplete.

$$\text{Informedness} = \sum_{i \in (0, n)} b_i (\text{TPR}_i - \text{FPR}_i) \quad \text{Eq 3.7}$$

In this *Informedness* equation, n is the number of predicate classes $i \in (0, n)$ are the indices of the predicates, and $b_i = \frac{\text{TP}_i + \text{FP}_i}{\text{TP}_i + \text{FP}_i + \text{TN}_i + \text{FN}_i}$ is the ratio of predicted positive over all samples.

3.3 Discussion

While picking good deep learning architecture and loss design can improve the model’s performance, one of the inconvenient truths is that the selected benchmarks or dataset play an important role in determining the model’s success. In this chapter, we explored the Visual Genome dataset in more detail, highlighted the common long tail problem, and described common rebalancing techniques used to tackle it. Beside the dataset, we also explored the evaluation metrics that are commonly used for Visual Relationship Detection. Here, we argue that optimizing the model using solely the Recall@n metric can lead to a bias classifier and propose the use of ROC curve and the Informedness metric to evaluate the system.

The next chapter presents the main contribution of this thesis. We propose two different contrastive learning approaches to tackle the VRD task. The first approach focuses on learning structural-preserving representations or embeddings for VRD, while the second approach emphasizes simplicity and examines the effect of scaling up the pre-trained vision and language model on the VG150 benchmark performance.

Chapter 4

Contrastive Visual and Language Learning for Visual Relationship Detection

In this chapter, we motivate and describe the use of contrastive learning in the visual relationship detection task. First, we briefly describe the visual relationship detection problem in the context of the current literature. In Section 4.2, we propose a contrastive VRD architecture that preserves the first order structure of the scene graph using translational losses. In Section 4.3, we experiment with pre-trained models of different sizes to measure the impact of size and scale on VRD.

Visual relationship detection (VRD) aims to facilitate real-world description by bridging the gap between low-level visual information and high-level symbolic visual relationships, written in the form of *(subject, predicate, object)* triples. Previous work has explored the use of contrastive learning to generate joint visual and language embeddings that aid the detection of both seen and unseen visual relation triples. However, these contrastive approaches often learn the mapping functions implicitly and do not fully explore the underlying structure of the triples. Section 4.2 in this chapter aims to build joint visual and language embedding models that can capture such hierarchical structure between objects and predicates by explicitly imposing structural loss constraints.

Thus, instead of tackling the VRD problem as an end-to-end classification task, the architecture in Section 4.2 assumes the graphical structure of these visual relation triples, interprets this structure as a knowledge graph (Ji et al., 2021), and formulates the VRD problem as a knowledge graph completion problem (Z. Chen et al., 2020). However, unlike traditional knowledge graphs that are based on factual knowledge bases, the knowledge graphs here are represented by a set of visual entities and their interactions, where the nodes are subjects and objects grounded in the image through bounding boxes, and the edges are the relation predicates that exist between pairs of subjects and objects (Johnson et al., 2015). In the literature reviewed in Chapter 2, such formulation of a knowledge graph is also called a scene graph, and the task of knowledge graph completion is called scene graph completion (Wan et al., 2018).

Central to the scene graph completion is the idea of scene graph embedding (SGE), which aims to build embedding models that transform the entities and relations into low-dimensional vector spaces while preserving the structure of the original knowledge graph. Such approach is beneficial to VRD in two ways. First, because these embedding spaces preserve the graphical structures, unseen relations can be inferred by aggregating the relevant neighbours' features (Maheshwari et al., 2021). Second, like any other knowledge graph, a scene graph can be augmented with other domain-specific knowledge graphs or common-sense knowledge graphs (Sap et al., 2019) during training, allowing the model to incorporate background knowledge that might assist the detection of rare visual relationship.

In this chapter, we perform scene graph embedding using the contrastive learning approach, which learns representations by pulling together the target vector (or anchor) and a matching (positive) vector, while pushing apart the anchor from non-matching (negative) vectors. We believe that such a contrastive approach, when paired with negative sampling, can help us construct a better scene graph representation that can be transferred to other downstream tasks while giving us more control over the output embedding spaces. Here, we experimented different contrastive loss functions on two architectures, as described in Section 4.2 and Section 4.3 below. In Chapter 5, we report the results for both architectures, along with their experiments. In summary, we found that translational losses can penalize the VRD task, but this negative effect can be alleviate using TransR over TransE approach (see Chapter 5 for more details). We also found that larger models yield better performance when compared with their smaller counterparts, while models pre-trained on larger datasets do not necessarily present the best performance amongst the existing benchmark models.

4.1 Visual Relationship Detection

Visual Relationship Detection aims to construct a symbolic representation from an unstructured scene by representing it as a set of visual relationship triples in the form of *(subject, predicate, object)* such as *(person, riding, horse)* or *(window, on, train)*. Here, the subject and object are labels grounded to the salient image regions through bounding boxes, and the predicate is defined as a real-world interaction between these object pairs. However, due to the large number of potential real-world interactions, existing visual relation datasets including VRD (C. Lu et al., 2016) and Visual Genome (Krishna et al.,

2017) are often sparse and unbalanced, where common relations occur more frequently than rarer but plausible ones.

To tackle this problem, existing work seek to incorporate *linguistic knowledge* as additional training features to the learning model. For example, C. Lu et al. (2016) showed that leveraging pre-trained word embeddings can help the models learn linguistic statistical priors; similarly, Yu et al. (2017) showed that distilling knowledge from external Wikipedia datasets can improve the model's performance. Other work have also directly targeted the bias nature of the benchmark by incorporating spatial information, and (*subject, object*) co-occurrence priors as learning features for the classifiers (T. Chen et al., 2019), improving the model's performance significantly. Recent approaches have also tackled the effect of bias by introducing model-agnostic counterfactual prediction during inference such as Total Direct Effect (TDE) (Tang et al., 2020) or by experimenting with different sampling strategies (Desai et al., 2021). For our works in Section 4.2 and Section 4.3, we do not incorporate any external information or co-occurrence prior during training. Instead, we make use of pre-trained language models that encodes linguistic knowledge from larger corpora such as BERT (Devlin et al., 2019) or CLIP (Radford et al., 2021).

Perhaps the work that is most relevant to ours are from Peyre et al. (2019) and J. Zhang et al. (2019), who interpreted the VRD problem as a zero-shot detection task, and uses contrastive metric learning to construct joint visual and language embedding spaces that can be transferred to detect unseen visual relation triples. Here, Peyre et al. (2019) emphasizes the importance of *analogy transfer*, which is a downstream neural network module that leverages compositional embedding parts to compose novel visual relation triples. While the method shown in our work also constructs joint embedding spaces for visual relationships using contrastive learning method, the entire pipeline is trained end-to-end and the method focuses on the use of scoring distance metric to rank and select potential visual relationships, as we shall see in Section 4.2 and Section 4.3.

To facilitate the training of the embedding models in Section 4.2 and 4.3, we used two different subsets of the Visual Genome (Krishna et al., 2017) dataset. In Section 4.2, we first use the smaller VRD dataset that contains 5000 images to examine the impact of the explicit incorporation of translational losses. Afterwards, we evaluate the remaining of the work in Section 4.3 using the larger VG150 dataset with 60K images to compare with previous work.

4.2 The VLTransE Architecture

This section proposes an architecture for detecting visual relations triples to test whether translational constraint can improve the performance of the model for rare visual relationships. The general architecture, visualized in Figure 3.2 below consists of three modules: (1) the **Visual and Spatial Module** that extracts visual features and bounding box features (Figure 4.1, left), (2) the **Language Module** that learns contextualized token embeddings (Figure 4.1, right), (3) the **Loss Functions** that enforce the translational property and visual-language consistency (Figure 4.1, center).

The final output of the method consists of six embeddings with three visual embeddings (v^s , v^p , and v^o) and three language embeddings (w^s , w^p , and w^o). These embeddings are then trained on two set of losses: ($L_{subj}^{vl}, L_{pred}^{vl}, L_{obj}^{vl}$) are the visual-language consistency losses, while L_v^{Tr}, L_w^{Tr} are the translational losses for visual and language embeddings triples.

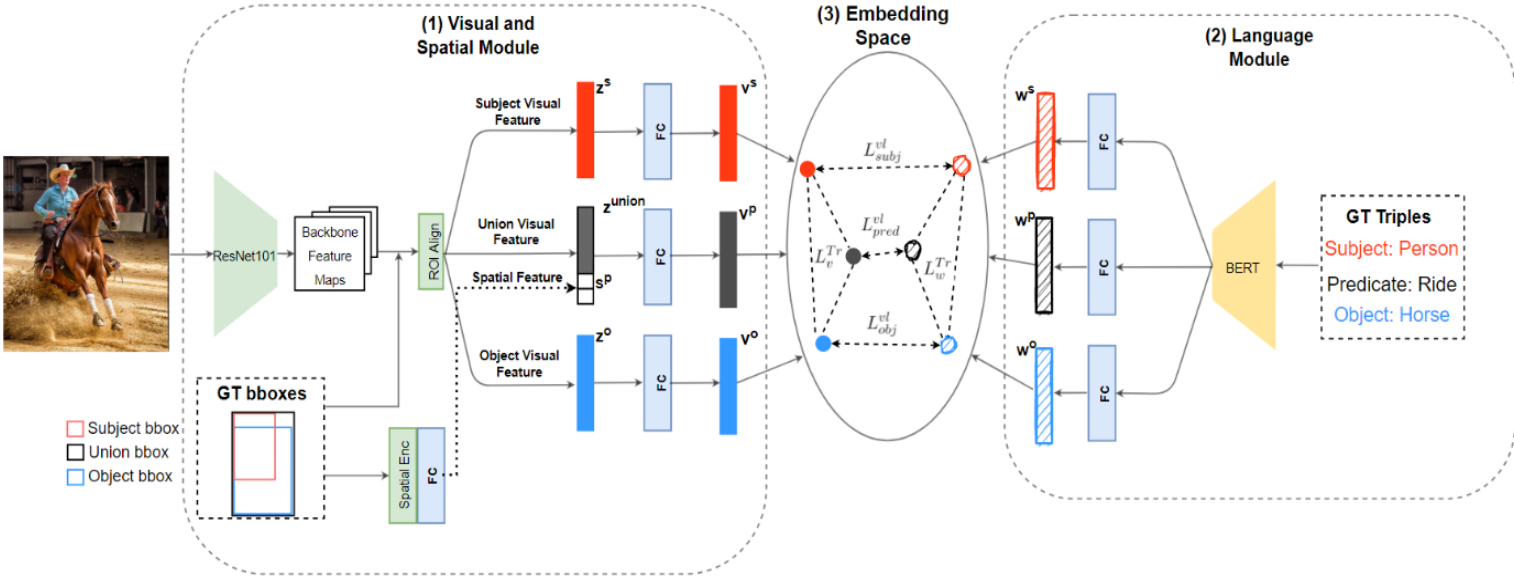


Figure 4.1: Overview of the proposed VLTransE architecture. Red, black, and blue colors represent subject, predicate, and object respectively.

In summary, the *visual and spatial module* (1) in Figure 4.1 consists of a shared backbone convolution neural network (CNN) that extracts visual features from the image based on the bounding box regions, and three neural network heads for subject, predicate, and object. Here, the predicate features are also concatenated with the spatial features extracted from the 2D bounding boxes' coordinates before passing through the predicate neural network head. Similarly, the language module consists of a shared token encoder and three separate neural network heads. For the *language module* (2) in Figure 4.1, we

decided to use BERT (Devlin et al., 2019), which is based on the self-attention mechanism and the Transformer architecture (Dong et al., 2021), to generate contextualized vectors that changes according to the context of the input triples. For the *loss functions* (3) in Figure 4.1, the model applies translational loss on the visual embedding triples and the language embedding triples independently from one another to preserve the first-order graph structure. To ensure the consistency between the visual embeddings and language embeddings, triplet margin loss is applied between the (visual, language) embeddings pairs for subject, predicate, and object respectively.

Below, each of the modules composing the architecture shown in Figure 4.1 are explained in more details.

4.2.1 Visual and Spatial Module

Visual Module: One of the main sub-tasks of visual relationship detection is to detect subjects and objects from a given image and extract their visual features for downstream embeddings. Given the success of CNN-based architecture in learning image representations from large scale pre-training, the visual feature extraction module in this work uses Faster R-CNN pre-trained on the COCO 2017 dataset. In the current implementation, Faster-RCNN consists of a shared ResNet-101 backbone network, a region proposal network (RPN), and a region of interest (ROI) detector. Thus, given the ground truth subject, object, and union bounding boxes, the visual features are extracted from the shared backbone and a region of interest pooling operation, yielding \mathbf{z}^s , \mathbf{z}^o , and \mathbf{z}^{union} respectively.

Spatial Module: The model also extracts spatial feature (Figure 4.1, bottom left) from the subject, object, and union bounding boxes to incorporate spatial and position priors. Unlike tradition object detection task which aims to detect salient objects in an image, our spatial module is introduced to capture crucial spatial relationship between objects. To do this, we defined a 22-dimensional feature vector similar to that of J. Zhang et al., (2019), Peyre et al., (2019), and Sadeghi et al., (2015). Here, given the three boxes b^s, b^{union}, b^o in $[x, y, w, h]$ format, where (x, y) is the starting coordinate and (w, h) is the width and height of the box, the *spatial encoder* first generates $\Delta(b^1, b^2)$, which defines the relative position, normalized displacements, and scale between two bounding boxes. It also generates $\mathbf{c}(b)$, which captures individual characteristics of the bounding box relative to the entire image. The resulting vector is a 22-dimensional feature vector defined in *Equation 4.3* below.

$$\Delta(b^1, b^2) = \left\langle \frac{x^1 - x^2}{w^2}, \frac{y^1 - y^2}{h^2}, \log\left(\frac{w^1}{w^2}\right), \log\left(\frac{h^1}{h^2}\right) \right\rangle \quad \text{Eq. 4.1}$$

$$\mathbf{c}(b) = \left\langle \frac{x}{w_{img}}, \frac{y}{h_{img}}, \frac{x+w}{w_{img}}, \frac{y+h}{h_{img}}, \frac{wh}{w_{img}h_{img}} \right\rangle \quad \text{Eq. 4.2}$$

$$\langle \Delta(b^s, b^o), \Delta(b^s, b^{union}), \Delta(b^{union}, b^o), \mathbf{c}(b^s), \mathbf{c}(b^o) \rangle \quad \text{Eq. 4.3}$$

The spatial feature vector in *Equation 4.3* is passed through two fully connected layers to get the final 64-dimensional spatial embedding, \mathbf{s}^p .

While the generated spatial feature vector is partially inspired by VisKE (Sadeghi et al., 2015), it should be noted that VisKE aimed to find the most probably explanation by comparing the detected spatial coordinates pair against multiple sets of images of shared visual phrase, making spatial coordinates the main feature for VisKE. In this work, we also include visual features in addition to the spatial features as described in the Visual Module.

These extracted visual and spatial feature vectors are then passed through three separate neural networks to generate the subject, predicate, and object embeddings. For the subject and object embeddings, the visual feature vectors go through three fully connected layers with RELU activation function to get 256-dimensional embedding vectors, v^s and v^o . Similarly, the union feature vector, z^{union} , is first concatenated with the spatial embedding, s^p , before going through three fully connected layers with RELU activation function to get the 256-dimensional predicate embedding vector, v^p .

4.2.2 Language Module

For the language module, the model also learns three separate neural networks for subject, predicate, and object that map the pre-trained language features toward the final joint visual and language spaces. Here, the architecture uses frozen BERT instead of word2vec as our pre-trained language encoder to leverage the contextualized information from the entire triplet. While the subject-predicate-object triples are unnatural input sequences for BERT which was pre-trained on Wikipedia and the BookCorpus dataset (Devlin et al., 2019), we believe that contextualized encoders like BERT are still beneficial for visual relationship detection because the same predicate can have different meanings under different (*subject, object*) contexts. However, empirical results from Ilinykh and Dobnik,

(2022) showed that training a text-only model on image captions data gives different semantics from a text model trained on general text. Given that visual relationship triples are extracted from image captions, the pre-trained BERT model used in this work is not fully applicable to visual relationship triples. To overcome this limitation, we also finetuned the extracted BERT features using the three separate neural networks, resulting in three 256-dimensional embedding vectors, w^s , w^p , and w^o , in the final output.

4.2.3 Loss Functions

The model is tested on triplet margin loss and contrastive cross entropy losses. Here, cosine similarity is used as the distance metric \mathbf{d} for all loss functions.

Triplet Losses for Visual and Language Consistency. The following set of loss function aims to bring the three positive visual embeddings (v^s, v^p, v^o) closer to the three positive language embeddings (w^s, w^p, w^o), while pushing apart negative pairs. To reduce the number of equations, the loss function in Equation 4.6 or L^{vl} is applied separately for the three subject, predicate, and object heads. Therefore, given the set $\mathbf{V} = \{\mathbf{v}, \mathbf{w}\}$ of positive visual and language embedding pairs, the set $\mathbf{V}^{v^-} = \{\mathbf{v}^-, \mathbf{w}\}$ of negative visual with positive language pairs, and $\mathbf{V}^{w^-} = \{\mathbf{v}, \mathbf{w}^-\}$ of positive visual with negative language pairs, the triplet losses are:

$$L_v^{vl} = \sum_{(v,w) \in \mathbf{V}} \frac{1}{|\mathbf{V}^{v^-}|} \sum_{(v^-,w) \in \mathbf{V}^{v^-}} [m + \mathbf{d}(v, w) - \mathbf{d}(v^-, w)]_+ \quad \text{Eq 4.4}$$

$$L_w^{vl} = \sum_{(v,w) \in \mathbf{V}} \frac{1}{|\mathbf{V}^{w^-}|} \sum_{(v,w^-) \in \mathbf{V}^{w^-}} [m + \mathbf{d}(v, w) - \mathbf{d}(v, w^-)]_+ \quad \text{Eq 4.5}$$

$$L^{vl} = L_v^{vl} + L_w^{vl} \quad \text{Eq 4.6}$$

where $[x]_+ = \max(0, x)$ denotes only the positive part of the input, m denotes a margin of 0.2, and \mathbf{d} is cosine similarity distance metric. L^{vl} is applied correspondingly for objects, subjects, and predicates pairs.

Contrastive Cross Entropy Losses for Visual and Language Consistency. Similar to Hermans et al. (2017), we also observe that triplet losses yield slow convergence to a sub-optimal solution when random sampling is applied. To overcome selection bias problem with random negative sampling (B. Yu et al., 2018) that leads to local minima, we also apply the contrastive cross entropy losses for the three *subject, predicate, object* heads.

Thus, given the set \mathbf{S}_v of all visual embeddings and set \mathbf{S}_w of all word embeddings for the given head, the contrastive cross entropy loss is represented by:

$$L^{vl-CE} = \frac{1}{|\mathbf{S}_v|} \sum_{v \in \mathbf{S}_v} \sum_{w \in \mathbf{S}_w} -\mathbf{1}_{y_v=y_w} \cdot \log \left(\frac{e^{\mathbf{d}(v,w)}}{\sum_{t \in \mathbf{S}_w} e^{\mathbf{d}(v,t)}} \right) \quad \text{Eq 4.7}$$

where $\mathbf{1}_{y_v=y_w} = \begin{cases} 1, & \text{if } (v, w) \text{ is positive} \\ 0 & , \text{otherwise} \end{cases}$ and \mathbf{d} is the cosine similarity distance. The same L^{vl-CE} is applied correspondingly for subject, predicate, and object heads.

Translational Loss. To enforce the hierarchical structural priors of visual relation triples, the model also enforces the translational loss on the visual embeddings and language embeddings. Thus, given a set \mathbf{S} of valid triples (s, p, o) , and \mathbf{S}^- of randomly selected negative triples (s', p, o') , the translational losses are defined as:

$$L_v^{Tr} = \sum_{(s,p,o) \in \mathbf{S}} \frac{1}{|\mathbf{S}^-|} \sum_{(s',p,o') \in \mathbf{S}^-} [m + \mathbf{d}(v^s + v^p, v^o) - \mathbf{d}(v^{s'} + v^p, v^{o'})]_+ \quad \text{Eq 4.8}$$

$$L_w^{Tr} = \sum_{(s,p,o) \in \mathbf{S}} \frac{1}{|\mathbf{S}^-|} \sum_{(s',p,o') \in \mathbf{S}^-} [m + \mathbf{d}(w^s + w^p, w^o) - \mathbf{d}(w^{s'} + w^p, w^{o'})]_+ \quad \text{Eq 4.9}$$

Here, the predicate embeddings act as the translational vector between the subject and object embeddings. Thus, the final combined loss function is defined as:

$$L = L_{subj}^{vl} + L_{obj}^{vl} + L_{pred}^{vl} + L_v^{Tr} + L_w^{Tr} \quad \text{Eq 4.10}$$

or

$$L = L_{subj}^{vl-CE} + L_{obj}^{vl-CE} + L_{pred}^{vl-CE} + L_v^{Tr} + L_w^{Tr} \quad \text{Eq 4.11}$$

Test-Time Inference. To perform test-time inference on the generated visual and language embeddings, the evaluation algorithm computes the cosine similarity distances between the visual embeddings and language embeddings and ranks them to select the top predictions. Depending on the evaluation task, different embedding parts of (*subject, predicate, object*) can be used (Figure 3.2) In this work, we evaluated the model on two tasks: (i) the predicate prediction task, and (ii) the tail entity prediction task.

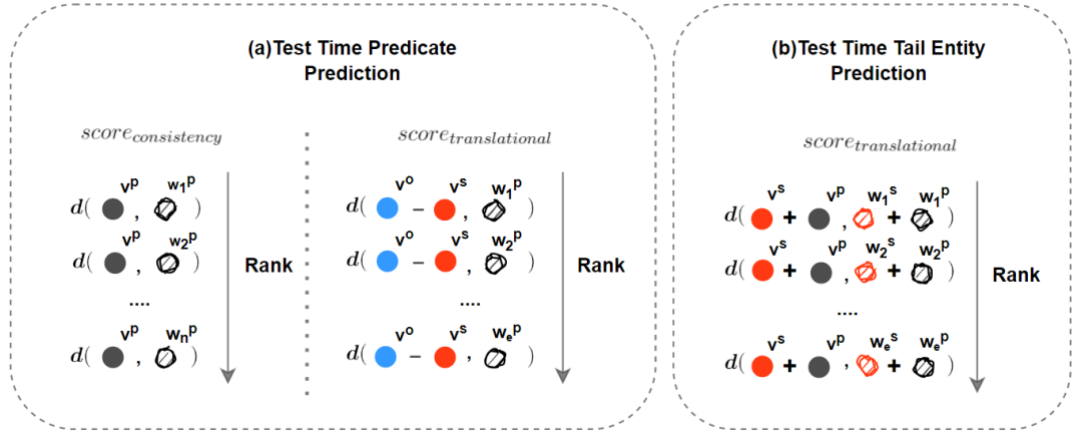


Figure 4.2: A visualization for test time scoring functions. Red, black, and blue colours represent subject, predicate, and object respectively. $\mathbf{d}(\mathbf{x}, \mathbf{y})$ computes the cosine distance between x and y , and the distances are ranked in an ascending order. In (a), \mathbf{n} is the number of predicate classes in the dataset. In (b), \mathbf{e} is the number of object classes in the dataset.

For the predicate prediction task (Figure 4.2a), both the ground truth bounding boxes and labels for subject and object are given. Thus, given the ground truth bounding boxes and an image, the three visual embeddings (v^s, v^p, v^o) for subject, predicate and object are first generated by the **visual and spatial module** (Figure 4.2, 1). For the language modality, due to the usage of BERT contextualized encoder, the evaluation algorithm first enumerates all possible $(subject, predicate_i, object)$ triples where $i \in (0, \mathbf{n})$ and \mathbf{n} is the number of predicates. These triples are then passed through the **language module** (Figure 4.2, 2) to generate \mathbf{n} language embeddings triples, $(w^s, w^p, w^o)_{i \in (0, \mathbf{n})}$. Thus, given the visual (v^s, v^p, v^o) embedding triple and the language $(w^s, w^p, w^o)_{i \in (0, \mathbf{n})}$ embedding triples, the visual-language consistency score for the predicate is computed as:

$$SCORE_{consistency} = \mathbf{d}(v^p, w^p) \quad Eq 4.12$$

and the translational score is computed from:

$$SCORE_{translational} = \mathbf{d}(v^o - v^s, w^p) \quad Eq 4.13$$

These two scores are then multiplied to get the final ranking score

$$SCORE_{combine} = SCORE_{consistency} * SCORE_{translational} \quad Eq 4.14$$

For the tail entity prediction task (Figure 4.2b), we aim to predict the object entity given only the ground truth bounding box for subject and ground truth labels for subject

and predicate. Unlike the predicate prediction task, where both the subject and object ground truth bounding box information are given, this task only has the subject's bounding box information, leaving the object to be predicted. This introduces ambiguity to the task, making it more challenging. First, the object's location becomes uncertain due to the absence of the object's bounding box information, making it harder than the predicate prediction task where bounding box information is available for both the subject and object. Moreover, the model has to deal with ambiguity and immense variability of potential objects, broadening the search space considerably compared to predicate prediction task. Despite its challenges, the tail entity prediction task encourages the model to interpret subject-predicate pairs, which we believe can help the model relies more upon the information from the visual context provided by the subject and its surrounding.

Thus, without the ground truth object bounding box, the union box is set to be the subject bounding box. Therefore, given the image and the subject bounding box, the **visual and spatial module** (Figure 4.2, 1) generates the visual embeddings (v^s, v^p) for the subject and predicate. Similarly, from the subject and predicate ground truth labels, the evaluation algorithm first enumerates all possible $(subject, predicate, object_i)$ triples where $i \in \mathbf{e}$ and \mathbf{e} is the number of object classes. These triples are then passed through the **language module** (Figure 4.2, 2) to generate $(w^s, w^p, w^o)_{i \in (0, \mathbf{e})}$ embedding triples. Given the visual (v^s, v^p) embeddings and the language $(w^s, w^p, w^o)_{i \in (0, \mathbf{e})}$ embedding triples, the translational score function is computed as:

$$score_{translational} = \mathbf{d}(v^s + v^p, w^s + w^p) \quad \text{Eq 4.15}$$

The computed scores above are then used to rank and select the top predicate or tail entity for the task. The evaluation results for the experiments in this section is presented in Chapter 5.

4.3 A Simple Contrastive Learning Architecture

We observed that it is not optimal to create separate contextualized embeddings for each *subject*, *predicate*, and *object*. Thus, we propose a simple contrastive learning approach that is more aligned to the VRD task. During this process, we examine whether contrastive embedding models pre-trained from the abundant amount of unstructured web-scraped image and text pairs (See Table 4.1) can improve the model’s performance.

Dataset	No. unique image-text pairs
LAION 400M	413 million pairs
LAION2B-EN	2.32 billion pairs

Table 4.1: Number of unique image-text pairs for two different datasets used in pre-training.

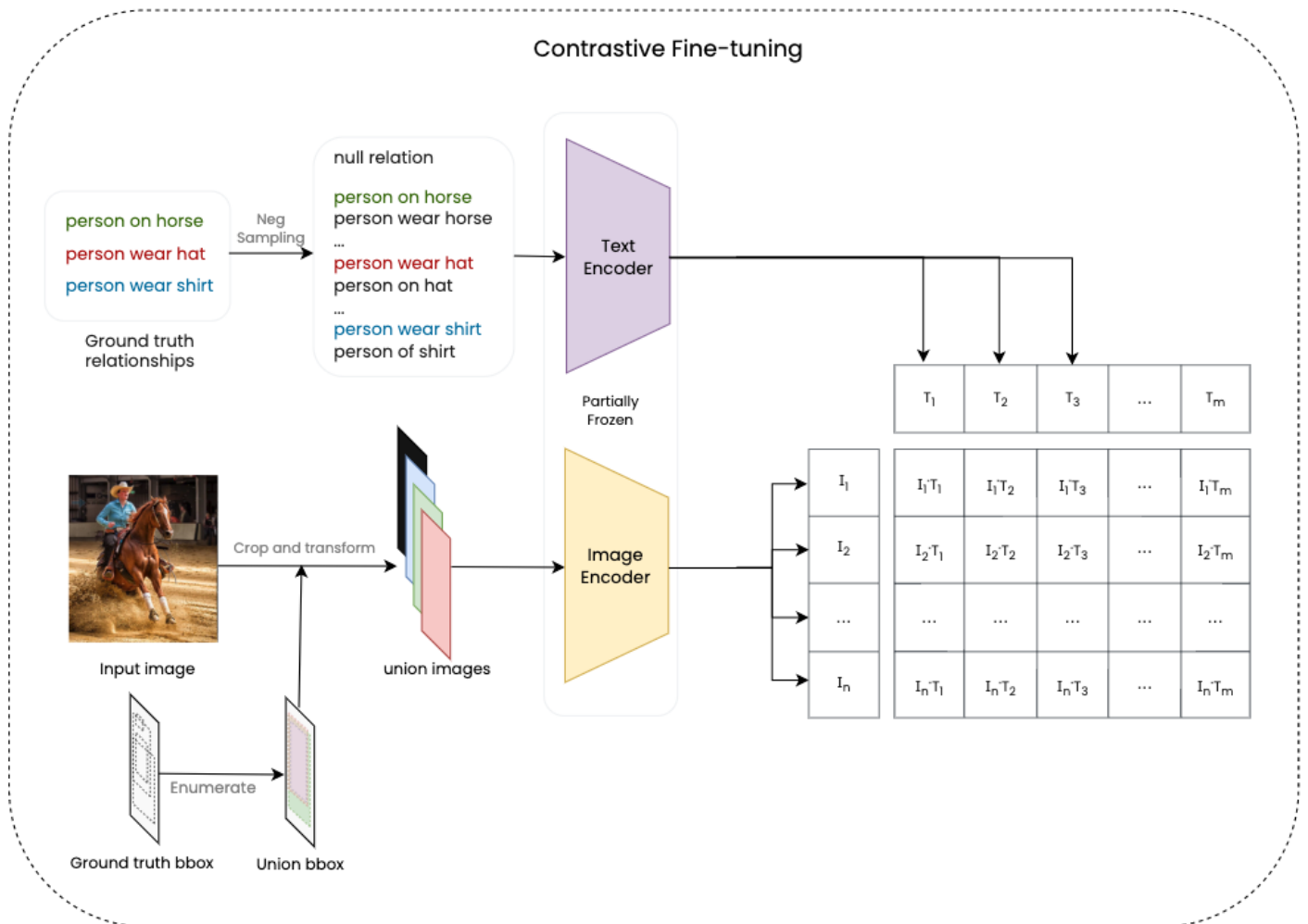


Figure 4.3: Overview of the proposed contrastive architecture. Visualization is partially adapted from pairs Radford et al. (2021).

This section describes the architecture and outlines the details of the implementation. While the general architecture here consists of three modules similar to the one described in Section 4.2, each module deviates from the one described in Section

4.2 as follows: (1) The **Visual Module** now generates visual embeddings based on the extracted features from the (*subject, object*) union image regions rather than individual objects, (2) the **Language Module** generates one contextualized embedding by concatenating the (*subject, predicate, object*) into a string instead of three separate embeddings for every elements in the triple, and (3) the **Contrastive Loss Module** now applies Cross Entropy variants instead of the Triplet Loss or TransE loss. Like VLTransE in Section 4.2, we use the cosine similarity metric as the distance metric and the Cross Entropy loss as our main loss objective.

Dataset	Image encoder	No. Params
CLIP	ResNet50	25.6M
CLIP	ResNet101	44.5M
OpenCLIP	VIT-B-16	86M
OpenCLIP	VIT-B-32	86M
OpenCLIP	VIT-L-14	307M
OpenCLIP	VIT-H-14	632M

Table 4.2: Number of parameters for the visual encoder used in CLIP and OpenCLIP.

Table 4.1 describes the size of the two datasets used to pretrained CLIP and OpenCLIP model, and Table 4.2 describes the number of parameters used by different image encoders. We used the open-source CLIP and OpenCLIP models and did not perform the pre-training. OpenCLIP is the open-source version of CLIP with released pre-trained models for the larger encoder sizes. For the image encoder, VIT-B-16 denotes a Vision Transformer Base model with an input sequence of 16x16 patch sizes. The VIT Base is smaller than VIT Large, and VIT Large is smaller than VIT Huge. We expand upon the architecture of the Vision Transformer in greater details in Section 5.1.2.

4.3.1 Visual Module

Visual Encoder. Given the success of CNN-based architecture and Transformer-based architecture in learning image representations from large-scale datasets, we applied two different types of pre-trained backbone as our encoders: (i) the ResNet50 CNN-based visual backbone, and (ii) the ViT Transformer-based image backbone. We used the OpenCLIP pre-trained image encoder and language encoders (Section 3.2) on either the LAION 400m or LAION 2b-en datasets (See Table 4.1) to evaluate the impact of scaling up the size of the model and dataset on the final performance of the VRD task.

Image Pre-processing. Given the ground truth bounding boxes from an input image, we enumerated all n possible union bounding boxes and extracted $I_{i \in (0,n)}$ embedding vectors using one of the image encoders. It is worth noting that most of the encoder parameters were frozen during the fine-tuning process due to the limited resource setting, and we only trained the last two encoder layers. We also applied this fine-tuning approach to the language encoder described in Section 4.3.2 below.

4.3.2 Language Module

Language Encoder. Similar to the Vision Transformer used in the Visual Encoder, the language encoder used in this work is a 12-layer 512-wide Transformer architecture (Vaswani et al., 2017) with 8 attention heads that can leverage the contextualized information from the entire input sentence. Here, the input sentence is a concatenation of the subject, predicate and object triples as described in the pre-processing step below.

Language Pre-processing. For each triple in the set of k ground truth triples, we first enumerated p (*subject, predicate* $_{i \in (0,p)}$ *, object*) triples where p is the number of predicates. We then concatenated each of these triples into '*subject predicate object*' string format, resulting in $m = k * p$ sentences. We also added a 'null relation' string as a matched pair with union image regions for (subject, object) pairs that have no visual relationship.

4.3.3 Contrastive Loss Module

Similar to the VLTransE architecture in Section 4.2, we apply a dual Cross Entropy loss function for the vision and language consistency loss. Here, cosine similarity was used as a distance metric d for the loss function

Negative Sampling. We performed negative sampling during the image and language enumeration and pre-processing step, where embeddings of each modality were paired against multiple negative embeddings of the other modality. This process resulted in a non-symmetric table as shown in Figure 4.3, where the row represents the n image embedding vectors and the column represents the m text embedding vectors. Using this table mask of positive and negative examples, we constructed the labels y for both image and text.

Visual and Language Consistency Loss. We computed a cross entropy loss along the table's rows and a cross entropy loss along the table's columns. Thus, given the set \mathbf{S}_v ,

containing all possible image embeddings I and the set \mathcal{S}_l containing all possible text embeddings T , the cross-entropy loss equation that aligns image to text is:

$$L^{v-CE} = \frac{1}{|\mathcal{S}_v|} \sum_{I \in \mathcal{S}_v} \sum_{T \in \mathcal{S}_l} -\mathbf{1}_{y_I=y_T} \cdot \log \left(\frac{e^{d(I,T)}}{\sum_{T \in \mathcal{S}_l} e^{d(I,T)}} \right) \quad Eq 4.16$$

Similarly, the cross-entropy loss equation that aligns text to image is:

$$L^{l-CE} = \frac{1}{|\mathcal{S}_l|} \sum_{T \in \mathcal{S}_l} \sum_{I \in \mathcal{S}_v} -\mathbf{1}_{y_I=y_T} \cdot \log \left(\frac{e^{d(I,T)}}{\sum_{I \in \mathcal{S}_v} e^{d(I,T)}} \right) \quad Eq 4.17$$

where

$$-\mathbf{1}_{y_I=y_T} = \begin{cases} 1, & \text{if } (I,T) \text{ are matching pairs} \\ 0, & \text{otherwise} \end{cases}$$

The final visual and language consistency loss can be defined as:

$$L^{vl-CE} = L^{v-CE} + L^{l-CE} \quad Eq 4.18$$

Test Time Inference. At test time, given a set of ground truth (*subject, object*) class pairs and a set of objects' bounding box regions, the evaluation algorithm first enumerates all possible (*subject, predicate, object*) triples and union image regions. These relation triples are then pre-processed into strings as described in Section 3.2, yielding p textual embeddings for each pair, where p is the number of predicates. Similarly, the union bounding boxes and the image were pre-processed and encoded according to the method described in Section 3.1, yielding n possible embeddings.

For each image embedding $I_{i \in (1,n)}$, the evaluation code measured the similarity scores between the given image embedding and all possible corresponding textual embeddings in $T_{j \in (1,p)}$. The evaluation algorithm then ranks $n * p$ scores and selects the top result to compare against the ground truth labels using the Recall metrics described in Han et al. (2021).

4.5 Discussion

Visual Relationship Detection is the cornerstone of many modern machine learning tasks that require a comprehensive understanding of the visual scene. Current contrastive distance metric approaches in learning joint visual-language embeddings for

VRD often rely on neural networks learning the necessary transformations implicitly without any structural constraints. To this end, we propose VLTransE (Section 4.2), a contrastive visual-language embedding model that preserves the first-order structure of the graph using the translational constraint.

We also proposed a simplified contrastive vision and language embedding model in Section 4.3 to reduce the memory requirements, and investigated whether large neural encoders pre-trained on a large amount of web image-text pairs can assist the detection of visual relations. Here, we fine-tuned deep neural encoders of different scales and proceed to report the results in Chapter 5 below.

Chapter 5

Experimental Results and Analysis

In this chapter, we give a detail overview of the experimental details used for the visual relationship detection task and analyse the evaluation results for the models presented in Chapter 4.

5.1 Implementation

Both architectures described in Chapter 4 are implemented using PyTorch on a single RTX 3090 GPU. We first perform the evaluation for the VLTransE experiments in Section 4.2 on the smaller VRD dataset, and evaluate the simplified contrastive models described in Section 4.3 on the larger Visual Genome 150 (VG150) dataset. It should be noted that the experimental results for the model described in Section 4.2 cannot be compared against experiments for the model described in Section 4.3 because they are trained and evaluated on two different datasets. A summary of the train and test splits for each of the above datasets are shown in Table 5.1 below.

Dataset	No. Objects	No. Predicates	Training Set	Test Set
VRD	100	70	4,000	1,000
VG150	150	50	60,784	26,466

Table 5.1: The train and test split for the smaller VRD dataset and larger VG dataset

For the VLTransE architecture described in Section 4.2, we performed smaller scale experiments to see the impact of additional first-order constraint has on the model. Here, we used the common Recall@n metric for VRD to compare different models. The details of the implementations and design decisions of the model is described in Section 5.1.1. For the simple vision and language contrastive learning model described in Section 4.3, we performed experiments to examine the impact of large-scale pretraining have on the model. Here, we opted for a larger Visual Genome 150 (VG150) dataset and replicated the experiments using the Scene Graph Benchmark codebase. We also take advantage of the open-source implementation of CLIP (Wortsman et al., 2022), and implement our own set of evaluation metrics described in Section 3.2. The details of the implementations of both the CLIP model and the benchmark models are described in Section 5.1.2 below.

5.1.1 Implementation details of the VLTransE architecture

One of the main problems with the visual relationships dataset is that they are highly unbalanced, resulting in a biased object detector (See Appendix A for details). Here, a poorly performed object detector penalizes downstream tasks, which has a significant implication on more complex scene graph generation tasks. We use a decoupled CNN architecture as described in Section 2.4.5, where the backbone network of the object detector is separate from that of the visual relationship detection task. Here, we performed a two-stage training strategy where we train the object detector in the first stage and the end-to-end visual relationship detection network in the second stage. We also assumed that the object detection task is solved and freeze the object detector backbone during the second stage of training.

Visual Module. We used Detectron 2’s Faster R-CNN with FPN and the ResNet101 backbone as the base implementation (Yuxin et al., 2019) of our object detector. In the second training stage, we duplicated the finetuned ResNet101 and applied the decoupled architecture to train our relationship network. For the three (*subject*, *predicate*, *object*) projection heads in the visual relation network, we use MLP and the Leaky ReLU activation function with a negative slope of 0.01 (B. Xu et al., 2015). To regularize the network, all MLP heads are set to have a dropout rate of 30 percent.

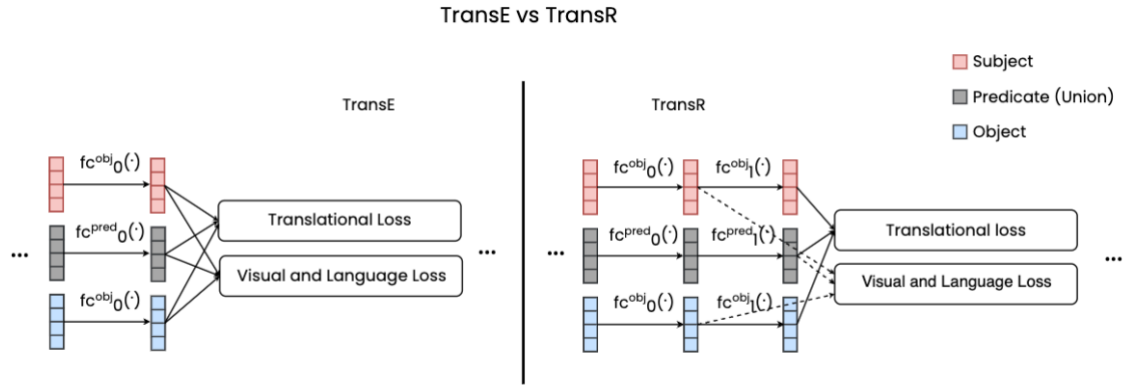


Figure 5.1: A visualization of TransE vs TransR architectural differences, where $fc(\cdot)$ is a MLP with Leaky ReLU.

Based on our initial experiments, we observed that the gradients are unstable when using only one projection function for the translational loss. With this observation, we also implemented a variant of TransE knowledge graph embedding technique called TransR (Y. Lin et al., 2015). Instead of optimizing for both the translational loss and the consistency loss on the same embedding space, we incorporate additional projection

functions $f_{c_1}(\cdot)$ to map the subject and object from the entity space to the predicate space before applying the translational loss function (see Figure 5.1). While the implementation of the projection function used an MLP instead of a projection matrix like that of the original paper, the core idea remains the same. Since every pair of visual objects have more than one predicate, the additional projection function allows for multi-relationship modeling as each visual relationship exists in its own space. During test time, we use the appropriate projected embeddings to evaluate the visual relationship detection task and observe a consistent performance of TransR over TransE approach.

Language Module. Different from the visual module which used CNN-based architecture to extract the visual feature, the language module extracts the features by taking the average of the last four layers of BERT. The embeddings of the corresponding visual objects' labels and relationship labels are extracted from this one embedding. In a hindsight, this is not the optimal approach to extracting the language features. To exploit BERT's contextualized embedding, perhaps a more optimal approach is to directly compute both the (*subject*, *predicate*, *object*) embeddings and the corrupt triple embeddings. We also added the three *subject*, *predicate*, and *object* projection heads using MLP with Leaky ReLU activation like the visual module implementation to test both the TransE approach and the TransR approach. Due to the high memory demand under the contrastive training regime, we precomputed all the BERT embeddings for the *subject*, *predicate*, *object*, and stored them in the disk.

Training Procedure. In the first training stage, we finetuned the object detector module until convergence, with a 65.2 average precision on the train set. In the second training stage, we attached the frozen backbone network to our relationship networks. After setting up both the visual and language modules, we train the relationship networks using both the translational loss function and the visual-language consistency loss function simultaneously for four epochs. Here, we use momentum stochastic gradient descent with an initial learning rate of 0.001 and a momentum of 0.9. We also enforce a learning schedule that decreases the learning rate by a factor of 10 in the second and fourth epochs.

Contrastive Losses. We ran two experiments to test different approaches for constructing both the translational losses and the visual-language consistency losses: (i) the Hinge margin loss and (ii) the Cross Entropy loss (or InfoNCE). In all experiments, we used the cosine distance metric, which computes the angle between the two projected vectors on the unit sphere. In our initial experiments, we used this cosine distance with the Hinge

loss function, also known as the Cosine Embedding loss, for both the translational losses and the visual-language consistency losses. During the preliminary tests, we observed that the Hinge loss for visual-language consistency losses yielded worse results than that of the Cross Entropy loss. Thus, we mainly applied the Cross Entropy loss for the visual-language consistency losses while keeping the Hinge loss for the translational losses.

Sampling Strategy. One of the main bottlenecks of the contrastive learning approach is the GPU memory footprint and the negative sampling strategy. In the VLTransE work, we use a random negative sampling strategy for both the translational loss and the visual-language consistency loss. This has a huge implication for the result of the learning model because random negatives can adversely affect the learned features. Moreover, due to the limited memory footprint, we must limit the number of negative examples that can be used to train the model at every iteration. These two reasons combined have contributed to the underperformance of the model.

To tackle these contrastive learning related problems, we had to (1) reduce the number of negative examples needed at every training iteration and (2) improve the negative selection process. For these two main reasons, we propose a simpler approach to tackle the visual relationship representational learning task in Section 4.3 above. We will further describe the implementation details in Section 5.1.2 below.

5.1.2 Implementation details for a Simple Contrastive Learning Architecture

Different from VLTransE, the Simple Contrastive Learning architecture focuses on simplifying the architecture and loss functions to scale down the required GPU memory footprint during training. One of the design decisions is to remove the additional translational loss constraint which took up training memory for the negative examples but did not improve the smaller VRD benchmark metrics. Instead, we only applied the conventional InfoNCE contrastive loss functions that align the image and text modality. Given that the fine-tuned Faster-RCNN object detector on the smaller VRD test dataset has only an average precision of 13.7 (See Appendix A), another decision we made is to replace the Faster R-CNN object detection network and backbone with a vision Transformer-based network. Here, we used the open-sourced version of CLIP called OpenCLIP which was pretrained on a large image and text dataset scraped from the web (Radford et al., 2021).

Visual Module. By replacing the CNN-based architecture with the Transformer-based architecture, we traded off accuracy and efficiency for simplicity. Instead of training a fine-grained object detector to extract features from the object’s bounding box region, we first cropped the union bounding box regions of the (subject, object) entity and further subdivided the cropped regions into square patches. Here, each patch is treated as an input node to the Transformer network to generate the visual embedding vector (See Figure 5.3). The downside to this approach is that there is no fine-grained spatial information signal to guide the network, and we assumed that the network could attend to the relevant objects implicitly. Whether this assumption holds requires further examinations of the learned model via linear probing or other explainable AI techniques that are outside of the scope of this work.

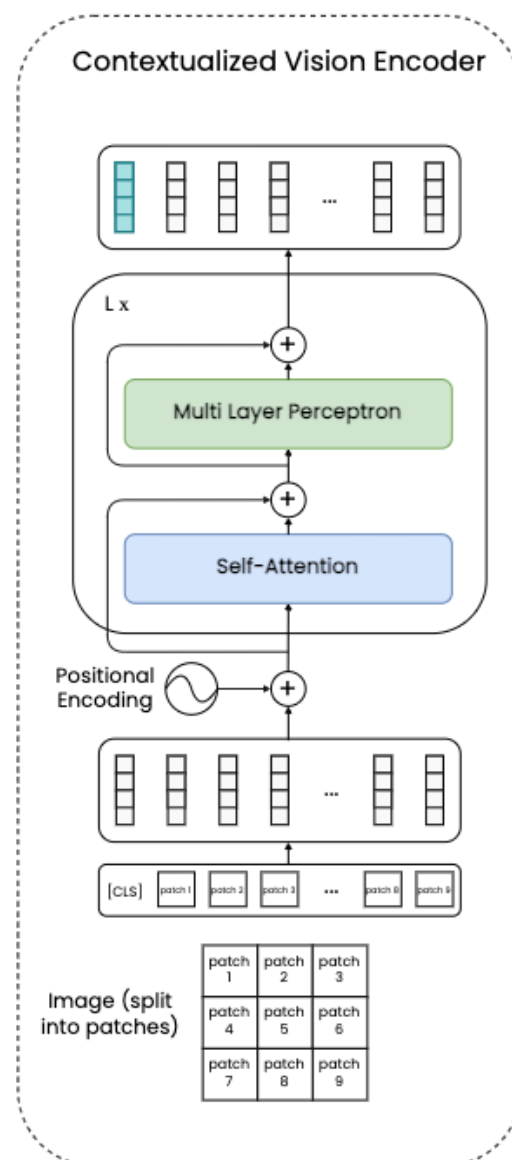


Figure 5.2: A visualization of a Vision Transformer encoder (Adapted from Dosovitskiy et al., 2020).

Language Module. Like VLTransE, we also used a Transformer-based encoder for our language module. However, instead of using BERT which is pretrained solely on textual corpora, we used the OpenCLIP models that were pretrained on image and text descriptions. Due to limited resource setting, we only unfroze the last two layers of the Transformer.

Training Procedure. We unfroze the last two layers of both the image and text encoders and trained them for one epoch on the VG150 dataset. Here, we used an adaptive stochastic gradient descent, AdamW, with an initial learning rate of 0.05, β_1 of 0.9, and β_2 of 0.98 (Loshchilov & Hutter, 2019). We also enforced a learning schedule that decays the learning rate every 15,000 steps, and an initial warm up period of 1000 steps (see Figure 5.3). The scheduler with an initial warm up period prevents the model from overfitting the initial training samples by assigning lower learning rate, allowing the model converges faster in the later iterations.

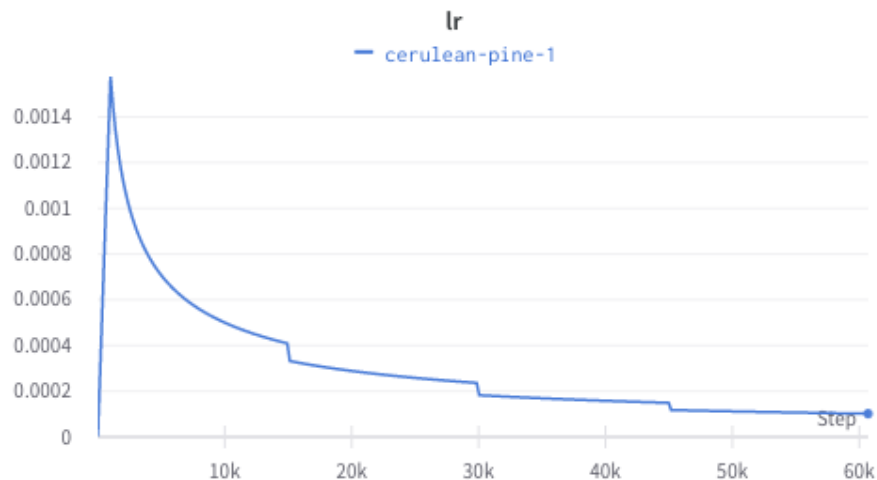


Figure 5.3: Learning rate schedule that warms up for 1000 steps and decays every 15,000 steps.

Negative Sampling. One of the limitations of VLTransE learning was that the negative sampling process for the visual and language consistency loss is random. This naïve implementation introduced a couple of issues, including inefficient learning, incorrect negative examples selection, or biases due to the long-tail distribution. To overcome these problems while solving the limited GPU memory problem, we introduced an Adaptive Negative Sampling Strategy that drew negative samples from the inverse distribution of the drawn samples. The intuition here is that the negative triples that has been drawn before for a given anchor triple should not be penalized as much in later iterations. To do this, we kept track of the drawn negative samples for a given positive sample in memory

and computed the reciprocal distribution of its multinomial distribution. This reciprocal distribution is initialized as a uniform distribution and slowly changed depending on the selected samples. To prevent the negative sampling process from selecting incorrect negative samples, we also keep track of other correct labels and adjust the distribution appropriately prior to the sampling process. Still, this negative sampling process can only be used for the selection of negative language embeddings. For the visual negative selection process, we naively selected all the other union regions of the given image that do not have the same predicate label. Such an approach can potentially cause issues in the final representation since there are overlapping relationship regions.

5.1.3 Replication details for Scene Graph Benchmarks models

To facilitate comparison with existing work, we also re-trained a few of the benchmark models on the same VG150 dataset under the same environment (GPU). Here, we used the scene graph benchmark provided by Han et al. (2021). All re-trained models used Faster R-CNN with ResNet50 backbone. Like above, we trained these models for one epoch. However, we followed the author’s default learning rate, optimization algorithm, and scheduler. We observed that there was a learnable bias logits vector added to the final prediction logits as a priori. Here, the bias logits vector is learned from an initialized co-occurrence matrix between the subject, object, and predicate. To ensure fair comparisons, we trained two versions for each model where the bias term is added and not added. The experimental results and comparisons between our models and the replicated models are shown in Section 5.2 below.

5.2 Experimental Results and Analysis

The following section presents the evaluation results and analysis for the described experiments. We present the VLTransE experiments on the VRD dataset in Section 5.2.1 and the Simple Contrastive experiments on the VG150 dataset in Section 5.2.2. For VLTransE, we only present our results using the traditional Recall@n metric to compare with existing work as we did not replicate their results. For the second experiment, we present the results using the traditional Recall metric, the more recent mean Recall metric, and the proposed Informedness metric from Chapter 3. We then compare our results with other models in the Microsoft’s scene graph benchmark (Han et al., 2021).

5.2.1 VLTransE Experimental Results

This section evaluates the performance of VLTransE on the VRD dataset, which contains 4000 images for training and 1000 images for testing. Here, the VRD dataset contains 100 object classes, 70 predicate classes, and 37,993 relationships.

Evaluation. All evaluation results are computed using the $Recall@n$ metric and evaluated on the predicate prediction task (Table 5.2 and 5.3) and tail entity prediction task (Table 5.4) similar to the metrics used in C. Lu et al., (2016) and R. Yu et al. (2017). Here, we select the $top\ n$ predicates predicted by the model, and k is the maximum number of predicates we can select for each (subject, object) pair. For the Relationship Detection task, where the ground truth bounding box and labels are not given, the IoU threshold or ratio between the Faster R-CNN predicted subject and object boxes is set to at least 0.5 with the ground truth boxes. Moreover, for each ($subject$, $object$) pair, we selected k different predicted $predicates$. The baseline model used a simple frequency counting approach, where we computed the estimated probability $P(pred | subj, obj)$. It should also be noted that we were using a random negative sampling technique, which might have resulted in suboptimal convergence. From Table 5.3, it can be observed that triplet margin loss functions yield worse performance than cross entropy loss functions. This finding is in line with analysis from Schroff et al. (2015), which indicates that triplet margin losses require semi-hard negatives to converge.

Model Name	External Knowledge	Predicate Prediction (k = 1)		Relationship Detection (k = 70)	
		R@50	R@100	R@50	R@100
Baseline by counting frequency	No	33.11	33.11	—	—
VRD (C. Lu et al., 2016)	No	47.87	47.87	13.86	14.70
VTransE (H. Zhang et al., 2017)	No	44.76	44.76	14.07	15.20
SA-Full (Peyre et al., 2017)	Yes	52.60	52.60	15.80	19.50
KL-distillation (R. Yu et al., 2017)	Yes	55.89	55.89	22.68	31.89
Zoom-Net (Yin et al., 2018)	Yes	50.69	50.69	21.37	27.30
CAI + SCA-M (Yin et al., 2018)	Yes	55.98	55.98	22.34	28.52
RelDN (J. Zhang et al., 2019)	Yes	—	—	28.15	33.91
VLTransE (Ours)	No	41.40	41.40	11.35	13.99

Table 5.2: comparison with state-of-the-art results on VRD test set (— means unavailable or unknown). None of the other models are replicated.

Table 5.2 shows that the predicate prediction and relationship prediction results between different models. We can observe that models that incorporate additional

contextual information from external data such as KL-distillation, Zoom-Net, and CAI + SCA-M, and ReIDN achieve the best results for the predicate prediction task. However, one observation is that the model with KL-distillation regularization technique can achieve similar predicate prediction performance with a recall@n of 55.89 and better relationship detection performance with a recall@50 of 22.68 with compared to the other two methods that use external knowledge. This indicates that the long-tail problem in the dataset pose a challenge to the predictive model, and the use of a regularization approach can benefit the model’s performance significantly. Besides the use of regularization techniques, we can also observe that the contrastively learned ReIDN model external Wikipedia data can achieve the highest performance on the relationship detection task with a Recall@50 of 28.15. While we also use contrastive learning objectives, our model performs worse than that of ReIDN and other models. We hypothesize that the poor performance is caused by three main reasons: (1) the naïve negative sampling strategy, (2) the lack of language likelihood bias and external language information, and (3) the non-usage of regularization techniques.

model	method	scoring function	all triples			unseen triples only	
			k=1	k=5	k=70	k=1	k=5
L^{vl}	–	$score_{consistency}$	24.18	44.20	–	7.10	22.07
$L^{vl} + L^{Tr}$	TransE	$score_{consistency}$	15.21	36.71	–	3.93	16.60
		$score_{translation}$	6.83	26.01	–	4.62	16.25
		$score_{combined}$	18.64	43.75	–	6.33	22.58
L^{vl-CE}	–	$score_{consistency}$	41.54	72.1	76.3	13.09	42.34
$L^{vl-CE} + L^{Tr}$	TransR	$score_{consistency}$	34.63	71.03	75.39	13.34	43.37
		$score_{translation}$	29.58	59.57	65.16	10.18	35.76
		$score_{combined}$	41.40	70.22	74.57	12.83	41.92

Table 5.3: Predicate prediction results on VRD test set for different loss functions using the Recall@n metrics.

Table 5.3 compares the results of different experiments and methods that we performed. For the model, we compared the use of L^{vl} which applies the triplet margin loss for visual and language consistency and the use of L^{vl-CE} which applies the cross entropy contrastive loss for visual and language consistency. We also compared these models against those that trained with the auxiliary translational loss, L^{Tr} , which preserves the first-order hierarchical structure of the (*subject, predicate, object*) triple. For these models with the translational loss, we also compare the results for the TransE

embedding method and the TransR embedding method which separates the visual object space and the visual relationship space.

The results in Table 5.3 also shows that by multiplying the visual-language consistency score ($score_{consistency}$) and the translational score ($score_{translational}$), the model’s detection results improve for both the L^{vl} and L^{vl-CE} losses for all triples in the test set. However, this could be a result from the joint training methodology. When compared the model trained on both losses against the model trained on solely the vision and language consistency loss, we can observe that the model with single objective performs better on the VRD task. This indicates that the additional translational loss is interfering with the optimization process for the VRD task. However, this negative effect can be mitigated significantly by applying TransR transformation as shown in Section 5.1.1. In fact, when evaluated on unseen triples, the TransR model trained on dual objectives has better performance than the model trained on a single cross entropy objective. This demonstrates that introducing first-order structural preserving inductive biases can benefit the scene graph representational learning process for rare visual relationships.

Another benefit of the additional translational structural loss is that the embeddings can be extended to tasks other than visual relationship detection. Like a knowledge graph, we evaluated model on the tail entity prediction task where the goal is to infer potential objects given only the subject and predicate ground truth label.

loss	method	scoring function	all triples		unseen triples only	
			Recall top@1	Recall top@5	Recall top@1	Recall top@5
$L^{vl} + L^{Tr}$	TransE	$score_{translational}$	10.72	33.80	3.51	14.37
$L^{vl-CE} + L^{Tr}$	TransR	$score_{translational}$	24.60	47.21	8.85	23.62

Table 5.4: Tail entity prediction results on VRD test set.

Table 5.4 describes the result for the tail entity prediction task for models trained with the auxiliary translational loss. The results indicate that the embeddings learned from the additional translational loss can be used to predict the object (tail entity) label given only the ground truth subject label and bounding boxes. Here, we observe that the TransE method where both the object and visual relationship embeddings exist on the same space performed worse than the TransR method which separates the object embedding and visual relationship embedding spaces. This shows that the additional neural network

projection function from the object space to the relationship space used in TransR improves the model’s overall expressivity, doubling its recall@n metric with compared with the TransE counterpart. Still, most of the predictive power of the model still depends on the bias of the given dataset. For example, the model’s predictions are biased towards the most common predicate like ‘on’ and fail to predict the relevant predicates such as ‘sit on’ or ‘lay on’. To counteract such dataset bias, we performed another experiment on a larger dataset and used the CLIP-based simple contrastive learning architecture and present the results in Section 5.3.2 below.

5.3.2 Simple Contrastive Learning Architecture Experimental Results

This section presents an evaluation of the performance of different contrastive language and image visual relationship detection models fine-tuned on the Visual Genome dataset. Here, we first evaluate the model using the Recall@n metric on the entire test set to compare with other work. It should be noted here that Recall@n is not the same metric as the one described in Table 5.2 because we are only evaluating on the predicate classification task using a different benchmark dataset. Different from the Recall@n in Table 2, where n predictions can reach the combined total of subject, predicate, and object, the scope of potential predictions n for this section is confined to the 150 predicate labels. We then optimize the models’ thresholds using the one-vs-all Youden’s J Statistic (see Section 3.2) and evaluate the model using the Informedness on both the default and optimized thresholds.

Model	Layers	Hidden size	# Attention Heads	#Params
ViT-Base	12	768	12	86M
ViT-Large	24	1024	16	307M
ViT-Huge	32	1280	16	632M

Table 5.5: Statistics of the ViT models (Dosovitskiy et al., 2020).

Table 5.5 shows the scale of different Vision Transformer encoder. Here, the smallest model ViT-Base consists of 12 encoder layers with 86 million parameters. ViT-Large has 24 encoder layers and 3.56 times the number of parameters as ViT-Base, and ViT-Huge has 32 encoder layers with twice as many parameters as ViT-Large. ViT-Base consists of only 12 attention heads at each encoder layer, whereas the ViT-Large and ViT-Huge have 16 attention heads at each encoder layer.

Model	Visual Encoder	Finetuned MLP	Finetuned Attention
CLIP-ResNet50-laion400m	ResNet50	53.2	–
CLIP-ResNet101-laion400m	ResNet101	47.7	–
OpenCLIP-ViT-B/16-laion400m	ViT-B/16	42.2	59.1
OpenCLIP-ViT-B/32-laion400m	ViT-B/32	57.0	59.8
OpenCLIP-ViT-B/32-laion2b-en	ViT-B/32	50.35	60.0
OpenCLIP-ViT-L/14-laion400m	ViT-L/14	55.6	61.8
OpenCLIP-ViT-L/14-laion2b-en	ViT-L/14	54.8	62.0
OpenCLIP-ViT-H/14-laion2b-en	ViT-H/14	51.4	63.1

Table 5.6: Predicate Classification Results using the Recall@1 metrics for different contrastive models on the Visual Genome dataset.

Table 5.6 describes the Recall@1 results for the different contrastive models. Here, results for two fine-tuning strategies are shown for these models: (i) *Finetuned MLP* and (ii) *Finetuned Attention*. In the (i) approach, MLPs were attached on top of the frozen visual and language encoders, whereas in the (ii) approach, the last two encoder blocks of the Transformer were unfrozen for both language and image. The results indicate that fine-tuning these Transformer-based models without unfreezing the attention layers did not lead to better performance as the model size was increased. On the contrary, fine-tuning the Transformer models with unfrozen attention layers yield a consistent improvement in the model’s performance as the model size increases. Here, ViT-B/32 is smaller than ViT-L/14, and ViT-L/14 is smaller than ViT-H/14 (see Table 5.5). From Table 5.6, a slight improvement in performance was observed between models pretrained on the *laion400m* dataset and those pre-trained on the larger *laion2b-en* dataset. For Vision Transformer models, an input image can be split into either 14x14 patches, 16x16 patches, or 32x32 patches. By comparing the model with ViT-B/16 visual encoder and the model with ViT-B/32 encoder trained on the *laion400m* dataset, we observe that larger patch sizes can slightly improve the model’s performance. Since a model with a larger patch size has a shorter sequence length and is less expensive to run, the ViT-B/32 model generalizes better than that the ViT-B/16 model.

Table 5.7 describes the predicate classification results on the Recall@1 metric for our model and other existing benchmark models. Results that were marked with ¹ have been computed by retraining the models using the PyTorch implementation from Han et al. (2021), and results that were marked with ² are not replicated. From Table 5.7, we can observe that the CLIP-based model performs reasonably well when compared with other

state-of-the-art models on the Recall@1 metric despite having little inductive biases from structural or spatial configurations. Still, it can be flawed to analyse the model’s performance based solely on Recall@1 since the model can still be biased toward the commonly occurring classes. This can be observed in Figure 5.4 below, where most of the model’s predictions are concentrated on the most prevalent classes.

Model	Visual Encoder	PredCls
IMP ¹ (Xu et al., 2017)	ResNet50	57.6
MSDN ¹	ResNet50	59.6
G-RCNN ¹ (Yang et al., 2018)	ResNet50	59.94
ReIDN ²	ResNet50	60.9
Neural Motif ¹	ResNet50	63.0
GPS-Net ²	ResNet50	66.9
ITS+RTS ²	ResNet101	67.3
OpenCLIP-ViT H/14-laion2b-en	ViT-H/14	63.1

Table 5.7: Comparing Predicate Prediction Results on Visual Genome dataset using the recall metrics. The replicated results are slightly different from those in Han et al., (2021).

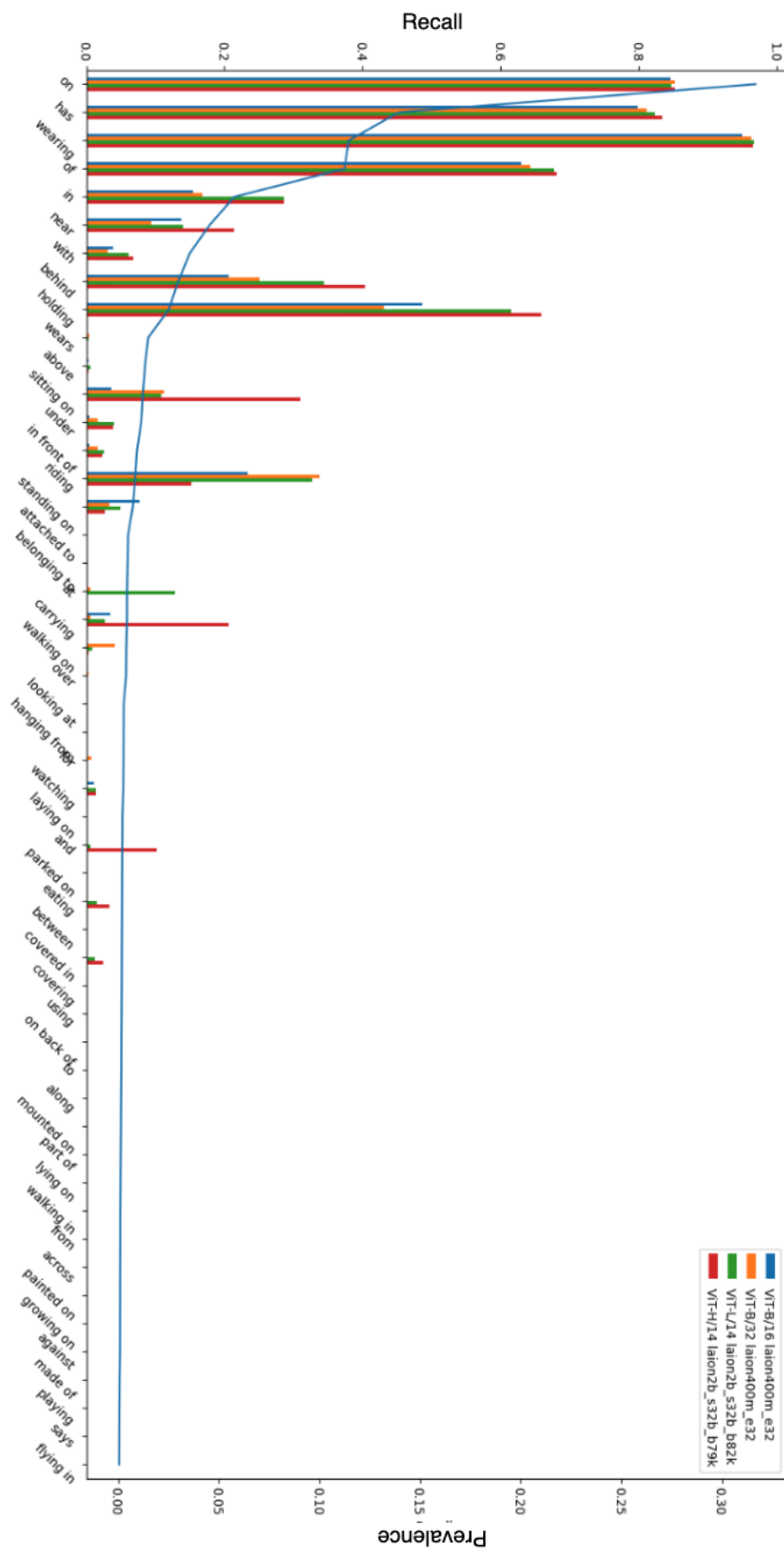


Figure 5.4: Predicate Prediction Results on Visual Genome dataset by individual relationship class. The blue line represents the prevalence of the class labels in the dataset.

Figure 5.4 presents a visualization of how models of different sizes perform on individual predicate classes on the Recall@1 metric. Here, we observe that the contrastive model trained using ViT-Huge/14 performs better than the smaller models on relationships like ‘sitting on’ and ‘above’ without penalizing the performance of relationship ‘on’. However, the ViT-Huge/14 performs worse than the smaller models on predicate ‘riding’, leading us to believe that there are little differences between the model of different sizes. Still, one common observation across all models is that the model’s performance on a predicate class is highly dependent on the number of occurrences of that class indicated by the blue line.

To better compare the different models above, we also computed the confusion matrices for our models’ predictions and the replicated models’ predictions and plotted the one-vs-all ROC curves to calculate the Informedness metric, which measures the probability that the model is making an informed decision over chance. Here, we also evaluate the informed decision after optimizing the prediction thresholds for each predicate class. We also computed the mean Recall metric which takes the average of the one-vs-all Recalls for every predicate class.

Model	Biased Priori	Mean Recall	Informedness	
			Default threshold	Optimized threshold
IMP ¹	No	0.168	0.591	0.639
	Yes	0.122	0.641	0.707
MSDN ¹	No	0.151	0.590	0.629
	Yes	0.159	0.662	0.719
G-RCNN ¹	No	0.138	0.526	0.577
	Yes	0.166	0.658	0.708
Neural Motif ¹	No	0.112	0.659	0.682
	Yes	–	–	–
OpenCLIP-ViT H/14-laion2b-en	No	0.118	0.640	0.704

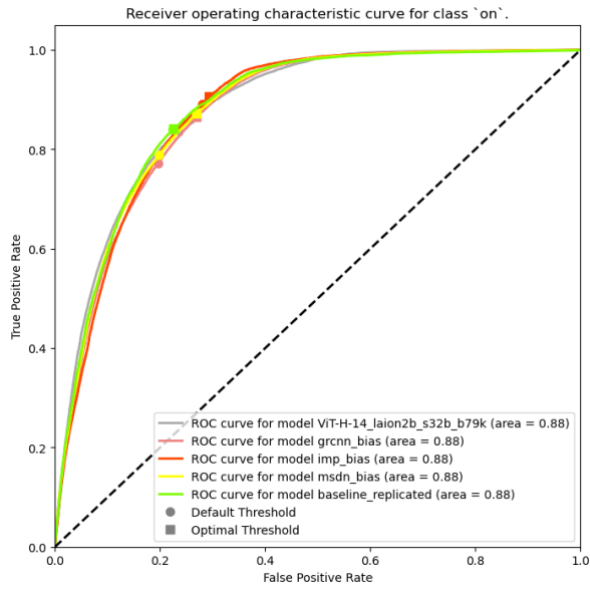
Table 5.8: Comparing Predicate Prediction Results on VG150 dataset with other work using the mean Recall and Informedness metric.

Table 5.8 compares the performance of our model and other replicated models. The biased priori column indicates whether the model adds an additional learned co-occurrence logit vector. Mean Recall is the average of the Recall of all predicate classes, and Informedness is the biased weighted sum of one-vs-all Youden’s J Statistics. While the IMP model without added biases achieved the highest mean recall of 0.168, the

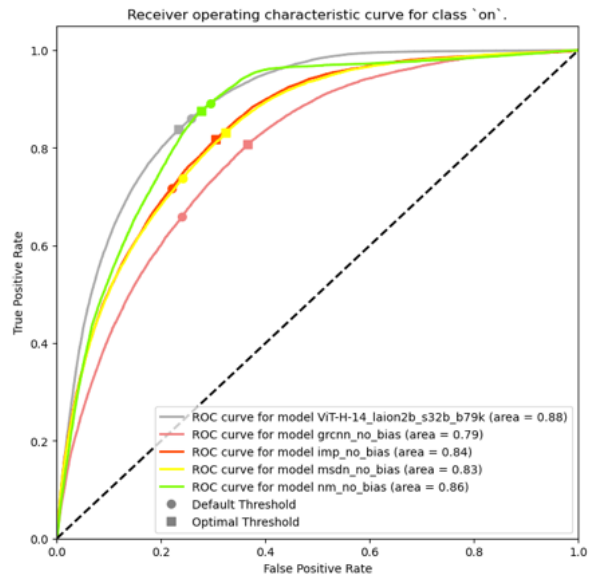
MSDN model achieved the highest Informedness of 0.662 with the default threshold. We can also observe that most replicated models with added co-occurrence biases have improved performance when compared to those without on the mean recall metric. Interestingly, the iterative message passing (IMP) approach with added co-occurrence biases performed worse on mean recall but better on Informedness when compared against IMP without the added co-occurrence biases.

We also observe that the simplified contrastive OpenCLIP model without any layout constraints or biases perform worse than other models on the mean Recall metric, indicating that the model’s prediction is not as diverse as the others. In contrast, the IMP model with customized structure layouts for message passing performed better than the OpenCLIP model. This demonstrates that larger models without inductive biases do not necessarily perform better than models with engineered inductive biases on long-tailed relationships. However, the mean recall metric can be a pessimistic evaluation for all models since it assigns equal weighting to every predicate class which do not reflect long-tail nature of the problem. Thus, we also compute the biased weighted Youden’s J Statistics or Informedness for the model’s prediction. While the OpenCLIP model does not perform as well as the other replicated models using the default threshold, it performs better than the optimized models without the added co-occurrence biases. Still, the models with the added biased priori perform better than OpenCLIP on their default threshold.

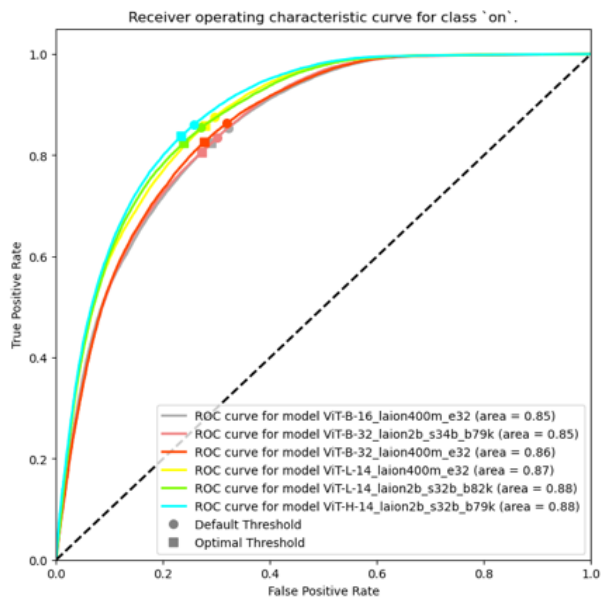
Here, the thresholds are optimized using one-vs-all Youden’s J Statistics which measures the model’s absolute performance over the chance line indicated by the dotted diagonal line ($TPR = FPR$) on the ROC curves. To better visualize the optimization process, we also plotted the ROC curves to show the trade-offs between the true positive rate (TPR) and false positive rate (FPR) at different thresholds. For example, we can observe the model’s performance for the predicate ‘on’ and ‘lying on’ in Figure 5.5 and Figure 5.6 below. An ideal model should have a curve that is as close to the optimal (0,1) point on as possible. For example, if we compare the ROC curves for the most common predicate ‘on’, we can see that our OpenCLIP model with ViT-H/14 has a similar shape with other scene graph benchmark models with co-occurrence bias (Figure 5.5a), indicating that these models have similar performances for the given predicate. On the contrary, the ROC curve for our OpenCLIP model with ViT-H/14 model rises faster and is closer to the optimal point than the ROC curves for other scene graph benchmark models without co-occurrence bias (Figure 5.5b), indicating that our model has better performance.



(a)

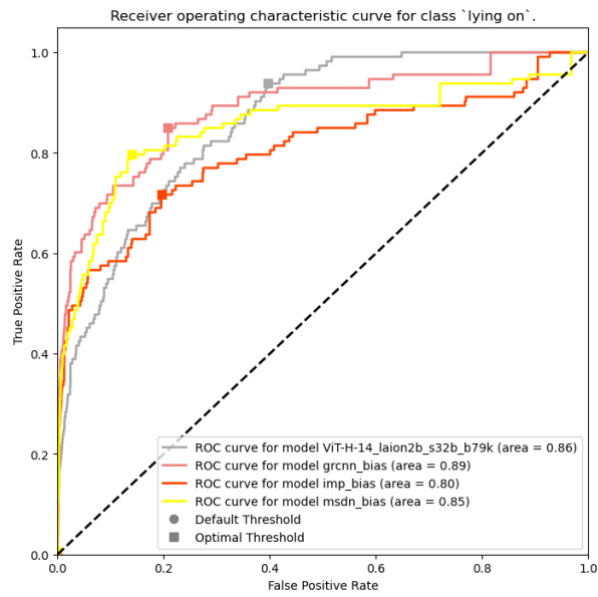


(b)

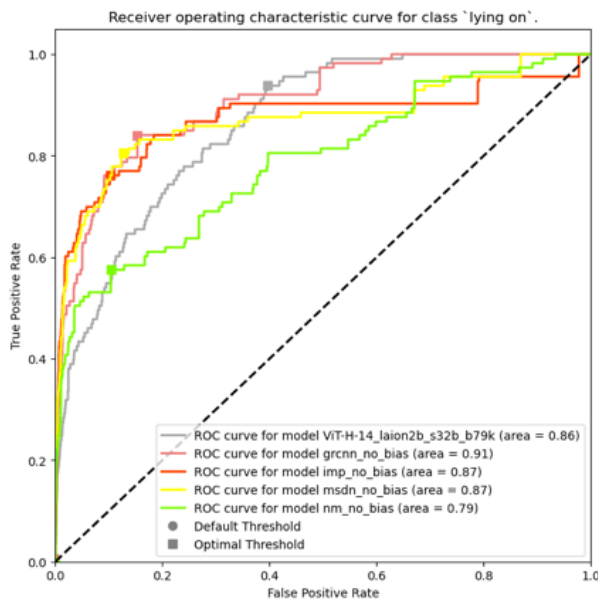


(c)

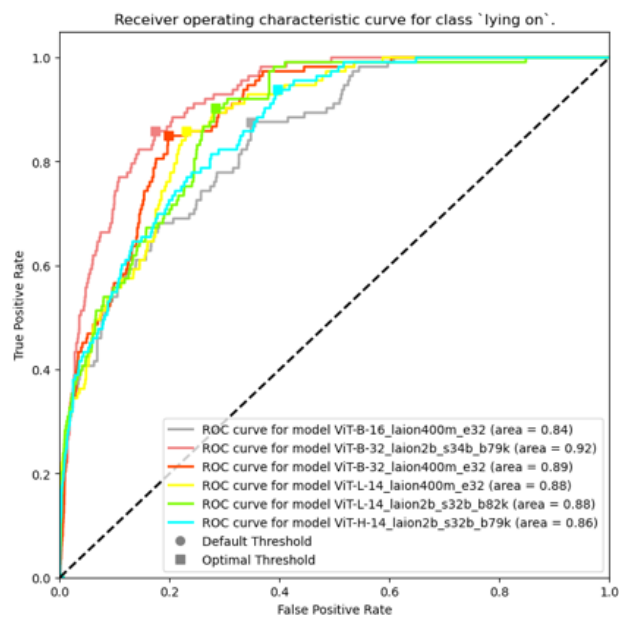
Figure 5.5: Receiver operating characteristic curves for the predicate class 'on'.



(a)



(b)



(c)

Figure 5.6: Receiver operating characteristic curves for the predicate class 'lying on'.

In Figure 5.5c, we can also observe the ROC curve for a larger model contrastive model is closer to the optimal point than the ROC curves for the smaller counterparts, indicating a consistent improvement in performance as we scale up the model for the predicate ‘on’.

Different from the ROC curves for the predicate ‘on’ shown in Figure 5.5, the ROC curves for predicate ‘lying on’ in Figure 5.6 tells a different story. First, because there are fewer test samples for the predicate ‘lying on’ than those for the predicate ‘on’, there are fewer thresholds, making the curves more jagged. This time, the ROC curve for the OpenCLIP model with ViT-H/14 rises slower than the ROC curves for other models in both Figure 5.6a and Figure 5.6b, indicating that OpenCLIP model with ViT-H/14 is not the best model for predicting the predicate ‘lying on’. In Figure 5.6c, we can also observe that scaling up the size of the model does not necessary improve the performance of the predictor. In fact, when comparing the area under the curve for these ROC Curves in Figure 5.6a, Figure 5.6b, and Figure 5.6c, we can observe that the most potential model is the OpenCLIP model with ViT-B/32 pre-trained on the LAION_2B dataset. Because there are a total of 50 predicate classes, we can only show a few examples here. The top ROC curves for the top 16 predicate classes are shown in Appendix B.

Ground Truth	Predicted	Ground Truth	Predicted
(man, in, shirt)	(man, wearing, shirt)	(person, in, snow)	(person, on, snow)
(wing, on, plane)	(wing, of, plane)	(man, sitting on, bench)	(man, on, bench)
(man, wears, shirt)	(man, wearing, shirt)	(ear, on, cat)	(ear, of, cat)
(man, has, leg)	(man, wearing, leg)	(car, parked on, street)	(car, on, street)
(window, of, building)	(window, on, building)	(man, riding, horse)	(man, on, horse)
(man, has, shirt)	(man, wearing, shirt)	(window, in, building)	(window, on, building)
(women, with, hair)	(women, has, hair)	(wheel, of, bike)	(wheel, on, bike)
(window, of, train)	(window, on, train)	(person, riding, horse)	(person, on, horse)
(man, with, hair)	(man, has, hair)	(window, near, building)	(window, on, building)
(man, riding, bike)	(man, on, bike)	(woman, in, shirt)	(woman, wearing, shirt)
(window, of, bus)	(window, on, bus)	(man, holding, surfboard)	(man, on, surfboard)
(leg, on, giraffe)	(leg, of, giraffe)	(man, has, shoe)	(man, wearing, shoe)
(head, on, person)	(head, of, person)	(person, sitting on, bench)	(person, on, bench)
(man, riding, skateboard)	(man, on, skateboard)	(leg, on, zebra)	(leg, of, zebra)
(tail, on, plane)	(tail, of, plane)	(man, in, jacket)	(man, wearing, jacket)

Table 5.9: Top 30 misclassified visual relationships made by the OpenCLIP model with ViT-B/32 pretrained LAION_400M.

Table 5.9 presents the thirty most notable misclassifications made by the OpenCLIP model with ViT-B/32 visual encoder. From an analysis of the textual triples alone, it is evident that the predictions deemed incorrect by the evaluation code are not entirely merit. For instance, the top misclassified prediction, which involves the triplet (man, wearing, shirt), can also be interpreted as (man, in, shirt) or (man, has, shirt). Similarly, the predictions (window, on, train) or (window, on, bus) correctly express the intended meaning but are nevertheless evaluated as erroneous. These observations show the inherent limitations of the VG150 benchmark, where the constructed predicate classes and triples are ambiguous and inconsistent. Thus, the evaluation results may not convey an accurate and holistic picture of how the model would perform. To further demonstrate the ambiguous nature of the labelled triples, we also show six examples of misclassified visual relationships along with the ground truth image and label in Figure 5.7.




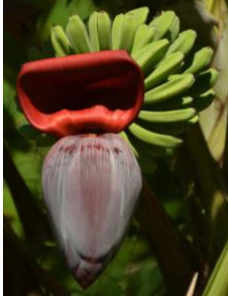

		
<p>GT: (vase, with, flower) Pred: (vase, has, flower)</p>	<p>GT: (building, with, clock) Pred: (building, has, clock)</p>	<p>GT: (girl, with, kite) Pred: (girl, holding, kite)</p>
		
<p>GT: (dog, eating, food) Pred: (dog, looking at, food)</p>	<p>GT: (bannana, growing on, tree) Pred: (bannana, on, tree)</p>	<p>GT: (window, of, bus) Pred: (window, on, bus)</p>

Figure 5.7: A visualization of misclassified visual relationships made by the OpenCLIP model with ViT-B/32 pretrained LAION_400M.

In the six images in Figure 5.7, we can see that either the ground truth relationships or the predicted relationships can be interpreted as the correct answers. In a way, the problem stems from the ambiguity and confounding nature of the fifty visual predicates in the dataset, making it difficult to truly evaluate and compare the performance of different models quantitatively.

5.3 Discussion

In this work, we proposed the use of contrastive learning as a tool for learning scene graph embeddings that can be used to tackle the VRD task. While the proposed VLTransE architecture performance is lower than existing benchmark models (see Table 5.3), we speculate that part of the issue can be traced back to the naïve implementation of the random negative sampling strategy which only selects a few negative examples for each batch. Still, further experiments are needed to verify this speculation. From the VLTransE training and evaluation results, we observed two interesting findings related to the architectural and loss design space. First, optimizing two different objectives such as the vision-language consistency loss and the translational loss on the same embedding space led to suboptimal convergence and poor final performance on the Recall metric. To tackle this problem, there should be a clear separation of responsibility between the embedding space that preserves the translational property of the visual relationship and the embedding space that holds the relevant concepts for subjects and objects. This architectural change is also known as TransR knowledge graph embedding, and it gives the model the additional ability to perform knowledge graph completion tasks such as predicting the tail entity class (see Table 5.4). Second, both translational loss and vision-language consistency loss are contrastive learning objectives, which requires a large negative batch size and a higher GPU memory usage. Despite utilizing the same set of negative examples, the model requires additional gradients for the translational loss in addition to the vision-language consistency loss. This requirement for negative examples introduces complexity to the training of both the visual and textual encoders as we have to maintain a sufficient number of negatives in each batch. Consequently, this leads to an increase in the number of training epochs and an overfitting model.

In our simple contrastive vision and language model (Section 4.3), we directly optimized for the VRD task and reduced the memory footprint during training by removing the translational loss and freezing most of the parameters. During this process, we also made use of the large scale pre-trained vision and language models called CLIP. While we observed that scaling up the model from ViT-B to ViT-L to ViT-H improved the overall model’s performance, scaling up the number pre-trained image and text pair showed little improvements (see Table 5.6). Moreover, despite the use of contrastive learning, we conclude that scaling up the model and dataset alone is inefficient and does not guarantee a high performance across different classes on the VG150 benchmark (see Table 5.8). For example, based on the mean Recall results in Table 5.8, we can observe

that the CLIP models performed worse than existing models on rarer classes. Still, with enough negative examples and GPU memory, we believe that contrastive learning can learn distributed representations from the scene graphs labels. However, the approach should take graph structural layout priors into account when designing the network to improve the overall learning efficiency for the targeted task.

From the ROC curves presented in Figure 5.5 and Figure 5.6 (more in Appendix B), we can see that the CLIP-based model with ViT-H encoder performed on par or even better than other replicated models on commonly occurred classes, while it performed worse than other models on rare classes like ‘on back of’, ‘painted on’, or ‘over’. This long tail problem can also be clearly observed in Figure 5.4, where the models performed well on classes with high prevalence and poorly on tail classes with low prevalence. Still, the model performance cannot be observed through the algorithmic lens alone. By examining the misclassified visual relationships qualitatively, we observed that data inconsistency and ambiguity penalize the quantitative performance of the model even when the predictions are not necessarily incorrect. This shows that further work on the dataset and not the model is needed to build a benchmark that accurately represents the structure of the visual world.

While both the VLTransE and CLIP-based experiments offer insight into their respective areas of VRD, the differences in architectural design, benchmark used, and experimental objective make a direct comparison difficult. Mainly, the VLTransE model was designed to assess the value of first-order translational structural prior for VRD, using a smaller benchmark specific to this aim. On the other hand, the CLIP-based model was created to investigate the feasibility of applying large-scale pretrained contrastive image and text models to the VRD task, necessitating a larger dataset and a different benchmark. Moreover, the differences in training techniques and scale of the models further hinder any direct comparison. Still, we can make an indirect and non-straightforward comparison through observations while training these two models.

From the VLTransE experiment, we can speculate how techniques such as the addition of TransE structural constraints might have an impact on the CLIP-based model. While we speculate that the incorporation of TransE does not improve the overall performance of the CLIP-based model, the VLTransE experiment did highlight the importance of maintaining a clear separation of embedding spaces through techniques such as TransR, which can be beneficial to the CLIP-based model. Moreover, results from

the CLIP-based experiment suggest that the integration of additional spatial information, a technique similar to other benchmarked models like MSDN, can enhance the model's overall performance on the VRD task. However, we acknowledge that these are speculations, and it is of considerable value for future research to facilitate a fair and direct comparison between the techniques used for these two experiments.

To facilitate such a comparison, we could modify the original CLIP architecture to include multiple heads for the vision and language encoders, similar to the VLTransE experiment. The model could then be trained contrastively using multiple InfoNCE losses. One of the main challenges lies in the appropriate integration of the subject and object positions into the Vision Transformer architecture used by the CLIP model. Our CLIP-based experiment cropped the relevant union image region, but this method fails to incorporate additional contextual visual information or clearly separate the features for each of the subject, predicate, and object heads. One potential workaround involves crafting additional attention masks for subject, object, and their union regions. These crafted masks could then refocus the model's attention at later layers to create subject, predicate, and object embeddings. This can be achieved either via additional learning objectives or by taking a weighted average between the crafted attention masks and the learned attention masks during training. Afterwards, we can introduce translational losses alongside the contrastive losses to effectively gauge the impact of TransE and TransR approaches on the CLIP-based model.

In conclusion, both the VLTransE and CLIP-based experiments, despite their divergent benchmarks and experimental goals, enrich our understanding of the VRD task, highlighting both the strengths and limitations of current methodologies. Lessons learned, such as the need for embedding space delineation and the limitations of mere model and data scaling, illuminate potential pitfalls and set a trajectory for future research. Future models could potentially draw from both approaches, creating a unified model that can be used to effectively measure and compare the impact of the different proposed techniques in this work. Therefore, while direct comparison isn't feasible currently, these experiments pave the way for further refinement in the VRD space.

Chapter 6

Conclusions and Future Work

The aim of this work was to develop a visual relationship detection system using contrastive representational learning. To this end, we interpreted a scene graph as a knowledge graph and built a contrastively learned knowledge graph embedding model called VLTransE that preserves the first-order structure of the scene graph. Here, we examined the impact of the TransE knowledge graph embedding technique on the contrastively learned model and found that additional constraints on (*subject, predicate, object*) adversely affected the performance of the VRD task. Thus, we separated the entity embedding space and the predicate embedding space by adding additional projection functions from the entity space to the predicate space. Through this architectural change, the negative effect of the translational constraint was reduced, and we were able to extend the contrastively learned embeddings' capabilities to the tail entity prediction task. However, both translational loss and vision-language consistency loss required heavy memory resource usage which is not readily available.

Taking a divergent approach from VLTransE, we also proposed the new use case of a simplified visual and language contrastive learning model called CLIP for the VRD task. Here, we explored the impact of scale on VRD and found that scaling up the model improved the overall performance consistently on all metrics, including the Recall and the Informedness metrics. While the model achieved comparable results on the Visual Relationship Detection task to other benchmarked models, the lack of inductive biases such as spatial priors or structure-preserving techniques prevented the model from generalizing to rare predicate classes as seen from the mean Recall metric or ROC curves. This indicated that scaling up deep learning model blindly is not an efficient approach to tackle the Visual Relationship Detection problem which suffered mainly from the long-tailed distribution. To prevent overfitting while overcoming the GPU memory constraint, we also proposed a new adaptive negative sampling approach by sampling the inverse distribution of the selected samples for a given relationship triple with careful consideration of other positive relationships in the given image. Still, the negative sampling strategy is only applied for language within the context of the local image, showing that there is room for further improvement on negative sampling techniques in

future work. This highlights an important trade-off when using contrastive learning approach in the self-supervised learning framework. While contrastive learning can benefit from having a large training batch size, every additional sample requires more memory storage for the models' gradients, leading to an ever-increasing need for memory as batch size increases.

In general, the results from the above two models indicate that an encoder-encoder contrastive learning can be applied to tackle the visual representational learning problem for VRD. However, because of the high memory demand, the optimal approach moving forward is not to use solely contrastive learning, but to pair it with other self-supervised learning losses and architectural changes. While we presented the use case of translational knowledge graph embedding techniques on the scene graph dataset for VRD, the method only preserves the first-order hierarchical structure of the scene graph. Thus, future study can examine knowledge graph embedding techniques that preserve higher level structure of the scene graph. Another challenge that needs to be addressed is the long-tail nature of the visual relationship triples. While the scene graph datasets provide us a structured knowledge representation of the visual scene, representational learning of sparse data structure using deep learning approaches are highly susceptible to biases. Thus, future work may also explore new loss engineering approaches, including the use of regularization losses, clustering losses, or contrastive learning losses during the representational learning process. Still, focusing on improving the neural architecture and loss function do not necessarily lead to a better model.

We must also re-examine the validity of the common benchmarks used to build these visual relationship detection systems. In a way, the curation process for these benchmarks, especially the VG150 dataset, over-relied on heuristics and automation techniques, leading to inconsistency and ambiguity in the data labels. While this ambiguity can be thought as a feature of natural language, such a feature can make it difficult to truly evaluate and compare the model's performance. Thus, future work should seek to improve the existing benchmarks or define a narrower visual relationship detection task for a more specific use case. Another limitation of this study is that the scene graph dataset required expensive manual curation processes. Thus, future work may explore newer automation approaches for generating and combining both structured and unstructured sub-symbolic representations, curating a dataset that incorporate not only the scene-level structures but also the relevant background knowledge from unstructured language. Finally, this work did not explore the explainability for these deep learning

models, leading to the blind assumption that the models learn the relevant features for the visual relationships. This is most likely not the case for our simple contrastive vision and language architecture since we did not introduce any spatial priors or task to the learning model. Future work can explore the explainability problem in more details for higher level visual relationships.

Bibliography

- Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., & Parikh, D. (2015). VQA: Visual Question Answering. *2015 IEEE International Conference on Computer Vision (ICCV)*, 2425–2433.
<https://doi.org/10.1109/ICCV.2015.279>
- Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2019). *Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI* (arXiv:1910.10045; Version 1). arXiv. <http://arxiv.org/abs/1910.10045>
- Balntas, V., Riba, E., Ponsa, D., & Mikolajczyk, K. (2016). Learning local feature descriptors with triplets and shallow convolutional neural networks. *Proceedings of the British Machine Vision Conference 2016*, 119.1-119.11.
<https://doi.org/10.5244/C.30.119>
- Belongie, S., Malik, J., & Puzicha, J. (2000). Shape Context: A New Descriptor for Shape Matching and Object Recognition. *Advances in Neural Information Processing Systems, 13*.
<https://proceedings.neurips.cc/paper/2000/hash/c44799b04a1c72e3c8593a53e8000c78-Abstract.html>
- Bengio, Y., LeCun, Y., & Henderson, D. (1993). Globally Trained Handwritten Word Recognizer using Spatial Representation, Convolutional Neural Networks, and Hidden Markov Models. *Advances in Neural Information Processing Systems, 6*.
<https://proceedings.neurips.cc/paper/1993/hash/3b5dca501ee1e6d8cd7b905f4e1bf723-Abstract.html>

- Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., & Yakhnenko, O. (2013). Translating Embeddings for Modeling Multi-relational Data. *Advances in Neural Information Processing Systems*, 26.
<https://papers.nips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html>
- Bosman, P. A. N., & Thierens, D. (n.d.). *Negative Log-Likelihood And Statistical Hypothesis Testing As The Basis Of Model Selection In IDEAs*.
- Cao, K., Wei, C., Gaidon, A., Arechiga, N., & Ma, T. (2019). Learning Imbalanced Datasets with Label-Distribution-Aware Margin Loss. *Advances in Neural Information Processing Systems*, 32.
<https://proceedings.neurips.cc/paper/2019/hash/621461af90cadfdaf0e8d4cc25129f91-Abstract.html>
- Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-End Object Detection with Transformers. *ArXiv:2005.12872 [Cs]*.
<http://arxiv.org/abs/2005.12872>
- Chen, J., Lu, Y., Yu, Q., Luo, X., Adeli, E., Wang, Y., Lu, L., Yuille, A. L., & Zhou, Y. (2021). *TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation* (arXiv:2102.04306). arXiv. <http://arxiv.org/abs/2102.04306>
- Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A Simple Framework for Contrastive Learning of Visual Representations. *Proceedings of the 37th International Conference on Machine Learning*, 1597–1607.
<https://proceedings.mlr.press/v119/chen20j.html>
- Chen, T., Yu, W., Chen, R., & Lin, L. (2019). Knowledge-Embedded Routing Network for Scene Graph Generation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6156–6164.
<https://doi.org/10.1109/CVPR.2019.00632>

- Chen, Y.-C., Li, L., Yu, L., Kholy, A. E., Ahmed, F., Gan, Z., Cheng, Y., & Liu, J. (2020). UNITER: UNiversal Image-Text Representation Learning. *ArXiv:1909.11740 [Cs]*. <http://arxiv.org/abs/1909.11740>
- Chen, Z., Wang, Y., Zhao, B., Cheng, J., Zhao, X., & Duan, Z. (2020). Knowledge Graph Completion: A Review. *IEEE Access*, 8, 192435–192456. <https://doi.org/10.1109/ACCESS.2020.3030076>
- Cheng, J., Dong, L., & Lapata, M. (2016). Long Short-Term Memory-Networks for Machine Reading. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 551–561. <https://doi.org/10.18653/v1/D16-1053>
- Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K., & Bengio, Y. (2015). Attention-Based Models for Speech Recognition. *Advances in Neural Information Processing Systems*, 28. <https://proceedings.neurips.cc/paper/2015/hash/1068c6e4c8051cfd4e9ea8072e3189e2-Abstract.html>
- Cireşan, D. C., Meier, U., Masci, J., Gambardella, L. M., & Schmidhuber, J. (2011). *High-Performance Neural Networks for Visual Object Classification* (arXiv:1102.0183). arXiv. <http://arxiv.org/abs/1102.0183>
- Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297. <https://doi.org/10.1007/BF00994018>
- Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 1, 886–893 vol. 1. <https://doi.org/10.1109/CVPR.2005.177>
- Davis, J., & Goadrich, M. (2006). The relationship between Precision-Recall and ROC curves. *Proceedings of the 23rd International Conference on Machine Learning*, 233–240. <https://doi.org/10.1145/1143844.1143874>

- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255.
<https://doi.org/10.1109/CVPR.2009.5206848>
- Desai, A., Wu, T.-Y., Tripathi, S., & Vasconcelos, N. (2021). Learning of Visual Relations: The Devil is in the Tails. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 15384–15393.
<https://doi.org/10.1109/ICCV48922.2021.01512>
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.
ArXiv:1810.04805 [Cs]. <http://arxiv.org/abs/1810.04805>
- Dong, Y., Cordonnier, J.-B., & Loukas, A. (2021). Attention is Not All You Need: Pure Attention Loses Rank Doubly Exponentially with Depth. *ArXiv:2103.03404 [Cs]*. <http://arxiv.org/abs/2103.03404>
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Hounsby, N. (2020a). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ArXiv:2010.11929 [Cs]*. <http://arxiv.org/abs/2010.11929>
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Hounsby, N. (2020b). An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ArXiv:2010.11929 [Cs]*.
- Ericsson, L., Gouk, H., Loy, C. C., & Hospedales, T. M. (2022). Self-Supervised Representation Learning: Introduction, Advances and Challenges. *IEEE Signal Processing Magazine*, 39(3), 42–62. <https://doi.org/10.1109/MSP.2021.3134634>

- Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2), 303–338. <https://doi.org/10.1007/s11263-009-0275-4>
- Fei-Fei, L., Fergus, R., & Perona, P. (2006). One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4), 594–611. <https://doi.org/10.1109/TPAMI.2006.79>
- Frosst, N., Papernot, N., & Hinton, G. (2019). Analyzing and Improving Representations with the Soft Nearest Neighbor Loss. *Proceedings of the 36th International Conference on Machine Learning*, 2012–2020. <https://proceedings.mlr.press/v97/frosst19a.html>
- Ge, W., Huang, W., Dong, D., & Scott, M. R. (2018). Deep Metric Learning with Hierarchical Triplet Loss. In V. Ferrari, M. Hebert, C. Sminchisescu, & Y. Weiss (Eds.), *Computer Vision – ECCV 2018* (Vol. 11210, pp. 272–288). Springer International Publishing. https://doi.org/10.1007/978-3-030-01231-1_17
- Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., & Wichmann, F. A. (2020). Shortcut Learning in Deep Neural Networks. *Nature Machine Intelligence*, 2(11), 665–673. <https://doi.org/10.1038/s42256-020-00257-z>
- Gentile, C., & Warmuth, M. K. K. (1998). Linear Hinge Loss and Average Margin. *Advances in Neural Information Processing Systems*, 11. <https://proceedings.neurips.cc/paper/1998/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html>
- Gutmann, M., & Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. *Proceedings of the Thirteenth*

International Conference on Artificial Intelligence and Statistics, 297–304.

<https://proceedings.mlr.press/v9/gutmann10a.html>

Han, X., Yang, J., Hu, H., Zhang, L., Gao, J., & Zhang, P. (2021). *Image Scene Graph Generation (SGG) Benchmark* (arXiv:2107.12604). arXiv.

<https://doi.org/10.48550/arXiv.2107.12604>

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2018). *Mask R-CNN*

(arXiv:1703.06870). arXiv. <https://doi.org/10.48550/arXiv.1703.06870>

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *ArXiv:1512.03385 [Cs]*. <http://arxiv.org/abs/1512.03385>

Herskovits, A. (1986). *Language and Spatial Cognition*. Cambridge University Press

Retrieved June 1, 2022, from

https://www.academia.edu/44336750/Herskovits_Language_and_Spatial_Cognition_Cambridge_University_Press_1986_2009_

Hermans, A., Beyer, L., & Leibe, B. (2017). In Defense of the Triplet Loss for Person Re-Identification. *ArXiv:1703.07737 [Cs]*. <http://arxiv.org/abs/1703.07737>

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5), 359–366.

[https://doi.org/10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8)

Hospedales, T., Antoniou, A., Micaelli, P., & Storkey, A. (2022). Meta-Learning in Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(09), 5149–5169.

<https://doi.org/10.1109/TPAMI.2021.3079209>

Hudson, D. A., & Manning, C. D. (2019). Learning by Abstraction: The Neural State Machine. *ArXiv:1907.03950 [Cs]*. <http://arxiv.org/abs/1907.03950>

Hung, Z.-S., Mallya, A., & Lazebnik, S. (2020). Contextual Translation Embedding for Visual Relationship Detection and Scene Graph Generation. *IEEE Transactions*

on Pattern Analysis and Machine Intelligence, 1–1.

<https://doi.org/10.1109/TPAMI.2020.2992222>

Hyvärinen, A. (2005). Estimation of Non-Normalized Statistical Models by Score Matching. *Journal of Machine Learning Research*, 6(24), 695–709.

Ilinykh, N., & Dobnik, S. (2021). How Vision Affects Language: Comparing Masked Self-Attention in Uni-Modal and Multi-Modal Transformer. *Proceedings of the 1st Workshop on Multimodal Semantic Representations (MMSR)*, 45–55.

<https://aclanthology.org/2021.mmsr-1.5>

Ilinykh, N., & Dobnik, S. (2022). Attention as Grounding: Exploring Textual and Cross-Modal Attention on Entities and Relations in Language-and-Vision Transformer. *Findings of the Association for Computational Linguistics: ACL 2022*, 4062–4073. <https://doi.org/10.18653/v1/2022.findings-acl.320>

Jaiswal, A., Babu, A. R., Zadeh, M. Z., Banerjee, D., & Makedon, F. (2021). A Survey on Contrastive Self-Supervised Learning. *Technologies*, 9(1), Article 1.

<https://doi.org/10.3390/technologies9010002>

Ji, S., Pan, S., Cambria, E., Marttinen, P., & Yu, P. S. (2021). A Survey on Knowledge Graphs: Representation, Acquisition and Applications. *IEEE Transactions on Neural Networks and Learning Systems*, 1–21.

<https://doi.org/10.1109/TNNLS.2021.3070843>

Johnson, J., Krishna, R., Stark, M., Li, L.-J., Shamma, D. A., Bernstein, M. S., & Fei-Fei, L. (2015). Image retrieval using scene graphs. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3668–3678.

<https://doi.org/10.1109/CVPR.2015.7298990>

Keith, M. G., Tay, L., & Harms, P. D. (2017). Systems Perspective of Amazon Mechanical Turk for Organizational Research: Review and Recommendations.

Frontiers in Psychology, 8.

<https://www.frontiersin.org/article/10.3389/fpsyg.2017.01359>

Krishna, R., Chami, I., Bernstein, M., & Fei-Fei, L. (2018). Referring Relationships.

2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition,

6867–6876. <https://doi.org/10.1109/CVPR.2018.00718>

Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis,

Y., Li, L.-J., Shamma, D. A., Bernstein, M. S., & Fei-Fei, L. (2017). Visual

Genome: Connecting Language and Vision Using Crowdsourced Dense Image

Annotations. *International Journal of Computer Vision*, 123(1), 32–73.

<https://doi.org/10.1007/s11263-016-0981-7>

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with

deep convolutional neural networks. *Communications of the ACM*, 60(6), 84–90.

<https://doi.org/10.1145/3065386>

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning

applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.

<https://doi.org/10.1109/5.726791>

Lecun, Y., Chopra, S., & Hadsell, R. (2006). *A tutorial on energy-based learning*.

LeCun, Y., Haffner, P., Bottou, L., & Bengio, Y. (1999). Object Recognition with

Gradient-Based Learning. In D. A. Forsyth, J. L. Mundy, V. di Gesù, & R.

Cipolla (Eds.), *Shape, Contour and Grouping in Computer Vision* (pp. 319–

345). Springer. https://doi.org/10.1007/3-540-46805-6_19

Le-Khac, P. H., Healy, G., & Smeaton, A. F. (2020). Contrastive Representation

Learning: A Framework and Review. *IEEE Access*, 8, 193907–193934.

<https://doi.org/10.1109/ACCESS.2020.3031549>

Li, H., Fu, T., Dai, J., Li, H., Huang, G., & Zhu, X. (2022). AutoLoss-Zero: Searching

Loss Functions from Scratch for Generic Tasks. *2022 IEEE/CVF Conference on*

Computer Vision and Pattern Recognition (CVPR), 999–1008.

<https://doi.org/10.1109/CVPR52688.2022.00108>

Li, L. H., Yatskar, M., Yin, D., Hsieh, C.-J., & Chang, K.-W. (2019). VisualBERT: A Simple and Performant Baseline for Vision and Language. *ArXiv:1908.03557 [Cs]*. <http://arxiv.org/abs/1908.03557>

Li, R., Zhang, S., & He, X. (2022). SGTR: End-to-end Scene Graph Generation with Transformer. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 19464–19474.

<https://doi.org/10.1109/CVPR52688.2022.01888>

Li, Y., Ouyang, W., Wang, X., & Tang, X. (2017). ViP-CNN: Visual Phrase Guided Convolutional Neural Network. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7244–7253.

<https://doi.org/10.1109/CVPR.2017.766>

Li, Y., Ouyang, W., Zhou, B., Shi, J., Zhang, C., & Wang, X. (2018). Factorizable Net: An Efficient Subgraph-based Framework for Scene Graph Generation. *ArXiv:1806.11538 [Cs]*.

Liang, K., Guo, Y., Chang, H., & Chen, X. (2018). Visual Relationship Detection With Deep Structural Ranking. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Article 1. <https://doi.org/10.1609/aaai.v32i1.12274>

Lin, B., Zhu, Y., & Liang, X. (2022). Atom correlation based graph propagation for scene graph generation. *Pattern Recognition*, 122, 108300.

<https://doi.org/10.1016/j.patcog.2021.108300>

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature Pyramid Networks for Object Detection. *ArXiv:1612.03144 [Cs]*.

<http://arxiv.org/abs/1612.03144>

- Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., & Dollár, P. (2015). *Microsoft COCO: Common Objects in Context* (arXiv:1405.0312). arXiv. <http://arxiv.org/abs/1405.0312>
- Lin, Y., Liu, Z., Sun, M., Liu, Y., & Zhu, X. (2015). Learning Entity and Relation Embeddings for Knowledge Graph Completion. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1).
<https://doi.org/10.1609/aaai.v29i1.9491>
- Loshchilov, I., & Hutter, F. (2019). *Decoupled Weight Decay Regularization* (arXiv:1711.05101). arXiv. <http://arxiv.org/abs/1711.05101>
- Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2), 91–110.
<https://doi.org/10.1023/B:VISI.0000029664.99615.94>
- Lu, C., Krishna, R., Bernstein, M., & Fei-Fei, L. (2016). Visual Relationship Detection with Language Priors. *ArXiv:1608.00187 [Cs]*. <http://arxiv.org/abs/1608.00187>
- Lu, J., Batra, D., Parikh, D., & Lee, S. (2019). ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks. *Advances in Neural Information Processing Systems*, 32.
<https://proceedings.neurips.cc/paper/2019/hash/c74d97b01eae257e44aa9d5bade97baf-Abstract.html>
- Maheshwari, P., Chaudhry, R., & Vinay, V. (2021). Scene Graph Embeddings Using Relative Similarity Supervision. *ArXiv:2104.02381 [Cs]*.
<http://arxiv.org/abs/2104.02381>
- McCann, B., Bradbury, J., Xiong, C., & Socher, R. (2018). Learned in Translation: Contextualized Word Vectors. *ArXiv:1708.00107 [Cs]*.
<http://arxiv.org/abs/1708.00107>

- Mi, L., & Chen, Z. (2020). Hierarchical Graph Attention Network for Visual Relationship Detection. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 13883–13892.
<https://doi.org/10.1109/CVPR42600.2020.01390>
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *ArXiv:1301.3781 [Cs]*.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (n.d.). *Distributed Representations of Words and Phrases and their Compositionality*. 9.
- Miller, G. A. (1995). WordNet: A lexical database for English. *Communications of the ACM*, 38(11), 39–41. <https://doi.org/10.1145/219717.219748>
- Oord, A. van den, Li, Y., & Vinyals, O. (2019). Representation Learning with Contrastive Predictive Coding. *ArXiv:1807.03748 [Cs, Stat]*.
<http://arxiv.org/abs/1807.03748>
- Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: A Method for Automatic Evaluation of Machine Translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 311–318.
<https://doi.org/10.3115/1073083.1073135>
- Pengzhen, R., Xiao, Y., Chang, X., Huang, P.-Y., Chen, X., & Wang, X. (2021). A Comprehensive Survey of Neural Architecture Search: Challenges and Solutions. *ACM Computing Surveys*, 54, 1–34. <https://doi.org/10.1145/3447582>
- Peters, M. E., Ammar, W., Bhagavatula, C., & Power, R. (2017). Semi-supervised sequence tagging with bidirectional language models. *ArXiv:1705.00108 [Cs]*.
<http://arxiv.org/abs/1705.00108>
- Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *ArXiv:1802.05365 [Cs]*.
<http://arxiv.org/abs/1802.05365>

- Peyre, J., Laptev, I., Schmid, C., & Sivic, J. (2017). Weakly-Supervised Learning of Visual Relations. *2017 IEEE International Conference on Computer Vision (ICCV)*, 5189–5198. <https://doi.org/10.1109/ICCV.2017.554>
- Peyre, J., Sivic, J., Laptev, I., & Schmid, C. (2019). Detecting Unseen Visual Relations Using Analogies. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 1981–1990. <https://doi.org/10.1109/ICCV.2019.00207>
- Powers, D. (2015). *Evaluation Evaluation a Monte Carlo study*.
- Powers, D. M. W. (2020). *Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation* (arXiv:2010.16061). arXiv. <http://arxiv.org/abs/2010.16061>
- Quilty-Dunn, J. (2021). Polysemy and thought: Toward a generative theory of concepts. *Mind & Language*, 36(1), 158–185. <https://doi.org/10.1111/mila.12328>
- Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). Learning Transferable Visual Models From Natural Language Supervision. *ArXiv:2103.00020 [Cs]*. <http://arxiv.org/abs/2103.00020>
- Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (n.d.). *Robust Speech Recognition via Large-Scale Weak Supervision*. 28.
- Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (n.d.). *Improving Language Understanding by Generative Pre-Training*. 12.
- Raghu, M., Unterthiner, T., Kornblith, S., Zhang, C., & Dosovitskiy, A. (2022). *Do Vision Transformers See Like Convolutional Neural Networks?* (arXiv:2108.08810). arXiv. <http://arxiv.org/abs/2108.08810>
- Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Advances in Neural Information Processing Systems*, 28.

<https://papers.nips.cc/paper/2015/hash/14bfa6bb14875e45bba028a21ed38046-Abstract.html>

Rodrigues, E. J., Santos, P. E., Lopes, M., Bennett, B., & Oppenheimer, P. E. (2020). Standpoint Semantics for Polysemy in Spatial Prepositions. *Journal of Logic and Computation*, 30(2), 635–661.

Rosasco, L., Vito, E. D., Caponnetto, A., Piana, M., & Verri, A. (2004). Are Loss Functions All the Same? *Neural Computation*, 16(5), 1063–1076.
<https://doi.org/10.1162/089976604773135104>

Sadeghi, F., Divvala, S. K., & Farhadi, A. (2015). VisKE: Visual knowledge extraction and question answering by visual verification of relation phrases. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1456–1464.
<https://doi.org/10.1109/CVPR.2015.7298752>

Sap, M., LeBras, R., Allaway, E., Bhagavatula, C., Lourie, N., Rashkin, H., Roof, B., Smith, N. A., & Choi, Y. (2019). ATOMIC: An Atlas of Machine Commonsense for If-Then Reasoning. *ArXiv:1811.00146 [Cs]*.
<http://arxiv.org/abs/1811.00146>

Scarselli, F., Gori, M., Ah Chung Tsoi, Hagenbuchner, M., & Monfardini, G. (2009). The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1), 61–80. <https://doi.org/10.1109/TNN.2008.2005605>

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 815–823.
<https://doi.org/10.1109/CVPR.2015.7298682>

Schuster, S., Krishna, R., Chang, A., Fei-Fei, L., & Manning, C. D. (2015). Generating Semantically Precise Scene Graphs from Textual Descriptions for Improved

- Image Retrieval. *Proceedings of the Fourth Workshop on Vision and Language*, 70–80. <https://doi.org/10.18653/v1/W15-2812>
- Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (n.d.). *Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization*. 9.
- Shekhar, R., Pezzelle, S., Klimovich, Y., Herbelot, A., Nabi, M., Sangineto, E., & Bernardi, R. (2017). FOIL it! Find One mismatch between Image and Language caption. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 255–265. <https://doi.org/10.18653/v1/P17-1024>
- Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv:1409.1556 [Cs]*. <http://arxiv.org/abs/1409.1556>
- Song, Y., & Kingma, D. P. (2021). *How to Train Your Energy-Based Models* (arXiv:2101.03288). arXiv. <http://arxiv.org/abs/2101.03288>
- Speer, R., Chin, J., & Havasi, C. (2018). ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. *ArXiv:1612.03975 [Cs]*. <http://arxiv.org/abs/1612.03975>
- Su, W., Zhu, X., Cao, Y., Li, B., Lu, L., Wei, F., & Dai, J. (2020). VL-BERT: Pre-training of Generic Visual-Linguistic Representations. *ArXiv:1908.08530 [Cs]*. <http://arxiv.org/abs/1908.08530>
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). Going Deeper with Convolutions. *ArXiv:1409.4842 [Cs]*. <http://arxiv.org/abs/1409.4842>
- Szyc, K., Walkowiak, T., & Maciejewski, H. (2021). Checking Robustness of Representations Learned by Deep Neural Networks. In Y. Dong, N. Kourtellis, B. Hammer, & J. A. Lozano (Eds.), *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track* (Vol. 12979, pp. 399–

414). Springer International Publishing. https://doi.org/10.1007/978-3-030-86517-7_25

Tang, K., Niu, Y., Huang, J., Shi, J., & Zhang, H. (2020). Unbiased Scene Graph Generation from Biased Training. *ArXiv:2002.11949 [Cs]*.

<http://arxiv.org/abs/2002.11949>

Tang, K., Zhang, H., Wu, B., Luo, W., & Liu, W. (2018). Learning to Compose Dynamic Tree Structures for Visual Contexts. *ArXiv:1812.01880 [Cs]*.

<http://arxiv.org/abs/1812.01880>

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., & Jegou, H. (2021). Training data-efficient image transformers & distillation through attention. *Proceedings of the 38th International Conference on Machine Learning*, 10347–10357. <https://proceedings.mlr.press/v139/touvron21a.html>

Tran, T., Neau, M., Santos, P., & Powers, D. (2022). Contrastive Visual and Language Learning for Visual Relationship Detection. *Proceedings of the The 20th Annual Workshop of the Australasian Language Technology Association*, 170–177.

<https://aclanthology.org/2022.alt-1.23>

Tran, T., Santos, P. E., & Powers, D. (2022). Contrastive Visual and Language Translational Embeddings for Visual Relationship Detection: AAI 2022 Spring Symposium on Machine Learning and Knowledge Engineering for Hybrid Intelligence, AAI-MAKE 2022. *CEUR Workshop Proceedings*, 3121.

<http://www.scopus.com/inward/record.url?scp=85128701548&partnerID=8YFLogxK>

van Ravenzwaaij, D., Cassey, P., & Brown, S. D. (2018). A simple introduction to Markov Chain Monte–Carlo sampling. *Psychonomic Bulletin & Review*, 25(1), 143–154. <https://doi.org/10.3758/s13423-016-1015-8>

- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (n.d.). *Attention is All you Need*. 11.
- Voita, E., Talbot, D., Moiseev, F., Sennrich, R., & Titov, I. (2019). Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 5797–5808. <https://doi.org/10.18653/v1/P19-1580>
- Wan, H., Luo, Y., Peng, B., & Zheng, W.-S. (2018). Representation Learning for Scene Graph Completion via Jointly Structural and Visual Embedding. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 949–956. <https://doi.org/10.24963/ijcai.2018/132>
- Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors* (arXiv:2207.02696). arXiv. <https://doi.org/10.48550/arXiv.2207.02696>
- Wang, T., & Isola, P. (2020). Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere. *Proceedings of the 37th International Conference on Machine Learning*, 9929–9939. <https://proceedings.mlr.press/v119/wang20k.html>
- Wei, X., Wang, H., Scotney, B., & Wan, H. (2020). Minimum margin loss for deep face recognition. *Pattern Recognition*, 97, 107012. <https://doi.org/10.1016/j.patcog.2019.107012>
- Wortsman, M., Ilharco, G., Kim, J. W., Li, M., Kornblith, S., Roelofs, R., Gontijo-Lopes, R., Hajishirzi, H., Farhadi, A., Namkoong, H., & Schmidt, L. (2022). *Robust fine-tuning of zero-shot models* (arXiv:2109.01903). arXiv. <http://arxiv.org/abs/2109.01903>

- Wu, M., Zhuang, C., Mosse, M., Yamins, D., & Goodman, N. (2020). *On Mutual Information in Contrastive Learning for Visual Representations* (arXiv:2005.13149). arXiv. <http://arxiv.org/abs/2005.13149>
- Xu, B., Wang, N., Chen, T., & Li, M. (2015). *Empirical Evaluation of Rectified Activations in Convolutional Network* (arXiv:1505.00853). arXiv. <http://arxiv.org/abs/1505.00853>
- Xu, D., Zhu, Y., Choy, C. B., & Fei-Fei, L. (n.d.). *Scene Graph Generation by Iterative Message Passing*.
- Xu, D., Zhu, Y., Choy, C. B., & Fei-Fei, L. (2017a). Scene Graph Generation by Iterative Message Passing. *ArXiv:1701.02426 [Cs]*. <http://arxiv.org/abs/1701.02426>
- Xu, D., Zhu, Y., Choy, C. B., & Fei-Fei, L. (2017b). Scene Graph Generation by Iterative Message Passing. *ArXiv:1701.02426 [Cs]*.
- Yang, J., Lu, J., Lee, S., Batra, D., & Parikh, D. (2018). Graph R-CNN for Scene Graph Generation. In V. Ferrari, M. Hebert, C. Sminchisescu, & Y. Weiss (Eds.), *Computer Vision – ECCV 2018* (Vol. 11205, pp. 690–706). Springer International Publishing. https://doi.org/10.1007/978-3-030-01246-5_41
- Yin, G., Sheng, L., Liu, B., Yu, N., Wang, X., Shao, J., & Loy, C. C. (2018). Zoom-Net: Mining Deep Feature Interactions for Visual Relationship Recognition. In V. Ferrari, M. Hebert, C. Sminchisescu, & Y. Weiss (Eds.), *Computer Vision – ECCV 2018* (Vol. 11207, pp. 330–347). Springer International Publishing. https://doi.org/10.1007/978-3-030-01219-9_20
- Youden, W. J. (1950). Index for rating diagnostic tests. *Cancer*, 3(1), 32–35. [https://doi.org/10.1002/1097-0142\(1950\)3:1<32::AID-CNCR2820030106>3.0.CO;2-3](https://doi.org/10.1002/1097-0142(1950)3:1<32::AID-CNCR2820030106>3.0.CO;2-3)

- Yu, B., Liu, T., Gong, M., Ding, C., & Tao, D. (2018). Correcting the Triplet Selection Bias for Triplet Loss. In V. Ferrari, M. Hebert, C. Sminchisescu, & Y. Weiss (Eds.), *Computer Vision – ECCV 2018* (pp. 71–86). Springer International Publishing. https://doi.org/10.1007/978-3-030-01231-1_5
- Yu, B., & Tao, D. (2019). Deep Metric Learning With Tuplet Margin Loss. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 6489–6498. <https://doi.org/10.1109/ICCV.2019.00659>
- Yu, R., Li, A., Morariu, V. I., & Davis, L. S. (2017). Visual Relationship Detection with Internal and External Linguistic Knowledge Distillation. *2017 IEEE International Conference on Computer Vision (ICCV)*, 1068–1076. <https://doi.org/10.1109/ICCV.2017.121>
- Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, Ross Girshick & Wan-Yen Lo and Ross Girshick. (2019). *Detectron2* [Python]. Meta Research. <https://github.com/facebookresearch/detectron2> (Original work published 2019)
- Zellers, R., Yatskar, M., Thomson, S., & Choi, Y. (2018). Neural Motifs: Scene Graph Parsing with Global Context. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5831–5840. <https://doi.org/10.1109/CVPR.2018.00611>
- Zhang, H., Kyaw, Z., Chang, S.-F., & Chua, T.-S. (2017a). Visual Translation Embedding Network for Visual Relation Detection. *ArXiv:1702.08319 [Cs]*. <http://arxiv.org/abs/1702.08319>
- Zhang, H., Kyaw, Z., Chang, S.-F., & Chua, T.-S. (2017b). Visual Translation Embedding Network for Visual Relation Detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3107–3115. <https://doi.org/10.1109/CVPR.2017.331>

- Zhang, J., Elhoseiny, M., Cohen, S., Chang, W., & Elgammal, A. (2017). Relationship Proposal Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5226–5234. <https://doi.org/10.1109/CVPR.2017.555>
- Zhang, J., Shih, K. J., Elgammal, A., Tao, A., & Catanzaro, B. (2019a). Graphical Contrastive Losses for Scene Graph Parsing. *ArXiv:1903.02728 [Cs]*. <http://arxiv.org/abs/1903.02728>
- Zhang, J., Shih, K. J., Elgammal, A., Tao, A., & Catanzaro, B. (2019b). Graphical Contrastive Losses for Scene Graph Parsing. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 11527–11535. <https://doi.org/10.1109/CVPR.2019.01180>
- Zhang, J., Shih, K. J., Elgammal, A., Tao, A., & Catanzaro, B. (2019c). Graphical Contrastive Losses for Scene Graph Parsing. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 11527–11535. <https://doi.org/10.1109/CVPR.2019.01180>
- Zhu, S.-C., & Mumford, D. (2006). A Stochastic Grammar of Images. *Foundations and Trends® in Computer Graphics and Vision*, 2(4), 259–362. <https://doi.org/10.1561/06000000018>
- Zhu, X., Su, W., Lu, L., Li, B., Wang, X., & Dai, J. (2021). *Deformable DETR: Deformable Transformers for End-to-End Object Detection* (arXiv:2010.04159). arXiv. <http://arxiv.org/abs/2010.04159>
- Zhuang, B., Liu, L., Shen, C., & Reid, I. (2017). Towards Context-Aware Interaction Recognition for Visual Relationship Detection. *2017 IEEE International Conference on Computer Vision (ICCV)*, 589–598. <https://doi.org/10.1109/ICCV.2017.71>

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2020). *A Comprehensive Survey on Transfer Learning* (arXiv:1911.02685). arXiv.

<http://arxiv.org/abs/1911.02685>

Appendices

Appendix A: Biased Object Detector for VRD

```
In [179]: object_area_ap = beatify_detector2_results(eval_results)

Evaluation results for bbox
|         AP  -> 13.69 ||         AP50 -> 21.38 |
|         AP75 -> 15.02 ||         APs  ->  1.43 |
|         APm  ->  7.13 ||         AP1  -> 16.15 |
|-----|-----|
Evaluation results by object category
|         AP-person -> 34.58( 76559.36 area ) ||         AP-sky -> 34.00( 240566.56 area ) |
|         AP-building -> 12.51( 143178.06 area ) ||         AP-truck -> 26.60( 115784.02 area ) |
|         AP-bus -> 58.39( 157357.81 area ) ||         AP-table -> 26.36( 171163.44 area ) |
|         AP-shirt -> 14.84( 35092.40 area ) ||         AP-chair ->  6.35( 58440.02 area ) |
|         AP-car -> 23.45( 40891.77 area ) ||         AP-train -> 43.85( 165328.08 area ) |
|         AP-glasses -> 10.19( 16188.93 area ) ||         AP-tree ->  5.20( 78067.21 area ) |
|         AP-boat -> 13.36( 131939.91 area ) ||         AP-hat -> 13.68( 6856.33 area ) |
|         AP-trees ->  5.41( 168642.62 area ) ||         AP-grass -> 18.18( 188001.73 area ) |
|         AP-pants -> 11.50( 27497.62 area ) ||         AP-road ->  1.34( 219012.50 area ) |
|         AP-motorcycle -> 26.59( 137659.02 area ) ||         AP-jacket ->  3.14( 44384.67 area ) |
|         AP-monitor -> 39.43( 58283.25 area ) ||         AP-wheel ->  9.22( 13802.85 area ) |
|         AP-umbrella -> 17.14( 58044.14 area ) ||         AP-plate -> 10.30( 43204.28 area ) |
|         AP-bike -> 27.16( 77017.67 area ) ||         AP-clock -> 18.36( 20313.14 area ) |
|         AP-bag ->  4.14( 18558.25 area ) ||         AP-shoe ->  0.00( 5325.92 area ) |
|         AP-laptop -> 44.53( 69151.68 area ) ||         AP-desk -> 16.58( 187479.03 area ) |
|         AP-cabinet ->  1.11( 106261.52 area ) ||         AP-counter -> 11.78( 113178.15 area ) |
|         AP-bench -> 16.40( 101857.77 area ) ||         AP-shoes ->  4.75( 10061.13 area ) |
|         AP-tower -> 20.10( 91290.62 area ) ||         AP-bottle -> 13.40( 15821.26 area ) |
|         AP-helmet ->  6.43( 5840.40 area ) ||         AP-stove -> 11.73( 99298.34 area ) |
|         AP-lamp ->  6.91( 24420.36 area ) ||         AP-coat ->  2.38( 26689.43 area ) |
|         AP-bed -> 37.67( 283294.00 area ) ||         AP-dog -> 26.26( 91087.78 area ) |
|         AP-mountain -> 10.72( 156701.84 area ) ||         AP-horse -> 48.87( 124443.15 area ) |
|         AP-plane -> 14.65( 218616.00 area ) ||         AP-roof ->  5.26( 71968.27 area ) |
|         AP-skateboard -> 33.26( 14634.12 area ) ||         AP-traffic light ->  1.70( 21492.54 area ) |
|         AP-bush ->  4.75( 61116.31 area ) ||         AP-phone ->  9.55( 7087.81 area ) |
|         AP-airplane -> 23.27( 207302.50 area ) ||         AP-sofa -> 22.73( 179195.12 area ) |
|         AP-cup ->  2.68( 8058.37 area ) ||         AP-sink ->  7.93( 23896.81 area ) |
|         AP-shelf ->  0.45( 40503.39 area ) ||         AP-box ->  0.45( 22467.32 area ) |
|         AP-van ->  4.25( 42637.69 area ) ||         AP-hand ->  0.75( 11124.61 area ) |
|         AP-shorts ->  9.45( 17229.33 area ) ||         AP-post ->  0.00( 19911.62 area ) |
|         AP-jeans ->  3.37( 21702.57 area ) ||         AP-cat -> 39.72( 35152.50 area ) |
|         AP-sunglasses ->  5.76( 2637.89 area ) ||         AP-bowl ->  4.27( 33583.89 area ) |
|         AP-computer ->  0.00( 39283.40 area ) ||         AP-pillow ->  4.37( 37444.18 area ) |
|         AP-pizza -> 16.27( 52745.75 area ) ||         AP-basket ->  1.39( 38354.06 area ) |
|         AP-elephant -> 41.61( 164398.64 area ) ||         AP-kite -> 17.29( 37164.50 area ) |
|         AP-sand -> 10.25( 264610.78 area ) ||         AP-keyboard -> 36.28( 25222.50 area ) |
|         AP-plant ->  3.61( 24744.39 area ) ||         AP-can ->  7.33( 18185.31 area ) |
|         AP-vase ->  1.58( 15479.43 area ) ||         AP-refrigerator -> 16.17( 105438.10 area ) |
|         AP-cart ->  0.00( 103229.00 area ) ||         AP-skis -> 10.28( 38120.55 area ) |
|         AP-pot ->  1.90( 10412.88 area ) ||         AP-surfboard ->  2.59( 22901.00 area ) |
|         AP-paper ->  1.11( 28240.95 area ) ||         AP-mouse -> 28.40( 4136.33 area ) |
|         AP-trash can ->  3.04( 11847.54 area ) ||         AP-cone ->  2.38( 33400.53 area ) |
|         AP-camera ->  0.12( 14791.06 area ) ||         AP-ball ->  0.00( 2925.22 area ) |
|         AP-bear -> 20.03( 86962.55 area ) ||         AP-giraffe -> 44.46( 100031.80 area ) |
|         AP-tie -> 25.69( 8472.00 area ) ||         AP-luggage ->  4.69( 133869.73 area ) |
|         AP-faucet ->  0.00( 11172.43 area ) ||         AP-hydrant -> 44.27( 41876.83 area ) |
|         AP-snowboard -> 12.44( 55021.80 area ) ||         AP-oven ->  0.00( 163212.27 area ) |
|         AP-engine ->  4.51( 31553.91 area ) ||         AP-watch ->  1.59( 2093.95 area ) |
|         AP-face ->  0.85( 6665.91 area ) ||         AP-street -> 10.28( 236292.02 area ) |
|         AP-ramp ->  1.43( 111472.66 area ) ||         AP-suitcase ->  3.43( 43487.75 area ) |

Total Objects Detected: 5521
Total Labeled Objects: 6693
```

Figure 0.1: A screenshot output of the Average Precision of the Object detection module (on the test set).

From Figure 7.1, we can observe that object detection module suffers from bias and the majority of classes have a low average precision. This effect is due to the sparse and unbalanced nature of the VRD dataset, where many object classes occur only a dozen of times whereas a small number of classes occur up to thousands of times in the training data. We further explore the impact of object's size in term of area = $pixel^2$ on the performance of the object detector in Figure 7.2 below.

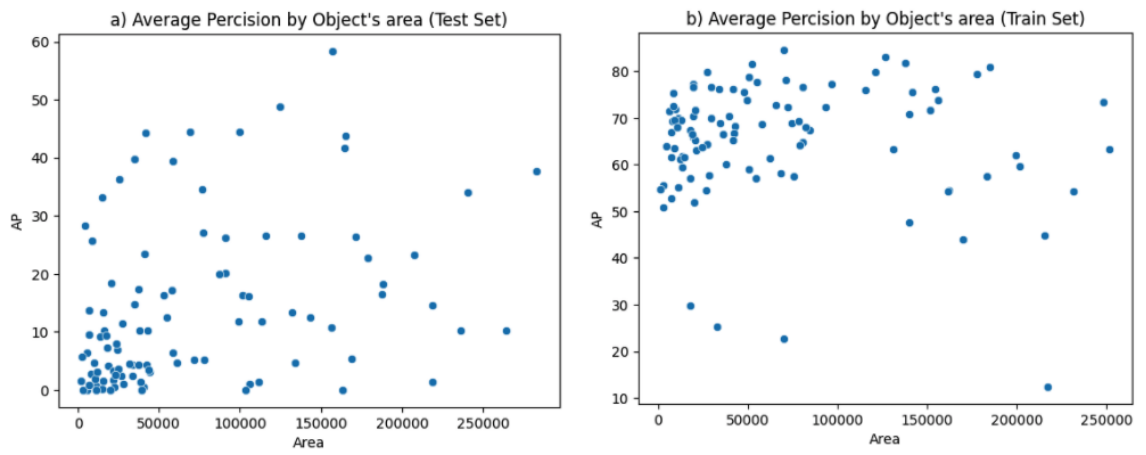
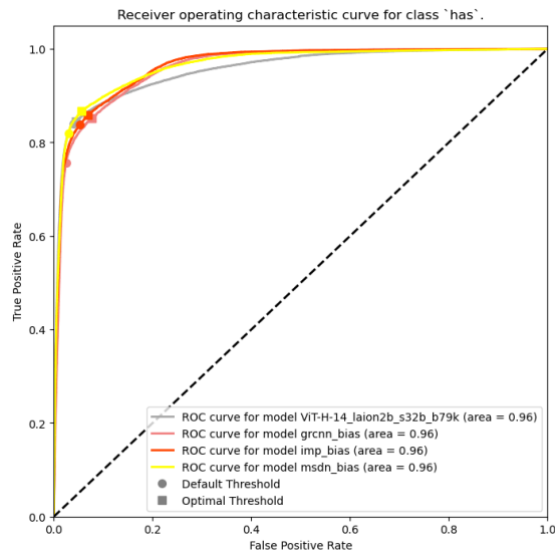


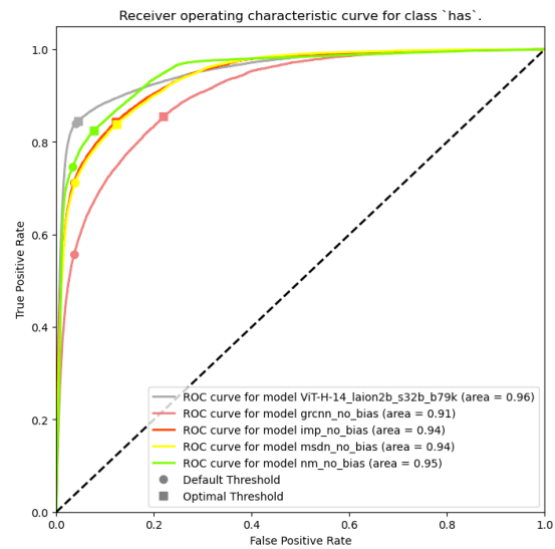
Figure 0.2: A scatterplot of object's area versus object detector performance in average precision.

On surface level observations of Figure 7.1, objects' areas do not impact the object detector performance beyond a certain scale. However, we can also observe a cluster of small objects around the origin that have low performance on the Test Set. Since the detector does not have the same issue on the Training Set, we can conclude that the detector has overfit the training data for the small objects' samples which are rarer.

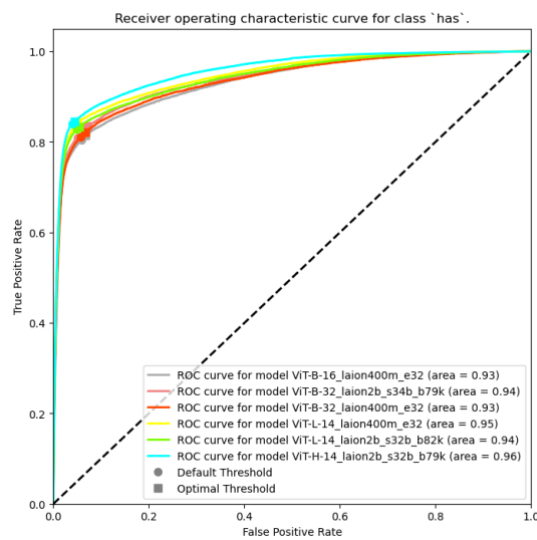
Appendix B: ROC Curves for top sixteen predicate classes



(a)

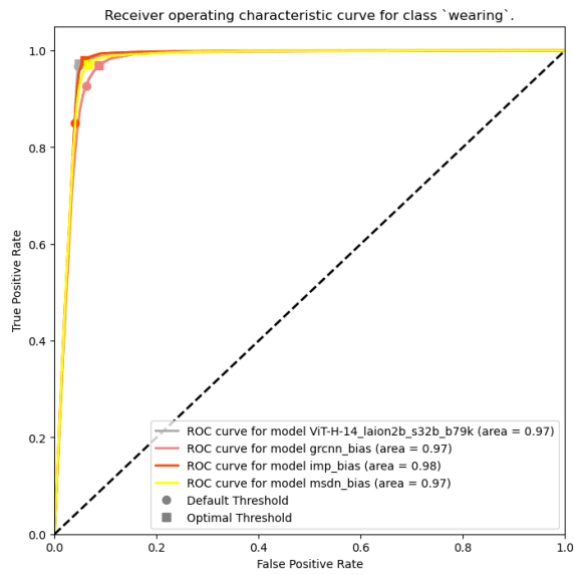


(b)

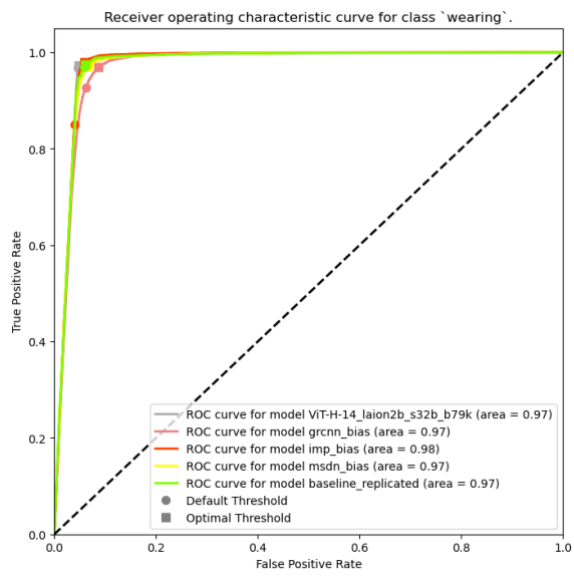


(c)

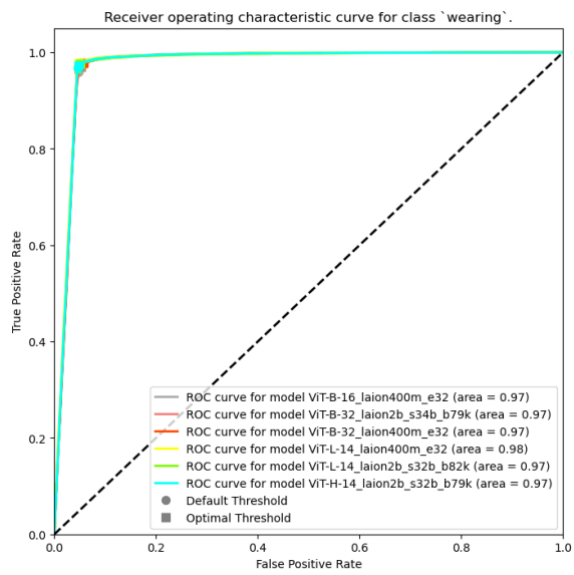
Figure 7.3: Receiver operating characteristic curves for the predicate class 'has'.



(a)

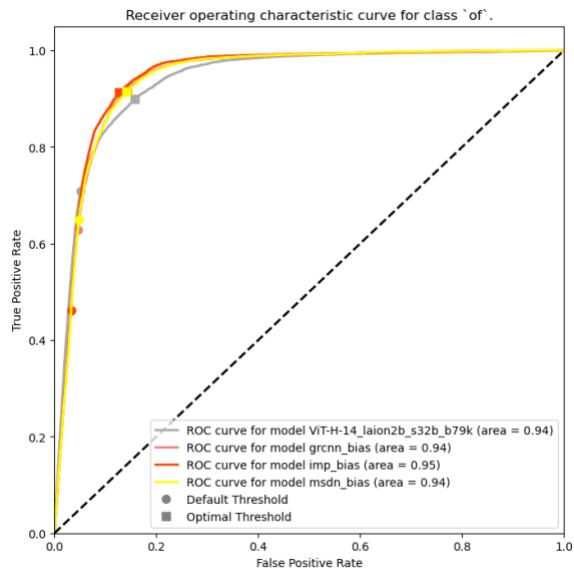


(b)

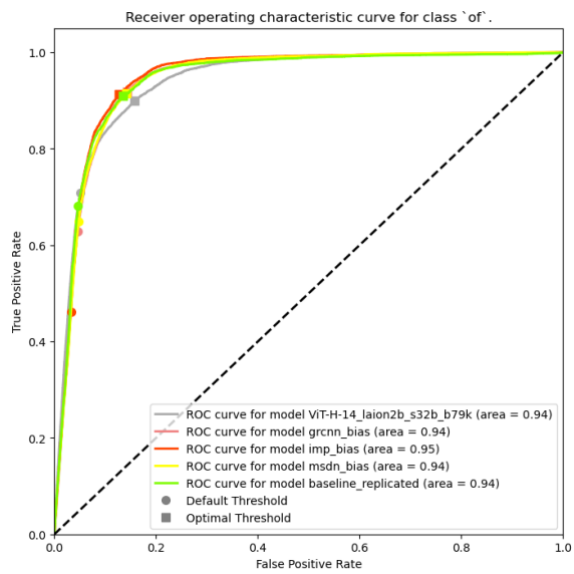


(c)

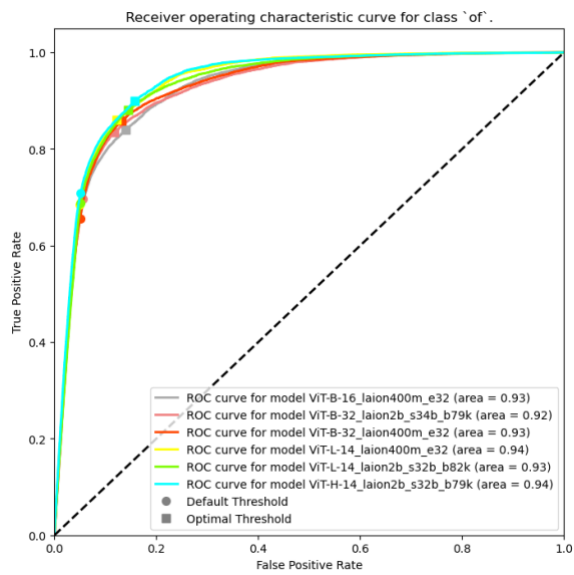
Figure 7.4: Receiver operating characteristic curves for the predicate class 'wearing'.



(a)

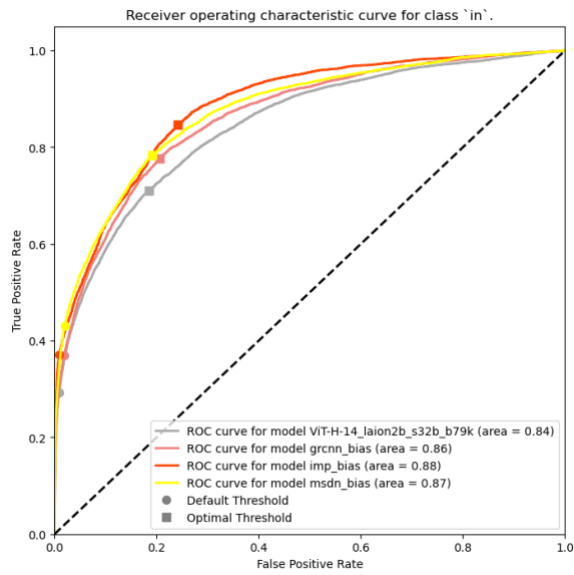


(b)

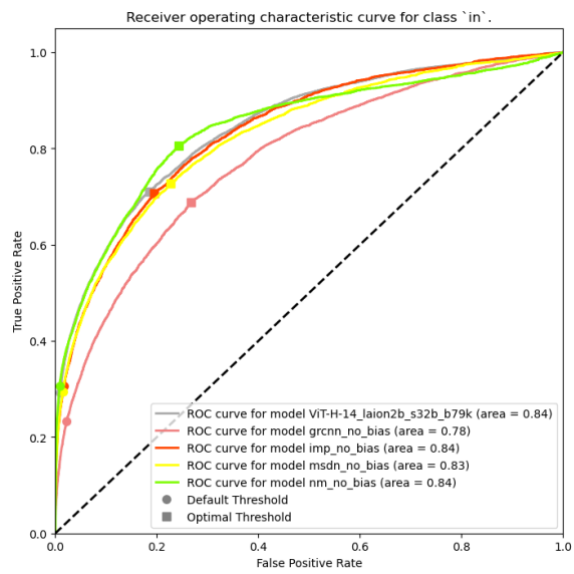


(c)

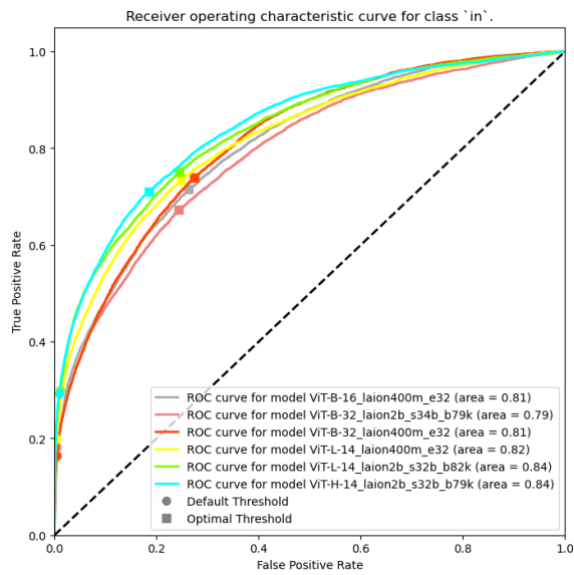
Figure 7.5: Receiver operating characteristic curves for the predicate class 'of'.



(a)

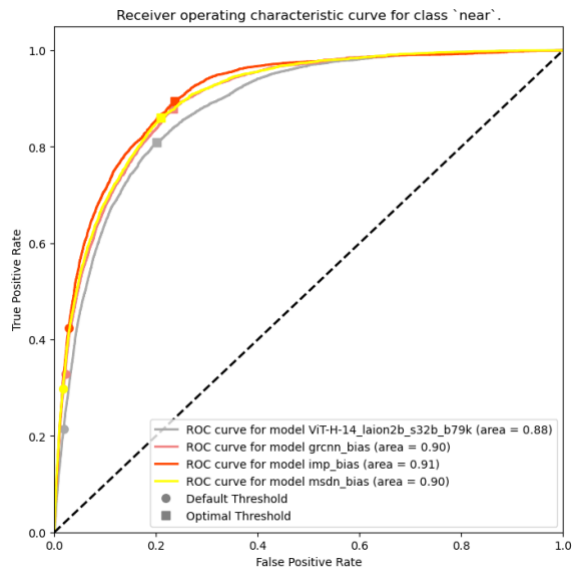


(b)

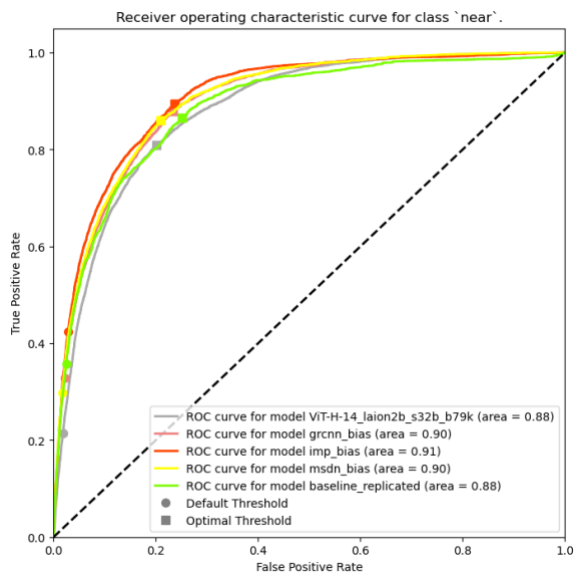


(c)

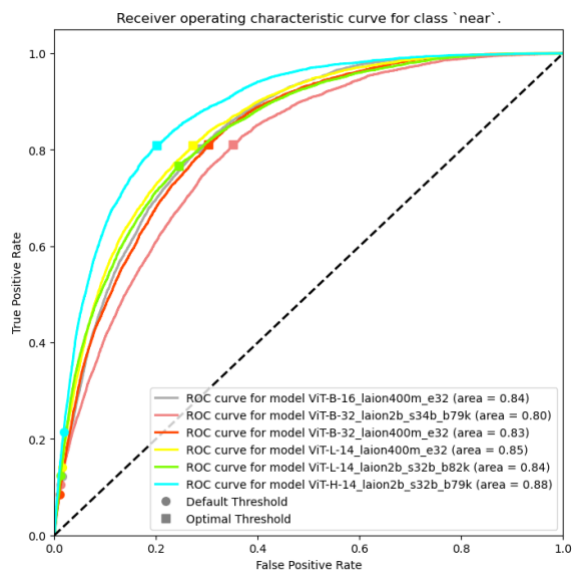
Figure 7.6: Receiver operating characteristic curves for the predicate class 'in'.



(a)

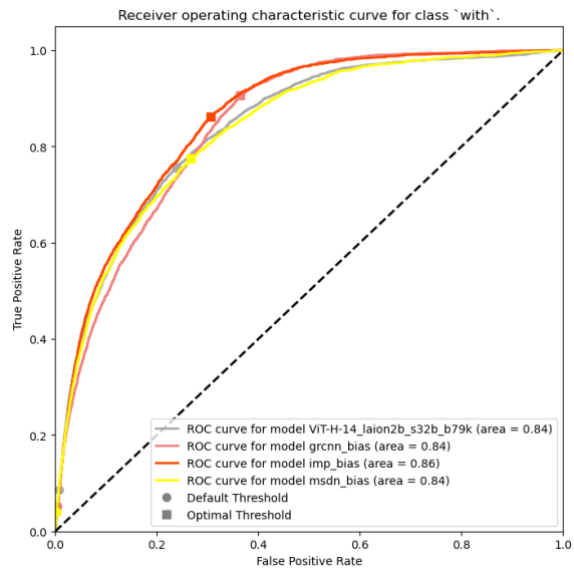


(b)

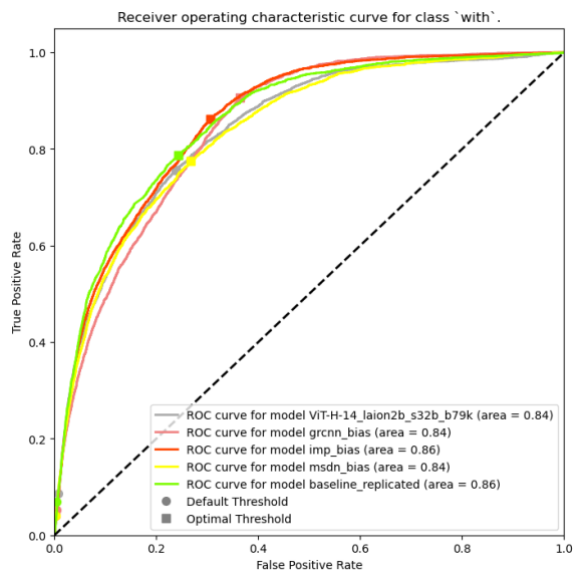


(c)

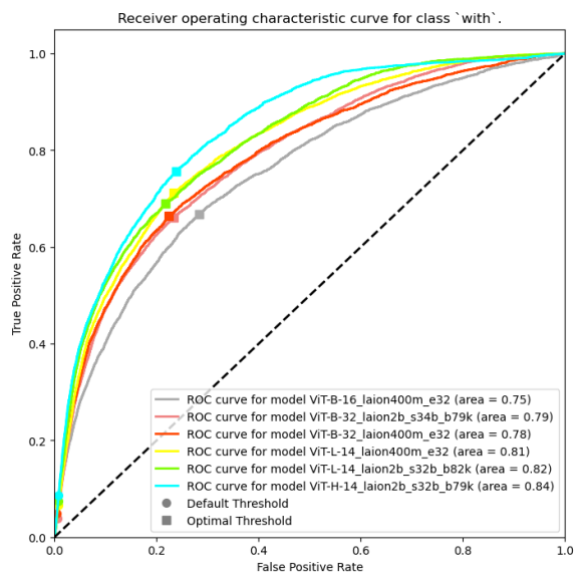
Figure 7.7: Receiver operating characteristic curves for the predicate class 'near'.



(a)

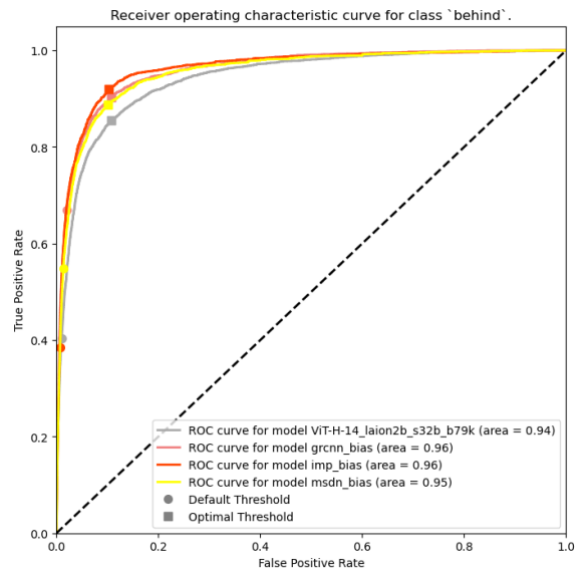


(b)

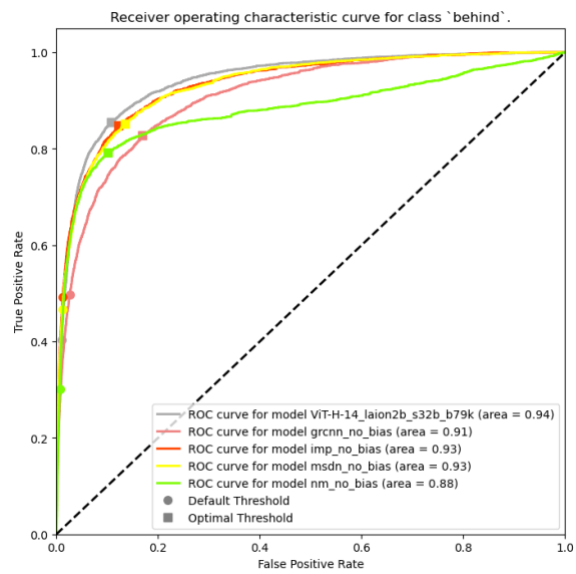


(c)

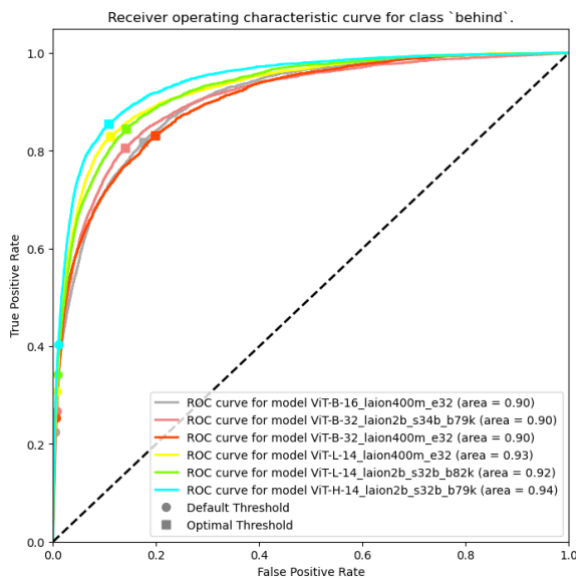
Figure 7.8: Receiver operating characteristic curves for the predicate class 'with'.



(a)

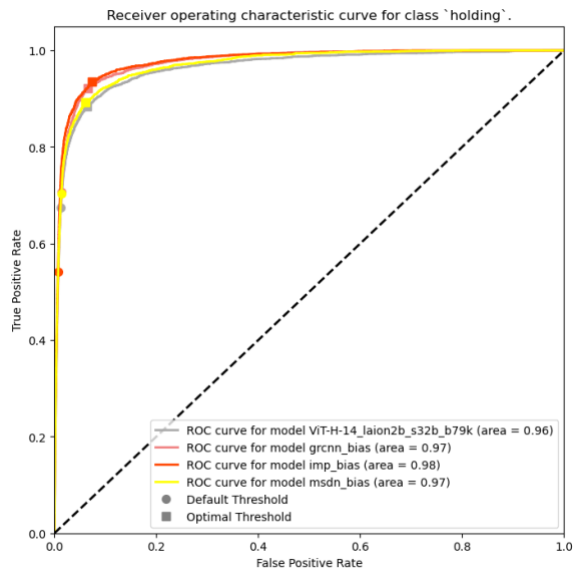


(b)

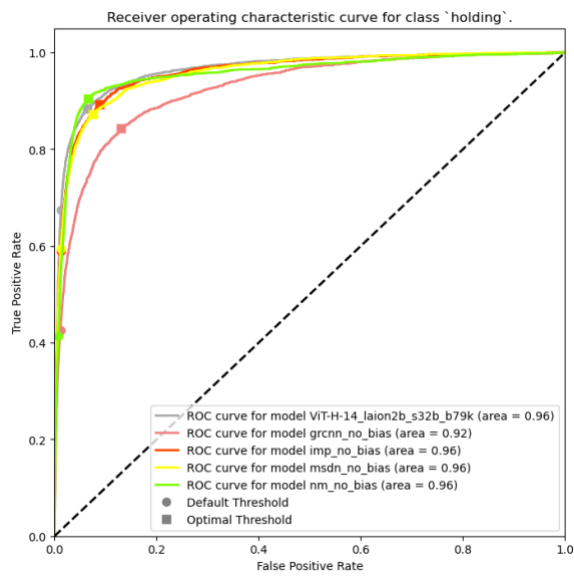


(c)

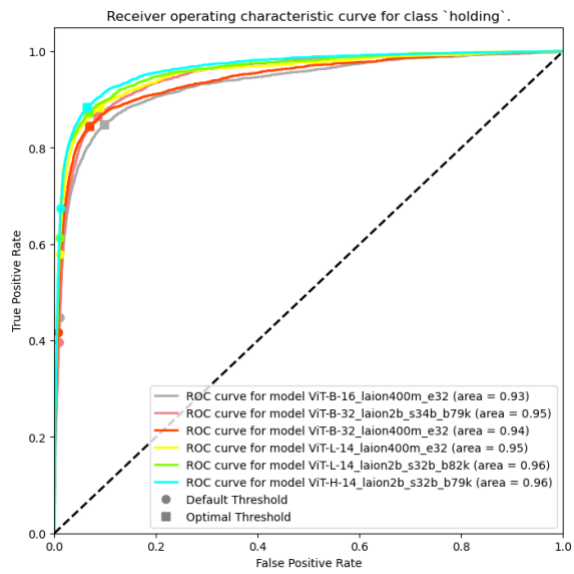
Figure 7.9: Receiver operating characteristic curves for the predicate class 'behind'.



(a)

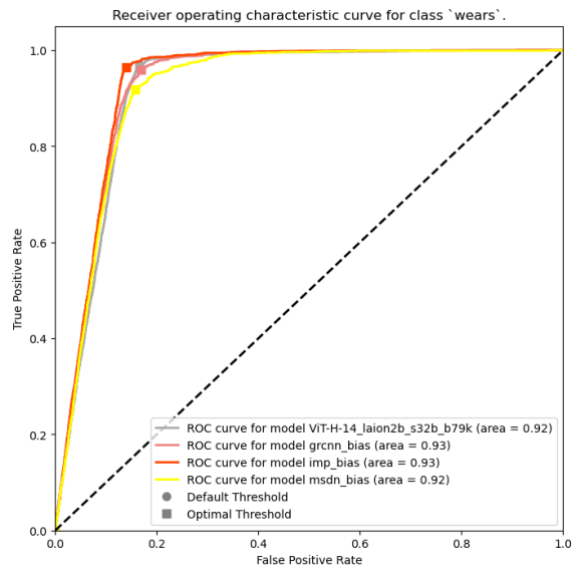


(b)

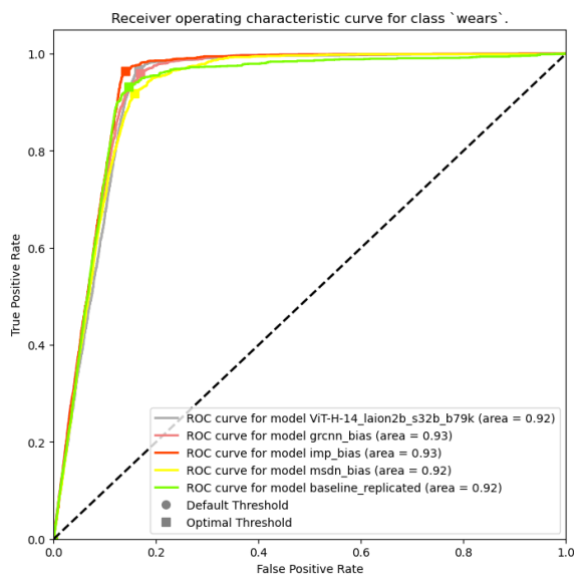


(c)

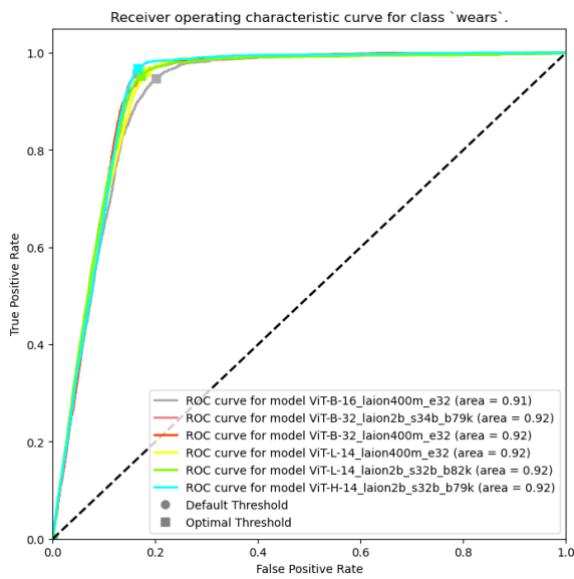
Figure 7.10: Receiver operating characteristic curves for the predicate class 'holding'.



(a)

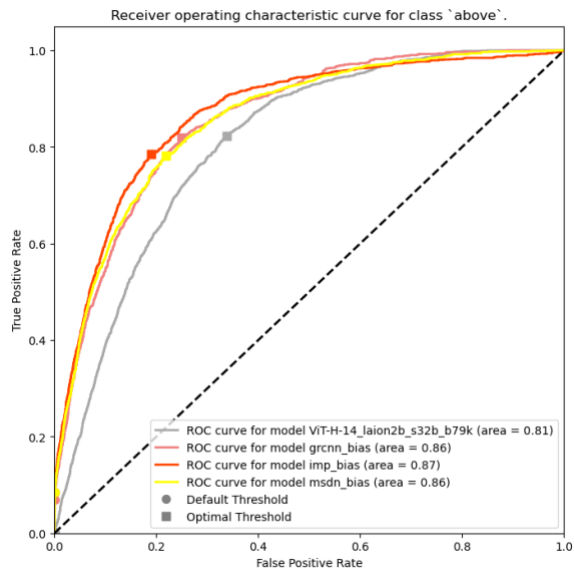


(b)

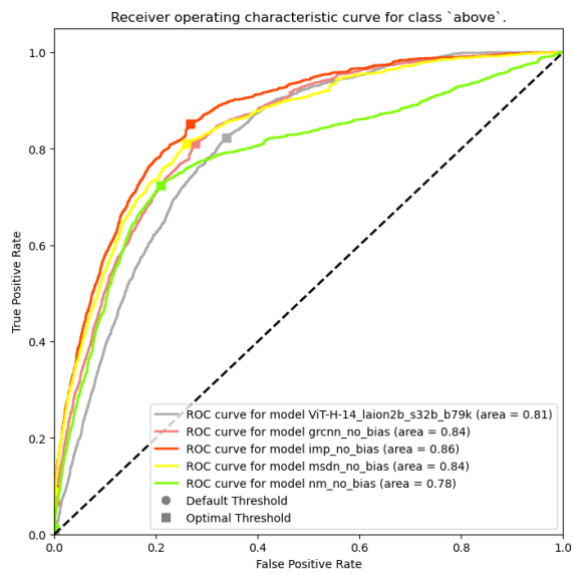


(c)

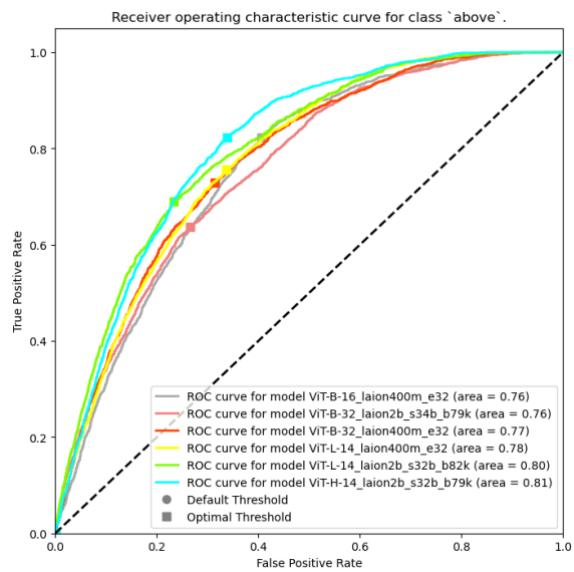
Figure 7.11: Receiver operating characteristic curves for the predicate class `wears`.



(a)

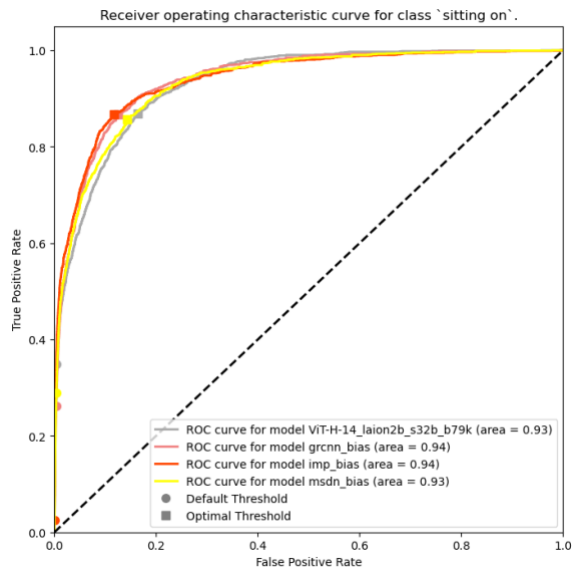


(b)

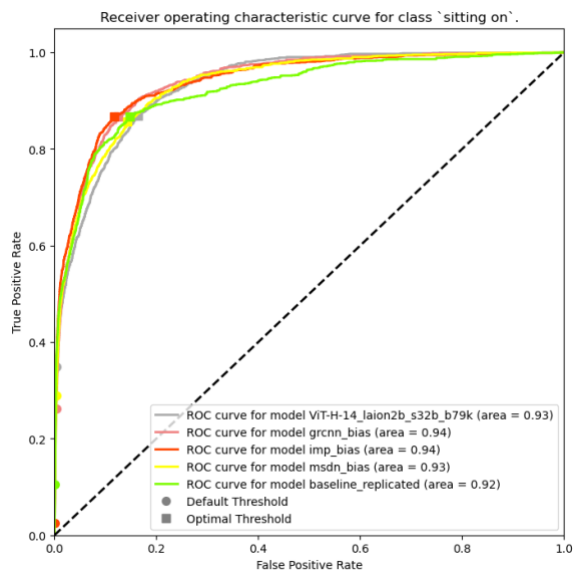


(c)

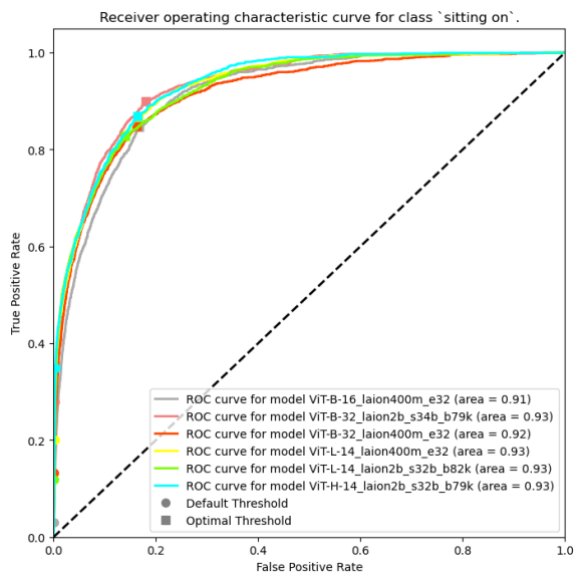
Figure 7.12: Receiver operating characteristic curves for the predicate class `above`.



(a)

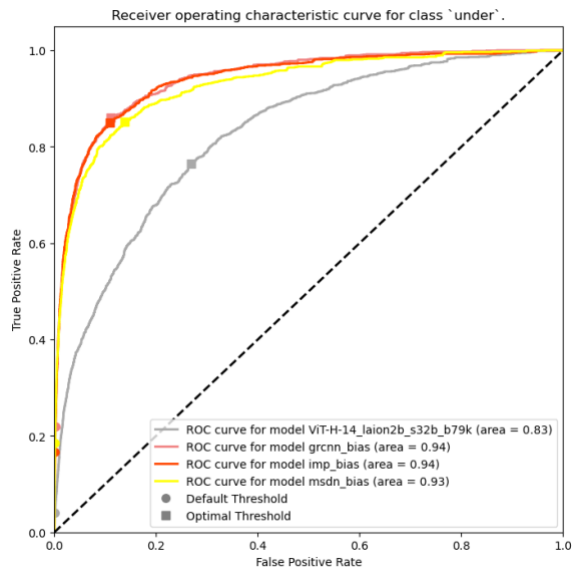


(b)

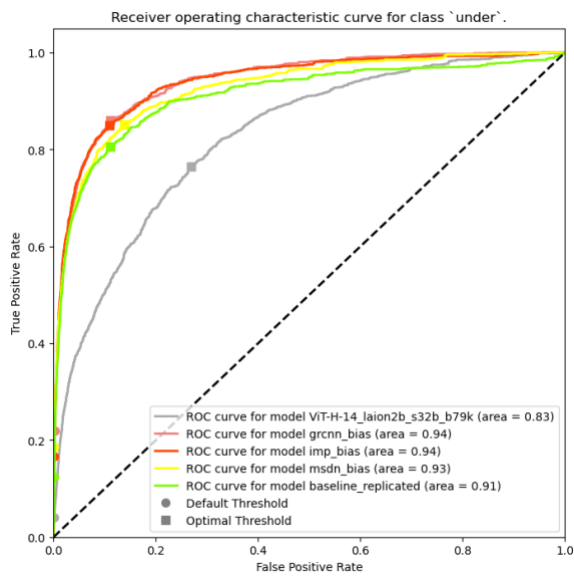


(c)

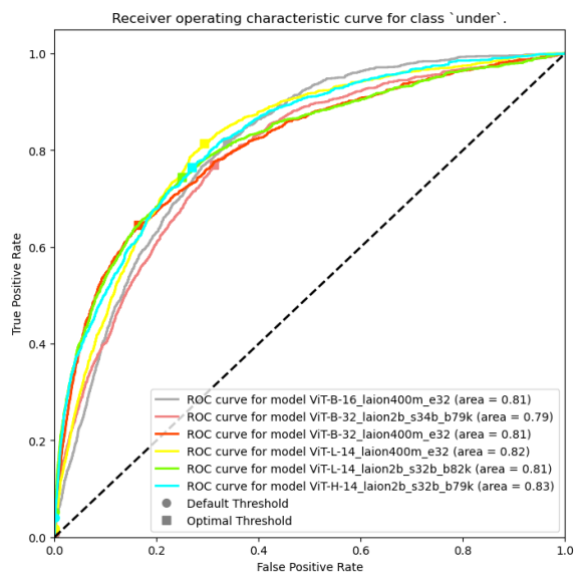
Figure 7.13: Receiver operating characteristic curves for the predicate class 'sitting on'.



(a)

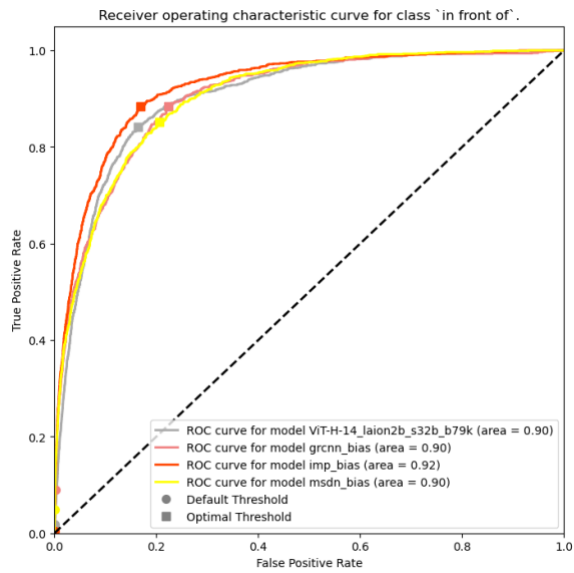


(b)

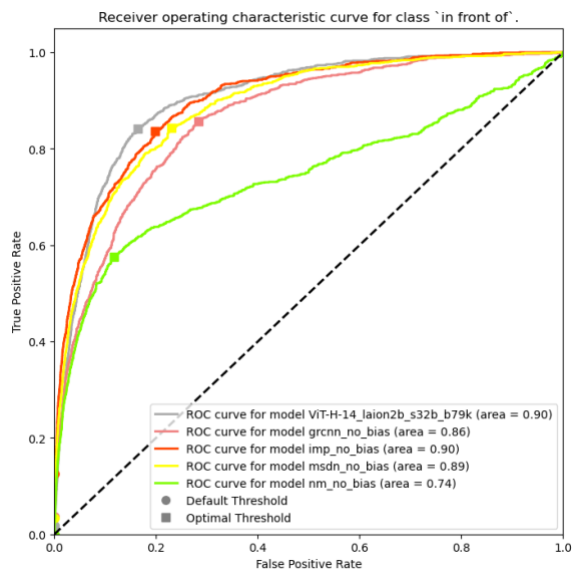


(c)

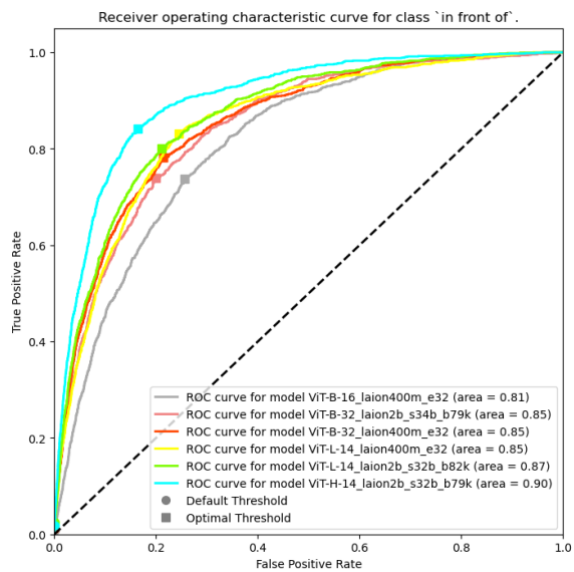
Figure 7.14: Receiver operating characteristic curves for the predicate class 'under'.



(a)

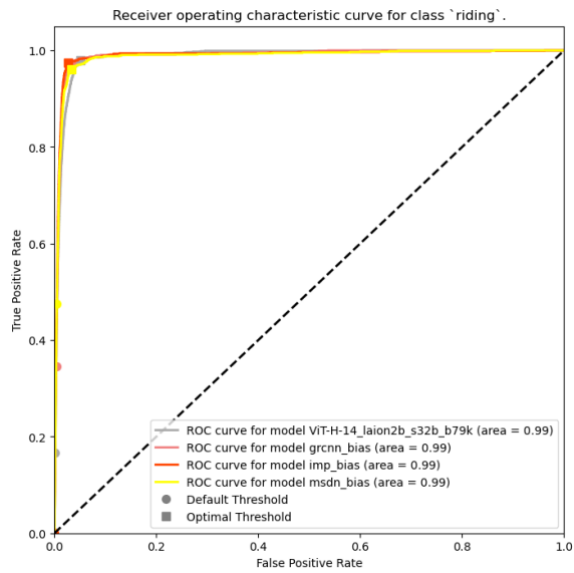


(b)

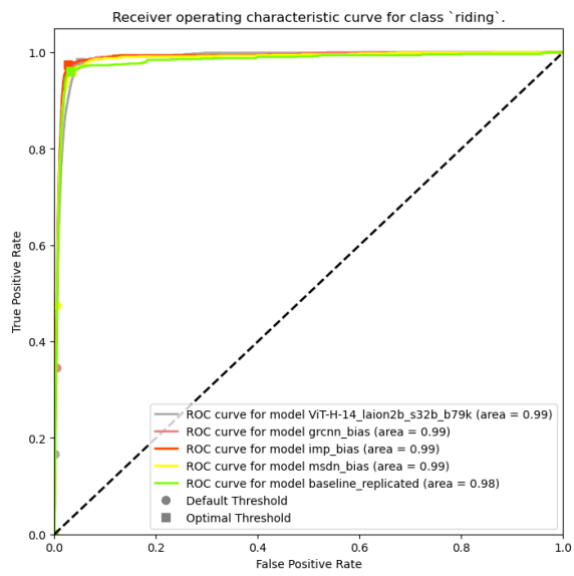


(c)

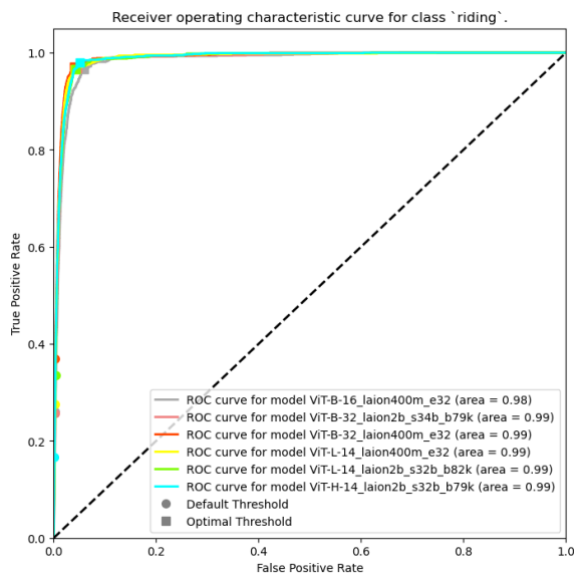
Figure 7.15: Receiver operating characteristic curves for the predicate class 'in front of'.



(a)

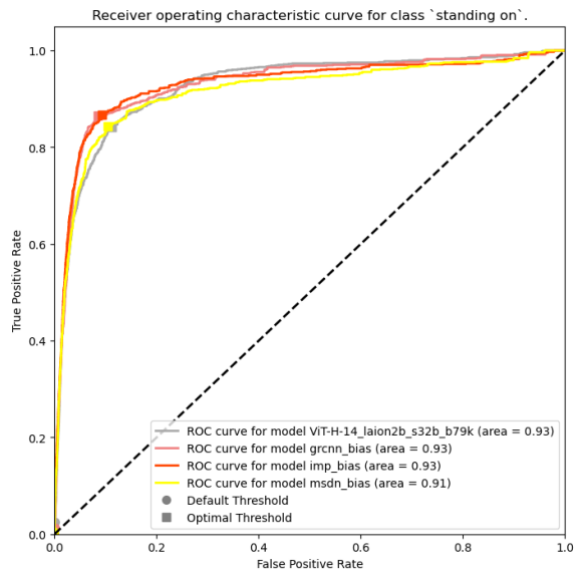


(b)

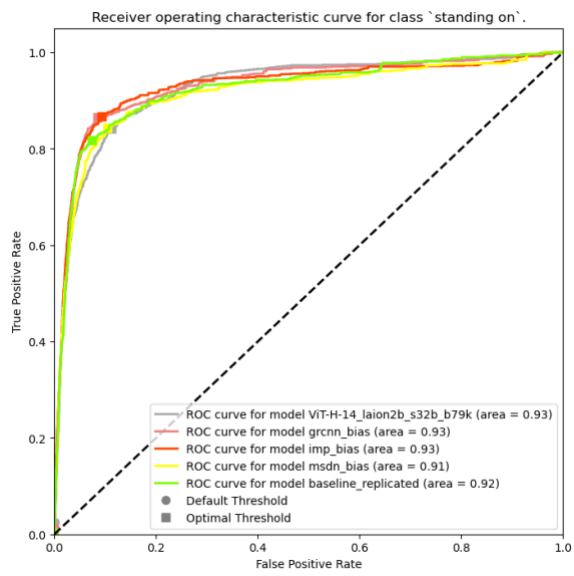


(c)

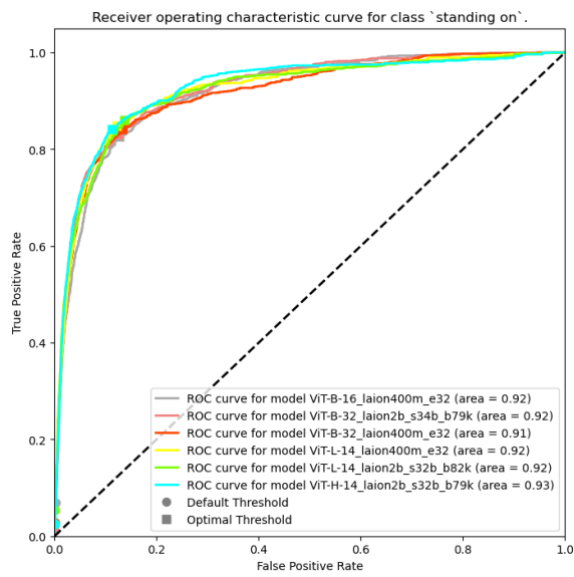
Figure 7.16: Receiver operating characteristic curves for the predicate class 'riding'.



(a)



(b)



(c)

Figure 7.17: Receiver operating characteristic curves for the predicate class 'standing on'.

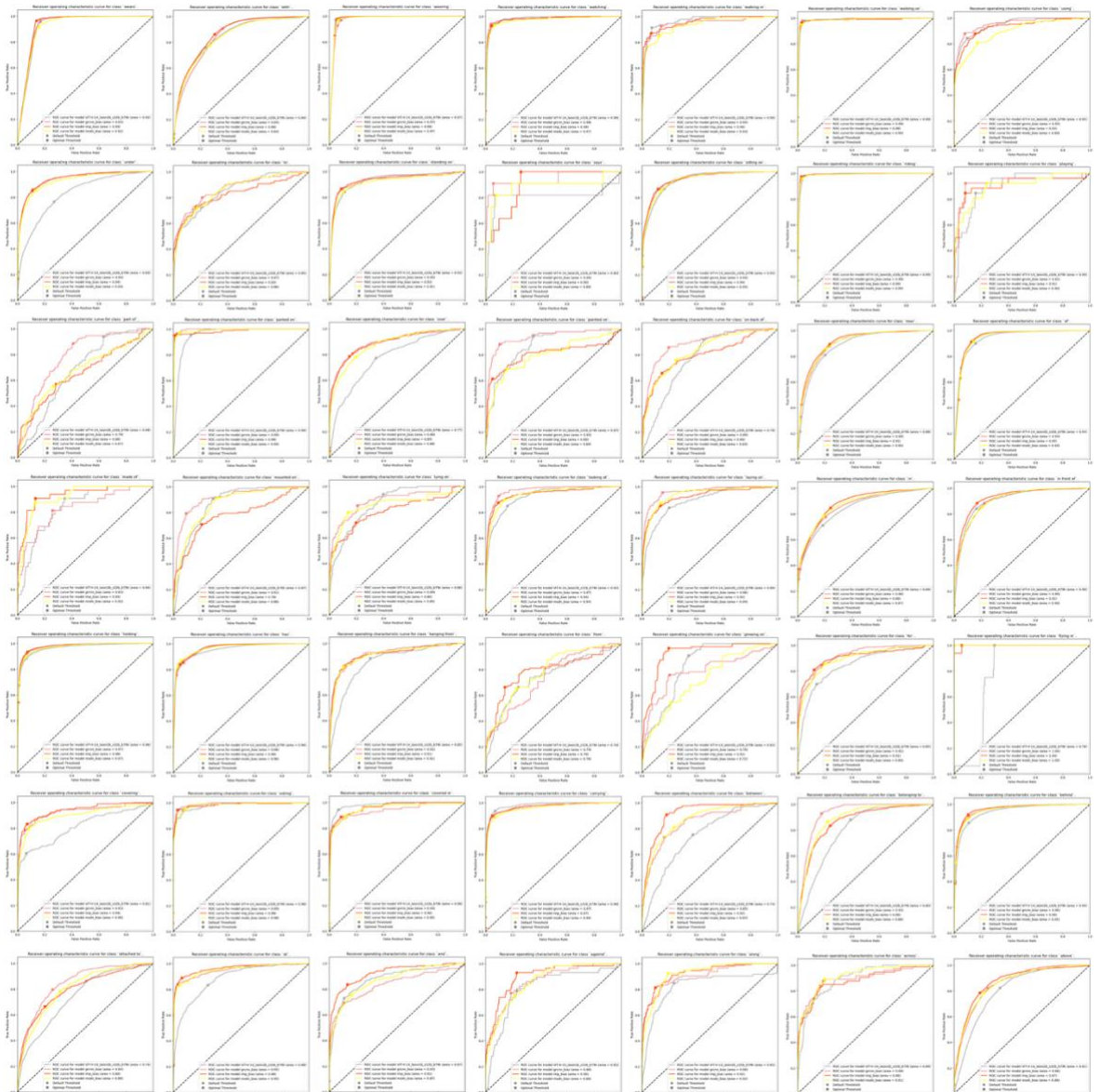


Figure 7.18: ROC Curves for all predicate classes except for ‘on’ in VG150 for ViT and replicated models with biased priori.

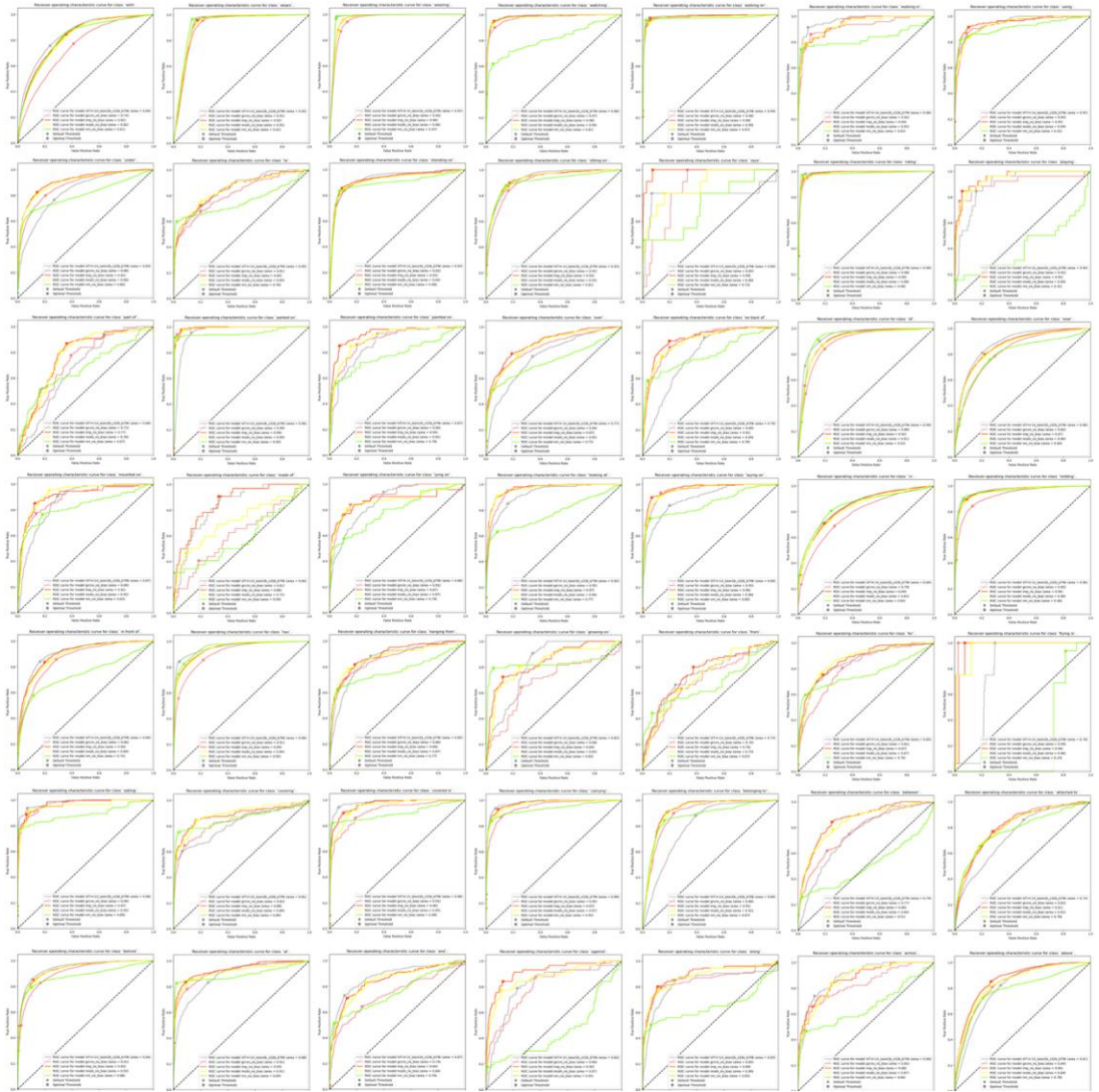


Figure 7.19: ROC Curves for all predicate classes except for ‘on’ in VG150 for ViT and replicated models with no biased priori.

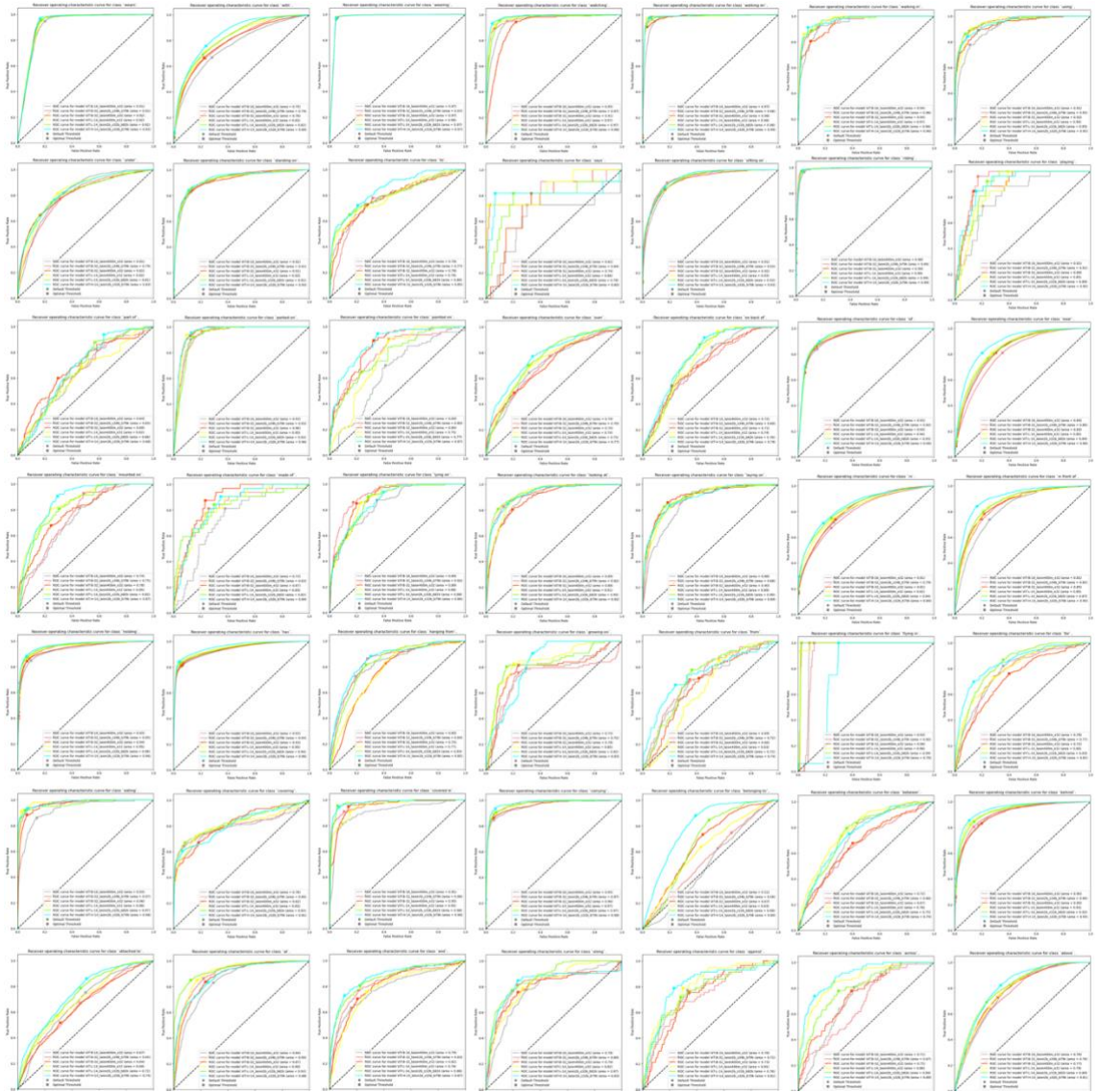


Figure 7.20: ROC Curves for all predicate classes except for 'on' in VG150 for OpenCLIP-based models.