Finding Bounding Boxes for Visual Answers to Natural Language Questions

Manuel Miranda

Instituto Superior Técnico, 1049-001 Lisboa
University of Lisbon, Portugal
manuel.saldanha.m@tecnico.ulisboa.pt

Abstract. This report describes my M.Sc. thesis proposal, which is related with the Toloka visual question answering challenge, i.e. a competition aimed at advancing the state-of-the-art in the field of visual question answering. This research problem combines natural language processing and computer vision, involving the encoding of multimodal information. To adress the Toloka challenge, my M.Sc. research project will specifically focus on the development and evaluation of groundbreaking methods for visual question answering. By harnessing the capabilities of large pre-trained models, this research aims to push the boundaries of the current state-of-the-art in several directions.

Keywords: Neural Networks · Transformers · Computer Vision · Visual Questioning Answering · Object Detection.

1 Introduction

There has been a growing interest in utilizing deep learning techniques for overcoming the challenge of answering natural language questions regarding the contents of images, also known as Visual Question Answering (VQA). Notable advancements have been made in various related tasks, including open-vocabulary detection and semantic segmentation. This exciting research problem combines the fields of Natural Language Processing (NLP) and computer vision, requiring the integration of multi-modal information such as object relationships and textual entities.

The Toloka visual question answering challenge at WSDM Cup 2023 concerns a computer vision task related to VQA, where a system is given a text-based question about an image, and it must infer the corresponding answer. Questions can be arbitrary and they encompass many subproblems in computer vision as well as in NLP, requiring a deep understanding of vision, language, and commonsense knowledge to find the answers. While most previous work in VQA considered a task formulation based on multi-class classification, this challenge involves a different formulation: for a given image and a textual question, the objective is to draw the bounding box around the object correctly responding to the answer for that question. This way, the task has some similarities with previous work on open world object detection or referring expression comprehension. In recent years, VQA has made significant progress, and there are many state-of-the-art models that achieve high accuracy on benchmark datasets. The challenge dataset consists of images associated with textual questions. One entry in the dataset is a (image, question) pair labeled with the ground truth coordinates of a bounding box containing the visual answer to the given question.

In this M.Sc. research project, the primary focus will be on the advancement and evaluation of cutting-edge techniques for VQA, considering the formulation from the Tokola challenge. Building upon the existing state-of-the-art, this study aims to explore new directions and push the boundaries further. Specifically, the project will delve into methods inspired by the latest advancements in vision and language transformers, which have shown great promise in this domain. My research aims to contribute to the field by developing and evaluating innovative approaches that can enhance the performance of VQA systems.

The rest of this document has the following structure: Section 2 starts by introducing the fundamental concepts needed for a better understanding of the proposed approach and its related work. Then, Section 3, reviews some of the current state-of-the-art approaches to different computer vision problems, first describing the evolution on the referring expression comprehension field, over the past couple of years, and then presenting recent studies focused on open-vocabulary detection and referring image segmentation. Section 4 presents the work that will be done for the development of a new and improved open-vocabulary detection method applied to VQA. In the end, Section 5 concludes the document and presents a schedule for the development of the project.

2 Fundamental Concepts

This section describes some of the key concepts, theories, and principles needed for a better understanding of the current state-of-the-art approaches. It covers the essential knowledge that must be acquired to build a solid understanding of the the basics of learning with deep neural networks, transformer models for NLP, computer vision, and vision and language tasks.

2.1 Learning with Neural Networks

A neural network is machine learning algorithm that is designed to recognize underlying relationships in a set of data, trying to replicate the way that biological brains operate. This process is achievable through processing nodes, also known as neurons, that interconnected together form a network. Each neuron gets input from other neurons, processes that input, and then propagates its output to other neurons. This is the basic principle of communication between neurons in a neural network. The strength of the connections between neurons is controlled by weights, which are adjust throughout a learning process to optimize the performance of the network.

The artificial neurons are referred as perceptrons and were originally proposed by Rosenblatt (1957) with the purpose of helping with binary classification tasks. The perceprton was one of the first neural network models to be designed, and it is a algorithm used for supervised learning that has one single node, which processes input data and generates a binary output. The perceptron model is thus a single-layer neural network consisting of distinct parts, which are the input values, the weights, the bias, the net sum, and the activation function. It can be mathematically represented by:

$$\hat{y_i} = f\left(\sum_{i=1}^{n} w_i \times x_i + b\right). \tag{1}$$

The model starts by multiplying all the input values x_i by the respective weights w_i, and adds all the multiplied values in order to obtain the weighted sum. The bias term b is also added to this weighted sum to improve the model's performance, by allowing it to adjust the output range of the activation function. Then, the result is fed to an activation function f to determine the desired output. This activation function is normally a unit step function, which outputs the value 1 for inputs above a certain threshold, or the value 0 otherwise. In vector form, the output of the perceprton can be computed as:

$$\hat{y_i} = f(\mathbf{w}^T\mathbf{x} + b), \tag{2}$$

where $\mathbf{x}$ and $\mathbf{w}$ are both vectors, with the same dimensionality, that represent the input values and the respective weights. The values inside the weight vector, along with the bias, can be learned through a process called training, that modifies the model to minimize the error between the predicted output and the actual output, by using the perceptron learning rule. This rule allows the model to update the values of the weights through running multiple iterations, until the predicted output fits the actual one. Random values are used to initialize the starting weight vector.

$$\mathbf{w}^{new} = \mathbf{w}^{old} + \eta(y_i - \hat{y_i})\mathbf{x}_i. \tag{3}$$

The expression above shows how the weights are updated in each learning iteration according to the perceptron rule, where $\boldsymbol{\eta}$ is a hyperparameter known as the learning rate, that is set before the training procedure and determines the size of the weight update. The difference between the actual output $\boldsymbol{y_i}$ and the predicted one $\boldsymbol{\hat{y}_i}$ represents the error. Despite being revolutionary, the perceptron had limitations due to its linear nature, which makes it incapable of learning non-linearly separable functions, and allowing it only to classify data points that could be separated by a straight hyperplane, as exposed by Minsky and Papert (1969). To overcome this problem, multi-layer perceptrons came into play.

A Multi-Layer Perceptron (MLP), also known as Feed-forward Neural Network (FFNN), is a combination of several perceptrons, or neurons, that when grouped together form a graph without any cycles, in order for the computation to be done sequentially. A MLP consists of multiple layers. Each layer has several identical neurons and each neuron is connected to every neuron in the next layer, making it a fully connected layer. The layer that comes first is the input layer, the final layer is the output layer, and all the layers in between are called hidden layers, since it is not possible to predict what they will compute until the training is finished. The MLP can be mathematically described by the following equation:

$$\mathbf{h}^{(i)} = f^{(i)}\Big(\mathbf{W}^{(i)}\mathbf{h}^{(i-1)} + \mathbf{b}^{(i)}\Big), \ \ \forall i \in \{0, 1, ..., n\}, \tag{4}$$

where $\mathbf{h^{(\boldsymbol{i})}}$ is a vector that has the values of all the neurons in a certain hidden layer i, considering that $\mathbf{h^{(0)}}$ corresponds to the input layer vector $\mathbf{x}$, and $\mathbf{h^{(\boldsymbol{n})}}$ corresponds to the output layer $\boldsymbol{\hat{y}}$. There is a weight matrix for every pair of neurons in two consecutive layers $\mathbf{W^{(\boldsymbol{i})}}$. Each layer also has a bias vector $\mathbf{b^{(\boldsymbol{i})}}$. The activation function in each layer is represented as $\boldsymbol{f}^{(i)}$, since different layers may have different activation functions. The final output of a MLP with n hidden layers can be computed with the expression shown below:

$$\hat{\mathbf{y}} = f^{(n)}\Big(\mathbf{W}^{(n)}f^{(n-1)}\Big(\mathbf{W}^{(n-1)}...f^{(2)}\Big(\mathbf{W}^{(2)}f^{(1)}\Big(\mathbf{W}^{(1)}\mathbf{x}+\mathbf{b}^{(1)}\Big)+\mathbf{b}^{(2)}\Big)...+\mathbf{b}^{(n-1)}\Big)+\mathbf{b}^{(n)}\Big). \tag{5}$$

The MLP learns complex patterns, due to the structure of multiple layers involving non-linear activation functions. Some of the most commonly used activation functions are the logistic sigmoid, the ReLU, the hyperbolic tangent and the softmax. This last function normally appears in the output layer for multi-class classification problems, since it computes a probability distribution over a set of K mutually exclusive classes. These activation functions are, respectively, represented by the following equations:

$$\text{sigmoid}(h) = \frac{1}{1+e^{-h}}, \tag{6}$$

$$\text{ReLU}(h) = \max\{0, h\}, \tag{7}$$

$$\tanh(h) = \frac{e^h - e^{-h}}{e^h + e^{-h}}, \tag{8}$$

$$\text{softmax}(h)_i = \frac{e^{h_i}}{\sum_{j=1}^{K} e^{h_j}}, j \in \{1, ..., K\}. \tag{9}$$

The error of a MLP can be defined as the difference between the actual and the predicted outputs ($\mathbf{y}$ and $\hat{\mathbf{y}}$), and it is measured by a loss function that indicates how well the model performs. For regression problems, this function can be simply represented by the mean squared error, with N being the number of training examples.

$$\mathcal{L}(\hat{\mathbf{y}}) = \frac{1}{N} \cdot \sum_{i=1}^{N}(\mathbf{y_i} - \hat{\mathbf{y}}_\mathbf{i})^2. \tag{10}$$

For multi-class classification problems, when softmax is used in the output layer, the cross entropy function can be applied to compute the loss.

$$\mathcal{L}(\hat{\mathbf{y}}) = -\sum_{i=1}^{N} \mathbf{y_i} \log \hat{\mathbf{y}}_\mathbf{i}. \tag{11}$$

Training two MLP consists of adjusting the weights such that their combination minimizes the loss function. In order for that to happen, MLPs recur to a process called gradient descent, which is an optimization algorithm corresponding to the following expression:

$$\mathbf{W}^{new} = \mathbf{W} - \eta \nabla \mathcal{L}(\mathbf{W}). \tag{12}$$

This equation is similar to Equation 3, with the exception of $\nabla \mathcal{L}$, i.e. the gradient of the loss function, that is used for determining the direction to find the minimum of the loss function. The gradient can be determined through back-propagation, which is an algorithm that recursively applies the chain rule of calculus, allowing the gradients of the loss function to be computed with respect to the weights and biases of each layer of the network, starting from the output layer and working backwards towards the input layer. In order find the gradient with respect to the network's final weights, $\mathbf{W^{(n)}}$, the chain rule needs to be applied:

$$\nabla \mathcal{L}(\mathbf{W}^{(n)}) = \frac{\partial \mathcal{L}(\hat{\mathbf{y}})}{\partial \mathbf{W}^{(n)}} = \frac{\partial \mathcal{L}(\hat{\mathbf{y}})}{\partial \hat{\mathbf{y}}} \cdot \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{h}^{(n)}} \cdot \frac{\partial \mathbf{h}^{(n)}}{\partial \mathbf{W}^{(n)}}. \tag{13}$$

After that the gradient with respect to $\mathbf{W^{(n-1)}}$ also needs to be determined, having again as reference Equation 13. This process is continuously repeated until the first weights of the network $\mathbf{W^{(1)}}$ are reached. Gradient optimization methods, such as Stochastic Gradient Descent (SGD) and its variants, are the backbone of deep neural network training. However, one of the most common optimization techniques is the Adam optimizer (Kingma and Ba, 2014), which extends SDG by storing the diminishing average of the previous gradients m_t, and the diminishing average of the previous squared gradients v_t.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla \mathcal{L}(\mathbf{W}), \tag{14}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla \mathcal{L}(\mathbf{W}))^2. \tag{15}$$

The equations above demonstrate how Adam calculates the mean m_t and the uncentered variance v_t of the gradients. Since these are initialized as vectors of zeros, it is important to denote that m_t and v_t are biased towards zero, particularly in the initial time steps and when the decay rates, β_1 and β_2, are small. The Adam optimizer fixes this problem by computing bias-corrected m_t and v_t with the formulas:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \tag{16}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}, \tag{17}$$

where β_1^t and β_2^t are the corresponding decay rates at time step t. These bias-corrected weight parameters $\hat{m}_t$ and $\hat{v}_t$, along side with the constant ε, are used on the following update rule:

$$\mathbf{W}^{new} = \mathbf{W} - \hat{m}_t \Big(\frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} \Big). \tag{18}$$

The Adam optimizer gives much higher performance when compared with other optimization algorithms like AdaGrad, RMSProp and AdaDelta, outperforming all of them by a big margin and resulting in a highly optimized gradient descent.

2.2 The Transformer Architecture

The Transformer is a powerful deep learning model architecture, introduced by Vaswani et al. (2017), that has been widely adopted for natural language processing tasks such as language translation, text classification, and text generation. The key innovation of the Transformer model is the self-attention mechanism. This mechanism allows the Transformer to selectively focus on different parts of an input sequence during processing, which has proven to be particularly effective for NLP tasks, where understanding the relationships between tokens and their context is crucial. The Figure 1 describes the self-attention mechanism. In order to calculate self-attention, the embedded

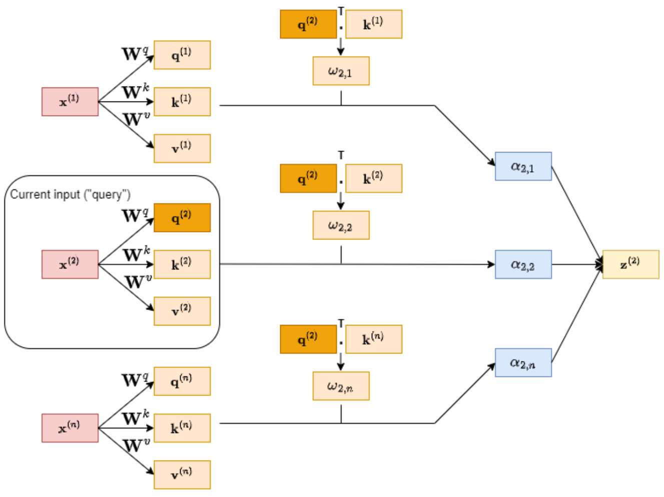


Fig. 1. Illustration for the self-attention mechanism, exemplifying the computations for second input element.

input sequence $\mathbf{x}$ needs to be projected into query, key, and value components. By recurring to three weight matrices, that are adjusted during the training process, known as $\mathbf{W^q}$, $\mathbf{W^k}$, and $\mathbf{W^v}$, it is possible to determine the respective query, key, and value sequences via matrix multiplication between the embedded inputs $\mathbf{x}$ and the weight matrices $\mathbf{W}$:

$$\begin{aligned} \mathbf{q}^{(i)} &= \mathbf{W}^q \cdot \mathbf{x}^{(i)} \qquad \text{for } i \in [1, n], \\ \mathbf{k}^{(i)} &= \mathbf{W}^k \cdot \mathbf{x}^{(i)} \qquad \text{for } i \in [1, n], \\ \mathbf{v}^{(i)} &= \mathbf{W}^v \cdot \mathbf{x}^{(i)} \qquad \text{for } i \in [1, n], \end{aligned} \tag{19}$$

where the index i refers to the token index position in the input sequence, which has length n. The weight matrices $\mathbf{W^q}$ and $\mathbf{W^k}$ are of dimensionality $d_k \times d$, and both vectors $\mathbf{q}^{(i)}$ and $\mathbf{k}^{(i)}$ must have the same size, d_k, since the dot-product between the query and key vectors is going to be computed. It is also important to have in mind that d represents the size of each word vector $\mathbf{x}$. As for the number of elements in $\mathbf{v}^{(i)}$, which determines the size of the resulting context vector, it is arbitrary, d_v, making $\mathbf{W^v}$ have the shape $d_v \times d$. Figure 1 presents an example on how to compute the attention vector for the second input element $\mathbf{x}^{(2)}$. First, the keys and values of all the remaining input elements need to be determined as well, since they are required to obtain the unnormalized attention weights, ω, given by

$$\omega_{ij} = \mathbf{q}^{(i)^T} \mathbf{k}^{(j)}. \tag{20}$$

The following step consists of calculating the normalized attention weights, α, achieved by making the product between ω and $1/\sqrt{d_k}$, before applying the softmax function:

$$\alpha_{i,j} = \text{softmax}\left(\frac{\omega_{i,j}}{\sqrt{d_k}}\right). \tag{21}$$

The factor d_k is used to scale ω, which ensures that the Euclidean length of the weight vectors will be approximately the same, reinforcing the model's ability to converge during training. Finally, the context vector $\mathbf{z}^{(i)}$ is determined, and it represents the attention-weighted version of the original input element $\mathbf{x}^{(i)}$, with the context of all the other input elements,

$$\mathbf{z}^{(i)} = \sum_{j=1}^{n} \alpha_{i,j} \mathbf{v}^{(j)}. \tag{22}$$

When these previous steps are applied to all the input elements in the sequence, n context vectors are outputted. By stacking all the vectors $\mathbf{z}^{(i)}$ together, a matrix $\mathbf{Z}$, known as a self-attention head,

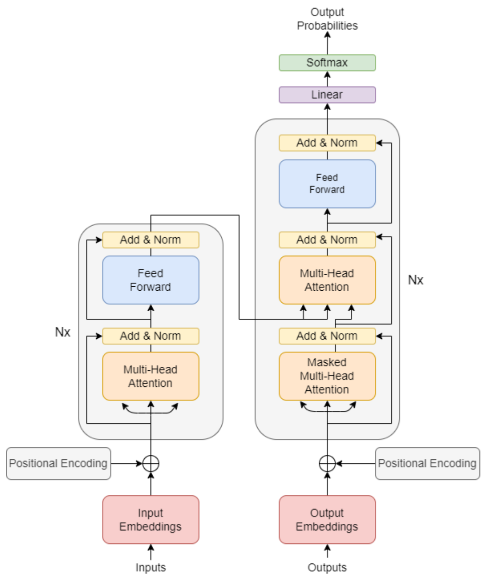


Fig. 2. The Transformer architecture.

is formed. However, in order to improve the self-attention layer, the Transformer uses a mechanism known as multi-head attention, which increases the set of query, key, and value weight matrices from one to eight ($h = 8$). These multiple sets ($\mathbf{W}_i^q, \mathbf{W}_i^k, \mathbf{W}_i^v$) generate multiple output matrices, that are called heads $\mathbf{Z}_i$, and allow the model to attend to information from different representations at different positions. The multi-head attention $\mathbf{M}$ is then obtained by concatenating the eight output matrices into single one, and multiplying it by a weight matrix $\mathbf{W}^{\mathbf{O}}$, that was trained jointly with the model.

$$\mathbf{M} = \text{Concat}(\mathbf{Z}_1, ..., \mathbf{Z}_h) \cdot \mathbf{W}^O. \tag{23}$$

Transformers use an encoder-decoder architecture, where the encoder is responsible for mapping an input sequence into an abstract representation that possess all the learned information of that input. The decoder takes that representation and tries to generate, step by step, a single output while also being fed the previous output. Figure 2 represents the Transformer architecture, where the encoder block is a stack of six identical encoder layers on top of each other. Each layer has two sub-layers. The first is a multi-head self-attention layer, as explained before, allowing the encoder to look at other tokens in the input sentence while it encodes a specific token. The second is a position-wise fully connected feed-forward network, with a couple of linear layers featuring a ReLU activation function in between. The model also uses a residual connection around each of the two sub-layers, followed by a layer normalization. The residual connections are used to help with the flow of information through the network, by allowing gradients to go through the network directly, improving the performance of the encoder. The layer normalizations are used to stabilize the network, which reduces the necessary training time and allows the network to more easily learn features and patterns in the input sequence.

The decoder block is similar to the encoder, since it also has a stack of six identical decoder layers on top of each other. However, each layer has three sub-layers. The first sub-layer is a masked multi-head self-attention layer, that forces the decoder to only attend to earlier positions in the output sequence that were already predicted. This mechanism, combined with the fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i. The second sub-layer is a multi-head self-attention layer, where the queries come from the previous decoder sub-layer and the memory keys and values from the output of the encoder block. This process allows the decoder to decide which encoder input is relevant to put focus on. The output goes through a position-wise fully connected feed-forward network, that is the third sub-layer, for further processing. The decoder also has residual connections around each of the sub-layers, followed by layer normalization.
The decoder block outputs a vector that goes through a linear layer, which computes a vector as

big as the number of tokens present in the model's output vocabulary. This vector gets fed into a softmax layer, which produces probability scores. The index with the highest probability score corresponds to the index of the predicted token.

Since neural networks learn through numbers, the need for representing word tokens in a numerical way arises. To solve this issue, both encoder and decoder inputs go through an embedding layer that map tokens to a set of real-valued vectors. However, the Transformer needs to receive some information about the positions in the input embeddings, in order to discriminate between tokens at different positions. This is achievable through a positional encoding. In the original paper, the authors used sine and cosine functions of different frequencies, due to their continuous and periodic properties.

$$\mathrm{PE}_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}}), \tag{24}$$

$$\mathrm{PE}_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}}). \tag{25}$$

For every odd index on the input vector, this mechanism creates a vector using Equation 25, and for every even index, a vector is converted using Equation 24. By adding these vectors to their corresponding input embeddings, the network encodes information on the position of each vector.

2.3 Transformers for Natural Language Processing

Bidirectional Encoder Representations from Transformers (BERT) and Generative Pre-trained Transformer 2 (GPT-2) are two of the most significant breakthroughs in the field of NLP. These two models, based on the Transformer architecture, were developed to improve some of the limitations of the original Transformer.

The BERT model, introduced by Devlin et al. (2018), is basically a trained Transformer encoder with a stack of multiple encoder layers (twelve for BERT$_{\text{base}}$ and twenty four for BERT$_{\text{large}}$). The model takes a sequence of tokens as input, which keep flowing up the stack. Each layer applies multi-head self-attention, passes its results through a feed-forward network, and then hands the result to the next layer.

In order to handle a variety of down-stream tasks, BERT receives as input a sequence of tokens, which may represent a single sentence or two sentences packed together. The first token of every sequence is always a special classification token known as [CLS], that represents the start of the sequence. The sentences in the sequence can be distinguish by another special token, [SEP], that represents the end of each sentence. These special tokens, along with the normal tokens that can represent a word or part of a word of each input sentence, are converted into token embeddings, when mapped to a set of real-valued vectors. These vectors added to a learned embedding, known as segment embedding, indicating which sentence does a token belong to. Finally, BERT introduces a learned position embedding, which encodes the position of each token in the sequence. For a given token, its input representation is constructed by adding the corresponding token, segment, and position embeddings. BERT is a bidirectional model, meaning that it can take into account the context of tokens both to the left and right of a given token. The authors proposed using two pre-training tasks, called Masked Language Modeling (MLM) and Next Sentence Prediction

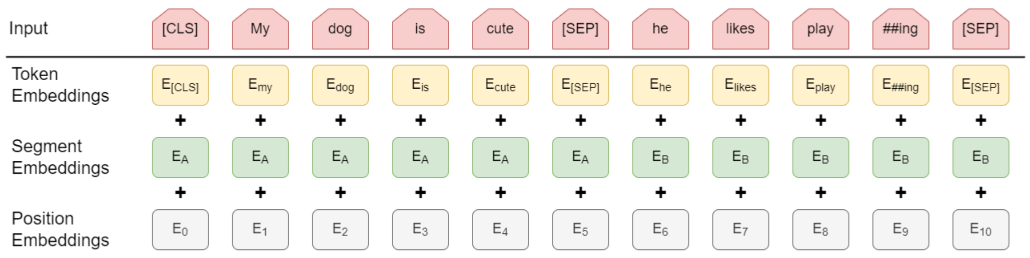


Fig. 3. The BERT input representation.

(NSP), for initializing the model parameters from a collection of unlabeled textual documents. In MLM, a certain percentage of the input tokens are randomly masked (i.e, normally 15%), and the model is trained to predict the original value of these masked tokens based on the context of the surrounding words. Beyond masking the input, BERT also mixes things a bit in order to improve how the model later fine-tunes. Sometimes, it randomly replaces a token with another token and asks the model to predict the correct word in that position. In NSP, the model is trained to predict whether two sentences in a given input sequence are contiguous or not. This helps the model to learn relationships between sentences and capture context across sentence boundaries.

After being pre-trained, a fine-tuning mechanism can be used to adapt the model to a specific NLP task, by training it on a task-specific dataset. Fine-tuning allows the model to learn task-specific features and representations, while still leveraging the general language understanding learned during pre-training.

The GPT-2 model, presented by Radford et al. (2019), only has a stack of twelve decoder layers. Since there is no encoder in this set up, the decoder layers do not have the encoder-decoder attention sub-layer from the original Transfomer architecture. During pre-trainig, GPT-2 uses a Language Modeling (LM) task, which is an auto-regressive formulation that is based on having the model predict the next token in a sequence of tokens, by forcing it to only have in consideration the previously predicted tokens. After being trained on a massive amount of text data, GPT-2 has learned useful representations of the input data, in a way that captures its underlying structure and meaning. The GPT-2 model, similarly to BERT, can be fine-tuned on specific downstream tasks, such as text classification, question-answering, or language translation, by providing task-specific labeled data and fine-tuning the pre-trained model on this data.

2.4 Transformers for Computer Vision

The Vision Transformer (ViT) model (Dosovitskiy et al., 2020) corresponds to a standard transformer encoder that receives as input a vector sequence of token embeddings. In order to process 2D images, ViT structures the input image data, of height H, width W and number of channels C, into a feature matrix, of dimensions $(N + 1) \times D$. In order to achieve this goal, the image is partitioned into N smaller two-dimensional patches with a resolution of $P \times P$ pixels, where P is a pre-defined parameter. These patches are flattened, resulting in N vectors, of size $P^2 \cdot C$. The vectors are multiplied by a trainable weight matrix $\mathbf{E}$, of shape $(P^2 \cdot C) \times D$, that learns to linearly project each vector into another vector with size D. Similarly to BERT, the ViT model has a learnable class token $\mathbf{x}_{class}$, of dimensionality $1 \times D$, which is added at the beginning of the patch embeddings sequence, which was outputted by the linear projection operation. The last representation of this special token, that comes from the last layer L of the transformer encoder, is fed through the classification layers. An additional trainable positional embedding $\mathbf{E}_{\boldsymbol{pos}}$, with shape $(N + 1) \times D$, is added to the concatenated sequence of projections, allowing the model to have a spatial representation of each patch within the sequence. This logic can be mathematically expressed in the following equation:

$$\mathbf{z}_0 = [\mathbf{x}_{class}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; ...; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos} \qquad \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C)\times D}, \mathbf{E}_{pos} \in \mathbb{R}^{(N+1)\times D}. \tag{26}$$

The aforementioned procedure results in the first input of the transformer encoder block $\mathbf{z_0}$, where each input feature is represented as a $(N + 1) \times D$ matrix. The encoder block has the ability to learn more abstract features from this input, by recurring to a stack of L encoder layers, each one described by the equations bellow:

$$\mathbf{z}'_l = \mathrm{MSA}(\mathrm{LN}(\mathbf{z}_{l-1})) + \mathbf{z}_{l-1}, \qquad l = 1...L, \tag{27}$$

$$\mathbf{z} = \mathrm{MLP}(\mathrm{LN}(\mathbf{z}'_l)) + \mathbf{z}'_l, \qquad l = 1...L, \tag{28}$$

MSA stands for multiheaded self-attention and LN for layer normalization. The MLP contains two layers with a GELU non-linearity activation function. As mentioned previously, only the last representation of the learnable class token ($\mathbf{x}_{class} = \mathbf{z}_0^0$) is used for classification, through a classification head. This head is composed by a MLP with one hidden layer during pre-training, and by a single linear layer during fine-tuning. In each case, the final output is a vector $\mathbf{y}$ with the same size as the number of classes present for classification, containig the probabilities associated to each class.

$$\mathbf{y} = \text{MLP}(\text{LN}(\mathbf{z}_L^0)), \qquad \text{for pre-training,} \tag{29}$$

$$\mathbf{y} = \text{LINEAR}(\text{LN}(\mathbf{z}_L^0)), \qquad \text{for fine-tuning.} \tag{30}$$

In summary, the ViT employs the encoder part of the original Transformer architecture, by sending a sequence of embedded image patches, after being linear projected and augmented with positional information, as inputs to the encoder. The information flows through the encoder block, which finally that outputs the value of the learnable class embedding, to generate a classification output. This is achievable by attaching a classification head to the output of the encoder.

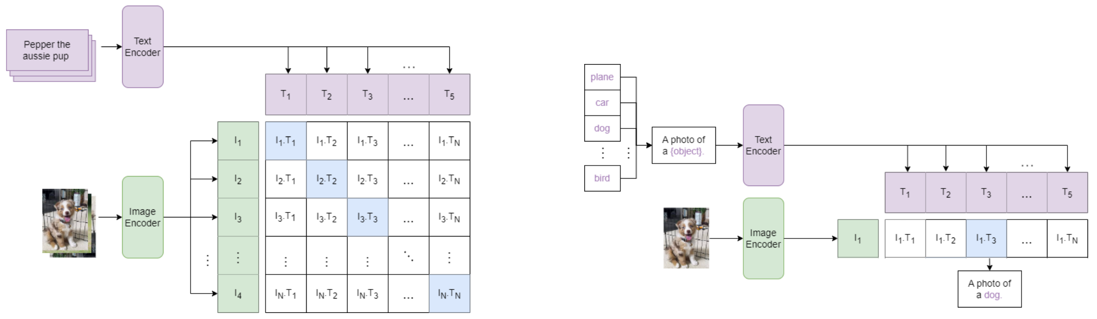
2.5 Transformers for Vision and Language Tasks

Due to the Transformers' success in NLP tasks, researchers started looking at whether they could be used for multimodal tasks, including both visual and linguistic data. Vision and Language (V&L) Transformers are a group of models that use Transformers to process both visual and linguistic modalities in a joint representation. It is possible to divide these models into two categories, namely single-stream and dual-stream models, as explained by Bugliarello et al. (2021).

Single-stream models process both visual and linguistic input together, recurring only to a single stack of Transformer sub-layers. Single-stream models concatenate the embedded visual inputs and the embedded textual inputs into a single input fed to the Transformer. This procedure aims to use the attention mechanism of the transformer, over both modalities, to discover alignments between visual and languages embeddings, allowing an early and unconstrained fusion of cross-modal information. This type of models are simpler and easier to implement when compared with dual-stream models, despite not capturing more complex interactions between the two modalities, limiting the performance over some tasks. In dual-stream models the visual and linguistic embeddings are first processed by two independent stacks of transformer sub-layers, separately, and then fed into cross-modal sub-layers where intra-modal interactions are often alternated with inter-modal interactions. The intra-modal interactions can be achieved through a transformer sub-layer computing the attention of each modality independently. In turn, for the inter-modal interactions, each stack can first compute query, key, and value matrices, before passing the keys and values to the stack that is processing the inputs of the other modality, via a cross-modal attention module. By doing this, these models explicitly limit interactions across modalities at each layer, restricting some of the interactions that are possible in a single-stream encoder while escalating their expressive power by separate sets of learnable parameters.

The Contrastive Language-Image Pre-Training (CLIP) model presented by Radford et al. (2021), can be seen as a dual-stream model where the inter-modal interactions are made through a shallow computation. The model is trained on a variety of (image, text) pairs, through a contrastive objective. CLIP has a multimodal architecture that leverages more than one domain, namely NLP and computer vision, to learn cross-modal alignments. Additionaly, CLIP is able to accurately recognize classes and objects that it has never seen before, through its impressive zero-shot capabilities.

During training, given a batch of N (image, text) pairs, CLIP uses a contrastive pre-training method to learn a multimodal embedding space, by jointly training an image encoder (i.e. a ResNet-50 or ViT model) and a text encoder (i.e. a standard Transformer) to maximaize the cosine similarities of the image an text embeddings of the N correct (image, text) pairs, while minimizing the the cosine similarities of the dissimilar pairs. For every image in the batch, the image encoder computes an image embedding $[\mathbf{I}_1, \mathbf{I}_2, ..., \mathbf{I}_N]$. Each vector is of size S, where S is the size of the latent dimensionality, resulting on a $N \times S$ matrix. Similarly the text prompt embeddigns, created by the text encoder, can be also described as $[\mathbf{T}_1, \mathbf{T}_2, ..., \mathbf{T}_N]$, forming a matrix of size $N \times S$ as well. By multiplying these matrices to obtain the cosine similarities scores


Pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_5
I_1 I_2 I_3 ⋮ I_4
$I_1{\cdot}T_1$ $I_1{\cdot}T_2$ $I_1{\cdot}T_3$ ... $I_1{\cdot}T_N$
$I_2{\cdot}T_1$ $I_2{\cdot}T_2$ $I_2{\cdot}T_3$ ... $I_2{\cdot}T_N$
$I_3{\cdot}T_1$ $I_3{\cdot}T_2$ $I_3{\cdot}T_3$ ... $I_3{\cdot}T_N$
$I_N{\cdot}T_1$ $I_N{\cdot}T_2$ $I_N{\cdot}T_3$ ... $I_N{\cdot}T_N$
plane
car
dog
bird
A photo of a {object}.
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_5
I_1
$I_1{\cdot}T_1$ $I_1{\cdot}T_2$ $I_1{\cdot}T_3$ ... $I_1{\cdot}T_N$
A photo of a dog.

(a) Contrastive pre-training. (b) Zero-shot prediction.

Fig. 4. The Contrastive Language-Image Pre-Training model.

between the image and the text embeddigns, a new $N \times N$ matrix is created, as shown in Figure 4(a). A symmetric cross entropy loss is then optimized over these cosine similarity scores, to encourage the model to learn meaningful representations of both image and text data, and make accurate predictions based on their relationships. The symmetric cross entropy loss can be given by the expression.

$$\mathcal{L} = \frac{\mathcal{L}_I + \mathcal{L}_T}{2}, \tag{31}$$

where $\mathcal{L}_I$ and $\mathcal{L}_T$ are the cross-entropy losses for the cosine similarities of the image and text embeddings, respectively. After pre-training the image and text encoders, CLIP can be used to perform zero-shot classification, as shown in Figure 4(b), encoding a set of text prompts into text embeddings. The image encoder is fed with the image to be classified and transforms it into a image embedding. CLIP then computes the pairwise cosine similarities between the image and the text embeddings. The text prompt with the highest similarity is chosen as the prediction. Although CLIP was created to use full textual descriptions, many public datasets feature images paired with just one single word labels. To surpass this problem, single word labels, such as *bird*, or *car*, are converted to sentences through the prompt *a photo of a* {*object*}, where *object* represents any single word label of the dataset.

2.6 Object Detection with Deep Learning

Object detection is a computer vision task that aims to localize and classify objects within an image or video. Specifically, object detection draws bounding boxes around these detected objects and gives a class label for each box, providing the location where known objects are in a given scene. Deep learning based object detection models solve two tasks: first, they find an arbitrary number of objects, possibly even zero, and then classify every single object and estimate its size with a bounding box. The models that split these tasks into two separate stages are categorized as two-stage object detectors, while the ones that combine both tasks into one step, achieving higher performance at the cost of accuracy, are categorized as one-stage object detectors.

The Region-Based Convolutional Neural Network (R-CNN) is a two-stage object detector, corresponding to one of the first successful applications of Convolutional Neural Networks (CNNs) to the problem of object detection. The original R-CNN, proposed by Girshick et al. (2014), generates category independent region proposals, i.e. candidate bounding boxes, by using a selective search algorithm (Uijlings et al., 2013) that works as follows: first, it generates a initial sub-segmentation of an input image. Then, recursively combines the smaller similar regions, from the initial sub-segmentation, into larger ones. Finally, uses the segmented region proposals to generate candidate object locations, the selective search paper considers four types of similarity methods when combining the initial small segmentation into larger ones, namely colour similarity, texture similarity, size similarity, and fill similarity. After the algorithm being completed, the features from each region proposal (i.e. candidate bounding boxes) are then extracted by a deep convolutional neural network. The output of the CNN is a 4096 element vector that is fed to a linear SVM for classification. Despite achieving good results, this object detector is slow, since it operates upon a big number of proposed regions (approximately 2000 per image at test time).

To overcome this issue, Fast R-CNN (Girshick, 2015), was proposed as another two-stage object detector, using a pre-trained CNN to extract the features from the region proposals, followed by a Region of Interest (RoI) pooling layer, that extracts features specific for a given input candidate region. The output of the CNN is then interpreted by a fully connected layer, and then the model divides into two outputs: one for the class prediction via a softmax layer, and another with a linear output for the bounding box. Despite the model being significantly faster to train and to make predictions, it still needs a set of candidate regions to be proposed along with each input image.

The Faster R-CNN (Ren et al., 2016) is another two-stage detector that uses a Region Proposal Network (RPN) and a Fast R-CNN on the same features extracted from a pre-trained CNN. The RPN acts as an attention mechanism, by informing the Fast R-CNN network of where to look or pay attention. The region proposals that are outputted by the RPN are bounding boxes based on anchor boxes, which are pre-defined bounding boxes with useful shapes and sizes that are tailored during training, accelerating and improving the region proposals. The class prediction is binary, indicating the objectness of the proposed region. The Mask R-CNN (He et al., 2017) is also an advancement over Fast R-CNN, with the difference that it is capable of predicting an object mask and the respective bounding box at the same time. Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN.

Another well known model, different from the previous ones, is the You Only Look Once (YOLO). Despite not being so accurate as the R-CNN models is much faster. YOLO (Redmon et al., 2016) is a single convolutional network capable of, simultaneously, predicting multiple bounding boxes and class probabilities for those boxes, which makes it a one-stage object detector, through the years new and more improved YOLO versions have been released, the latest until this date being the YOLOv8.

To better understand the fundamentals of YOLO models (Terven and Cordova-Esparza, 2023), it is important to know that they divide the input image into an $S \times S$ grid, if the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell can predict N bounding boxes and the respective confidence scores for the boxes, these scores represent how confident the model is on assuming that a box contains an object. The confidence score should be equal to the IoU between the predicted box and the ground truth, if there is not any object inside a cell, the confidence score of that cell should be zero. Each bounding box is represented by a vector of five elements x, y, w, h, c, each element corresponding to a prediction. The (x, y) elements represent the center of the box relative to the bounds of the grid cell. The width w and height h of the box are predicted relative to the whole image. The confidence prediction c reflects the IoU between the predicted box and any ground truth box. Each grid cell also predicts conditional class probabilities, that are conditioned on every grid cell containing an object. The class probabilities map and the bounding boxes with confidences are then combined into a final set of bounding boxes and class labels, as seen in Figure 5.

YOLO predicts multiple bounding boxes per grid cell. However, during training, only one bounding box prediction can be assigned to each object. To surpass this issue, the authors proposed a non-maximal suppression (NMS) technique, that aims at selecting the best bounding box out of a set of overlapping boxes. This algorithm goes as follows: first, it is defined a value for a confidence threshold and for a IoU threshold. All the bounding boxes are sorted in a descending order of confidence. The boxes that have a confidence score lower than the confidence threshold are removed. The IoU score of the box that has the highest confidence is calculated with every remaining box that belongs to the same class. If the IoU of two boxes are bigger than the IoU threshold, the box with a lower confidence is removed as well, since it is probably covering the same object as the one with a higher confidence score. This procedure is repeated for every box in the image.

This model shares some similarities with R-CNN. Each grid cell proposes potential bounding boxes and scores those boxes using convolutional features. However, YOLO imposes spatial constraints on the grid cell proposals, effectively mitigating the issue of multiple detections of the same object. It also needs much fewer bounding boxes, only 98 per image compared to about 2000 from R-CNN.

None of these models make use of Transformer features to improve the object detection task, the Detection Transformer (DETR), proposed by Carion et al. (2020), was the first to use a Transformer along with a CNN for the purpose. Unlike the other models, the DETR has a very simple architecture, it uses a conventional CNN backbone to learn a 2D representation of an input image, this representation is flatten and supplemented with a positional encoding before passing through a Transformer encoder. Then, the Transformer decoder takes as input a small fixed number of learned positional embeddigns, known as object queries, along with the enconder output. Each decoder output embedding goes through a shared feed-forward network that predicts a class and a bounding box, it exists also a no object class for when the DETR does not detect any object in a image. The assignment of the predictions to the ground truth is a one to one assignment that minimizes the total loss.

3 Related Work

Several previous studies have addressed the combined use of vision and language data. In particular, this section goes through a survey related to advancements on referring expression comprehension (Qiao et al., 2020), and it overviews state-of-the-art methods proposed for open-vocabulary detection and referring image segmentation.

3.1 Referring Expression Comprehension

Referring Expression Comprehension (REC) has the goal of localizing objects in an image based on referring expressions. REC is typically structured to select the best region, out of a set of regions retrieved from an image, that represents the referring expression. The survey by Qiao et al. (2020) reviewed several REC methods across seven categories, starting with the joint embedding approach, that can be divided into CNN-LSTM frameworks and attention mechanisms. In CNN-LSTM frameworks, the LSTM takes a region-level CNN feature and a word vector as input, with the purpose of maximizing the likelihood of the expression for a given referred region, at each time step. From this conventional framework, more accurate similar methods were latter developed, like the Maximum Mutual Information (MMI) method (Mao et al., 2016), the Visual Comparative (Visdif) method (Yu et al., 2016), and the Spatial Context Recurrent ConvNet (SCRC) model (Hu et al., 2016). As for attention mechanisms, they allow machines to focus on a certain part of the input when processing a large amount of information, which brought out many breakthroughs in the REC field. An attention mechanism is able to build element-wise connections between visual and textual information, allowing the information from some specific image regions to be used

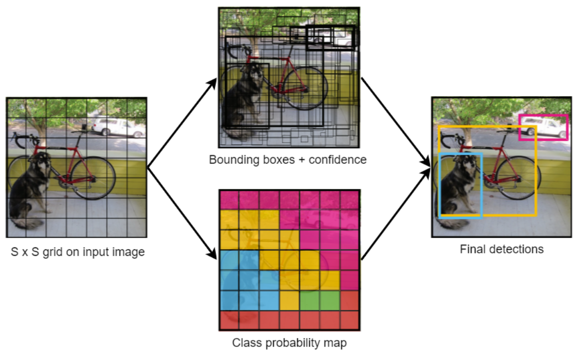


Fig. 5. The You Only Look Once model from Redmon et al. (2016).

when encoding each word in the text and vice versa, leading to semantically-enriched visual and textual representations. This mechanism was explored by authors like Zhuang et al. (2018), that proposed a Parallel Attention (PLAN) approach, and Deng et al. (2018) that proposed an Accumulated Attention (AccumAtt) mechanism.

The second category explored in the survey (Qiao et al., 2020) corresponds to modular-based models, which model parse the input expression into three phrase embeddings, such as subject, relationship, and location embeddings. These embeddings are provided as input to three modules that process the visual region in different ways. After computing individual matching scores, the model combines the module scores to output the final score. This method was used by authors like Hu et al. (2017), that proposed Compositional Modular Networks (CMNs), and Yu et al. (2018) that proposed the Modular Attention Network (MAttNet).

The third category mentioned in the survey corresponds to graph based models, where the graph construction builds a graph over the candidate objects, in which the nodes highlight related objects and the edges are used to recognize the object relationship existing in the expression. After the construction of the graph, the language representation of the expression is fused into the graph to obtain the updated graph representation, allowing the matching module to compute the matching score between the objects and the expression. Other papers went deeper on this subject Wang et al. (2019) proposed a language-guided graph attention network (LGRAN) and Yang et al. (2019) proposed a Dynamic Graph Attention Network (DGA).

The fourth category involves the use of an external parser, where the model regularizes the REC task along the dependency parsing tree of the sentence, and computes visual attention according to its linguistic features. Afterwards, the target object is localized by the model, since the model is capable of accumulating the grounding confidence score along the dependency parsing tree in a bottom-up direction. This approach was explored by authors like Cirik et al. (2018), that introduced GroundNet, and Liu et al. (2019) that developed a Neural Module Tree network (NMTree).

The fifth category in the survey corresponds to vision-language pre-training. The important to attain is that the model is a stack of transformer-like modules that can process multiple modalities jointly. The features of candidate regions as well as the word embeddings are fed together into the vision-language transformer model to capture the relations among the image regions and text tokens. A classifier is applied on top of the hidden state of each candidate region to predict its matching score. Based on this model, Lu et al. (2019) proposed the ViLBERT framework and Su et al. (2019) proposed the VL-BERT framework.

The sixth category corresponds to one-stage approaches, which have significantly faster inference speed, since they do not generate object proposals nor do they extract features from the proposals, unlike two-stage REC methods. The one-stage REC framework consists on fusing the pixel-wise image features and the text features together to obtain a heatmap for the region-of-interest. Then, a target proposal module is adopted to predict the target bounding box from the heatmap. This approach was implemented by authors like Sadhu et al. (2019), that proposed Zero-Shot Grounding (ZSG), and Luo et al. (2020) that proposed a Multi-task Collaborative Network (MCN).

All the categories presented so far are considered supervised. However, supervised models can only handle certain types of grounding in limited training data, and cannot meet the needs of practical applications. The seventh category of weakly supervised approaches refers to locating the object in the image according to the language query, when the mapping between the reference object and the query is unknown in the training phase. In other words, the model trains the system by learning to reconstruct the language information contained in the input expression from the predicted object. Some authors used weakly supervised learning to implement a Knowledge Aided Consistency Network (KAC Net) (Chen et al., 2018), and a Multi-scale Anchored Transformer Network (MATN) (Zhao et al., 2018).

Summing up the results of the aforementioned categories on different datasets, it is possible to denote that the categories that used supervised models performed better that weakly supervised models on the ReferItGame (Kazemzadeh et al., 2014), Flickr 30k entities (Plummer et al., 2015) and Cops-Ref (Chen et al., 2020) datasets. For supervised models, pretraining-based approaches like ViLBERT and VL-BERT had better performances on the RefCOCO & RefCOCO+ datasets (Lin et al., 2015), while the CNN/LSTM framework that had the worst performance on its early attempts. Moreover, some one-stage approaches are indeed much faster, and they can still perform competitively compared to many two-stage approaches.

3.2 Open-vocabulary Detection

Open-vocabulary Detection (OVD) refers to an object detection task that aims at detecting objects from novel categories C_N, beyond the base classes C_B on which the detector is trained, where $C_B \cap C_N = \emptyset$. The approaches have various applications, including sentiment analysis, topic modeling, named entity recognition, and information extraction. OVD enables a system to handle a wide range of inputs and adapt to evolving language usage, making it particularly valuable in dynamic linguistic environments.

3.3 Open-vocabulary Object Detection upon Frozen Vision and Language Models

Kuo et al. (2023) proposed a simple and scalable OVD method, built upon frozen vision and language models (F-VLMs) and capable of reducing training complexity and computing requirement, while achieving the state-of-the-art performance at system level. The main goal of this study F-VLM was to build a open-vocabulary object detector by only training the detector upon frozen features, which preserves the open-vocabulary classification ability of pre-trained VLMs. At test time, the scores from the detector are combined with the VLM scores to achieve the open-vocabulary object detection scores.

During pre-training, the model has only access to the detection labels of base categories C_B, and the main focus is upon contrastively pre-trained VLMs, like CLIP. The frozen image encoder of the VLM is used as the detector backbone, and the text encoder is used for caching the text embeddings of the detection dataset vocabulary. The image encoder has a feature extractor $\mathcal{F}(\cdot)$ and a pooling layer $\mathcal{P}(\cdot)$. By adopting the same backbone architecture as the image feature extractor, the model is able to directly use the frozen weights and inherit the rich semantic knowledge. As for the pooling layer, it is only used at test time for open-vocabulary recognition. In order to build upon the frozen backbone features, the authors proposed a Mask R-CNN head (He et al., 2017) and a feature pyramid network (Lin et al., 2017) as the detector head, which is then randomly initialized and corresponds to the only trainable component of the F-VLM, as mentioned before. The training and inference processes are illustrated in Figure 6.

Considering an input image I, $\mathcal{F}(I)$ represents the backbone features from the image encoder, while $\mathcal{Q}(\cdot)$ represents the function used to obtain a region embedding $\mathbf{r}_b$ from $\mathcal{F}(I)$ and a given region proposal b.

$$\mathbf{r}_b = \mathcal{Q}(\mathcal{F}(I), b). \tag{32}$$

To generate the text embeddings, it is crucial to use the matching text encoder to the image encoder in the VLM, since they were pre-trained together. During training, the proposals that do not match any ground truth boxes in C_B are treated as background. The cosine similarity of $\mathbf{r}_b$ with the text embeddings of C_B and background is computed for each region. A learnable temperature τ is also applied on the the logits. The detection scores $\mathrm{z}(\mathbf{r}_b)$ are given by:

$$\mathbf{z}(\mathbf{r}_b) = \text{softmax}\left(\frac{1}{\tau}[\cos(\mathbf{r}_b, t_{\text{background}}), \cos(\mathbf{r}_b, t_1), ..., \cos(\mathbf{r}_b, t_{|C_B|})]\right). \tag{33}$$

Having in consideration that

$$\cos(a, b) = \frac{a^T b}{||a||||b||}, \tag{34}$$

and that t_i represents the text embeddings of category i, a standard softmax cross entropy loss is also applied on the logits. At test time, the text embeddings are augmented with novel categories, passing from C_B to $C_B \cup C_N$ without excluding the background category. The F-VLM performs the open-vocabulary classification only at test time, and since its backbone features are frozen they do not overfit to the C_B classes and can be directly cropped for region-level classification. At test time, in order to determine the VLM region embedding $\mathbf{v}_b$, the VLM pooling layer $\mathcal{P}(\cdot)$ is applied on the cropped backbone output features $\mathcal{F}(I)$, to obtain the features for a region b. Since the pooling layer requires fixed-size inputs, the region features are cropped and resized with ROI-Align $\mathcal{R}(\cdot)$, as described by He et al. (2017). The detector head is trained in one stage, which is simpler and more space-efficient. The VLM region embeddings are given by

$$\mathbf{v}_b = \mathcal{P}(\mathcal{R}(\mathcal{F}(I), b)). \tag{35}$$

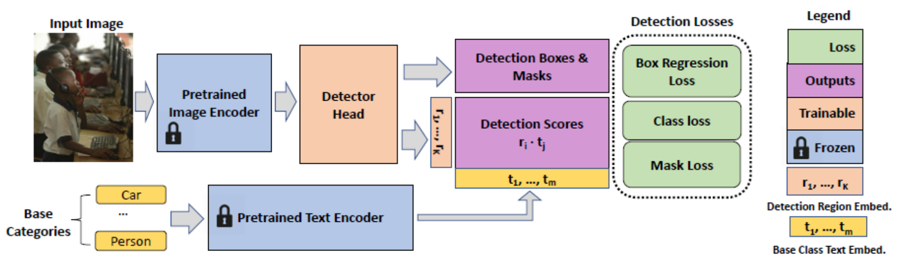
The VLM scores can be calculated on a similar way to the detection scores, with the difference that, for the VLM scores, the text embedding domain is given by $C_B \cup C_N$ while also including the background category:

$$\mathbf{w}(\mathbf{v}_b) = \text{softmax}\left(\frac{1}{T}[\cos(\mathbf{v}_b, t_{\text{background}}), \cos(\mathbf{v}_b, t_1), ..., \cos(\mathbf{v}_b, t_{|C_{B\cup C}|})]\right) \tag{36}$$

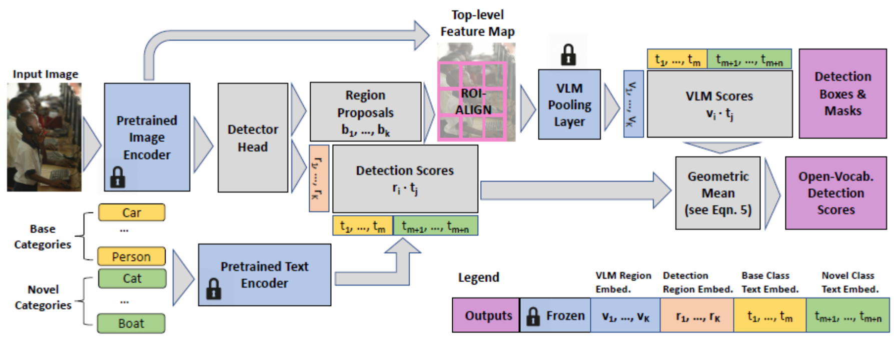
where T is a fixed temperature used to adjust the scale of the VLM scores relative to the detection scores. Since the cropped region features are not sensitive enough to the localization quality of the regions, a geometric mean, to combine the VLM scores $w(\mathbf{v}_b)_i$ with the detection scores $z(\mathbf{r}_b)_i$, is applied for each region b and category i. The final detection scores are given by.

$$s(\mathbf{r}_b)_i = \begin{cases} z(\mathbf{r}_b)_i^{(1-\alpha)} \cdot w(\mathbf{v}_b)_i^{\alpha} & \text{if } i \in C_B \\ z(\mathbf{r}_b)_i^{(1-\beta)} \cdot w(\mathbf{v}_b)_i^{\beta} & \text{if } i \in C_N. \end{cases} \tag{37}$$

Both variables $\alpha, \beta \in [0, 1]$ control the VLM score weights for C_B and C_N categories. For open-vocabulary localization, F-VLM uses class-agnostic box regression and mask prediction heads, which means that, for each region proposal, the model predicts one box and one mask for all categories, rather than one per category. This allows the model to localize novel objects in the open-vocabulary setting.



(a) F-VLM training architecture.



(b) F-VLM inference architecture.

Fig. 6. The F-VLM architecture from Kuo et al. (2023).

The model was trained for a 46.1k iterations with a 1024×1024 image size, large scale jittering for data augmentation (Ghiasi et al., 2021), a batch size 256, weight decay of 1e-4, momentum of 0.9, and an initial learning rate of 0.36. For the score combination, the authors decided to use $\alpha = 0.35$ and $\beta = 0.65$ in Equation 37, a maximum of 300 detections per image, and in Equation 36 the temperature was set to $T = 0.01$. The CLIP (Radford et al., 2021) prompt templates were used to take the average text embeddings of each category. F-VLM offers significant training speed up and compute savings, achieving a new state-of-the-art on the LVIS benchmark at system level, while showing very competitive transfer detection.

3.4 CORA: Adapting CLIP for Open-Vocabulary Detection

Wu et al. (2023) proposed a different OVD approach, introducing the CORA framework with the goal of adapting CLIP for the OVD task. There are two important challenges to have in consideration. One is the fact that object detection conducts recognition on image regions, while the CLIP model is trained on a whole-image input, leading to a distribution gap that challenges the classification performance. The other is the fact that the detector needs to learn object localization for new categories C_N while only being trained on a limited number of base categories. To solve the first obstacle, CORA uses region prompting to modulate the region features for better generalizable region embeddings. To solve the second obstacle, anchor pre-matching is used, to encourage class-aware object localization that can generalize to novel classes during inference.

In region prompting, given an image and a RoI, the whole image is encoded first into a feature map by the CLIP encoder's first 3 blocks. This result is then pooled by RoI-Align (Ren et al., 2016) into a regional feature f_{region}. However, there is a distribution gap between the CLIP image encoder's whole-image feature map and the pooled regional features. This misalignment is fixed by augmenting the region features with learnable prompts $p \in \mathbb{R}^{S \times S \times C}$, with S being the spatial size of the regional feature and C the dimensionality of the regional feature. Given the input regional feature f_{region}, the region prompting is obtained by,

$$v_{prompt} = \mathcal{P}(f_{region} \oplus p), \tag{38}$$

where $\oplus$ represents the element-wise addition and $\mathcal{P}(\cdot)$ the attention pooling module of the CLIP visual encoder. The region prompts are trained on a detection database with the base classes. In order to be optimized, the prompts are also trained by a standard cross-entropy loss to classify the ground truth boxes with their pooled regional features f_{region}. The other model weights are kept frozen when optimizing the region prompts, allowing only the region prompts to be learned. Another common practice similar to region prompting consists of cropping the RoIs of an image and encode them as separate images, before feeding them to the CLIP for text embedding comparisons. This practise in less efficient, since when encoding regions with overlaps, the overlapping regions are encoded more than once in different region crops, which drops the accuracy of the model by making it lose context information. Region prompting provides good results when classifying new unseen classes, and these results are due to the fact that region prompting directly fixes the distribution mismatch right after region pooling occurs, unlike other methods that use irrelevant parameters to compensate for the distribution mismatch.

In anchor pre-matching, the object localizer is implemented by a detection transformer (DETR), which is an object detection architecture based on transformers that formulates object detection as a set-to-set matching problem. The DETR encoder refines the feature map from the frozen CLIP image encoder, while the DETR decoder decodes a set of object queries into box predictions. The type of DETR chosen for this task was a DAB-DETR (Liu et al., 2022), since it formulates the object queries in the DETR architecture as anchor boxes, which accelerates detector training. Each ground truth box is pre-matched to a set of queries with same label. The label $\hat{c}_i$ of an object query is assigned by classifying the associated anchor box b_i. This is given by the following expression:

$$\hat{c}_i = \underset{c \in C_B}{\operatorname{argmax}} \cos(v_i, l_c). \tag{39}$$

The element with the biggest cosine similarity between v_i and l_c is determined, where v_i represents the region feature of the anchor box b_i, and l_c represents the class name embedding of class c. The conditioned object query is given by

$$q_i = \mathrm{MLP}(l_{c_i}). \tag{40}$$

The DETR decoder refines, iteratively, each object query with its associated anchor box (q_i, b_i) into $\hat{y}_i = (\hat{p}_i, \hat{b}_i)$, where $\hat{p}_i$ represents the matching probability to the query's pre-matched class $\hat{c}_i$, and $\hat{b}_i$ represents the refined box coordinates. In order to make the decoder more sensible to the conditioned text embedding, given model predictions, each ground truth box is assigned to the prediction with the same pre-matched label. In other words, for a class c, given N^c box predictions $\hat{y}^c = \{\hat{y}_i|\hat{c}_i = c\}$ that are pre-matched to class c, and given the set of ground truth boxes y^c in class c, it is possible to optimize a permutation of N^c elements that minimizes the following cost

$$\hat{\sigma}_c = \operatorname{argmax} \sum_{i}^{N^c} \mathcal{L}_{cost}(y_i^c, \hat{y}^c_{\sigma(i)}), \tag{41}$$

where $\mathcal{L}_{cost}$ is the matching cost, represented by:

$$\mathcal{L}_{cost}(y, \hat{y}) = \mathcal{L}_{match}(p, \hat{p}) + \mathcal{L}_{box}(b, \hat{b}). \tag{42}$$

In the previous expression, $\mathcal{L}_{match}(p, \hat{p})$ is a binary classification loss, and $\mathcal{L}_{box}(b, \hat{b})$ illustrates the localization error of $\hat{b}$ with reference to b. The model is optimized by the following loss expression:

$$\mathcal{L} = \sum_{c \in C_B} \mathcal{L}_{match}(p^c, \hat{p}^c_{\hat{\sigma}_c}) + \mathcal{L}_{box}(b^c, \hat{b}^c_{\hat{\sigma}_c}) = \lambda_{focal}\mathcal{L}_{focal} + \lambda_{L_1}\mathcal{L}_{L_1} + \lambda_{GIoU}\mathcal{L}_{GIoU}, \tag{43}$$

where $\mathcal{L}_{match}$ implements a focal loss, proposed by Lin et al. (2018), and $\mathcal{L}_{box}$ corresponds to a weighted sum of a L_1 and GIoU losses, mentioned in Zhai et al. (2020). In the inference stage, the region prompting method that was introduced previously, is used to classify the predicted boxes $\hat{b}_i$ for better classification accuracy. The product between the class score and the pre-matching score measures the box quality:

$$P(\hat{b}_i \in c) = \hat{p}_i \cos(\hat{v}_i, l_c). \tag{44}$$

Conditional matching is similar to the anchor pre-matching method, since it also proposes to condition the queries on the text embedding for class-aware regression. However, it suffers from repetitive per-class inference, which causes the computation and the memory consumption to scale lineraly with the vocabulary size. Anchor pre-matching overcomes this issue, since all the classes can be decoded together in one pass, eliminating the need for repetitive per-class decoding.

The region prompts were trained for 5 epochs with a base learning rate of 1e-4, which decays after epoch number 4 by a factor of 0.1. The localizer is trained for 35 epochs with a learning rate of 1e-4, without learning rate decay. Both the region prompts and the localizer are trained with batch size of 32 by the AdamW optimizer, with 1e-4 weight decay. The model is tested on the COCO and LVIS OVD benchmarks. For the larger pre-trained model, CORA improves the previous state-of-the-art RegionCLIP (Zhong et al., 2021) by 2.4 AP50 on the COCO benchmark.

3.5 Region-aware Open-vocabulary Vision Transformers

Kim et al. (2023) presented a region-aware open-vocabulary vision Transformer (RO-ViT), noting that most VLMs do not utilize adequately the notion of regions in the pre-training process. For contrastive image-text pre-training, the authors propose a ViT for the image encoder, that receives as input the image patch embeddings added with cropped positional embeddings, and outputs the image embeddings by using a global average pooling at the ViT last layer. The text encoder is a standard 12-layer Transformer. Both the image and the text embeddings are L2 normalized and trained with a focal loss on the same image-text dataset as Jia et al. (2021). The RO-ViT architecture is shown in Figure 7.

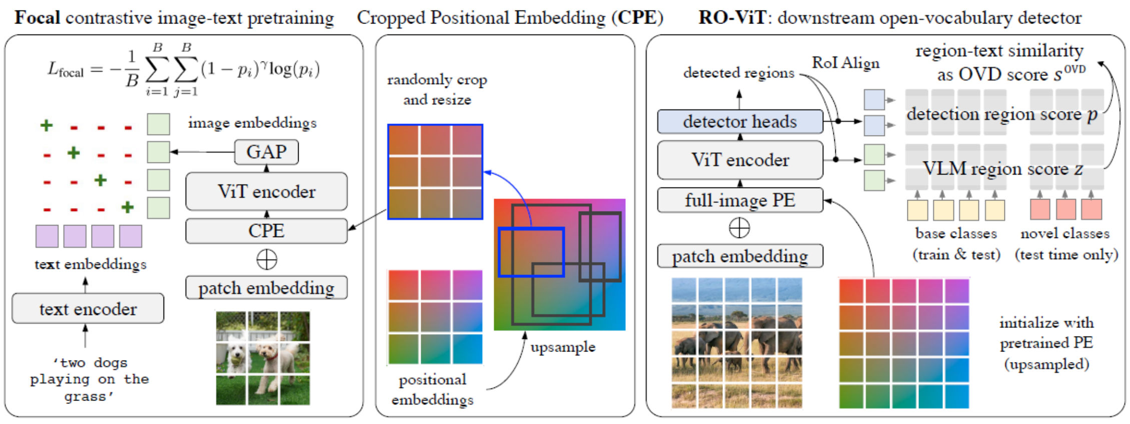

Focal contrastive image-text pretraining
Cropped Positional Embedding (CPE)
RO-ViT: downstream open-vocabulary detector
image embeddings
GAP
ViT encoder
CPE
patch embedding
text embeddings
text encoder
'two dogs playing on the grass'
randomly crop and resize
upsample
positional embeddings
detected regions
RoI Align
region-text similarity as OVD score s^OVD
detector heads
detection region score p
ViT encoder
VLM region score z
full-image PE
patch embedding
base classes (train & test)
novel classes (test time only)
initialize with pretrained PE (upsampled)

Fig. 7. RO-ViT architecture from Kim et al. (2023).

Positional embeddings play a big part on the Transformer's mechanism, as they provide information about the position of each embedding in the input sequence, which is valuable for recognition and localization tasks. However, there is a gap between how the positional embeddigns are used in existing contrastive pre-training approaches and OVD fine-tuning. The RO-ViT tries to bridge the gap between these two tasks, by using Cropped Positional Embeddings (CPEs). A CPE up-samples the positional embeddings from the image size, normally used for pre-training, to the one used for detection, and it then randomly crops and resizes a region from the up-sampled positional embeddings and use it as the image-level positional embeddings during pre-training. This technique helps the model to view an image as a region crop from a larger unknown image, rather than a full image in itself. This can better match the fine-tuning task for detection, where recognition occurs at region-level instead of image-level. The softmax cross entropy loss is replaced by a focal loss that offers a natural option to tune the weights of hard examples. The detection scores p_i computed for each region i, during pre-training, are given by:

$$p_i = \begin{cases} \text{sigmoid}(v_i l_i/\tau) & \text{if } i = j \\ 1 - \text{sigmoid}(v_i l_i/\tau) & \text{if } i \neq j, \end{cases} \tag{45}$$

where v_i and l_i represent the normalized image and text embeddings. These scores are then used to determined the focal loss:

$$\mathcal{L}_{focal} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}(1-p_i)^{\gamma}\log(p_i). \tag{46}$$

At test time, the pre-trained ViT backbone is transferred to the downstream OVD task by replacing the global average pooling with detector heads. The RO-ViT detector takes the image-level positional embeddings as input, and the detections are used to extract the VLM embedding of region i by using RoI-Align on the feature map outputted from the ViT backbone. The VLM region scores z_i are then computed as the cosine similarity between the $C_B \cup C_N$ text embeddings and the region embeddings, identically the detection scores p_i that are now determined with the $C_B \cup C_N$ text embeddings. It is possible to combine both scores, z_i and p_i, to get the open-vocabulary detection score s_i^{OVD}, recurring to an expression similar to Equation 37, that goes as follows:

$$s_i^{OVD} = \begin{cases} z_i^{(1-\alpha)} \cdot p_i^{\alpha} & \text{if } i \in C_B \\ z_i^{(1-\beta)} \cdot p_i^{\beta} & \text{if } i \in C_N. \end{cases} \tag{47}$$

The variables $\alpha, \beta \in [0,1]$ control the weights for the C_B and C_N categories. Knowing that some existing approaches tend to classify novel objects as background, RO-ViT adopts a localization quality-based objectness score, from Kim et al. (2021), instead of a object-or-not binary classification score. It uses a single anchor per location and combines the predicted objectness score o_i with the set of open-voacbulary detection scores determined in Equation 47 to compute the final OVD score:

$$S_i^{OVD} = o_i^{\delta} \cdot s_i^{OVD}. \tag{48}$$

The authors proposed, to use a ViT-B/16 or ViT-L/16 as the image encoder, where the input image size is 224×224, which results in 14×14 positional embeddings with patch size 16×16. The CPE is generated by first interpolating the positional embeddings to size 64×64. The regions are randomly cropped with a scale ratio in [0.1, 1.0], and the aspect ratio in [0.5, 2.0]. The maximum length of the input are 64 tokens. They used a batch size of 4096, the AdamW optimizer with learning rate of 5e-4 and a linear warm up of 10k steps, and trained the model for 500k iterations. For the downstream task, the RO-ViT was trained with base categories C_B for 46.1k/11.3k iterations with image size 1024, large scale jittering, batch size 256/128, the SGD optimizer with weight decay of 1e-4/1e-2, momentum of 0.9, and an initial learning rate of 0.36/0.02 for the LVIS/COCO datasets, achieving its best performance on LVIS. For detection fine-tuning, the authors froze all ViT backbone layers, and only the added detector layers (neck and heads) were trained. This frozen RO-ViT backbone practise improved the baseline by +6.8 AP.

3.6 Referring Image Segmentation

Referring Image Segmentation (RIS) concerns finding the specific region in an image given a natural language text describing the region. The goal is to understand and interpret textual descriptions to accurately identify the objects of interest in the image. This task requires the integration of both visual and linguistic information, where the model must comprehend the language instructions and extract relevant visual cues to perform precise segmentation.

Yu et al. (2023) proposed a new baseline for the zero-shot referring image segmentation task, using a pre-trained CLIP model, that handles the global and local context of an image and an expression in a consistent way. The authors used a mask-guided visual encoder that captures global and local context information of an image mask. They also used a global and local textual encoder, where the global context is captured by a whole sentence of an expression, and the local context is captured by a target noun phrase. By combining features in two different context levels, the model is capable of understanding comprehensive knowledge, as well as a specific trait of the target object. This method outperforms all the other baselines and weakly supervised RIS methods, despite not requiring any additional training over the CLIP model.

In a nutshell, the framework consists on, given an image and an expression as inputs, extracting global and local visual features, using mask proposals, and extracting global and local context textual features. The cosine similarity scores between these features are computed and the mask with the highest score is chosen. The model's framework is represented as in Figure 8.

It is crucial to understand a global relationship between multiple objects in the image alongside a local semantic information of the target. The global context visual features, for each mask proposal, are extracted using a CLIP pre-trained model. However, the visual encoder from CLIP

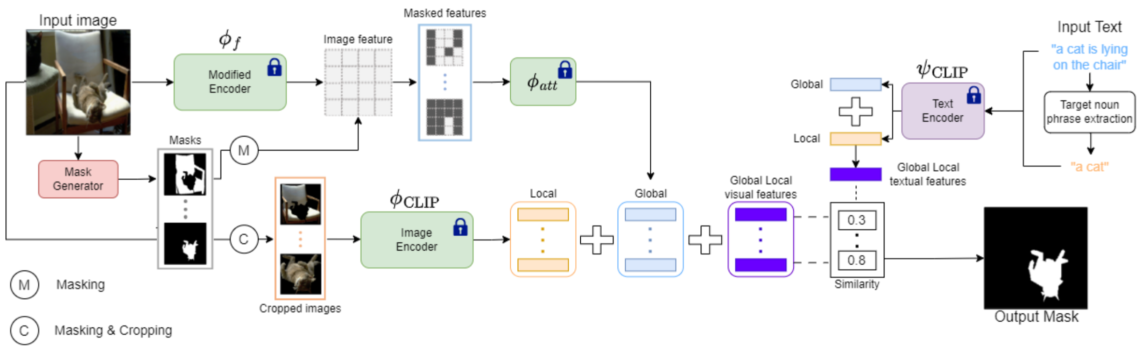


Fig. 8. The referring image segmentation architecture from Yu et al. (2023).

is modified to extract features that contain information from the masked region, as well as the surrounding regions, in order to understand relationships between multiple objects. The authors used two different architectures for the visual encoder, namely a ResNet (He et al., 2016) and a ViT (Dosovitskiy et al., 2020). For the ResNet visual encoder, the authors adopted a visual feature extractor without a pooling layer ϕ_f and its attention pooling layer ϕ_{att}. The visual feature $\mathbf{f}$ can be given by:

$$\mathbf{f} = \phi_{att}(\phi_f(I)), \tag{49}$$

where I represents a given image. As for the ViT visual encoder, since it has multi-head attention layers, its last k layers are denoted by ϕ_{att}, while the remaining layers are represented by ϕ_f. The global context visual feature $\mathbf{f}^G_m$, for a given image I and mask m, can be expressed as follows:

$$\mathbf{f}^G_m = \phi_{att}(\phi_f(I) \odot \bar{m}). \tag{50}$$

In the previous equation, $\bar{m}$ is the resized mask scaled to the size of the feature map, and $\odot$ denotes the Hadamard product operation. As for obtaining the local context visual features, the image is first masked and then cropped to determine a new image, surrounding only an area of the mask proposal. After that, the image goes through the CLIP visual encoder to extract the local context visual feature $\mathbf{f}^L_m$, represented by

$$\mathbf{f}^L_m = \phi_{\text{CLIP}}(\mathcal{T}_{crop}(I \odot m)), \tag{51}$$

where ϕ_{CLIP} represents the CLIP visual encoder, and $\mathcal{T}_{crop}$ represents a cropping operation. This method focuses on the masked region in the image, where the irrelevant regions are removed, allowing the model to concentrate only on the target object itself. It is possible to obtain the combination of the context of global and local visual features $\mathbf{f}_m$ with the following expression:

$$\mathbf{f}_m = \alpha \mathbf{f}^G_m + (1-\alpha)\mathbf{f}^L_m. \tag{52}$$

In this scenario, $\alpha \in [0, 1]$ is a constant, m is a mask proposal, $\mathbf{f}^G$ and $\mathbf{f}^L$ are the global context and local context visual features, computed respectively through Equations 50 and 51.

Despite the visual features, it is also very important to understand the target object noun, as well as the holistic meaning in a given expression. With a referring expression T, the model extracts a global context text feature $\mathbf{t}^G$, using the pre-trained CLIP text encoder ψ_{CLIP}:

$$\mathbf{t}^G = \psi_{\text{CLIP}}(T). \tag{53}$$

Since the input expression of this task is normally formed as a complex sentence containing multiple clauses, the authors exploited dependency parsing using spaCy (Honnibal and Johnson, 2015), to find the target noun phrase NP(T) for a given text expression T. The local context text feature $\mathbf{t}^L$ can be given by:

$$\mathbf{t}^L = \psi_{\text{CLIP}}(NP(T)). \tag{54}$$

The combination of the context of global and local text features $\mathbf{t}$, similarly to the visual features, can be determined by a weighted sum of $\mathbf{t}^G$ and $\mathbf{t}^L$.

$$\mathbf{t} = \beta \mathbf{t}^G + (1-\beta)\mathbf{t}^L. \tag{55}$$

In the previous expression, $\beta \in [0, 1]$ is a constant, $\mathbf{t}^G$ and $\mathbf{t}^L$ are the global context and local context text features, computed respectively through Equations 53 and 54. Finally, given as inputs an image I and a referring expression T, the model determines the score for each mask proposal, by computing the cosine similarity between the visual and textual features, obtained from Equations 52 and 55, and chooses the mask proposal with the highest score.

$$\hat{m} = \underset{m \in M(I)}{\text{argmax}} \cos(\mathbf{t}, \mathbf{f}_m). \tag{56}$$

The model is evaluated on RefCOCOg (Kazemzadeh et al., 2014), RefCOCO, and RefCOCO+ (Nagaraja et al., 2016). The images and masks in the MS COCO dataset (Lin et al., 2015) are used to annote the ground-truth of the RIS task. RefCOCO, RefCOCO+ and RefCOCOg have respectively 19,994, 19,992, and 26,711 images, with 142,210, 141,564, and 104,560 referring expressions, respectively. The metrics used for evaluation were the overall Intersection over Union (oIoU) and the mean Intersection over Union (mIoU), which are the common metrics for the RIS task.

The model uses a previous unsupervised instance segmentation method, namely FreeSOLO (Wang et al., 2022), to obtain mask proposals. The shorter size of an input image is set to 800.For the CLIP model, the size of an image is set to 224×224. The number of masking layers k in ViT is set to 3, α is set to 0.85 for RefCOCOg, 0,95 for RefCOCO and RefCOCO+, and β is set to 0.5 for all datasets. All methods, including baselines, use FreeSOLO mask proposals to produce the final output mask, for a fair comparison. The experimental results show that the RIS model demonstrates superior performance compared to all baselines and has shown a performance improvement of over 3.5%, particularly on RefCOCOg which includes longer expressions.

4 Thesis Proposal

My M.Sc. research project will address the development and evaluation of approaches based on large multi-modal Transformers for addressing the Toloka VQA challenge, advancing in several directions over the previous state-of-the-art. The proposed approaches will use recently proposed for vision-language models, as well as strategies in the realm of data augmentation, in order to generate new (image, question) pairs from existing object detection datasets.

4.1 Research Statement

For this competition, as seen in Ustalov et al. (2023), some of the participants proposed interesting methods that achieved good results, a couple of the teams fine-tuned the pre-trained multi-modal OFA model, which is a task-agnostic and modality-agnostic framework (that supports task comprehensiveness), on the challenge dataset, and additionally used data from pre-processed GQA dataset. However, the team that achieved better results created a variant detector using Uni-Perceiver as the multi-modal backbone network, with ViT-Adapter for cross-modal localization, and DINO as the prediction head, also including an auxiliary loss and a a test-time augmentation module for improved performance.

My research project is going to have in consideration two state-of-the-art methods that I described in this project, namely a framework proposed by Kim et al. (2023) to leverage pre-trained VLMs for open-vocabulary detection fine-tuning, and a simple open-vocabulary object detection method built upon frozen VLMs presented by Kuo et al. (2023).

First, the image and text of encoders of my model are pre-trained with the focal contrastive image-text pre-training modular framework, proposed in Kim et al. (2023), that uses a ViT with CPEs as image encoder and a Transformer as text encoder, as explained in Section 3.5. This method attempts to improve the image-level VLMs pre-training for a downstream open-vocabulary detection task. The focal loss applied in this pre-training process allows both encoders to learn from hard examples, and can be expressed by Equation 46.

At training time, based on the Kuo et al. (2023) object detection framework, the pre-trained image and text encoder backbone features are frozen, and a detector head is attached to the VLM, to predict candidate bounding boxes, there are also computed detection scores which represent the cosine similarity between a region bounded by a candidate box and a certain question, the highest score corresponds to the final prediction. The detector head is the only trainable part of the system during this stage, I'm going to choose YOLO (Terven and Cordova-Esparza, 2023) as a detector head for this system. Since each data instance consists of an image, a textual question, and a ground truth box around the region of the image corresponding to that question, the detector should be trained over a box regression loss, like the IoU loss that measures the similarity between a predicted bounding box and the ground truth bounding box.

At test time, the model uses the region proposals to crop out the top-level features of VLM backbone and compute the VLM score per region, these scores are combined with the detection scores from the trained YOLO detector, through a geometric mean shown in Equation 37. The (box region, question) pair with the highest score is chosen for the final box prediction.

In a nutshell, the model that I am proposing is going to have an architecture that merges components from Figures 6 and 7. The text and vision encoders (Transformer and ViT) are going to be contrastively pre-trained with CPEs according to a focal loss, as shown in Figure 7. The encoders are then frozen and the YOLO detector is going to be trained on the challenge dataset to minimize the detection losses, a batch of N images and N questions are fed to the pre-trained encoders, similarly to Figure 6(a). During inference, the model follows the same pattern as the one illustrated in Figure 6(b).

4.2 Data Description and Evaluation Methodology

This proposed method will be evaluated on the data from the Toloka visual question answering challenge at WSDM Cup 2023 (Ustalov et al., 2023). The dataset is composed by images that are associated with textual questions and, as mentioned before, one entry (instance) in the dataset is a (image, question) pair labeled with the ground truth coordinates of a bounding box containing the visual answer to the given question. Each image contains only one correct response to the given question, and the images of the Toloka dataset are a subset of the MS COCO dataset, (Lin et al., 2015).

The Microsoft Common Objects in Context (MS COCO) dataset is widely used by researchers and practitioners in the field of computer vision, for tasks like object detection, instance segmentation, image captioning, and visual question answering. It has played a significant role in advancing the state-of-the-art in these areas, and serves as a standard benchmark for evaluating the performance of algorithms and models, containing more than 200,000 images labeled with annotations for object bounding boxes, object categories, and segmentation masks.

An annotation campaign, on the Toloka crowdsourcing platform, was done to collect additional annotations over MS COCO. The workers had to select images containing objects that they found subjectively interesting, and then they had to come up with questions about those objects. For each (image, question) pair, they selected the answer on the image using a bounding box, excluding unanswerable questions from the Toloka dataset. For the challenge, it was decided to stick to the more natural data formulations made by real humans, rather than recurring to synthetic data.

The Toloka dataset has a total of 45,199 instances, divided among three subsets. The *train* subset is composed by 38,990 instances, the *public test* by 1,705 instances, and the *private test* by 4,504 instances. The data were provided as CSV files, and each file has eight columns, where the values are separated by commas and can be described in Table 1.

Column	Type	Description
image	string	URL of an image on a public content delivery network
width	integer	image width
height	integer	image height
left	integer	bounding box coordinate: left
top	integer	bounding box coordinate: top
right	integer	bounding box coordinate: right
bottom	integer	bounding box coordinate: bottom
question	string	question in English

Table 1. characterization for the fields in the Tokola dataset

Since the images used in the challenge dataset come from MS COCO, the proponents of the task have carefully checked the overlap between bounding boxes in both datasets. About 20% of bounding boxes had non-empty overlap, all those instances were put into the *train* subset.

Having in consideration that the answer, for this type of task, comes with predicting bounding box coordinates, there are some metrics that can be used to evaluate the performance of bounding box matching. The intersection over union (IoU) is a metric used to capture the overlap or similarity between predicted and ground truth bounding boxes, given by:

$$\text{IoU} = \frac{I}{U}. \tag{57}$$

In the previous expression I is the intersection between the ground truth bounding box area and the predicted bounding box area, and U is the area of union of these boxes. For object detection tasks, one typically considers a threshold value for the IoU. If a prediction has a IoU value bigger than the threshold, it is considered a True Positive (TP), which stands for a correct prediction, and otherwise it is considered a False Positive (FP), that corresponds to an incorrect prediction. By being able to classify each prediction, based on its IoU value, it is possible to calculate the precision of the model, which measures how many of the predictions made were actually correct. This metric is computed by:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{58}$$

Another important metric is the recall, which measures how well the model can of find all the existing objects, since there can be some cases, i.e. False Negatives (FN), where the model fails to detect an object that was marked in the image with a ground truth box.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{59}$$

The precision and recall can be used to compute the Average Precision (AP), which is a summary of the precision-recall curve. Still, the most commonly used metric for object detection challenges is called the mean Average Precision (mAP). This is simply the mean of the Average Precisions computed over all the existing classes. The mAP metric avoids to have extreme specialization in few classes, and thus a weak performance in others. Object detection methods like Faster R-CNN and YOLO used this metric to evaluate their models.

The challenge had 48 overall participants in the competition, 9 of whom submitted their code during the reproduction stage. Table 2 shows the competition results, where the IoU values were multiplied by 100 for visual convenience. The participants had access to two baselines to compare their models with, one being the zero-shot YOLOR + CLIP baseline from the starter kit available since the beginning of the challenge, the other a crowdsourcing baseline that no team was able to outperform. The competition had three key phases: the practice phase, to let the contestants get used to the task and training data. The evaluation phase, when the *public test* dataset was released, for the teams to evaluate their models. The reproduction phase, where the participants

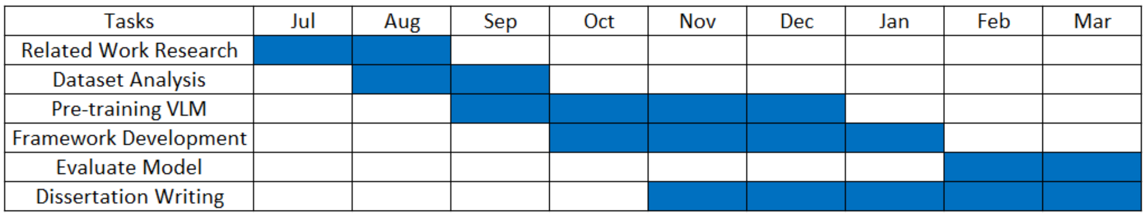
Rank	**Participants**	**IoU**
-	Crowdsourcing Baseline	87.154
1	wztxy89	76.347
2	jinx, Zhouyang_Chi	76.342
3	komleva.ep	75.591
4	xexanoth	74.667
5	Man_of_the_year	72.768
6	Haoyu_Zhang, KhylonWong	71.998
7	nika-li	70.525
8	blinoff	62.037
9	Ndhuynh	61.247
-	YOLOR + CLIP Baseline	21.292

Table 2. Baselines and final team standings on the *private test* subset

submitted their final solutions, so that the Tokola organizing committee was able to determine the winners.

5 Conclusions and Planning for Next Semester

The present document marks the beginning of my M.Sc. research project, which will be further developed in the upcoming semester. Within these pages, I explained into the fundamental principles of deep learning, and explored relevant research that will form the foundation for methods that are to be developed. Additionally, the report encompasses a description of the proposed research, accompanied by an explanation of the evaluation methodology that will be used. Figure 9 shows a calendarization for the development of this research project.



Tasks	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar
Related Work Research	🟦	🟦							
Dataset Analysis		🟦	🟦						
Pre-training VLM			🟦	🟦	🟦	🟦			
Framework Development				🟦	🟦	🟦	🟦		
Evaluate Model								🟦	🟦
Dissertation Writing					🟦	🟦	🟦	🟦	🟦

Fig. 9. Work Schedule.

Bibliography

Frank Rosenblatt. *The perceptron: a probabilistic model for information storage and organization in the brain.* American Psychological Association, 1957.

Marvin Minsky and Seymour Papert. *Perceptrons.* MIT press, 1969.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Proceedings of the Annual Meeting on Neural Information Processing Systems*, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Emanuele Bugliarello, Ryan Cotterell, Naoaki Okazaki, and Desmond Elliott. Multimodal pre-training unmasked: A meta-analysis and a unified framework of vision-and-language berts. *arXiv preprint arXiv:2011.15124*, 2021.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *Proceedings of the International Conference on Machine Learning*, 2021.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv preprint arXiv:1311.2524*, 2014.

Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 2013.

Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2015.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. *arXiv preprint arXiv:1703.06870*, 2017.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2016.

Juan Terven and Diana Cordova-Esparza. A comprehensive review of yolo: From yolov1 and beyond. *arXiv preprint arXiv:2304.00501*, 2023.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *arXiv preprint arXiv:2005.12872*, 2020.

Yanyuan Qiao, Chaorui Deng, and Qi Wu. Referring expression comprehension: A survey of methods and datasets. *IEEE Transactions on Multimedia*, 2020.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. Generation and comprehension of unambiguous object descriptions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016.

Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. Modeling context in referring expressions. In *Proceedings of the European Conference on Computer Vision*, 2016.

Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. Natural language object retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016.

Bohan Zhuang, Qi Wu, Chunhua Shen, Ian Reid, and Anton Van Den Hengel. Parallel attention: A unified framework for visual object discovery through dialogs and queries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

Chaorui Deng, Qi Wu, Qingyao Wu, Fuyuan Hu, Fan Lyu, and Mingkui Tan. Visual grounding via accumulated attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

Ronghang Hu, Marcus Rohrbach, Jacob Andreas, Trevor Darrell, and Kate Saenko. Modeling relationships in referential expressions with compositional modular networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017.

Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L Berg. Mattnet: Modular attention network for referring expression comprehension. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2018.

Peng Wang, Qi Wu, Jiewei Cao, Chunhua Shen, Lianli Gao, and Anton van den Hengel. Neighbourhood watch: Referring expression comprehension via language-guided graph attention networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Sibei Yang, Guanbin Li, and Yizhou Yu. Dynamic graph attention for referring expression comprehension. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

Volkan Cirik, Taylor Berg-Kirkpatrick, and Louis-Philippe Morency. Using syntax to ground referring expressions in natural images. *arXiv preprint arXiv:1805.10547*, 2018.

Daqing Liu, Hanwang Zhang, Feng Wu, and Zheng-Jun Zha. Learning to assemble neural module tree networks for visual grounding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Proceedings of the Annual Meeting on Neural Information Processing Systems*, 2019.

Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-BERT: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019.

Arka Sadhu, Kan Chen, and Ram Nevatia. Zero-shot grounding of objects from natural language queries. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

Gen Luo, Yiyi Zhou, Xiaoshuai Sun, Liujuan Cao, Chenglin Wu, Cheng Deng, and Rongrong Ji. Multi-task collaborative network for joint referring expression comprehension and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Kan Chen, Jiyang Gao, and Ram Nevatia. Knowledge aided consistency for weakly supervised phrase grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

Fang Zhao, Jianshu Li, Jian Zhao, and Jiashi Feng. Weakly supervised phrase localization with multi-scale anchored transformer network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. ReferItGame: Referring to objects in photographs of natural scenes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2014.

Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2015.

Zhenfang Chen, Peng Wang, Lin Ma, Kwan-Yee K. Wong, and Qi Wu. Cops-ref: A new dataset and task on compositional referring expression comprehension. *arXiv preprint arXiv:2003.00403*, 2020.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common objects in context. *arXiv preprint arXiv:1405.0312*, 2015.

Weicheng Kuo, Yin Cui, Xiuye Gu, AJ Piergiovanni, and Anelia Angelova. F-VLM: Open-vocabulary object detection upon frozen vision and language models. *arXiv preprint arXiv:2209.15639*, 2023.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017.

Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D. Cubuk, Quoc V. Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. *arXiv preprint arXiv:2012.07177*, 2021.

Xiaoshi Wu, Feng Zhu, Rui Zhao, and Hongsheng Li. CORA: Adapting CLIP for open-vocabulary detection with region prompting and anchor pre-matching. *arXiv preprint arXiv:2303.13076*, 2023.

Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang. Dab-DETR: Dynamic anchor boxes are better queries for detr. *arXiv preprint arXiv:2201.12329*, 2022.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2018.

Hongyu Zhai, Jian Cheng, and Mengyong Wang. Rethink the iou-based loss functions for bounding box regression. In *Proceedings of the IEEE/CVF International Conference on Information Technology and Artificial Intelligence*, 2020.

Yiwu Zhong, Jianwei Yang, Pengchuan Zhang, Chunyuan Li, Noel Codella, Liunian Harold Li, Luowei Zhou, Xiyang Dai, Lu Yuan, Yin Li, and Jianfeng Gao. RegionCLIP: Region-based language-image pretraining. *arXiv preprint arXiv:2112.09106*, 2021.

Dahun Kim, Anelia Angelova, and Weicheng Kuo. Region-aware pretraining for open-vocabulary object detection with vision transformers. *arXiv preprint arXiv:2305.07011*, 2023.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yunhsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. *arXiv preprint arXiv:2102.05918*, 2021.

Dahun Kim, Tsung-Yi Lin, Anelia Angelova, In So Kweon, and Weicheng Kuo. Learning open-world object proposals without learning to classify. *arXiv preprint arXiv:2108.06753*, 2021.

Seonghoon Yu, Paul Hongsuck Seo, and Jeany Son. Zero-shot referring image segmentation with global-local context features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016.

Matthew Honnibal and Mark Johnson. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2015.

Varun K Nagaraja, Vlad I Morariu, and Larry S Davis. Modeling context between objects for referring expression understanding. In *Proceedings of the European Conference on Computer Vision*, 2016.

Xinlong Wang, Zhiding Yu, Shalini De Mello, Jan Kautz, Anima Anandkumar, Chunhua Shen, and Jose M Alvarez. Freesolo: Learning to segment objects without annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Dmitry Ustalov, Nikita Pavlichenko, Daniil Likhobaba, and Alisa Smirnova. WSDM Cup 2023 Challenge on Visual Question Answering. In *Proceedings of the Crowd Science Workshop on Collaboration of Humans and Learning Algorithms for Data Labeling*, 2023.