Deep Learning Assignment 2023
Visual Grounding

March 22, 2023

1 Introduction

This assignment focuses on **visual grounding**, which involves linking language and perception by grounding linguistic symbols in the visual world. You will be provided with a visual grounding benchmark consisting of a dataset containing pairs of images and textual descriptions. Your task will be to build, fine-tune, and evaluate a deep learning framework that can learn to perform visual grounding on this dataset. Specifically, you will train your model to predict a set of bounding boxes in each image that correspond to the entities referred to in the textual description.

To complete this task, you will need to use the **CLIP** [2] (Contrastive Language-Image Pre-training) model as a foundation model for your framework. This is beneficial for two reasons. First, pre-trained models such as CLIP offer a starting point for transfer learning, thereby reducing the amount of training resources needed for the task. Second, foundation models like CLIP are intended to provide a wide range of capabilities out of the box, even for tasks for which they have no prior experience. Using CLIP as a foundation model means you will be starting with a neural network that has been trained on a large dataset of image-text pairs, allowing you to leverage its pre-existing strengths. Your objective is to fine-tune the model for the specific task of visual grounding, which is not one of its original purposes. This highlights the principle of transfer learning, which involves using a pre-trained model to build a customized one that excels in a specific task.

To fine-tune CLIP for visual grounding, you may use any relevant technique or algorithm from the literature, such as attention mechanisms, convolutional neural networks (CNNs), or recurrent neural networks (RNNs), among others. The implementation will be evaluated using metrics such as localization accuracy, grounding accuracy, and semantic similarity.

2 Dataset

The visual grounding task in this assignment will employ the **RefCOCOg** dataset [3], a variant of the Referring Expression Generation (REG) dataset. The RefCOCOg dataset consists of approximately 25,799 images, each of which contains an average of 3.7 referring expressions. As with RefCOCO+, location words are not allowed in the referring expressions, which contains only appearance-based descriptions that are independent of viewer perspective. This makes the dataset well-suited for visual grounding tasks, as it necessitates creating a mapping between the visual appearance of an object and its corresponding linguistic label.

An essential component of the visual grounding task is generating accurate bounding boxes precisely around each referred object in the image. Your model should learn to locate and label each object described in the referring expressions, taking into account the surrounding context and visual properties of the image.

To access and download the dataset, you can use the Google Drive link provided [1]. Please note that in the annotations folder, there are two available refer files in the pickle format (`ref(google).p` and `ref(umd).p`). For this exercise, we will use the second split. As the zipped dataset is hosted on Google Drive, it is advisable to add the shortcut to the Drive folder, allowing easy access from Colab without downloading files to the local storage.

During the assignment, you will be expected to train (on the train set) and evaluate your visual grounding framework (on the test set), and provide an interpretation of the achieved results. The complete dataset

[1] `https://drive.google.com/uc?id=1xijq32XfEm6FPhUb7RsZYWHc2UuwVkiq`

comprises roughly 80,000 instances for training; however, it is not necessary to use all of them. You are, however, required to test your model on the entirety of the test set. Additionally, there is no predefined torchvision dataset class appropriate for the visual grounding task, requiring you to create a custom dataset class to load and read the dataset correctly.

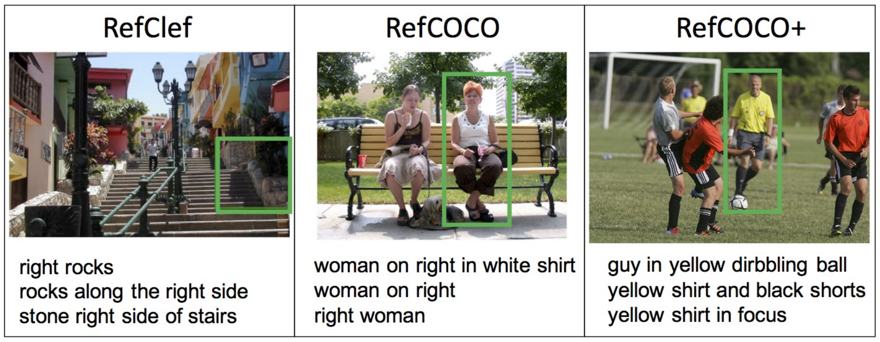
RefClef
RefCOCO
RefCOCO+
right rocks
rocks along the right side
stone right side of stairs
woman on right in white shirt
woman on right
right woman
guy in yellow dirbbling ball
yellow shirt and black shorts
yellow shirt in focus

Figure 1: RefCOCO Dataset

3 Task

Visual grounding is a task that connects natural language expressions with perception by identifying visual objects within an image corresponding to a given language expression. In this assignment, we will use the RefCOCOg dataset and fine-tune the CLIP model to train a visual grounding framework. The model seeks to predict a set of bounding boxes that match the object(s) referred to in a given natural language expression by learning a mapping between the visual and textual feature vectors of the images using CLIP model as a foundation. Unless justified by design choices, you are constrained to use the ResNet50 version of CLIP.

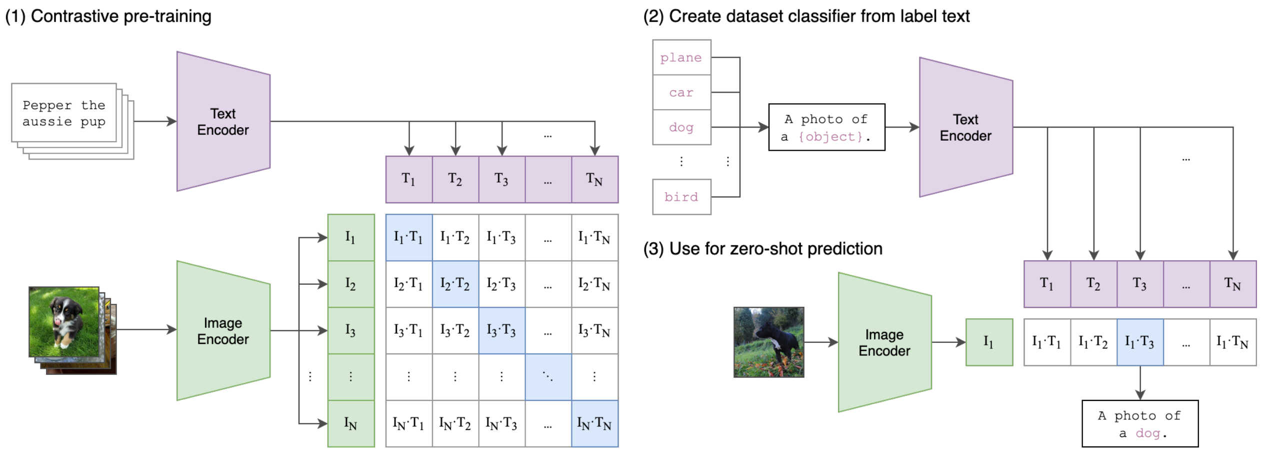
(1) Contrastive pre-training
Pepper the aussie pup
Text Encoder
Image Encoder
T1 T2 T3 ... TN
I1 I2 I3 ... IN
I1·T1 I1·T2 I1·T3 ... I1·TN
I2·T1 I2·T2 I2·T3 ... I2·TN
I3·T1 I3·T2 I3·T3 ... I3·TN
IN·T1 IN·T2 IN·T3 ... IN·TN
(2) Create dataset classifier from label text
plane
car
dog
bird
A photo of a {object}.
Text Encoder
(3) Use for zero-shot prediction
Image Encoder
I1
T1 T2 T3 ... TN
I1·T1 I1·T2 I1·T3 ... I1·TN
A photo of a dog.

Figure 2: The CLIP framework was designed to enable simultaneous processing of images and text, enabling it to perform various tasks, including image classification. On one side of the framework, it utilizes a training methodology that involves maximizing the similarity score between corresponding image-text pairs. On the other hand, it leverages class names to create a customized classifier capable of performing zero-shot image classification.

The CLIP model is a foundation machine learning algorithm that achieves joint understanding of both images and text, enabling the model to perform tasks like image recognition and classification. CLIP can analyze images and texts simultaneously by learning to map visual and linguistic information to a unified space. These visual and textual mappings of the same image or text are utilized to learn the association between them by maximizing the similarity score during training. In this way, CLIP model can perform tasks such as image classification with great accuracy even for classes the model has never seen before. However, for tasks like visual grounding and object detection, the model requires additional fine-tuning or additional components as it necessitates identifying specific objects within an image and linking them to textual descriptions to accomplish the task.

To extend CLIP for the visual grounding task, some possible steps to follow are:

- Read in the RefCOCOg dataset and preprocess it into appropriate data structures for fine-tuning your model.
- Fine-tune the CLIP model on the RefCOCOg dataset to learn a mapping between visual and textual feature vectors.
- Define your custom visual grounding neural network architecture that incorporates CLIP as its foundation, allowing you to input the textual description and obtain predictions of bounding boxes.
- Train this model on the RefCOCOg dataset, and evaluate it on the validation set by computing standard evaluation metrics.
- Analyze the model's performance and provide insights into its strengths and weaknesses, in addition to suggesting potential avenues for further research.

It is important to note that the steps outlined above are not comprehensive, and there are other crucial factors to consider when implementing a visual grounding framework, such as regularization techniques, data augmentation, and hyperparameter tuning. Careful consideration must be given to the fine-tuning process of the CLIP model, as extreme approaches may not always be beneficial. It is advisable to avoid fine-tuning the entire network and instead adapt only specific layers or incorporate additional MLPs. Adopting a thoughtful approach to these considerations will lead to a more effective framework.

When evaluating a visual grounding framework, it is essential to consider appropriate metrics to ensure that the model is performing as intended. Some good metrics to consider include localization accuracy, grounding accuracy, and semantic similarity. Localization accuracy measures how accurately the system can localize an object in the image, while grounding accuracy measures how accurately it can ground the localized object to a language description. Semantic similarity measures the similarity between the predicted bounding boxes and the ground-truth descriptions. Intersection over Union (IoU) is a common measure of localization accuracy, while recall is used to measure grounding accuracy. Cosine similarity and Euclidean distance are commonly used to measure semantic similarity. Evaluating the model using these metrics can provide valuable insights into the model's performance and areas for improvement.

3.1 Baseline

To start the project, it is recommended to implement a baseline method. One suggested method is a training-free approach that combines CLIP zero-shot with a YOLO architecture [1]. This approach involves extracting all the bounding boxes proposed by YOLO and evaluating their similarity with the textual query with CLIP.

You can easily access the YOLO model on TorchHub[2], making it a convenient starting point for your project. By implementing this baseline method, you can familiarize yourself with the task of visual grounding, the dataset being used, and the CLIP model. Additionally, it may serve as a foundation from which you can build and improve upon as you continue to develop your model.

4 Evaluation

Your delivery will be **self-contained within a single Jupyter Notebook** hosted on Google Colab. The notebook will contain your code, properly divided into multiple executable cells in a clean, modularized and readable fashion. In addition, you are required to exploit the text cells provided in Google Colab in order to integrate comments into your notebook, which should come together as an actual project report. In the report you are required to provide:

- a description of the **solution** that you adopted. This must include a detailed overview of the developed architecture, as well as the losses that govern the training. Please make sure to highlight all your original contributions as well as to correctly cite the literature. Pay attention to properly motivating your design choices and all the components you decide to include, especially when it is not obvious to

[2]https://pytorch.org/hub/ultralytics_yolov5/

simply infer it. Keep in mind that the text cells also allow you to include pictures: diagrams and charts are always very helpful!

- a description of the **details** of the training and evaluation strategy employed. Provide a detailed overview of how your model was built and trained, extensively motivating your methodological choices with respect to hyperparameters choice and other variables
- an extensive presentation and **discussion** of the results obtained with your framework. Organize the scores you obtained in tables and consider including charts depicting learning curves, confusion matrices and any other useful visual representation. Make an effort in order to structure this section in such as a way as to emphasize the take-home message deriving from your findings
- **comments** to the code. It is important to employ the utility of the Jupyter notebook in order to provide a step-by-step description of your code. Try to be clear but concise, proving that you are aware of all the steps that you go through during the development and the training/evaluation pipeline
- **instructions** for correctly running your code, if needed

Your project will be evaluated according to the following metrics:

- originality of the solution
- methodological thoroughness
- clarity of the report
- performance with respect to validation accuracy
- quality of the code

In order to deliver the solution, send an **e-mail** to alessandro.conti-1@unitn.it, putting e.ricci@unitn.it as cc. The email should have object `[DL2023] - Assignment delivery`. Attach to the email your self-contained notebook in `.ipynb` format, with name `[student_id_1]_[student_id_2].ipynb`. For example, the delivery of students with IDs (matricola) 123456 and 987654 will be `123456_987654.ipynb`. **IMPORTANT**: make sure that the file can be seamlessly loaded and executed into Google Colab. The solution must be delivered strictly **not later than 7 days before the date in which you decide to take the oral exam**.

5 Group registration

The project is intended for groups of 2 or 3 people. You can register your group by filling the Google Form[3], strictly within the end of April.

6 Policies

We require that you **don't start from an existing GitHub repository** containing code developed by other people: your architecture doesn't have to be overly complex, but you are expected to either build it from scratch or to start from the code used in the lab sessions. You can, of course, install the CLIP repository with pip or conda[4]. If needed, only small specific snippets of code may be borrowed from existing GitHub repositories. This policy derives from the purpose of having you practically familiarize with the tools involved in building a deep learning framework rather than just assemble existing components. We should specify that this does not refer to the choice of a **feature extractor** for your model: you are allowed to choose any existing architecture (e.g. ResNet) - possibly with pretrained weights, if available - as a model *backbone*. You may generically discuss the project with other groups, but it is strictly forbidden to share the source code. Keep also in mind that your delivery may be checked against an automatic utility for plagiarism detection. Any violation of these policies will result in actions being taken towards all involved groups, including those potentially lending their code: it is the responsibility of each group to comply to the policy.

[3]**Google form**: `https://forms.gle/EPwLTDiJNsv2yGY1A`

[4]`pip install ftfy regex tqdm git+https://github.com/openai/CLIP.git`

References

[1] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768, 2018.

[2] 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 *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[3] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. Modeling context in referring expressions. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 69–85. Springer, 2016.