CLIP - Contrastive Language-Image Pretraining

Link: https://arxiv.org/pdf/2203.07190.pdf

Useful resources:

https://huggingface.co/alan-turing-institute/mt5-large-finetuned-mnli-xtreme-xnli

Abbrevations

VLU - Visual Language Understanding

VQA - Visual Question Answering

TAP-C: Template Answer Prompt then CLIP method

NLI: Natural Language Inference

Highlights of the paper

—> CLIP can be a strong vision-language few-shot learner by leveraging the power of language.

—> Evaluate CLIP’s zero-shot performance on a typical visual question answering task and demonstrate a zero-shot cross-modality transfer capability of CLIP on the visual entailment task.

—> Proposes a parameter-efficient fine-tuning strategy to boost the few-shot performance on
the vqa task.

Detailed Summary

Visual entailment tasks require the understanding of both the visual world and the language world. Previous works related to this involve human annotated datasets that

are expensive or require expert knowledge. Hence, the data available for visual entailment is nothing as compared to the text corpora

CLIP consists of a visual encoder and a text encoder and it learns visual representations by aligning images and texts through contrastive loss

There are two major differences between CLIP and previous visual encoders:

1) it is trained on much larger yet noisy web data, hence promises the generalization ability of CLIP

2) it has a shallow interaction between vision and language, equips alignment ability across modalities.

To help CLIP answer questions about images, the authors use a two-part method. First, they change the questions into statements that can have parts of them "filled in". Then, they use another model called T5 to fill in those parts and get possible answers.

Instead of training with images, the authors train with textual descriptions of those images. Only a small part of the system (a layer) is changed or "updated" when they train with text descriptions. Even though they trained with text descriptions, they test the system's understanding using both the original images and their descriptions.

Found that optimizing only bias and normalization (BiNor) parameters would make better use of
limited examples and yield better results than the latest few-shot model Frozen.

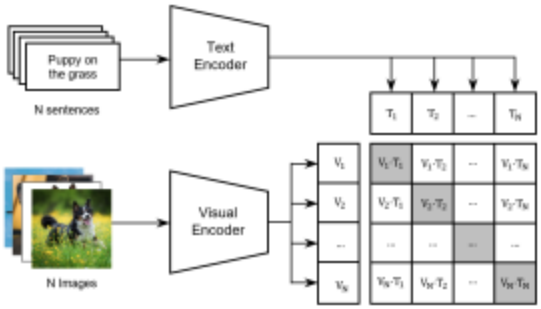

Figure 2: CLIP consists of a visual encoder $\mathbb{V}$, a text encoder $\mathbb{T}$, and a dot product between their outputs. It is trained to align images and texts with a contrastive loss. The dot product is used as an alignment score.

CLIP stands for Contrastive Language-Image Pretraining. It consists of a visual encoder(V) and a text encoder (T). The output from both these encoders is multiplied and the result is used as an alignment score between the input image and the text. Because of the pretraining it distinguishes aligned image-text pairs from randomly combined ones by a contrastive loss.

The task of VQA requires the model to answer questions about the details of the input image. The dataset used for experiment is VQAv2. This dataset is essentially a big set of pictures and related questions. The pictures come from another collection called Microsoft COCO. In this VQAv2 dataset, there are specific types of questions they ask about the pictures. Some examples of these questions are "how many?" (like how many cats are in the picture) and "what color is?" (like what color is the shirt). When it comes to answering, there are three main categories of responses the model can give:

1. "Yes" or "No"
2. A number (like "3")
3. Any other kind of answer (like "blue" or "apple").

Natural Language Inference (NLI): Imagine two sentences. NLI is like a game where you decide the relationship between these sentences. For example, if Sentence A is "All

dogs bark" and Sentence B is "Some dogs make noise", then Sentence A implies Sentence B, so the relationship is "entailment".

Visual Entailment is similar to the above game, but instead of two sentences, you have one image and one sentence. For instance, if the image shows a red apple and the sentence says "The fruit is red", the relationship is "entailment". But if the sentence says "The fruit is blue", then it's a "contradiction".

Researchers took a well-known dataset called SNLI, which is all about sentence relationships as described in NLI. Then, they modified it to create SNLI-VE by replacing one of the sentences with an image from another dataset called Flickr30k.

The researchers are testing how good CLIP is at a game where it needs to figure out the relationship between an image and a sentence. They're using a modified dataset where one of the sentences is replaced with an image. This is to see how well CLIP understands both images and text together. the researchers are testing the zero shot capability of CLIP i.e. checking if CLIP can play the visual entailment game without being trained for it specifically.

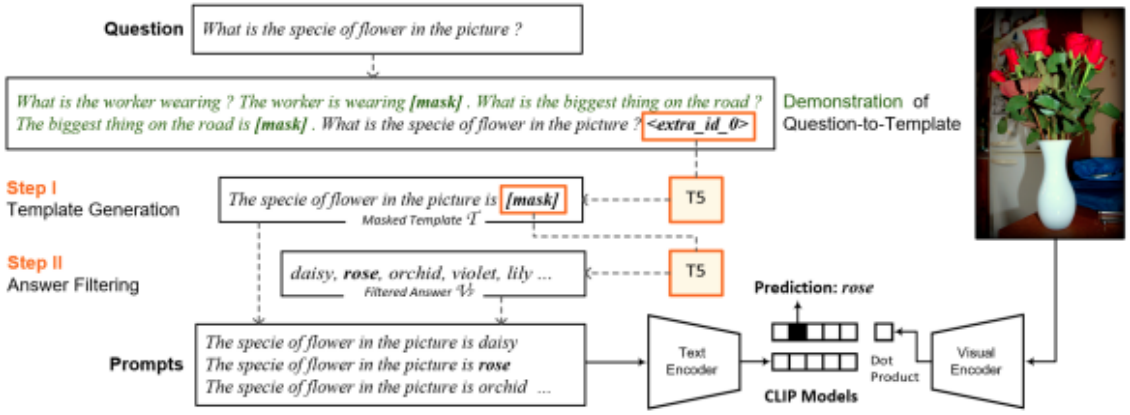

Figure 3: The overall framework of the proposed TAP-C method for zero-shot VQA. TAP-C first generates a masked template from the question by demonstrating examples to T5 and then filters out impossible answers according to the language model. Infilling the masked template with selected answers results in prompts, which could be paired with images to calculate image-text alignment scores by the CLIP. The dashed line denotes the process of prompts generation (§ 3.1), and the solid line denotes prompting CLIP to conduct zero-shot VQA (§ 3.2).

Some previous studies have tried using CLIP directly for tasks that involve understanding both images and text without specific training (that's the "zero-shot" part). But this approach didn't work well. When they tried using CLIP directly on a dataset where the task is to answer questions about pictures, the results were almost as bad as just guessing randomly.

The researchers came up with a new method to help CLIP understand and answer questions about images. They used another model called T5 to assist in this. They use a two step process -

1. Change the question into a special format (they call it a "masked template").
2. Use a language model (like T5) to help remove answers that don't make sense and come up with a list of possible good answers i.e. generate the list of candidate answers.

The infilled template (filling possible answers from step-2 into the template step-1) makes it easier for CLIP to understand both the question and the possible answers together.

STEP-1: Automatic Template Generation

This step is designed to convert the question into a template, which is a statement with a mask token.

Example: **Question:** What is the specie of flower in the picture? **Template:** The specie of flower in the picture is **<mask>**

Two way to implement this -

A. **Demonstration to T5:** The researchers introduced a straightforward way to turn questions into a template. This involves changing parts of a question with placeholders, such as "[mask]". They show the model several examples of this conversion for different types of questions, and the model learns to recognize the patterns and do the conversion itself. An illustration in "Figure 3" shows how this conversion works. To achieve this transformation, the researchers use a model called T5. T5 is designed to fill in missing parts of a sentence. So, by showing T5 a combination of their examples, the original question, and a special placeholder, T5 generates the desired format, which they call "T_demo”

B. **Dependency Parsing:** The researchers noticed that while the T5 conversion method was effective in many cases, it didn't always work perfectly. In situations where T5 struggled, they decided to use a more traditional approach called "dependency parsing." In this method, they dissected a question based on its grammatical structure, identifying key parts like "wh-words" (e.g., what, where, when), main verbs, and related words. They then rearranged and transformed the question into a statement format using established grammar rules. To help with this grammatical analysis, they utilized a tool called "Stanza", which can label words with their roles in a sentence (like noun, verb, adjective) and determine their relationships. Once the question was transformed into a statement, the answer's place was marked with a placeholder or "mask" token. The specific transformation rules they employed came from another study by Demszky and colleagues in 2018. The end result, or the transformed statement, was termed "Tparsing." Essentially, it's a more grammar-based approach to changing questions into a format that might be easier for models like CLIP to understand.

STEP-2: Answer filtering

Imagine you have a question: "What is the specie of the flower?" and one of the potential answers is "vase". We know that a vase can't be a specie of a flower. So, using machines that have been trained on lots of text (and therefore have a sense of common knowledge), we can filter out such unlikely answers. What the researchers in this paper do is they use the previous generated templates and use a model like T5 to guess what word might fit in the place of [mask] based on the rest of the sentence. They rank the potential answers based on how likely the machine thinks they fit into the masked template. The machine gives a score for each possible answer, and they pick the top 'k' scores. These top answers are the ones that make the most sense to the machine. Now that they have a list of good answers (let's say "rose", "tulip", "daisy" for our example), they put each of these answers in the place of [mask] in the template to create prompts. So, "The specie of the flower is rose" would be one prompt, "The specie of the flower is tulip" would be another, and so on.

So after these two steps, we have generated a set of prompts from one question.

TAP-C method for VQA

Given an image i and the generated prompts P, the TAP-C method can get a zero-shot VQA

prediction by:

$$max(V(i).T(p_v))$$

where V and T are visual and text encoders in the CLIP model.

Zero-shot Cross-modality Transfer

Can the CLIP models understand and relate information across two different types of data, namely text (language) and images (vision)?

Training Phase

Step-1: The premise is given to text encoder to produce v_1 and the hypothesis text is given to the text encoder to produce v_2

For example, if your premise is "The sky is blue" and the hypothesis is "It is daytime", both of these sentences are individually fed into the CLIP's text encoder. The encoder transforms each of these sentences into corresponding vectors, v_1 for premise and v_2 for hypothesis.

Step-2: The researchers then use fusion which is a process that combines two vectors (representations of the text inputs) in a way that captures their individual features and their interactions.

Given two vectors v_1(representing the premise) and v_2 (representing the hypothesis):

1. **Concatenation ([*v*1,*v*2]):** This step places one vector right after the other. It ensures that the individual features of both vectors are preserved.
2. **Addition ([*v*1+*v*2]):** This captures common features between the two vectors. If both vectors have high values in specific positions, the addition will accentuate that.
3. **Subtraction([*v*1-*v*2]):** This helps in identifying differences between the vectors. If a feature is strong in v_1 but weak or absent in v_2 this operation will highlight it.
4. **Element-wise multiplication([*v*1.*v*2]):** This captures interactions between the two vectors. It will give high values in positions where both vectors have high values.

Step-3: We give these fused vectors as an input to the MLP that carries a lot of information — not just about the individual texts represented by v1 and v2, but also about their interactions and differences. The MLP uses all this information to discern the relationship between the premise and hypothesis.

The aim of the fusion and the subsequent training of the MLP is to create a system that deeply understands the relationship between two pieces of information (the premise and hypothesis). During training, both of these are textual. But the ultimate goal is to challenge the system during testing by replacing the textual premise with an image. If the system can still accurately predict the relationship, then it shows that the model has effectively learned to bridge the gap between text and vision, achieving cross-modality understanding.

Testing Phase

- For testing how well the model understands, they change things up: instead of feeding it a text premise, they give it an image. This image is processed by CLIP's image-handling part (visual encoder).
- They again create a mixed representation, but this time, it's a fusion of the image data and the textual hypothesis.
- Using the previously trained small model (MLP classifier), they now predict the relationship based on this new fusion of image and text.

Terminologies

Number of ways: This refers to the distinct classes or categories in a task. They have 65 question types and 3 answer types, leading to

65×3=195 different combinations or "ways". This approach ensures the model isn't just memorizing specific answers but is actually generalizing across different types of questions and answers.

Number of shots: This term describes how many distinct examples are available for each "way". In this context, a "shot" consists of an image plus its associated question and answer.

Support set and Query set: The researchers create a subset of the VQAv2 training dataset containing 195 different ways, with *K* examples (or "shots") for each way. So, there are a total of 195×K examples.

During each training cycle (or epoch), they select *C* ways out of the 195. For each of these *C* ways, they split the *K* shots into two groups:

Support Set: This set is used for the actual training of the model.

Query Set: This set functions as a test or evaluation set, allowing the researchers to gauge the model's performance.

The researchers aim to enhance CLIP's visual question answering (VQA) abilities using few-shot learning. Instead of retraining the vast CLIP model entirely, they smartly target only a tiny subset of its parameters. Specifically, they focus on the bias and certain normalization parameters. These parameters are then fine-tuned using a common loss function applied to image-question pairs. In addition, to further refine the model's answering ability, they introduce "in-context demonstrations". This technique uses already-filled templates from the available limited examples to better predict probable answers for a type of question. By combining traditional few-shot learning and this in-context demonstration technique, the researchers aim to optimize CLIP's VQA performance, even when very few examples are available.