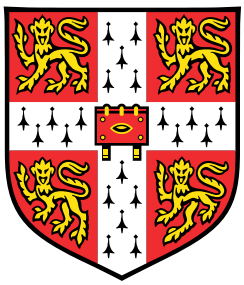
Vision Encoders in Visual Question Answering



Ryan Anderson

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy

Magdalene College

August 2022

Dedicated to my Mom, my brothers, and my late Dad.

Declaration

I, Ryan Stephen Anderson of Magdalene College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose. All software used for this project has been written by Weizhe Lin (the overall design) and me (the specific implementation required for this project) using Python and PyTorch, and the code is available here. This dissertation contains 14056 words including appendices.

Ryan Anderson
August 2022

Acknowledgements

As this report reflects the product of my entire time in Cambridge, I will acknowledge those who have assisted me during and before this project.

Firstly, I would like to thank my supervisor, Professor Bill Byrne, who gently guided me through the initial uncertainty of formulating a direction for the project, and provided invaluable insights into how to formalise my ideas into a coherent report. His unwavering commitment to all of the students that he supervises, and his passion for the MLMI program, is commendable.

Secondly, I owe a great deal of gratitude to my co-supervisor, Weizhe Lin, who selflessly spent a significant amount of time setting up a well-written and designed code-base which served as the starting point for rapid experimentation. Without this repository, and his patient guidance in the early stages of the project, I would have been lost.

I'd like to make a shout-out to Shane, Vishaal, Alistair, Sam, Kevin, and many more people for making this year about so much more than just getting an academic education.

Lastly, I am forever indebted to the The Cecil Renaud Educational and Charitable Trust for turning a dream into reality.

Abstract

Most existing methods that apply pretrained Visual Language Models (VLMs) to vision and language tasks do not sufficiently explore the effect of the format of their inputs on downstream performance. We show that utilising appropriate prompt formatting is a simple yet effective approach to improving the few-shot performance of VLMs that use relatively small language models on the Visual Question Answering (VQA) task. We format the inputs used to prompt a VLM using a modified text-only template from a closed-book question answering task that the language-model component of the VLM was pretrained on. By doing this, we *explicitly align* the VQA task with a task that this language model has already seen, enabling the VLM to leverage the similarities between the tasks, such as the answer-length distribution, when generating answers to the visual questions.

In order to test our claims, we implement a simple architecture based on Frozen (Tsimpoukelli et al., 2021) and ClipCap (Mokady et al., 2021), whereby, through image captioning, the VLM learns to integrate powerful pretrained vision-only and language-only models via a relatively simple learnt mapping network. Furthermore, we contextualise our approach relative to existing work by presenting a unified view of VLMs.

Our results show that explicit alignment enables our VLMs to achieve a significantly higher zero-shot (34.49% vs 20.89%) and best overall (40.39% vs 30.83%) VQA score on the VQA2.0 dataset (Goyal et al., 2017) than when the prompt template from Frozen (Tsimpoukelli et al., 2021) and Flamingo (Alayrac et al., 2022) is used. Furthermore, our zero-shot and best overall performance is better than Frozen's (34.49% vs 29.5% and 40.39% vs 38.2%, respectively) despite Frozen using a language model with more than double the number of parameters. Our code is available here.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Recent developments in vision and language modelling have demonstrated that Visual Language Models (VLMs), which utilise visually-conditioned autoregressive language models (LMs), are capable of impressive performance on a wide range of open-ended vision and language tasks (Alayrac et al., 2022; Li et al., 2022, 2021; Wang et al., 2021; Yu et al., 2022; Zeng et al., 2021). However, the novelty of many of these VLMs is not primarily in the design of their architectures, but rather in their ability to learn effectively from either *task-agnostic web scraped data* (Alayrac et al., 2022; Wang et al., 2021; Yu et al., 2022), or through *large-scale multi-tasked training* (Li et al., 2022, 2021; Zeng et al., 2021). This trend is consistent with developments in the language-modelling domain, where state-of-the-art (SOTA) results are generally only obtained with significant scale in data and model size (Hoffmann et al., 2022a). As only few institutions have the computational resources to achieve this scale, researchers have to find ways of leveraging these powerful pretrained models, without making any changes to their parameters.

Since large LMs have been shown to possess exceptional text-generation abilities (Brown et al., 2020; Radford et al., 2019; Raffel et al., 2020; Sanh et al., 2021), there are a growing number of approaches in vision and language modelling that attempt to insert visual information into the space of a *pretrained* large LM (Alayrac et al., 2022; Chen et al., 2022; Cho et al., 2021; Li et al., 2022, 2021; Luo et al., 2022; Mokady et al., 2021; Tsimpoukelli et al., 2021; Yang et al., 2022; Yu et al., 2022; Zeng et al., 2021), thus retaining the powerful text-generation abilities of the pretrained LM, and providing access to the information retained from LM pretraining (Roberts et al., 2020). The resulting *visual-conditioned text-generation* abilities can then be applied to downstream vision and language tasks that can be cast as a text-generation problem, such as visual question answering, visual entailment, visual captioning, and visual grounding.

Despite the growing literature pointing to the importance of the input-format to downstream performance (Alayrac et al., 2022; Reynolds and McDonell, 2021; Sanh et al., 2021), many VLMs that utilise a vision-condition LM do not sufficiently explore the effect of the format

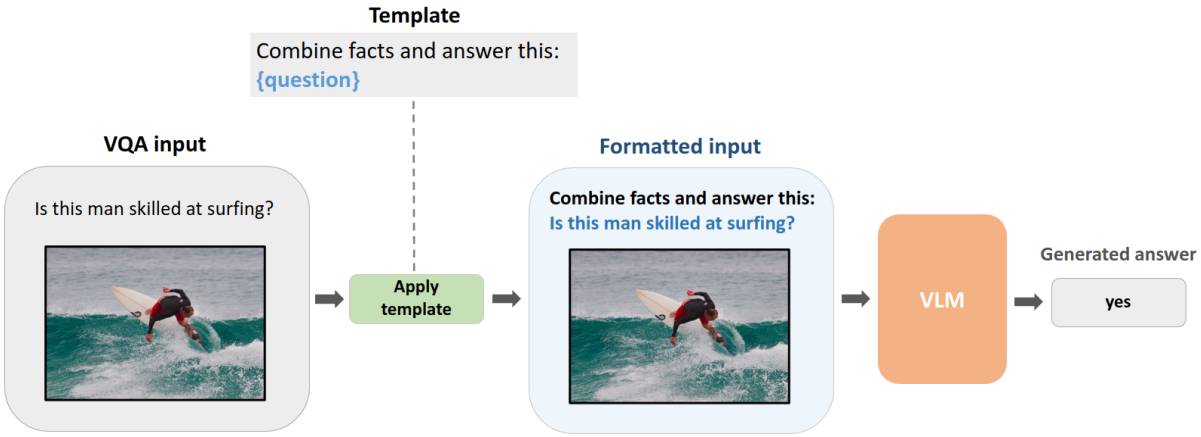


Fig. 1.1 **Explicit task alignment for VQA via prompt formatting.** A high-level overview of our approach to improving the performance of VLMs on the VQA task, where the VLM is required to predict an answer to a question about an input image. The inputs are first formatted using a prompt template, then the VLM is conditioned on the formatted input when autoregressively generating an answer.

of their inputs on downstream performance. With a focus on the Visual Question Answering (VQA) task, Flamingo (Alayrac et al., 2022) and Frozen (Tsimpoukelli et al., 2021) simply place "Question:" and "Answer:" before the input question and answer, respectively, with no justification despite the former stating that "the format of the task matters a lot". MAGMA (Eichenberg et al., 2021) uses "Q:" and "A:" as it results in "superior performance", but do not outline the alternative prompt formats they explored. However, Sanh et al. (2021) argue that although these models have not *explicitly* seen this formatting before, they might have seen it *implicitly*, given the scale of recent LMs' pretraining corpora. Hence, these prompts might implicitly align the VQA task with their training task, improving performance.

However, the ability to implicitly learn multiple tasks during training requires a sufficiently large LM (Press et al., 2021; Reynolds and McDonell, 2021; Sanh et al., 2021). With this in mind, in this project we investigate whether utilising appropriate prompt formatting can improve the few-shot performance of VLMs, that use relatively small LMs, on the VQA task. In particular, we explore formatting the inputs used to prompt the VLMs using a modified text-only template from a closed-book question answering task that the LM component of the VLM was pretrained on (see Figure 1.1). By doing this, we hope to *explicitly align* the VQA task with a task that this LM has already seen, enabling the VLM to leverage the similarities between the tasks, such as the answer-length distribution, when generating answers to the visual questions.

In order to test our claims, we implement a simple architecture based on Frozen (Tsimpoukelli et al., 2021) and ClipCap (Mokady et al., 2021), whereby, through image captioning, the VLM learns to bridge powerful pretrained vision-only and language-only models via a relatively simple learnt mapping network. We contextualise our approach relative to existing work by presenting a unified view of VLMs. We then evaluate our trained VLMs on the VQA2.0 (Goyal

et al., 2017) dataset via few-shot in-context learning in order to identify the effect of prompt formatting on VQA performance.

To summarise, our contributions are as follows:

1. We demonstrate that a VLM's few-shot VQA performance can be improved by explicitly aligning the VQA task with a text-only closed-book question answering task that the LM component of the VLM was pretrained on. To the best of our knowledge, we are the first to demonstrate that explicit alignment can be implemented in this way for vision and language tasks, and we show that the corresponding performance gains result in our VLM outperforming Frozen (Tsimpoukelli et al., 2021), who use an LM with more than double the number of parameters.

2. We provide a unifying view of the rapidly developing field of joint vision and language modelling. By doing this, researchers can easily comprehend the high-level architectural components of many VLMs and how the VLMs are related to each other.

3. To the best of our knowledge, we are the first to provide an open-source, reproducible implementation of in-context learning for VQA using a vision-condition LM[1]. Our code is available here.

4. To the best of our knowledge, we are the first to use an approximate implementation of RICES (Yang et al., 2022) that is suitable when only moderate computational resources are available. In particular, we outline how a FAISS (Johnson et al., 2019) index can be used to implement a fast approximation to RICES (see Section 5.3.6).

1.1 Report Outline

The outline of the report is as follows:

In Chapter 2, we provide the necessary context upon which the project's methodology relies. In particular, we present the field of vision and language modelling by first introducing the related methods from language modelling which inspired many of the developments in the vision-and-language domain. We conclude the chapter with a review of the importance of the "task format" for few-shot performance.

In Chapter 3, we present a unified view of a subset of the recent developments in vision and language modelling. Specifically, we outline the different Visual Language Models (VLMs)

[1]Neither Flamingo (Alayrac et al., 2022) nor Frozen (Tsimpoukelli et al., 2021) have made their implementations publicly available, and the implementation of PICa (Yang et al., 2022) requires a GPT-3 (Brown et al., 2020) API key.

that have been developed and demonstrate how their respective architectures can be viewed as variations on some common elements.

In Chapter 4, we describe our approach to using a VLM to produce free-form answers to visual questions, and contextualise our approach relative to previous work using the unified view from the previous chapter. Most notably, we then detail how template-based prompt formatting can be used to improve the few-shot performance of our trained VLMs.

In Chapter 5, we outline the setup of our experiments. This involves (1) listing the specific models used as the architectural components of our VLM; (2) describing how the VLM is trained; and (3) outlining the setup for our main experiments: evaluating whether using template-based prompt formatting – explicitly aligning the VQA task with a task that the LM component of our VLM was pretrained on – can improve few-shot performance.

In Chapter 6, we present and discuss the main experimental results: finding that the few-shot VQA performance of a VLM can be improved by explicitly aligning the VQA task with a text-only closed-book question answering task. We conclude the chapter with the limitations of our approach and our recommendations for future work.

In Chapter 7, we conclude the report.

Chapter 2

Background

The field of vision and language modelling has been influenced significantly by developments in language modelling (Alayrac et al., 2022). In order to capture the resulting dependencies, we will first introduce a selection of topics from language modelling (Section 2.1) before diving into vision and language modelling (Section 2.2), highlighting where text-only techniques have been adapted to the multi-modal domain. We then outline how vision and language models can be used in the few-shot setting (Section 2.3), and finish off by exploring the importance of the "task format" for few-shot performance (Section 2.4).

2.1 Language modelling

The language-modelling field has rapidly progressed in recent years (Brown et al., 2020; Devlin et al., 2018; Radford et al., 2019; Raffel et al., 2020; Sanh et al., 2021), following the widespread adoption of Transformers (Vaswani et al., 2017). Transformers are better equipped to model long-range dependencies over RNN-based approaches, while significantly increasing the throughput of models and therefore enabling the training of models on significantly larger datasets (Alayrac et al., 2022). The resulting paradigm of pretraining models on vast quantities of noisy data and then adapting the models for downstream usage has become standard (Liu et al., 2021b).

Given the success of pre-trained large language models (LMs), a range of techniques have arisen to adapt these general-purpose models to downstream tasks. The dominant adaptation technique has been model-tuning (or "fine-tuning"), where a subset of the LM's parameters are

updated via supervision on a downstream task[1]. However, as the scale of LMs continues to grow, updating their parameters is becoming increasingly computationally infeasible for most.

Adapting *frozen* large LMs has become an attractive alternative, whereby a relatively small number of task-specific trainable parameters are *added* to a frozen LM to optimize performance on a specific task. Some of the leading approaches include prompt tuning (Lester et al., 2021a), prefix tuning (Li and Liang, 2021), adapter tuning (Houlsby et al., 2019) and low-rank adaptation (Hu et al., 2021). In this paper, we utilise a method similar to prompt tuning, which falls under the umbrella of prompting methods for language models.

2.1.1 Prompting large language models

As an alternative to the *pre-train, fine-tune* paradigm, prompting methods follow the *pre-train, prompt, and predict* paradigm (Liu et al., 2021b). Prompting is the approach of providing additional inputs for an LM to condition on when generating an output sequence. Appropriate prompts can inform the model behavior so that a pre-trained LM itself can be used to predict the desired output, sometimes even without any additional task-specific training (e.g. Brown et al. (2020); Radford et al. (2019); Raffel et al. (2020)). The advantage of this method is that a single LM, trained in an entirely unsupervised fashion, can be used to solve a great number of tasks, simply by selecting appropriate prompts for each task (Brown et al., 2020; Sanh et al., 2021). For example, GPT-3 (Brown et al., 2020) can be adapted to QA-tasks by prompting the model with inputs such as, "Q: Where did Harry Potter go to school? A:" (Brown et al., 2020).

In order to provide context for later sections, we will outline two approaches to prompting: (1) *learning soft prompts*, where additional prompt-relevant parameters are introduced and learnt, and (2) *in-context learning*, where an LM is prompted with training examples.

Learning soft prompts. Since manually finding the most appropriate prompts for any given task is cumbersome, methods have been developed to learn them. Initial "prompt-tuning" methods aimed to find an optimal "hard prompt" by selecting a sequence of prompt tokens, through either manual search or non-differentiable search methods, *from the vocabulary* of the LM (Jiang et al., 2020; Shin et al., 2020). However, because the purpose of prompting is to find information that, when conditioned on, enables an LM to effectively perform a task, it is not necessary to limit the prompt-search to human-interpretable natural language. A simpler alternative, prompt tuning (Lester et al., 2021a), directly learns a "soft prompt" for each task. A soft prompt is a learnable sequence of embeddings that is not necessarily associated with tokens from the LM's vocabulary. Levine et al. (2022) expanded on this idea by conditioning the learnt prompts on the tokenized input, resulting in *input-dependant prompt tuning (ID-PT)*. See

[1]This method of transfer learning was proposed fairly recently by Howard and Ruder (2018) for language modelling, but has been common practice in the computer vision field for a number of years.

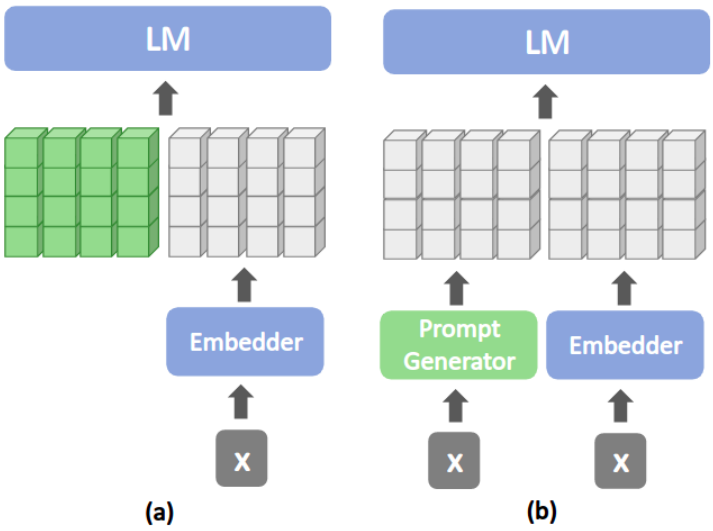


Fig. 2.1 **Learning soft prompts.** (a) Prompt tuning (Lester et al., 2021a), where the embedding of the soft prompt is learnt directly. (b) Input-dependent prompt tuning (ID-PT; Levine et al. (2022)), where the soft prompt is generated by passing the input through a *prompt generator*, a model that accepts tokenized input sequences and outputs a sequence of learnt embeddings. Blue indicates frozen parameters, whereas green indicates trainable parameters.

Figure 2.1 for a comparison of the two methods. The soft prompts can be learnt jointly while fine-tuning the LM, or can be learnt while keeping the LM frozen. We utilise a method that can be viewed as a form of ID-PT, where the learned soft prompts are conditioned on visual inputs and the LM is frozen.

In-context learning. Brown et al. (2020) introduced the idea of *in-context learning*, where, at inference time, an LM is conditioned on a prompt consisting of training examples followed by a test query which requires a completion. The parameters of the LM are left unchanged. An example of an in-context-learning prompt is shown in Figure 2.2. Although the idea of in-context learning is simple, there are several aspects that make it challenging: (1) the choice of in-context examples can result in very different performance, ranging from near state-of-the-art accuracy on some tasks to near random guess (Lu et al., 2021). It is desirable to use a form of example selection that both limits the sequence length[2] as well as finds examples that are more informative for correctly predicting the response to the test input, leading to better performance (Liu et al., 2021a). (2) The order of the examples can influence model performance (Lu et al., 2021). These considerations are explored further in Section 2.3.

Few-shot learning. A popular application of in-context learning is *few-shot learning*, where the model is only allowed to see a very small number of training examples before being evaluated on a held-out set. This is in contrast to *full-data learning*, where a reasonably large number of

[2]Press et al. (2021) showed that there is a risk of poor generalisation when the prompt size exceeds the size of the sequences seen during training (i.e. the max input length of the LM).

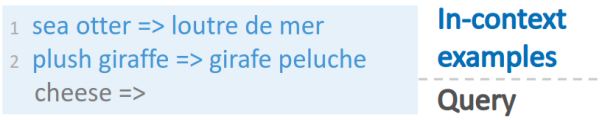


Fig. 2.2 **In-context learning.** A demonstration of in-context learning through the task of translating English sentences to French. The prompt consists of two in-context examples of English to French translations followed by the test input (i.e. the query). The Figure is adapted from Brown et al. (2020).

training examples are used to train the model. To clarify terminology, the *number of shots* (when using in-context learning) is the number of distinct, complete in-context examples from the task's training set presented to the model prior to the test query. For example, in visual question answering, a shot is an image along with an associated question and answer.

Given this context from the language-modelling domain, we can now introduce the field of vision and language modelling.

2.2 Joint vision and language modelling

We will focus on two families of related models for vision and language modelling: contrastive dual encoder approaches and visual language models (VLMs). For simplicity, we omit the field of language-conditioned visual generation (e.g. Ramesh et al. (2021)) and BERT-based (Devlin et al., 2018) approaches.

2.2.1 Contrastive dual encoder approaches

In the last few years, a large family of vision-language models, based on contrastive learning have emerged (Alayrac et al., 2022; Jia et al., 2021; Radford et al., 2021; Zhai et al., 2022). These models generally encode vision and text inputs with separate encoders, producing individual vision and language vectors embedded into a joint space using a contrastive loss (Jia et al., 2021; Radford et al., 2021). The contrastive loss encourages representations of *paired* images and texts to be similar, and representations of *non-paired* images and texts to be dissimilar. The trained dual-encoders can be used to produce highly generic textual and visual representation that can be used for downstream tasks, such as image classification, image retrieval, image-text matching and cross-modal retrieval (Alayrac et al., 2022; Jia et al., 2021; Radford et al., 2021; Zhai et al., 2022). We leverage a *frozen* pretrained vision encoder, that was trained with a contrastive loss, in order to extract powerful image-representations.

2.2.2 Visual language models (VLMs)

Visual language models (Alayrac et al., 2022; Cho et al., 2021; Li et al., 2022; Luo et al., 2022; Mokady et al., 2021; Tsimpoukelli et al., 2021; Wang et al., 2021; Yang et al., 2022; Yu et al., 2022) can model vision and language jointly, *and* are able to generate text in an autoregressive manner.

Visual-conditioned text-generation. Since large LMs have been shown to possess exceptional text-generation abilities, there is a growing number of approaches (Alayrac et al., 2022; Chen et al., 2022; Cho et al., 2021; Li et al., 2022, 2021; Luo et al., 2022; Mokady et al., 2021; Tsimpoukelli et al., 2021; Wang et al., 2021; Yang et al., 2022; Yu et al., 2022; Zeng et al., 2021) that attempt to insert visual information into the space of a *pretrained* large LM, thus retaining the powerful text-generation abilities of the pretrained LM, and providing access to the information retained from LM pretraining (Roberts et al., 2020). The resulting *visual-conditioned text-generation* abilities can then be applied to downstream vision and language tasks that can be cast as a text-generation problem, such as visual question answering, visual entailment, visual captioning, and visual grounding. The manner in which these approaches fuse the visual signal into the LM is discussed and unified in Chapter 3. We will expand on the most relevant approach to this project here, namely, learning image-conditioned soft-prompts.

Learning image-conditioned soft-prompts. Frozen (Tsimpoukelli et al., 2021), ClipCap (Mokady et al., 2021) and MAGMA (Eichenberg et al., 2021) inject the visual signal into a pretrained LM by prompting the model with a learnt sequence of continuous image representations. This can be viewed as a form of input-dependent prompt-tuning (ID-PT; see *learning soft prompts* in Section 2.1.1) where the input passed to the prompt generator is an image, and the output is a learnt *image-conditioned soft-prompt*. The VLM architecture used in this project is very similar to Frozen and ClipCap.

From pixels to image representations. In order to condition LMs on visual signals, an important consideration is how to map the input images to embeddings with which the LM can be conditioned. Two main approaches have emerged for this, both of which make use of pre-trained models: (1) transforming the images into textual features, such as captions, tags and object names (Gao et al., 2022; Gui et al., 2021; Yang et al., 2022), and (2) utilising learnt continuous image representations obtained from the output of a vision encoder trained, most popularly, with a contrastive loss (Alayrac et al., 2022; Mokady et al., 2021; Tsimpoukelli et al., 2021), or with a combination of losses (Li et al., 2022, 2021; Yu et al., 2022; Zeng et al., 2021). The former approach is attractive as it enables the use of pretrained LMs "out-of-the-box" since the visual inputs are converted directly into the text-domain. However, this approach has two inherent limitations: (a) if the textual descriptions of the image do not capture the

necessary information to complete the task, then the model can only guess the correct response[3]; (b) numerous specialised models have to be used in order to obtain the textual description, including captioning models (Li et al., 2020; Zhang et al., 2021), OCR models (Du et al., 2020) and object-detection models (Ren et al., 2015). We favour utilising learnt continuous image representations because they generally require only one model, a vision encoder, and have been shown to posses rich generic representations of images (Alayrac et al., 2022; Jia et al., 2021; Mokady et al., 2021; Radford et al., 2021; Shen et al., 2021; Tsimpoukelli et al., 2021). In particular, the image embeddings obtained from CLIP's vision encoders (Radford et al., 2021) have been shown to be particularly advantageous for VLMs on vision and language tasks (Shen et al., 2021).

2.3 Few-shot learning in vision and language modelling

With the modelling of vision and language tasks moving into the language-modelling domain, methods that are well-developed for LMs are being transferred to VLMs. In-context learning (Brown et al., 2020) is one such method that has recently been extended to vision and language tasks (Alayrac et al., 2022; Eichenberg et al., 2021; Tsimpoukelli et al., 2021) with very minor modifications, as shown in Figure 2.3. An attractive characteristic of VLMs that use image-conditioned soft-prompts for multimodal fusion (e.g. Frozen, ClipCap, MAGMA) is that the treatment of the in-context examples is the same as in language modelling – the LM is conditioned on this prompt when generating a response.

In-context example selection. As mentioned in Section 2.2, in language modelling, the choice of in-context examples can result in very different performance, ranging from near state-of-the-art accuracy on some tasks to near random guess (Liu et al., 2021a). It has been shown that the same is true for in-context learning in vision and language tasks (Alayrac et al., 2022; Yang et al., 2022). Although it would be desirable from a computational complexity standpoint to randomly select in-context examples from the support set, a more principled approach to selecting these examples has been shown to be beneficial to performance (Alayrac et al., 2022; Yang et al., 2022). In the language-modelling domain, Liu et al. (2021a) proposed an effective k nearest neighbours (kNN) based method for choosing in-context examples, called *KATE*[4]. For a given test input, KATE works by finding the test input's kNNs from the training set, where the similarity of any two inputs is the distance (e.g. cosine similarity) between their respective embeddings in the embedding space of a text encoder[5]. Following this work, Yang et al. (2022)

[3]Using an example from Yang et al. (2022), if the task is to count the number of giraffes in an image, but the textual description does not enumerate the giraffes, then the best the LM can do is guess.

[4]KATE: Knn-Augmented in-conText Example selection.

[5]This approach is also very similar to the work of Gao et al. (2020).

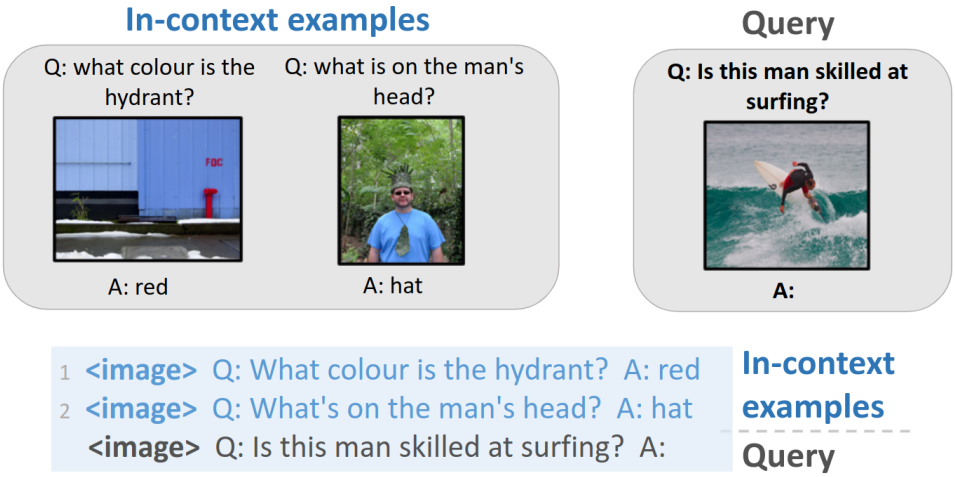


Fig. 2.3 **In-context learning for VLMs.** A demonstration of in-context learning for VLMs, following the methodology of Frozen (Tsimpoukelli et al., 2021), through the task of visual question answering. The prompt consists of two in-context examples comprised of interleaved visual inputs and textual inputs, followed by a final test input (i.e. the query). The "<image>" token represents the location at which the embedding of the soft prompt will be inserted (this is expanded on in Section 4.3.2).

proposed an effective method that extends KATE into the vision and language domain, called *RICES*[6]. Rather than using text similarity alone, RICES averages the textual and visual similarity when finding the k nearest neighbours, resulting in improved performance compared to when KATE is applied directly to vision and language tasks (Alayrac et al., 2022; Yang et al., 2022). Given these findings, we utilise RICES for in-context example selection.

2.4 Task format

Brown et al. (2020) raise the question of whether LMs actually "learn" new tasks at inference time from the in-context examples based on the provided input-output mappings, or simply recognize and identify tasks learned during training. The findings of Reynolds and McDonell (2021) suggest that the function of few-shot examples is the latter: to find the *task location* in the model's existing space of learned tasks. This is supported by Min et al. (2022), who found that demonstrating the ground-truth mapping from input to output through the in-context examples generally has limited impact on few-shot performance, as opposed to specifying the overall format of the examples. In particular, they showed that the characteristics of the in-context examples that are most important to downstream performance are that they provide a few examples of (1) the label space, (2) the distribution of the input text, and (3) the overall format of the sequence (Min et al., 2022). The importance of the last characteristic is supported by Reynolds and McDonell (2021), who found that zero-shot prompts can significantly outperform

[6]RICES: Retrieval-based In-Context Example Selection.

few-shot prompts when the zero-shot prompts are formatted appropriately. For example, they show that, for the English-to-French translation task, prepending "English:" and "French:" before the English source sentence and French target sentence, respectively, leads to better translations when no shots are presented to the model, compared to the translations when 10 shots are provided in the format shown in Figure 2.2, taken from the original GPT-3 paper (Brown et al., 2020). This motivates the question: Is there a methodology which can be followed to find prompts that are more likely to yield desired behavior? We explore several methodologies that aim to do this in the next section.

2.4.1 Aligning downstream tasks with training

A number of approaches have been developed for both text-only and vision-and-language modelling that aim to format tasks such that they more closely resemble the tasks that their respective LM was pretrained on (Gao et al., 2020; Liu et al., 2022; Song et al., 2022).

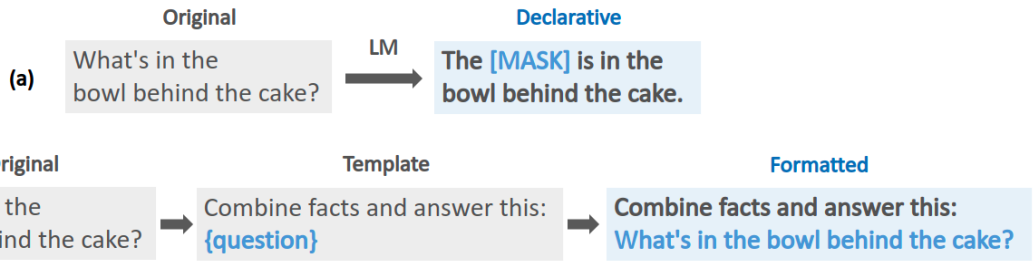


Fig. 2.4 **Aligning downstream tasks with training.** (a) Transforming tasks into masked LM problems: an example, taken from Song et al. (2022), demonstrating how closed-book QA tasks can be recast into a masked LM task by transforming the question (via a learnt mapping) into a masked declarative sentence. (b) Template-based formatting of tasks: an illustrative example of formatting a question using a template.

Learnt alignments. This idea has been used to successfully enhance task adaptation through fine-tuning (Gao et al., 2020; Liu et al., 2022) and zero-shot transfer (Song et al., 2022), where the alignment of test tasks with training tasks has been *learnt*. Expanding on this, the pretrained LMs used in Gao et al. (2020); Liu et al. (2022); Song et al. (2022) are pretrained with masked-language-modelling tasks (Devlin et al., 2018), in which they were trained to predict masked text pieces based on the surrounding context. Thus, they each transform downstream tasks into this form, inserting *MASK* tokens where the response should be generated (see Figure 2.4 for an example). This transformation is achieved by adapting a pretrained LM which learns the relevant mapping either through fine-tuning (Gao et al., 2020; Liu et al., 2022) or in-context learning (Song et al., 2022). However, these methods induce extra complexity into the modelling process and run the risk of propagating errors, when converting tasks into the masked LM form, to downstream performance.

Template-based alignments. The development of LMs trained via multitask prompted training (Mishra et al., 2021; Raffel et al., 2020; Sanh et al., 2021; Wei et al., 2021) provides a simpler alternative: the use of *templates* to format downstream tasks in the *same form* as training tasks, *without* the need for learnt mappings (see Figure 2.4 for an example). We demonstrate that performance gains can be observed by formatting vision and language tasks using templates from LMs pretrained using multitask prompted training.

Implicit alignments. Interestingly, despite the growing literature pointing to the importance of the formatting of the in-context examples to downstream performance (Alayrac et al., 2022; Reynolds and McDonell, 2021), many VLMs do not include this consideration in their analysis. With a focus on the visual question answering (VQA) task, Flamingo (Alayrac et al., 2022) and Frozen (Tsimpoukelli et al., 2021) simply place "Question:" and "Answer:" before the input question and answer, respectively, with no justification despite Flamingo stating that "the format of the task matters a lot". MAGMA (Eichenberg et al., 2021) uses "Q:" and "A:" as it results in "superior performance", despite not outlining what other alternatives they explored. However, Sanh et al. (2021) argue that although these models have not *explicitly* seen this formatting before, they might have seen it *implicitly*, given the scale of recent LMs' pretraining corpora. Hence, these prompts might implicitly align the VQA task with their training task, improving performance.

Explicit alignments. However, the ability to implicitly learn multiple tasks during training requires a sufficiently large LM (Press et al., 2021; Reynolds and McDonell, 2021; Sanh et al., 2021). We explore whether *explicit alignment* (Sanh et al., 2021) via appropriate prompt formatting can be used to improve the performance of relatively small LMs, closing the performance gap between smaller and larger LMs.

2.5 Conclusion

In this chapter, we have provided the necessary context upon which the proceeding chapters rely. We started the chapter by introducing a few topics from the language-modelling domain so that we could demonstrate how the developments in vision and language modelling have been influenced by language modelling (Section 2.1). We briefly introduced how VLMs have been developed that can model vision and language jointly (Section 2.2), how these models can be applied in the few-shot setting through in-context learning (Section 2.3), and highlighted the importance of the task format to downstream performance (Section 2.4).

In the next chapter, we will look deeper into the recent VLMs that have been developed and provide a unified view of these methods, such that the commonalities between the approaches can be identified.

Chapter 3

Unified View of Visual Language Models

The field of vision and language modelling is very fast-moving: there have been numerous papers published over the last two years alone. In this chapter, we review these developments by presenting a unified view of a subset[1] of the recent approaches. By doing this, researchers can easily comprehend the high-level architectural components of many visual language models (VLMs) and how they relate to other architectures. We start by outlining the different VLMs that have been developed (Section 3.1) before demonstrating how their respective architectures can be unified (Section 3.2).

3.1 The state of VLMs

Before unifying a subset of VLMs, we provide a review of their methodologies. In particular, we focus on the manner in which they condition language-model decoders on visual and textual signals through cross-modality fusion. We do not review the low-level details of the papers, such as their training objectives.

3.1.1 Recent VLM approaches

There has been a recent movement (Cho et al., 2021; Li et al., 2022, 2021; Wang et al., 2022; Zeng et al., 2021) to formulate some vision tasks as text generation problems, including classification, captioning, VQA, visual entailment and visual grounding. This simplified prior work relying on learning task-specific classification layers, and opened the door to utilising transformer-

[1] Due to the rate at which the field has grown over the last few years, there are simply too many approaches to unify within a reasonable period of time. Thus, we have used judgment to select a few influential approaches in order to demonstrate recurring themes.

based decoders, conditioned on visual and textual signals, to autoregressively generate textual responses to these tasks.

However, training large-scale language models (LMs) is computationally expensive and data-hungry (Hoffmann et al., 2022b). As a consequence, building on top of a powerful pretrained text-only LM is an attractive alternative and has been widely explored. The following methods condition a decoder on a mixture of visual and textual signals, either through the decoder's cross-attention layers or by prompting the decoder directly.

Modality fusion through decoder cross-attention. VisualGPT (Chen et al., 2022), Flamingo (Alayrac et al., 2022) and the vanilla approach of VC-GPT (Luo et al., 2022) showed that a causally masked decoder-only LM can be conditioned on visual inputs by adding learnt cross-attention layers to the decoder. In order to do this, the visual input is first mapped into a continuous representation compatible with the hidden-dimensions of the transformer LM, before being passed to the decoder as the keys and values of the attention mechanism. Furthermore, VisualGPT and Flamingo found it beneficial to use gated cross-attention layers, such that the decoder could be initialised with the weights of a pretrained LM. A similar approach to cross-modality fusion is observed in X-VLM (Zeng et al., 2021), ALBEF (Li et al., 2021) and CoCa (Yu et al., 2022), except that they encode the text modality with a transformer-based encoder prior to decoding, such that the inputs to the decoder are learnt text-encodings and not tokenized text. In contrast to these approaches, Flamingo only learns the cross-attention layers during training. The reason for this design is that, by building on top of a strong frozen LM, the VLM may retain the LM's powerful language-only abilities, such as few-shot language adaptation, external knowledge retrieval and dialogue capabilities (Alayrac et al., 2022).

Modality fusion through LM prompting. Frozen (Tsimpoukelli et al., 2021), ClipCap (Mokady et al., 2021) and MAGMA (Eichenberg et al., 2021) subscribe to this idea too, albeit using a different approach to modality fusion. They condition a decoder-only LM on visual inputs using a form of input-dependent prompt tuning (ID-PT; Levine et al. (2022)) where a *visual prefix*, encoded by a trainable visual encoder, is used to prompt a pre-trained LM. In Frozen and ClipCap, the pretrained LMs are frozen and unmodified, whereas MAGMA extends these methods by inserting bottleneck adapters (Houlsby et al., 2019; Sung et al., 2022) into the frozen LM. Finally, instead of using continuous vector representations of images, PICA (Yang et al., 2022) and Socratic Models (Zeng et al., 2022) make use of off-the-shelf vision-language models to transform images into language descriptions which can then be used to prompt GPT-3 (Brown et al., 2020).

These two approaches to modality fusion have been similarly applied to encoder-decoder architectures. For instance, SimVLM (Wang et al., 2021) and VL-T5 (Cho et al., 2021) insert the visual signal as input to the encoder of an encoder-decoder architecture. The decoder can

then attend to the resulting joint vision-text encoding when generating a response. Similarly, BLIP (Li et al., 2022) also pass jointly-encoded vision and text inputs to a decoder, but the joint encodings are obtained by passing the visual signal to a text-encoder via added cross-attention layers.

3.2 Unifying VLM architectures

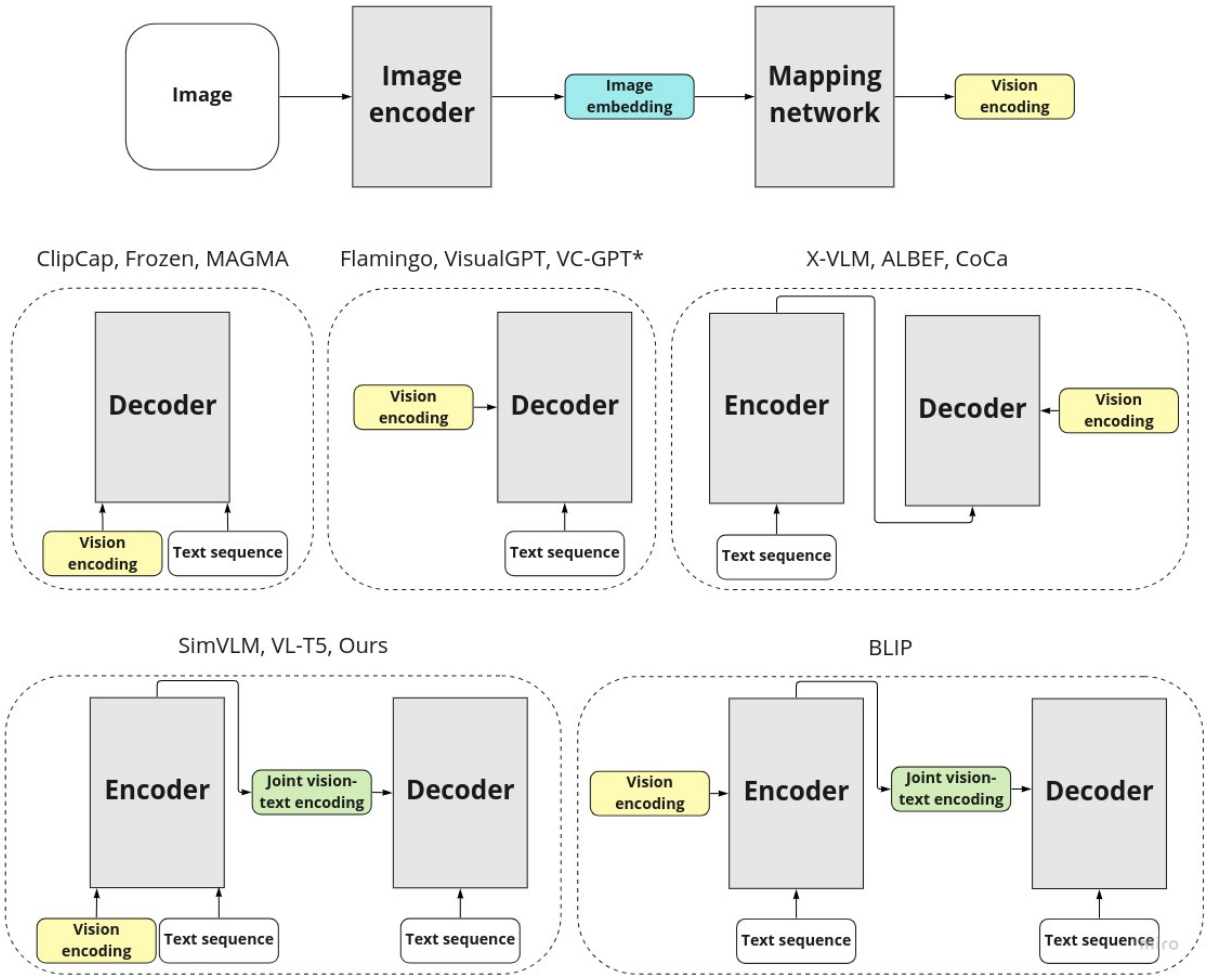


Fig. 3.1 **Unified view of VLM architectures.** We provide a unified view of a subset of recent VLMs. The high-level architectural structures of each method are shown and grouped. All of the VLMs share the same high-level mapping of images to vision encodings. The papers unified are: ClipCap (Mokady et al., 2021), Frozen (Tsimpoukelli et al., 2021), Flamingo (Alayrac et al., 2022), X-VLM (Zeng et al., 2021), ALBEF (Li et al., 2021), SimVLM (Wang et al., 2021), VL-T5 (Cho et al., 2021), BLIP (Li et al., 2022), CoCa (Yu et al., 2022), VisualGPT (Chen et al., 2022), MAGMA (Eichenberg et al., 2021), and the *vanilla framework* presented in VC-GPT (Luo et al., 2022). 'Ours' refers to the approach used in this project, expanded on in Chapter 4.

It is apparent from the review above that these VLMs share the same core idea: they want to find the most effective way to condition an LM on a mixture of visual and textual signals. As a result, there are natural commonalities between the methods. In this section we attempt to highlight the architectural similarities of the VLMs. However, choosing the level of abstraction at which to unify these models is an important consideration. We determined this level by considering what information would be necessary to design a flexible software suite to implement these VLMs. Thus, we focus on the high-level architectural components of the models, and choose not to focus on their respective training objectives or low-level implementation details. The resulting unified view is demonstrated in Figure 3.1, with more details of how each VLM fits into this framework shown in Table 3.1.

Table 3.1 **High-level architectural components.** Each model's implementation of the high-level components visualised in Figure 3.1. * indicates that the component's weights are initialised from a pretrained model. ** indicates that the component's weights are initialised from a pretrained model *and* the component is kept frozen.

Model	Vision encoder	Mapping network	Encoder	Decoder
Frozen	NF-ResNet-50*	Linear mapping	-	GPT-2** with relative position encodings
ClipCap	CLIP-ViT**	MLP / Transformer	-	GPT-2**
MAGMA	CLIP-ViT* / CLIP-ResNet*	Linear mapping	-	GPT-J** with adapters
VisualGPT	Transformer	Identify mapping	-	GPT-2* with cross-attention layers
Flamingo	NF-ResNet-F6*	Perceiver Resampler	-	Chinchila** with cross-attention layers
VC-GPT - vanilla	CLIP-ViT*	Not specified	-	GPT2** with cross-attention layers
VL-T5	Faster R-CNN**	Linear mapping + learnt embeddings	T5*	T5*
SimVLM	ViT	Identity mapping	Transformer	Transformer
CoCa	ViT	Attentional pooler	Transformer	Transformer
X-VLM	ViT*	Not specified	BERT*	BERT*
ALBEF	ViT*	Not specified	BERT*	BERT*
BLIP	ViT*	Not specified	Transformer	Transformer

Image to vision encoding. At this level of abstraction, we find that all of the listed models share the same treatment of the visual signal: the image is passed through an image encoder in order to obtain an image embedding. These image embeddings can take the form of flattened feature grids produced by Normalizer-Free ResNets (NF-ResNet; Brock et al. (2021), e.g. MAGMA, Flamingo), a selection of hidden states from the final layer of ViTs (Dosovitskiy et al. (2020), e.g. CoCa, ClipCap, SimVLM, X-VLM, ALBEF, BLIP), or region features obtained from Faster R-CNN (Ren et al. (2015); e.g. VL-T5). However, for the LM to incorporate the visual signal, the visual representation has to be compatible with the LMs hidden-state dimensions. Thus, a mapping network is applied in order to transform the image embedding into a form that is compatible with the LM of interest. The mapping network can take the form of a simple linear layer (e.g. Frozen, VL-T5, MAGMA), an MLP (e.g. ClipCap), a Transformer (ClipCap, Flamingo, CoCa), or an identity mapping (SimVLM, VisualGPT).

Grouping similar architectures. As demonstrated in Figure 3.1, the VLMs can be unified further by grouping those that have similar architectural structures. Evidently, there is large overlap between many of the VLMs, with respect to the high-level architectural components that they use as well as the manner in which they fuse the multi-modal signals.

3.3 Conclusion

Based on this investigation, it may be that the novelty of many of these VLMs is not primarily in the design of their architectures, but rather in their ability to learn effectively from *task-agnostic web scraped data*. Flamingo (Alayrac et al., 2022) is a clear example of this as it shares numerous ideas with some of the aforementioned VLMs, yet achieves state-of-the-art (SOTA) performance on many downstream tasks (Alayrac et al., 2022). In particular: (i) it relies on a frozen pretrained LM (same as Frozen, ClipCap and MAGMA), (ii) it uses a Transformer-based mapping network between the vision encoder and the frozen LM (same as ClipCap and CoCa), and (iii) it learns cross-attention layers that are interleaved with the frozen LM's layers (same as VisualGPT). This trend is consistent with the language-modelling domain, where SOTA results are generally only obtained with significant scale in data and model size (Hoffmann et al., 2022a). As only few institutions have the computational resources to achieve this scale, researchers must find alternative ways of improving the performance of large pretrained models.

With this in mind, in the next chapter we outline how utilising appropriate prompt formatting can be used as a simple, yet effective, approach to improving downstream performance.

Chapter 4

Approach

This chapter describes our approach to using a visual language model (VLM) to produce free-form answers to visual questions. We start by outlining the architectural components that comprise the model (Section 4.1), before moving onto the details of how the model is trained (Section 4.2). We then describe how the trained model can be used at inference-time, outlining the methodology used to incorporate few-shot in-context examples when generating responses (Section 4.3). Most notably, in Section 4.3.1, we detail how template-based prompt formatting can be used to improve the few-shot performance of the trained VLM.

4.1 Architecture

In order to contextualise the VLM used in this project relative to existing work, we will describe our VLM using the groupings and architectural components introduced in Figure 3.1. We use a VLM that is similar to SimVLM (Wang et al., 2021) and VL-T5 (Cho et al., 2021) from the perspective of their abstract *architectural form*, in that the visual signal is inserted as input to the encoder of an encoder-decoder Transformer language model (LM; Vaswani et al. (2017)). However, our VLM can equally be viewed as a modification of ClipCap (Mokady et al., 2021) and Frozen (Tsimpoukelli et al., 2021), whereby an encoder-decoder Transformer replaces their decoder-only Transformer. Thus, like ClipCap and Frozen, our VLM fuses textual and visual modalities through prompting the LM with learnable *image-conditioned soft-prompts* (see Section 2.2.2).

An overview of our VLM is shown in Figure 4.1. Like in Flamingo (Alayrac et al., 2022), we choose to freeze both the image encoder and LM, such that **only the mapping network is learnt**. Thus, during training, the VLM learns to bridge powerful pretrained vision-only and language-only models – an image encoder and a Transformer-based LM, respectively – via a

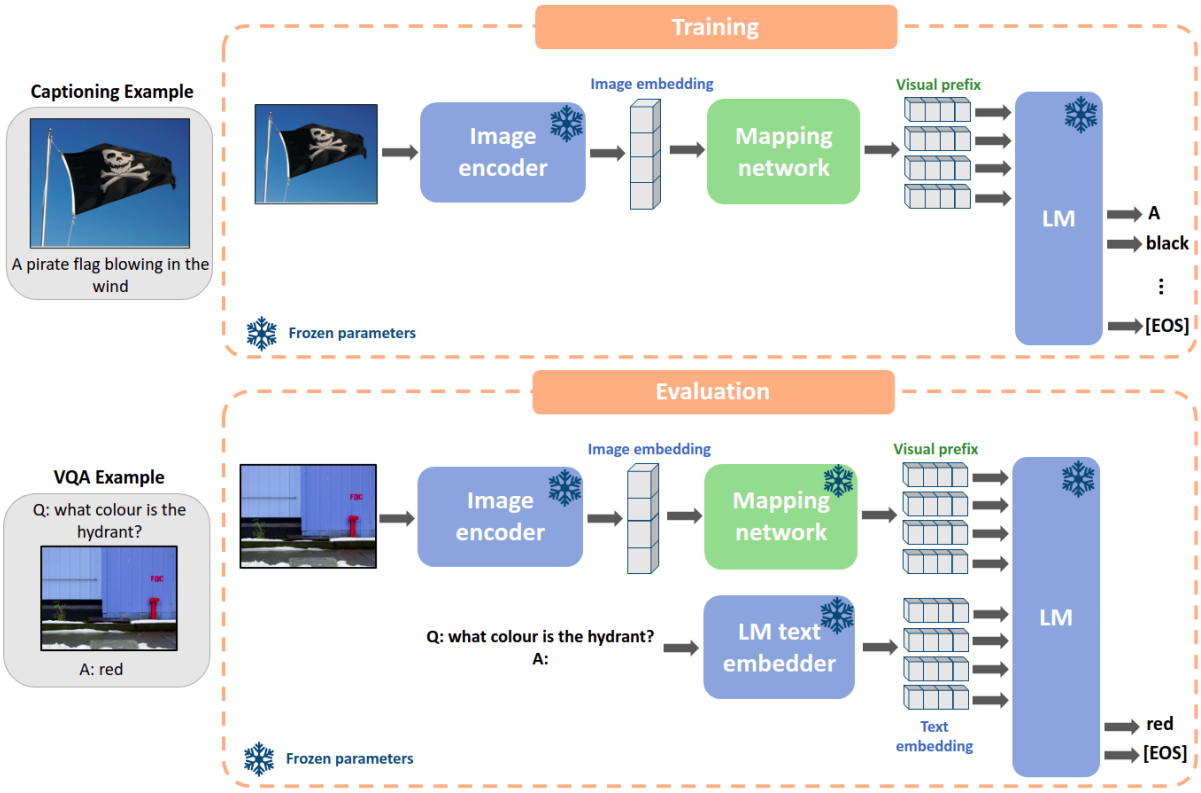


Fig. 4.1 **Overview of our VLM.** Our VLM bridges two unimodal pretrained models – an image encoder and a Transformer-based LM – via a relatively simple mapping network. Only the mapping network is learnt during training on the captioning task. The trained VLM is then frozen and evaluated on the VQA task.

relatively simple learnt mapping network. In doing so, the powerful unimodal abilities of the frozen models are retained: the pretrained image encoder can be used to extract highly generic image representations; and the text-generation ability of the powerful pretrained LM can be utilised, aided by the large amount of knowledge embedded in the pretrained LM (Roberts et al., 2020).

4.1.1 Frozen pretrained LM

The only constraints on the choice of Transformer LM are that it can be used for autoregressive text generation (ruling out non-causal models such as BERT; Devlin et al. (2018)) and that its pretraining inputs are formatted using prompt templates (for example, see Figure 2.4). The latter constraint is expanded on in Section 4.3.1. Although a decoder-only LM could also be used, we focus on using an encoder-decoder LM, primarily due to the availability of encoder-decoder LMs that meet the latter criterion.

4.1.2 Frozen pretrained image encoder

Similar to the other VLMs unified in Chapter 3, each image is passed through an image encoder in order to obtain an image embedding. The image encoder, $v(.)$, maps an input image $\boldsymbol{x}$ to an r-dimensional continuous vector representation, such that

$$\boldsymbol{i} = [i_1, \ldots, i_r]^T = v(\boldsymbol{x})$$

where $i_j \in \mathbb{R}, \forall j \in \{1, \ldots, r\}$.

Our methodology differs from Frozen (Tsimpoukelli et al., 2021) in that we *do not* update the parameters of the image encoder, as it is computationally advantageous[1] and beneficial from an image-representation perspective, as we could use a more powerful pretrained image encoder than we could feasibly train ourselves (see Section 5.1.2).

4.1.3 Mapping network and visual prefix

In order to utilise the strong text-generation ability of the LM while conditioning it on both text and vision inputs, in a similar fashion to ClipCap (Mokady et al., 2021), Frozen (Tsimpoukelli et al., 2021) and MAGMA (Eichenberg et al., 2021), the image embedding is mapped into the LM's embedding space via a learnt *mapping network*, $m(.)$. In particular, each r-dimensional image embedding is mapped to an n-length sequence of D-dimensional embeddings, $\boldsymbol{e}^*_1, \ldots, \boldsymbol{e}^*_n$, where D is the internal dimension of the LM. Following the convention of Frozen, we call this learnt embedding the *visual prefix*[2] as, from the perspective of the LM, these learnt embeddings are functionally equivalent to a sequence of n prefix tokens that have been embedded using the LM's text embedder.

4.1.4 Indifference to prompt modalities

Since text sequences and images are embedded into the same space through the LMs text embedder and mapping network, respectively, the LM is designed to be agnostic to the modality of the prompt. Thus, our VLM can easily be trained or evaluated on a variety of vision and language tasks by simply modifying the composition of the prompt. For example, the prompt can be composed of images only (e.g. captioning), or image and text pairs (e.g. VQA), or interleaved sequences of images and text (e.g. few-shot VQA), and the treatment of the prompt

[1] As the image encoder is frozen, we are able to extract all of the necessary image embeddings before training/inference and re-use them. Furthermore, we do not have to calculate the gradients or update the parameters of the image encoder.

[2] Visual prefix is an alternative name for the *image-conditioned soft-prompts* introduced in Section 2.2.2.

by the LM will be the same – the decoder will attend to the prompt during a forward pass. This means that the probability of the next output token, y_l, given previous output tokens, $y_{<l}$, and the visual and textual inputs can be compactly expressed as

$$p(y_l|\boldsymbol{Z}, y_{<l}) \tag{4.1}$$

where $\boldsymbol{Z}$ is a sequence of embeddings in the embedding space of the LM which can be comprised of sequences of visual prefixes, text embeddings or interleaved sequences of both, depending on the task. We refer to $\boldsymbol{Z}$ as the *prompt embedding* as it encapsulates the whole prompt on which the LM is conditioned, irrespective of the modalities of the prompt.

4.2 Training

In order to be comparable with Frozen, the mapping network is trained on image-captioning data (e.g. Conceptual Captions; Sharma et al. (2018)). Thus, only a very small proportion of the VLM's parameters are updated, making the model significantly faster to train and requiring significantly less computational resources.

We frame the captioning task as the conditional generation of the target caption $\boldsymbol{y} = y_1, \ldots, y_l$ given an input image $\boldsymbol{x}$. In order to do this, the frozen image encoder is used to obtain an image embedding, which the mapping network transforms into the visual prefix:

$$\boldsymbol{e}_1^*, \ldots, \boldsymbol{e}_n^* = m(v(\boldsymbol{x})) \ .$$

Since, for the captioning task, there are no textual inputs passed to the encoder of the LM, the prompt embedding is only comprised of the visual prefix, such that $\boldsymbol{Z} = \boldsymbol{e}_1^*, \ldots, \boldsymbol{e}_n^*$. By inserting the visual signal into the prompt embedding, the decoder of the LM can attend to the image through its cross-attention layers when generating a caption.

Following from Eqn. (4.1), we train the parameters of the mapping network with "teacher forcing" (Williams and Zipser, 1989) to maximise the likelihood

$$\begin{aligned} \log p(\boldsymbol{y}|\boldsymbol{x}) &= \sum_l \log p(y_l|\boldsymbol{Z}, y_{<l}) \\ &= \sum_l \log p(y_l|\boldsymbol{e}_1^*, \ldots, \boldsymbol{e}_n^*, y_{<l}) \ . \end{aligned}$$

During training, the loss is computed using a cross-entropy loss and gradients are back-propagated *through* the frozen LM to update the parameters of the mapping network via stochastic gradient descent (SGD).

4.3 Task adaptation with few-shot in-context learning

Our VLM, along with Frozen, ClipCap and MAGMA, can be easily adapted to a variety of vision and language tasks – that can be formulated as text-generation problems – at inference time by changing the composition of the prompt embedding, $\boldsymbol{Z}$. We evaluate the ability of our VLM to rapidly transfer its knowledge from the captioning task to the VQA task using in-context learning, following an analogous approach to Frozen. The high-level interface for this method, applied to the VQA2.0 task (Goyal et al., 2017), is shown in Figure 4.2.

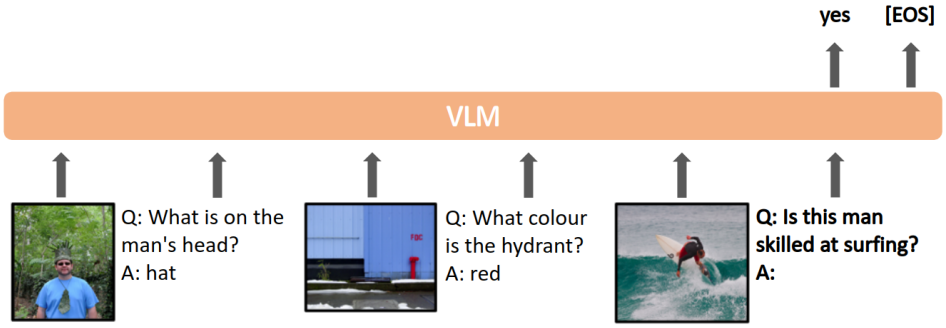


Fig. 4.2 **High-level interface for in-context learning.** Given in-context examples from the VQA2.0 (Goyal et al., 2017) training set and a query, the model is required to generate a textual response. The in-context examples and query are passed to the VLM as a sequence of interleaved visual and textual inputs.

4.3.1 Prompt format

As described in Section 2.4, the way that the task is formatted can significantly influence an LM's few-shot performance. In particular, it was highlighted in Section 2.4.1 that for both LMs and VLMs, aligning downstream tasks with the task that their respective LM was pretrained on can be beneficial to performance on the downstream tasks.

Despite this, as described in Section 2.4, many VLMs do not sufficiently explore the format of their inputs on downstream performance. Instead, with a focus on the VQA task, they simply format inputs by, for example, placing "Question:" and "Answer:" or "Q:" and "A:" before the input question and answer, respectively (Alayrac et al., 2022; Eichenberg et al., 2021; Tsimpoukelli et al., 2021). Sanh et al. (2021) argues that although these VLMs have not *explicitly* seen this formatting before, they might have seen it *implicitly*. Expanding on this further, the pretraining data used to train the frozen LMs used in each of these VLMs was not *explicitly* formatted using specific prompt templates. However, given the scale of their LMs' pretraining corpora, it is reasonable to expect that some common natural language processing

(NLP) tasks would appear in an explicit form[3] in their pretraining corpora (Sanh et al., 2021). Hence, these prompt formats might *implicitly* align the VQA task with their training task, improving performance.

However, the ability to implicitly learn multiple tasks during training requires a sufficiently large LM (Press et al., 2021; Reynolds and McDonell, 2021; Sanh et al., 2021). We hypothesise that template-based prompt formatting can be used to improve the performance of VLMs that use relatively small LMs by *explicitly* aligning the VQA task with the LM's training tasks. In order to try achieve this alignment, we format the VQA task using the formatting of a text-only closed-book QA task (e.g. hotpotqa (Yang et al., 2018), shown in Figure 2.4) that our VLM's LM was pretrained on. The specific templates used are tied to the LM, and thus will be expanded on in Section 5.1.1.

In order to illustrate how a prompt template is used to format inputs, we will use the naïve[4] template from MAGMA. This template has the form

`<image>` Q: `{question}` A: `{answer}`

where "`{question}`" and "`{answer}`" are placeholders for the input question and answer, respectively, and "`<image>`" is a token used to identify the location at which the visual prefix will be inserted relative to the text embedding when constructing the prefix embedding, $\boldsymbol{Z}$. An example of using the MAGMA template to define the construction of $\boldsymbol{Z}$ is shown in Figure 4.3.

4.3.2 Building the prompt embedding

In order to adapt the trained VLM to the VQA task using in-context learning, we need to build an appropriate prompt embedding that encapsulates the visual and textual signals of all the in-context examples and the test input.

Let $(\boldsymbol{x}, \boldsymbol{q}, \boldsymbol{a})$ denote an item from a VQA dataset, comprised of an image ($\boldsymbol{x}$), a question ($\boldsymbol{q}$) and an answer ($\boldsymbol{a}$). Then, given a set of K in-context examples, $\{(\boldsymbol{x}_i, \boldsymbol{q}_i, \boldsymbol{a}_i) : i \in 1, \ldots, K\}$, and a query, $(\boldsymbol{x}^*, \boldsymbol{q}^*)$, we build the overall prompt embedding, $\boldsymbol{Z}$, by concatenating the respective prompt embeddings for each in-context example and the query. Thus,

$$\boldsymbol{Z} = \boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_K, \boldsymbol{Z}^*$$

[3] Using the example from Sanh et al. (2021), there are many websites that simply contain lists of trivia questions and answers. Using one such website as an example, the website presents question-answer pairs in an explicit format: "What does www stand for in a website browser? Answer: World Wide Web" (the example question-answer pair is from https://www.quizbreaker.com/trivia-questions).

[4] We call this template "naïve" because Eichenberg et al. (2021) do not justify why it is used, however, it is a sensible format for question-answer pairs.

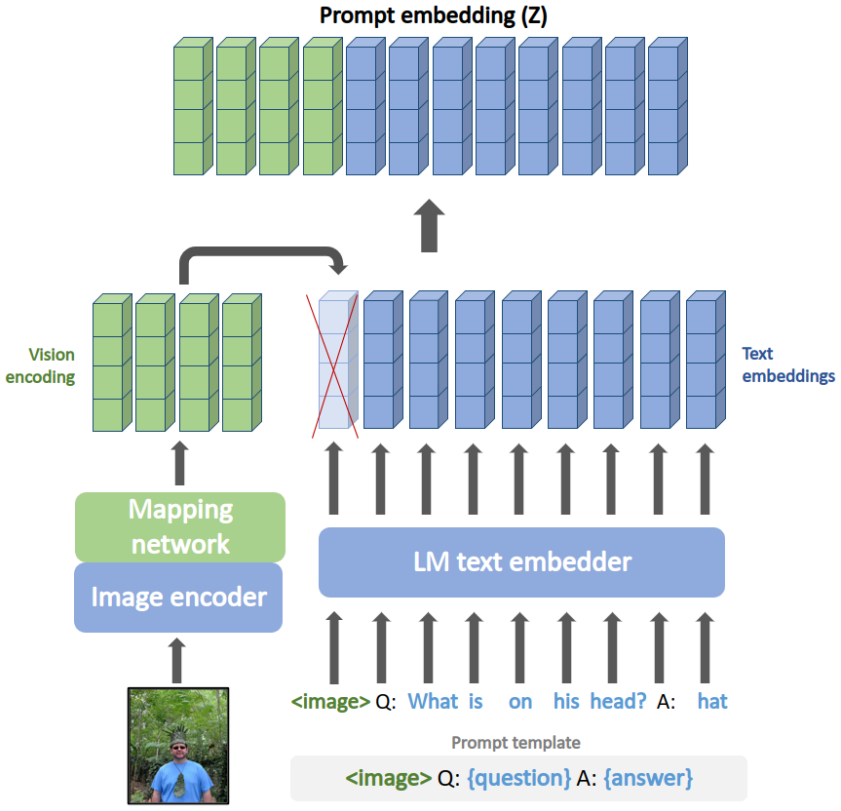


Fig. 4.3 **Constructing the prompt embedding from a prompt template.** The prompt embedding, $\boldsymbol{Z}$, is constructed by replacing the embedding of the `<image>` token with the visual prefix (i.e. the sequence of continuous vectors comprising the vision encoding). The prompt template is taken from MAGMA (Eichenberg et al., 2021).

where $\boldsymbol{Z}_i$ is the prompt embedding for in-context example i and $\boldsymbol{Z}^*$ is the prompt embedding for the query. The construction of the prompt embedding for an arbitrary in-context example, is visualised in Figure 4.3. The construction is identical for a query except that no answer is provided to the VLM. The frozen encoder-decoder LM is then prompted with the overall prompt embedding when generating an answer to the query.

4.3.3 In-context example selection

In K-shot learning, in-context example selection (Liu et al., 2021a) tries to find the best K examples for each inference-time input among all available examples. Although it would be desirable from a computational complexity standpoint to randomly choose in-context examples from the training set, a more principled approach to selecting these examples has been shown to be beneficial to performance (Alayrac et al., 2022; Liu et al., 2021a; Yang et al., 2022). As introduced in Section 2.3, RICES (Retrieval-based In-Context Example Selection; Yang et al. (2022)) has been shown to be effective at finding in-context examples that result in superior

few-shot performance relative to the random selection of in-context examples (Alayrac et al., 2022; Yang et al., 2022).

RICES method. For a given test input, $(\boldsymbol{x}^*, \boldsymbol{q}^*)$, similar examples from the training set are retrieved by averaging the similarity of their respective image embeddings and text embeddings, obtained from a pretrained image and text encoder, such as CLIP (Radford et al., 2021). More formally, the similarity between the test input and the i^{th} training example, $\{\boldsymbol{x}_i, \boldsymbol{q}_i, \boldsymbol{a}_i\}$, given a pretrained image encoder, $E_{image}(.)$, and text encoder, $E_q(.)$, is

$$sim(\{\boldsymbol{x}^*, \boldsymbol{q}^*\}, \{\boldsymbol{x}_i, \boldsymbol{q}_i, \boldsymbol{a}_i\}) = \frac{sim_{image} + sim_q}{2} \tag{4.2}$$

where

$$sim_{image} = d(E_{image}(\boldsymbol{x}^*), E_{image}(\boldsymbol{x}_i)) \quad \text{and} \quad sim_q = d(E_q(\boldsymbol{q}^*), E_q(\boldsymbol{q}_i)) \ ,$$

and $d(.,.)$ is a *similarity* metric in the embedding space of $E_{image}(.)$ and $E_q(.)$, such as cosine similarity. The K most similar training examples are then used as the in-context examples when evaluating the K-shot performance of the model. Since LMs are sensitive to the ordering in the prompt due to recency bias (Zhao et al., 2021), following the methodology of Alayrac et al. (2022), we order the examples so that the most similar in-context example appears right before the query.

Given the need for an image and text encoder when implementing RICES, an attractive characteristic of using the image encoder of a contrastive dual-encoder model, such as CLIP (Radford et al., 2021), as the VLM's image encoder (see Section 4.1.2) is that the dual-encoder can be reused for RICES. In particular, the dual-encoder's image encoder can be *reused* to extract the image embeddings needed for RICES and the dual-encoder's text encoder can be used to extract the necessary text embeddings.

4.3.4 Prompt ensembling

We also explore ensembling the outputs of the model across multiple prompts. This was shown in Alayrac et al. (2022) and Yang et al. (2022) to be beneficial to performance. Prompt ensembling can notably be combined with RICES where ensembling can be done over multiple permutations of the ranked nearest neighbours. Specifically, we permute the top K in-context examples r times, thus generating r answers. Then, following the methodology from Yang et al. (2022), we select the answer with the highest log-probability, among the r predicted answers, as the final answer.

4.4 Summary

In this chapter, we outlined the approach followed in this project. To the best of our knowledge, this is the first research that attempts to explicitly align the VQA task with a text-only closed-book QA task – that a VLM's frozen LM was pretrained on – in order to improve the few-shot performance of a VLM.

We started off by specifying the type of VLM architecture used and how this architecture relates to the other VLMs unified in Chapter 3. In particular, we showed that our VLM can be viewed as a modified version of Frozen (Tsimpoukelli et al., 2021) and ClipCap (Mokady et al., 2021), in that a learnt mapping network is used to embed the visual signal into the prompt of a frozen pretrained LM. We then elaborated on how this mapping network is trained, following the methodology of Frozen in order to make our results comparable. Finally, we provided the details for how the trained VLM can be used for few-shot evaluation via in-context learning. Specifically, we outlined how prompt templates can be used to explicitly align the VQA task with tasks that the frozen LM was pretrained on, in order to improve few-shot performance. Finally, we specified two other methods that we will use to potentially improve few-shot performance, namely: RICES for in-context example selection and prompt ensembling.

In the next chapter, we outline the setup of our experiments. Most notably, we describe how we will test the hypothesis that few-shot performance gains can be achieved for VLMs that use relatively small LMs by using template-based prompt formatting, explicitly aligning the VQA task with the LM's training tasks.

Chapter 5

Experimental Setup

This chapter outlines the setup of our experiments. We start by listing the specific models used as the architectural components of our VLM (Section 5.1). We then describe the details for training our VLM on a captioning dataset (Section 5.2). Finally, we outline the setup for our main experiments: evaluating whether using template-based prompt formatting, explicitly aligning the VQA task with our frozen pretrained LM's training tasks, can improve few-shot performance (Section 5.3).

5.1 Architectural components

In Section 4.1, we outlined the abstract architectural components of our visual language model (VLM) without specifying any particular instantiation of the components, so that the VLM can be implemented with many different models. In this section, we list the specific models used for the experiments.

5.1.1 Frozen pretrained LM

We use T0 (Sanh et al., 2021) from HuggingFace (Wolf et al., 2020) as the frozen pretrained encoder-decoder language model (LM). The reason for choosing T0 is that the examples used to train T0 were formatted by applying *prompt templates* from the Public Pool of Prompts (P3; Bach et al. (2022)), such that each example in each dataset was converted into a specific format defined by a template. Refer to Appendix A.1 for more information on how T0 was trained.

T0 utilises the SentencePiece (Kudo and Richardson, 2018) tokenizer to encode text as WordPiece (Kudo, 2018; Sennrich et al., 2015) tokens. A vocabulary of size 32 000 wordpieces is used. Each sequence of tokens $t_1, \ldots, t_k$ is mapped to a sequence of D-dimensional embeddings

before being fed into the model. T0 was released in 3B and 11B parameter variants, referred to as *T0-3B* and *T0pp*, respectively. We use T0-3B due to its relative simplicity. The internal dimension of T0-3B is $D = 1024$.

It must be noted that T0 is trained to do the following: given a formatted input from one of its pretraining tasks, it is prompted to generate a response. Thus, T0 is familiar with observing only *one* formatted input. Hence, T0 has been successfully applied in the zero-shot regime (Sanh et al., 2021), since this is the exact formatting it was exposed to during training. However, it is not certain how the performance will change when in-context examples are presented to the model, as the more in-context examples that are presented to the model at test time, the further this input is from T0's training data/distribution.

5.1.2 Frozen pretrained image encoder

We use the Vision Transformer (ViT; Dosovitskiy et al. (2020)) of a pre-trained CLIP (Radford et al., 2021) model as the frozen pretrained image encoder. In particular, we use the 307M parameter "ViT-L/14@336px" variant which maps each input image $\boldsymbol{x}$ of size 336x336 pixels to a 768-dimensional continuous image embedding. We favour "ViT-L/14@336px" over "ViT-L/14@224px" because using higher-resolution images at inference time has been shown to result in superior downstream performance (Jia et al., 2021; Radford et al., 2021; Yuan et al., 2021).

Frozen (Tsimpoukelli et al., 2021) trained their image encoder along with their mapping network from scratch on ∼3M examples. We choose to use a *frozen* CLIP-ViT because it was trained on many more examples (∼400M examples) which we assume will result in better image representations. Furthermore, freezing the image encoder allows us to extract all of the necessary image embeddings before training/inference and re-use them for different training/test runs and for RICES.

5.1.3 Mapping network and visual prefix

For the mapping network, we follow the methodology of ClipCap (Mokady et al., 2021) and utilise a single-hidden-layer MLP. The MLP maps each 768-dimensional image embedding to a $(D * n)$-dimensional vector which is then reshaped to form the visual prefix, $\boldsymbol{e}_1^*, \ldots, \boldsymbol{e}_n^*$, where $D = 1024$ is the internal dimension of T0-3B.

The optimal length of the visual prefix, n, has been shown to differ depending on the context: Frozen (Tsimpoukelli et al., 2021) and MAGMA (Eichenberg et al., 2021) found that a visual prefix of length 2 was optimal, whereas ClipCap (Mokady et al., 2021) used a length of 10. The length of the visual prefix has a significant impact on the number of trainable parameters. Thus,

in order to balance the number of trainable parameters while ensuring that the visual signal is adequately represented, we evaluate $n \in \{5, 10\}$. The number of trainable parameters for each configuration is shown in Table 5.1.

Table 5.1 **Number of trainable parameters.** The number of trainable parameters (i.e. the number of parameters in the mapping network) for different visual prefix lengths. Frozen (Tsimpoukelli et al., 2021) is included for comparison. * The number of trainable parameters in Frozen is not explicitly stated, thus, we obtained this value by estimating that the ResNet-50 and mapping network contribute 23 million and 17 million trainable parameters, respectively.

	Visual prefix length (n)	# Trainable parameters (million)
T0-3B	5	56
	10	217
Baseline:		
Frozen	2	40*

5.2 Training

We train the mapping network from scratch on Conceptual Captions (Sharma et al., 2018). This dataset contains $\sim$ 3.3M images annotated with captions harvested from the Alt-text HTML attribute associated with web images. Training was terminated when the validation loss reached a minimum. All training was run on an Nvidia A100 GPU. The batch sizes used, number of steps before terminating training, and training durations for each of the respective configurations are shown in Table 5.2. We use the Adam optimizer (Kingma and Ba, 2014) with a constant learning rate of 3×10^{-4}, and gradients are accumulated over every two steps. The models are implemented using the PyTorch and PyTorch Lightning frameworks (Falcon et al., 2019).

The ability of the trained VLMs to generate captions for images is qualitatively explored in Section 6.1.

Table 5.2 **Additional training details.** Additional training details for the different VLM configurations.

	Visual prefix length (n)	Batch size	Training steps	Training time (hrs)
T0-3B	5	64	3418	14
	10	64	2739	18

5.3 Few-shot VQA

We test the trained VLMs' ability to transfer from the captioning task to a VQA task using few-shot in-context learning.

5.3.1 Dataset

Visual Question Answering (VQA) tasks require the model to predict an answer to a question about an input image. Our chosen VQA task is the VQA2.0 dataset (Goyal et al., 2017), which contains 83k/41k/81k images, and 444k/214k/448k questions for training/validation/testing. Following Frozen (Tsimpoukelli et al., 2021) and PICa (Yang et al., 2022), we search over the training set for in-context examples and report the accuracy on the validation set. Instead of treating VQA as a classification task over a pre-selected answer vocabulary (e.g. Goyal et al. (2017); Li et al. (2020)), we predict the answer through open-ended text generation.

5.3.2 Evaluation metric

All questions are human-annotated with 10 concise, open-ended answers. In order to evaluate a particular predicted answer, the VQA score (Goyal et al., 2017) is used instead of the accuracy to account for the inter-human variability in the annotations:

$$score(ans) = \min\left\{\frac{\#\text{ humans that said ans}}{3}, 1\right\}.$$

All predicted answers are processed and evaluated using the evaluation script from the VQA API[1] (Goyal et al., 2017). It is noted that this evaluation metric may unfairly penalise open-ended answer generation since only exact matches with the human annotations are considered successful[2]. Despite this limitation, we still adopt the VQA score to be consistent with previous work.

5.3.3 Baselines

Although many visual language models (VLMs) have been developed in the last two years to solve this task, only a small subset have evaluated their VLMs via in-context learning (Alayrac et al., 2022; Eichenberg et al., 2021; Tsimpoukelli et al., 2021; Yang et al., 2022). We note

[1] https://github.com/GT-vision-lab/VQA.

[2] For example, if the answer "grey" is predicted, but all of the ground truth annotations have the answer "gray", then the predicted answer will get a VQA score of 0.

that Flamingo (Alayrac et al., 2022) and PICa (Yang et al., 2022) are not comparable with our approach since they both utilise significantly larger scale than our VLM: (1) Flamingo uses an 80B parameter frozen language model (Hoffmann et al., 2022a) and PICa uses GPT-3 (Brown et al., 2020) which has 175B parameters, whereas T0-3B only has 3B parameters; (2) Flamingo was trained on over 1B images, whereas we trained our VLM on only $\sim$ 3.3M images. MAGMA (Eichenberg et al., 2021) claims to evaluate their model on the VQA2.0 validation set using few-shot in-context learning. However, they include the VQA2.0 training set as part of their pretraining corpus, and thus their results are not truly few-shot. Thus, we use only Frozen (Tsimpoukelli et al., 2021) as our baseline.

Since a large proportion of the methodology followed in this project was inspired by Frozen, our methods are very similar: we condition a frozen pretrained language model (LM) on a learnt image-conditioned visual prefix. However, Frozen uses a 7B parameter decoder-only LM and learn the parameters of their mapping network *and* image encoder during training. Despite these differences in our approaches, the main modification we explore is the use of prompt templates to improve performance. Following the notation introduced in Section 4.3.1, Frozen uses the template

`<image>` Question: `{question}` Answer: `{answer}`

to format VQA2.0 examples, but their frozen LM has never explicitly seen this formatting before. In contrast, we use modified versions of the text-only closed-book QA templates seen by T0 during training (see Section 5.3.4).

5.3.4 Prompt templates

As outlined in Section 4.3.1, we investigate whether formatting the inputs used to prompt the VLMs with modified text-only closed-book QA prompt-templates can improve VQA performance. In order to evaluate this, we explore using two different templates:

hotpotqa. We modify one of the P3 (Bach et al., 2022) templates used to train T0 on the hotpotqa (Yang et al., 2018) task, by prepending the visual prefix to the text embedding.

```
<image>
Combine facts and answer this:
{question}
{answer}
```

frozen. The template used in Frozen. The text-only version of this format was not seen by T0-3B during pretraining.

```
<image>
Question: {question}
Answer: {answer}
```

To emphasize, the text-only version of the **hotpotqa** template was *seen* by T0 during pretraining, while the text-only equivalent of the **frozen** template was *not seen* during pretraining.

5.3.5 Text-only performance

We explore whether incorporating the visual prefix is adding any value, or whether, by aligning the VQA task with the text-only closed-book QA task, the visual prefix is being mainly disregarded. In order to test this, we remove the `<image>` token from the **hotpotqa** and **frozen** templates such that they are text-only. Notably, this means that this text-only **hotpotqa** template is exactly the same as the template used to train T0 on the hotpotqa task (Yang et al., 2018).

However, even when the visual prefix is removed, the visual signal will still influence the predicted answers as the visual signal informs the selection of in-context examples (see Eqn. (4.2)). Thus, we investigate two different text-only setups:

text-only prompt. The visual prefix is removed from the prompt, but the visual signal is used by RICES.

text-only prompt with text-only RICES. The visual prefix is removed from the prompt, *and* RICES selects in-context examples based on the question similarity only.

Unless otherwise mentioned, all results are reported with the visual prefix included in the prompt and with the default RICES implementation. The results are reported in Section 6.2.5.

5.3.6 RICES implementation

Implementing RICES for the VQA2.0 dataset is very computationally expensive. In order to find the most similar training examples for any given test input, following from Eqn. (4.2), we have to calculate the average similarity of the test input's question and image with every training example's question and image. Given our compute budget, we decided to implement an *approximate* k nearest neighbours (kNNs) algorithm, using a FAISS index (Johnson et al., 2019): a method for efficient similarity search over dense vectors. The details of this algorithm, and the motivation for why we chose to use an approximate algorithm, are given in Appendix A.2.

Although the algorithm is approximate, by qualitatively examining the in-context examples selected for a few random test inputs we found that the selected in-context examples are very

Fig. 5.1 **In-context example selection.** An illustration of the four in-context examples (comprised of an image and a question – the answer has been omitted) selected for a given test query (the bottom-most image and question), when using different in-context example selection methods. The methods used are: (a) RICES; (b) Random selection of in-context examples from the training set. For RICES, the in-context examples are sorted such that the closer the example is to the test query, the more similar it is.

similar to the test inputs with respect to both the image and question. For example, referring to Figure 5.1, when using RICES, the selected in-context examples and test input all have a skateboarder (with their board lifted off the ground) in the image, and all have a question relating to the "background" of the image. The in-context examples selected for additional test queries when using RICES is shown in Figure A.1.

We re-use the CLIP-ViT image encoder to obtain the necessary image embeddings, and use the same CLIP model's Transformer-based text encoder to obtain the necessary question embeddings.

5.3.7 Decoding and answer generation

As there are over 200k validation examples, in order to speed up evaluation, we use greedy decoding when generating answers. We also constrain the maximum length of the generated answers to 20 tokens in order to prevent degenerative, long predictions from slowing evaluation too significantly.

5.4 Conclusion

In this chapter, we outlined the experimental setup as well as the implementation details for training and evaluating our VLMs. We started off by specifying the particular architectural components for our VLMs (Section 5.1) before describing how the VLMs are trained (Section 5.2). We then expanded on the key experiments for this project, by introducing the task on which we evaluate our models, the baseline which we compare our results to, the specific prompt templates used, and the implementation details for in-context example selection and decoding (Section 5.3).

In the next chapter, we present the results for each of our experiments and discuss their implications.

Chapter 6

Results and Discussions

This chapter presents and discusses the experimental results. We start off by investigating the captions generated by the respective VLMs (Section 6.1) in order to glean insight into the ability of the visual prefix to capture the visual features of the image. The main results are then presented: evaluating the few-shot performance of the trained models on the VQA2.0 (Goyal et al., 2017) dataset (Section 6.2). We then summarise and discuss these results (Section 6.3), ending with the limitations of our approach (Section 6.4) and our recommendations for future work (Section 6.5).

6.1 Generated captions

Although it is not possible to directly interpret the visual prefix, we can get a sense of the visual information it contains by examining the output of the frozen LM when prompted with it. In order to generate captions, we freeze the whole VLM and prompt the frozen VLM with the visual prefix *only*[1]. Furthermore, since T0 (Sanh et al., 2021) was trained on summarisation tasks, we can prompt T0 to summarise the visual prefix in order to get a different perspective on the information contained within it. We use a modification of the prompt template used for one such summarisation task[2] to achieve this. Specifically, we replace the text that needs to be summarised with the visual prefix, such that the template is:

Summarize: `<image>`

[1] This resembles the "Training" setup in Figure 4.1, except that the mapping network is also frozen.

[2] The original prompt template is from P3 (Bach et al., 2022) for the Extreme Summarization (XSum) dataset (Narayan et al., 2018)

Finally, following the convention of Eichenberg et al. (2021); Radford et al. (2021), we pass the phrase "A picture of" to the **decoder** of T0-3B, while still passing the visual prefix to the encoder, and report the resulting captions.

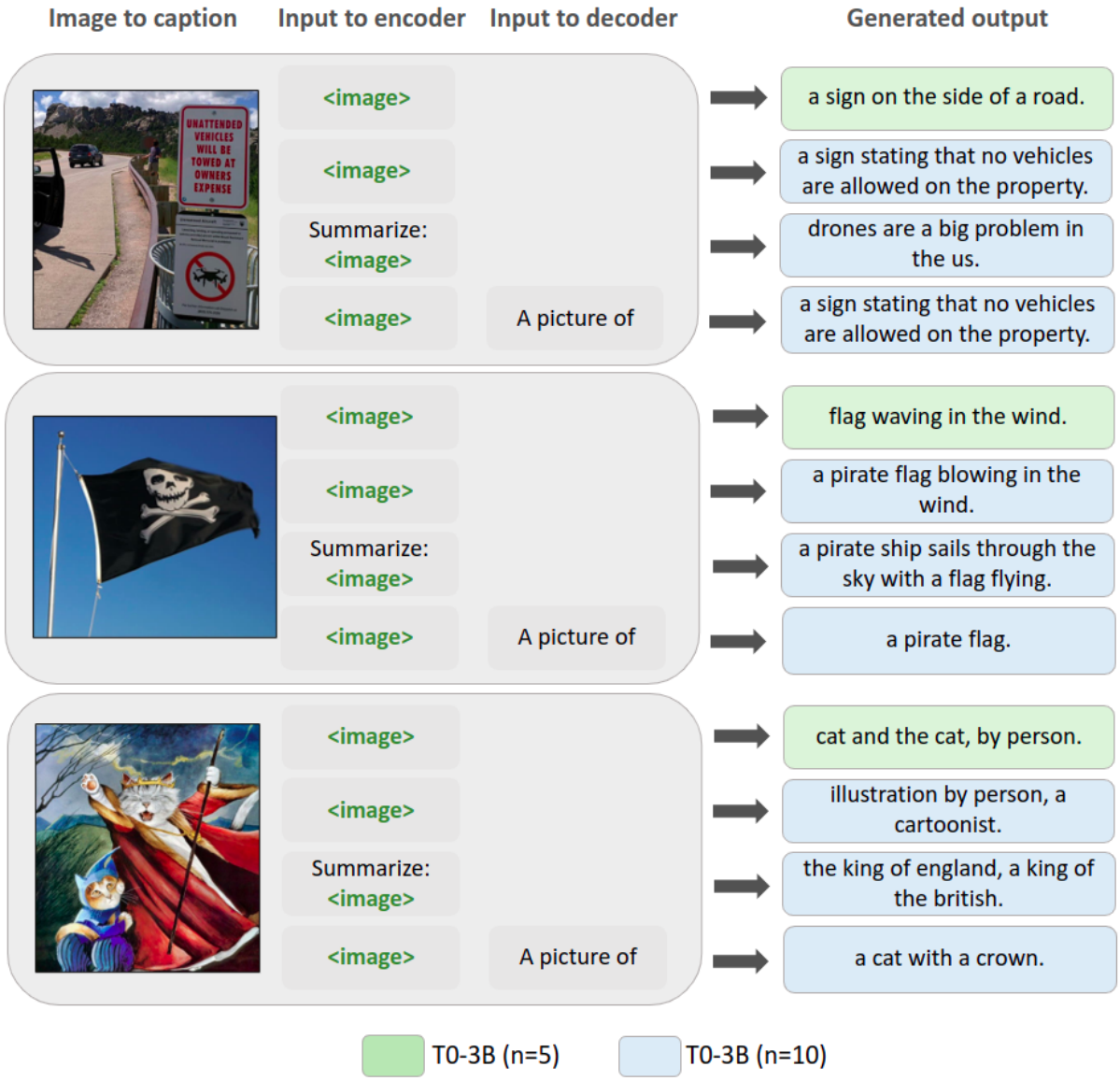


Fig. 6.1 **Generated captions and summaries.** The generated captions and summaries for three unseen images from the Conceptual Captions (Sharma et al., 2018) validation set for both VLM configurations, where n represents the visual prefix length. We specify the input to the encoder and decoder of T0-3B, respectively, that is used to obtain the generated captions.

The captions and summaries generated by each of the VLM configurations for unseen images are shown in Figure 6.1. From the captions that are generated when only the visual prefix is used to prompt the model, it is evident that, for both visual prefix lengths, there are visual concepts from the images that are contained in the visual prefix. In particular, it seems that the visual prefix may contain information relating to objects (e.g. "road", "sign", "pirate flag", "cat"), how

they relate to each other (e.g. "a sign on the *side* of the road"), and the actions associated with the objects (e.g. "waving", "blowing"). However, there are also instances where the captions are not very descriptive of the image (e.g. "illustration by person, a cartoonist"), which may suggest that, in these instances, the visual prefix either does not contain the relevant information, or does not prompt T0-3B effectively with the visual information.

The summaries of the visual prefix (i.e. when "Summarize: `<image>`" is passed to the encoder) reveal that there is potentially more visual information embedded in the visual prefix than the captions suggest. The summary of the first image identifies that there is a drone in the image and that there is a negative association with them. In the final image, the summary contains "king", suggesting that the royalty of the cat (either through the crown or general attire) is potentially embedded in the visual prefix.

Thus, although the visual prefix is not interpretable, and hence we cannot know for certain how it is prompting the LM, it seems plausible that it captures a number of visual features from the images. Furthermore, from the reasonable fluency of the generated captions, it appears that the visual prefix is able to incorporate the visual signal into the prompt while keeping the strong text-generation abilities of T0-3B mainly intact.

6.2 Visual Question Answering results

We start off by presenting the project's main results: that explicitly aligning the VQA2.0 (Goyal et al., 2017) task with a task that T0-3B was trained on, via a prompt template, results in superior VQA performance (Section 6.2.1). We then explore these results further, by examining the effect of the prompt template on zero-shot (Section 6.2.2) and few-shot (Section 6.2.3) performance. Using only T0-3B ($n = 10$) with the **hotpotqa** template, we then investigate how important the in-context example selection method is (Section 6.2.4), whether the visual prefix is adding any value (Section 6.2.5), and whether prompt ensembling can be used to improve few-shot performance (Section 6.2.6).

6.2.1 Explicit alignment improves performance

In Table 6.1, we present our best VQA scores when using the **hotpotqa** and **frozen** templates, and compare these results to Frozen (Tsimpoukelli et al., 2021). These results imply the following:

- Comparing rows 1 & 2, explicitly aligning the VQA2.0 (Goyal et al., 2017) task with the text-only hotpotqa (Yang et al., 2018) task (which T0-3B was trained on) results in a 31%

Table 6.1 **Best VQA scores.** The best VQA scores obtained when using the **hotpotqa** and **frozen** templates, compared to the best results reported in Frozen (Tsimpoukelli et al., 2021).

		Template	# Shots	VQA score (%)
1	T0-3B ($n = 10$)	hotpotqa	1	**40.39**
2		frozen	1	30.83
3	Frozen	-	4	38.2

relative increase (30.83% vs 40.39%) in the VQA score compared to the best VQA score when using the **frozen** template.

- Comparing rows 1 & 3, the explicit alignment results in our best VLM outperforming Frozen by a relative VQA score of 5.7% (38.2% vs 40.39%), despite Frozen using a frozen LM with more than double the number of parameters (3B vs 7B).
- Furthermore, we outperform Frozen with fewer shots (1 vs 4). This is beneficial since performing in-context learning with Transformers is very computationally expensive: the compute cost scales linearly with the number of shots, if one can reuse the few-shot prompt for multiple test queries (by caching the keys and values), and quadratically otherwise (Alayrac et al., 2022).

6.2.2 Explicit alignment improves zero-shot performance

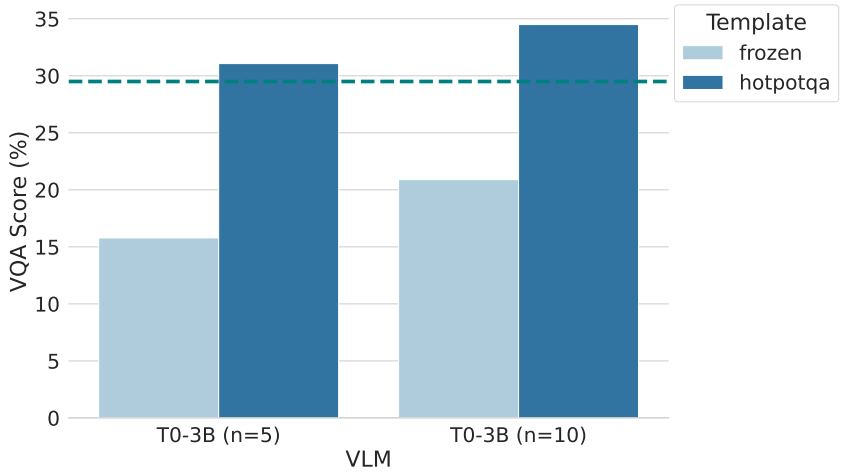


Fig. 6.2 **Zero-shot results.** The zero-shot results for the VLM configurations, when using the **frozen** and **hotpotqa** templates. The green dashed line indicates the zero-shot VQA score of 29.5% achieved by Frozen (Tsimpoukelli et al., 2021).

We present the zero-shot results in Figure 6.2. Evidently, for both visual prefix lengths, using the **hotpotqa** template improves the VQA score relative to the **frozen** template. These differences are significant, with a 96% (15.78% vs 31.07%) and 65% (20.89% vs 34.49%) relative increase in the VQA score when a visual prefix of length 5 and 10 are used, respectively. Furthermore, the performance when using the **hotpotqa** template exceeds Frozen's zero-shot performance, with T0-3B ($n = 10$) obtaining a 17% relative increase (29.5% vs 34.49%) and T0-3B ($n = 5$) obtaining a 5% relative increase (29.5% vs 31.07%).

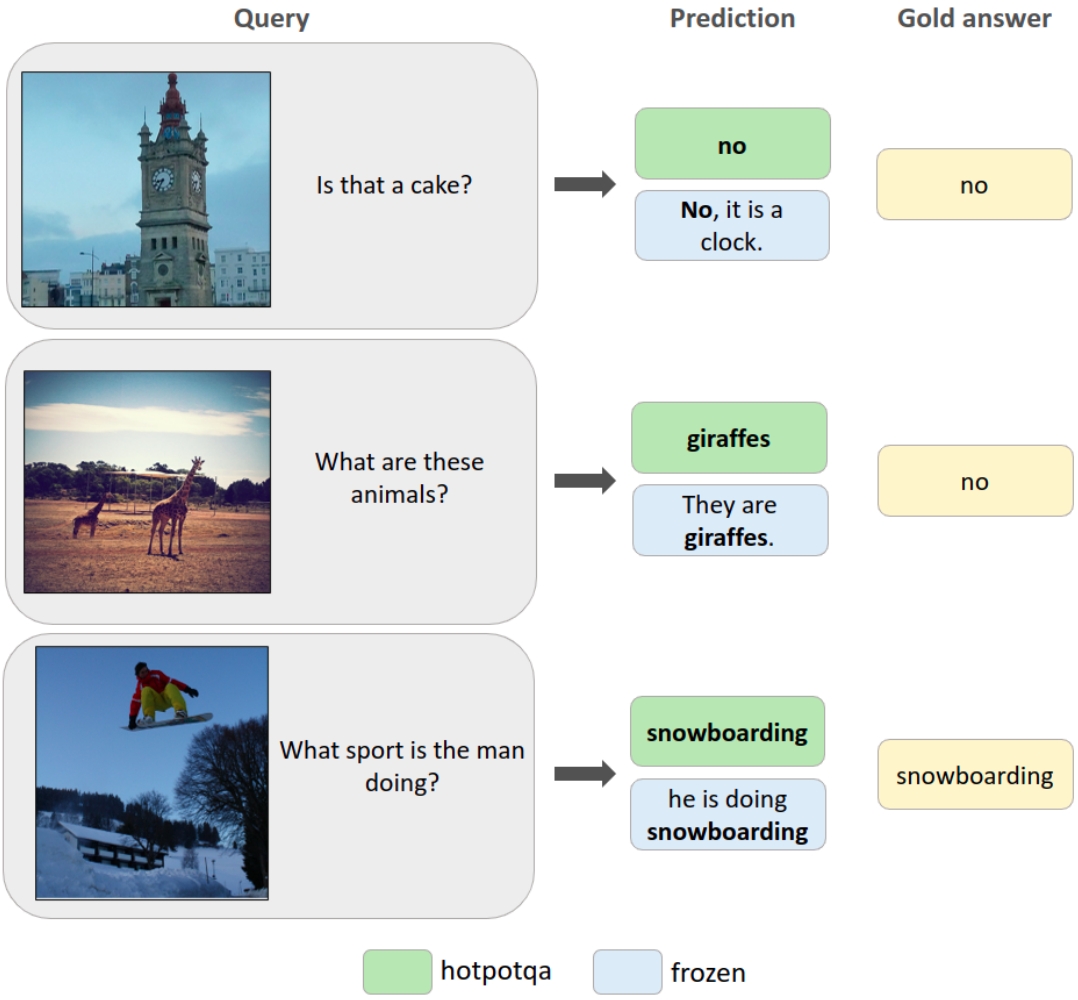


Fig. 6.3 **Zero-shot generated answers.** The zero-shot generated answers for three inputs from the test set, when using the **hotpotqa** and **frozen** templates. T0-3B ($n = 10$) was used to generate the answers.

Referring to the generated answers in Figure 6.3, a possible reason for the performance gain when using the **hotpotqa** template over the **frozen** template is that the former template is more likely to induce *more concise* predicted answers, such that they better match the expected output. With reference to Figure 6.3, the answers generated using the **frozen** template are sometimes correct (with some of them even containing correct justifications) but do not conform to the short answers expected for the VQA2.0 task. Conversely, since the hotpotqa (Yang et al., 2018)

task has similarly short answers, when the **hotpotqa** template is used the answers are more concise. This claim is supported by Figure 6.4, where the distribution of the number of words in the generated answers when using the **hotpotqa** template has relatively more mass around 1- and 2-word answers, whereas the distribution when using the **frozen** template has a much heavier right tail.

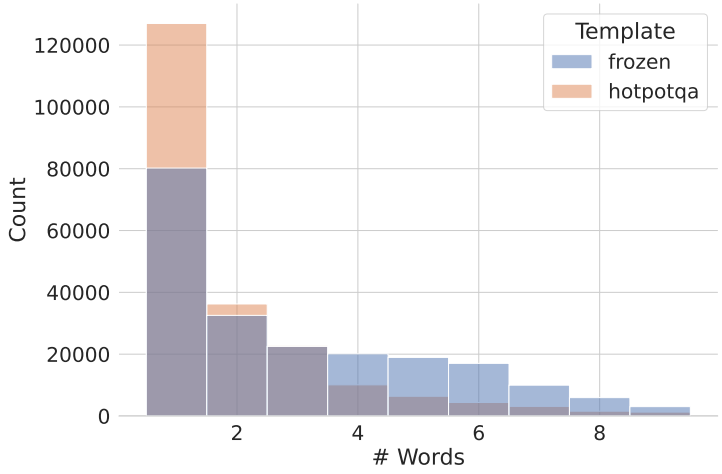


Fig. 6.4 **The number of words in the generated answers.** A histogram showing the distribution of the number of words in the zero-shot generated answers, when using the **frozen** and **hotpotqa** templates. To improve the visualisation, we have only included answers that have less than 10 words and omitted the irregular answers with more than 10 words.

Due to the superior zero-shot performance when a visual prefix of length $n = 10$ is used, we proceed with this VLM for the remainder of the experiments.

6.2.3 Explicit alignment improves few-shot performance

We present the few-shot results in Figure 6.5. The **hotpotqa** template once again consistently outperforms the **frozen** template by a significant margin, irrespective of the number of shots. Furthermore, the 1- and 2-shot performance when using the **hotpotqa** template is comfortably higher than the Frozen benchmark, but the 4-shot performance of the Frozen benchmark marginally outperforms the **hotpotqa** template.

Interestingly, for both the **hotpotqa** and **frozen** template, the 1-shot performance is significantly higher than the zero-shot performance, but the VQA score declines as more shots are added. When looking at the few-shot generated output, it is apparent that in many instances, the LM is simply selecting an answer from the in-context examples. Furthermore, if we simply replace each generated answer with the answer from the most similar in-context example to the test query (based on Eqn. (4.2)), the resulting VQA score of 36.97% is surprisingly impressive, surpassing the score for the **frozen** template by 6.14% (see Table 6.2). This is not true for the

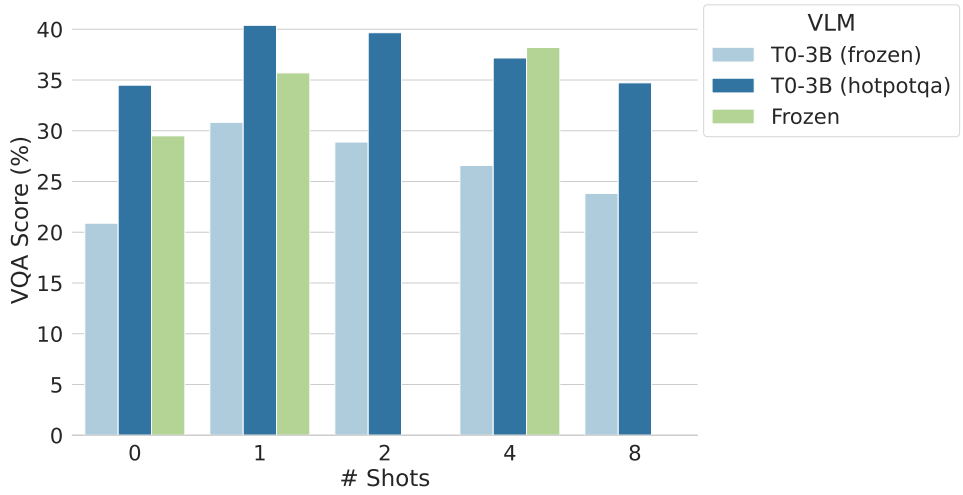


Fig. 6.5 **Few-shot results.** The VQA score for T0-3B ($n = 10$), when in-context examples (shots) are presented to the model. We compare using the **frozen** and **hotpotqa** templates, and compare our results to Frozen (Tsimpoukelli et al., 2021).

hotpotqa template, where replacing the original generated answers with the nearest in-context example's answer would significantly decrease the VQA score from 40.39% to 36.97%. In order to examine the impact of the prompt template, we need to account for what proportion of the changes in performance are attributed to simply selecting answers from the in-context examples, and what proportion represents the model's ability to generate original answers given the in-context examples.

Table 6.2 **1-shot VQA scores compared to RICES baseline.** The 1-shot VQA scores for the **hotpotqa** and **frozen** templates compared to the RICES baseline. This baseline predicts answers by simply selecting the answer from the most similar in-context example to the test query (i.e. the nearest neighbour's answer). We call this baseline RICES since the RICES method is effectively predicting the answer.

		Template	# Shots	VQA score (%)
1	T0-3B ($n = 10$)	hotpotqa	1	**40.39**
2		frozen	1	30.83
3	RICES	-	1	36.97

In Table 6.3, we show the breakdown of the 1-shot VQA scores, separating the results by whether the generated answer is the same as the answer presented in the in-context example, or whether the generated answer is original. Notably, the answers generated by the VLM with the **hotpotqa** and **frozen** templates are the same answer as the in-context example used to prompt the LM 63.4% and 45.4% of the time, respectively. Comparing rows 1&2 and rows 4&5, it is also apparent that most of the improvement in the 1-shot performance relative to the zero-shot performance is through generating answers that are the same as the relevant in-context example's

answer. This suggests that the RICES method is good at selecting examples from the training set that are very similar (both with regard to the image and question) to the test inputs, and hence, are relatively likely to contain the correct answer.

Table 6.3 **Breakdown of the 1-shot VQA score.** The 1-shot VQA scores broken down by answer type. (1) *in-context*: the generated answers that are the same as the in-context example's answer. (2) *original*: the generated answers that are not the same as the in-context example's answer. (3) *all*: the overall VQA score for the template (i.e. with the *in-context* and *original* answers combined). We also report the proportion of the generated answers that are *in-context* and *original*. The final column shows how the VQA score, over the test queries for which an original answer was generated, changes when the original generated answers are replaced by the relevant in-context example's answer.

	Template	Answer type	Proportion (%)	VQA score (%)	In-context VQA score (%)
1	hotpotqa	original	36.6	30.6	21.24
2		in-context	63.4	46.04	-
3		all	100	40.39	-
4	frozen	original	54.6	20.48	31.72
5		in-context	45.4	43.26	-
6		all	100	30.83	-

Continuing with our analysis of Table 6.3, looking at the VQA score for the answers that were original (i.e. not the same as the in-context example's answer), the answers produced when using the **hotpotqa** template were significantly better than the answers produced by the **frozen** template (30.6% vs 20.48%). However, these scores are calculated over different subsets of the test set, and thus are not directly comparable.

Looking at this result from a different perspective, it appears that when the **hotpotqa** template is used, the VLM is better at "deciding" when the in-context example's answer should be used as the output, and when the model should generate an original response, as it obtained a higher VQA score for the answers that were the same as the in-context examples' answers than when the **frozen** template is used (46.04% vs 43.26%). Furthermore, for the test queries where the VLM did generate an original answer, the in-context examples' answers would have performed relatively poorly (only 21.24% VQA score), compared to the actual generated answers (30.6% VQA score) when the **hotpotqa** template is used. The opposite is true for the **frozen** template.

We do not deem it beneficial for the purposes of this project to continue this analysis of the broken-down VQA scores when the number of shots is increased, as we suspect that similar trends will be observed. Thus, we conclude that it is empirically true that using the **hotpotqa** template results in a superior few-shot VQA score relative to the **frozen** template, but the reasons for this seems to be attributed to two confounding factors:

- The **hotpotqa** original answers appear to perform better than the **frozen** original answers. But this is dependent on the test queries for which the VLM generates original answers.
- The VLM, when using the **hotpotqa** template, appears to be better at "deciding" when to generate an original answer and when to simply use the most similar in-context example's answer.

6.2.4 The selection of good in-context examples is important

The influence of the in-context examples is demonstrated further by using randomly selected in-context examples in place of the RICES in-context examples. The results are shown in Figure 6.6. Evidently, the VQA score significantly decreases, even below the zero-shot performance. Thus, although the few-shot performance when using the **hotpotqa** template exceeds the Frozen benchmark, the performance of the VLM is significantly influenced by the ability of RICES to select good in-context examples.

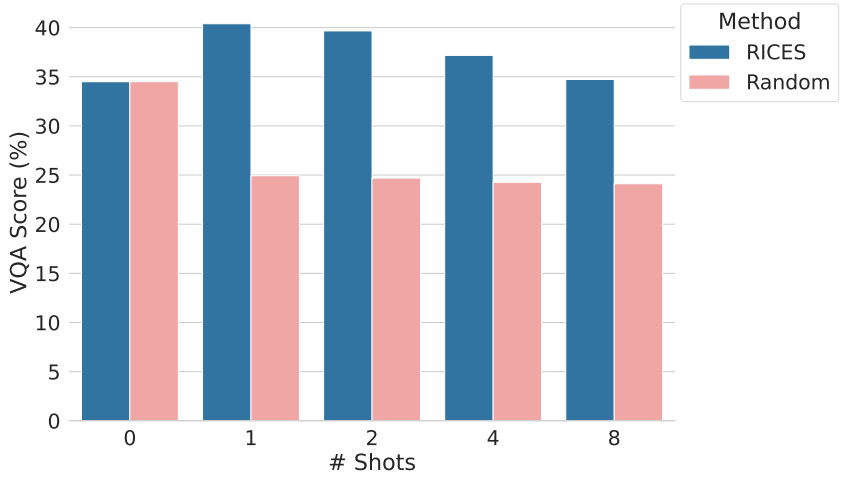


Fig. 6.6 **Few-shot results with different in-context example selection methods.** The influence of the in-context example selection method on the VQA score for T0-3B ($n = 10$) when the **hotpotqa** template is used. We compare the performance of RICES to the random selection of in-context examples from the training set.

6.2.5 The visual prefix is informative

Although we found that the VLMs' answers are heavily dependent on the in-context examples, the results in Figure 6.7 suggest that the visual prefix is still beneficial to the VQA score, especially when no shots are presented to the model. Specifically, comparing when the **text-only prompt** is used to the default configuration, we see that the default configuration achieves

significantly higher zero-shot performance (34.49% vs 27.3%) and marginally better few-shot performance.

However, if we remove the visual prefix *and* remove the visual signal from the RICES similarity calculations (referred to as **text-only prompt with text-only RICES**), we see that the default configuration obtains a consistently superior few-shot VQA score. This is consistent with the findings of the previous section, that the few-shot performance of the VLMs is highly dependent on the in-context examples, and thus, in the few-shot setting, the visual signal mostly influences the VLMs through the selection of in-context examples.

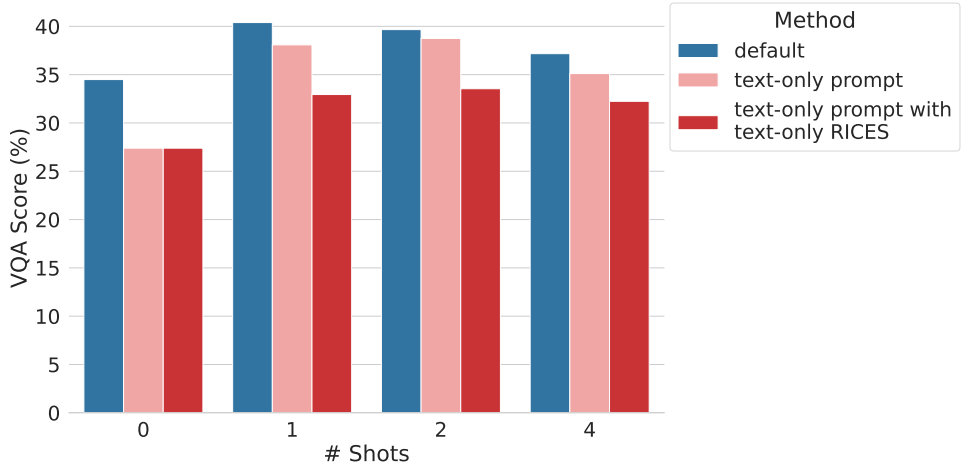


Fig. 6.7 **The influence of the visual signal.** The influence of the extent to which the visual signal is incorporated into the VLM, when using the **hotpotqa** template. Referring to Section 5.3.5, **text-only prompt** denotes when the visual prefix is removed from the prompt and **text-only prompt with text-only RICES** denotes when the visual prefix is removed from the prompt, *and* RICES selects in-context examples based on the question similarity only. Lastly, **default** denotes the default configuration, where the visual prefix is not removed and RICES is implemented with the image and question similarity.

6.2.6 Prompt ensembling does not significantly improve performance

In order to examine the effect of the ordering of the in-context examples on the few-shot performance, we use prompt ensembling, following the method outlined in Section 4.3.4. In particular, we permute the top K in-context examples r times, thus generating r answers, and select the answer with the highest log-probability. We choose r to be 5.

Looking at the results in Figure 6.8, it is evident that ensembling the prompts had very little impact on performance. Thus, either the ordering of the shots within the prompt is not very important, or the default ordering chosen (i.e. ordering the in-context examples such that the more similar the in-context example is to the test query, the closer it is situated to the test-query) is predominantly the best ordering among the permutations.

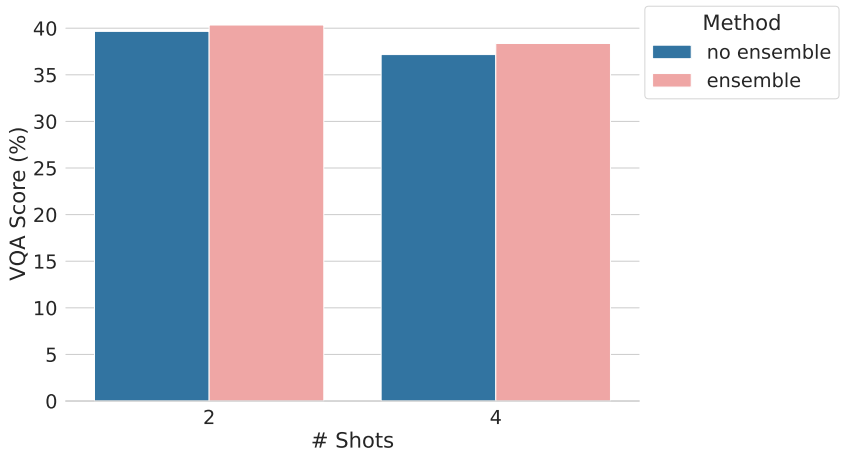


Fig. 6.8 **Few shot results with ensembling.** The effect of prompt ensembling on the few-shot performance of T0-3B ($n = 10$), when using the **hotpotqa** template, is shown.

6.3 Discussion

We showed in Section 6.2.1 that our best performing VLM outperforms the best results from the Frozen baseline, despite needing relatively few shots. However, the zero-shot results most clearly demonstrate that the project's goal were achieved (see Section 6.2.2), while the confounding effect of the in-context examples on few-shot performance makes drawing any concrete conclusions about the few-shot performance difficult (see Section 6.2.3). We will reiterate the zero-shot and few-shot results in order to justify these claims.

6.3.1 Zero-shot summary

It is evident from Section 6.2.2, that explicitly aligning the VQA2.0 task with the text-only hotpotqa (Yang et al., 2018) task (which T0-3B was pretrained on) significantly improved the zero-shot VQA score relative to the Frozen (Tsimpoukelli et al., 2021) baseline and relative to when the **frozen** template was used. This is particularly impressive given that the frozen language model used in Frozen is more than double the size of T0-3B.

A possible reason for this is that the VQA2.0 task expects short, concise answers, and so does the hotpotqa task, thus aligning these tasks generally results in more concise generated answers, improving performance. This was shown empirically in Figure 6.2 and Figure 6.4, and qualitatively in Figure 6.3.

6.3.2 Few-shot summary

With reference to Section 6.2.3, the performance when using the **hotpotqa** template exceeds the performance of the **frozen** template irrespective of the number of shots used, suggesting that the explicit alignment is beneficial for few-shot performance too. However, it is evident from Section 6.2.3, that a large proportion of the answers generated are simply selected from the in-context examples, and thus it is very hard to infer with certainty why the explicit alignment is improving few-shot performance.

Furthermore, it is evident that the performance of the VLM deteriorates as the number of shots is increased. We take the view that this is because T0 is not designed for in-context learning.

6.3.3 T0 is not designed for in-context learning

As noted in Section 6.2.3, irrespective of the template used, the few-shot performance decreases as the number of shots is increased. This is contrary to the empirical trends that have been observed when performing in-context learning (Alayrac et al., 2022; Brown et al., 2020; Yang et al., 2022), where one expects the performance to increase with the number of in-context examples, and then reach a plateau where additional shots do not improve performance.

Referring to Section 5.1.1, we argue that this could be because T0 was pretrained with only one input at a time, and thus the more in-context examples that are presented to the model at test time, the further this input is from the training data/distribution. This could explain why, instead of generalising from the in-context examples, the model appears to be overfitting to the in-context examples, and thus the performance is deteriorating as the number of shots increases.

This training setup is contrary to models such as GPT-3 (Brown et al., 2020) and Flamingo (Alayrac et al., 2022), which have been shown to be strong in-context learners. These models were trained with multiple inputs at a time via "packing"[3], where the input sequence can be comprised of multiple different training inputs that require completions. The different training inputs within the input sequence are delimited with a special end of text token, instead of specifying attention masks that restrict the completion for a particular input to only be dependent on the relevant input. Thus, these models are used to having to make completions when prompted with a number of different inputs, potentially making them more suitable for in-context learning.

[3]In fact, T0 is also pretrained using packing, but instead of using special end of text tokens like Flamingo (Alayrac et al., 2022) and GPT-3 (Brown et al., 2020), T0 uses attention masks to ensure that the completion for a particular output can only attend to the relevant input.

It may be that the regime used to train T0 has resulted in T0 being unsuitable for in-context learning. However, this needs to be explored further in order to justify these claims, and is left for future work.

6.4 Limitations

Here, we describe some limitations of our approach.

In-context learning not improving performance. As described in Section 6.3.3, we believe that T0 is not suitable for in-context learning. Thus, although we use in-context learning as our "go-to" few-shot learning method, this was possibly a misguided decision.

Inheriting bias and fairness issues from T0. Like most large language models, the T0 family of models sometimes exhibit the undesirable social biases represented in the large corpus they are pre-trained on (see the T0 paper for more details; Sanh et al. (2021)). Since we make use of a frozen T0 model to generate answers, we inherit this undesirable behaviour.

6.5 Future work

Possibilities for future work include:

- Aligning other vision and language tasks with text-only tasks that the frozen language model has been pretrained on, via an appropriately modified prompt template, either during evaluation or *during training*. For example, the captioning task could be cast as a summarisation task (the summaries generated in Section 6.1 serve as a proof-of-concept) and the multiple choice visual question-answering task can be cast as a text-only multiple choice task. We have shown that explicit alignment *at evaluation time* can be beneficial to performance, but it would be interesting to investigate how alignment *during training* impacts training and downstream performance.
- Repeating our methodology and experiments, but using a frozen language model that has been shown to be effective at in-context learning.

Chapter 7

Conclusion

We have demonstrated that a visual language model's (VLM) few-shot Visual Question Answering (VQA) performance can be improved by explicitly aligning the VQA task with a text-only closed-book question answering (QA) task that the VLM's frozen language model (LM) was pretrained on. Our results show that this explicit alignment enables our VLMs to outperform a similar VLM that uses a considerably larger frozen LM. In order to test our claims, we implemented a simple architecture based on Frozen (Tsimpoukelli et al., 2021) and ClipCap (Mokady et al., 2021), whereby, through image captioning, the VLM learnt to bridge powerful pretrained vision-only and language-only models via a relatively simple learnt mapping network. Furthermore, we contextualised our approach relative to existing work by presenting a unified view of VLMs.

We have further shown that, when prompting our frozen LM with examples *in-context*, the generated answers are often simply selected from the in-context examples. We note that this irregular behaviour may be a sign of the unsuitability of our chosen LM for in-context learning, and recommend that future research considers utilising a frozen LM that has been shown to be effective at few-shot in-context learning.

Due to our VLMs' reliance on the in-context examples, we also demonstrated that the selection of in-context examples is very influential on our VLMs' few-shot results. We empirically justified that the in-context example selection method that we implemented is particularly beneficial to VQA performance, by showing that a model that simply selects the answer from the nearest in-context example to each test query obtains a very impressive VQA score.

We also presented a qualitative evaluation of the ability of the mapping network to embed visual information into the prompt of the LM by examining the captions generated for unseen images, as well as through "summarising" this visually-conditioned prompt. We found that

visual features, such as objects, the relation between objects, and the actions associated with objects, may be embedded in this learnt prompt.

Our findings suggest that explicitly aligning vision and language tasks with text-only training tasks is a simple yet promising approach to improving the performance of VLMs that are of modest scale. We encourage future research into aligning other vision and language tasks, such as image captioning, with text-only tasks in an attempt to decrease the performance differences between relatively small and large VLMs.

References

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al. (2022). Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*.

Bach, S. H., Sanh, V., Yong, Z.-X., Webson, A., Raffel, C., Nayak, N. V., Sharma, A., Kim, T., Bari, M. S., Fevry, T., Alyafeai, Z., Dey, M., Santilli, A., Sun, Z., Ben-David, S., Xu, C., Chhablani, G., Wang, H., Fries, J. A., Al-shaibani, M. S., Sharma, S., Thakker, U., Almubarak, K., Tang, X., Tang, X., Jiang, M. T.-J., and Rush, A. M. (2022). Promptsource: An integrated development environment and repository for natural language prompts.

Brock, A., De, S., Smith, S. L., and Simonyan, K. (2021). High-performance large-scale image recognition without normalization. In *International Conference on Machine Learning*, pages 1059–1071. PMLR.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Chen, J., Guo, H., Yi, K., Li, B., and Elhoseiny, M. (2022). Visualgpt: Data-efficient adaptation of pretrained language models for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18030–18040.

Cho, J., Lei, J., Tan, H., and Bansal, M. (2021). Unifying vision-and-language tasks via text generation. In *International Conference on Machine Learning*, pages 1931–1942. PMLR.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Du, Y., Li, C., Guo, R., Yin, X., Liu, W., Zhou, J., Bai, Y., Yu, Z., Yang, Y., Dang, Q., et al. (2020). Pp-ocr: A practical ultra lightweight ocr system. *arXiv preprint arXiv:2009.09941*.

Eichenberg, C., Black, S., Weinbach, S., Parcalabescu, L., and Frank, A. (2021). Magma–multimodal augmentation of generative models through adapter-based finetuning. *arXiv preprint arXiv:2112.05253*.

Falcon et al., W. (2019). Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3.

Gao, F., Ping, Q., Thattai, G., Reganti, A., Wu, Y. N., and Natarajan, P. (2022). A thousand words are worth more than a picture: Natural language-centric outside-knowledge visual question answering. *arXiv preprint arXiv:2201.05299*.

Gao, T., Fisch, A., and Chen, D. (2020). Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., and Parikh, D. (2017). Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913.

Gui, L., Wang, B., Huang, Q., Hauptmann, A., Bisk, Y., and Gao, J. (2021). Kat: A knowledge augmented transformer for vision-and-language. *arXiv preprint arXiv:2112.08614*.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. (2022a). Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. (2022b). Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Jia, C., Yang, Y., Xia, Y., Chen, Y.-T., Parekh, Z., Pham, H., Le, Q., Sung, Y.-H., Li, Z., and Duerig, T. (2021). Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pages 4904–4916. PMLR.

Jiang, Z., Xu, F. F., Araki, J., and Neubig, G. (2020). How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Johnson, J., Douze, M., and Jégou, H. (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kudo, T. (2018). Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*.

Kudo, T. and Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Lester, B., Al-Rfou, R., and Constant, N. (2021a). The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Lester, B., Al-Rfou, R., and Constant, N. (2021b). The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Levine, Y., Dalmedigos, I., Ram, O., Zeldes, Y., Jannai, D., Muhlgay, D., Osin, Y., Lieber, O., Lenz, B., Shalev-Shwartz, S., et al. (2022). Standing on the shoulders of giant frozen language models. *arXiv preprint arXiv:2204.10019*.

Li, J., Li, D., Xiong, C., and Hoi, S. (2022). Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. *arXiv preprint arXiv:2201.12086*.

Li, J., Selvaraju, R., Gotmare, A., Joty, S., Xiong, C., and Hoi, S. C. H. (2021). Align before fuse: Vision and language representation learning with momentum distillation. *Advances in neural information processing systems*, 34:9694–9705.

Li, X., Yin, X., Li, C., Zhang, P., Hu, X., Zhang, L., Wang, L., Hu, H., Dong, L., Wei, F., et al. (2020). Oscar: Object-semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision*, pages 121–137. Springer.

Li, X. L. and Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Liu, J., Shen, D., Zhang, Y., Dolan, B., Carin, L., and Chen, W. (2021a). What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. (2021b). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Liu, Y., Wei, W., Peng, D., and Zhu, F. (2022). Declaration-based prompt tuning for visual question answering. *arXiv preprint arXiv:2205.02456*.

Lu, Y., Bartolo, M., Moore, A., Riedel, S., and Stenetorp, P. (2021). Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*.

Luo, Z., Xi, Y., Zhang, R., and Ma, J. (2022). Vc-gpt: Visual conditioned gpt for end-to-end generative vision-and-language pre-training. *arXiv preprint arXiv:2201.12723*.

Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., and Zettlemoyer, L. (2022). Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*.

Mishra, S., Khashabi, D., Baral, C., and Hajishirzi, H. (2021). Natural instructions: Benchmarking generalization to new tasks from natural language instructions.

Mokady, R., Hertz, A., and Bermano, A. H. (2021). Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734*.

Narayan, S., Cohen, S. B., and Lapata, M. (2018). Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *ArXiv*, abs/1808.08745.

Press, O., Smith, N. A., and Lewis, M. (2021). Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2021). Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

Reynolds, L. and McDonell, K. (2021). Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–7.

Roberts, A., Raffel, C., and Shazeer, N. (2020). How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*.

Sanh, V., Webson, A., Raffel, C., Bach, S. H., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Scao, T. L., Raja, A., et al. (2021). Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Sennrich, R., Haddow, B., and Birch, A. (2015). Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Sharma, P., Ding, N., Goodman, S., and Soricut, R. (2018). Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565.

Shen, S., Li, L. H., Tan, H., Bansal, M., Rohrbach, A., Chang, K.-W., Yao, Z., and Keutzer, K. (2021). How much can clip benefit vision-and-language tasks? *arXiv preprint arXiv:2107.06383*.

Shin, T., Razeghi, Y., Logan IV, R. L., Wallace, E., and Singh, S. (2020). Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

Song, H., Dong, L., Zhang, W.-N., Liu, T., and Wei, F. (2022). Clip models are few-shot learners: Empirical studies on vqa and visual entailment. *arXiv preprint arXiv:2203.07190*.

Sung, Y.-L., Cho, J., and Bansal, M. (2022). Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5227–5237.

Tsimpoukelli, M., Menick, J. L., Cabi, S., Eslami, S., Vinyals, O., and Hill, F. (2021). Multimodal few-shot learning with frozen language models. *Advances in Neural Information Processing Systems*, 34:200–212.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, P., Yang, A., Men, R., Lin, J., Bai, S., Li, Z., Ma, J., Zhou, C., Zhou, J., and Yang, H. (2022). Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. *arXiv preprint arXiv:2202.03052*.

Wang, Z., Yu, J., Yu, A. W., Dai, Z., Tsvetkov, Y., and Cao, Y. (2021). Simvlm: Simple visual language model pretraining with weak supervision. *arXiv preprint arXiv:2108.10904*.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. (2021). Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yang, Z., Gan, Z., Wang, J., Hu, X., Lu, Y., Liu, Z., and Wang, L. (2022). An empirical study of gpt-3 for few-shot knowledge-based vqa. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 3081–3089.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. (2018). Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Yu, J., Wang, Z., Vasudevan, V., Yeung, L., Seyedhosseini, M., and Wu, Y. (2022). Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*.

Yuan, L., Chen, D., Chen, Y.-L., Codella, N., Dai, X., Gao, J., Hu, H., Huang, X., Li, B., Li, C., et al. (2021). Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*.

Zeng, A., Wong, A., Welker, S., Choromanski, K., Tombari, F., Purohit, A., Ryoo, M., Sindhwani, V., Lee, J., Vanhoucke, V., et al. (2022). Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*.

Zeng, Y., Zhang, X., and Li, H. (2021). Multi-grained vision language pre-training: Aligning texts with visual concepts. *arXiv preprint arXiv:2111.08276*.

Zhai, X., Wang, X., Mustafa, B., Steiner, A., Keysers, D., Kolesnikov, A., and Beyer, L. (2022). Lit: Zero-shot transfer with locked-image text tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18123–18133.

Zhang, P., Li, X., Hu, X., Yang, J., Zhang, L., Wang, L., Choi, Y., and Gao, J. (2021). Vinvl: Revisiting visual representations in vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5579–5588.

Zhao, Z., Wallace, E., Feng, S., Klein, D., and Singh, S. (2021). Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR.

Appendix A

A.1 T0

T0 (Sanh et al., 2021) is based on T5 (Raffel et al., 2020), a transformer-based encoder-decoder language model (Vaswani et al., 2017) that was pre-trained via a span-corruption objective on C4 (Raffel et al., 2019), a large corpus of unlabeled text data. T0 was created by fine-tuning a T5-variant[1] on a multitask mixture of datasets in order to enable zero-shot generalization, i.e. the ability to perform unseen tasks without any additional gradient-based training. The examples used to train T0 were formatted by applying prompt templates from the Public Pool of Prompts (P3; Bach et al. (2022)) such that each example in each dataset was converted into a specific format defined by a template (see the P3 repo[2] for some examples). Note that these templates are designed for text-only tasks.

A.2 RICES implementation details

Finding the k-nearest neighbours (KNNs) based on questions or images *only*, can be efficiently implemented with a FAISS index (Johnson et al., 2019): a data structure for efficient similarity search over dense vectors. However, finding the similarities *jointly* (i.e. averaged) is not supported by the FAISS index. Thus, an algorithm for efficiently finding the kNNs based on joint similarity is:

1. Using the frozen pretrained CLIP model (Radford et al., 2021) from Section 5.1.2, obtain embeddings for both the images and questions for each example, $\{\boldsymbol{x}_i, \boldsymbol{q}_i, \boldsymbol{a}_i\}$, in the VQA2.0 (Goyal et al., 2017) training set.

[1] Lester et al. (2021b) showed that T5's default span-corruption objective is not well-suited for training frozen models to be later conditioned by prompts. In order to resolve this, they "unlearnt" the span-corruption objective by continuing T5's self-supervised training for 100 000 additional steps, replacing the span-corruption objective with the standard language modelling objective, to obtain an *LM adapted T5* model.

[2] https://github.com/bigscience-workshop/promptsource

2. Build a FAISS index for all of these question-embeddings. Denote this index FAISS_q.
3. For each test input, $(\boldsymbol{x}^*, \boldsymbol{q}^*)$:
   (a) Use FAISS_q to find the question-only tNNs, and hence sim_q, for $\boldsymbol{q}^*$. Denote these nearest neighbours $t\text{NN}_q$.
   (b) Build a FAISS index for the image-embeddings of all $(\boldsymbol{x}_i, \boldsymbol{q}_i, \boldsymbol{a}_i) \in t\text{NN}_q$. Denote this index FAISS_{image}.
   (c) Use FAISS_{image} to find the image-only pNNs, and hence sim_{image}, for $\boldsymbol{x}^*$. Denote these nearest neighbours $p\text{NN}_{image}$.
   (d) Calculate $sim(\{\boldsymbol{x}^*, \boldsymbol{q}^*\}, \{\boldsymbol{x}_i, \boldsymbol{q}_i, \boldsymbol{a}_i\})$ for all $\{\boldsymbol{x}_i, \boldsymbol{q}_i, \boldsymbol{a}_i)\} \in p\text{NN}_{image}$.
   (e) Rank the similarities to obtain the RICES kNNs.

When searching the FAISS index on GPU, the most neareast neighbours that can be returned is 2048 (Johnson et al., 2019). Thus, we set $t = 2048$ as this will result in the most accurate approximation to the true nearest neighbours, and we set p to be the number of unique images in $t\text{NN}_q$.

A.3 RICES examples

In Figure A.1, we provide further qualitative support for the ability of RICES to select very similar in-context examples to any given test query, with respect to both the image and question similarity.

Fig. A.1 **RICES examples.** Each column shows the four in-context examples selected (starting from the top image-question pair) for a given test query (the bottom image-question pair) using RICES. The in-context examples are sorted such that the closer the example is to the test query, the more similar it is.