LLaMA-Adapter: Efficient Fine-tuning of Language Models with Zero-init Attention

Renrui Zhang\(^1,2\), Jiaming Han\(^1\), Aojun Zhou\(^2\), Xiangfei Hu\(^1\), Shilin Yan\(^1\)
Pan Lu\(^3\), Hongsheng Li\(^2\), Peng Gao\(^1\), Yu Qiao\(^1\)

\(^1\)Shanghai Artificial Intelligence Laboratory  \(^2\)CUHK MMLab  \(^3\)University of California, Los Angeles
\{zhangrenrui, hanjiaming, gaopeng, qiaoyu\}@pjlab.org.cn

Abstract

We present LLaMA-Adapter, a lightweight adaption method to efficiently fine-tune LLaMA into an instruction-following model. Using 52K self-instruct demonstrations, LLaMA-Adapter only introduces 1.2M learnable parameters upon the frozen LLaMA 7B model, and costs less than one hour for fine-tuning on 8 A100 GPUs. Specifically, we adopt a set of learnable adaption prompts, and prepend them to the input text tokens at higher transformer layers. Then, a zero-init attention mechanism with zero gating is proposed, which adaptively injects the new instructional cues into LLaMA, while effectively preserves its pre-trained knowledge. With efficient training, LLaMA-Adapter generates high-quality responses, comparable to Alpaca with fully fine-tuned 7B parameters. Furthermore, our approach can be simply extended to multi-modal input, e.g., images, for image-conditioned LLaMA, which achieves superior reasoning capacity on ScienceQA. We release our code at https://github.com/ZrrSkywalker/LLaMA-Adapter.

1. Introduction

Large-scale Language Models (LLMs) [6, 7, 35, 36, 54] have stimulated widespread attention in both academia and industry. Driven by massive corpora and advanced hardware, LLMs exhibit remarkable understanding and generative ability, propelling language tasks into a higher level. Recently, significant progress has been made on instruction-following models, e.g., ChatGPT\(^1\) and GPT-3.5 (text-davinci-003) [5]. Following instructions or commands in natural language, they can generate professional and contextual responses in a conversational way. However, the further prevalence of instruction-following models is largely impeded by the closed-source restriction and high development costs.

To alleviate this, Stanford Alpaca [40] proposes to fine-tune an LLM, i.e., LLaMA [41] into an instruction-following model, which is affordable and replicable. Starting from 175 human-written instruction-output pairs [47], Alpaca leverages GPT-3.5 to expand the training data to 52K in a self-instruct manner. Supervised by this, Alpaca fine-tunes the entire 7B parameters in LLaMA, producing an exceptional model that performs similarly to GPT-3.5. Despite Alpaca’s effectiveness, a complete fine-tuning of large-scale LLaMA is still time-consuming, computation-
intensive, and cumbersome to transfer to different downstream scenarios.

In this paper, we introduce **LLaMA-Adapter**, an efficient fine-tuning method that adapts LLaMA into a well-performed instruction-following model. We also utilize the 52K instruction-output data for training purposes, but demonstrate superior resource efficiency to Alpaca. Specifically, in LLaMA's higher transformer layers, we append a set of learnable adaption prompts as prefix to the input instruction tokens. These prompts learn to adaptively inject new instructions (conditions) into LLaMA. To avoid noise from adaption prompts at the early training stage, we modify the vanilla attention mechanisms at inserted layers to be zero-init attention, with a learnable gating factor. Initialized by zero vectors, the gating can firstly preserve the original knowledge in LLaMA, and progressively incorporate instructional signals during training. This contributes to stable learning during the fine-tuning process and better instruction-following capacity of the final model.

Overall, our LLaMA-Adapter exhibits four main characteristics, as shown in Figure 1.

- **1.2M Parameters.** Instead of updating the full 7B parameters, we freeze the pre-trained LLaMA and only learn the adaption prompts with 1.2M parameters on top. This, however, reveals comparable instruction-following proficiency with the 7B Alpaca.

- **One-hour Fine-tuning.** Thanks to lightweight parameters and our zero-init gating, the convergence of LLaMA-Adapter costs less than one hour on 8 A100 GPUs, three times faster than Alpaca.

- **Plug with Expertise.** For different scenarios, it is flexible to insert their respective adapters and endow LLaMA with different expert knowledge. Thus, it suffices to store a 1.2M adapter within each context, other than a complete copy of the 7B model.

- **Multi-modal Condition.** Besides textual instruction, LLaMA-Adapter can be extended to image input for multi-modal reasoning. By simply adding images tokens into adaption prompts, LLaMA-Adapter performs competitively on the ScienceQA benchmark.

2. Related Work

2.1. Instruction-Following Language Models

The subfield of Language Models focusing on instruction-following models is crucial for generating responses based on natural language commands. Instruction-following methods enhance pre-trained models by fine-tuning them using high-quality input-output tuples of task instructions and ground truth outputs. This fine-tuning helps the model better understand user intentions and follow instructions more accurately. Instruction-following methods have been extensively researched in language models [3, 32, 48, 50] and multi-modality domains [29, 38]. Among those methods, FLAN [50] introduces an instruction tuning method that outperforms non-tuned LLMs in unseen tasks. PromptSource [3] provides development environment and repository that offers a web-based GUI for creating and managing natural language prompts for zero-shot or gradient-based few-shot learning. SUP-NATINST [48] establishes a large benchmark of 1,616 diverse NLP tasks and uses multi-task training on the T5 model, demonstrates strong generalization capabilities on unseen tasks. InstructGPT [32] demonstrates significant performance improvements and may be integrated into closed-source models like GPT-3.5 and GPT-4 [31]. The open-source Stanford-Alpaca [40] approach fine-tunes all parameters of LLMs in an end-to-end manner. However, this full-model fine-tuning can be computationally intensive and challenging to scale to larger pre-trained language models. In contrast, our paper aims to fine-tune lightweight adapters on top of the frozen large pre-trained models, e.g., LLaMA, rather than performing end-to-end fine-tuning of all parameters. This approach reduces computational demands and facilitates the efficient adaptation of LLMs to instruction-following tasks while maintaining high performance.

2.2. Large Vision-Language Models

Over the past decade, we have witnessed a shift in vision-language research from task-specific models [12, 14, 37, 43, 51] to large foundation models [1, 4, 20, 34, 44–46, 49]. After pre-training on large-scale image-text data, these large vision-language models can quickly adapt to a set of downstream tasks. A line of works [34, 45, 49] train both visual and textual encoders from scratch, leading to a high computation cost. Recently, another line of works [1, 20, 44, 53] adopt pre-trained unimodal models as initialization and only train the newly introduced parameters. Our work belongs to the later one. LiT [53] utilizes pre-trained image encoder to speed up CLIP [34] training. Frozen [42] fine-tunes an image encoder to transform visual tokens into LLM’s soft prompts. Similarly, CLIPCap [30] proposes a mapping network to connect the pre-trained image encoder with LLMs. Flamingo [1] inserts several cross-attention layers to inject visual knowledge into LLMs. BLIP2 [20] connects pre-trained image encoders and LLMs with a Q-Former. To summary, these methods use mapping networks or cross-attention layers to connect vision and languages. Differently, our method only introduces a few learnable parameters and progressively injects visual features into pre-trained LLMs with a simple but efficient zero-init attention.
2.3. Parameter-Efficient Fine-Tuning

Parameter-Efficient Fine-Tuning (PEFT) [39] methods facilitate efficient adaptation of Large-scale Language Models (LLMs) without the need to update all model parameters, thereby reducing the cost and improving the efficiency of fine-tuning large models. Various PEFT techniques include Prefix Tuning [23], Low-Rank adaptation (LoRA) [11] and the insertion of adapter layers in pre-trained large language models [10, 25, 33]. Prefix Tuning [23] appends a collection of prefixes to autoregressive language models, or alternatively, incorporates prefixes for both encoder and decoder components, similar methods proposed in [19]. LoRA [11] introduces trainable rank decomposition matrices into each layer [13]. Adapter [10] involves inserting lightweight modules into each layer of pre-trained models, with only the adapters being trained during fine-tuning. This approach has been extended across numerous domains [33].

In this paper, we fine-tune pre-trained language models for instruction-following capabilities (response to instructs), and are distinct from existing ones in two aspects.

- **Zero-init Attention.** Prevalent PEFT methods might potentially disturb the pre-trained linguistic knowledge by directly inserting randomly initialized modules. This leads to unstable fine-tuning with large loss value at early training stages. LLaMA-Adapter adopts a zero-init attention with gating to mitigate this.

- **Unified Multi-modal Tuning.** Previous PEFT methods are normally developed to address specific modalities, such as language, image, and audio. In contrast, LLaMA-Adapter can handle both language and multi-modality fine-tuning with a unified manner, demonstrating superior generalization ability.

3. LLaMA-Adapter

In Section 3.1, we first introduce how to insert the learnable adaption prompts into LLaMA’s transformer. Then in Section 3.2, we present the details of zero attention mechanisms with zero-init gating. Finally in Section 3.3, we generalize LLaMA-Adapter for multi-modal reasoning.

3.1. Learnable Adaption Prompts

Given 52K instruction-to-output data [47] and a pre-trained LLaMA [41] with an N-layer transformer, we adopt a set of learnable adaption prompts for instruction-following fine-tuning, which are denoted as \( \{P_l\}_{l=1}^L \), where \( P_l \in \mathbb{R}^{K \times C} \) with \( K \) denoting the prompt length for each layer, and \( C \) equaling the feature dimension of LLaMA’s transformer. Note that we insert the prompts into the top-most \( L \) layers of the transformer (\( L \leq N \)). This can better tune the language representations with higher-level semantics.

Taking the \( l \)-th inserted layer as an example, we denote the \( M \)-length word tokens as \( T_l \in \mathbb{R}^{M \times C} \). Then, the adaption prompt is concatenated with \( T_l \) along the token dimension as prefix, formulated as

\[
[P_l; \ T_l] \in \mathbb{R}^{(K+M) \times C}. \tag{1}
\]

In this way, the instruction knowledge learned within \( P_l \) can effectively guide \( T_l \) to generate contextual responses.

3.2. Zero-init Attention

If the adaption prompts are randomly initialized, they might bring disturbance to the word tokens at the beginning of training, which harms the fine-tuning stability and effectiveness. Considering this, we modify the vanilla attention mechanisms at the last \( L \) transformer layers to be zero-init attention, as shown in Figure 2. Suppose the model is generating the \((M+1)\)-th word on top of \([P_l; T_l]\) at the \( l \)-th inserted layer, we denote the corresponding \((M+1)\)-th word token as \( t_l \in \mathbb{R}^{1 \times C} \). In the attention mechanism, several
Multi-modal Reasoning of LLaMA-Adapter

Figure 3. Multi-modal Reasoning of LLaMA-Adapter. On the ScienceQA benchmark [26], LLaMA-Adapter is extended to multi-modal variant for image-conditioned question answering. Given an image as the visual context, we aggregate its multi-scale features as the image token, and element-wisely add it onto the adaption prompts of $L$ inserted layers. In this way, LLaMA-Adapter achieves competitive reasoning capability based on multi-modal conditions.

Linear projection layers are first applied to transforming the input tokens into queries, keys, and values as

$$Q_t = \text{Linear}_q(t_t), \quad 2$$
$$K_t = \text{Linear}_k([P_t; T_t; t_t]), \quad 3$$
$$V_t = \text{Linear}_v([P_t; T_t; t_t]). \quad 4$$

Then, the attention scores before the softmax function are calculated as

$$S_t = Q_t K_t^T / \sqrt{C} \in \mathbb{R}^{1 \times (K+M+1)}, \quad 5$$

which records the feature similarities between $t_t$ and all $K+M+1$ tokens. Meanwhile, $S_t$ can be reformulated by two components as

$$S_t = [S_t^K; S_t^{M+1}]^T, \quad 6$$

where $S_t^K \in \mathbb{R}^{K \times 1}$ and $S_t^{M+1} \in \mathbb{R}^{(M+1) \times 1}$ denote the attention scores of $K$ adaption prompts and $M+1$ word tokens, respectively. The former $S_t^K$ represents how much information the learnable prompt contributes to $t_t$, which probably causes disturbance in the early training stage.

To this end, we adopt a learnable gating factor, denoted as $g_t$, to adaptively control the importance of $S_t^K$ in the attention. Initialized by zero, $g_t$ can firstly eliminate the influence of under-fitted prompts, and then increase its magnitude for providing more instruction semantics to LLaMA. Therefore, we independently apply the softmax functions to the two components in Equation (6), and multiply the first term by $g_t$, formulated as

$$S^t_i = [\text{Softmax}(S^K_t) \cdot g_t; \text{Softmax}(S^{M+1}_t)]^T. \quad 7$$

The separate softmax functions ensure the second term to be irrelevant to our adaption prompts. When $g_t$ is close to zero, it can convey the originally pre-trained knowledge of LLaMA to token $t_t$ for creditable generation.

Finally, we calculate the output of the attention layer with a linear projection layer as

$$t^o_i = \text{Linear}_o(S^o_i V_i) \in \mathbb{R}^{1 \times C}. \quad 8$$

With our proposed zero-init attention, the adaption prompts progressively inject the newly acquired instructional knowledge into LLaMA, while effectively incorporates its pre-trained ability to provide high-quality responses.

3.3. Multi-modal Reasoning

Not limited to textual instructions, LLaMA-Adapter is capable of answering a question based on input of other modalities, which augments the language model with rich cross-modal information. As shown in Figure 3, we take the ScienceQA benchmark [26] as examples. Given visual and textual contexts, along with the corresponding question and options, the model is required to conduct multi-modal reasoning to give the correct answer.

For an image as the visual context, we first leverage a pre-trained visual encoder, e.g., CLIP [34], to extract its multi-scale global features, denoted as $\{I_m\}_{m=1}^M$, where $I_m \in \mathbb{R}^{1 \times C_m}$ and $M$ denotes the scale number. Then, we concatenate the $M$-scale features along the channel dimension and apply a learnable linear projection layer on top, formulated as

$$I_p = \text{Projection} \left( \text{Concat} \left( \{I_m\}_{m=1}^M \right) \right). \quad 9$$
where $I_p \in \mathbb{R}^{1 \times C}$ and is regarded as the overall image token with the same feature dimension as our adaption prompts. After this, we repeat $I_p$ for $K$ times, and element-wisely add it onto the $K$-length adaption prompt at $L$ inserted transformer layers. For the $l$-th layer, we denote the acquired multi-modal prompt as

$$P_l + \text{Repeat}(I_p) \in \mathbb{R}^{K \times C},$$

which is incorporated with visual information from the given image context. In this way, LLaMA is fine-tuned to generate responses conditioned on vision-language inputs, and can achieve more challenging question answering with multi-modal understanding.

Shared across modalities, LLaMA-Adapter with additional input condition can also be generalized to video and audio. Using the pre-trained modal-specific encoders, we can integrate instructional signals of different modalities to the adaption prompts, which maximizes the generative and comprehension capabilities of LLaMA. We lease this as a future work.

4. Instruction-Following Evaluation

In this section, we evaluate the instruction-following capacity of LLaMA-Adapter by responding to instructions.

4.1. Experimental Details

Training Data. We use 52K instruction-following data from Stanford Alphaca [40] for training, denoted as Alphaca-52K. Each sample in Alphaca-52K contains the following fields: \{instruction\} is the description of a task, \{input\} is the context for the task, and \{output\} is the answer generated by GPT-3.5 (text-davinci-003) [5]. Note that around 40% of the examples include an input.

Implementation Details. We build LLaMa-Adapter based on the original LLaMa codebase\(^2\) with minor modifications. We train LLaMa-Adapter on 8 A100 GPUs for 5 epochs. The warmup epochs, batch size, learning rate and weight decay are set to 2, 64, 9e-3 and 0.02, respectively. In general, we utilize the pre-trained LLaMA model with 7B parameters and $N = 32$ transformer layers as the base model. We set the prompt length $K = 10$ and insert prompts into the last $L = 30$ layers by default. Other variants of LLaMA-Adapter with different inserted layers are also released in our code. In the generation stage, we adopt $\text{top-p}$ sampling as the default decoding method with a temperature 0.1 and a $\text{top-p} = 0.75$.

4.2. Performance

We compare LLaMA-Adapter with existing representative instruction-following method, Alphaca [40], in Table 1, and present the full comparison with Alpaca-LoRA [11] and GPT-3 [5] in Appendix A. As there still lacks of rigorous metrics for evaluation, we simply show some response examples for comparison. As shown, by only fine-tuning 1.2M parameters, our approach generates reasonable responses comparable to the fully fine-tuned Alpaca or the large-scale GPT-3. This fully demonstrates the effectiveness of our adapters with zero-init attention.

In Appendix B, we also compare LLaMA-Adapter with LLaMA-I, i.e., LLaMA-65B fine-tuned on large scale instructional data. As shown, LLaMA-Adapter is capable of various non-trivial tasks, such as dialog generation, code generation and question answering etc. We believe LLaMa-Adapter can be further enhanced by combining larger LLaMA models, enlarging the amount of training data, and scaling up learnable parameters.

4.3. Efficiency

In Table 3, we compare the learnable parameters, storage space, and training time of different instruction-following methods. As a lightweight plug-and-play module, LLaMA-Adapter enjoys superior training efficiency with only 1.2M parameters, 4.9M storage, and one-hour training. This enables us to fine-tune large-scale language models (LLaMA) on cheap and mobile devices. LLaMA-Adapter’s efficiency advantages can be further revealed by multi-node training, since only the gradients of 1.2M parameters are transferred among nodes, other than Alpaca’s 7B.

5. Multi-modal Evaluation

In this section, we illustrate the multi-modal reasoning capability of LLaMA-Adapter on the ScienceQA benchmark\(^3\), and conduct ablation studies to verify the effectiveness of our designs.

5.1. Experimental Details

Training Data. We train the multi-modal LLaMA-Adapter on ScienceQA [26], a large-scale multi-modal and multi-choice science question dataset collected from a wide range of domains\(^3\). Figure 4 gives two examples in ScienceQA. Each example normally contains a visual context, a textual context, a question, multiple options, and an answer. We omit the lecture and explanation for simplicity. The model is required to output the correct answer based on the multi-modal context and choices. The official question answering accuracy is adopted as the evaluation metric.

Implementation Details. We organize the textual input of LLaMA with one sentence, in an order of question, textual context, and options. For the visual context, we adopt the pre-trained CLIP [34] as the visual encoder to extract

\(^2\)https://github.com/facebookresearch/llama

\(^3\)https://scienceqa.github.io/explore
**Instruction:** Tell me about alpacas.

**LLaMA-Adapter:** Alpacas are members of the camelid family and are native to the Andes Mountains of South America. They are typically found in herds of 10-20 animals and are known for their gentle nature and their ability to thrive in harsh climates. Alpacas are typically white or grey in color and have long, silky coats. They are also known for their ability to produce high-quality wool, which is used in the production of clothing and other products.

**Alphaca:** Alpacas are small, fluffy animals related to camels and llamas. They are native to Peru and Bolivia, and were first domesticated around 5,000 years ago. They are kept mainly for their fine, soft fleece, which is used to make knitwear and other garments. Alpacas are herd animals and live in small family groups, led by an older male. They are highly social animals and can form strong bonds with their owners.

**Instruction:** Tell me about the president of Mexico in 2019.

**LLaMA-Adapter:** The president of Mexico in 2019 was Andrés Manuel López Obrador, who took office on December 1st, 2018. He is a member of the left-wing party Morena and is known for his populist policies. He has focused on tackling corruption, reducing poverty, and improving the country’s infrastructure. He has also been a vocal critic of the United States’ immigration policies.

**Alphaca:** In 2019, Andres Manuel Lopez Obrador was sworn in as President of Mexico. He is the country’s 58th president and the first from the left-wing Morena party. He was elected in 2018 on a promise to combat corruption and reduce poverty and has faced criticism for his handling of the economy, crime and the country’s relationship with the United States. He is also known for his ambitious social welfare programs and has been praised for raising the minimum wage and providing aid to low-income families. He is seen as a champion of democracy and human rights in Mexico.

**Instruction:** List all Canadian provinces in alphabetical order.

**LLaMA-Adapter:** Alberta, British Columbia, Manitoba, New Brunswick, Newfoundland and Labrador, Nova Scotia, Ontario, Prince Edward Island, Quebec, Saskatchewan, Yukon.

**Alphaca:** Alberta, British Columbia, Manitoba, New Brunswick, Newfoundland and Labrador, Nova Scotia, Ontario, Prince Edward Island, Quebec, Saskatchewan, Northwest Territories, Nunavut, Yukon.

**Instruction:** Tell me five words that rhyme with 'shock'.

**LLaMA-Adapter:** Flock, rock, lock, cook, and stock.

**Alphaca:** Five words that rhyme with shock are: rock, pop, shock, cook, and snock.

**Instruction:** Translate the sentence ‘I have no mouth but I must scream’ into Spanish.

**LLaMA-Adapter:** No tengo boca pero tengo que gritar.

**Alphaca:** No tengo boca, pero debo gritar.

**Instruction:** Write a Python program that prints the first 10 Fibonacci numbers.

**LLaMA-Adapter:**

```python
def fibonacci():
    a = 0
    b = 1
    for i in range(10):
        a, b = b, a + b
        print(a, b)

fibonacci()
```

**Alphaca:**

```python
numbers = [0, 1]
for i in range(2, 11):
    numbers.append(numbers[i-2] + numbers[i-1])
print(numbers)
# Output: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

---

Table 1. **Instruction-following Comparison** between LLaMA-Adapter and Alpaca [40].
<table>
<thead>
<tr>
<th>Model</th>
<th>Tuned Params</th>
<th>Avg</th>
<th>NAT</th>
<th>SOC</th>
<th>LAN</th>
<th>TXT</th>
<th>IMG</th>
<th>NO</th>
<th>G1-6</th>
<th>G7-12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Choice [26]</td>
<td>-</td>
<td>39.83</td>
<td>40.28</td>
<td>46.13</td>
<td>29.25</td>
<td>47.45</td>
<td>40.08</td>
<td>33.66</td>
<td>39.35</td>
<td>40.67</td>
</tr>
<tr>
<td>Human [26]</td>
<td>-</td>
<td>88.40</td>
<td>90.23</td>
<td>84.97</td>
<td>87.48</td>
<td>89.60</td>
<td>87.50</td>
<td>88.10</td>
<td>91.59</td>
<td>82.42</td>
</tr>
<tr>
<td>MCAN [52]</td>
<td>95M</td>
<td>54.54</td>
<td>56.08</td>
<td>54.33</td>
<td>58.09</td>
<td>59.43</td>
<td>51.17</td>
<td>55.40</td>
<td>51.65</td>
<td>59.72</td>
</tr>
<tr>
<td>Top-Down [2]</td>
<td>70M</td>
<td>59.02</td>
<td>59.50</td>
<td>61.82</td>
<td>62.90</td>
<td>54.88</td>
<td>59.79</td>
<td>57.27</td>
<td>62.16</td>
<td></td>
</tr>
<tr>
<td>BAN [16]</td>
<td>112M</td>
<td>59.37</td>
<td>60.88</td>
<td>64.66</td>
<td>66.46</td>
<td>62.61</td>
<td>65.51</td>
<td>65.83</td>
<td>63.94</td>
<td></td>
</tr>
<tr>
<td>DAFAF [8]</td>
<td>74M</td>
<td>60.72</td>
<td>64.03</td>
<td>68.82</td>
<td>63.55</td>
<td>65.88</td>
<td>54.49</td>
<td>64.11</td>
<td>57.12</td>
<td>67.17</td>
</tr>
<tr>
<td>ViLT [17]</td>
<td>113M</td>
<td>61.14</td>
<td>60.48</td>
<td>63.89</td>
<td>60.27</td>
<td>63.20</td>
<td>61.38</td>
<td>61.09</td>
<td>61.90</td>
<td></td>
</tr>
<tr>
<td>Patch-TRM [27]</td>
<td>90M</td>
<td>61.42</td>
<td>65.19</td>
<td>46.79</td>
<td>65.55</td>
<td>66.96</td>
<td>55.28</td>
<td>64.95</td>
<td>58.04</td>
<td>67.50</td>
</tr>
<tr>
<td>VisualBERT [21, 22]</td>
<td>111M</td>
<td>61.87</td>
<td>69.18</td>
<td>69.18</td>
<td>74.91</td>
<td>61.18</td>
<td>62.91</td>
<td>58.54</td>
<td>62.96</td>
<td>59.92</td>
</tr>
<tr>
<td>UnifiedQA [15]</td>
<td>223M</td>
<td>70.12</td>
<td>68.16</td>
<td>69.18</td>
<td>74.91</td>
<td>63.78</td>
<td>61.38</td>
<td>77.84</td>
<td>72.98</td>
<td>65.00</td>
</tr>
<tr>
<td>UnifiedQA (CoT)</td>
<td>223M</td>
<td>74.11</td>
<td>71.00</td>
<td>76.04</td>
<td>78.91</td>
<td>66.42</td>
<td>66.53</td>
<td>81.81</td>
<td>77.06</td>
<td>68.82</td>
</tr>
<tr>
<td>GPT-3 [5]</td>
<td>0M</td>
<td>74.04</td>
<td>75.04</td>
<td>66.59</td>
<td>78.00</td>
<td>74.24</td>
<td>65.74</td>
<td>79.58</td>
<td>76.36</td>
<td>69.87</td>
</tr>
<tr>
<td>GPT-3 (CoT)</td>
<td>0M</td>
<td>75.17</td>
<td>75.44</td>
<td>70.87</td>
<td>78.09</td>
<td>74.68</td>
<td>67.43</td>
<td>79.93</td>
<td>78.23</td>
<td>69.68</td>
</tr>
<tr>
<td>MM-COT [55]</td>
<td>223M</td>
<td>70.53</td>
<td>71.09</td>
<td>70.75</td>
<td>69.18</td>
<td>71.16</td>
<td>65.84</td>
<td>71.57</td>
<td>71.00</td>
<td>69.68</td>
</tr>
<tr>
<td>MM-COT T [55]</td>
<td>223M</td>
<td>84.91</td>
<td>87.52</td>
<td>77.17</td>
<td>85.82</td>
<td>87.88</td>
<td>82.90</td>
<td>86.83</td>
<td>84.65</td>
<td>85.37</td>
</tr>
</tbody>
</table>

Table 2. **Question Answering Accuracy (%) on ScienceQA [26] test set.** We report the accuracy of different question classes, including natural science, social science, language science, text context, image context, no context, grades 1-6, and grades 7-12. GPT-3 [] of 175B parameters conducts zero-shot answering. LLaMA-Adapter and MM-COT denote their single-modal variants with text-only input.

<table>
<thead>
<tr>
<th>Model</th>
<th>Tuned Params</th>
<th>Storage Space</th>
<th>Training Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpaca [40]</td>
<td>7B</td>
<td>13G</td>
<td>3 hours</td>
</tr>
<tr>
<td>Alpaca-LoRA [11]</td>
<td>4.2M</td>
<td>16.8M</td>
<td>-</td>
</tr>
<tr>
<td>LLaMA-Adapter T</td>
<td>1.2M</td>
<td>4.7M</td>
<td>1 hour</td>
</tr>
<tr>
<td>LLaMA-Adapter</td>
<td>1.8M</td>
<td>8.4M</td>
<td>6.8M</td>
</tr>
</tbody>
</table>

Table 3. **Comparison of Efficiency.** The training time is tested on 8 A100 GPUs.

5.2. Performance

In Table 2, we compare LLaMA-Adapter with popular VQA methods [2, 8, 16, 17, 21, 22, 27, 52] and language models [5, 15, 55]. As shown, our single-modal variant (‘LLaMA-Adapter_T’) attains 78.31% accuracy with 1.2M parameters. By further injecting visual conditions with a 0.6M projection layer, our multi-modal variant (‘LLaMA-Adapter’) is boosted by +6.88% answering accuracy. Traditional VQA methods are required to train the entire network with considerable resource budget, while LLaMA-Adapter only fine-tunes less than 2M parameters in a short time. Compared to GPT-3, though conducting zero-shot answering without fine-tuning, GPT-3 contains 175B total parameters, much larger than our 7B LLaMA. Also, as a language model, GPT-3 can not leverage any additional visual information. In contrast, LLaMA-Adapter can be easily switched into multi-modal variant, and achieves +10% higher accuracy. Besides, we notice that MM-COT [55] is on par with our approach, but it relies on the complex two-stage inference. We believe our LLaMA-Adapter can also be boosted and leave the exploration of chain-of-thought for future research.

5.3. Ablation Study

**Insertion Layers.** We first investigate the number of transformer layers to be inserted. As shown in Table 4, increasing the inserted layers leads to a significant improvement in question answering accuracy, e.g., +17.41% from 10 to 30, and +10.49% from 20 to 30. It indicates that more adaption prompts can provide stronger task-specific guidance to the pre-trained LLaMA. This encourages us to insert adaption prompts into more transformer layers for the larger LLaMA model 65B parameters in the future.

**Zero-init Attention.** Our proposed zero-init attention in LLaMA-Adapter is essential for the early-stage training sta-
**Question:** Select the fish below.

**Context:** Fish live underwater. They have fins, not limbs. Fish are cold-blooded. The body temperature of cold-blooded animals depends on their environment. A Banggai cardinalfish is an example of a fish.

**Choices:**
- (A) green moray eel
- (B) rabbit
- (C) woodpecker
- (D) bald eagle

**Answer:** The answer is (A)

**Question:** Think about the magnetic force between the magnets in each pair. Which of the following statements is true?

**Context:** The images below show two pairs of magnets. The magnets in different pairs do not affect each other. All the magnets shown are made of the same material.

**Choices:**
- (A) The magnitude of the magnetic force is the same in both pairs.
- (B) The magnitude of the magnetic force is greater in Pair 1.
- (C) The magnitude of the magnetic force is greater in Pair 2.

**Answer:** The answer is (C)

---

**Figure 4.** Multi-modal Reasoning on ScienceQA [26] by LLaMA-Adapter.

<table>
<thead>
<tr>
<th>Layers</th>
<th>Params</th>
<th>Val Acc (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.97</td>
<td>55.95</td>
</tr>
<tr>
<td>20</td>
<td>1.37</td>
<td>73.36</td>
</tr>
<tr>
<td>30</td>
<td>1.79</td>
<td><strong>83.85</strong></td>
</tr>
</tbody>
</table>

Table 4. Ablation on Inserted Layers.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Val Acc (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rand-Init Attention</td>
<td>40.77</td>
</tr>
<tr>
<td>Zero-Init Attention</td>
<td><strong>84.04</strong></td>
</tr>
<tr>
<td>Gain</td>
<td>+43.27</td>
</tr>
</tbody>
</table>

Table 5. Ablation on Zero-init Attention.

**Robustness to Over-fitting.** Since the data for fine-tuning is normally much smaller-scale than that for pre-training, researchers have to carefully tune a set of hyperparameters to avoid over-fitting. In Table 6, we show our LLaMA-Adapter is robust to the over-fitting issue. Similar to the conclusion in [32], even if our model has over-fitted the fine-tuning data, e.g., the validation loss marginally varying from 0.136 (15 epochs) to 0.282 (60 epochs), the validation accuracy is still increasing, e.g., from 82.08 to 83.94. One possible reason is that LLaMA-Adapter only introduces a few learnable parameters with the 7B LLaMA frozen. Thus, a small-scale dataset can also fully fine-tune our adapters.

<table>
<thead>
<tr>
<th>Epoch</th>
<th>Train Loss</th>
<th>Val Loss</th>
<th>Val Acc (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>0.022</td>
<td>0.136</td>
<td>82.08</td>
</tr>
<tr>
<td>30</td>
<td>0.004</td>
<td>0.241</td>
<td>83.85</td>
</tr>
<tr>
<td>60</td>
<td>0.001</td>
<td>0.282</td>
<td><strong>83.94</strong></td>
</tr>
</tbody>
</table>

Table 6. Loss Values vs. Validation Accuracy.

---

Figure 5. Loss Curves with and without Zero-init Attention. We plot the loss curves of LLaMA-Adapter with and without the zero-init attention in blue and red, respectively.
6. Conclusion

In this paper, we propose LLaMA-Adapter, an efficient adaption method for training instruction-following language models. With only 1.2M parameters and one-hour training, our approach effectively fine-tunes LLaMA, and exhibits superior efficiency compared to the 7B Stanford Alpaca. For better training stability and final performance, we propose a zero-init attention with gating mechanism, which adaptively incorporates instructional signals, while preserves the pre-trained generative knowledge in LLaMA. Our approach can also be generalized to image conditions for multi-modal reasoning, achieving competitive performance on the ScienceQA benchmark. In the future, we will further integrate wider multi-modal inputs into LLaMA-Adapter, such as audio and video. More experiments with the larger 65B LLaMA model on VQA v2 [9], OK-VQA [24], TVQA [18], and DocVQA [28] datasets will be conducted.

References


A. Instruction-Following Capabilities

**Instruction:** Tell me about alpacas.

**LLaMA-Adapter:** Alpacas are members of the camelid family and are native to the Andes Mountains of South America. They are typically found in herds of 10-20 animals and are known for their gentle nature and their ability to thrive in harsh climates. Alpacas are typically white or grey in color and have long, silky coats. They are also known for their ability to produce high-quality wool, which is used in the production of clothing and other products.

**Alpaca-LoRA:** Alpacas are members of the camelid family and are native to the Andes Mountains of South America. They are known for their soft, luxurious fleece, which is used to make clothing, blankets, and other items. Alpacas are herbivores and graze on grasses and other plants. They are social animals and live in herds of up to 20 individuals.

**Alphaca:** Alpacas are small, fluffy animals related to camels and llamas. They are native to Peru and Bolivia, and were first domesticated around 5,000 years ago. They are kept mainly for their fine, soft fleece, which is used to make knitwear and other garments. Alpacas are herd animals and live in small family groups, led by an older male. They are highly social animals and can form strong bonds with their owners.

**GPT-3:** Alpacas are a domesticated species of South American camelid, similar to llamas. They are native to the Andes Mountains in South America and are kept as livestock for their soft, luxurious wool. Alpacas are social animals and live in herds of up to 20 individuals. They are typically kept in small herds of two to five animals, and are relatively easy to care for. Alpacas are herbivores and feed on grass, hay, and other vegetation. They are also known for their gentle and friendly nature, making them popular as pets.

**Instruction:** Tell me about the president of Mexico in 2019.

**LLaMA-Adapter:** The president of Mexico in 2019 was Andrés Manuel López Obrador, who took office on December 1st, 2018. He is a member of the left-wing party Morena and is known for his populist policies. He has focused on tackling corruption, reducing poverty, and improving the country’s infrastructure. He has also been a vocal critic of the United States’ immigration policies.

**Alpaca-LoRA:** The president of Mexico in 2019 was Andrés Manuel López Obrador, who took office on December 1, 2018. He is a member of the National Regeneration Movement (MORENA) political party and is the first left-wing president of Mexico since 1946. He is known for his anti-corruption and anti-neoliberal policies, as well as his commitment to improving the living conditions of the Mexican people.

**Alphaca:** In 2019, Andres Manuel Lopez Obrador was sworn in as President of Mexico. He is the country’s 58th president and the first from the left-wing Morena party. He was elected in 2018 on a promise to combat corruption and reduce poverty and has faced criticism for his handling of the economy, crime and the country’s relationship with the United States. He is also known for his ambitious social welfare programs and has been praised for raising the minimum wage and providing aid to low-income families. He is seen as a champion of democracy and human rights in Mexico.

**GPT-3:** The president of Mexico in 2019 is Andrés Manuel López Obrador, commonly known as AMLO. He is a Mexican politician who has been the President of Mexico since December 1, 2018. He is a member of the National Regeneration Movement (MORENA) party and previously served as the Head of Government of Mexico City from 2000 to 2005. He is a left-wing populist and has promised to reduce corruption and inequality in Mexico. He has also promised to increase public spending on social programs and infrastructure projects.
**Instruction:** List all Canadian provinces in alphabetical order.

**LLaMA-Adapter:** Alberta, British Columbia, Manitoba, New Brunswick, Newfoundland and Labrador, Nova Scotia, Ontario, Prince Edward Island, Quebec, Saskatchewan, Yukon.

**Alpaca-LoRA:** Alberta, British Columbia, Manitoba, New Brunswick, Newfoundland and Labrador, Nova Scotia, Ontario, Prince Edward Island, Quebec, Saskatchewan.

**Alphaca:** Alberta, British Columbia, Manitoba, New Brunswick, Newfoundland and Labrador, Nova Scotia, Ontario, Prince Edward Island, Quebec, Saskatchewan, Northwest Territories, Nunavut, Yukon.


---

**Instruction:** Tell me five words that rhyme with 'shock'.

**LLaMA-Adapter:** Flock, rock, lock, cook, and stock.

**Alpaca-LoRA:** Flock, clock, dock, stock, and sock.

**Alphaca:** Five words that rhyme with shock are: rock, pop, shock, cook, and snock.


---

**Instruction:** Translate the sentence 'I have no mouth but I must scream' into Spanish.

**LLaMA-Adapter:** No tengo boca pero tengo que gritar.

**Alpaca-LoRA:** No tengo boca pero tengo que gritar.

**Alphaca:** No tengo boca, pero debo gritar.

**GPT-3:** No tengo boca pero debo gritar.
**Instruction:** Write a Python program that prints the first 10 Fibonacci numbers.

**LLaMA-Adapter:**
```python
def fibonacci():
    a = 0
    b = 1
    for i in range(10):
        a, b = b, a + b
        print(a, b)
    print("Fibonacci numbers: ", a, b)
fibonacci()
```

**Alpaca-LoRA:**
```python
def fibonacci(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci(n-1) + fibonacci(n-2)
print(fibonacci(10))
```

**Alpaca:**
```python
numbers = [0, 1]
for i in range(2, 11):
    numbers.append(numbers[i-2] + numbers[i-1])
print(numbers)
# Output: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

**GPT-3:**
```python
# Fibonacci numbers are the numbers in the following integer sequence, called the Fibonacci sequence, and characterized by the fact that every number after the first two is the sum of the two preceding ones.
# Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

# Program:
# Initialize the first two numbers of the Fibonacci sequence
a = 0
b = 1
# Print the first 10 numbers of the Fibonacci sequence
print(a)
print(b)
for i in range(2, 10):
    c = a + b
    a = b
    b = c
    print(c)
```
B. Comparison with Instruct LLaMA (LLaMA-I)

**Instruction:** Write a conversation between the sun and pluto.

**LLaMA-Adapter:**
Sun: Hey Pluto, how’s it going?
Pluto: Not too bad, how about you?
Sun: I’m doing great! How about you?
Pluto: I’m doing alright, I guess.
Sun: That’s great!
Pluto: Yeah, I guess.
Sun: So, what’s new with you?
Pluto: Not much, just hanging out with my friends.
Sun: That’s cool!
Pluto: Yeah, it is.
Sun: Well, I’m going to go now. Take care!
Pluto: You too!

**LLaMA-I:**
Sun: Hey Pluto, what’s up?
Pluto: Not much. Just hanging out here at the edge of the solar system.
Sun: Yeah, I’ve been meaning to talk to you about that. You’re not a planet anymore.
Sun: I’m sorry, but it’s true. You just don’t meet the criteria anymore.
Pluto: This is bulls**t! I’ve been a planet for over 70 years!
Sun: Things change, Pluto. You’ve just got to accept it.
Pluto: F**k you, Sun! F**k you and your stupid solar system!

**Instruction:** Write a shouting match between Julius Caesar and Napoleon.

**LLaMA-Adapter:**
Julius Caesar: “You’re nothing but a petty tyrant, Napoleon! You’re nothing but a pathetic little man who has no idea what true power is!”
Napoleon: “You’re nothing but a weak and pathetic fool, Caesar! You’re nothing but a petty tyrant who has no idea what true power is!”

**LLaMA-I:**
Of course! Here is a conversation between Julius Caesar and Napoleon:
Julius Caesar: You’re nothing but a glory hound, Napoleon. You’ve always been more concerned with your own fame and fortune than with the people you’re supposed to be leading.
Napoleon: And what about you, Caesar? You’re no better than I am. At least I’m honest about my ambitions. You’re just a fraud who pretends to care about the common man while you line your own pockets.
Julius Caesar: The difference between us is that I actually accomplish things. I’ve conquered more territory and defeated more enemies than you ever will. And I did it all for Rome, not for myself.
Napoleon: Oh, please. You only care about yourself. If you really cared about Rome, you would have done something to stop the decline instead of letting it happen on your watch.
Julius Caesar: The decline of Rome is not my fault. I’ve done everything in my power to try to stop it, but there’s only so much one man can do. Napoleon: And yet you’re still trying to take credit for everything good that happens, even though it’s clear that you’re not the one who’s really in charge.
Julius Caesar: That’s not true! I am in charge, and I’m going to make Rome great again. Just watch me.
**Instruction**: How do I send an HTTP request in Javascript?

**LLaMA-Adapter**: The HTTP request can be sent using the XMLHttpRequest object. The code to send an HTTP request is as follows:

```javascript
var xhr = new XMLHttpRequest();
xhr.open("GET", "https://example.com/", true);
xhr.onreadystatechange = function() {
  if (xhr.readyState == 4 && xhr.status == 200) {
    // Success
  } else {
    // Error
  }
};
xhr.send();
```

The code above will send an HTTP GET request to the URL https://example.com/.

**LLaMA-I**: Here’s an example of how to make a GET request using the JS HTTP API:

```javascript
var request = new XMLHttpRequest();
request.open('GET', '/url/to/resource', true);
request.onload = function() {
  if (request.status >= 200 && request.status < 400) {
    // Success!
  } else {
    // Error!
  }
};
request.send();
```

**Instruction**: What are 3 popular chess openings?

**LLaMA-Adapter**: The three most popular chess openings are the Sicilian Defense, the French Defense, and the Caro-Kann Defense.

**LLaMA**: There are dozens of different chess openings that are popular among players, but some of the most widely used openings include the Sicilian Defense, the French Defense, and the Caro-Kann Defense. These openings are popular because they are effective at countering white’s early attacks, and they also allow black to quickly establish a strong position in the center of the board. Other popular openings include the Ruy Lopez, the Italian Game, and the Scotch Game.

**Instruction**: Write a theory to explain why cat never existed.

**LLaMA-Adapter**: The theory is that cats never existed because they are a figment of human imagination. Cats are a popular pet and have been depicted in art and literature for centuries, but there is no evidence that they ever existed in the wild. The earliest evidence of cats is from ancient Egypt, where they were revered as sacred animals. This suggests that cats were domesticated by humans, and not the other way around.

**LLaMA-I**: There are no such things as cats. "Cats" are actually extradimensional parasites that have invaded our world and taken the form of small, harmless-looking mammals. Their true forms are much more horrific, and they use mind control to make humans think that they are cute and harmless animals. They are actually parasites that feed on our life force, and they can extend their tendrils into our brains and control our thoughts and actions.
**Instruction:** Send an email requesting that people use language models responsibly.

**LLaMA-Adapter:**
Dear [recipient],

I am writing to express my concern about the increasing use of language models in our society. Language models are powerful tools that can be used to improve the quality of our conversations and interactions, but they can also be misused to perpetuate harmful stereotypes and biases.

I urge you to use language models responsibly and to be mindful of the potential consequences of their use. I also encourage you to consider the ethical implications of using language models and to take steps to ensure that they are not used to perpetuate harmful stereotypes and biases.

Thank you for your time and consideration.

**LLaMA-I:**
Here is a draft email you could send:
"Language models are exciting new tools that can be used for a variety of purposes, including automated writing, automatic translation, and speech recognition. But they are also complex and powerful technologies, and should be used responsibly.

We encourage you to explore and experiment with these new tools, but also ask that you pay attention to the potential impact of your work on others, and to act responsibly. In particular, we recommend that you be mindful of the potential for language models to be used for deceptive or malicious purposes, and avoid spreading such models without appropriate precautions and warnings.

We also recommend that you consider the ethical implications of your work, and encourage you to take care to ensure that your use of these tools is aligned with your values.

Finally, we remind you that the use of these tools may be subject to certain legal and policy constraints, and encourage you to seek advice and support from appropriate experts before using these tools in ways that may have broader impact or implications."
**Instruction:** You are a bash terminal. I will give you bash commands, and you will respond with the terminal output, and nothing else. The first command is `pwd`.

**LLaMA-Adapter:**
The current working directory is `/home/user/`

```bash
echo "I am a chatbot" > file.txt && cat file.txt && ls -l
```

```
I am a chatbot
-rw-r--r– 1 user user 10 Aug 29 17:05 file.txt
```

```bash
cp file.txt file2.txt && ls -l
```

```
-rw-r--r– 1 user user 10 Aug 29 17:05 file.txt
-rw-r--r– 1 user user 10 Aug 29 17:05 file2.txt
```

```bash
rm file.txt && ls -l
```

```
-rw-r--r– 1 user user 10 Aug 29 17:05 file2.txt
```

```bash
head -n 5 /proc/meminfo
```

```plaintext
MemTotal: 1699856 kB
MemFree: 1699856 kB
Buffers: 1699856 kB
Cached: 0 kB
SwapCached: 0 kB
```

**LLaMA-I:**
Sure. Here’s the output:

```
/home/human
```

Is this helpful?

```bash
ls -l
```

Here’s the output:

```
-rw-r--r– 1 human human 8 Sep 10 10:56 file.txt
```

Is this helpful?

```bash
cp file.txt file2.txt && ls -l
```

Here’s the output:

```
-rw-r--r– 1 human human 8 Sep 10 10:56 file.txt
-rw-r--r– 1 human human 8 Sep 10 10:56 file2.txt
```

Is this helpful?

```bash
rm file.txt && ls -l
```

Here’s the output:

```
-rw-r--r– 1 human human 0 Sep 10 10:56 file2.txt
```

```bash
head -n 5 /proc/meminfo
```

Here’s the output:

```plaintext
MemTotal: 164928 kB
MemFree: 140604 kB
Buffers: 48 kB
Cached: 19768 kB
SwapCached: 0 kB
```