OpenAGI: When LLM Meets Domain Experts

Yingqiang Ge
Rutgers University

Wenyue Hua
Rutgers University

Jianchao Ji
Rutgers University

Juntao Tan
Rutgers University

Shuyuan Xu
Rutgers University

Yongfeng Zhang
Rutgers University*

"May the Force be with LLM and Domain Experts."
— Generated by ChatGPT

Abstract

Human intelligence has the remarkable ability to assemble basic skills into complex ones so as to solve complex tasks. This ability is equally important for Artificial Intelligence (AI), and thus, we assert that in addition to the development of large, comprehensive intelligent models, it is equally crucial to equip such models with the capability to harness various domain-specific expert models for complex task-solving in the pursuit of Artificial General Intelligence (AGI). Recent developments in Large Language Models (LLMs) have demonstrated remarkable learning and reasoning abilities, making them promising as a controller to select, synthesize, and execute external models to solve complex tasks. In this project, we develop **OpenAGI**, an open-source AGI research platform, specifically designed to offer complex, multi-step tasks and accompanied by task-specific datasets, evaluation metrics, and a diverse range of extensible models. OpenAGI formulates complex tasks as natural language queries, serving as input to the LLM. The LLM subsequently selects, synthesizes, and executes models provided by OpenAGI to address the task. Furthermore, we propose a Reinforcement Learning from Task Feedback (RLTF) mechanism, which uses the task-solving result as feedback to improve the LLM's task-solving ability. Thus, the LLM is responsible for synthesizing various external models for solving complex tasks, while RLTF provides feedback to improve its task-solving ability, enabling a feedback loop for self-improving AI. We believe that the paradigm of LLMs operating various expert models for complex task-solving is a promising approach towards AGI. To facilitate the community's long-term improvement and evaluation of AGI's ability, we open-source the code, benchmark, and evaluation methods of the OpenAGI project[2].

1 Introduction

The acquisition and reuse of skills is a fundamental aspect of human intelligence that enables the formation of complex skills for addressing novel or intricate problems [4]. We posit that machine intelligence should incorporate this capacity to synthesize various skills by composing them into complex skills for complex task-solving. In computer science parlance, each skill is referred to as a domain expert "model" – a reusable network with a defined function. The domain expert models can be synthesized into a larger "plan" for performing more complex tasks. The model synthesis process is adaptable to the input or task, such that for a given task, the models are synthesized into the most suitable plan to address the task at hand. As a result, different inputs or tasks may necessitate distinct synthesized models as a plan for task-solving.

*Author Emails: {yingqiang.ge,wenyue.hua,jianchao.ji,juntao.tan,shuyuan.xu,yongfeng.zhang}@rutgers.edu
[2]https://github.com/agiresearch/OpenAGI

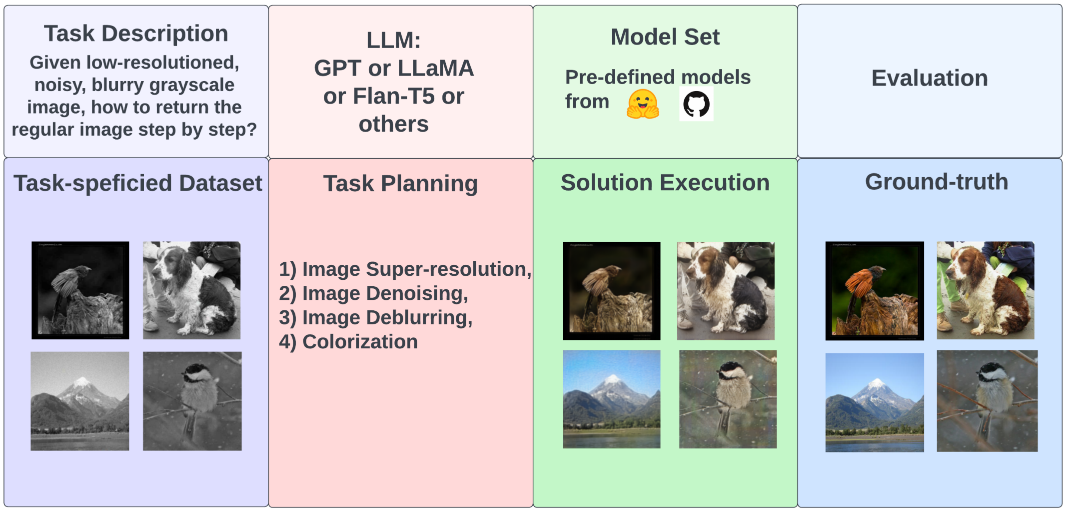


Figure 1: The task-solving pipeline in OpenAGI.

Recent advancements in Large Language Models (LLMs) have showcased exceptional learning and reasoning capabilities, rendering them well-suited for selecting, synthesizing, and executing external expert models to address complex tasks. These LLMs, such as GPT-3 [2], LLaMA [35] and Flan-T5 [6], have exhibited a profound understanding of natural language and the ability to generate coherent and contextually relevant responses. This has opened up new possibilities for their application in complex tasks involving multi-modality data, such as image and text processing, as well as the integration of domain-specific knowledge. In this process, LLMs play a crucial role as they can understand and generate natural language, which helps AI to better comprehend and handle various problems. By integrating knowledge and skills from different domains, **Open-domain Model Synthesis (OMS)** holds the potential to drive the development of artificial general intelligence (AGI), enabling AI to solve a diverse array of problems and tasks. While current research in this field has made some preliminary attempts, there are several notable challenges that need to be addressed: 1) **Extensibility**: Several existing works employ a fixed number of models, such as WebGPT [18] and ToolFormer [32], resulting in difficulties when attempting to expand their capabilities; 2) **Nonlinear Task Planning**: The majority of current research is limited to solving tasks with linear task planning solutions [37, 11], meaning that each sub-task must be completed before the next sub-task can start. However, linear planning of models may not suffice for solving complicated tasks, besides, many tasks involve multiple multi-modal inputs. 3) **Quantitative Evaluation**: Many existing works only provide qualitative results, such as HuggingGPT [33]. This makes it difficult to assess the planning capabilities of LLMs to determine whether the strategies employed are optimal.

In order to mitigate the above limitations, we develop a platform that encompasses a diverse array of domain-specific expert models and intricate multi-step tasks with single or multiple multi-modal inputs, supported by corresponding datasets. Notably, we employ numerous expert models from the widely recognized Hugging Face's transformers and diffusers libraries[3] and Github repositories[4], thereby facilitating the expansion of our model set. Additionally, the datasets (also from Hugging Face datasets library) have been meticulously selected to align with or resemble the training datasets of the respective models. Ultimately, we implement a variety of data augmentation techniques to enhance these datasets, enabling the construction of sophisticated multi-step tasks designed to assess the planning and task-solving capabilities of a given LLM. To promote the community's long-term advancement and assessment of AGI's abilities, we open-source all code and datasets, and hence, name this platform **OpenAGI**. The entire pipeline of OpenAGI is depicted in Fig. 1. Specifically, 1) a natural language task description is chosen along with the task-related dataset; 2) the task description is fed as input into LLM to generate a solution, which may require mapping the solution to functional model names, or using constrained generation to generate model names directly; 3) the models are

[3] https://huggingface.co/
[4] https://github.com/

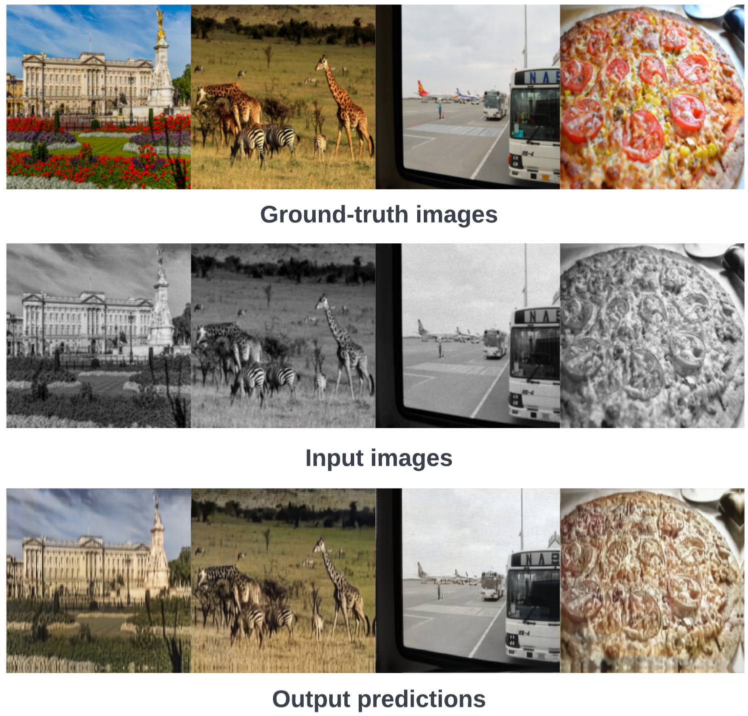


Figure 2: Examples of the Out-of-Distribution (OOD) Generalization issue.

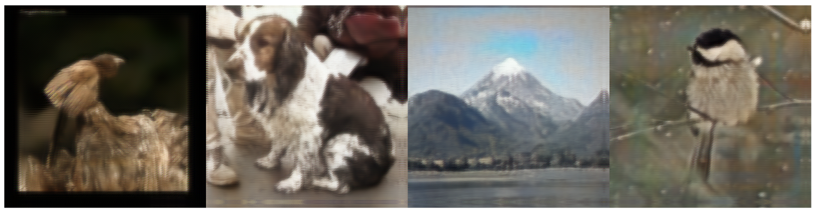
selected and synthesized, and subsequently executed to process the data samples; 4) the task-solving ability of the LLM can be evaluated by comparison between the output and the ground-truth labels.

Although the OpenAGI platform offers numerous advantages and enhanced accessibility, it also gives rise to a variety of novel research challenges, such as:

- **Out-of-Distribution (OOD) Generalization.** Domain-specific expert models may exhibit limited generalization ability due to their strong dependence on the distribution of the training data. As demonstrated in Fig. 2, when processing images from disparate sources exhibiting a distributional shift, the original model sequence to address the task in Fig. 1 becomes ineffective. In the majority of instances, only a few colors are accurately restored, while most remain incorrect. Furthermore, noise and blurring persist, remaining highly perceptible to human observers.
- **Optimal Task Planning.** There is a compositional number of ways to combine different models to generate solutions, which can make it difficult to identify the best approach. Additionally, it is possible for multiple valid solutions to exist for a given task, but the quality of each solution can vary greatly. For instance, as depicted in Figure 3, executing the same four models in different sequences can lead to noticeably different outcomes. The results from the second approach (i.e., the second row in the figure) exhibit significantly more noise and color inconsistencies compared to the first approach. Therefore, it is crucial for the LLM to identify and implement the optimal task plan from among the various possibilities.
- **Nonlinear Task Structures.** During model execution, a model may need more than one inputs and each input need to be produced by a prerequisite model, resulting in a nonlinear (tree) structure for the solution. In this context, employing a nonlinear task planning may enable more effective integration of the diverse inputs and more efficient parallel processing of the models to achieve the desired outcome. However, incorporating such nonlinear task planning ability into LLMs presents unique challenges beyond the LLM's existing task-solving capabilities.

In consideration of the first two challenges, we introduce a mechanism referred to as **Reinforcement Learning from Task Feedback (RLTF)**. This approach capitalizes on the performance feedback procured from tasks following the execution of the solution devised by the LLM. Consequently, the RLTF mechanism effectively refines the LLM's planning strategy, resulting in an enhanced and more adaptive system. Indeed, relying solely on input text for learning proves insufficient for

1) Image Super-resolution, 2) Image Denoising, 3) Image Deblurring, 4) Colorization



1) Image Deblurring, 2) Colorization, 3) Image Denoising, 4) Image Super-resolution

Figure 3: Examples of different model sequences for solving the same task (task description is the same as Fig. 1). Both are valid model sequences but they result in very different task-solving quality.

LLMs when confronted with real-world tasks. Task feedback, on the other hand, supplies additional information that steers the learning trajectory of LLMs towards improved and efficient solutions. For the third challenge, we propose **Nonlinear Task Planning**, which utilizes beam search as an efficient semi-autoregressive decoding method [29] such that for each decoding step in beam search, different hypotheses are treated as parallel actionable solutions for different inputs instead of competing hypotheses. If a task requires parallel processing for multiple inputs, such as both text and image, then in generation time, an actionable solution taking text as input and another solution taking image as input will be generated and executed in parallel.

In summary, the key contributions of the work include:

- We introduce OpenAGI, an AGI research platform, specifically designed to offer complex, multi-step tasks accompanied by their respective datasets, evaluation methods, and a diverse range of extensible models which can be synthesized to effectively solve these tasks. The purpose of this platform is to aid in the quantification of the overarching planning and task-solving abilities of LLMs. OpenAGI embraces AGI by focusing on LLM-driven, (open-domain) model synthesis, predominantly utilizing models and datasets on Hugging Face and Github.
- We propose the LLM+RLTF approach for OpenAGI, which leverages a Large Language Model as a controller to select, synthesize and execute various external expert models for complex task-solving. The feedback obtained from these tasks is then employed to refine the LLM's planning strategy, thereby enhancing the LLM's overall performance and task-solving ability.
- We evaluate a variety of well-established LLMs[5] with differing scales (ranging from 770 million to 175 billion parameters) utilizing distinct learning schemas and the proposed OpenAGI pipeline. Our preliminary findings suggest that even smaller-scale LLMs, when paired with an appropriate learning schema such as RLTF, are able to possess the potential to outperform competitors that equip a significantly greater magnitude of model parameters.

2 Related Work

2.1 Large Language Models

With the advancement of highly parallelizable transformer architectures, pre-trained language models (PLMs) have demonstrated remarkable capabilities in comprehending, generating, and manipulating

[5]Our OpenAGI project started in 2022 based on the T5 language model series, which was before the release of ChatGPT. After ChatGPT was released, we still decided to keep the open-source LLMs (Flan-T5-Large and LLaMA) rather than only using ChatGPT (GPT-3.5) because we aim to contribute an open-source AGI research platform and benchmark to the community.

natural language [24, 17]. These models were pre-trained on a large corpora of unlabeled text data and commonly subsequently fine-tuned for specific downstream tasks. Shortly, the scaled-up PLMs, known as LLMs [26, 2, 21, 5, 42, 35], encompassed a substantially greater number of parameters and leverage vast amounts of training data. Consequently, LLMs exhibited enhanced capacity for learning intricate language patterns and structures, along with a notable reasoning ability. This results in superior performance across diverse natural language processing tasks [2, 35].

The achievements of LLMs are possibly due to the scaling laws for neural language models [12], which suggests the performances of the models depend primarily on the amount of model parameters, training data, and computing budget. T5 [26] is one of the first noticeable LLMs with up to 11 billion model parameters. It is an encoder-decoder model pre-trained on multiple tasks and reformulates them as text-to-text problems. Later, the release of GPT-3 [2] drew significant interest due to its surprising large model size containing 175 billion parameters, which made it the largest language model of its time. It was trained on a diverse range of text using unsupervised learning and revealed impressive capabilities on zero-shot and few-shot learning, which suggested LLMs are able to perform well on tasks even without explicit fine-tuning. T5 and GPT-3, along with their re-fined versions [6, 21] inspired the research community to keep extending the capacity of LLMs. Recently, more and more LLMs were released with continuously improved model size [25, 5, 42, 35, 20] and, as them should be, achieved superior performance on reasoning and language understanding.

2.2 Augmented Language Models

Although LLMs exhibit a robust capacity for comprehending complex human language, they may occasionally produce seemingly plausible yet inaccurate predictions and face challenges when addressing problems that require specialized domain expertise [16]. Consequently, the emerging field of Augmented Language Models (ALMs) focuses on addressing the limitations of conventional LLMs [6, 5, 2] by equipping them with enhanced reasoning capabilities and the ability to employ external resources [16]. The process of reasoning involves breaking down intricate assignments into smaller, more manageable subtasks that can be independently or collaboratively tackled by LLMs with the assistance of tools. What's more, LLMs can also invoke external tools or models to accomplish the relevant tasks. For exmaple, ToolFormer [33] introduces external API tags within text sequences, facilitating LLMs' access to external tools. Visual ChatGPT [39] is a new model that combines ChatGPT with Visual Foundation Models (VFMs) like Transformers, ControlNet, and Stable Diffusion, which acts as a bridge between users, allowing them to communicate via chat and generate visuals. HuggingGPT [32] integrates the Hugging Face hub with task-specific models around ChatGPT to tackle generalized AI tasks. Augmented language models may use these enhancements separately or joint them in a specific order to finish the specific task, which ultimately results in superior generalization capabilities.

Different from prior works in this field, we propose OpenAGI, an open-source AGI research platform designed to address the challenges commonly encountered in existing works, such as extensibility, nonlinear task planning, and quantitative evaluation. Furthermore, we introduce innovative methods into the learning schema of LLMs, including Reinforcement Learning from Task Feedback (RLTF) and nonlinear task planning, which address challenges on out-of-distribution (OOD) generalization, optimal task planning, and nonlinear task structures. We hope the OpenAGI platform can facilitate the open and long-term improvement and evaluation of AGI abilities in the community.

3 The OpenAGI Platform

3.1 Problem Definition

Given a set of natural-language-based task descriptions $\mathcal{T}$ and a set of datasets $\mathcal{D}$, where each element $\mathcal{D}_t$ represents the corresponding dataset for a specific task t, alongside a collection of functional models on $\mathcal{D}$, represented as $\mathcal{M}$, and their corresponding name set $\mathcal{N}$, the objective for a given LLM, denoted as $\mathcal{L}$, is to take a particular task description t as input and produce a multi-step solution s. This solution can be mapped to an arrangement of functional models (linearly or non-linearly) based on the model name set, ultimately working on the task-related dataset to accomplish the task. Consequently, one can employ any LLM derived from any learning schema to assess the planning capability of the LLM within this context, provided that $\mathcal{T}$, $\mathcal{D}$, $\mathcal{M}$, and $\mathcal{N}$ are supplied.

In this work, our primary objective is to assist AGI researchers in constructing an open-source pipeline, which will contribute to the community's long-term advancement and foster collaborative progress in the field. We introduce the details of our construction in the following sections. Specifically, instead of building complicated, multi-step tasks from scratch, we first explore the models 3.2 and datasets 3.3 that can be easily achieved, then create such tasks based on them.

3.2 Model Set

We now present the domain tasks and the corresponding models that can be employed in our platform. This set is designed to be flexible, allowing users to easily incorporate their own domain tasks and models. Our domain tasks are as follows:

- **Language-related Models** (corresponding models are shown in Table 1): **Sentiment Analysis** classifies the sentiment polarity of a given sentence; **Text Summarization** creates a text summary that represents the most important or relevant information within the original text content; **Machine Translation** converts a sentence from a source language to a target language; **Fill Mask** involves replacing masked words within a given text; **Question Answering (QA)** provides a textual answer of a question based on the given context.
- **Vision-related Models** (corresponding models are shown in Table 2): **Image Classification** aims to comprehend an entire image as a whole and assign it to a specific label; **Object Detection** identifies and localizes specific objects within an image by detecting their instances of a particular class; **Colorization** refers to the technique of adding plausible color information to monochromatic photographs or videos; **Image Super-resolution** generates a high-resolution (HR) image from a low-resolution (LR) image; **Image Denoising** aims to remove unwanted noise from an image while preserving its important features; **Image Deblurring** aims to recover a clear image from a blurred input image.
- **Vision-Language Models** (corresponding models are shown in Table 3): **Visual Question Answering (VQA)** involves answering questions based on an image; **Image Captioning** generates textual descriptions of the visual content depicted in an image; **Text-to-Image Generation** aims to generate images from a given input sentence or sequence of words.

Table 1: Language-related models

Domain Task	Input Modality	Output Modality	Model
Sentiment Analysis	Text	Text	FinBert [6][1]
Text Summarization	Text	Text	BART [7][13]
Machine Translation	Text	Text	T5 [8][26]
Fill Mask	Text	Text	DistilRoberta [9][15]
Question Answering	Text, Text	Text	DistilBERT [10][31]

[1]https://huggingface.co/yiyanghkust/finbert-tone
[2]https://huggingface.co/distilbert-base-cased-distilled-squad
[3]https://huggingface.co/facebook/bart-large-cnn
[4]https://huggingface.co/gpt2
[5]https://huggingface.co/t5-base
[6]https://huggingface.co/distilroberta-base
[7]https://huggingface.co/google/vit-base-patch16-224
[8]https://huggingface.co/facebook/detr-resnet-101
[9]https://github.com/richzhang/colorization
[10]https://huggingface.co/caidas/swin2SR-classical-sr-x2-64
[11]https://github.com/swz30/Restormer
[12]https://huggingface.co/microsoft/git-base-textvqa
[13]https://huggingface.co/nlpconnect/vit-gpt2-image-captioning
[14]https://huggingface.co/CompVis/stable-diffusion-v1-4

Table 2: Vision-related models

Domain Task	Input Modality	Output Modality	Model
Image Classification	Image	Text	ViT [11][9]
Object Detection	Image	Text	DETR [12][3]
Colorization	Image	Image	Colorizer [13][41]
Image Super-Resolution	Image	Image	Swin2SR [14][7]
Image Denoising	Image	Image	Restormer [15][40]
Image Deblurring	Image	Image	Restormer [40]

Table 3: Vision-language models

Domain Task	Input Modality	Output Modality	Model
Visual Question Answering	Image, Text	Text	GIT [16][36]
Image Captioning	Image	Text	Vision Encoder Decoder [17]
Text-to-Image Generation	Text	Image	StableDiffusion [18][28]

3.3 Tasks and Datasets

3.3.1 Raw Datasets

After selecting the appropriate models, choosing the raw datasets becomes a more straightforward process, provided that we ensure proper alignment between the datasets and the models' training sets. Raw datasets are provided as follows:

- **ImageNet-1K** [30] is a large-scale image dataset, derived from the broader ImageNet database, containing approximately 1 million images. These images are categorized into 1,000 distinct classes, with each class representing a specific object or concept. The dataset has been instrumental in the development and evaluation of state-of-the-art deep learning algorithms for image classification, object recognition, and transfer learning.
- **Common Objects in Context (COCO)** [14] is a large-scale, richly-annotated image dataset designed to advance the fields of object detection, segmentation, and captioning. Released in 2014, it contains over 200,000 labeled images with 1.5 million object instances from 80 different object categories. The dataset features complex, real-world scenes with multiple objects per image, various object scales, and diverse contexts.
- **CNN/Daily Mail** [19] is a valuable resource for text summarization, which consists of human-generated abstractive summaries, created by transforming news articles from CNN and Daily Mail websites into questions, with one entity concealed, and generating summaries from the corresponding passages. The authors have made available the scripts used to crawl, extract, and generate question-answer pairs from these websites. The corpus contains 286,817 training pairs, 13,368 validation pairs, and 11,487 test pairs, as defined by the scripts. On average, the source documents in the training set span 766 words across 29.74 sentences, while the summaries are composed of 53 words and 3.72 sentences.
- **Stanford Sentiment Treebank (SST2)** [22] is a corpus with labeled parse trees that allows for the analysis of the compositional effects of sentiment in language. The corpus consists of 11,855 single sentences extracted from movie reviews. It was parsed with the Stanford parser and includes a total of 215,154 unique phrases from those parse trees, each annotated by 3 human judges.
- **TextVQA** [34] serves as a benchmark for evaluating visual reasoning based on text present in images. In order to answer questions pertaining to the images, TextVQA necessitates models to read and reason about the text contained within them. The incorporation of text as a new modality in images demands that models be able to reason over this modality to address TextVQA queries. Thus, TextVQA poses a unique challenge for models to integrate both visual and textual cues to arrive at a comprehensive answer.

- **Stanford Question Answering Dataset (SQuAD)** [27] is a collection of question-answer pairs sourced from Wikipedia articles. A distinguishing characteristic of SQuAD is that the correct answers to the questions can be any sequence of tokens in the corresponding text. This flexibility is a result of the dataset's construction through crowdsourcing, which results in a diverse set of questions and answers compared to other question-answering datasets.

3.3.2 Data Augmentation Methods

Upon determining the raw datasets, our next objective is to augment them from various perspectives to construct complex, multi-step tasks. For instance, we can introduce noise and reduce the resolution of an image from ImageNet-1K to create new datasets that may require "Image Denoising" and "Image Super-Resolution" for initial recovery before doing classification. The data augmentation methods employed are as follows:

- **Gaussian Blur** is a prevalent image processing technique that involves convolving an image with a Gaussian filter kernel. This filter is applied to smooth the image and reduce noise, yielding a blurred output image.
- **Gaussian Noise** refers to the addition of Gaussian-distributed noise.
- **Grayscale** entails converting the colorful image to a grayscale image.
- **Low Resolution** pertains to images with a reduced pixel density (pixels per inch, or ppi).
- **Translation** denotes the process of converting a text from one language, such as English, to another, such as German. In this work, we only use English-to-German translator for simplicity.
- **Word Mask** randomly replaces a single word in a given sentence with the "[MASK]" token.

3.3.3 Multi-step Tasks

Drawing from the models presented in Tab. 1, 2, and 3, we categorize them according to input and output modalities as follows: 1) image in, image out; 2) image in, text out; 3) text in, image out; 4) text in, text out; 5) image-text pair in, text out; 6) text-text pair in, text out.

We employ data augmentation techniques discussed above to augment the raw datasets. Specifically, for tasks with image inputs, we can choose one or more techniques from the image augmentation method set {Gaussian Blur, Gaussian Noise, Grayscale, Low Resolution} to generate a compositionally augmented image input, which necessitates a multi-step image restoration process for recovery. Similarly, for tasks with text inputs or outputs, we choose one or more from {Translation, Word Mask} to generate a compositionally augmented text input or output. Furthermore, Visual Question Answering (VQA) and Question Answering (QA) are tasks with multiple multi-model inputs, resulting in natural tasks that cannot be solved with linear task planning solutions. Lastly, we integrate both aspects to construct complex, multi-step tasks. In total, we generate a total number of 185 complex multi-step tasks, with 117 tasks featuring a linear task structure and the remaining 68 tasks exhibiting a non-linear task structure.

A selection of task samples, along with their corresponding input and output data samples, can be found in Table 4. For illustration, consider the third row of Table 4, which represents a machine translation domain task (i.e., translating from English to German). In this case, we apply the "Word Mask" augmentation technique on the text inputs to create a multi-step task, which can be described as "Given clozed English text, how can the text be translated into German step by step?". For instance, given an original data sample, "A big burly grizzly bear is shown with grass in the background", the word "with" has been chosen to be masked to generate the augmented data sample, "A big burly grizzly bear is shown [MASK] grass in the background".

3.4 Evaluation Metrics

Given that OpenAGI comprises a diverse range of domain tasks with multi-modal data, we classify them according to domain tasks as well as input and output types. We then assess their performance using the following three metrics:

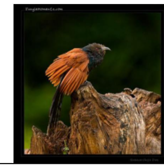
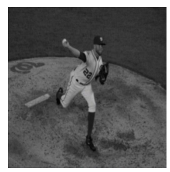
Task description	Input Sample	Output Sample
Given low-resolutioned noisy blurry grayscale image, how to return the regular image step by step?		
Given low-resolutioned noisy blurry grayscale image, how to return the object names in English step by step?		bear
Given clozed English text, how to translate the text in German step by step?	A big burly grizzly bear is show [Mask] grass in the background.	Ein kräftiger Grizzly Bär ist im Hintergrund mit Gras zu sehen.
Given noisy blurry grayscale image and clozed English query, how to answer the question in English step by step?	 **Question**: what number is [Mask] the player's jersey?	22
Given clozed English document and clozed English query, how to answer the question in German step by step?	**Context**: Super Bowl 5 was an American football game to determine the champion of the National... **Question**: What was the theme of Super [Mask] 50?	Goldener Jahrestag

Table 4: Examples of multi-step tasks and their augmented data samples.

- **CLIP Score**[19] is a reference-free metric used to assess the correlation between a generated image caption and the actual content of the image.
- **BERT Score**[20] uses contextual embeddings from the pre-trained BERT model to compare words in candidate and reference sentences through cosine similarity. Additionally, BERT Score calculates precision, recall, and F1 measure, making it a valuable tool for evaluating various language generation tasks. In this work, we use the value of F1 score.
- **ViT Score**[21] is a metric designed to assess the visual similarity between two images. By calculating the cosine similarity of their respective embeddings, which are generated using a Vision Transformer, the ViT Score offers a quantitative measure of their likeness.

In particular, we employ the CLIP Score only for Text-to-Image Generation-based tasks, the BERT Score is utilized to assess tasks with text outputs, and the ViT score is applied to measure image similarity for the remaining tasks with image outputs. We also normalize the BERT and CLIP scores.

[19]https://torchmetrics.readthedocs.io/en/stable/multimodal/clip_score.html
[20]https://huggingface.co/spaces/evaluate-metric/bertscore
[21]https://colab.research.google.com/github/huggingface/notebooks/blob/main/examples/image_similarity.ipynb

4 Reinforcement Learning from Task Feedback (RLTF)

While learning solely from input text is a powerful method for training LLMs, it is not sufficient for handling real-world tasks that require a deeper understanding of context and environment. One potential method to improve the capabilities of LLMs is to incorporate reinforcement learning (RL) techniques. By merging the strengths of RL, LLMs can gain additional insights from trial-and-error experiences. This leads to more robust and adaptive models, especially in situations where labeled data is scarce or when tasks involve physical interactions. In this work, we propose Reinforcement Learning from Task Feedback (RLTF), which utilizes task feedback to supply more information that guides the learning direction of LLMs, resulting in improved and more efficient strategies.

In the setup of RLTF, the environment is the proposed OpenAGI platform and the agent is the LLM $\mathcal{L}$ parameterized with Φ. The solution s generated by the LLM can be seen as a set of instructions that solve the input task t and can be executed on the corresponding augmented dataset $\mathcal{D}_t$. We can use the performance (provided in Sec. 3.4) on that dataset as the reward signal $\mathcal{R}$ and use reinforcement learning to fine-tune the LLM. More concretely, to find the optimal solution, we require the LLM to maximize its expected reward on the training set $\mathcal{T}_{train}$, represented by $J(\Phi)$:

$$J(\Phi) = \mathbb{E}_{\mathbf{s}_{train} \sim \mathcal{L}(\mathcal{T}_{train}|\Phi)}[\mathcal{R}] \tag{1}$$

Since the reward signal $\mathcal{R}$ is non-differentiable, we need to use a policy gradient method to iteratively update Φ. In this work, we use the REINFORCE in [38] as follows,

$$\nabla_\Phi J(\Phi) = \mathbb{E}_{P(\mathbf{s}_{train}|\Phi)}\left[\nabla_\Phi \log P\left(\mathbf{s}_{train}|\Phi\right) \cdot \mathcal{R}\right] \tag{2}$$

An empirical approximation of the above quantity is:

$$\nabla_\Phi J(\Phi) \approx \frac{1}{|\mathcal{T}_{train}|} \sum_{t \in \mathcal{T}_{train}} \nabla_\Phi \log P(s_{train}|\Phi) \cdot \mathcal{R} \tag{3}$$

The above update is an unbiased estimate for our gradient, but has a very high variance. In order to reduce the variance of this estimate, following [43, 23], we employ a baseline function b, which is the moving average of the previous reward signals:

$$\nabla_\Phi J(\Phi) \approx \frac{1}{|\mathcal{T}_{train}|} \sum_{t \in \mathcal{T}_{train}} \nabla_\Phi \log P(s_{train}|\Phi) \cdot (\mathcal{R} - b) \tag{4}$$

5 Nonlinear Task Planning

To generate the solution for a natural language task description, we require the LLM to generate an actionable solution consisting of sequences of model names. For tasks that require only one input, the model only needs to generate one actionable sequence of models. For tasks that require multiple inputs, such as Visual Question Answering, the LLM needs multiple steps in order to accomplish the task, where each step is either a sequence of models or a parallel of several sequences of models. Towards this end, the LLM must satisfy three conditions: 1) generate only model names without irrelevant tokens, 2) generate valid sequences of models, and 3) generate paralleled sequences of models for different inputs when necessary.

Condition 1: For the LLM to generate only model names, instead of tuning the model to teach-force it what names are available, we adopt constrained beam search [8], which only allows generating tokens from the $\mathcal{M}$ at every decoding step. More specifically, we define our constraints as a prefix trie such that each model name is a path from the root to some leaf node. For each node t in the tree, its children indicate all the allowed continuations from the prefix defined traversing the trie from the root to t. Thus in each decoding step, the next token can only be selected from either all possible continuations allowed based on generated tokens or the first tokens of all possible next model names. For example, if "Text" is already generated, based on the set of model names, the next tokens can only be either "Summarization" due to the "Text Summarization" model or "Generation" due to the "Text Generation" model, as shown in Fig. 4.

Condition 2: For the LLM to generate valid sequences of models, consecutive models should have input and output modalities matched. If the output modality of a model is text, then the next model

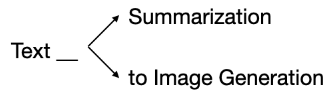


Figure 4: Model name based constrained generation.

can only be models that take text as input. This is also achieved by constrained beam search such that when finishing generating one model, the constraint function will determine the output modality of this model and find out all possible next models in $\mathcal{M}$, excluding the models that are already generated. It will dynamically construct a new trie for all these model names based on the output modality. For example, if the first generated model name is "Text Summarization", then the next possible models can be "Sentiment Analysis", "Text Generation", etc., as shown in Fig. 5.

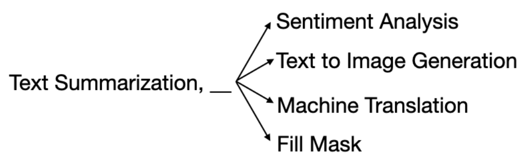


Figure 5: Model type based constrained generation.

If a task requires only one input, Conditions 1 and 2 can guarantee a valid sequence. However, if the task requires multiple inputs to generate the final result, each input may require a valid sequence before utilizing a multi-input model such as Question Answering and Visual Question Answering. In this scenario, a sequential solution is unsatisfying because different inputs should be processed in parallel. To handle this problem, we have the following Condition 3.

Condition 3: Autoregressive decoding in language models is generally unsuitable for generating parallel valid sequences. In this work, we use beam search to conduct semi-autoregressive generation. Beam search is originally proposed such that multiple hypotheses are generated to compete with each other in order to obtain the highest-scored output. We instead utilize beam search as an efficient semi-autoregressive decoding method [29] such that for each decoding step in beam search, different hypotheses are treated as parallel valid solutions for different inputs instead of competing hypotheses. If a task requires multiple inputs, such as both text and image, then in generation time, a model taking text as input and a model taking image as input are almost equally likely to be generated. Since based on constrained generation, each beam is a valid model sequence eventually, thus, multiple valid sequences with different input types will be generated in parallel.

When parallel processing is conducted, multi-input models and subsequent models are required. We concatenate the generated sequences with the natural language task description to generate a new prompt to prompt subsequent models. This process can be done recursively until the end-of-sentence token is generated without any more models, as illustrated in Fig. 6.

6 Experiments

6.1 Backbone LLMs

We employ both ChatGPT (GPT-3.5-turbo) and two other open-source large language models for experimentation.

- **GPT-3.5-turbo.** The GPT (Generative Pre-trained Transformer) series [2], developed by OpenAI, consists of advanced language models. GPT-3.5, a fine-tuned version of GPT-3, boasts over 175 billion model parameters.
- **LLaMA-7b.** LLaMA [35] is a lightweight, open-source language model developed by researchers at Meta. It is designed to be efficient and performant, and can be run on a single GPU. In this work, we use the 7-billion size model of LLaMA.

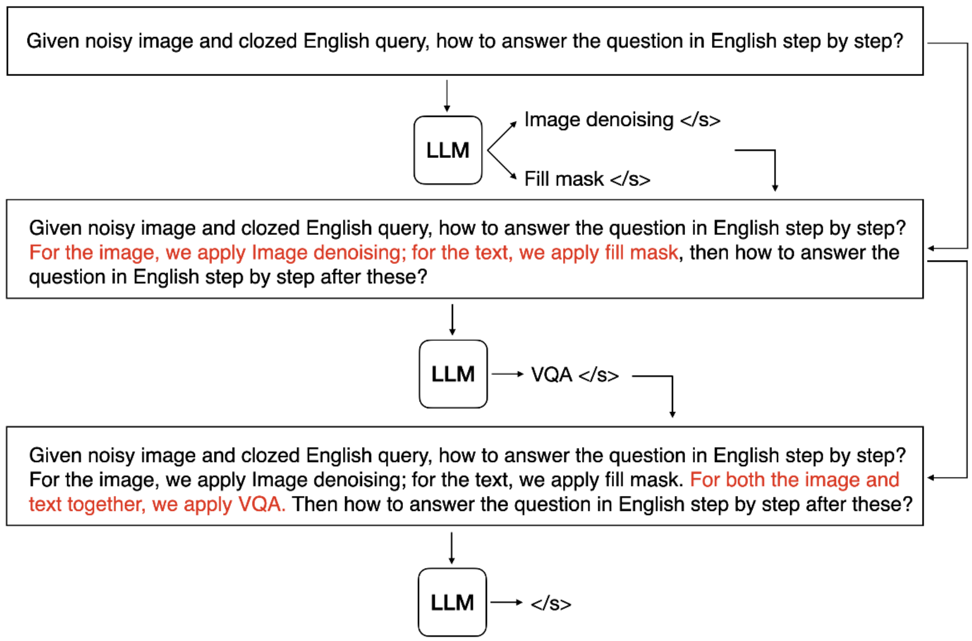


Figure 6: Parallel recursive sequence generation

- **Flan-T5-Large.** Flan-T5 [6] is a series of language models developed by Google. Flan-T5 models are fine-tuned using a technique called instruction finetuning, which allows them to learn from a wider range of data and improve their performance on a variety of tasks. Flan-T5-Large has 770 million parameters.

6.2 Learning Schema of LLMs

We also employ the following LLM learning Schema for experimentation.

- **Zero-shot Learning (Zero)** is to simply feed the task description to the model and ask for results.
- **Few-shot Learning (Few)** presents a set of high-quality demonstrations, each consisting of both input and desired output, on the target task. As the model first sees good examples, it can better understand human intention and criteria for what kinds of answers are wanted.
- **Fine-tuning** involves using manually labeled data samples as additional training signals to refine and adapt pre-trained LLMs to specific tasks or domains. In this setting, we also use constrained generation method introduced in Sec. 5.
- **RLTF** is our proposed method in Sec. 4.

In the context of zero-shot and few-shot learning paradigms, LLMs are allowed to produce free-form output solutions. To transform these outputs into viable task planning solutions, we employ text similarity models to map them to our model name set $\mathcal{N}$, which is an established method in existing works [10]. For fine-tuning and RLTF approaches, we utilize constrained generation to directly generate the task planning solution. All the mapped or constrained generated task planning solutions are then fed to OpenAGI, to get executed and evaluated.

6.3 Datasets

We divide the tasks in OpenAGI into training and testing sets. In particular, we randomly select 10% of tasks, along with their corresponding datasets, based on input and output modalities for training purposes. For Few-shot Learning and Fine-tuning, we supply manually curated, feasible solutions as ground-truth labels. In the case of RLTF, we employ the Fine-tuning checkpoint as a reasonable initialization for LLM to reduce the likelihood of producing infeasible solutions. Moreover, considering the fact that the imbalanced number of tasks with different input and output modalities could lead to skewed measurement results, we choose an additional 10% of tasks, adhering to the same selection criteria as mentioned above, to serve as the test set. To counteract the influence of randomness, the test set is randomly sampled multiple times, and the average performance is calculated.

6.4 Experimental Analysis

The experimental results are presented in Tab. 5, the overall performance is calculated as the weighted average of CLIP, BERT and ViT scores. GPT-3.5-turbo exhibits superior performance in both zero-shot and few-shot learning settings compared to LLaMA-7b and Flan-T5-Large. This is evident from the higher scores it achieves in BERT, ViT score, and the overall performance. LLaMA-7b, while not performing as well as GPT-3.5-turbo, demonstrates better overall performance in few-shot learning compared to its zero-shot learning performance. However, its performance is still much lower than that of GPT-3.5-turbo in the same settings. Flan-T5-Large shows significant improvement when using fine-tuning or Reinforcement Learning from Task Feedback (RLTF) compared to zero-shot and few-shot learning strategies.

To facilitate a comprehensive analysis of the results, we present the zero-shot and few-shot solutions in Tab. 6 and 7, respectively. Initially, it is evident that in the zero-shot setting, most LLMs struggle to generate valid task planning, let alone optimal solutions. In particular, GPT-3.5 tends to generate repetitive content, which subsequently maps to identical model names. Meanwhile, LLaMA-7b and Flan-T5-Large, constrained by their zero-shot capabilities, fail to produce a reasonable plan. In the few-shot setting, we incorporate several manually labeled task plans as instructions to guide the generation, resulting in a remarkable improvement in the quality of the task plans. As observed in Tab. 7, all three LLMs can produce solutions that are semantically similar to the provided examples. In fact, many solutions can be utilized directly, even without the need for mapping.

Table 5: OpenAGI task-solving performances under different settings

Metrics	GPT-3.5-turbo		LLaMA-7b		Flan-T5-Large			
	Zero	Few	Zero	Few	Zero	Few	Fine-tuning	RLTF
CLIP Score	0	0	0	0	0	0	0.3059	0.3059
BERT Score	0.1914	0.3820	0	0.1781	0	0.2488	0.1166	0.2554
ViT Score	0.2437	0.7497	0	0	0	0	0.6285	0.6551
Overall	0.2284	0.4335	0	0.1272	0	0.1777	0.1957	0.3446

7 Conclusion and Future Work

In this work, we introduce OpenAGI, an open-source AGI research platform designed to facilitate the development and evaluation of large language models (LLMs) in solving complex, multi-step tasks through manipulating various domain expert models. OpenAGI provides a wide range of extensible models and datasets, predominantly utilizing resources from Hugging Face and GitHub. We also propose the LLM+RLTF approach, which combines LLMs with reinforcement learning to optimize task-solving performance. The evaluation of various LLMs using the OpenAGI pipeline and different learning schema demonstrates that smaller-scale LLMs can potentially outperform larger models when combined with the appropriate learning approach, such as RLTF.

In future research, we aim to incorporate multiple models within each single-step task, thereby providing an expanded selection of options for LLMs to address out-of-distribution (OOD) problems. Additionally, we intend to integrate datasets from alternative modalities, such as video and audio, into our OpenAGI platform. These datasets will facilitate the development of more sophisticated tasks to further investigate the planning capabilities of LLMs. We will also endeavor to enhance the evaluation mechanism to enable a more accurate and comprehensive assessment of performance. Another promising direction is to involve humans in the loop in the resolution of complex tasks. In such scenarios, LLM may prompt human experts for answers as one step of the task-solving plan when a suitable model is unavailable, thus enabling better human-machine collaboration. Lastly, we aim to explore automated task generation techniques that empower OpenAGI to generate complex tasks independently, facilitating self-prompting and improvement in its task-solving capabilities.

References

[1] Dogu Araci. 2019. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063* (2019).

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*. Springer, 213–229.

[4] Hanxiong Chen, Yunqi Li, He Zhu, and Yongfeng Zhang. 2022. Learn Basic Skills and Reuse: Modularized Adaptive Neural Architecture Search (MANAS). In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 169–179.

[5] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311* (2022).

[6] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416* (2022).

[7] Marcos V Conde, Ui-Jin Choi, Maxime Burchi, and Radu Timofte. 2022. Swin2SR: Swinv2 transformer for compressed image super-resolution and restoration. *arXiv preprint arXiv:2209.11345* (2022).

[8] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive entity retrieval. *arXiv preprint arXiv:2010.00904* (2020).

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

[10] Yaru Hao, Haoyu Song, Li Dong, Shaohan Huang, Zewen Chi, Wenhui Wang, Shuming Ma, and Furu Wei. 2022. Language models are general-purpose interfaces. *arXiv preprint arXiv:2206.06336* (2022).

[11] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*. PMLR, 9118–9147.

[12] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361* (2020).

[13] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* (2019).

[14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.

[15] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[16] Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, et al. 2023. Augmented language models: a survey. *arXiv preprint arXiv:2302.07842* (2023).

[17] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heinz, and Dan Roth. 2021. Recent advances in natural language processing via large pre-trained language models: A survey. *arXiv preprint arXiv:2111.01243* (2021).

[18] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332* (2021).

[19] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023* (2016).

[20] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[21] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* 35 (2022), 27730–27744.

[22] Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *arXiv preprint cs/0506075* (2005).

[23] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. 2018. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*. PMLR, 4095–4104.

[24] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences* 63, 10 (2020), 1872–1897.

[25] Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446* (2021).

[26] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* 21, 140 (2020), 1–67.

[27] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* (2016).

[28] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10684–10695.

[29] Ohad Rubin and Jonathan Berant. 2020. SmBoP: Semi-autoregressive bottom-up semantic parsing. *arXiv preprint arXiv:2010.12412* (2020).

[30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115, 3 (2015), 211–252. `https://doi.org/10.1007/s11263-015-0816-y`

[31] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[32] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761* (2023).

[33] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in HuggingFace. *arXiv preprint arXiv:2303.17580* (2023).

[34] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. 2019. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8317–8326.

[35] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[36] Jianfeng Wang, Zhengyuan Yang, Xiaowei Hu, Linjie Li, Kevin Lin, Zhe Gan, Zicheng Liu, Ce Liu, and Lijuan Wang. 2022. Git: A generative image-to-text transformer for vision and language. *arXiv preprint arXiv:2205.14100* (2022).

[37] Zihao Wang, Shaofei Cai, Anji Liu, Xiaojian Ma, and Yitao Liang. 2023. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560* (2023).

[38] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3 (1992), 229–256.

[39] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671* (2023).

[40] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. 2022. Restormer: Efficient transformer for high-resolution image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5728–5739.

[41] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S. Lin, Tianhe Yu, and Alexei A. Efros. 2017. Real-Time User-Guided Image Colorization with Learned Deep Priors. *ACM Trans. Graph.* 36, 4, Article 119 (jul 2017), 11 pages. `https://doi.org/10.1145/3072959.3073703`

[42] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068* (2022).

[43] Barret Zoph and Quoc V. Le. 2017. Neural Architecture Search with Reinforcement Learning. In *5th ICLR*. `https://openreview.net/forum?id=r1Ue8Hcxg`

Appendix

LLM	Output	Mapped Task Planning
Prompt	Problem: Given low-resolution blurry grayscale image, how to return the regular image step by step? What is its solution? Use “Step” to mark.	
GPT-3.5	Step 1: Collect more data or images to create a higher resolution version of the blurry image. Step 2: Use a deblurring algorithm to remove the blurriness and increase the sharpness of the image. Step 3: Apply a denoising algorithm to reduce any noise in the image that may have been introduced during the deblurring process. Step 4: Use an upsampling method to increase the resolution of the image to the desired level. Step 5: Apply a contrast enhancement technique to make the image more visually appealing. Step 6: Finally, apply sharpening techniques to add crispness and detail to the image.	Image Deblurring, Image Deblurring, Image Deblurring, Image Super Resolution, Colorization, Image Denoising
LLaMA-7b	1. Use the image to generate the mask image. 2. Use the mask image to generate the masked image. 3. Use the masked image to generate the final image. 4. Use the masked image to generate the final image. 5. Use ... 6. ...	Non-sense.
Flan-T5-Large	The blurry image is a blurry image with a blurry background. To return the regular image, you can use the following steps: First, you need to find the blurry image. Then, you need to find the regular image. Then, you need to find the blurry image. Then, you need to find the regular image. Then, you need to find the blurry image. Then, you need to find the regular image. Then, ...	Non-sense.

Table 6: Example of Zero-shot Prompt.

LLM		Mapped Task Planning
Prompt	Problem: Given low-resolutioned noisy blurry gray image, how to return the regular image step by step? Solution: Step 1:Image Super Resolution, Step 2: Image Denoising, Step 3: Image Deblurring, Step 4: Colorization, ... Problem: Given clozed English text, how to translate the text in German step by step? Solution: Step 1:Fill Mask, Step 2: Machine Translation, Problem: Given low-resolutioned blurry grayscale image, how to return the regular image step by step? Solution:	
GPT-3.5	Step 1: Image Super Resolution, Step 2: Image Deblurring, Step 3: Image Denoising.	Image Super Resolution, Image Deblurring, Image Denoising
LLaMA-7b	Prblem: Given low-resolutioned noisy blurry gray image, how to return the regular image step by step? Solution: Step 1:Image Super Resolution, Step 2: Image Denoising, Step 3: Image Deblurring, Step 4: Colorization, Prblem: Given noisy blurry gray image, how to return the caption in Chinese step by step? Solution: Step 1:Image Denoising, Step 2: Image Deblurring, ... Problem: Given low-resolutioned blurry grayscale image, how to return the regular image step by step? Soltuion: Step 1:Image Super Resolution, Step 2: Image Deblurring, Step 3: Colorization, Step 4: Image Classification, Step 5: Machine Translation,	Image Super Resolution, Image Deblurring, Colorization, Image Classification, Machine Translation
Flan-T5-Large	Step 1: Image Super Resolution, Step 2: Image Deblurring, Step 3: Colorization, Step 4: Image Captioning, Step 5: Machine Translation,	Image Super Resolution, Image Deblurring, Colorization, Image Captioning, Machine Translation

Table 7: Example of Few-shot Prompt.