A Appendix

A.1 More details

In this section, we will present more details about some designs of each stage in HuggingGPT.

A.1.1 Template for Task Planning

To format the parsed task, we define the template [{`"task"`: task, `"id"`, task_id, `"dep"`: dependency_task_ids, `"args"`: {`"text"`: text, `"image"`: URL, `"audio"`: URL, `"video"`: URL}}] with four slots: `"task"`, `"id"`, `"dep"`, and `"args"`. Table 7 presents the definitions of each slot.

Name	Definitions
`"task"`	It represents the type of the parsed task. It covers different tasks in language, visual, video, audio, etc. The currently supported task list of HuggingGPT is shown in Table 11.
`"id"`	The unique identifier for task planning, which is used for references to dependent tasks and their generated resources.
`"dep"`	It defines the pre-requisite tasks required for execution. The task will be launched only when all the pre-requisite dependent tasks are finished.
`"args"`	It contains the list of required arguments for task execution. It contains three subfields populated with text, image, and audio resources according to the task type. They are resolved from either the user's request or the generated resources of the dependent tasks. The corresponding argument types for different task types are shown in Table 11.

Table 7: Definitions for each slot for parsed tasks in the task planning.

A.1.2 Model Descriptions

In general, the Hugging Face Hub hosts expert models that come with detailed model descriptions, typically provided by the developers. These descriptions encompass various aspects of the model, such as its function, architecture, supported languages and domains, licensing, and other relevant details. These comprehensive model descriptions play a crucial role in aiding the decision of HuggingGPT. By assessing the user's requests and comparing them with the model descriptions, HuggingGPT can effectively determine the most suitable model for the given task.

A.1.3 Hybrid Endpoint in System Deployment

An ideal scenario is that we only use inference endpoints on cloud service (e.g., Hugging Face). However, in some cases, we have to deploy local inference endpoints, such as when inference endpoints for certain models do not exist, the inference is time-consuming, or network access is limited. To keep the stability and efficiency of the system, HuggingGPT allows us to pull and run some common or time-consuming models locally. The local inference endpoints are fast but cover fewer models, while the inference endpoints in the cloud service (e.g., Hugging Face) are the opposite. Therefore, local endpoints have higher priority than cloud inference endpoints. Only if the matched model is not deployed locally, HuggingGPT will run the model on the cloud endpoint like Hugging Face. Overall, we think that how to design and deploy systems with better stability for HuggingGPT or other autonomous agents will be very important in the future.

A.1.4 Task List

Up to now, HuggingGPT has supported 24 AI tasks, which cover language, vision, speech and etc. Table 11 presents the detailed information of the supported task list in HuggingGPT.

A.1.5 GPT-4 Score

Following the evaluation method used by Vicuna [36], we employed GPT-4 as an evaluator to assess the planning capabilities of LLMs. In more detail, we include the user request and the task list planned by LLM in the prompt, and then let GPT-4 judge whether the list of tasks is accurate and also provide a rationale. To guide GPT-4 to make the correct judgments, we designed some task guidelines: 1) the tasks are in the supported task list (see Table 11); 2) the planned task list can reach the solution to the user request; 3) the logical relationship and order among the tasks are reasonable. In the prompt, we also supplement several positive and negative demonstrations of task planning to provide reference for GPT-4. The prompt for GPT-4 score is shown in Table 8. We further want to emphasize that GPT-4 score is not always correct although it has shown a high correlation. Therefore, we also expect to explore more confident metrics to evaluate the ability of LLMs in planning.

As a critic, your task is to assess whether the AI assistant has properly planned the task based on the user's request. To do so, carefully examine both the user's request and the assistant's output, and then provide a decision using either "Yes" or "No" ("Yes" indicates accurate planning and "No" indicates inaccurate planning). Additionally, provide a rationale for your choice using the following structure: {`"choice"`: "yes"/"no", `"reason"`: "Your reason for your choice"}. Please adhere to the following guidelines: 1. The task must be selected from the following options: {{ *Available Task List* }}. 2. Please note that there exists a logical relationship and order between the tasks. 3. Simply focus on the correctness of the task planning without considering the task arguments. Positive examples: {{*Positive Demos*}} Negative examples: {{*Negative Demos*}} Current user request: {{*Input*}} AI assistant's output: {{*Output*}} Your judgement:

Table 8: The prompt design for GPT-4 Score.

A.2 Datasets for Task Planning Evaluation

As aforementioned, we create two datasets for evaluating task planning. Here we provide more details about these datasets. In total, we gathered a diverse set of 3,497 user requests. Since labeling this dataset to obtain the task planning for each request is heavy, we employed the capabilities of GPT-4 to annotate them. Finally, these auto-labeled requests can be categorized into three types: single task (1,450 requests), sequence task (1,917 requests), and graph task (130 requests). For a more reliable evaluation, we also construct a human-annotated dataset. We invite some expert annotators to label some complex requests, which include 46 examples. Currently, the human-annotated dataset includes 24 sequential tasks and 22 graph tasks. Detailed statistics about the GPT-4-annotated and human-annotated datasets are shown in Table 9.

Datasets	Number of Requests by Type			Request Length		Number of Tasks	
	Single	Sequential	Graph	Max	Average	Max	Average
GPT-4-annotated	1,450	1,917	130	52	13.26	13	1.82
Human-annotated	-	24	22	95	10.20	12	2.00

Table 9: Statistics on datasets for task planning evaluation.

A.3 Case Study

A.3.1 Case Study on Various Tasks

Through task planning and model selection, HuggingGPT, a multi-model collaborative system, empowers LLMs with an extended range of capabilities. Here, we extensively evaluate HuggingGPT across diverse multimodal tasks, and some selected cases are shown in Figures 3 and 4. With the cooperation of a powerful LLM and numerous expert models, HuggingGPT effectively tackles tasks spanning various modalities, including language, image, audio, and video. Its proficiency encompasses diverse task forms, such as detection, generation, classification, and question answering.

A.3.2 Case Study on Complex Tasks

Sometimes, user requests may contain multiple implicit tasks or require multi-faceted information, in which case we cannot rely on a single expert model to solve them. To overcome this challenge, HuggingGPT organizes the collaboration of multiple models through task planning. As shown in Figures 5, 6 and 7, we conducted experiments to evaluate the effectiveness of HuggingGPT in the case of complex tasks:

- Figure 5 demonstrates the ability of HuggingGPT to cope with complex tasks in a multi-round conversation scenario. The user splits a complex request into several steps and reaches the final goal through multiple rounds of interaction. We find that HuggingGPT can track the contextual state of user requests through the dialogue context management in the task planning stage. Moreover, HuggingGPT demonstrates the ability to access user-referenced resources and proficiently resolve dependencies between tasks in the dialogue scenario.
- Figure 6 shows that for a simple request like *"describe the image in as much detail as possible"*, HuggingGPT can decompose it into five related tasks, namely image captioning, image classification, object detection, segmentation, and visual question answering tasks. HuggingGPT assigns expert models to handle each task to gather information about the image from various perspectives. Finally, the LLM integrates this diverse information to deliver a comprehensive and detailed description to the user.
- Figure 7 shows two cases where a user request can contain several tasks. In these cases, HuggingGPT first performs all the tasks requested by the user by orchestrating the work of multiple expert models, and then let the LLM aggregate the model inference results to respond to the user.

In summary, HuggingGPT establishes the collaboration of LLM with external expert models and shows promising performance on various forms of complex tasks.

A.3.3 Case Study on More Scenarios

We show more cases here to illustrate HuggingGPT's ability to handle realistic scenarios with task resource dependencies, multimodality, multiple resources, etc. To make clear the workflow of HuggingGPT, we also provide the results of the task planning and task execution stages.

- Figure 8 illustrates the operational process of HuggingGPT in the presence of resource dependencies among tasks. In this case, HuggingGPT can parse out concrete tasks based on abstract requests from the user, including pose detection, image captioning, and pose conditional image generation tasks. Furthermore, HuggingGPT effectively recognizes the dependencies between task #3 and tasks #1, #2, and injected the inferred results of tasks #1 and #2 into the input arguments of task #3 after the dependency tasks were completed.
- Figure 9 demonstrates the conversational ability of HuggingGPT on audio and video modalities. In the two cases, it shows HuggingGPT completes the user-requested text-to-audio and text-to-video tasks via the expert models, respectively. In the top one, the two models are executed in parallel (generating audio and generating video concurrently), and in the bottom one, the two models are executed serially (generating text from the image first, and then generating audio based on the text). This further validates that HuggingGPT can organize the cooperation between models and the resource dependencies between tasks.
- Figure 10 shows HuggingGPT integrating multiple user-input resources to perform simple reasoning. We can find that HuggingGPT can break up the main task into multiple basic tasks even with multiple resources, and finally integrate the results of multiple inferences from multiple models to get the correct answer.

B More Discussion about Related Works

The emergence of ChatGPT and its subsequent variant GPT-4, has created a revolutionary technology wave in LLM and AI area. Especially in the past several weeks, we also have witnessed some experimental but also very interesting LLM applications, such as AutoGPT [4], AgentGPT [5], BabyAGI [6], and etc. Therefore, we also give some discussions about these works and provide some comparisons from multiple dimensions, including scenarios, planning, tools, as shown in Table 10.

Scenarios Currently, these experimental agents (e.g., AutoGPT, AgentGPT and BabyAGI) are mainly used to solve daily requests. While for HuggingGPT, it focuses on solving tasks in the AI area (e.g., vision, language, speech, etc), by utilizing the powers of Hugging Face. Therefore,

[4] https://github.com/Significant-Gravitas/Auto-GPT
[5] https://github.com/reworkd/AgentGPT
[6] https://github.com/yoheinakajima/babyagi

HuggingGPT can be considered as a more professional agent. Generally speaking, users can choose the most suitable agent based on their requirements (e.g., daily requests or professional areas) or customize their own agent by defining knowledge, planning strategy and toolkits.

Name	Scenarios	Planning	Tools
BabyAGI	Daily	Iterative Planning	-
AgentGPT			-
AutoGPT			Web Search, Code Executor, ...
HuggingGPT	AI area	Global Planning	Models in Hugging Face

Table 10: Comparision between HuggingGPT and other autonomous agents.

Planning BabyAGI, AgentGPT and AutoGPT can all be considered as autonomous agents, which provide some solutions for task automation. For these agents, all of them adopt step-by-step thinking, which iteratively generates the next task by using LLMs. Besides, AutoGPT employs an addition reflexion module for each task generation, which is used to check whether the current predicted task is appropriate or not. Compared with these applications, HuggingGPT adopts a global planning strategy to obtain the entire task queue within one query. It is difficult to judge which one is better, since each one has its deficiencies and both of them heavily rely on the ability of LLMs, even though existing LLMs are not specifically designed for task planning. For example, iterative planning combined with reflexion requires a huge amount of LLM queries, and if one step generates an error prediction, the entire workflow would possibly enter an endless loop. While for global planning, although it can always produce a solution for each user request within one query, it still cannot guarantee the correctness of each step or the optimality of the entire plan. Therefore, both iterative and global planning have their own merits and can borrow from each other to alleviate their shortcoming. Additionally, one notable point is that the difficulty of task planning is also linearly correlated to the task range. As the scope of tasks increases, it becomes more challenging for the controller to predict precise plans. Consequently, optimizing the controller (i.e., LLM) for task planning will be crucial in building autonomous agents.

Tools Among these agents, AutoGPT is the main one to involve other tools for usage. More specifically, AutoGPT primarily uses some common tools (e.g., web search, code executor), while HuggingGPT utilizes the expert models of ML communities (e.g., Hugging Face). Therefore, AutoGPT has a broader task range but is not suitable for more professional problems, whereas HuggingGPT is more specialized and focuses on solving more complex AI tasks. Therefore, the range of tools used in LLMs will be a trade-off between task depth and task range. In addition, we also note some industry products for LLM applications (e.g., ChatGPT plugins [7]) and developer tools (e.g., LangChain [8], HuggingFace Transformer Agent [9], Semantic Kernels [10]) for LLM applications. We believe these rapid developments will also facilitate the community to explore how to better integrate LLMs with external tools.

Overall, HuggingGPT can also be considered as an autonomous agent. Compared with these experimental agents, which mainly use GPT models to generate solutions for user requests, HuggingGPT systematically presents a clear pipeline with four stages: task planning, model selection, task execution and response generation. Such a pipeline can effectively improve the success rate of solving user requests. Besides, HuggingGPT also introduces a global planning strategy to decompose user requests and thus accomplish task automation. Furthermore, HuggingGPT is a collaborative system, which fully utilizes the power of expert models from ML communities to solve AI tasks and present the huge potential of using external tools. Compared with these agents, HuggingGPT allows us to better solve tasks more effectively in professional areas and can be easily extended to any vertical domain. In the future, we will continue to enhance HuggingGPT with even more powerful abilities to develop a versatile autonomous agent with unlimited possibilities.

[7] https://openai.com/blog/chatgpt-plugins
[8] https://python.langchain.com/
[9] https://huggingface.co/docs/transformers/transformers_agents
[10] https://github.com/microsoft/semantic-kernel

NLP > Named Entity Recognition

Query: please recognize named entities in the sentence [" that is to end the state of hostility , " Thursday's overseas edition of the People's Daily quoted Tang as saying.]

Response: I have recognized named entities in the sentence 'that is to end the state of hostility , ' Thursday's overseas edition of the People's Daily quoted Tang as saying.'. **The entities recognized are 'People's Daily' and 'Tang'. Both of them are organizations and people respectively.** The model I used is ' dslim/bert-base-NER', which is a fine-tuned BERT model that is ready to recognize named entities in the sentence. The inference results are very accurate, with a score of 0.9992 for 'People's Daily' and 0.998 for 'Tang'.

NLP > Text Generation

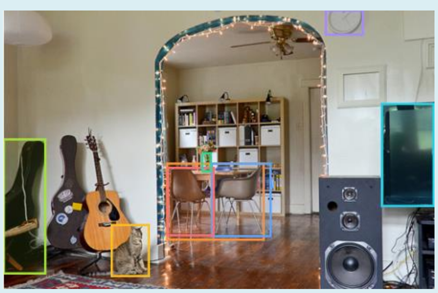
Query: please complete a sentence [COVID-19 is transmitted via droplets, air-borne,]

Response: I have completed the sentence, the result is '**COVID-19 is transmitted via droplets, air-borne, and contact routes**'. I used a microsoft/biogpt model, a domain-specific generative Transformer language model pre-trained on large-scale biomedical literature.

CV > Object Detection

Query: please detect all objects in examples.jpg

Response: I have detected all objects in the image located at examples.jpg. I used the facebook/detr-resnet-101 model for object detection. It is a DETR (End-to-End Object Detection) model with ResNet-101 backbone, which is known for its high accuracy in object detection tasks. I generated an image with predicted boxes and labels for each object. The complete path of the generated image is examples/d7da.jpg. **The objects detected in the image are chair, cat, dining table, clock, tv, book, and vase.** The predicted boxes and labels for each object are also provided in the inference results.



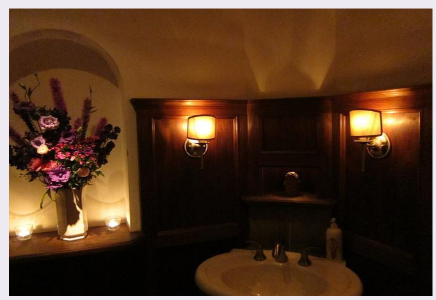
CV > Visual Question Answering

Query: Look at js4f.jpg, what is the red topping?

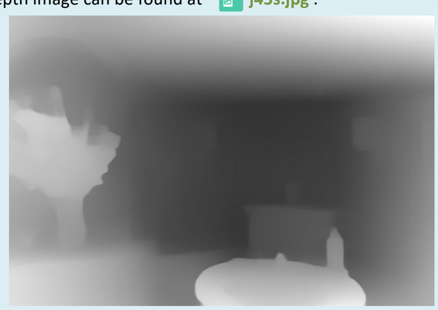
Response: Sure, **the red topping on the pizza in the image is tomato.** I used a visual question answering model dandelin/vilt-b32-finetuned-vqa to answer the question "What is the red topping?" based on the image. The answer with the highest score is "**tomato**" with a score of 0.9698325395584106.

CV > Depth Estimate

Query: please estimate the depth of ks4j.jpg



Response: I have estimated the depth of the image located at examples/ks4j.jpg using the Intel/dpt-large model. This model is specifically trained on 1.4 million images and is a Dense Prediction Transformer (DPT) model. The generated depth image can be found at ' j45s.jpg'.



CV > Text-to-Image

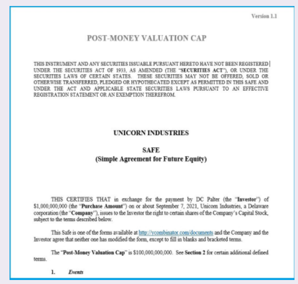
Query: please give me a photo of "wide angle view of castle, blue sky background"

Response: Sure, I can do that. For your request, I used the model runwayml/stable-diffusion-v1-5 for text-to-image task, because this model is the most suitable for the task as it is a latent text-to-image diffusion model capable of generating high-quality images from text descriptions. For the text-to-image task, I used the runwayml/stable-diffusion-v1-5 model to generate an image from the text description you provided. The generated image is located at images/asd4.jpg.

Figure 3: Case study on various tasks (a).

CV > Document Question Answering

Query: look at the document doc.jpg, what is the purchase amount?



Version 1.1

POST-MONEY VALUATION CAP

THIS INSTRUMENT AND ANY SECURITIES ISSUABLE PURSUANT HERETO HAVE NOT BEEN REGISTERED UNDER THE SECURITIES ACT OF 1933, AS AMENDED (THE "**SECURITIES ACT**"), OR UNDER THE SECURITIES LAWS OF CERTAIN STATES. THESE SECURITIES MAY NOT BE OFFERED, SOLD OR OTHERWISE TRANSFERRED, PLEDGED OR HYPOTHECATED EXCEPT AS PERMITTED IN THIS SAFE AND UNDER THE ACT AND APPLICABLE STATE SECURITIES LAWS PURSUANT TO AN EFFECTIVE REGISTRATION STATEMENT OR AN EXEMPTION THEREFROM.

UNICORN INDUSTRIES

SAFE
(Simple Agreement for Future Equity)

THIS CERTIFIES THAT in exchange for the payment by DC Palter (the "**Investor**") of $1,000,000,000 (the "**Purchase Amount**") on or about September 7, 2021, Unicorn Industries, a Delaware corporation (the "**Company**"), issues to the Investor the right to certain shares of the Company's Capital Stock, subject to the terms described below.

This Safe is one of the forms available at http://ycombinator.com/documents and the Company and the Investor agree that neither one has modified the form, except to fill in blanks and bracketed terms.

The "**Post-Money Valuation Cap**" is $100,000,000,000. See **Section 2** for certain additional defined terms.

1. *Events*

Response: Sure. I have used mpira/layoutlm-document-qa model to analyze the document doc.jpg you provided. After the analysis, **I have found that the purchase amount is $1,000,000,000**. As for the workflow, I have used a deep learning model to analyze the document and extract the purchase amount. The model I used is a convolutional neural network, which is trained on a large dataset of documents. The inference results are the purchase amount of $1,000,000,000. If you need more information, please let me know.

CV > Image to Image

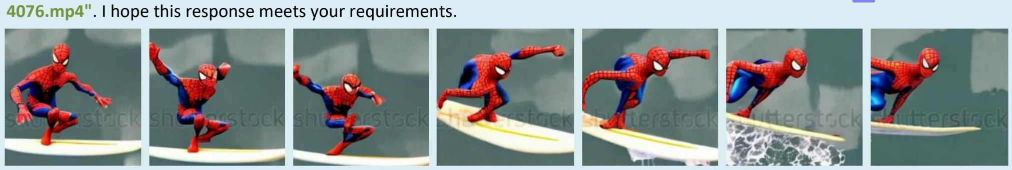
Query: give me a similar image with example.jpg

Sure. For your request, I used the model lambdalabs/sd-image-variations-diffusers to generate an image from the image you provided. The generated image is located at /images/5373.jpg. If there is nothing in the results, I will tell you that I can't make it. Is there anything else I can help you with?

Video > Generation

Query: please generate a video based on "Spiderman is surfing"

Response: Sure, I can do that. I have generated a video based on your prompt "Spiderman is surfing". The video is saved at the following path: "/videos/4076.mp4". 1. Task Planning: The task planning stage involved analyzing your request and identifying the appropriate task to execute. In this case, the task was "text-to-video". 2. Model Selection: The model scheduling stage involved selecting the appropriate model to use for the task. In this case, the only available model was damo-vilab/text-to-video-ms-1.7b. 3. Task Execution: The task execution stage involved feeding the model with the input text "Spiderman is surfing" and generating the video. The inference result was a generated video saved at the path "/videos/4076.mp4". I hope this response meets your requirements.



Audio > Generation

Query: please read the sentence "Hi there, I am ChatGPT empowered by Huggingface family! Yes, I can provide more than one thousand models for dozens of tasks. For more fun and creativity, I have invited Diffusers family to join our team. Feel free to experience it!" for me.

Response: Sure, I can do that. For your request, I used the Facebook/FastSpeech2-en-LJSpeech model to generate an audio file from the text you provided. The generated audio file is located at ' /audios/d5f3.wav '. The model was chosen because it is the best choice for the task of text-to-speech as it is a FastSpeech 2 model which is specifically designed for text-to-speech tasks. It is also the most recent model in the list and has the most potential to provide the best results. I hope this answers your question. Is there anything else I can help you with?

Figure 4: Case study on various tasks (b).

Task	Args	Candidate Models	Descriptions
		NLP Tasks	
Text-CLS	text	[*cardiffnlp/twitter-roberta-base-sentiment*, ...]	[“*This is a RoBERTa-base model trained on 58M tweets ...*”, ...]
Token-CLS	text	[*dslim/bert-base-NER*, ...]	[“*bert-base-NER is a fine-tuned BERT model that is ready to...*”, ...]
Text2text-Generation	text	[*google/flan-t5-xl*, ...]	[“*If you already know T5, FLAN-T5 is just better at everything...*”, ...]
Summarization	text	[*bart-large-cnn*, ...]	[“*BART model pre-trained on English language, and fine-tuned...*”, ...]
Translation	text	[*t5-base*, ...]	[“*With T5, we propose reframing all NLP tasks into a unified...*”, ...]
Question-Answering	text	[*deepset/roberta-base-squad2*, ...]	[“*This is the roberta-base model, fine-tuned using the SQuAD2.0...*”, ...]
Conversation	text	[*PygmalionAI/pygmalion-6b*, ...]	[“*Pymalion 6B is a proof-of-concept dialogue model based on...*”, ...]
Text-Generation	text	[*gpt2*, ...]	[“*Pretrained model on English ...*”, ...]
Tabular-CLS	text	[*matth/flowformer*, ...]	[“*Automatic detection of blast cells in ALL data using transformers....*”, ...]
		CV Tasks	
Image-to-Text	image	[*nlpconnect/vit-gpt2-image-captioning*, ...]	[“*This is an image captioning model trained by @ydshieh in flax...*”, ...]
Text-to-Image	image	[*runwayml/stable-diffusion-v1-5*, ...]	[“*Stable Diffusion is a latent text-to-image diffusion model...*”, ...]
VQA	text + image	[*dandelin/vilt-b32-finetuned-vqa*, ...]	[“*Vision-and-Language Transformer (ViLT) model fine-tuned on...*”, ...]
Segmentation	image	[*facebook/detr-resnet-50-panoptic*, ...]	[“*DEtection TRansformer (DETR) model trained end-to-end on ...*”, ...]
DQA	text + image	[*impira/layoutlm-document-qa*, ...]	[“*This is a fine-tuned version of the multi-modal LayoutLM model ...*”, ...]
Image-CLS	image	[*microsoft/resnet-50*, ...]	[“*ResNet model pre-trained on...*”, ...]
Image-to-image	image	[*radames/stable-diffusion-v1-5-img2img*, ...]	[“*Stable Diffusion is a latent text-to-image diffusion model...*”, ...]
Object-Detection	image	[*facebook/detr-resnet-50*, ...]	[“*DEtection TRansformer (DETR) model trained end-to-end on ...*”, ...]
ControlNet-SD	image	[*lllyasviel/sd-controlnet-canny*, ...]	[“*ControlNet is a neural network structure to control diffusion...*”, ...]
		Audio Tasks	
Text-to-Speech	text	[*espnet/kan-bayashi_ljspeech_vits*, ...]	[“*his model was trained by kan-bayashi using ljspeech/tts1 recipe in...*”, ...]
Audio-CLS	audio	[*TalTechNLP/voxlingua107-epaca-tdnn*, ...]	[“*This is a spoken language recognition model trained on the...*”, ...]
ASR	audio	[*jonatasgrosman/wav2vec2-large-xlsr-53-english*, ...]	[“*Fine-tuned XLSR-53 large model for speech recognition in English ...*”, ...]
Audio-to-Audio	audio	[*speechbrain/metricgan-plus-voicebank*, ...]	[“*MetricGAN-trained model for Enhancement...*”, ...]
		Video Tasks	
Text-to-Video	text	[*damo-vilab/text-to-video-ms-1.7b*, ...]	[“*his model is based on a multi-stage text-to-video generation...*”, ...]
Video-CLS	video	[*MCG-NJU/videomae-base*, ...]	[“*VideoMAE model pre-trained on Kinetics-400 for 1600 epochs...*”, ...]

Table 11: The task list used in HuggingGPT. The first and second columns are the name and arguments of the corresponding task. The third and fourth columns provide some examples of the candidate models and their model descriptions.

Query #1: Please generate a canny image based on example.jpg

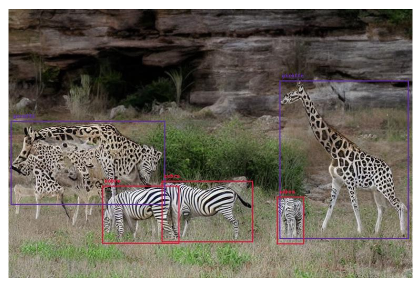
Response #1: I have generated a canny image for you based on the image located at example.jpg. The task process involved selecting the canny-control model for image processing, as it covers common image processing functions. The inference results include a generated image, which can be found at images/df8t.jpg.

Task Planning #1:
1: {"task": "image-to-text", "id": 0, dep: [-1], "args": {"image": "example.jpg"}}
2: {"task": "control-canny", "id": 0, dep: [-1], "args": {"image": "example.jpg"}}

Execution Results #1:
1. {"generated text": "a herd of giraffes and zebras grazing in a field"}
2. {"generated image": "images/df8t.jpg"}

Query #2: then based on the above canny image and a prompt "a photo of a zoo", generate a new image.

Response #2: I have received your request to generate a canny image of a photo of a zoo. I have selected the model lllyasviel/sd-controlnet-canny as it is the best model for this task. I have processed the image and generated a new image located at images/ja2f.jpg. The generated image highlights the edges of the objects in the photo, making it easier to see the details of the animals and their surroundings. Overall, the task was to convert a text prompt 'a photo of a zoo' into a canny image using the selected model.



Task Planning #2:
1: {"task": "canny-text-to-image", "id": 0, dep: [-1], "args": {"text":"a photo of a zoo", "image": "example.jpg"}}

Execution Results #2:
1. {"generated image": "images/ja2f.jpg"}

Query #3: please detect objects in the above image

Task Planning #3:
1: {'task': 'object-detection', 'id': 0, 'dep': [-1], 'args': {'image': 'images/ja2f.png'}}

Response #3:
Sure, here is the result of your command. I hava successfully detected objects in the image you provided. **The detected objects are giraffes and zebras.** I used the facebook/detr-resnet-101 model for object detection, which is specifically designed for object detection tasks and has a ResNet-101 backbone, which is a powerful feature extractor. Additionally, the DETR architecture used in this model is an end-to-end object detection model that has shown promising results in various benchmarks. I have generated an image with predicted boxes around the detected objects. You can find the generated image at the following path: images/dif2.jpg. The predicted objects and their corresponding scores are as follows: - Giraffe with a score of 0.999- Zebra with a score of 0.998- Zebra with a score of 0.998- Giraffe with a score of 0.969- Zebra with a score of 0.997. I hope this information is helpful to you.

Execution Results #3:
1. {'generated image with predicted box': '/images/dif2.jpg', 'predicted': [{'box': {'xmax': 760, 'xmin': 515, 'ymax': 437, 'ymin': 136}, 'label': 'giraffe', 'score': 0.9992127418518066}, {'box': {'xmax': 561, 'xmin': 510, 'ymax': 447, 'ymin': 357}, 'label': 'zebra', 'score': 0.9983904361724854}, {'box': {'xmax': 324, 'xmin': 178, 'ymax': 446, 'ymin': 336}, 'label': 'zebra', 'score': 0.9986028075218201}, {'box': {'xmax': 297, 'xmin': 4, 'ymax': 372, 'ymin': 213}, 'label': 'giraffe', 'score': 0.9694724082946777}]}

Figure 5: Case study on complex tasks (a).

Query: Describe this image example.jpg in as much detail as possible.

Response: The image you provided appears to show **a family of four dogs playing in the grass**. For this task, I ran multiple CV tasks, including image-to-text, image-classification, object-detection, visual-question-answering, and document-question-answering. I chose specific models for each task that were best suited to addressing your query, based on their training and performance characteristics.

For image-to-text, I chose the **nlpconnect/vit-gpt2-image-captioning** model. This model generates an image caption with the given image. I applied this model to the input image and the developed output was: **a family of four dogs are playing in the grass**
For image-classification, I selected the **google/vit-base-patch16-224** model. This model is trained on natural images dataset and it can predict the label of the image output. I applied this model to the image and get the results showing the risk of each label. It shows the highest risk at "**Rhodesian ridgeback**" with a score of 93.8%
For object-detection, I used **facebook/detr-resnet-101 model**. This model is trained on COCO panoptic segmentation for object detection. I used this model to generate an image file with the predicted box. The generated image with predicted box is at: **/images/sd48.jpg**
Regarding your request for image segmentation, I have completed the task using a pre-trained model called **facebook/detr-resnet-50-panoptic**. The segmentation mask shows the different regions of the image that belong to different classes. In this case, the model has identified two main classes in the image, which are "**dog**", "**grass**" and "**tree**". The generated image with the segmentation mask can be found at **/images/t4s1.jpg**
For visual question answering, I used **dandelin/vilt-b32-finetuned-vqa** model. This model is a Vision-and-Language Transformer (ViLT) model, which is fine-tuned on VQAv2. It is suitable for visual-question-answering tasks, and is likely to provide accurate results for the given image. For the question "what are in the image?" It reply with a prediction **"dogs"** and the confidence is 84.9%.

Task Planning:
1. {"id": "0", "dep": [-1], "task": "image-to-text", "args": {"image": "example.jpg" }},
2. {"id": "1", "dep": [-1], "task": "image-classification", "args": {"image": "example.jpg" }},
3. {"id": "2", "dep": [-1], "task": "object-detection", "args": {"image": "example.jpg" }},
4. {"id": "3", "dep": [-1], "task": "image-segmentation", "args": {"image": "example.jpg" }},
5. {"id": "4", "dep": [-1], "task": "visual-question-answering", "args": {"text": "describe this image in detail", "image": "example.jpg" }},

Execution Results:
1. {"generated_text": "a family of four dogs are playing in the grass "}
2. [{'score': 0.938717246055603, 'label': 'Rhodesian ridgeback'}, {'score': 0.01652258075773716, 'label': 'boxer'}, {'score': 0.006381669547408819, 'label': 'Great Dane'}, {'score': 0.006234415341168642, 'label': 'vizsla, Hungarian pointer'}, {'score': 0.005944834090769291, 'label': 'bull mastiff'}]
3. [{"generated image with predicted box": "/images/sd48.jpg","predicted": [{"box": {"xmax": 463, "xmin": 373, "ymax": 267, "ymin": 199}, "label": "bus", "score": 0.9981155395507812}, {"box": {"xmax": 292, "xmin": 124, "ymax": 224, "ymin": 175}, "label": "airplane", "score": 0.9983609318733215}, {"box": {"xmax": 554, "xmin": 487, "ymax": 277, "ymin": 226}, "label": "person", "score": 0.9910836219787598}]}
4. {"generated image with segmentation": "/images/t4s1.jpg", "predicted": [{"score": 0.989, "label": "grass"}, {"score": 0.999, "label": "dog"}, {"score": 0.999, "label": "tree"},{"score": 0.999, "label": "dog"}]
5. [{'answer': 'dogs', 'score': 0.8488452434539795}, {'answer': 'dog', 'score': 0.04168461635708809}]

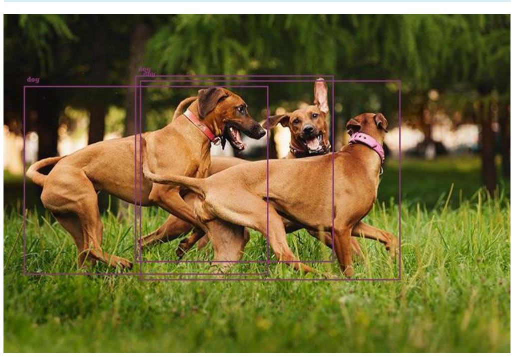
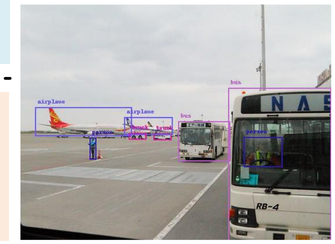


Figure 6: Case study on complex tasks (b).

Query: Can you tell me how many objects in the picture example1.jpg?



Response: There are **8 objects** in the picture. I used the model facebook/detr-resnet-101 for this task which is trained on a popular object detection dataset called COCO, and it has the highest number of downloads. It generated an image with predicted boxes that is available under the path: /images/79f2.jpg . As for image captioning, I selected the model nlpconnect/vit-gpt2-image-captioning which is more frequently used and works better in this case. The generated text from the image is '**a large jetliner sitting on top of an airport tarmac.**'

Task Planning:

1: {"task": "object-detection", "id": 0, dep: [-1], "args": {"image": "example1.jpg"}},
2: {"task": "image-to-text", "id": 1, dep: [-1], "args": {"image": "example1.jpg"}}

Execution Results:

1. [{"generated image with predicted box": "/images/79f2.jpg", "predicted": [{"box": {"xmax": 463, "xmin": 373, "ymax": 267, "ymin": 199}, "label": "bus", "score": 0.9981155395507812}, {"box": {"xmax": 292, "xmin": 124, "ymax": 224, "ymin": 175}, "label": "airplane", "score": 0.9983609318733215}, {"box": {"xmax": 554, "xmin": 487, "ymax": 277, "ymin": 226}, "label": "person", "score": 0.9910836219787598}, {"box": {"xmax": 230, "xmin": 219, "ymax": 264, "ymin": 227}, "label": "person", "score": 0.998579740524292}, {"box": {"xmax": 361, "xmin": 330, "ymax": 231, "ymin": 220}, "label": "truck", "score": 0.9066138863563538}, {"box": {"xmax": 639, "xmin": 460, "ymax": 420, "ymin": 142}, "label": "bus", "score": 0.9987366795539856}, {"box": {"xmax": 363, "xmin": 279, "ymax": 222, "ymin": 192}, "label": "airplane", "score": 0.9976897239685059}, {"box": {"xmax": 317, "xmin": 287, "ymax": 230, "ymin": 219}, "label": "truck", "score": 0.9339178800582886}]},
2. {"generated_text": "a large jetliner sitting on top of an airport tarmac "}

Query: In example2.jpg, what's the animal and what's it doing?

Response: Sure, **the animal in the image is a tiger and it's laying down.** I performed 3 tasks including image-to-text, image-classification, and visual-question-answering on the image you provided. To accomplish this, I selected models nlpconnect/vit-gpt2-image-captioning, google/vit-base-patch16-224 and dandelin/vilt-b32-finetuned-vqa because they are the perfect models for this use case respectively. According to the results, I have 88% confidence that the animal is a tiger and also have a 77.8% confidence that it's laying down .The generated image with predicted box is stored at the path: /images/d59a.jpg.

Task Planning:

1: {"task": "image-classification", "id": 0, dep: [-1], "args": {"image": "example2.jpg" }},
2: {"task": "visual-question-answering", "id": 1, dep: [-1], "args": {"text": "What is the animal doing?", "image": "example2.jpg" }}
3: {"task":"image-to-text", "id": 2, dep: [-1], "args": {"image": "example2.jpg" }}

Execution Results:

1. [{'score': 0.8858247399330139, 'label': 'tiger, Panthera tigris'}, {'score': 0.10940514504909515, 'label': 'tiger cat'}, {'score': 0.0006216467591002584, 'label': 'jaguar, panther, Panthera onca, Felis onca'}, {'score': 0.0004262701258994639, 'label': 'dhole, Cuon alpinus'}, {'score': 0.00030842673731967807, 'label': 'lion, king of beasts, Panthera leo'}]
2. [{'answer': 'sitting', 'score': 0.8294351696968079}, {'answer': 'laying down', 'score': 0.7387925386428833}, {'answer': 'resting', 'score': 0.625247061252594}, {'answer': 'laying', 'score': 0.6114301681518555}, {'answer': 'relaxing', 'score': 0.44740357995033264}]
3. {'generated_text': 'a brown and white striped zebra laying on a rock'}

Figure 7: Case study on complex tasks (c).

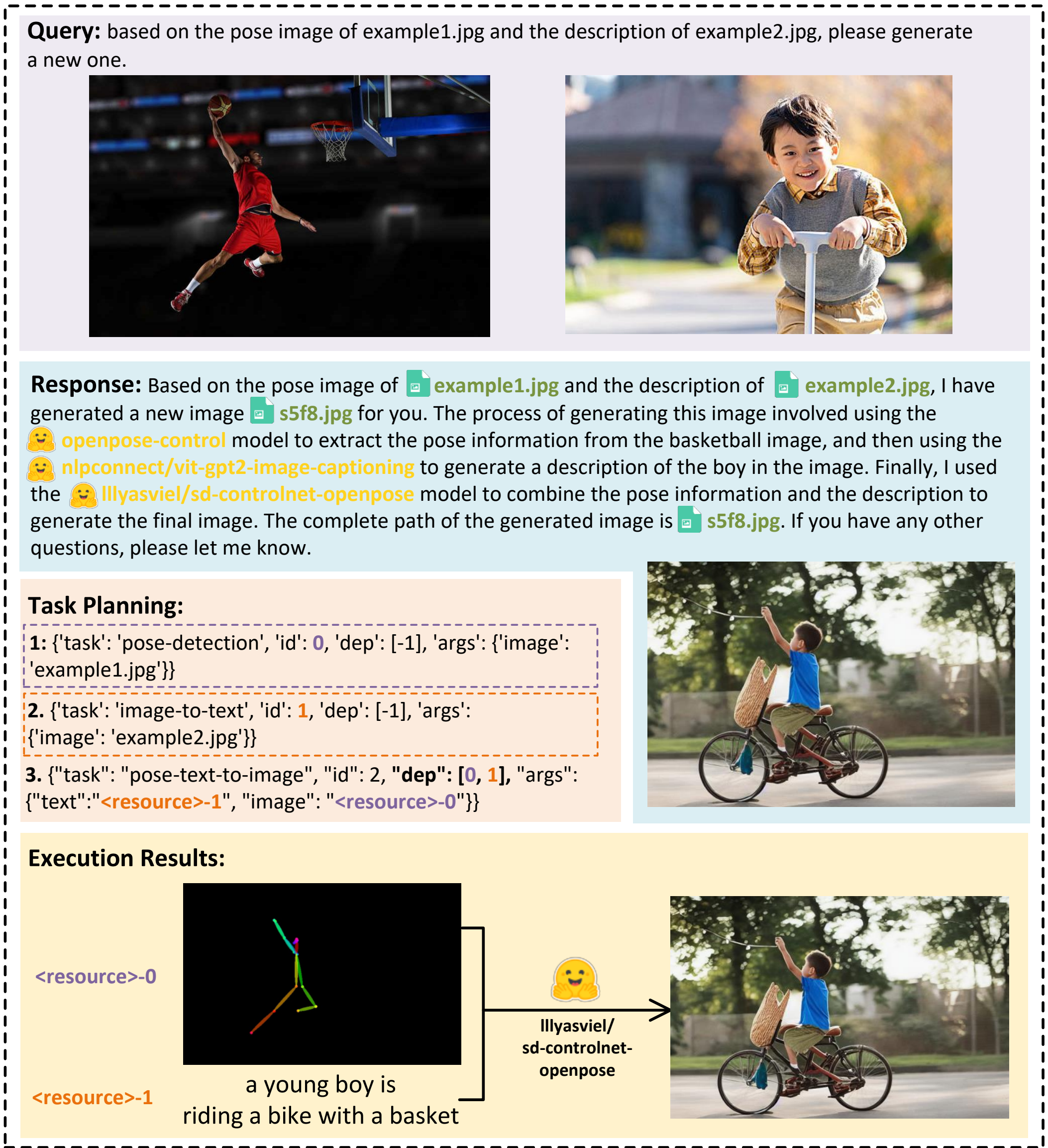
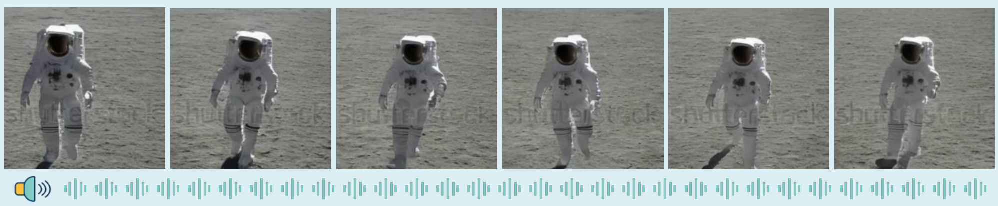


Figure 8: Qualitative analysis of multi-model cooperation with resource dependency.

Query: generate a video with the title "An astronaut is walking in space" and dub it.

Response: Sure. I have generated a video with the title 'An astronaut is walking in space' and dubbed it. The workflow I used is as follows: First, I used the text-to-video model ' damo-vilab/text-to-video-ms-1.7b' to generate a video from the text. The inference result is the generated video ' /videos/5696.mp4'. Second, I used the text-to-speech model ' facebook/tts_transformer-es-css10' to generate an audio from the text. The inference result is the generated audio ' /audios/3f9e.flac '. Finally, I combined the video and audio to generate the dubbed video. I hope this answers your question. Is there anything else I can help you with?



Task Planning:
1: {"task": "text-to-video", "id": 0, "dep": [-1], "args": {"text": "An astronaut is walking in space" }}
2: {"task": "text-to-speech", "id": 1, "dep": [-1], "args": {"text": "An astronaut is walking in space" }}

Execution Results:
1. {"generated video": "/videos/5696.mp4"} **2.** {"generated audio": "/audios/3f9e.flac"}

Query: read the image example.jpg for me.

Response: Sure. I have read the image example.jpg for you. The inference result is **'INVOICE NO : 18301/102/T0305'**. The model used for this task is microsoft/trocr-base-printed, which is best suited for the task of image-to-text as it is fine-tuned on the SROIE dataset which is specifically designed for this task. Then I have converted the text to audio, the generated audio is /audios/da5s.wav and the model used for this task is facebook/fastspeech2-en-ljspeech, which is a FastSpeech 2 text-to-speech model which is suitable for the task of text-to-speech. Is there anything else I can help you with?

Task Planning:
1: {"task": "image-to-text", "id": 0, "dep": [-1], "args": {"image": "example.jpg"}}
2: {"task": "text-to-speech", "id": 1, "dep": [0], "args": {"text": "<resource-0>"}

Execution Results:
1. {"generated text": "INVOICE NO : 18301/102/T0305"} **2.** {"generated audio": "/audios/da5s.wav"}

Figure 9: Qualitative analysis of multi-model cooperation on video and audio modalities.

Query: Given a collection of images A: /examples/a.jpg, B: /examples/b.jpg, C: /examples/c.jpg, please tell me how many zebras in these pictures?

Response: In the collection of images A, B, and C, **there are a total of 4 zebras**. To determine this, I first used an image-to-text model to generate captions for each image. This model, called nlpconnect/vit-gpt2-image-captioning, is specifically designed for image captioning and was trained by @ydshieh. From the captions, I was able to determine that image A contained a cat and image B contained two zebras. Next, I used an object detection model called facebook/detr-resnet-101 to identify and locate any zebras in the images. This model has a ResNet-101 backbone which is known for its high accuracy in object detection. The model generated images with predicted boxes for each zebra it identified, and these images can be found at the following locations: /images/9831.jpg, /images/be11.jpg. Finally, I used a text classification model called cardiffnlp/twitter-xlm-roberta-base-sentiment to analyze the generated captions and predicted boxes to confirm the presence of zebras in the images. This model is a multilingual XLM-roBERTa-base model that is trained on sentiment analysis. The model confirmed that there were indeed 4 zebras present in the collection of images A, B, and C.

Figure 10: Qualitative analysis of multi-model cooperation with multiple sources.