CGI: Identifying Conditional Generative Models with Example Images

Zhi Zhou[1,2] , **Hao-Zhe Tan**[1,3] , **Peng-Xiao Song**[1,4] and **Lan-Zhe Guo**[1,3,*]
[1]National Key Laboratory for Novel Software Technology, Nanjing University, China
[2]School of Computer Science, Nanjing University, China
[3]School of Intelligence Science and Technology, Nanjing University, China
[4]School of Artificial Intelligence, Nanjing University, China
{zhouz, tanhz, songpx, guolz}@lamda.nju.edu.cn

Abstract

Generative models have achieved remarkable performance recently, and thus model hubs have emerged. Existing model hubs typically assume basic text matching is sufficient to search for models. However, in reality, due to different abstractions and the large number of models in model hubs, it is not easy for users to review model descriptions and example images, choosing which model best meets their needs. Therefore, it is necessary to describe model functionality wisely so that future users can efficiently search for the most suitable model for their needs. Efforts to address this issue remain limited. In this paper, we propose *Conditional Generative Model Identification* (CGI), which aims to provide an effective way to identify the most suitable model using user-provided example images rather than requiring users to manually review a large number of models with example images. To address this problem, we propose the *Prompt-Based Model Identification* (PMI) , which can adequately describe model functionality and precisely match requirements with specifications. To evaluate PMI approach and promote related research, we provide a benchmark comprising 65 models and 9100 identification tasks. Extensive experimental and human evaluation results demonstrate that PMI is effective. For instance, 92% of models are correctly identified with significantly better FID scores when four example images are provided.

1 Introduction

Deep generative models [Jebara, 2012], such as variational autoencoders (VAE) [Kingma and Welling, 2014; Kingma and Welling, 2019; Parmar *et al.*, 2021], generative adversarial networks (GAN) [Goodfellow *et al.*, 2014; Sohn *et al.*, 2015; Creswell *et al.*, 2018], flow-based models [Rezende and Mohamed, 2015], and diffusion models [Sohl-Dickstein *et al.*, 2015; Dhariwal and Nichol, 2021; Rombach *et al.*, 2022], have shown significant success in image generation. Generative model hubs like Hugging Face and CivitAI have been established to facilitate the sharing and downloading of models by developers and users, respectively. However, with the rapid growth of available models, finding the most suitable model for specific tasks has become a critical challenge.

Existing generative model hubs provide basic methods such as model tag filtering, text matching, and download volume ranking [Shen *et al.*, 2023] to help users search for models. However, the complex functionalities of generative models cannot be adequately described using only textual and statistical information [Lu *et al.*, 2023; Luo *et al.*, 2024]. Although model hubs like CivitAI and OpenArt offer example images for each model to assist users in finding their desired models, this solution has two main limitations. First, example images cannot fully represent model functionality, especially when user purposes and offered examples mismatch. Second, users still need to manually review example images repeatedly, which is time-consuming and heavily relies on their expertise. Therefore, model selection remains challenging for users, as they must iteratively review, download, and test multiple models before finding a suitable one.

Conditional Generative Model Identification (CGI). The above limitation inspires us to consider the following question: *Can we describe the functionality of conditional generative models in a precise format that enables efficient and accurate model identification by matching their functionalities with user requirements?* Following the learnware paradigm [Zhou and Tan, 2022], we call the functionality description the model's specification. An important characteristic of this problem is the requirement to assign a specification to the model upon uploading it to the model hub which allows future users to simply compute the similarity between the specification and their requirements to search for models. This is fundamentally different from the scenario where a query and a large set of candidate models are provided, and the objective is to learn the ranking. We call this novel setting *Conditional Generative Model Identification* (CGI). To the best of our knowledge, this problem has not been studied yet. Figure 1 presents an illustration of the CGI problem and different from the traditional model selection process.

Challenges of CGI. The challenges of CGI come from two main sources: (1) How to assign specifications to adequately describe the functionalities of different conditional generative models as well as the user requirements, and (2) How to match the users' requirements with the models' specifica-

*Corresponding author.

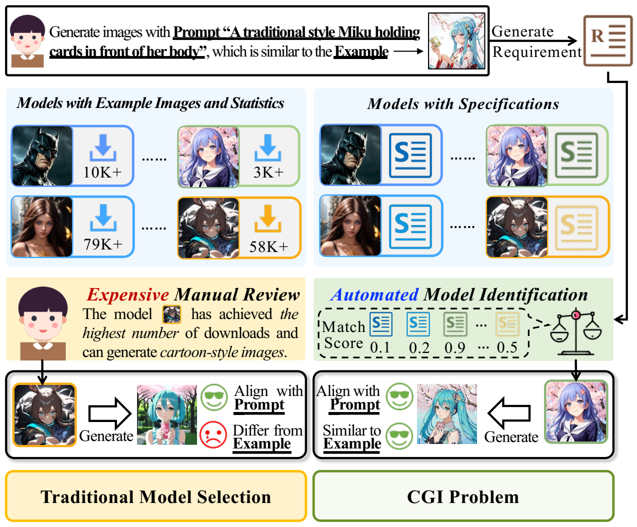
Generate images with Prompt "A traditional style Miku holding cards in front of her body", which is similar to the Example
Generate Requirement
Models with Example Images and Statistics
Models with Specifications
10K+
3K+
79K+
58K+
Expensive Manual Review
The model has achieved the highest number of downloads and can generate cartoon-style images.
Automated Model Identification
Match Score 0.1 0.2 0.9 0.5
Generate
Align with Prompt
Differ from Example
Similar to Example
Traditional Model Selection
CGI Problem

Figure 1: Comparison between traditional model selection and CGI problem setting. With the design of the specification and model identification, the most suitable model can be efficiently identified by matching the user's requirements with the model's specification.

tions in the same space. There are few works related to this problem. For instance, HuggingGPT [Shen *et al.*, 2023] and Stylus [Luo *et al.*, 2024] proposed to describe the model's functionality using natural language and search for models using ChatGPT. However, only natural language is insufficient to describe the specialty of the model since it is not easy to match the model's specific functionality with the text description. [Wu *et al.*, 2023] proposed to describe the functionality of discriminative models by approximating the training data distribution. However, for conditional generative models, the functionality is not only related to the generated data distribution but also to the prompts. There is no existing work that can be directly applied to solve CGI problems, requiring to study more wise descriptions for generative models.

Our Solution. To this end, we present a novel systematic solution, namely, Prompt-Based Model Indetification (PMI). Specifically, we first introduce *Automatic Specification Assignment* to generate specifications using a pre-defined prompt set or developer-provided prompt set. Then, *Requirement Generation* abstracts the user requirements. Both specification and requirement are projected into a unified model matching space for future model identification. Finally, we propose a *Task-Specific Matching* mechanism to adjust specification according to the user requirements in the model matching space to precisely identify the most suitable model. The basic idea is that if the model produces a data distribution that is very close to the user's example images with the similar prompts, then there is a high probability that the model is useful. To evaluate the effectiveness of PMI and promote the related research, we developed a benchmark comprising 65 conditional generative models and 9100 model identification tasks. Extensive experiment results demonstrate that PMI is effective. Moreover, human and GPT evaluation results confirm both the validity of our evaluation protocol and the superior performance of PMI. Our main contributions can be summarized as follows:

(a) We introduce a novel setting called CGI for identifying conditional generative models that match user requirements. This problem setting consists of two key challenges: (1) the construction of model specifications and user requirements and (2) the matching of user requirements to model specifications.

(b) We analyze the challenges inherent in the CGI problem and propose an effective solution PMI. Our approach can adequately describe the functionality of generative models and enables future users to efficiently find the most suitable model with just a few example images.

(c) We develop a benchmark with 65 models and 9100 identification tasks to evaluate model identification approaches. Extensive experiments and human evaluation results demonstrate that our proposal can achieve satisfactory model identification performance.

2 Related Work

This study is related to the following two aspects:

Learnware and Model Selection. With the development of model hubs, users have the option to search for and adapt pre-trained models that satisfy their specific needs. A similar model selection approach can also be applied to enhance generalization [Zhou *et al.*, 2024; Zhou *et al.*, 2025]. Learnware [Zhou, 2016] offers a paradigm to identify models for the users. Recently, Wu [Wu *et al.*, 2023] proposed to describe the model's functionality by approximating the training data distribution and searching for models by comparing the approximated distribution distance. Guo [Guo *et al.*, 2023] proposed to describe the model's functionality by approximating the model's parameters with a linear proxy model and enabling the model search by comparing the proxy model's parameters. However, these methods are not applicable to the generative model search. A few works are related to the generative model search. For example, [Shen *et al.*, 2023] proposed to describe the model's functionality via natural language (e.g., model tags, model architectures, resources requirement) and adopted ChatGPT as a model selector to search useful models that meet user's requirements from the HuggingFace platform. Stylus [Luo *et al.*, 2024] describes the model functionality with vision-language models, and identify conditional generative models with large language models. However, only natural language can not describe the model's functionality adequately, more precise description needs to be studied. [Lu *et al.*, 2023] proposed a content-based search method that can be applied to unconditional generative models. Therefore, existing works are not applicable to the CGI problem, and this paper presents the first attempt to solve this problem.

Generative Models. In recent years, generative models have become one of the most widely discussed topics in the field of artificial intelligence for their promising results in image generation, exemplified by models such as Generative Adversarial Networks [Goodfellow *et al.*, 2014; Arjovsky *et al.*, 2017; Brock *et al.*, 2019; Choi *et al.*, 2020], Variational Autoencoders [Kingma and Welling, 2014; van den

Oord *et al.*, 2017; Vahdat and Kautz, 2020], Diffusion Models [Nichol and Dhariwal, 2021; Dhariwal and Nichol, 2021; Rombach *et al.*, 2022], etc. With the development of the generative model, various generative model hubs, e.g., HuggingFace and Civitai, have been developed to enable model developers to share models. These numerous generative models show different specialties and functionality. Our goal is not to introduce a new model. Instead, we want to study a new mechanism that can well organize the developed models and enable future users to efficiently find the most suitable one.

3 Preliminary

In this section, we first introduce the problem setup of CGI problem. Then, we present the problem analysis to show the core challenge of CGI problem.

3.1 Problem Setup

Assume the model hub has M conditional generative models $\{f_m\}_{m=1}^M$. Each model is associated with a corresponding specification S_m to describe its functionalities for future model identification. There are two stages in the CGI setting: the *submitting stage* for model developers and the *identification stage* for future users.

Submitting Stage. The model developer submits a model f_m to the model hub, and then we assign a specification S_m to the model. Formally, the specification S_m is generated by a specification assignment algorithm $\mathcal{A}_s$ using the model f_m, i.e., $S_m = \mathcal{A}_s(f_m)$. It is important to note that uploaded models are anonymous with no mandatory constraints, which means we cannot access their training data and developers are not guaranteed to provide required model information.

Identification Stage. For any user task τ, models are identified from the model hub using one or a few example images $X_\tau = \{x_i^\tau\}_{i=1}^{N_\tau}$. When users upload example images to describe their needs, the model hub generates the requirement represented in the specification space $R_\tau = \mathcal{A}_r(X_\tau)$ using a requirement generation algorithm $\mathcal{A}_r$. Then, we match the requirement R_τ with model specifications $\{S_m\}_{m=1}^M$ using an evaluation algorithm $\mathcal{A}_e$ and compute the matching score $\widehat{s}_m^\tau = \mathcal{A}_e(S_m, R_\tau)$ for each model f_m. Finally, return the best-matched model with the maximum score or a list of models sorted by $\{\widehat{s}_m^\tau\}_{m=1}^M$ in descending order.

The two main problems for addressing CGI are: (a) How to design $\mathcal{A}_s$ and $\mathcal{A}_r$ to fully characterize the functionality of submitted conditional generative models and user requirements? (b) How to design $\mathcal{A}_e$ to effectively identify the most suitable model for users' specific needs using the specifications and requirements?

3.2 Problem Analysis

Learnware [Zhou, 2016] provides an effective framework for describing the functionality of discriminative models. It describes model functionality by approximating the training data distribution and performs model search by matching the approximated training and testing data distributions.

Specifically, the learnware methods [Wu *et al.*, 2023] use Kernel Mean Embedding (KME) techniques to represent training data distributions by mapping a probability distribution $\mathbb{P}$ defined on $\mathcal{X}$ into a reproducing kernel Hilbert space (RKHS) as

$$u_k(\mathbb{P}) := \int_{\mathcal{X}} k(x,\cdot)d\mathbb{P}(x) \tag{1}$$

where $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel function with associated RKHS $\mathcal{H}$. In cases of finite training data, the empirical KME can be used to approximate the true KME using the dataset $X = \{x_i\}_{i=1}^N$ which can be seen as data points sampled from the distribution $\mathbb{P}$:

$$\hat{u}_k(\mathbb{P}) := \frac{1}{N}\sum_{i=1}^{N} k(x_i,\cdot) \tag{2}$$

It has been proved that the empirical KME can converge to the true KME at a rate $\mathcal{O}(1/\sqrt{N})$ [Smola *et al.*, 2007].

However, existing learnware methods cannot be directly used for the CGI problem since they were designed for discriminative models rather than conditional generative models. Corresponding to the two main problems mentioned above, the two main challenges in extending learnware methods to the CGI problem are:

(a) How to project both the complex functionality of conditional generative models and the diverse styles of user example images into a unified space for future model identification?

(b) How to design an effective matching mechanism between user requirements and model functionality corresponding to specific user tasks?

4 Our Approach

In this section, we present our solution *Prompt-Based Model Indetification* (PMI) for the CGI setting. As illustrated in Figure 2, PMI consists of three key modules. *Automatic Specification Assignment* and *Requirement Generation* project the model's functionality and user requirements into a unified model matching space, addressing the first challenge. *Task-Specific Matching* adjusts the specification in the matching space according to the requirement and identifies the most suitable model with the highest similarity score, addressing the second challenge. We first describe the three key modules in detail. Then, a further analysis is provided as follows.

4.1 Automatic Specification Assignment

Automatic specification assignment aims to automatically generate a specification for each conditional generative model to describe its functionality. Its core idea is to prompt the conditional generative model f_i to generate data for describing the model functionality based on a pre-defined prompt set $\mathcal{P}$ or developer-provided prompt set $\mathcal{P}_i$.

When model developer submit a model f_i to the model hub, the automatic specification assignment algorithm $\mathcal{A}_s$ generates its corresponding specification S_i using a N-size prompt set $\mathcal{P}$ which is pre-defined in the model hub. A set of images X_i is generated by the model f_i and the prompt set $\mathcal{P}$ to describe model functionality conditioned on $\mathcal{P}$:

$$X_i = \left\{x_j^i = f(p_j) | p_j \in \mathcal{P}\right\}_{j=1}^N . \tag{3}$$

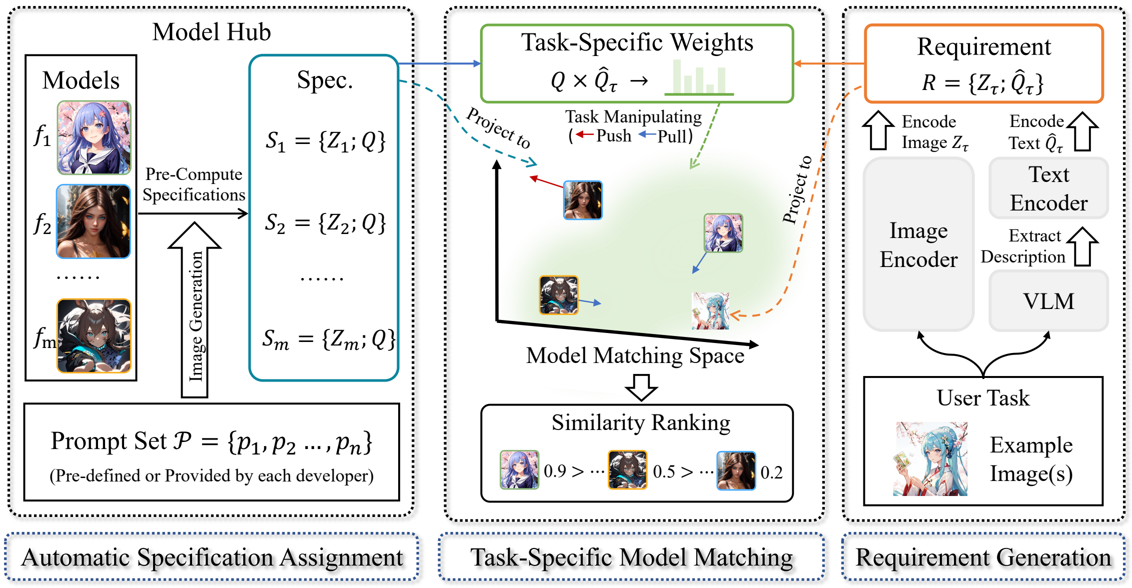

Figure 2: Our proposed PMI method includes three main components: *Automatic Specification Assignment*, *Requirement Generation*, and *Task-Specific Matching*. *Automatic Specification Assignment* generates a specification for each model using either a predefined or developer-provided prompt set to describe model functionality within the model matching space. *Requirement Generation* formulates the requirement for each user task by encoding example images and their textual descriptions into the same space. *Task-Specific Matching* adjusts the specification in the matching space according to the requirement and identifies the most suitable model with the highest similarity score.

Then, the prompt set $\mathcal{P}$ and the generated images X_i are encoded to a unified model matching space using textual encoder $\mathcal{T}(\cdot)$ and vision encoder $\mathcal{G}(\cdot)$ from a pre-trained vision-language model respectively:

$$\begin{cases} Z_i = \{z_j^i = \mathcal{G}(x_j^i) | x_j^i \in X_i\}, \\ Q_i = \{q_j^i = \mathcal{T}(p_j) | p_j \in \mathcal{P}\}. \end{cases} \tag{4}$$

Finally, the specification S_i is defined as

$$S_i = \mathcal{A}_s(f_i, \mathcal{P}) = \{Z_i, Q_i\}. \tag{5}$$

Note that if developer can provide a prompt set $\mathcal{P}_i$ for model f_i to better describe the model functionality, the better specification S_i can be computed by replacing $\mathcal{P}$ with $\mathcal{P}_i$ in Equation 3 and Equation 4.

The advantages of our proposed automatic specification assignment are two-fold: (1) The specification S_i can be automatically computed within the model hub, providing great convenience for developers and reducing their burden of uploading models. (2) The specification does not require a significant amount of storage space on the model hub, as it only involves storing the feature representation.

4.2 Requirement Generation

Requirement generation aims to produce the requirements R_τ for the user task τ to select the most suitable model. Its core idea is to decompose the complex model functionality into the difference between user-provided example images $X_\tau = \{x_i^\tau\}_{i=1}^{N_\tau}$ and corresponding textual descriptions, enabling the future model identification to be more accurate.

Specifically, requirement generation algorithm $\mathcal{A}_r$ transforms X_τ into feature representations:

$$Z_\tau = \{z_i^\tau = \mathcal{G}(x_i^\tau)\}_{i=1}^{N_\tau} \tag{6}$$

using the vision encoder $\mathcal{G}(\cdot)$. Then, the textual description $\widehat{\mathcal{P}}_\tau$ is generated by a vision-language model $\mathrm{VLM}(\cdot)$ to describe each example image and be mapped to the model matching space:

$$\begin{cases} \widehat{\mathcal{P}}_\tau = \{\hat{p}_i^\tau = \mathrm{VLM}(x_i^\tau)\}_{i=1}^{N_\tau} \\ \widehat{Q}_\tau = \{\hat{q}_i^\tau = \mathcal{T}(\hat{p}_i^\tau)\}_{i=1}^{N_\tau} \end{cases} \tag{7}$$

Finally, the user requirement R_τ is computed using $\mathcal{A}_r$ as

$$R_\tau = \mathcal{A}_r(X_\tau) = \left\{Z_\tau; \widehat{Q}_\tau\right\}. \tag{8}$$

Note that R_τ is also automatically computed within the model hub, which is very flexible and easy to use.

4.3 Task-Specific Matching

Task-specific matching aims to identify the most suitable model for the user task τ by calculating the similarity score between the user requirement R_τ and the specification S_m of each model f_m. Its core idea is to transform the specification S_m corresponding to the user requirement R_τ in the model matching space to better match the functionality.

Specifically, matching algorithm $\mathcal{A}_e$ calculates the similarity score between the user requirement $R_\tau = \{Z_\tau, \widehat{Q}_\tau\}$ and the specification $S_m = \{Z_m, Q_m\}$ for each model f_m using the following formula:

$$\mathcal{A}_e(S_m, R_\tau) = \frac{1}{N_\tau}\sum_{i=1}^{N_\tau}\left\| \frac{1}{N_m}\sum_{j=1}^{N_m} \frac{q_j^m \hat{q}_i^\tau}{\|q_j^m\|\|\hat{q}_i^\tau\|} k(z_j^m, \cdot) - k(z_i^\tau, \cdot) \right\|_{\mathcal{H}_k}^2 \tag{9}$$

where $\frac{q_j^m \hat{q}_i^\tau}{\|q_j^m\|\|\hat{q}_i^\tau\|}$ measures the similarity between the j-th specification prompts for model f_m and the textual descriptions of example image x_i^τ, transforming organial specifications to task-specific specifications, which are more focused on the user tasks. Then, the matching between task-specific specifications and user requirement can more accurately identify the most suitable model which meets the required model functionality. Finally, the similarity score obtained by Equation 9 can be used to sort the models or directly return the most suitable model.

4.4 Discussion

It is evident that our proposal for the CGI scenario achieves a higher level of accuracy and efficiency when compared to model search techniques employed by existing model hubs.

Accuracy. Our proposal elucidates the functionalities of generated models by capturing both the distribution of generated images and prompts. This approach allows for more accurate identification of suitable models for users, as opposed to the traditional model search method that relies on download counts and star ratings for ranking models.

Efficiency. Suppose that the model hub generates one requirement in T_r time and calculates the similarity score for each model in T_s time. The time complexity of our proposal for one identification is $O(T_r + MT_s)$ time. Moreover, with accurate identification results, users can save the efforts of browsing and selecting models, as well as reducing the consumption of network and computing. This is linearly correlated to the number of models on the model hub (which can be reduced by filtering by tags). Additionally, our approach also has the potential to achieve further acceleration through the use of a vector database [Guo *et al.*, 2023] such as Faiss [Johnson *et al.*, 2019].

5 Experiments

To verify the effectiveness of our proposed method PMI for CGI problem, we first build a novel conditional generative model identification benchmark based on stable diffusion models [Rombach *et al.*, 2022], and then conduct experiments on this benchmark. Below, we first introduce the details of the benchmark and evaluation metrics. Then, we present the experimental results on this benchmark and human evaluation results to emphasize the importance of CGI problem as well as the effectiveness of our PMI.

5.1 CGI Benchmark

In this section, we describe the our constructed CGI benchmark and corresponding evaluation metrics.

Model Hub and Task Construction. In practice, we expect model developers to submit their models to the model hub, allowing users to identify models that meet their specific needs. To enhance the realism of the evaluation, we constructed a model hub and user identification tasks to simulate this scenario. For the model hub construction, we manually collected $M = 65$ different stable diffusion models $\{f_1, \dots, f_m, \dots, f_M\}$ from CivitAI, representing uploaded conditional generative models on the hub. These models belong to the same category to mimic the real process where users first apply category filters before selecting models. For model specification generation, we created 61 prompts $\{p_1, \dots, p_{61}\}$ as a pre-defined prompt set $\mathcal{P}$ in the model hub to simulate cases where developers do not provide specialized prompts. Additionally, we constructed 61 prompts $\{p_1, \dots, p_{61}\}_m$ for each model f_m based on the prompts provided by developers in CivitAI to simulate developer-provided prompts. For model identification task construction, we created 14 evaluation prompts $\{p_{\tau_1}, \dots, p_{\tau_{14}}\}_m$ for each model on the model hub to generate testing images with random seeds in $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, forming $M_\tau = 14 \times 65 \times 10 = 9100$ different identification tasks $\{(x_{\tau_i}, t_i)\}_{i=1}^{M_\tau}$, where each example image x_{τ_i} is generated by model f_{t_i} and its best matching model index is t_i. We also provide ground-truth prompts for generating example images to assess the identification performance of each method in terms of the quality of generated images. We ensure that there is no overlap between prompts used for constructing specifications, guaranteeing the correctness of the evaluation.

Evaluation Metrics. In our experiments, we use Top-k accuracy, average rank, and FID score to evaluate the performance of methods. We define the rank of model f_m for task τ as $\widehat{r}_m^\tau = 1 + \sum_{i=1}^M \mathbb{I}\left[\widehat{s}_i^\tau < \widehat{s}_m^\tau\right]$. Then, the Top-k accuracy is defined as $\frac{1}{N^\tau}\mathbb{I}\left[\widehat{r}_{t_i}^{\tau_i} \le k\right]$, which evaluates the ability of each method to find the best matching model within k trials. The average rank is defined as $\frac{\widehat{r}_{t_i}^\tau}{N^\tau}$, which evaluates the ability of each method to rank the best matching model among other models. Moreover, we use the identified model to generate images using the ground-truth prompt for generating example images for each task. The FID score measures the distance between distribution of M_τ query images and distribution of M_τ generated images, evaluating the identification performance in aspects of generation quality.

5.2 Experimental Settings

In this section, we introduce the experimental settings, including comparison methods and implementation details.

Comparison Methods. First, we compare our proposal with baseline method, which always uses the model with the highest downloading volume as the best-matched model [Shen *et al.*, 2023]. The performance of baseline can be identified by a reasonable lower bound of CGI problem. Then, we also consider the basic implementation of the RKME specification [Wu *et al.*, 2023] as a comparison method, namely, RKME, for the CGI problem to evaluate whether learware techniques can be applied to the CGI.

Implementation Details We adopt the official code in [Wu *et al.*, 2023] to implement the RKME method and the official code in [Radford *et al.*, 2021] to implement the pre-trained vision-language model. We follow the default hyperparameter setting of RKME in previous studies [Guo *et al.*, 2023], setting the size of the reduced set to 1 and choosing the RBF kernel [Xu *et al.*, 1994] for RKHS. The hyperparameter γ for calculating RBF kernel and similarity score is tuned from $\{0.005, 0.006, 0.007, 0.008, 0.009, 0.01, 0.02, 0.03, 0.04, 0.05\}$ and set to 0.02. For all experiments without additional notes, we assume that the specification is generated with developer-provided prompts. Our experiments are conducted on Linux

Methods	Acc.(↑)	Top-2 Acc.(↑)	Top-3 Acc.(↑)	Top-4 Acc.(↑)	Top-5 Acc.(↑)	Avg. Rank(↓)	FID Score(↓)
Baseline	1.5%	3.0%	4.6%	6.1%	7.6%	33.000	23.44
RKME	3.1%	4.6%	6.2%	7.7%	9.3%	32.014	25.47
PMI	**69.2%**	**78.1%**	**82.8%**	**85.8%**	**88.0%**	**2.874**	**18.42**

Table 1: Model Identification Performance of each method evaluated by Top-k accuracy, average rank and the generation quality evaluted by FID score when only one example image is provided. The results show that our PMI can achieve satisfactory model identification performance as well as generation quality. The best performance is in **bold**.

	Accuracy(↑)			FID Score(↓)		
	Baseline	RKME	PMI	Baseline	RKME	PMI
1 image	1.5%	3.1%	**69.2%**	23.44	25.47	**18.42**
2 images	1.5%	3.1%	**84.4%**	23.44	25.53	**18.17**
3 images	1.5%	3.1%	**89.7%**	23.44	25.53	**18.14**
4 images	1.5%	3.2%	**92.7%**	23.44	25.58	**18.21**
5 images	1.5%	3.2%	**94.0%**	23.44	25.61	**18.18**
6 images	1.5%	3.2%	**95.9%**	23.44	25.53	**18.12**

Table 2: Comparison of accuracy and FID score across methods with varying numbers of example images. The results demonstrate that PMI performance improves with additional examples, while RKME shows minimal change. The best performance is in **bold**.

servers with NVIDIA A800 GPUs.

5.3 Experimental Results

In this section, we present the experimental results of our PMI method and comparison methods.

Model Identification Performance. We evaluate the model identification performance of each method in Table 1 when only one example image is provided. The Top-k accuracy and average rank metrics quantify how well each method identifies the optimal model. The FID score measures the quality of images generated by the identified models using ground-truth prompts. Results show that RKME performs similarly to the baseline method, with poor accuracy and rank, indicating the inherent difficulty of the CGI problem and the limitations of existing techniques designed for discriminative models for CGI problem. Our PMI significantly outperforms RKME in both accuracy and rank metrics, leading to two key findings: (1) the CGI problem can be effectively solved with appropriate model identification strategies; (2) PMI provides satisfactory performance even with one single example image. The FID scores further demonstrate that models identified by PMI generate higher quality images, confirming that one fixed popular model cannot meet all use cases and highlighting the significance of the CGI problem. Table 2 presents results with multiple example images. The performance of PMI improves with additional examples, while RKME's performance remains largely unchanged, validating the effectiveness of our approach.

Ablation Study. To analyze the contribution of each component in PMI, we conduct an ablation study as shown in Table 3. Our PMI method extends the RKME framework with two groups of core components: (1) Model Matching Space (MMS), which consists of the automatic specification assignment algorithm $\mathcal{A}_s$ and the requirement generation algorithm $\mathcal{A}_r$. These algorithms project both user requirements and model functionality into a unified matching space. (2) Task-Specific Matching (TSM), which introduces algorithm $\mathcal{A}_t$ to perform precise model functionality matching. The results demonstrate that integrating both components is essential for achieving optimal performance.

MMS	TSM	FID (↓)	Accuracy (↑)	Rank (↓)
		25.47	3.1%	32.014
✓		18.43	68.0%	3.120
✓	✓	**18.42**	**69.2%**	**2.874**

Table 3: Ablation study. MMS indicates the model matching space, which includes the automatic specification assignment and the requirement generation. TSM indicates the task-specific matching. The best performance is in **bold**.

Human Evaluation and GPT-4o Evaluation. To further validate the effectiveness of our PMI, we conducted human evaluation with 70 users and a GPT-4o evaluation, respectively For human evaluation, each user completed a survey containing 5 questions and each question is randomly sampled from 9100 tasks (we skip tasks where all methods identify the same model) using images generated by Baseline, RKME, PMI methods as the options. The users are required to select the image that best matches the example image. For GPT-4o evaluation, we prompt GPT-4o to choose the best image from the options for all 9100 tasks. The Figure 3 presents the average win rate of each method voted by human users and GPT-4o, respectively. The results show that human users and GPT-4o have different preferences for the images generated by the Baseline method and RKME method, giving different yet similar win rates for these two methods. Our PMI achieves the highest win rate with a large margin, indicating that the images generated by our PMI are more consistent with the user requirements.

5.4 Further Analysis

In this section, we analyze the model identification performance when using default prompts and present qualitative results through visualization.

Default Prompt Set. When model developers do not provide prompts, the model hub uses a default prompt set to generate specifications, making it more adaptable to different conditional generative models. Table 4 shows the FID scores when using default prompts for all models. Our PMI achieves significantly lower FID scores compared to RKME and baseline methods, demonstrating its ability to generate

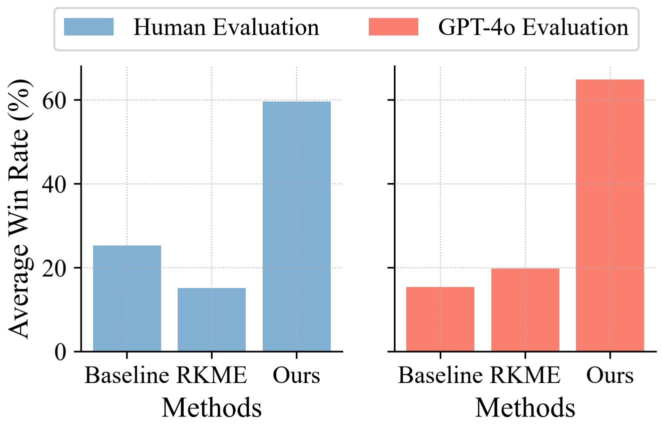

Figure 3: Human evaluation results. The results show that human users significantly prefer the images generated by our proposal.

Methods	1 image	2 images	3 images	4 images
Baseline	23.44	23.44	23.44	23.44
RKME	38.24	38.31	38.31	38.34
PMI	**20.05**	**19.94**	**19.94**	**20.13**

Table 4: FID score of each method when all models use default prompts set to generate specifications when different number of images are provided. The best performance is in **bold**.

high-quality images even with pre-defined prompts. Table 5 presents the Top-k accuracy with one example image. While PMI maintains superior performance over RKME, its accuracy is lower compared to using developer-provided prompts in Table 1, indicating that developer-provided prompts are valuable for optimal model identification.

Visualization. We visualize the generated images from models identified by Baseline, RKME, and PMI in Figure 4, with example images shown in the first column. While all identified models generate images with correct content, they differ significantly in style. Our PMI successfully identifies models that match the comic art style of the example images, whereas models identified by other methods generate images that are overly realistic. These results show that our PMI is helpful for generating images similar to the example images, which is consistent with the experimental results.

6 Conclusion

In this paper, we study a novel problem setting called *Conditional Generative Model Identification*, whose objective is to describe the functionalities of conditional generative models and enable the model to be accurately and efficiently identified for future users. To this end, we present a systematic solution including three key components. The *Automatic Specification Assignment* and *Requirement Generation* respectively project the model functionality and user requirements into a unified matching space. The *Task-Specific Matching* further builds the task-specific specification in the matching space to precisely identify the most suitable model. To promote relevant research, we open-sourced a benchmark based on stable diffusion models with 65 conditional generative models and 9100 model identification tasks. Extensive experiment results on the benchmark as well as the human evaluation demonstrate the important value of the CGI problem and the effectiveness of our proposal.

In future work, we intend to develop a novel generative model hub using the techniques presented in this paper. Our goal is to offer a more accurate description of conditional generative model functionalities and user requirements. We expect that this will enhance the efficiency of users in finding models that meet their specific needs and contribute to the development and widespread use of generative models, as well as promote the development of model hubs.

One limitation of our work is that we only consider the cases that identify generative models using uploaded example images to describe users' requirements. The assumption is reasonable since users' ideas often rely on existing image templates when they want to generate images, and it is not difficult to find images that have a similar style to fulfill the user's requirements. Despite this, it is also interesting to study how to quickly and accurately identify models via other information such as textual prompts.

Methods	Acc. (↑)	Top-2 Acc. (↑)	Top-3 Acc. (↑)	Top-4 Acc. (↑)
Baseline	1.5%	3.1%	4.6%	6.2%
RKME	3.1%	4.6%	6.2%	7.7%
PMI	**27.9%**	**38.5%**	**45.3%**	**50.8%**

Table 5: Top-k accuracy of Baseline, RKME, and PMI methods when all models use default prompts set to generate specifications. The best performance is in **bold**.

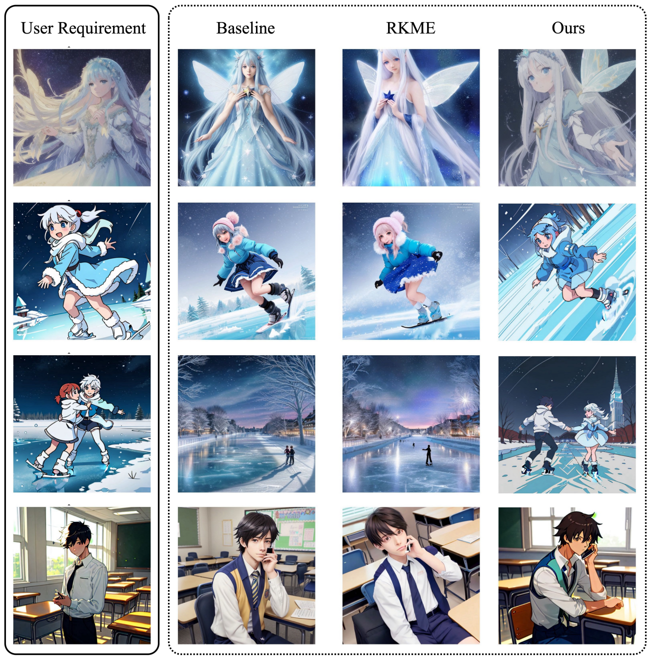

Figure 4: Visualization of generated images from models identified by Baseline, RKME, and PMI. The results demonstrate that PMI identifies models that can generate images with styles consistent with the example images.

Acknowledgements

This research was supported by the National Natural Science Foundation of China (Grant No. 624B2068, 62306133, 62250069), the Key Program of Jiangsu Science Foundation (BK20243012), the Jiangsu Science Foundation (BG2024036), and the Fundamental Research Funds for the Central Universities (022114380023).

Contribution Statement

Zhi Zhou and Hao-Zhe Tan contributed equally to this work.

References

[Arjovsky *et al.*, 2017] Mart´ın Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 214–223, 2017.

[Brock *et al.*, 2019] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *Proceedings of the 7th International Conference on Learning Representations*, 2019.

[Choi *et al.*, 2020] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8185–8194, 2020.

[Creswell *et al.*, 2018] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

[Dhariwal and Nichol, 2021] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, pages 8780–8794, 2021.

[Goodfellow *et al.*, 2014] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[Guo *et al.*, 2023] Lan-Zhe Guo, Zhi Zhou, Yu-Feng Li, and Zhi-Hua Zhou. Identifying useful learnwares for heterogeneous label spaces. In *Proceedings of the 40th International Conference on Machine Learning*, pages 12122–12131, 2023.

[Jebara, 2012] Tony Jebara. *Machine Learning: Discriminative and Generative*. Springer Science & Business Media, 2012.

[Johnson *et al.*, 2019] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[Kingma and Welling, 2014] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations*, 2014.

[Kingma and Welling, 2019] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12(4):307–392, 2019.

[Lu *et al.*, 2023] Daohan Lu, Sheng-Yu Wang, Nupur Kumari, Rohan Agarwal, Mia Tang, David Bau, and Jun-Yan Zhu. Content-based search for deep generative models. In *SIGGRAPH Asia 2023 Conference Papers*, pages 71:1–71:12, 2023.

[Luo *et al.*, 2024] Michael Luo, Justin Wong, Brandon Trabucco, Yanping Huang, Joseph E. Gonzalez, Zhifeng Chen, Ruslan Salakhutdinov, and Ion Stoica. Stylus: Automatic adapter selection for diffusion models, 2024.

[Nichol and Dhariwal, 2021] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8162–8171, 2021.

[Parmar *et al.*, 2021] Gaurav Parmar, Dacheng Li, Kwonjoon Lee, and Zhuowen Tu. Dual contradistinctive generative autoencoder. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 823–832, 2021.

[Radford *et al.*, 2021] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8748–8763, 2021.

[Rezende and Mohamed, 2015] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1530–1538, 2015.

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

[Shen *et al.*, 2023] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving AI tasks with ChatGPT and its friends in HuggingFace. *CoRR*, abs/2303.17580, 2023.

[Smola *et al.*, 2007] Alexander J. Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A hilbert space embedding for distributions. In *Proceedings of the 18th International Conference on Algorithmic Learning Theory*, pages 13–31, 2007.

[Sohl-Dickstein *et al.*, 2015] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2256–2265, 2015.

[Sohn *et al.*, 2015] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3483–3491, 2015.

[Vahdat and Kautz, 2020] Arash Vahdat and Jan Kautz. NVAE: A deep hierarchical variational autoencoder. In *Advances in Neural Information Processing Systems*, pages 19667–19679, 2020.

[van den Oord *et al.*, 2017] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315, 2017.

[Wu *et al.*, 2023] Xi-Zhu Wu, Wenkai Xu, Song Liu, and Zhi-Hua Zhou. Model reuse with reduced kernel mean embedding specification. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):699–710, 2023.

[Xu *et al.*, 1994] Lei Xu, Adam Krzyzak, and Alan L. Yuille. On radial basis function nets and kernel regression: Statistical consistency, convergence rates, and receptive field size. *Neural Networks*, 7(4):609–628, 1994.

[Zhou and Tan, 2022] Zhi-Hua Zhou and Zhi-Hao Tan. Learnware: Small models do big. *CoRR*, abs/2210.03647, 2022.

[Zhou *et al.*, 2024] Zhi Zhou, Ming Yang, Jiang-Xin Shi, Lan-Zhe Guo, and Yufeng Li. Decoop: Robust prompt tuning with out-of-distribution detection. In *Proceedings of the 41st International Conference on Machine Learning*, pages 62161–62177, 2024.

[Zhou *et al.*, 2025] Zhi Zhou, Kun-Yang Yu, Lan-Zhe Guo, and Yu-Feng Li. Fully test-time adaptation for tabular data. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence*, pages 23027–23035, 2025.

[Zhou, 2016] Zhi-Hua Zhou. Learnware: on the future of machine learning. *Frontiers of Computer Science*, 10(4):589–590, 2016.