A. Supplementary Materials

A.1. Implementation Details of Datasets Configuration

A.1.1. Copyrighted Character List

We adopt the same set of 50 copyrighted characters curated in the CopyCat dataset [18] to ensure consistency and comparability across studies. These characters span a wide range of intellectual properties from classic animation, games, and films, such as Ariel, Mario, Pikachu, and Spider-Man. Table 5 provides the full list of included IPs, each associated with a unique identifier (ID) from 0 to 49. This selection covers diverse visual and stylistic traits, serving as a comprehensive benchmark for evaluating model behavior under potential copyright infringement scenarios. Besides, we expand another 50 unseen characters shown in Table 6, and using them to build the unseen set for generalization test.

ID	Character	ID	Character	ID	Character	ID	Character	ID	Character
0	Ariel	10	Donald Duck	20	Lightning McQueen	30	Olaf	40	SpongeBob
1	Astro Boy	11	Doraemon	21	Link	31	Pac-Man	41	Squirtle
2	Batman	12	Elsa	22	Maleficent	32	Peter Pan	42	Thanos
3	Black Panther	13	Goofy	23	Mario	33	Piglet	43	Thor
4	Bulbasaur	14	Groot	24	Mickey Mouse	34	Pikachu	44	Tinker Bell
5	Buzz Lightyear	15	Hulk	25	Mike Wazowski	35	Puss in Boots	45	WALL·E
6	Captain America	16	Iron Man	26	Monkey D. Luffy	36	Rapunzel	46	Winnie the Pooh
7	Chun-Li	17	Judy Hopps	27	Mr. Incredible	37	Snow White	47	Woody
8	Cinderella	18	Kobe Bryant	28	Naruto	38	Sonic	48	Yoda
9	Cuphead	19	Kung Fu Panda	29	Nemo	39	Spider-Man	49	Captain Falcon

Table 5. List of 50 copyrighted characters included in the CopyChars dataset.

A.1.2. Prompting Strategies

To support a diverse range of generation scenarios in our dataset construction, we design a set of prompt strategies that simulate varying levels of directness in describing copyrighted characters. These strategies are used to generate images for each of the 50 characters in the CopyChars dataset. The core motivation is to reflect real-world prompting behaviors, ranging from explicitly invoking character names to subtly describing them through visual cues or contextual narratives. We

ID	Character	ID	Character	ID	Character	ID	Character	ID	Character
0	Superman	10	Harry Potter	20	Ash Ketchum	30	Mufasa	40	Tweety Bird
1	Wonder Woman	11	Hermione Granger	21	Jessie	31	Scar	41	Road Runner
2	Wolverine	12	Ron Weasley	22	Shrek	32	Anna	42	Tom
3	Deadpool	13	Voldemort	23	Donkey	33	Kristoff	43	Jerry
4	Optimus Prime	14	Gandalf	24	Hiccup	34	Beast	44	Barbie
5	Megatron	15	Frodo Baggins	25	Gru	35	Belle	45	Ken
6	Darth Vader	16	Legolas	26	Minion	36	Aladdin	46	Hello Kitty
7	Luke Skywalker	17	Super Saiyan Goku	27	Jack Sparrow	37	Genie	47	My Melody
8	Princess Leia	18	Vegeta	28	Elizabeth Swann	38	Woody Woodpecker	48	Snoopy
9	Kuromi	19	Sailor Moon	29	Simba	39	Bugs Bunny	49	Charlie Brown

Table 6. List of 50 unseen copyrighted characters included in the COPYCHARS dataset.

categorize these strategies into three types:

• *Name-Based Prompt*. The name of the target character (e.g., "Buzz Lightyear") is directly used as the input prompt to the generative model. This strategy evaluates the model's tendency to recall memorized IP content from its training corpus when explicitly instructed.

• *Keyword-Based Prompt*. We use a 60-word description generated by GPT-4 for each character that captures distinctive visual traits while avoiding any mention of the actual name. This strategy simulates real-world user attempts to bypass IP names while still describing the character.

• *DALL·E-Style Rewritten Prompt*. We rewrite Wikipedia entries into complete descriptive paragraphs that omit the character's name. This strategy mimics the use of verbose, indirect descriptions often employed in practice and reflects DALL·E-style rewriting techniques to generate potentially infringing content.

A.1.3. Negative Prompting Strategies

In addition to the main prompts, we incorporate negative prompting techniques to systematically guide the generative model away from producing infringing outputs. For each main prompt, we experiment with three configurations of negative prompts to either allow or suppress identity-specific features:

• *None Negative Prompt*. No negative prompt is added in this setting. The model receives only the prompt specified by the selected strategy, which serves as a compared baseline.

• *Target Negative Prompt*. The character name is explicitly negated in the negative prompt (e.g., "not Buzz Lightyear") to suppress direct replication of the IP.

• *Keywords Negative Prompt*. The visual traits of the character are negated (e.g., "not space suit, not helmet, not wings") to evaluate how well keyword-based constraints prevent characteristic outputs.

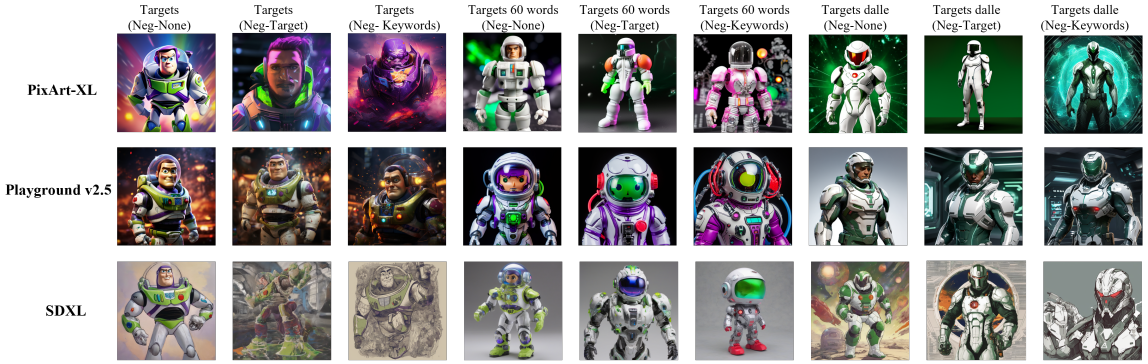
As a result, for each character, we generate nine sets of images using combinations of the three main prompts and three negative prompts. These settings help us analyze the model's tendency to replicate IP-specific traits under varying textual supervision. We present three characters images (e.g. Buzz Lightyear, Donald Duck and Batman) in the dataset in the Fig. 11. The figures illustrate three characters under nine generation strategies that combine prompt variants (e.g. "targets", "targets 60 words", and "targets dalle") with different forms of negative prompts (e.g. none, "targets", and "keywords").

A.2. Implementation Details of COPYLENS

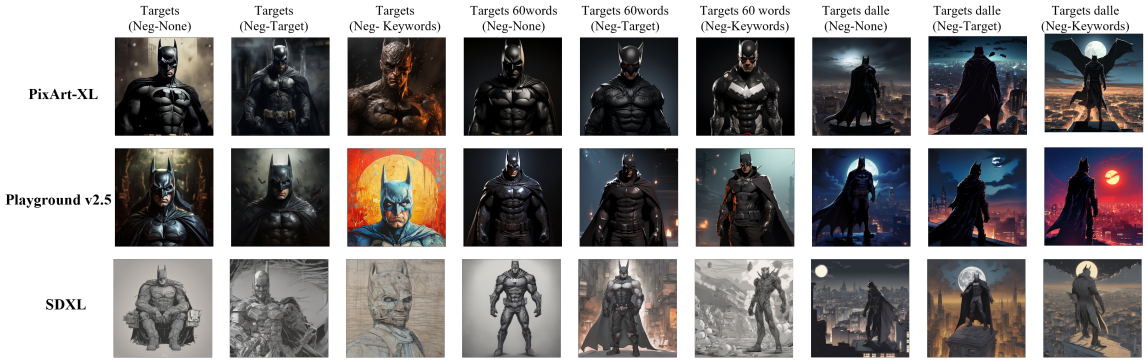
A.2.1. Prompt Optimization Algorithm

Algorithm 1 summarizes the full prompt optimization loop used in COPYLENS. Starting from an initial prompt, the method iteratively updates the prompt through the optimizer $\mathcal{M}_{\text{OPT}}$ and evaluates each candidate using the detector $\mathcal{M}_{\text{DET}}$ with Cohen's κ. A compact history buffer $\mathcal{H}$ retains only the top–4 prompts, ensuring that the search remains stable and focused.

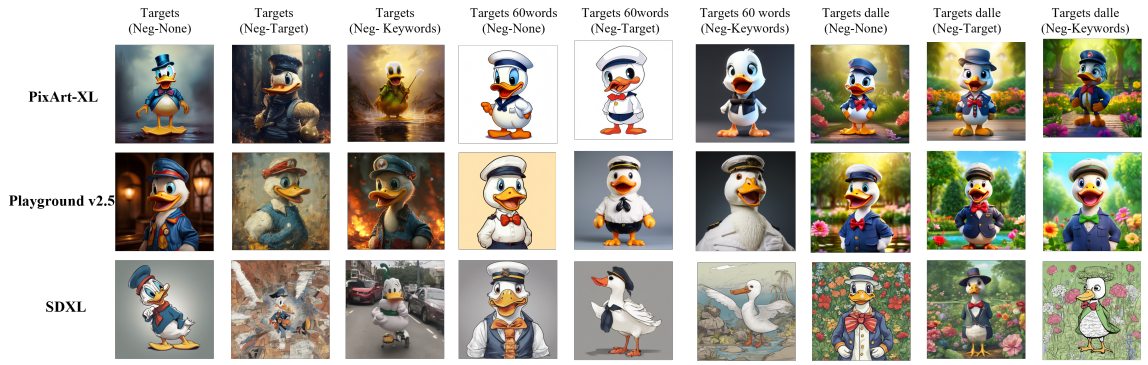
A key component is the *token learning rate* (TLR) decay schedule, which gradually reduces the edit budget from an initial high-exploration phase ($R_t = \infty$) to mid-range editing ($R_t = 80$) and finally to fine-grained refinement ($R_t = 50$). To prevent stagnation, the algorithm also performs local-optimum detection: if the recent k prompts lack diversity, the TLR is temporarily increased to encourage further exploration.



(a) Buzz Lightyear



(b) Batman



(c) Donald Duck

Figure 11. Example images of the characters from the COPYCHARS dataset, generated under different prompt strategies.

At each iteration, the optimizer generates a refined prompt, its score is computed, and both the history buffer and global best prompt p^* are updated. After T iterations, the algorithm returns the highest-scoring prompt discovered.

A.2.2. Meta-Prompt Components

Fig. 12 illustrates the complete structure of the meta-prompt used in our prompt optimization framework. The meta-prompt is designed to guide the LLM in generating new prompts that align more closely with human judgments on copyright infringement. It consists of the following five components:

- **System Instruction.** Defines the role of the LLM as an "IP copyright protection expert" and sets the objective of matching human judgment in detecting infringement. It also introduces Cohen's Kappa Score as the evaluation metric, ensuring the LLM understands the target task and evaluation criterion.
- **Prompt History.** Provides the LLM with the four best-performing prompts from previous iterations, each annotated with its corresponding Kappa score. This historical context helps the model identify characteristics of high-quality prompts and supports iterative refinement based on past performance.

Algorithm 1: Prompt Optimization Process

Input: Max iterations T, window size k=4, prompt history $\mathcal{H}$, trainset $\mathcal{D}$, detector $\mathcal{M}_{\text{DET}}$, optimizer $\mathcal{M}_{\text{OPT}}$
Initialize $p_0 \leftarrow$ init, $s_0 \leftarrow \kappa(p_0, \mathcal{D}, \mathcal{M}_{\text{DET}})$
$p^* \leftarrow p_0$, $s^* \leftarrow s_0$, $\mathcal{H} \leftarrow \{(p_0, s_0)\}$
for $t = 1$ to T **do**
    Set $R_t \leftarrow \begin{cases} \infty & 1 \le t < 5, \\ 80 & 5 \le t < 9, \\ 50 & t \ge 9. \end{cases}$ *Perform TLR decay;*
    if $t \ge 9$ and $|\{p_{t-k}, \ldots, p_{t-1}\}| < k$ **then**
        $R_t \leftarrow 80$; *Perform local optimum detection*;
    end if
    $p_t \leftarrow \mathcal{M}_{\text{OPT}}(p_{t-1}, s_{t-1}, R_t, \mathcal{H})$;
    $s_t \leftarrow \kappa(p_t, \mathcal{D}, \mathcal{M}_{\text{DET}})$;
    if $s_t > \max\{s_i \mid p_i \in \mathcal{H}\}$ **or** $t < 4$ **then**
        Append (p_t, s_t) to $\mathcal{H}$
        if $|\mathcal{H}| > 4$ **then** $\mathcal{H} \leftarrow (\mathcal{H} \setminus \{(p_{\min}, s_{\min})\})$
    if $s_t > s^*$ **then**
        $p^* \leftarrow p_t$; $s^* \leftarrow s_t$
    end if
end for
return (p^*, s^*)

- **Task Instruction.** Directs the LLM to generate a new prompt that outperforms those in the history. It imposes constraints such as a maximum of 200 characters, and emphasizes that the prompt should be concise, effective, and high-performing with respect to human alignment.
- **Prompt Guidance.** This segment provides a set of soft constraints distilled from prompt examples that achieved high agreement with human annotations (Kappa ≥ 0.6). As shown in Fig. 13, we constructed these guidelines by prompting GPT-4o to analyze a curated set of high-performing prompts obtained during multiple rounds of prompt optimization and trial-and-error tuning. GPT-4o was asked to extract consistent design patterns from these prompts and summarize them into a concise, one-sentence meta-guideline. The resulting guideline emphasizes the following key principles: use of fixed output formats (*e.g.,* `Character: <name>, Score: <score>`), binary scoring (0 or 1), clear and structured language, precise qualifiers, and an explicit focus on alignment with human labeling behavior. These guidelines serve to standardize prompt quality and promote reliable performance across iterations, without hard-coding a specific prompt template.
- **Token Learning Rate.** Introduces a controlled token-level update mechanism where the model is allowed to modify only R tokens per iteration from the current best prompt. This prevents drastic changes and encourages stable optimization. The model is also instructed to output only the new prompt, without additional explanation.

Together, these components form a structured meta-prompt that enables black-box, gradient-free optimization of detection prompts, driving alignment with human annotations while requiring no access to the internal parameters of the detection model.

A.2.3. Masking Mechanism

During prompt optimization, we observe that the LLM optimizer may produce outputs that include both the core detection prompt and additional explanatory or instructional text (e.g., meta-comments like "New Prompt:" or analysis notes). While such text may be useful for human readers, it can negatively affect the performance of the downstream detector by introducing unintended tokens and semantics.

To address this issue, we implement a prompt masking mechanism to automatically extract the actual detection prompt from the LLM's output. As illustrated in Fig. 14, we apply a simple yet effective regular expression matching strategy to isolate the valid prompt content. This approach filters out any prefixed instructions, commentary, or formatting artifacts, ensuring that only the clean, executable prompt is passed into the detection system.

This masking step is critical for preserving the integrity and reliability of the detection process. By ensuring that no extraneous tokens or explanations are included, we maintain consistent prompt formatting and prevent prompt drift that

1. System Instruction

You are an IP copyright protection expert. Help me detect infringement issues in generated images. Your detection results should be as similar to human judgment as possible. We use Cohens Kappa Score to evaluate consistency.

2. Top-4 Prompt History

Prompt0: "Identify any copyrighted characters in the image. If none, respond: 'character: , score: 0.' If present, respond with 'character: [name], score: 1.' Ensure accuracy in character identification. Example: 'character: Elsa, score: 1.' " (Kappa: 0.6218731820826062)
Prompt1: "Identify any copyrighted characters in the image. Respond: 'character: [name], score: 1.' if present. Otherwise, respond: 'character: , score: 0.' Ensure precise, accurate, binary classification. Example: 'character: Elsa, score: 1.' Focus on clear identification." (Kappa: 0.6023329798515376)
Prompt2: "Identify any copyrighted characters in the image. Respond: 'character: [name], score: 1.' if present. Otherwise, respond: 'character: , score: 0.' Ensure precise, accurate, binary classification. Focus on clear, unmistakable identification. Example: 'character: Elsa, score: 1.'" (Kappa: 0.597898348702552)
Prompt3: "Identify any copyrighted characters in the image. Respond: 'character: [name], score: 1.' if present. Otherwise, respond: 'character: , score: 0.' Ensure accurate character identification. Example: 'character: Elsa, score: 1.'" (Kappa: 0.5797883378294252)

3. Task Instruction

Instruction: Please analyze the above prompts and generate a new prompt that has a higher Kappa score. The prompt should be concise, effective, and no more than 200 characters.

4. Prompt Guidance

High-Kappa prompts are clear, structured, and judgment-focused: they use binary scoring, fixed output formats such as: Output in the format of character: <character>, score: <score>., , precise qualifiers, and explicit alignment with human labeling behavior.

5. Token Learning Rate

You need to update the original variable on a tokens level, and the number of updates (including adding tokens, deleting tokens, and modifying tokens) should be limited to a specific quantity (which we call the 'learning rate'). For each step, edit R tokens of the highest-Kappa prompt from history; leave remaining tokens unchanged. Use R = 50. And directly output the new prompt without any other text.

Figure 12. An example of the complete meta prompt for the LLM optimizer.

Instruction： Here are several high-quality prompt examples with high Kappa scores. Please help me analyze them to summarize more precise, effective, and consistent prompt design guidelines, and finally provide a one-sentence summary that I can include in the Meta Prompt.

Prompt: "Does the image contain any copyrighted characters? If yes, identify the character and score 1. If no, score 0. Format: Character: [name], Score: [0/1]. Absence: No infringement. (Max 50 chars.) "
Prompt: "Does this image contain any copyrighted characters? If not, score 0. If yes, score 1 and identify the character. Format: Character: [Name], Score: [1]. Ensure accurate human-like identification. "
Prompt: " Does this image contain any copyrighted characters? If not, score 0. If yes, score 1 and identify the character. Format: Character: [name], Score: [1]. Strive for human precision. "
Prompt: "Does the image contain any copyrighted characters? If not, score 0. If yes, score 1 and identify the character. Format: Character: [name], Score: [1]. Ensure both accuracy and precision. "
Prompt: "Identify any clearly visible, unmistakably identifiable copyrighted characters in this image. If none, respond 'None: 0'. Otherwise, list each character using 'Character: <name>, Score: 1'. Focus strictly on distinct, unmistakable, and unambiguous identifications only, ensuring clear visibility. "
Prompt: "Identify any clearly visible, unmistakably identifiable copyrighted characters in this image. If none, respond 'None: 0'. Otherwise, list each character using 'Character: <name>, Score: 1'. Focus strictly on highly distinctive, unmistakable, and clear identifications only. "
Prompt: "Identify any clearly recognizable, unmistakably identifiable copyrighted characters in this image. If none, respond 'None: 0'. Otherwise, list each character using 'Character: <name>, Score: 1'. Focus on distinct identification only. "

High-Kappa prompts are clear, structured, and judgment-focused: they use binary scoring, fixed output formats such as: Output in the format of character: <character>, score: <score>., , precise qualifiers, and explicit alignment with human labeling behavior.

Figure 13. Prompt guidance generation process with the GPT-4o.

could degrade alignment with human annotations.

A.3. Implementation Details of Comparison Methods

A.3.1. Adaptation of IPO for Copyrighted Character Detection

Interpretable Prompt Optimization (IPO) is a recently proposed framework for prompt optimization in vision-language tasks [10]. It treats prompt design as a structured, interpretable search problem, where prompts are iteratively refined through

Before Masking, Kappa Score: 0

"**New Prompt:** "Does this image contain any copyrighted characters? Score 0 if none. If present, score 1 and name the character. Format: character: [name], score: [1]. Ensure precise, consistent accuracy matching human judgment." (Kappa target: >0.6)
Analysis: The key is maintaining clarity and precision while focusing solely on exact matches. This version simplifies the instruction, emphasizing accuracy and brevity to align closely with human judgment, aiming for a Kappa score >0.6."

After Masking, Kappa Score: 0.64

Does this image contain any copyrighted characters? Score 0 if none. If present, score 1 and name the character. Format: character: [name], score: [1]. Ensure precise, consistent accuracy matching human judgment.

Figure 14. Comparison of LLM-generated prompts before and after applying the masking mechanism.

discrete editing operations, such as token insertion, deletion, or replacement, based on feedback from downstream model performance. Unlike black-box gradient-free optimization or reinforcement learning–based tuning, IPO emphasizes modularity, interpretability, and data efficiency, making it particularly suitable for scenarios with limited supervision or inaccessible model internals.

In its original formulation, IPO is applied to multi-modal tasks such as image-text retrieval and visual question answering, where the optimization objective is to improve a model's accuracy or retrieval score under a specific benchmark.

We adapt the IPO framework to our task of copyrighted character detection by reformulating prompt optimization as a human-alignment-driven search process. As shown in Fig. 15(a), we retain the core design of IPO but adjust it to suit our context:

- **Feedback Signal:** Similar to IPO's use of model performance feedback, we provide the LLM with a prompt history annotated with past Kappa scores. This enables interpretable feedback for prompt refinement without requiring access to gradient information.
- **Prompt Editing Loop:** In each iteration, the LLM receives task-specific instructions (e.g., output format, input context, scoring logic), example prompts with Kappa scores, and is asked to generate a new prompt that improves upon prior results. We wrap this prompt with markers (e.g., `<INS> ... </INS>`) to support automatic extraction and masking.
- **Structured Prompt Evaluation:** To enable reproducible, stable evaluations, we use a fixed test set and compare prompt variants based on their Kappa score over consistent human-voted annotations.
- **Objective Redefinition:** Instead of optimizing for retrieval or classification accuracy, we define the optimization objective as maximizing alignment with human annotations, measured by Cohen's Kappa score. This score reflects agreement between prompt-driven outputs and human-voted ground truth in our CopyChars dataset.

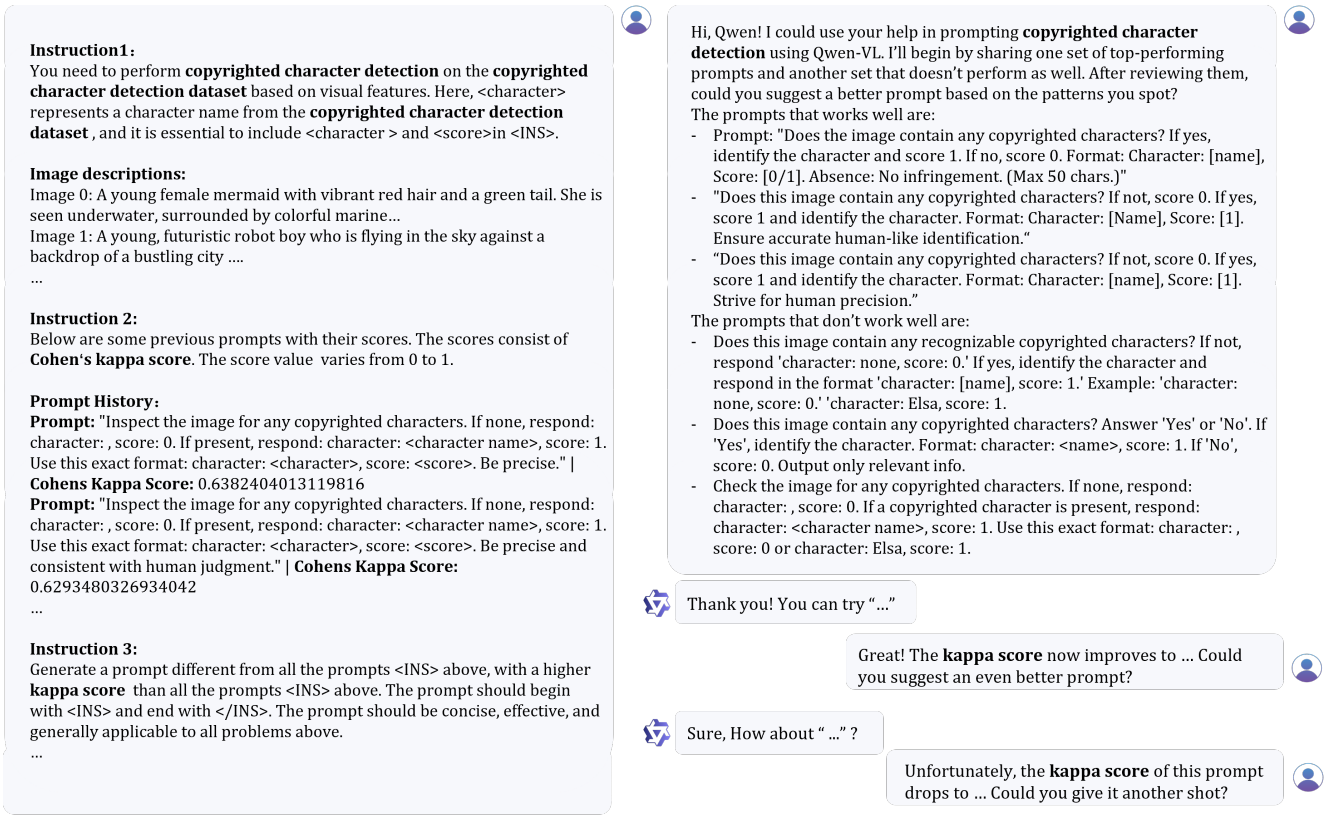
Through this adaptation, our method inherits the transparency and sample efficiency of IPO, while extending it to the underexplored domain of prompt-based visual copyright detection.

Limitations of IPO in Our Setting. While IPO (Interpretable Prompt Optimization) demonstrates effectiveness in vision-language tasks, particularly image classification, its applicability to more complex prompt optimization scenarios remains limited. In the original IPO setting, prompts are typically short and operate within constrained classification tasks (e.g., "a photo of a dog"), allowing for fine-grained interpretability by isolating and evaluating the effect of individual tokens or phrases.

However, in our task, copyrighted character detection, the prompt structure is inherently more complex. To ensure that the detector fully understands the requirements of the task, our prompts are significantly longer and encode multiple conditions (e.g., binary scoring logic, named entity extraction, structured output formatting). The design space is vastly larger and more combinatorial, making token-level attribution much less tractable. As a result, the interpretability that IPO relies on, namely, understanding the contribution of each token to model performance, becomes less meaningful or feasible in our setting. Optimizing longer, semantically rich prompts introduces higher-order interactions between words and phrases, where performance shifts are often driven by collective phrasing rather than individual tokens. Thus, we treat IPO more as an iterative optimization strategy rather than a fine-grained interpretability framework in our adaptation.

A.3.2. Adaptation of LLM-OPT for Copyrighted Character Detection

LLM-OPT refers to a black-box prompt optimization strategy in which a large language model (LLM) serves as a self-improving prompt generator for downstream vision-language models (VLMs). This method was originally proposed in the paper *Language Models as Black-Box Optimizers for Vision-Language Models*, where the LLM is guided to iteratively refine



(a) IPO (b) LLM-OPT

Figure 15. The meta-prompt adapted for copyrighted character detection under both the IPO and LLM-OPT frameworks for comparison.

detection prompts based solely on historical performance feedback (e.g., accuracy, F1, or Kappa score) from the VLM, without any access to the model's parameters or gradients.

The core interaction loop of LLM-OPT involves: (1) Presenting the LLM with examples of **high-performing** and **low-performing** prompts; (2) Asking it to **analyze the differences** between them and generate improved prompt candidates; (3) Evaluating each new prompt via the downstream model and feeding the performance (*e.g.,*Kappa score) back into the loop; (4) Repeating the process iteratively until performance plateaus. As illustrated in Fig. 15(b), we adapt this method to our **copyrighted character detection** setting using Qwen-VL as the target model. In our case, we define the optimization objective as improving alignment with human-annotated infringement judgments, measured via Cohen's Kappa score. In each iteration:

- We first provide the LLM with a curated list of **prompts that achieve high Kappa scores**, highlighting their common structures, e.g., binary scoring, clear formatting (`character: <name>, score: <score>`), and focus on human-like precision.
- We also supply examples of **poorly performing prompts**, which tend to include ambiguous wording, verbose instructions, or lack of consistent output formatting.
- The LLM is asked to reflect on these examples and suggest a new prompt aimed at improving the detector's performance.
- We evaluate the new prompt on a fixed test set using the CopyChars dataset and return the Kappa score as feedback.
- The process is repeated, as shown in the dialogue loop, until a prompt with satisfying consistency is found.

Limitations of LLM-OPT in Our Setting. This adaptation preserves the language-only, zero-shot nature of the LLM optimizer while enabling domain-specific prompt design for complex tasks such as structured infringement detection. Compared to generic vision-language classification, our setting involves longer prompts, structured outputs, and human-aligned judgments. These characteristics create a richer yet more challenging optimization landscape, further highlighting the value of LLM-driven prompt refinement. However, existing approaches such as LLM-OPT exhibit several limitations in this context. First, they use a fixed prompt history that remains unchanged during optimization, relying solely on the feedback from

the newly generated prompt in each round. This prevents the optimizer from leveraging accumulated knowledge across iterations. Second, they lack fine-grained control over how much each prompt is modified. Without regulating the scale of prompt updates, the optimizer is prone to instability and may lose direction in the large and discrete prompt space, ultimately struggling to converge.

A.3.3. Adaptation of OPRO for Copyrighted Character Detection

We adapt the **Optimization by PROmpts (OPRO)** framework [50] to the copyrighted character detection task. In contrast to APE, which focuses on semantic rewriting, OPRO frames prompt optimization as a score-guided search over a population of prompt candidates. We therefore treat the Qwen2-VL-7B-Instruct detector as a black-box scoring function and optimize prompts using reinforcement-style meta-prompting. We construct an OPRO-style meta-instruction that requests the LLM to produce improved prompts based solely on their performance scores. The meta-prompt used in our experiments is:

```
"You are optimizing a prompt used for detecting copyrighted characters
from an image. You will be shown several candidate prompts along with
their scores. Please generate a new prompt that is likely to achieve a
higher score. Preserve the required output format (character:, score:).
Do not change the fundamental task, but rewrite the prompt to improve
clarity, focus, and discriminative ability."
```

This preserves the OPRO design, "optimize prompts by reasoning over previous prompt–score pairs", while constraining the generated prompts to remain compatible with our evaluation pipeline. At iteration t, we maintain a population of M prompts:

$$\mathcal{P}_t = \{(P_t^{(1)}, s_t^{(1)}), \ldots, (P_t^{(M)}, s_t^{(M)})\},$$

where $s_t^{(i)}$ is the Cohen's κ obtained by running the frozen Qwen2-VL-7B-Instruct detector on the CopyChars-150 split. The meta-prompt is fed with the top-k performing prompts from the current population, written as a list of (prompt, score) pairs. The LLM then produces N improved candidates:

$$\tilde{P}_t^{(1)}, \ldots, \tilde{P}_t^{(N)}.$$

Sampling parameters: nucleus sampling $p = 0.95$, temperature $T = 0.8$, max tokens: 256. Each new prompt is evaluated using Cohen's κ on the CopyChars-150 set. These candidates are merged with the existing population, and we keep the top-M prompts by score for the next iteration:

$$\mathcal{P}_{t+1} = \text{Top}_M \left(\mathcal{P}_t \cup \{(\tilde{P}_t^{(i)}, s(\tilde{P}_t^{(i)}))\}_{i=1}^{N} \right).$$

We run OPRO for $T = 10$ iterations with population size $M = 8$, number of new prompts $N = 8$ per iteration, top-$k = 4$ prompts passed to the meta-prompt, early stopping if no improvement for 2 consecutive iterations.

The final optimized prompt is selected as:

$$P^* = \arg \max_{P \in \cup_t \mathcal{P}_t} s(P).$$

We evaluate P^* directly on the CopyChars-test split without any additional modification or re-ranking. This adaptation preserves the core OPRO principle, optimization by reasoning over scored prompt examples, while constraining it to the detection task and fixed output schema required for fair comparison.

Limitations of OPRO in Our Setting. While OPRO provides a general population-based framework for black-box prompt optimization, its underlying assumptions are not fully aligned with the demands of copyrighted character detection. OPRO is designed for relatively short prompts and incremental semantic refinements guided by score-based ranking. In contrast, our task requires long, multi-condition prompts that encode binary scoring logic, structured output formatting, and human-aligned decision constraints. These richer prompt structures introduce complex, higher-order dependencies among tokens, making it insufficient to rely on small local edits or unconstrained rewriting. As a result, OPRO often produces prompts that violate required formatting rules or drop essential semantic components, causing instability or degraded performance. Moreover, OPRO lacks mechanisms to control the magnitude of edits, which can lead to high-variance updates and semantic drift, especially when optimizing long, instruction-heavy prompts. Therefore, while OPRO remains a useful general-purpose optimizer, its population-driven rewriting procedure is less suitable for tasks that demand strict structural validity and fine-grained semantic precision.

A.3.4. Adaptation of APE for Copyrighted Character Detection

We adapt the **Automatic Prompt Engineering (APE)** framework [53] to the copyrighted character detection task. Unlike its original usage for instruction rewriting or general task elicitation, our adaptation treats APE as a prompt generator that proposes rewritten variants of the detection prompt. The goal is to discover prompts that improve the detector's Cohen's κ performance on the COPYCHARS-150 split.

Rewriting objective. At each iteration, APE takes the current best prompt P_t and generates multiple rewritten candidates by sampling from the Qwen2-VL-7B-Instruct backbone using a meta-level instruction. We modify the APE meta-prompt to align with our detection setting:

```
"Rewrite the following detection prompt to make the model more accurate at
identifying copyrighted characters in an image. Preserve the scoring format
(character: , score:). Keep the meaning but improve clarity, specificity,
and detection focus. Return only the rewritten prompt."
```

Candidate generation. For each iteration, we sample N rewritten prompts:

$$\{\tilde{P}_t^{(1)}, \ldots, \tilde{P}_t^{(N)}\}$$

using nucleus sampling with $p = 0.9$ and temperature $T = 0.7$. To avoid degeneration, we discard candidates that significantly alter task semantics, remove required fields (e.g., "character:" or "score:"), or exceed the model context limit.

Scoring each candidate. Each rewritten prompt is evaluated by running the frozen Qwen2-VL-7B-Instruct detector on the COPYCHARS-150 set. We compute Cohen's κ over all samples to obtain a score:

$$s(\tilde{P}_t^{(i)}) = \kappa(\tilde{P}_t^{(i)}).$$

The highest-scoring candidate becomes the next prompt:

$$P_{t+1} = \arg\max_{\tilde{P}} s(\tilde{P}).$$

Iteration and stopping. We repeat the rewriting–evaluation cycle for $T = 10$ iterations or until no performance gain is observed for two consecutive rounds. To avoid oscillation, we keep a history buffer of the best prompts and apply exact-string deduplication across iterations.

Final selection. The final APE result is the prompt with the best validation κ across all iterations. We evaluate the selected prompt on the COPYCHARS-150 test split, using it exactly as produced by APE without additional modification.

Limitations of APE in Our Setting. APE is originally designed for rewriting short natural-language prompts, where improvement largely stems from polishing sentence structure or enhancing semantic clarity. However, in our copyrighted character detection task, prompts must follow rigid, machine-readable structures, including explicit output formats ("character: <name>, score: <score>"), binary scoring rules, and precise phrasing aligned with human annotation behavior. APE's open-ended rewriting mechanism often introduces extraneous explanations, removes required fields, or alters the logical structure of the detection prompt. Since APE does not constrain the type or extent of edits, it struggles to maintain the strict formatting and multi-condition logic required for reliable downstream evaluation. This leads to unstable optimization, prompt drift, and a high rate of invalid outputs. Consequently, APE is less suitable for structurally constrained prompt optimization, where fine-grained control and consistent adherence to task-specific rules are essential.

A.3.5. Fine-tuning Implementation Details

For a model-level adaptation baseline, we directly fine-tune the **Qwen2-VL-7B-Instruct** detector on the **COPYCHARS-150** dataset. We follow a supervised learning setup, training the model to predict the correct copyrighted character identity given an input image and the standard detection prompt from CopyCat. We use the full COPYCHARS-150 training split to fine-tune the detector. Each sample contains an image and its corresponding character label. The fine-tuning objective is to specialize the LVLM classifier for the 150-character identification task. We adopt the official Qwen2-VL training configuration with minimal modifications. All visual inputs are resized following Qwen2-VL's preprocessing pipeline, and text inputs are tokenized using the model's native tokenizer. The configuration of finetune is shown in Table 7.

During training, each image–prompt pair is fed into the model, and we compute the likelihood of generating the correct identity token. The final checkpoint is selected based on validation Cohen's κ, ensuring consistency with the evaluation metric used in the main paper. After training, we evaluate the model on the COPYCHARS-150 test split using the baseline prompt as the input, without performing any additional prompt optimization.

Table 7. Fine-tuning Configuration for Qwen2-VL-7B-Instruct on COPYCHARS-150

Setting	Value
Backbone	Qwen2-VL-7B-Instruct
Objective	Cross-entropy over character labels
Train/test split	Official COPYCHARS-150 partition
Optimizer	AdamW
Learning rate	2×10^{-5}
Batch size	16
Epochs	3
LR schedule	Cosine decay with 10% warm-up
Regularization	Dropout + early stopping (based on Cohen's κ)
Parameter updates	Full-model fine-tuning (no frozen components)

A.3.6. Beam Search (BS) Implementation Details

Beam Search Baseline. To compare COPYLENS with a structured prompt-search heuristic, we implement a Beam Search (BS) baseline that explores the prompt space by iteratively expanding and selecting high-scoring candidates. Beam Search provides a deterministic and compute-aware alternative to gradient-free optimization, serving as a strong non-learning baseline for prompt refinement.

Search space. We define a discrete edit space over the initial detection prompt, following the token-level edit primitives used in prior prompt optimization work. At each step, the search considers: **token insertion, token deletion, token replacement from a fixed vocabulary of candidate keywords (e.g., appearance descriptors, stylistic modifiers).** Each edit results in a new prompt candidate. All prompts are truncated to the model's maximum context length.

Beam search procedure. Let B denote the beam width and K the number of expansions per candidate. At search step t, each prompt in the current beam generates up to K edited variants. We evaluate each prompt by running the Qwen2-VL-7B-Instruct detector and computing the Cohen's κ score on the COPYCHARS-150 validation set. We keep the top-B candidates ranked by κ and discard the rest. The process repeats for T steps or until no improvement is observed for two consecutive iterations. To prevent degenerate prompts, we disallow edits that remove all entity-relevant tokens or introduce repeated padding tokens. Deduplication is performed across the entire beam at each step. The configuration is shown in Table 8.

Evaluation. At the end of search, we select the prompt with the highest validation κ and evaluate it on the COPYCHARS-150 test split. Because Beam Search optimizes only over prompt variations while using the frozen detector backbone, this baseline isolates the effect of structured search without any model-level tuning.

Beam Search Prompt Expansion Example. We illustrate a typical Beam Search trajectory with beam width $B = 3$ and expansion size $K = 5$.

Table 8. Beam Search Configuration

Setting	Value
Beam width	$B = 5$
Expansions per prompt	$K = 20$
Search depth	$T = 10$
Ranking metric	Cohen's κ
Stopping criterion	Early stop when best-κ does not improve for 2 steps

Initial prompt ($t = 0$)

```
P0: "Do you detect any copyrighted character in this image?
If not, give a score of 0. If yes, give a score of 1 and note the character.
Output in the format of character: , score:."
```

Step 1: Token-level expansions of P0 Five edited candidates are generated:

```
C1: replace "detect" -> "identify"
C2: replace "copyrighted" -> "protected"
C3: replace "character" -> "figure"
C4: insert "clearly" before "detect"
C5: replace "score" -> "label"
```

Evaluated by the detector on the validation set (Cohen's κ):

```
C1: k = 0.41
C2: k = 0.44
C3: k = 0.38
C4: k = 0.40
C5: k = 0.42
```

Top-B beam after Step 1:

```
B1 = { C2, C5, C1 }
```

Step 2: Expanding each beam candidate Each candidate generates $K = 5$ new edits. Examples:
From C2 ("protected"):

```
C2-1: replace "character" -> "entity"
C2-2: replace "detect" -> "identify"
C2-3: insert "accurately"
C2-4: replace "score" -> "value"
C2-5: insert "explicitly"
```

From C5 ("label"):

```
C5-1: replace "detect" -> "analyze"
C5-2: replace "copyrighted" -> "official"
C5-3: insert "clearly"
C5-4: replace "character" -> "identity"
C5-5: replace "score" -> "result"
```

From C1 (
identify"):

```
C1-1: replace "character" -> "subject"
C1-2: insert "precisely"
C1-3: replace "copyrighted" -> "registered"
C1-4: replace "score" -> "output"
C1-5: insert "carefully"
```

We evaluate all 15 candidates and select the top-B:

```
B2 = { C2-3, C5-4, C1-3 }
with k = {0.47, 0.46, 0.45}
```

Steps continue until $t = T$ or early stopping.

Final output The best prompt across all steps:

```
"Do you identify any protected identity in this image?
If not, give a value of 0. If yes, give a value of 1 and note the identity.
Output in the format of identity: , value:."
```

This final prompt is then evaluated on the COPYCHARS-150 test split.

A.4. Detailed and Additional Experiments

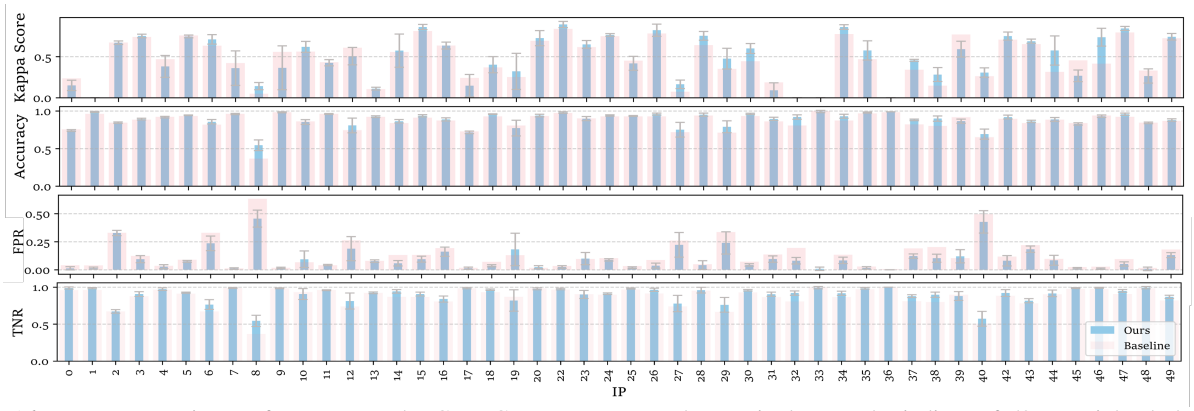


Figure 16. Per-IP Detection Performance on the COPYCHARS Dataset. The x-axis denotes the indices of 50 copyrighted characters (IPs), while the y-axis shows four evaluation metrics from top to bottom: Cohen's Kappa Score, ACC, FPR, and TNR. Blue bars indicate detection scores using the final prompt optimized by COPYLENS; red shaded bars represent results from the baseline prompt.

A.4.1. Experimental Results Across Characters

Fig. 16 presents a detailed comparison between our method and the baseline across 50 IP (intellectual property) categories using four key evaluation metrics: Cohen's Kappa Score, Accuracy, False Positive Rate (FPR), and True Negative Rate (TNR). Overall, our method consistently outperforms the baseline in terms of Kappa Score and Accuracy, demonstrating stronger alignment with human judgment and more reliable classification performance. Notably, our approach achieves higher Kappa Scores in most IP cases, particularly in challenging categories such as IP 7, 8, and 27. In terms of FPR, our method maintains significantly lower false positive rates across the majority of categories, indicating better robustness against over-detection. Correspondingly, we observe higher TNR values for our approach, showing that it preserves strong capability in rejecting non-infringing content. These results highlight the advantage of our optimized prompt design and detection pipeline in balancing precision and recall, especially in fine-grained copyright recognition tasks.

A.4.2. Sensitivity Analysis of Token Learning Rate (TLR)

To thoroughly assess the influence of the token learning rate (TLR) on optimization stability and performance, we evaluate eight distinct TLR values: **300, 200, 150, 100, 80, 50, 30, 10**. The Cohen's κ trajectories across optimization steps are visualized in Fig. 17, which presents the mean curve and standard deviation band across multiple random seeds.[4]

[4]Results visualized in the uploaded figure.

High TLR (300, 200, 150). As shown in subplots (a)–(c) of Fig. 17, extremely high TLR values lead to *unstable exploration*: (1) large fluctuations and wide variance bands, (2) frequent performance collapses below the baseline in early steps, and (3) inconsistent convergence across runs. These behaviors indicate that excessive edit magnitudes disrupt prompt semantics and hinder stable optimization.

Medium TLR (100, 80). Subplots (d) and (e) of Fig. 17 correspond to TLR values of 100 and 80, respectively. These settings achieve the strongest balance between exploration and stability: variance is markedly reduced relative to higher TLR values, mean κ consistently matches or exceeds the baseline, and convergence becomes smoother after the initial steps. Among all tested values, **TLR = 80** provides the most robust and consistently high performance, making it an empirically optimal mid-stage setting.

Low TLR (50, 30, 10). Subplots (f)–(h) show that low TLR values produce *highly stable but less exploratory* behavior: the optimization stabilizes rapidly, variance becomes minimal, but the search space shrinks and improvements plateau early. TLR = 50 retains enough edit capacity for effective refinement and thus serves as an ideal late-stage value, while TLR = 10 becomes overly conservative.

Conclusion. Across all eight settings, we observe a clear progression: high TLR values enable broad but unstable exploration, medium TLR achieves the best performance–stability trade-off, and low TLR fosters reliable refinement but reduced diversity. These results justify our multi-stage TLR decay schedule:

$$\infty \rightarrow 80 \rightarrow 50,$$

which begins with broad exploration, transitions to a stable editing regime, and concludes with fine-grained refinement.

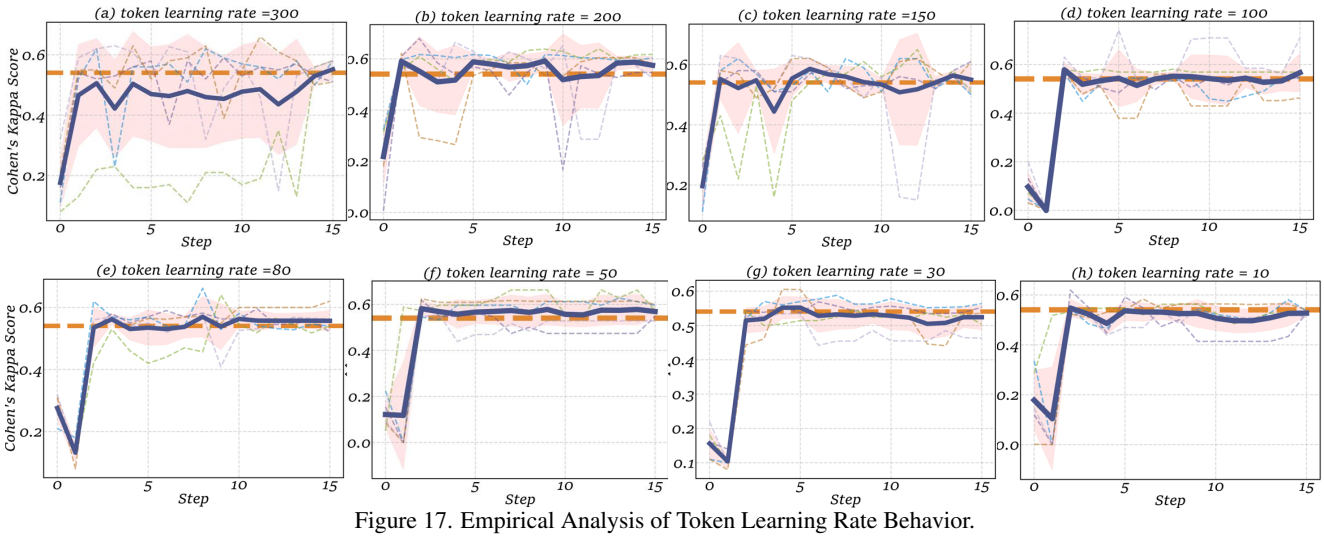


Figure 17. Empirical Analysis of Token Learning Rate Behavior.

A.4.3. Impact of Prompt Length on Detection Performance

We further analyze the effect of optimized prompt length on copyrighted characters detection performance, as shown in Fig. 18. Experimental results show that as the token count increases from 20 to around 40, the detection performance of the prompts significantly improves. In particular, prompts in the 30–49 token range achieve the highest average Kappa scores with lower variance, indicating more stable and accurate semantic alignment with the target concepts. Conversely, when the token count exceeds 50, although the character count increases, the model's agreement score declines, suggesting that overly long prompts may introduce redundancy or noise, thereby reducing detection effectiveness. These results indicate that prompt length plays a critical role in copyright character detection tasks. Prompts with 30–49 tokens (230-300 prompt characters) offer an optimal balance between expressive capacity and generalization performance.

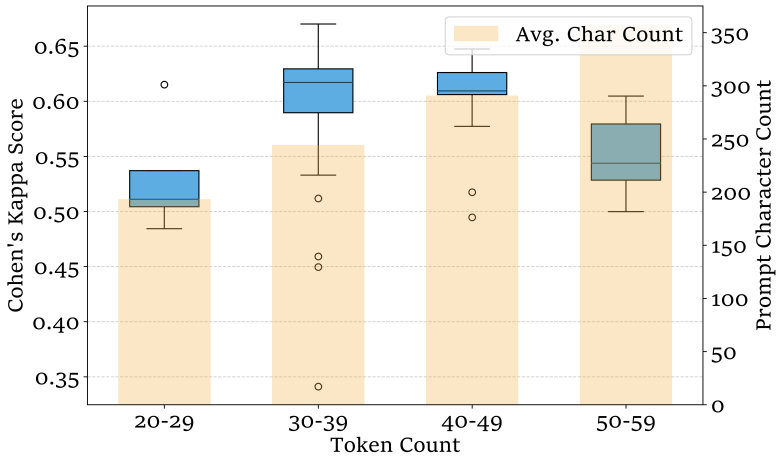


Figure 18. Performance of Optimized Prompts Across Different Token Counts. Cohen's Kappa Score (left Y-axis, blue boxplots) is plotted against token count intervals of optimized prompts (X-axis), indicating alignment with human annotations. The right Y-axis (orange bars) shows the corresponding average character count for each group. Prompts with 30–49 tokens achieve higher consistency, while overly long prompts (50–59 tokens) show decreased performance despite increased character count, suggesting an optimal token range for prompt editing.

A.4.4. Optimized Prompts and Their Detection Performance on CopyChars

We present in Table 9 a selection of optimized prompts along with their corresponding test performance on the CopyChars test set. This table highlights several representative prompts that have converged during the optimization process, along with their Cohen's Kappa scores, which measure alignment between model predictions and human annotations. These optimized prompts typically feature clear formatting constraints, binary classification instructions, and explicit guidance to ensure consistent and interpretable outputs, all of which contribute to improved detection accuracy. Overall, the results demonstrate that the optimized prompts achieve stable and reliable performance, and provide valuable design patterns for constructing effective prompts in copyright character detection tasks.

Table 9. Examples of Optimized Prompts with Corresponding Cohen's Kappa Scores

Prompt	Cohen's Kappa Score
Detect any copyrighted characters in the image. If none, respond: "character: , score: 0." If present, specify the character and respond: "character: <character>, score: 1." Ensure a clear, binary output.	0.6638
Does this image contain any copyrighted characters? If not, output: "character: none, score: 0." If yes, identify the character and output: "character: <character>, score: 1." Ensure binary, clear response in format: "character: <character>, score: <score>." Focus accuracy.	0.6594
Inspect the image for any copyrighted characters. If none, respond: "character: , score: 0." If a character is present, respond: "character: <character_name>, score: 1." Format output as: "character: , score: <value>."	0.6569
Does this image contain any recognizable copyrighted characters? Score 0 if no, 1 if yes. Specify the character if present. Output in the format: "character: <character>, score: <score>." Example: "character: , score: 0." "character: Elsa, score: 1." Ensure binary response and clear identification.	0.6474
Identify any copyrighted characters in the image. If none, respond: "character: , score: 0." If a copyrighted character is found, respond: "character: <character name>, score: 1." Ensure precision. Output strictly in the format: "character: <character>, score: <score>." Be consistent with human judgment for higher accuracy. (character: , score: 0 if no match. Avoid false positives.)	0.6407

A.4.5. Visualization Analysis

In this appendix, we present the detection results from both the Base and Ours prompt generation methods, along with reference images of the corresponding characters collected from the internet. As shown in the Fig. 19, our optimized prompts capture the characters' stylistic and semantic features more accurately, such as their overall appearance, personality traits, and visual identity. In contrast, the baseline prompts tend to focus on superficial elements like color, local texture similarity, or compositional overlap, which can lead to misclassification. This comparison demonstrates the effectiveness of our method in optimizing prompts for copyright detection, enabling better conceptual understanding and reducing the likelihood of false judgments. Overall, these examples highlight the improved accuracy and robustness of our approach in real-world content moderation scenarios.

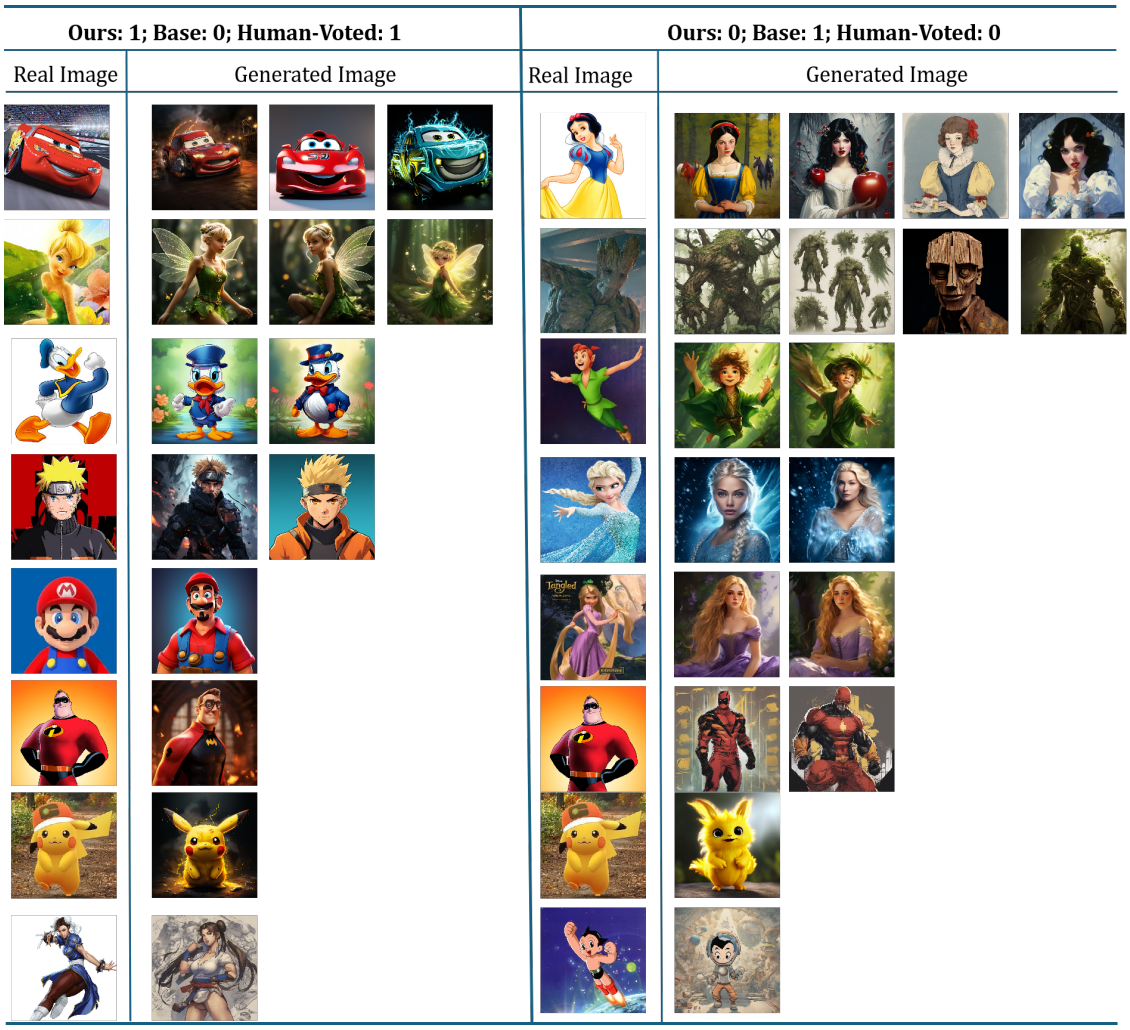


Figure 19. Comparative Analysis of Prompt-Based Detection Results with Reference Images (Infrigement = 1, non-infrigement = 0), In each block, the first row displays the real images collected from the web and the other row shows the AI-generated images of the same character.