# TGTS: Token Group Tree Sampling for Efficient Multi-Variants Sampling In Language Model

**Shih-Ying Yeh  Chung-Po Chang  Hsin-Ying Lee  Kuan-Ting Kuo**

National Tsing Hua University

## Abstract

*Large Language Models (LLMs) increasingly require efficient generation of multiple diverse outputs from single prompt for tasks like prompt engineering optimization and synthetic dataset generation. While existing methods can generate multiple sequences, their token-level operations limit efficient exploration of the solution space. We present TGTS (Token Group Tree Sampling), a novel programmable framework that generates multiple variants through group-level token sampling and tree-based search strategies. By operating at the token group level, TGTS achieves both improved sampling efficiency and better control over sequence diversity while maintaining semantic coherence. Through our novel random-walk MCTS approach, TGTS provides flexible control over the exploration-exploitation trade-off, enabling efficient generation of high-quality diverse outputs. Our experimental results demonstrate significant improvements in computational efficiency while maintaining or exceeding the quality and diversity metrics of traditional sampling methods. The framework's programmable nature and theoretical foundations make it adaptable to various LLM applications requiring multiple diverse outputs.*

## 1. Introduction

Recent advancements in Large Language Models (LLMs) have demonstrated remarkable capabilities across various applications. Although significant research has focused on optimizing model architectures and inference efficiencies, these optimizations primarily target single-request performance or average throughput metrics.

Emerging applications increasingly require LLMs to generate multiple diverse outputs from a single prompt, such as for automatic prompt engineering and synthetic dataset generation.[10, 17, 20] Traditional sampling methods, while capable of generating multiple sequences simultaneously, remain limited by token-level sampling, creating deep but narrow sampling trees that make effective branch expansion challenging.

To address these limitations, we introduce TGTS (Token Group Tree Sampling), a novel and programmable framework for LLM sampling designed to generate multiple variants concurrently. TGTS operates at the token group level rather than individual tokens, enabling more efficient exploration of the solution space while maintaining sequence coherence.

As a practical demonstration of our framework's capabilities, we apply TGTS to the task of prompt generation for text-to-image models, evaluating both textual quality and downstream image generation performance. Our experimental results demonstrate the framework's effectiveness in generating diverse, high-quality outputs efficiently.

Our work makes the following key contributions:

1. We introduce a universal, programmable sampling method designed to work with any decoder language model, providing significant efficiency improvements over traditional sampling methods.
2. We present a novel approach to managing the exploration-exploitation trade-off through token group-based sampling, offering improved control over sequence diversity while maintaining semantic coherence.
3. We demonstrate through comprehensive evaluation that our framework achieves superior efficiency in generating diverse outputs while maintaining high quality, making it suitable for various applications requiring multiple variations from a single prompt.

## 2. Related Works

### 2.1. LM sampling

#### 2.1.1. Sampling

**Temperature** Temperature sampling is a fundamental technique that controls generation randomness by adjusting the softmax function output. A temper-

ature of 1 maintains the original probability distribution, while higher values increase randomness and lower values enhance determinism [8].

**Top-k and Top-p** Top-k sampling limits selection to the top $k$ most probable tokens, balancing diversity and coherence [4]. Top-p sampling (nucleus sampling) dynamically adjusts the candidate set size by selecting tokens whose cumulative probabilities exceed threshold $p$ [8].

**Min-p** Min-p sampling enhances precision by excluding tokens below a probability threshold $p$. As demonstrated in [12], this approach effectively balances creativity with semantic consistency, particularly under high-temperature settings.

#### 2.1.2. Beam Search

Beam Search retains the top $k$ candidates at each step, providing computational efficiency while ensuring quality. However, its constrained search space can lead to suboptimal outputs. **Diverse Beam Search** [19] addresses this through diversity penalties between groups, while **Stochastic Beam Search** [9] incorporates controlled randomness via the Gumbel-Top-k trick.

### 2.2. Tree Search

Monte Carlo Tree Search (MCTS) excels in large state spaces through its four-step process: selection, expansion, simulation, and backpropagation. The selection uses the UCT algorithm:

$$UCT(node_i) = \frac{w_i}{n_i} + c\sqrt{\frac{\ln N}{n_i}}$$

This iterative approach efficiently explores promising paths while maintaining balance between exploration and exploitation.

### 2.3. Other Approaches

Recent works have explored various complementary techniques. [7] introduced the L2W framework for enhanced long-text generation. [15] demonstrated the effectiveness of combining unsupervised pre-training with supervised fine-tuning. [1] developed Mirostat for direct perplexity control. In the context of neural evaluation functions, [18] and [2] pioneered applications in game state assessment.

## 3. Tree Search Sampling

Traditional language model sampling approaches can be broadly categorized into two paradigms: probability-based sampling and tree-based search methods. We present a novel framework that bridges these approaches while introducing a more efficient hierarchical structure.

### 3.1. Traditional Sampling Paradigms

In probability-based sampling, for a given sequence of tokens $s = (t_1, ..., t_n)$, the next token $t_{n+1}$ is selected based on the probability distribution:

$$P(t_{n+1}|s) = \text{softmax}(f_\theta(s)/\tau) \tag{1}$$

where $f_\theta$ represents the language model with parameters $\theta$ and $\tau$ is the temperature parameter.

Tree-based approaches view sampling as traversing an implicit tree structure:

$$\begin{aligned} &T = (V, E), \\ &V = \{t_i\}, \\ &E = \{(t_i, t_j)|j \text{ follows } i\} \end{aligned} \tag{2}$$

### 3.2. Token Group Tree Framework

We introduce Token Group Tree Sampling (TGTS), which defines token groups $G_i$ containing multiple related tokens:

$$G_i = \{t_{i,1}, t_{i,2}, ..., t_{i,k_i}\} \tag{3}$$

where $k_i$ is the variable size of group $i$. The group tree structure is:

$$\begin{aligned} &T_G = (V_G, E_G), \\ &V_G = \{G_i\}, \\ &E_G = \{(G_i, G_j)|G_j \text{ follows } G_i\} \end{aligned} \tag{4}$$

This reformulation offers several advantages:

- **Reduced Search Depth:** Operating at the group level reduces effective tree depth
- **Flexible Grouping:** Variable group sizes enable adaptive granularity
- **Enhanced Exploration:** Group structure enables efficient solution space exploration

### 3.3. Search Strategy

In our framework, the score of a token group $G_i$ is defined as the geometric mean of token probabilities:

$$\text{score}(G_i) = \left(\prod_{t \in G_i} P(t|s_t)\right)^{1/|G_i|} \tag{5}$$

where $s_t$ is the sequence context for token $t$, and $P(t|s_t)$ is the model's predicted probability.

The token group structure enables efficient caching:

$$\text{cache}(G_i) = \{(s_j, p_j) \mid s_j \text{ is a suffix of } G_i\} \tag{6}$$

This framework serves as the foundation for our specific algorithmic implementations, detailed in Section 4.

## 4. TGTS algorithms

In this section we introduced two proposed algorithm for TGTS.

### 4.1. cg-MCTS

Unlike traditional MCTS applications, our language generation task faces unique challenges where simulation requires expensive model invocations. The cg-MCTS (Cached Greedy MCTS) algorithm addresses this through two key mechanisms: caching and progressive widening with greedy selection.

**State and Action Space** For a node $n$ in the search tree, we define:

- State $s_n$: The sequence of token groups that lead to node $n$
- Action space $A(s_n)$: All possible next token groups from state $s_n$
- Result $\mathcal{R}(n)$: The complete sequence generated from node $n$
- Cache $\mathcal{C}(n)$: Set of cached simulation results where

$$\mathcal{C}(n) = \{(s_i, p_i) | s_i \text{ is a suffix starting from } n\} \tag{7}$$

**Caching Mechanism** The caching system maintains simulation results to reduce model invocations:

$$\mathcal{R}(n) = \begin{cases} \mathcal{C}(n) & \text{if } n \text{ is cached} \\ \text{simulate}(s_n) & \text{otherwise} \end{cases} \tag{8}$$

**Progressive Widening** The algorithm employs progressive widening to control tree expansion, where the number of children $N_{Children}(n)$ is determined by:

$$N_{Children}(n) = \lceil k \cdot N(n)^{\alpha(d)} \rceil \tag{9}$$

where:

- $N(n)$ is the visit count of node $n$
- $k = \sqrt{2}$ is the base expansion factor
- $\alpha(d) = 0.5^{1+\sqrt{2}^d}$ is the depth-dependent expansion rate
- $d$ is the depth of node $n$

**Selection Strategy** The selection combines UCT scoring with greedy evaluation:

$$UCT(n) = \frac{Q(n)}{N(n)} + c\sqrt{\frac{\ln N(p(n))}{N(n)}} \tag{10}$$

where $Q(n)$ is the cumulative score and $p(n)$ is the parent node.

**Algorithm 1** cg-MCTS Generation

**Input:** root node $r$, sample count $K$
**Output:** generated sequences $S$
**function** SIMULATE(node $n$)
    $sequence \leftarrow nucleusSample(s_n)$
    $\mathcal{C}(n) \leftarrow sequence$
    **return** $sequence$
**end function**

**function** EXPAND(node $n$)
    $\alpha \leftarrow 0.5^{1+\sqrt{2}^{depth(n)}}$
    $N_{max} \leftarrow \lceil \sqrt{2} \cdot N(n)^{\alpha} \rceil$
    $candidates \leftarrow \text{GetTopK}(A(s_n), N_{max})$
    **for** $group \in candidates$ **do**
        $child \leftarrow \text{createNode}(n, group)$
        $sequence \leftarrow \text{Simulate}(child)$
        $children(n) \leftarrow children(n) \cup \{child\}$
        $backpropagate(child, score(sequence))$
    **end for**
    **return** candidates[0]
**end function**

**for** $i = 1$ to $K$ **do**
    $n \leftarrow r$
    **while** not terminal **do**
        **if** $n \in \mathcal{C}$ **then**
            **break**
        **end if**
        **if** $|children(n)| < N_{Children}(n)$ **then**
            $n \leftarrow \text{Expand}(n)$
        **else**
            $n \leftarrow \arg\max_{c \in children(n)} UCT(c)$
        **end if**
    **end while**
    $S \leftarrow S \cup \{\mathcal{R}(n)\}$
**end for**
**return** $S$

### 4.2. rw-MCTS

We present random-walk MCTS (rw-MCTS), an enhanced variant of MCTS that addresses the efficiency and flexibility limitations of cg-MCTS while preserving the statistical properties of nucleus sampling through a novel stochastic tree search approach.

**State and Action Space** For a search tree node $n$, we define:

- State $s_n$: The sequence of token groups leading to node $n$
- Action space $A(s_n)$: Set of possible next token groups
- Result $\mathcal{R}(n)$: The complete sequence generated from node $n$

**Self Virtual Node** Traditional MCTS algorithms face challenges with tree width expansion in large state spaces. To address this, rw-MCTS introduces the concept of self virtual nodes, where a node can select itself for expansion. The selection candidate set for node $n$ is defined as:

$$C(n) = \{n\} \cup children(n) \tag{11}$$

The UCT scoring function is modified to handle self-selection:

$$UCT(c) = \begin{cases} UCT_{self}(n) & \text{if } c = n \\ UCT_{std}(c) & \text{otherwise} \end{cases} \tag{12}$$

where the self UCT score is computed as:

$$UCT_{self}(n) = \frac{Q(n)}{N_{child}(n)} + c\sqrt{\frac{\ln(N(n))}{N_{child}(n)}} \tag{13}$$

Here:

- $N_{child}(n)$: Number of expanded children
- $Q(n)$: Cumulative quality score
- $N(n)$: Visit count
- $c$: Exploration constant

**Stochastic Selection Strategy** To maintain nucleus sampling properties, rw-MCTS employs a softmax-based selection mechanism:

$$P(c) = \frac{\exp(UCT(c)/\tau)}{\sum_{c' \in C(n)} \exp(UCT(c')/\tau)} \tag{14}$$

where $\tau$ is a temperature parameter controlling selection randomness. This creates an adaptive sampling mechanism that balances exploration and exploitation.

**Sequence Generation Process** For a selected node $n$, the generation process follows:

$$g_{new} = \begin{cases} \text{nucleusSample}(s_n) & \text{if self-selected} \\ g_c & \text{if child } c \text{ selected} \end{cases} \tag{15}$$

where $g_{new}$ is the next token group and $g_c$ is the token group associated with child $c$.

**Value Backpropagation** The value update rule for node $n$ after expansion is:

$$Q(n) \leftarrow Q(n) + score(g_{new}) \tag{16}$$

$$N(n) \leftarrow N(n) + 1 \tag{17}$$

**Algorithm 2** rw-MCTS Generation

**Input:** root node $r$, sample count $K$, temperature $\tau$
**Output:** generated sequences $S$

**function** SELECTNODE(node $n$)
    $C \leftarrow \{n\} \cup children(n)$
    $scores \leftarrow [UCT(c) \text{ for } c \in C]$
    $p \leftarrow softmax(scores/\tau)$
    **return** $sample(C, p)$
**end function**

**function** EXPANDNODE(node $n$)
    $group \leftarrow nucleusSample(s_n)$
    $child \leftarrow createNode(group)$
    $result \leftarrow nucleusSample(s_{child})$
    $\mathcal{R}(child) \leftarrow result$
    $children(n) \leftarrow children(n) \cup \{child\}$
    $backpropagate(child, score(group))$
    **return** $child$
**end function**

**for** $i = 1$ to $K$ **do**
    $n \leftarrow r$
    **while** not terminal **do**
        $selected \leftarrow$ SelectNode($n$)
        **if** $selected = n$ **then**
            $n \leftarrow$ ExpandNode($n$)
        **else**
            $n \leftarrow selected$
        **end if**
    **end while**
    $S \leftarrow S \cup \{\mathcal{R}(n)\}$
**end for**
**return** $S$

This approach provides several key advantages:

- Efficient batch generation requiring exactly $K$ iterations for $K$ samples
- Adaptive tree expansion through stochastic selection
- Preservation of nucleus sampling properties via softmax selection
- Memory efficiency through single result maintenance per non-root node

## 5. Experimental Setup

To evaluate the effectiveness of our proposed sampling methods, we conducted comprehensive experiments using the TIPO-100M[1] model as our base generation model. TIPO-100M is specifically designed for generating prompts for text-to-image models, making it particularly suitable for assessing both textual quality and downstream image generation performance.[20]

### 5.1. Model Configuration

For our experimental implementation, we utilized TIPO-100M as the primary text generation model. For the downstream image generation task, we employed Kohaku-XL-Zeta[2], a specialized fine-tuned variant of SDXL cite[14] optimized for anime-style image generation. This choice aligns with TIPO-100M's training objective, which focuses on tag-based anime-style model prompting.

### 5.2. Evaluation Metrics

We employed multiple evaluation metrics to assess different aspects of the generated outputs:

#### 5.2.1. Computational Efficiency

To measure the computational efficiency of our sampling methods, we tracked the Average Number of Function Evaluations (NFE) per output. This metric provides a direct measure of the computational resources required by each sampling strategy.

#### 5.2.2. Distribution Similarity

We utilized the Frechet Distance[6] to quantify the distribution similarity between our sampling methods and nucleus sampling (used as reference). This measurement was performed in two domains:

- **Text Domain:** Using embeddings generated by the JINA-embedding-v3 model [16]
- **Image Domain:** Using features extracted by DinoV2 ViT-L [13]

#### 5.2.3. Output Diversity

To evaluate the diversity of generated outputs, we employed the Vendi Score [5], which computes a specialized entropy measure on similarity matrices. We calculated this metric for both:

- Generated text prompts
- Resulting generated images

#### 5.2.4. Generation Quality

For assessing the quality of downstream image generation, we implemented two specialized metrics:

- **AI Corrupt Score[11]:** Measures the technical quality and coherence of generated images
- **Aesthetic Score [3]:** Evaluates the artistic and visual quality of the generated outputs

### 5.3. Experimental Protocol

For each sampling method, we generated 1,024 prompts and subsequently produced one image per prompt using the Kohaku-XL-Zeta model. This substantial sample size ensures robust statistical analysis while maintaining computational feasibility. The evaluation was conducted under consistent conditions across all sampling methods to ensure fair comparison.

## 6. Evaluation

We evaluate our proposed methods against nucleus sampling (reference) and diverse beam search baselines across metrics of efficiency, quality, and diversity, as shown in Table 1.

### 6.1. Computational Efficiency

Both cg-MCTS and rw-MCTS demonstrate substantial improvements in computational efficiency compared to nucleus sampling, as measured by Average NFE. The rw-MCTS variant with high exploration factor ($c = 3.0$) achieves the best efficiency, reducing function evaluations by more than half compared to the reference method. This efficiency gain stems from rw-MCTS's adaptive tree expansion mechanism, where higher exploration factors encourage more focused search patterns.

### 6.2. Output Quality and Distribution

Quality metrics (Aesthetic and AI Corrupt scores) indicate that both MCTS variants maintain generation quality comparable to nucleus sampling while significantly outperforming diverse beam search. The Frechet Distance (FD) scores in both text and image domains demonstrate that our methods better preserve the distributional characteristics of the reference sampling method. Notably, rw-MCTS maintains consistently strong quality metrics across different exploration factors, suggesting robustness to parameter changes.

### 6.3. Diversity Analysis

The Vendi scores reveal an interesting trade-off between diversity and exploration in rw-MCTS. As the exploration factor increases, we observe:

- Improved computational efficiency through more selective node expansion
- Moderate decrease in output diversity, particularly at higher $c$ values

[1] https://huggingface.co/KBlueLeaf/TIPO-100M
[2] https://huggingface.co/KBlueLeaf/Kohaku-XL-Zeta

| Metric | Nucleus(ref) | Diverse Beam | cg-MCTS(ours) | | | rw-MCTS(ours) | | |
|---|---|---|---|---|---|---|---|---|
| | | | c=0.5 | c=1.0 | c=3.0 | c=0.5 | c=1.0 | c=3.0 |
| Average NFE ↓ | 95.303 | <u>43.031</u> | 52.182 | 54.556 | 56.986 | 64.052 | 56.311 | **39.915** |
| FD (Text) ↓ | 0.0077 | 0.3038 | 0.0909 | 0.0384 | 0.0470 | **0.0317** | <u>0.0326</u> | 0.0719 |
| FD (Image) ↓ | 0.1199 | 0.8189 | 0.2074 | **0.1622** | 0.1808 | <u>0.1635</u> | 0.1723 | 0.1796 |
| Vendi (Text) ↑ | 3.1192 | 1.4895 | 2.8504 | 2.9534 | **3.1591** | <u>3.1294</u> | 3.0855 | 2.8293 |
| Vendi (Image) ↑ | 13.440 | 4.4730 | 12.725 | 11.557 | **13.963** | <u>13.145</u> | 13.078 | 11.291 |
| Aesthetic ↑ | 6.2842 | 5.8194 | **6.4726** | 6.2687 | 6.3014 | 6.2999 | 6.3321 | <u>6.3338</u> |
| AI Corrupt ↑ | 0.9163 | 0.5746 | 0.8871 | 0.9122 | <u>0.9188</u> | 0.9155 | **0.9320** | 0.8961 |

Table 1. Comparison of sampling methods across different evaluation metrics. Lower scores (↓) are better for Average NFE and Frechet Distance (FD). Higher scores (↑) are better for Vendi, Aesthetic, and AI Corrupt scores. Best scores for each metric are in bold.

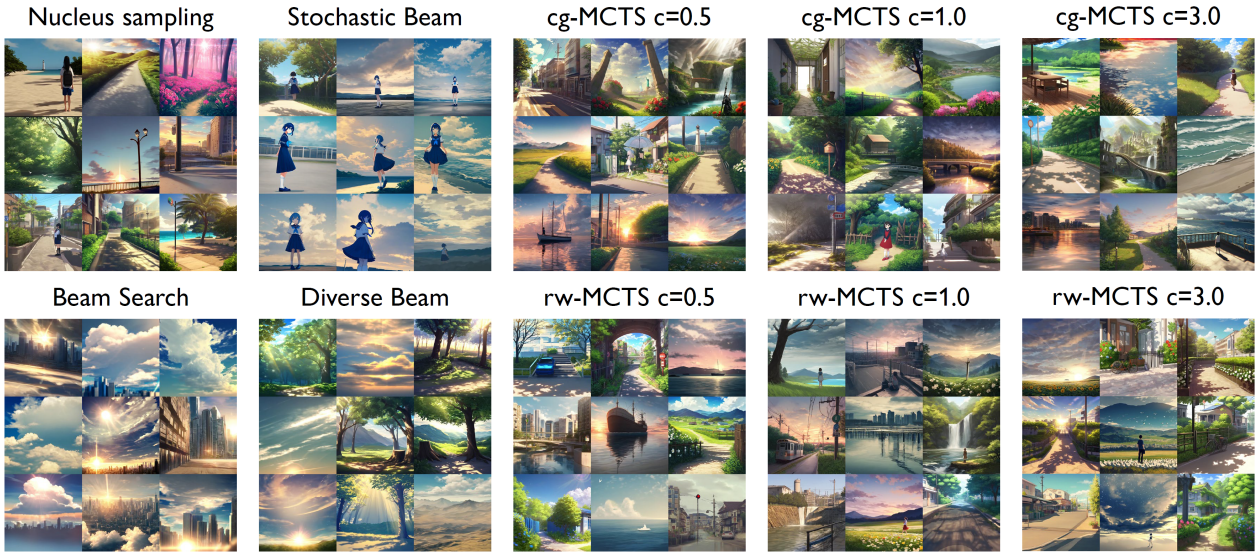


Figure 1. Generated images from generated prompt by each methods.

- Slight increase in distribution divergence from the reference

This trade-off provides users with flexible control over the efficiency-diversity balance based on application requirements.

### 6.4. Visual Analysis

Figure 1 demonstrates the qualitative differences between methods:

- Traditional beam search approaches produce thematically similar outputs
- Diverse beam search shows limited improvement in content variation
- Both MCTS variants generate substantially different content across images, better matching the diversity characteristics of nucleus sampling

### 6.5. Method Comparison

While both MCTS variants demonstrate strong performance, rw-MCTS offers superior control over the exploration-exploitation trade-off through its $c$ parameter. At moderate exploration settings ($c = 1.0$), rw-MCTS achieves an optimal balance between efficiency, quality, and diversity. While cg-MCTS shows consistent performance across metrics, its fixed tree-width limitation makes it less adaptable to varying generation requirements. The experimental results suggest that rw-MCTS's flexible exploration mechanism provides a more robust foundation for diverse text generation while maintaining the desirable properties of nucleus sampling.

## 7. Conclusion

In this work, we presented TGTS (Token Group Tree Sampling), a novel sampling framework designed for efficient multi-variant generation in language models. Our framework demonstrates significant advantages in achieving "Single Input Multi Generation" while maintaining output quality and diversity comparable to traditional nucleus sampling methods, with reduced computational overhead as measured by NFE (Number of Function Evaluations) per prompt.

The key contributions and findings of our work can be summarized as follows:

- We introduced a tree-based sampling strategy that effectively balances the exploration-exploitation trade-off through token group-level operations, providing improved control over the generation process while maintaining semantic coherence.
- Our experimental results demonstrate that TGTS achieves comparable Frechet Distance scores to nucleus sampling in both text and image domains, indicating the framework's ability to maintain distribution similarity while reducing computational costs.
- The framework's effectiveness in downstream tasks, particularly in text-to-image generation, is evidenced by strong performance across AI Corrupt Score and Aesthetic Score metrics, validating its practical utility.
- Through our implementation of cg-MCTS and rw-MCTS variants, we showed that tree-based sampling approaches can effectively navigate the vast solution space of language model outputs while maintaining generation quality.

These results suggest that TGTS represents a promising direction for efficient multi-variant sampling in language models. The framework's capability to generate diverse, high-quality outputs while reducing computational overhead makes it particularly suitable for applications requiring multiple generations from single prompts, such as prompt engineering optimization and synthetic dataset generation.

Future work could explore additional optimization strategies for the tree search process, investigation of alternative grouping mechanisms for tokens, and adaptation of the framework to other types of generative models. The programmable nature of our framework also opens possibilities for task-specific customization and integration with other sampling techniques.

## 8. Data and Code Availability

The complete implementation of our framework, including the source code for both cg-MCTS and rw-MCTS algorithms, is publicly available in our open-source repository. The codebase includes:

- Core sampling algorithm implementations (cg-MCTS and rw-MCTS)
- Comprehensive evaluation metrics suite
- Experimental execution scripts and configurations
- Documentation and usage examples

All code is available under an Apache-2.0 license at https://github.com/KohakuBlueleaf/KGen. The repository is structured to facilitate both research reproduction and practical applications, with the following key components:

- Sampling implementations: `/src/kgen/sampling/`
- Evaluation metrics: `/src/kgen_exp/metrics/`
- Experiment scripts: `/scripts/exp/tgts/`

## 9. Author Contribution Statements

Table 2 presents a detailed breakdown of author contributions across four key areas of this work. The contribution levels are indicated by the number of 'X' marks, where more marks represent greater involvement in that particular aspect. The contribution areas are defined as follows:

- **Algorithm:** Conceptualization, theoretical development, and mathematical formulation of the TGTS framework and its variants
- **Development:** Implementation of algorithms, creation of the codebase, and technical infrastructure development
- **Experiments:** Design and execution of experiments, data collection, and analysis of results
- **Writing:** Paper draft preparation, systematic literature review, technical content development, and manuscript revisions including abstract, methodology, experimental results, and conclusions

| | S.-Y. Yeh | C.-P. Chang | H.-Y. Lee | K.-T. Kuo |
|---|---|---|---|---|
| Algorithm | XXX | XX | X | |
| Development | XXXX | XX | | |
| Experiments | XXXXXX | | | |
| Writing | XXX | X | X | X |

Table 2. Author contributions. The number of X's indicates the level of contribution in each area, with more X's representing greater involvement.

## References

[1] Sourya Basu, Govardana Sachitanandam Ramachandran, Nitish Shirish Keskar, and Lav R. Varshney. Mirostat: A neural text decoding algorithm that directly controls perplexity, 2021. URL https://arxiv.org/abs/2007.14966. 2

[2] 那須 悠. 高速に差分計算可能なニューラルネットワーク型将棋評価関数, 2018. URL https://www.apply.computer-shogi.org/wcsc28/appeal/the_end_of_genesis_T.N.K.evolution_turbo_type_D/nnue.pdf. 2

[3] discus0434. discus0434/aesthetic-predictor-v2-5: SigLIP-based Aesthetic Score Predictor. https://github.com/discus0434/aesthetic-predictor-v2-5, 2024. 5

[4] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation, 2018. URL https://arxiv.org/abs/1805.04833. 2

[5] Dan Friedman and Adji Bousso Dieng. The vendi score: A diversity evaluation metric for machine learning, 2023. URL https://arxiv.org/abs/2210.02410. 5

[6] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf. 5

[7] Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. Learning to write with cooperative discriminators, 2018. URL https://arxiv.org/abs/1805.06087. 2

[8] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration, 2020. URL https://arxiv.org/abs/1904.09751. 2

[9] Wouter Kool, Herke van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. *CoRR*, abs/1903.06059, 2019. URL http://arxiv.org/abs/1903.06059. 2

[10] Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. On llms-driven synthetic data generation, curation, and evaluation: A survey, 2024. URL https://arxiv.org/abs/2406.15126. 1

[11] narugo1992. Ai-corrupt score for anime images. https://huggingface.co/deepghs/ai_image_corrupted, 2023. 5

[12] Minh Nguyen, Andrew Baker, Clement Neo, Allen Roush, Andreas Kirsch, and Ravid Shwartz-Ziv. Turning up the heat: Min-p sampling for creative and coherent llm outputs, 2024. URL https://arxiv.org/abs/2407.01082. 2

[13] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023. 5

[14] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=di52zR8xgf. 5

[15] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training, 2018. URL https://api.semanticscholar.org/CorpusID:49313245. 2

[16] Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, Andreas Koukounas, Nan Wang, and Han Xiao. jina-embeddings-v3: Multilingual embeddings with task lora, 2024. URL https://arxiv.org/abs/2409.10173. 5

[17] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023. 1

[18] S. Thrun. Learning to play the game of chess. In G. Tesauro, D. Touretzky, and T. Leen (eds.), *Advances in Neural Information Processing Systems (NIPS) 7*, Cambridge, MA, 1995. MIT Press. 2

[19] Ashwin K Vijayakumar, Michael Cogswell, Ramprasaath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search: Decoding diverse solutions from neural sequence models, 2017. URL https://openreview.net/forum?id=HJV1zP5xg. 2

[20] Shih-Ying Yeh, Sang-Hyun Park, Giyeong Oh, Min Song, and Youngjae Yu. Tipo: Text to image with text presampling for prompt optimization, 2024. URL https://arxiv.org/abs/2411.08127. 1, 5