Evolutionary Transformer Architecture Search via LLM-Guided Coarse-to-Fine Transformations

Lei Liu
College of Computer Science
Sichuan University
Chengdu, China
liulei.cx@gmail.com

Gary G. Yen*
College of Computer Science
Sichuan University
Chengdu, China
gyen@okstate.edu

Chenyang Li
College of Computer Science
Sichuan University
Chengdu, China
lcy.yang@foxmail.com

Zhenan He
College of Computer Science
Sichuan University
Chengdu, China
zhenan@scu.edu.cn

***Abstract*—Practical Transformer architecture search requires navigating a highly discrete and strongly coupled design space under limited computational budgets, while achieving an effective trade-off between model performance and complexity. However, in the absence of explicit search directionality, existing methods often generate a large proportion of infeasible candidates under resource constraints, thereby substantially reducing search efficiency. To address this issue, we propose CF-TAS, an LLM-guided coarse-to-fine framework for constrained Transformer architecture search. Built upon a multi-objective evolutionary backbone, CF-TAS employs a fine-tuned LLM to generate direction-aware architecture edit scripts, rather than directly synthesizing complete architectures. Specifically, Directional Architecture Transformation (DAT) generates edit scripts for a small set of seed architectures during initialization to improve the feasibility of the initial population, and is further applied to representative parents, including knee and boundary solutions, during iterative search to prioritize exploration of critical regions on the Pareto front. To handle infeasible candidates, Directional Local Refinement (DLR) performs lightweight local corrections that reduce constraint violation while largely preserving the original architecture. The LLM-guided branch operates in parallel with conventional evolutionary search, and all candidates are evaluated under a unified constraint-handling and environmental selection mechanism. Experiments on ImageNet-1K under two parameter budgets show that CF-TAS achieves performance comparable to, or better than, representative state-of-the-art baselines. These results demonstrate the effectiveness of combining evolutionary exploration with LLM-guided directional transformations for constrained Transformer architecture search.**

Keywords—transformer architecture search, large language model, architecture transformation, constrained neural architecture search, multi-objective evolutionary algorithm

I. INTRODUCTION

Transformer architectures have become a dominant modeling paradigm in computer vision, natural language processing, and multimodal learning [1] [2] [3] [4]. With the continuous growth in model scale and task complexity, Transformer design has evolved from relatively uniform and regular configurations toward increasingly heterogeneous architectures [5] [6]. Beyond coarse-grained hyperparameters such as depth and embedding dimension, performance depends on layer-wise MLP expansion ratios, per-layer head allocation, and their coupled interactions [7] [8]. As a result, the Transformer architecture space exhibits strong discreteness, tight structural coupling, and highly uneven performance distributions across different regions [9] [10]. Under such conditions, relying solely on heuristic, manual design has become insufficient for consistently achieving stable trade-offs between performance and efficiency [11] [12], thereby motivating the growing demand for automated architecture search methods.

Neural architecture search (NAS) has become an effective paradigm for systematically exploring complex architectural spaces [13]. Among existing NAS methods, evolutionary search [14] [15] [16] is particularly attractive because it is gradient-free, naturally handles discrete design variables, and can flexibly incorporate resource constraints. By iteratively generating candidates through variation operators and retaining diversity via population-based selection, it provides a robust mechanism for black-box architecture optimization. Nevertheless, its limitations become pronounced in Transformer architecture search. Specifically, evolutionary variation is fundamentally stochastic and therefore lacks explicit search directionality [17], limiting its ability to exploit the structural relationships among architecture, performance, and resource usage. Moreover, because the Transformer search space is highly discrete and strongly coupled [9] [18], even small random perturbations may lead to substantial changes in parameter count and computational cost, frequently resulting in infeasible architectures. Consequently, a large fraction of the search budget is consumed by low-quality or infeasible candidates, which ultimately degrades search efficiency.

Recent advances in large language models (LLMs) have demonstrated remarkable capabilities in reasoning, rule induction, and decision-making under complex conditions [19] [20]. Compared with stochastic variation, these capabilities enable LLMs to infer latent design regularities from contextual information and to provide condition-aware guidance for architectural modification [21] [22]. This suggests a promising direction for architecture search: if candidate generation can be guided by more directional architectural adjustments rather than relying solely on random perturbations, the search process may be steered toward more promising regions of the design space. Nevertheless, the way in which LLMs are incorporated is crucial. Directly using an LLM to generate complete architectures, or naively treating it as an "intelligent" variation operator [23] [24], often fails to produce stable improvements. Prior studies and empirical evidence suggest that such strategies may generate outputs that do not reliably conform to a predefined search space, thereby increasing implementation complexity. Moreover, excessive reliance on LLMs may reduce search diversity by concentrating candidates around a narrow set of architectural patterns, while frequent LLM queries introduce nontrivial computational overhead. Taken together, these observations indicate that the central challenge is not whether to use LLMs

This work was supported by the Research Fund for International Senior Scientists from the National Natural Science Foundation of China under Grant W2531051, the Fundamental Research Funds for the Central Universities under Grant YJ202501, the National Natural Science Foundation of China under Grant 62422604, and the Sichuan University-Suining Municipal Strategic Cooperation Science and Technology Project under Grant 2024CDSN-05.

*Corresponding author: Gary G. Yen (gyen@okstate.edu)

in architecture search, but how to integrate them in a controlled and principled manner.

Motivated by the above analysis, we aim to retain the robustness and diversity of conventional search while leveraging LLM-based reasoning to introduce controllable directionality into Transformer architecture search, thereby improving the generation of feasible candidates under resource constraints. To achieve this goal, we propose CF-TAS, an LLM-guided coarse-to-fine architecture transformation framework. In CF-TAS, the LLM is not used to synthesize complete architectures. Instead, its role is strictly limited to generating architecture edit scripts. This restriction ensures that candidate generation remains confined to the predefined search space and a bounded edit budget, improving both validity and controllability. CF-TAS consists of two complementary operators. **Directional Architecture Transformation (DAT)** generates a small number of edit scripts for representative architectures, guiding candidate generation toward reduced constraint violation and promising regions of the search space. DAT is used in both population initialization and iterative search to improve initial feasibility and focus exploration on high-value regions. **Directional Local Refinement (DLR)** performs lightweight local correction only on infeasible candidates. Through minimally perturbative edits, it reduces constraint violation and restores feasibility while preserving the original architectural pattern as much as possible, thereby limiting performance instability. Together, DAT and DLR form a coarse-to-fine transformation process under resource constraints. Importantly, the proposed LLM-guided transformations complement rather than replace conventional search operators. The latter maintain broad exploration and population diversity, whereas DAT and DLR provide directional guidance and constraint-aware refinement. The contributions of this work are threefold.

1) We propose a unified framework for LLM-guided Transformer architecture search, in which the LLM is explicitly constrained to serve as a direction-aware architecture transformation mechanism. This formulation enables stable and controllable candidate generation without sacrificing search diversity.
2) We design a Directional Architecture Transformation module that uses LLM-based conditional generation to perform directional architectural edits. DAT alleviates inefficient exploration induced by stochastic perturbations and steers the search toward promising feasible regions.
3) We propose a Directional Local Refinement module for repairing infeasible candidates through minimal local modifications. By preserving architectural inheritance and search stability, DLR reduces the overhead of candidate repair and evaluation under resource constraints.

The rest of this paper is organized as follows. Section II introduces the Transformer architecture space, the constrained multi-objective search formulation, and the fine-tuned LLM. Section III details the proposed CF-TAS framework and its two core modules. Section IV presents the experimental results and analysis. Section V concludes the paper with a discussion of limitations and future directions.

II. PRELIMINARY

Transformer Architecture Space and Representation. We consider a layer-configurable Transformer search space following prior works [7] [3] [25]. A candidate architecture is represented as

$$\mathcal{A} = (D, E, \boldsymbol{r}, \boldsymbol{h}) \tag{1}$$

where D denotes the network depth (number of blocks), E is the embedding dimension, $\boldsymbol{r} = [r_1, \dots, r_D]$ specifies the layer-wise MLP expansion ratios, and $\boldsymbol{h} = [h_1, \dots, h_D]$ specifies the layer-wise numbers of attention heads [7].

The search space Ω is determined by the discrete candidate values of the above fields and their composition rules. Specifically, D and E are selected from predefined candidate sets. For each layer $i \in [1, \dots, D]$, r_i and h_i are chosen from their corresponding discrete candidate sets. This construction ensures that every candidate architecture lies within the predefined discrete architecture space, and it enables unified architecture representations for candidate generation and architecture transformations in subsequent methods. Note that changing D alters the lengths of the layer-wise sequences $\boldsymbol{r}$ and $\boldsymbol{h}$, which affects executability of the representation. Accordingly, when decreasing D we truncate the sequences, when increasing D we complete the newly added layer configurations using deterministic rules, so that the representation remains consistent with the executable architecture.

Constrained Multi-Objective Search. Constrained multi-objective architecture search aims to optimize multiple objectives over a search space Ω while satisfying prescribed constraints [26]. In this work, we seek a trade-off between maximizing a performance proxy and minimizing computational cost, subject to a parameter-count constraint on candidate architectures. Specifically, for a candidate architecture $\mathcal{A} \in \Omega$, let $s(\mathcal{A})$ denote its performance proxy score on the target task. In our implementation, $s(\mathcal{A})$ is computed using the DSS-indicator from literature [27], a validated proxy that correlates with accuracy. Let $f(\mathcal{A})$ denote the computational cost (FLOPs), and let $p(\mathcal{A})$ denote the number of parameters.

Because exact constraint satisfaction is difficult during search and practical deployments often tolerate small fluctuations, we impose a range of constraints centered at a target parameter budget p with a $\pm 10\%$ tolerance, yielding a feasible range $[L, U] = [0.9p, 1.1p]$. To handle feasible and infeasible candidates in a unified manner, we define the constraint violation degree as

$$\phi(\mathcal{A}) = \max\{0, p(\mathcal{A}) - U, L - p(\mathcal{A})\} \tag{2}$$

where $\phi(\mathcal{A}) = 0$ indicates that $\mathcal{A}$ satisfies the constraint. The search problem is then formulated as a constrained bi-objective optimization over the discrete space:

$$\min_{\mathcal{A} \in \Omega} \big(-s(\mathcal{A}), f(\mathcal{A})\big) \text{ s.t. } \phi(\mathcal{A}) = 0 \tag{3}$$

where Ω is the architecture space determined by discrete variables including depth, embedding dimension, and layer-wise MLP expansion ratios and attention head numbers, together with their composition rules.

For candidate comparison, we adopt a feasibility-first principle [17] [28]. Given two architectures $\mathcal{A}_1$ and $\mathcal{A}_2$, the one with a smaller violation degree is preferred when $\phi(\mathcal{A}_1) \neq \phi(\mathcal{A}_2)$. When both are feasible ($\phi(\mathcal{A}_1) = \phi(\mathcal{A}_2) = 0$), they are compared according to Pareto dominance on the objective vector $(-s(\cdot), f(\cdot))$. Under this criterion, environmental selection constructs the next-generation population of size N from the combined set of current parents and offspring.

Two-Stage Fine-Tuning of the LLM. To equip a large language model with architecture understanding and conditional generation capabilities for Transformer architecture search, we adapt a base model (i.e., Qwen2.5-14B

Algorithm 1 Main framework of CF-TAS

Input: Resource constraint interval $[L, U]$ on $p(\mathcal{A})$, Population size N, generations T, architecture space Ω.
Output: Best feasible architecture

1: $seeds \leftarrow$*Random Sample from* Ω
2: $P_0 \leftarrow$Initialize from seeds via ***DAT*** and ***DLR***;
3: **For** each generation t from 0 to $T-1$:
4: $S_t \leftarrow$*SelectKneeAndBoundary* (P_t);
5: $Q_t^{EC} \leftarrow$*EC_Variation* (P_t);
6: $Q_t^{LLM} \leftarrow$***DAT*** (S_t);
7: $Q_t \leftarrow Q_t^{EC} \cup Q_t^{LLM}$;
8: $Q_t \leftarrow$ ***DLR*** (Q_t) on infeasible candidates;
9: $P_{t+1} \leftarrow$ *EnvironmentalSelection* $(P_t \cup Q_t)$
10: **End for**
11: **Return** the final feasible non-dominated architecture set.

[29]) via two-stage supervised fine-tuning (SFT). In the first stage, we construct a "design principle Q&A" dataset following the Easy Dataset [30] recipe. The objective is to help the model learn common structural regularities of Transformer architectures reported in the literature, so that it can provide reasonable modification suggestions given a design intent. In the second stage, we construct paired "architecture–metric" Q&A data, enabling the model to learn the correspondence between architecture descriptions and evaluation metrics. Specifically, the input x is a structured architecture description (i.e., an architecture summary based on $\mathcal{A} = (D, E, \boldsymbol{r}, \boldsymbol{h})$), and the output y is the metric information produced by a unified evaluation pipeline, including the parameter count $p(\mathcal{A})$, FLOPs $f(\mathcal{A})$, and the performance proxy score $s(\mathcal{A})$. The two-stage training objective can be written in a unified form as

$$\max_{\theta} \sum_{(x,y)\in\mathcal{S}_{prin}} log\ p_\theta(y \mid x) + \lambda \sum_{(x,y)\in\mathcal{S}_{arch}} log\ p_\theta(y \mid x) \qquad (4)$$

where $\mathcal{S}_{prin}$ and $\mathcal{S}_{arch}$ denote the datasets for the two stages, θ is the model parameters, and λ balances the training weights of the two data sources. The second stage uses evaluation metrics generated by the unified pipeline as supervision, allowing the model to exploit architecture–metric associations for conditional reasoning when editing candidate architectures.

Building on this two-stage fine-tuned LLM, we further use it as the script generation suite in the proposed method. Given conditional information such as a parent architecture, constraint violation status, and optimization preference, the model produces a small number of executable edit scripts.

III. Methodology

This section provides a detailed description of the proposed LLM-guided evolutionary Transformer architecture search method. We adopt NSGA-II as the search backbone, which could be easily replaced by a comparable MOEA, and introduce two script-based architecture transformation modules, DAT and DLR. These modules enable directional architecture search under constraints.

A. Search Framework

Building on the definitions in Section II, this section develops an LLM-guided evolutionary framework for Transformer architecture search, aiming to identify high-quality Transformer architectures that balance performance and computational cost under resource constraints. The framework is built upon a multi-objective evolutionary backbone (i.e., NSGA-II [17]), which preserves search coverage and population diversity. On top of this backbone, we introduce two types of architecture edit script-based operations that are conditionally generated by a two-stage fine-tuned LLM, providing directional guidance for search. Specifically, DAT generates a set of edit scripts for representative parents to produce candidates, thereby guiding the search toward more favorable trade-off while maintaining architectural inheritance. DLR is applied only to infeasible individuals, producing a repair script that pushes them back toward the feasible range as much as possible, thereby reducing wasted budget. The overall procedure is summarized in Algorithm 1.

Initialization. The initial population P_0 is constructed via a seed-expansion scheme (Lines 1–2). Starting from a small set of seed architectures, DAT generates a few scripts to edit candidates. For candidates that violate the constraint, DLR further outputs lightweight local edit scripts for refinement, yielding a more stable initial population. This expansion is repeated until the population size reaches N.

Offspring generation. In the main loop of each generation, candidates are produced by two parallel branches and then merged. The evolutionary branch generates offspring Q_t^{EC} via crossover and mutation (Line 5), providing broad exploration and maintaining diversity. The LLM branch is applied only to a representative parent composed of knee and boundary solutions (Line 4) to obtain more effective directional candidates. For each individual $\mathcal{A} \in S_t$, DAT generates a small number of edit scripts, forming Q_t^{LLM} (Line 6). Since the evolutionary branch also produces a substantial number of infeasible candidates, we apply DLR to repair infeasible individuals in both Q_t^{EC} and Q_t^{LLM} (Line 8), reducing unnecessary evaluation and preserving architectural inheritance.

Evaluation and selection. In each generation, we first evaluate the merged candidate set $P_t \cup Q_t$, and then perform environment selection based on constraint handling and multi-objective selection criteria [17] [28], thereby advancing the trade-off front while satisfying constraint. After the loop terminates, we return the final feasible non-dominated architecture set (Line 11).

B. Directional Architecture Transformation

During search, DAT leverages the two-stage fine-tuned LLM to generate a small number of executable edit scripts, which are then applied to parent architectures to produce candidate architectures. In this way, DAT provides explicit directionality for candidate generation and mitigates the undirected perturbations inherent in the search process. DAT is employed in both the initialization and iterative search stages. In the initialization, we first sample a small set of seed architectures from the search space as starting points. For each seed architecture, DAT generates multiple edit scripts and produces a batch of candidates, thereby increasing the feasibility probability of the initial population without relying on large-scale post hoc filtering, and providing a more stable starting point for subsequent evolution. In the search stage, DAT is applied only to a representative parent set S_t, which consists of knee solution and boundary solutions. Knee solution characterizes the trade-off region of the front, while boundary solutions correspond to the endpoints of the feasible non-dominated set with either the highest proxy score or the lowest FLOPs. By concentrating generation on representative parents, this strategy focuses the budget on the most critical regions in the front while mitigating the loss of candidate diversity. Algorithm 2 presents the complete procedure for generating the LLM candidate set Q^{LLM} from representative parents in each generation.

Algorithm 2 Directional Architecture Transformation (DAT)

Input: Representative parents S_t, each $\mathcal{A} \in S_t$ state (proxy $s(\mathcal{A})$, FLOPs $f(\mathcal{A})$, Params $p(\mathcal{A})$, violation $\phi(\mathcal{A})$), budgets (K scripts per parent, max m edits per script).

Output: LLM-generated offspring set Q^{LLM}.

1: $Q^{LLM} \leftarrow \emptyset$;
2: **For** each $\mathcal{A}$ in S_t:
3: $X \leftarrow$ Encode Architecture $\mathcal{A}$ and its current state;
4: *K scripts* $\leftarrow$ *LLM_GenerateScripts* (X);
5: $O \leftarrow$ *ExecuteScript* ($\mathcal{A}$, *K scripts*);
6: $Q^{LLM} \leftarrow Q^{LLM} \cup O$
7: **End for**
8: **Return** Q^{LLM}.

Prompt 1 Prompt specification used in DAT

Task:

Generate exactly K edit scripts (each with at most m edits) for a Transformer architecture.

Input fields:

Architecture encoding: $D, E, r[\cdot], h[\cdot]$;

Metrics: proxy $s(\mathcal{A})$, FLOPs $f(\mathcal{A})$, Params $p(\mathcal{A})$, violation $\phi(\mathcal{A})$;

Script budgets: K, m.

Output format (JSON):

Exactly K scripts;

Each script has $\leq m$ edits;

Each edit specifies: $\{op, field \in \{D, E, r, h\}, value \in allowed\ discrete\ space\}$.

Generation constraints:

If $\phi > 0$: prioritize scripts that reduce violation after applying the script;

If $\phi = 0$: prioritize improving proxy and/or reducing FLOPs while keeping Params feasible;

Diversity: first-edit fields across scripts cover at least two distinct fields.

Specifically, DAT adopts a script-based generation scheme that restricts the output of the two-stage fine-tuned LLM to a small number of executable edit scripts. This restriction controls the magnitude of transformations and preserves architectural inheritance, thereby enabling stable coupling with the evolutionary loop. For each parent, we construct a structured conditional input that includes the parent architecture encoding, proxy score, FLOPs, parameter count and its violation degree, and the script budget (K, m) (Line 3). The LLM outputs a JSON object containing K scripts (Line 4). Each script consists of at most m edit operations, and each operation explicitly specifies the field to be modified and its new value. Applying these scripts to the parent yields the corresponding candidates (Line 5), which are then aggregated into Q^{LLM} (Line 6). The directionality of DAT is jointly determined by the conditional input and the prompt rules. In particular, the two-stage fine-tuned LLM is conditioned on the parent's constraint violation degree and objective signals, with generation priorities explicitly specified in the prompt. When the parent is infeasible, script generation prioritizes reducing the violation degree. When the parent is feasible, script generation prioritizes improving the proxy score or reducing FLOPs, while keeping the parameter count within the constraint range. As illustrated in Prompt 1, to ensure executability and consistency, DAT employs a structured prompting specification that constrains the input fields, output format, and generation constraints.

C. Directional Local Refinement

During search, DLR performs lightweight local corrections on infeasible individuals to increase the feasibility rate of candidates and reduce evaluation cost. Its scope is restricted to infeasible candidates, namely those with violation degree $\phi(\mathcal{A}) > 0$. DLR is used in both the initialization and iteratives search stages. In both stages, DLR is applied only to the infeasible candidates for local correction. Algorithm 3 presents the complete procedure for generating and applying a single refinement script to each infeasible candidate.

Algorithm 3 Directional Local Refinement (DLR)

Input: Candidate set Q_t, each $\mathcal{A} \in S_t$ state (proxy $s(\mathcal{A})$, FLOPs $f(\mathcal{A})$, Params $p(\mathcal{A})$, violation $\phi(\mathcal{A})$), budget (max ℓ edits).

Output: Updated candidate set Q_t.

1: **For** each $\mathcal{A}$ in Q_t:
2: $X \leftarrow$ Encode Architecture $\mathcal{A}$ and its current state;
3: *Repair script* $\leftarrow$ *LLM_GenerateRepairScript* (X);
4: $\mathcal{A}' \leftarrow$ *ExecuteScript* ($\mathcal{A}$, *Repair script*);
5: **IF** $\phi(\mathcal{A}') < \phi(\mathcal{A})$:
6: Replace $\mathcal{A}$ with $\mathcal{A}'$ in Q_t
7: **End IF**
8: **End for**
9: **Return** Q_t.

Prompt 2 Prompt specification used in DLR

Task:

Generate exactly one local repair script (with at most ℓ edits) to reduce Params violation.

Input fields:

Architecture encoding: $D, E, r[\cdot], h[\cdot]$;

Metrics: proxy $s(\mathcal{A})$, FLOPs $f(\mathcal{A})$, Params $p(\mathcal{A})$, violation $\phi(\mathcal{A})$;

Script budgets: ℓ .

Output format (JSON):

Exactly 1 script;

The script has $\leq \ell$ edits;

Each edit specifies: $\{op, field \in \{D, E, r, h\}, value \in allowed\ discrete\ space\}$.

Generation constraints:

Primary goal: reduce violation ϕ after applying the script and reach $\phi = 0$ if possible;

Locality: keep edits few and small and avoid changing many fields in one script.

Specifically, DLR follows a script-based formulation but strictly restricts its output to a single short, local edit script to avoid introducing noticeable architecture drift. For each infeasible candidate $\mathcal{A}$, we construct a structured conditional input that includes the candidate architecture encoding, proxy score, FLOPs, parameter count and its violation degree, the constraint range, and a refinement step budget ℓ (Line 2). The two-stage fine-tuned LLM outputs a JSON object containing only one correction script (Line 3). This script consists of at most ℓ edit operations, and each operation explicitly specifies the field to be modified and its value. Applying the script to the candidate $\mathcal{A}$ yields the repaired candidate $\mathcal{A}'$ (Line 4). The directionality of DLR is jointly determined by the conditional input and the prompt rules. The violation degree is provided as the primary signal, and the prompt explicitly specifies the generation priority so that the script aims first to reduce violation and satisfy the constraint as quickly as possible. Meanwhile, by constraining the script length and limiting modification magnitudes, DLR ensures that feasibility recovery is achieved with needed changes. To ensure executability and consistency, DLR also adopts a structured prompting specification that constrains the input fields, output format, and generation priorities, as detailed in Prompt 2.

TABLE I. COMPARISON OF CF-TAS WITH STATE-OF-THE-ART METHODS UNDER TWO RESOURCE CONSTRAINTS.

Constraint	Models	Top-1 (%)	Top-5 (%)	Parameters (M)	FLOPS (B)	Model Type	Design Type
6 M	ResNet-18 [31]	72.5	-	11.7	1.8	CNN	Manual
	MobileNet-V3 [32]	75.2	-	5.5	-	CNN	Manual
	DeiT-Ti [25]	72.2	91.1	5.7	1.2	Transformer	Manual
	TNT-Ti [33]	73.9	91.9	6.1	1.4	Transformer	Manual
	ViT-Ti [3]	74.5	-	5.7	-	Transformer	Manual
	CPVT-Ti [34]	74.9	92.6	6.0	-	Transformer	Manual
	PVT-Tiny [35]	75.1	-	13.2	1.9	Transformer	Manual
	ViTAS-C [36]	74.7	91.6	5.6	1.3	Transformer	Auto
	AutoFormer-Ti [7]	74.7	92.6	5.7	1.3	Transformer	Auto
	GLiT-Ti [37]	76.3	-	7.2	1.4	Hybrid	Auto
	TF-TAS-Ti [27]	75.3	92.8	5.9	1.4	Transformer	Auto
	CF-TAS (6 M)	**75.5**	**92.9**	**6.5**	**1.5**	**Transformer**	**Auto**
54 M	ResNet-152 [31]	81.9	-	60.2	11.5	CNN	Manual
	RegNetY-16GF[38]	80.4	-	83.6	15.9	CNN	Manual
	ViT-B/16 [3]	79.7	-	86.0	18.0	Transformer	Manual
	PVT-Large [35]	81.7	-	61.0	9.8	Transformer	Manual
	DeiT-B [25]	81.8	95.6	86.0	18.0	Transformer	Manual
	CPVT-B [34]	82.3	-	88.0	-	Transformer	Manual
	TNT-B [33]	82.9	96.3	65.5	14.1	Transformer	Manual
	Swin-B [4]	83.5	-	88.0	15.4	Transformer	Manual
	T2T-ViT-24 [39]	82.6	-	64.1	-	Transformer	Manual
	GLiT-B [37]	82.3	-	96.0	17.0	Hybrid	Auto
	AutoFormer-B [7]	82.4	95.7	54.0	11.0	Transformer	Auto
	TF-TAS-B [27]	82.2	95.6	54.0	12.0	Transformer	Auto
	CF-TAS (54 M)	**82.2**	**95.8**	**58.3**	**12.0**	**Transformer**	**Auto**

IV. EXPERIMENTS

This section presents a series of experiments to evaluate CF-TAS**.** We first describe the experimental setup and implementation details. We then compare CF-TAS with existing SOTA methods under different budgets. Finally, we conduct ablation studies to analyze the contribution of DAT and DLR to the overall framework.

A. Implementation Details

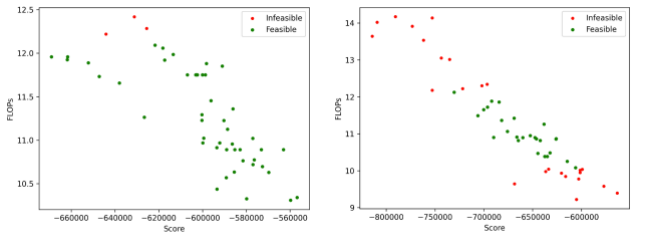
We conduct search and evaluation on ImageNet-1K. It contains approximately 1.2M training images across 1,000 classes. We report Top-1/Top-5 accuracy on the validation set, together with parameter count and FLOPs to characterize deployment cost. To cover different parameter budgets, we consider two target parameter counts $p \in \{6\text{M}, 54\text{M}\}$ and define the feasible parameter range as $[L, U] = [0.9p, 1.1p]$, where architectures within this range are treated as satisfying the constraint. During search, the performance proxy $s(\mathcal{A})$ is instantiated by the DSS-indicator [27]. All experiments are implemented in PyTorch and run on a machine equipped with 4 NVIDIA RTX PRO 6000 GPUs, two 64-core CPUs, and 1TB memory.

Search and training follow a two-stage protocol. In the search stage, we adopt NSGA-II [17] as the evolutionary backbone with population size $N = 50$ for 50 generations, using crossover and mutation probabilities of 0.9 and 0.2, respectively. In each generation, candidates are produced from two sources, in which one via crossover/mutation and the other via edit scripts generated by the fine-tuned LLM. For DAT, LLM sample $K = 6$ scripts, each containing at most $m = 3$ edits. For DLR, it generates one correction script with a maximum of $\ell = 3$ edits. The LLM employed is the two-stage supervised fine-tuned Qwen2.5-14B [29]. At inference time, we generate JSON-formatted edit scripts using fixed sampling parameters. In the training stage, to avoid the cost of training from scratch, the final selected architecture is initialized from the AutoFormer supernet [7] and then fine-tuned on ImageNet using AdamW for 30 epochs with an initial learning rate of 1×10^{-6} (minimum 1×10^{-7}) under cosine annealing, batch size 64, and weight decay 5×10^{-2}. Data augmentation follows the supernet training setting. To mitigate randomness, we repeat the search five times and select the median result for training. For each run, we select the architecture with the highest proxy score from the final feasible non-dominated architecture set for final comparison.

B. Comparisons with The-state-of-the-arts

To evaluate the effectiveness of the proposed method, we compare CF-TAS with a set of representative vision backbones on ImageNet under the 54M and 6M parameter budgets, and report the results in Table I. The baselines are chosen to span different architectural paradigms and design methodologies, enabling a comparison across manually designed models and automated architecture design models. Specifically, the first group includes manually designed CNNs, namely ResNet [31], MobileNet-V3 [32], and RegNetY [38]. These models serve as conventional references, providing a context for assessing the benefits and costs of Transformer-based backbones under resource budgets. The second group consists of manually designed Transformers, including DeiT [25], ViT [3], TNT [33], CPVT [34], PVT [35], Swin [4], and T2T-ViT [39]. This group covers representative design choices such as plain ViTs, local–global interaction mechanisms, pyramid or multi-scale representations, and token reorganization. The third group comprises automated design and search methods, including ViTAS [36], AutoFormer [7], TF-TAS [27], and GLiT [37], which fall within the same category of automated architecture design and therefore constitute the most direct baselines for CF-TAS. All methods are compared using the same set of metrics, including Top-1/Top-5 accuracy, parameter count, and FLOPs.

Under the 6M target parameter budget, the architecture searched by CF-TAS achieves 75.5% Top-1 and 92.9% Top-5, with 6.5M parameters and 1.5B FLOPs. Among hand-crafted CNN baselines, ResNet-18 reaches 72.5% Top-1 with 11.7M parameters, whereas MobileNet-V3 attains 75.2% Top-1 at a parameter scale closer to the 6M regime. Under comparable model size, CF-TAS yields a higher Top-1 than MobileNet-V3, and it exceeds ResNet-18 in accuracy while using markedly fewer parameters. For manually designed Transformers, the reported Top-1 accuracies are 72.2%



(a) Initialization based on DAT and DLR (b) Random initialization

Fig. 1. The population initialization results based on DAT and DLR, and random initialization. Green points represent solutions located within the feasible range, while red points represent solutions located outside the feasible range.

(DeiT-Ti), 73.9% (TNT-Ti), 74.5% (ViT-Ti), 74.9% (CPVT-Ti), and 75.1% (PVT-Tiny). CF-TAS achieves a higher Top-1 than these hand-crafted Transformer baselines while remaining near the target parameter budget, demonstrating improved performance under the same budget. For automated design and search baselines, ViTAS-C and AutoFormer-Ti both report 74.7% Top-1, and TF-TAS-Ti reports 75.3%. CF-TAS improves Top-1 by 0.8 percentage points over both ViTAS-C and AutoFormer-Ti, and by 0.2 percentage points over TF-TAS-Ti, while achieving 92.9% Top-5. GLiT-Ti reports 76.3% Top-1 with 7.2M parameters, which exceeds the 6M budget regime. Compared to these, CF-TAS attains a similar accuracy level with fewer parameters.

Under the 54M target parameter budget, the model produced by CF-TAS achieves 82.2% Top-1 and 95.8 % Top-5 accuracy, with 58.3M parameters and 12.0B FLOPs. Compared with hand-crafted CNN baselines, ResNet-152 reports 81.9% Top-1 and RegNetY-16GF reports 80.4% Top-1, with 60.2M and 83.6M parameters, respectively. CF-TAS attains a Top-1 that is higher than ResNet-152 and RegNetY-16GF, while avoiding the larger model size and computational cost. This comparison indicates that, Transformer architectures can achieve competitive accuracy under the same parameter budget. For hand-crafted Transformers, the reported accuracies are 79.7% Top-1 (ViT-B/16), 81.7% Top-1 (PVT-Large), 81.8% Top-1 with 95.6% Top-5 (DeiT-B), 82.3% Top-1 (CPVT-B), 82.6% Top-1 (T2T-ViT-24), 82.9% Top-1 with 96.3% Top-5 (TNT-B), and 83.5% Top-1 (Swin-B). Overall, stronger hand-crafted models such as Swin-B and TNT-B achieve higher Top-1, but typically at the cost of increased parameter counts and FLOPs. Under 58.3M parameters and 12.0B FLOPs, CF-TAS achieves 82.2% Top-1 and 95.8% Top-5, showing that its accuracy gains at this budget are obtained without excessive computational overhead. Finally, compared with automated design and search methods, AutoFormer-B reports 82.4% Top-1 with 95.7% Top-5, TF-TAS-B reports 82.2% Top-1 with 95.6% Top-5, and GLiT-B reports 82.3% Top-1. CF-TAS matches TF-TAS-B in Top-1 and achieves a slightly higher Top-5. Relative to AutoFormer-B, the Top-1 gap is small while Top-5 is higher. Taken all together, these results show that CF-TAS reaches an accuracy level comparable to representative NAS methods under the 54M budget, while maintaining a reasonable FLOPs cost, supporting its effectiveness in advancing the trade-off frontier under this constraint.

C. Ablation Study and Analysis

This section conducts ablation studies under the 54M parameter budget to assess the effectiveness of the proposed metho d, with emphasis on two questions. First, we evaluate whether DAT and DLR improve the proportion of the initialized population that lies within the feasible range.

TABLE II. COMPARISON RESULTS OF DIFFERENT SEARCH STRATEGIES

Method	Top-1	Top-5	#Params	FLOPs
Random Search	82.0%	95.6%	59.3M	12.2B
Evolutionary Search	82.1%	95.8%	61.3M	12.5B
CF-TAS	**82.2 %**	**95.8 %**	**58.3M**	**12.0B**

Second, we compare the accuracy and budget matching of architectures produced by different search strategies.

As shown in Fig. 1, initializing with DAT and DLR results in more candidates falling inside the feasible range, whereas random initialization yields a larger number of infeasible candidates outside the range. This observation is consistent with DAT producing edits and DLR applying local adjustments, which increases feasibility at initialization and reduces the number of infeasible candidates that would otherwise be evaluated in later iterations. Table II compares Random Search, Evolutionary Search, and CF-TAS under the 54M parameter budget. Here, Evolutionary Search is implemented as a standard NSGA-II baseline that optimizes proxy and FLOPs and selects the final architecture using the knee criterion, without explicitly enforcing the parameter-budget feasibility during selection. For a consistent comparison, Random Search uniformly samples candidates from the search space and returns the feasible architecture with the highest proxy score. CF-TAS achieves 82.2% Top-1 and 95.8% Top-5 with 58.3M parameters and 12.0B FLOPs. Random Search achieves 82.0% Top-1 and 95.6% Top-5 with 59.3M parameters and 12.2B FLOPs. Evolutionary Search achieves 82.1% Top-1 and 95.8% Top-5 with 61.3M parameters and 12.5B FLOPs, in which the parameter count exceeds the budget. These results show that CF-TAS attains higher Top-1 and Top-5 than Random Search. Compared with Evolutionary Search, CF-TAS achieves similar accuracy while producing an architecture whose parameter count lies within the budget range.

V. CONCLUSION

This paper studies constrained Transformer architecture search for resource-constrained deployment, focusing on the high proportion of infeasible candidates that can waste a substantial amount of evaluation budget. We propose CF-TAS, an LLM-guided coarse-to-fine search framework built on a multi-objective evolutionary backbone that preserves global exploration and population diversity. In the proposed framework, a fine-tuned LLM is used to generate executable edit scripts that provide controllable, direction-aware guidance for candidate generation. In particular, DAT performs directional transformations during both initialization and iterative search, guiding candidates toward promising feasible regions, while DLR applies lightweight local corrections to infeasible candidates to reduce constraint violation and recover feasibility with minimal architectural perturbation. Experiments under different budget settings show that CF-TAS consistently returns feasible architectures and achieves performance comparable to or better than state-of-the-art baselines. Future work will investigate broader deployment scenarios and more efficient LLM-integration strategies to reduce overhead and improve generality.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems,* vol. 30, 2017.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *Proceedings of the 2019 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171-4186.

[3] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[4] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows." pp. 9992-10002.

[5] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient Transformers: A Survey," *ACM Comput. Surv.,* vol. 55, no. 6, pp. Article 109, 2022.

[6] F. Chu, H. Li, L. Xie, and J. Zhao, "A survey of transformer architectures for autonomous driving," *Expert Systems with Applications,* vol. 299, pp. 130338, 2026/03/01/, 2026.

[7] M. Chen, H. Peng, J. Fu, and H. Ling, "AutoFormer: Searching Transformers for Visual Recognition." pp. 12250-12260.

[8] M. Chen, K. Wu, B. Ni, H. Peng, B. Liu, J. Fu, H. Chao, and H. Ling, "Searching the search space of vision transformer," *Advances in Neural Information Processing Systems,* vol. 34, pp. 8714-8726, 2021.

[9] Q. Zhou, K. Sheng, X. Zheng, K. Li, Y. Tian, J. Chen, and R. Ji, "Training-Free Transformer Architecture Search With Zero-Cost Proxy Guided Evolution," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 46, no. 10, pp. 6525-6541, 2024.

[10] C. Yan, X. Chang, W. Zhang, Z. Li, L. Yao, M. Luo, and F. Tian, "Self-supervised transformer architecture search via multi-level masked information fusion," *Information Fusion,* vol. 126, pp. 103640, 2026/02/01/, 2026.

[11] A. Castagnetti, A. Pegatoquet, B. Miramond, O. Montfort, and V. Huard, "Hardware-Aware Neural Architecture Search for Memory constrained Embedded Neural Networks Accelerators." pp. 1-5.

[12] L. Liu, G. G. Yen, and Z. He, "LatentTAS: Constrained Transformer Architecture Search Via Surrogate-Driven Latent Evolution." pp. 446-452.

[13] K. T. Chitty-Venkata, M. Emani, V. Vishwanath, and A. K. Somani, "Neural Architecture Search for Transformers: A Survey," *IEEE Access,* vol. 10, pp. 108374-108412, 2022.

[14] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A Survey on Evolutionary Neural Architecture Search," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 34, no. 2, pp. 550-570, 2023.

[15] S. Li, Y. Sun, G. G. Yen, and M. Zhang, "Automatic Design of Convolutional Neural Network Architectures Under Resource Constraints," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 34, no. 8, pp. 3832-3846, 2023.

[16] L. Liu, G. G. Yen, and Z. He, "EvolutionViT: Multi-objective evolutionary vision transformer pruning under resource constraints," *Information Sciences,* vol. 689, pp. 121406, 2025/01/01/, 2025.

[17] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation,* vol. 6, no. 2, pp. 182-197, 2002.

[18] H. Kamigaito, Y. Zhang, J. Kwon, K. Hayashi, M. Okumura, and T. Watanabe, "Diversity of Transformer Layers: One Aspect of Parameter Scaling Laws," *arXiv preprint arXiv:2505.24009*, 2025.

[19] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian, "A Comprehensive Overview of Large Language Models," *ACM Trans. Intell. Syst. Technol.,* vol. 16, no. 5, pp. Article 106, 2025.

[20] S. S. Chowa, R. Alvi, S. S. Rahman, M. A. Rahman, M. A. K. Raiaan, M. R. Islam, M. Hussain, and S. Azam, "From language to action: a review of large language models as autonomous agents and tool users," *Artificial Intelligence Review,* vol. 59, no. 2, pp. 71, 2026/01/06, 2026.

[21] F. Liu, X. Tong, M. Yuan, X. Lin, F. Luo, Z. Wang, Z. Lu, and Q. Zhang, "Evolution of heuristics: towards efficient automatic algorithm design using large language model," in Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria, 2024, pp. Article 1304.

[22] X. Zhou, X. Wu, L. Feng, Z. Lu, and K. C. Tan, "Design Principle Transfer in Neural Architecture Search via Large Language Models," *Proceedings of the AAAI Conference on Artificial Intelligence,* vol. 39, no. 21, pp. 23000-23008, 04/11, 2025.

[23] X. Wu, S. H. Wu, J. Wu, L. Feng, and K. C. Tan, "Evolutionary Computation in the Era of Large Language Model: Survey and Roadmap," *IEEE Transactions on Evolutionary Computation,* vol. 29, no. 2, pp. 534-554, 2025.

[24] R. Lange, Y. Tian, and Y. Tang, "Large Language Models As Evolution Strategies," in Proceedings of the Genetic and Evolutionary Computation Conference Companion, Melbourne, VIC, Australia, 2024, pp. 579–582.

[25] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou, "Training data-efficient image transformers & distillation through attention," in Proceedings of the 38th International Conference on Machine Learning, Proceedings of Machine Learning Research, 2021, pp. 10347--10357.

[26] J. Liang, X. Ban, K. Yu, B. Qu, K. Qiao, C. Yue, K. Chen, and K. C. Tan, "A Survey on Evolutionary Constrained Multiobjective Optimization," *IEEE Transactions on Evolutionary Computation,* vol. 27, no. 2, pp. 201-221, 2023.

[27] Q. Zhou, K. Sheng, X. Zheng, K. Li, X. Sun, Y. Tian, J. Chen, and R. Ji, "Training-free Transformer Architecture Search." pp. 10884-10893.

[28] K. Deb, A. Pratap, and T. Meyarivan, "Constrained Test Problems for Multi-objective Evolutionary Optimization." pp. 284-298.

[29] Q. A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, G. Dong, H. Wei, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, M. Xue, P. Zhang, Q. Zhu, R. Men, R. Lin, T. Li, T. Xia, X. Ren, X. Ren, Y. Fan, Y. Su, Y.-C. Zhang, Y. Wan, Y. Liu, Z. Cui, Z. Zhang, Z. Qiu, S. Quan, and Z. Wang, "Qwen2.5 Technical Report," *ArXiv,* vol. abs/2412.15115, 2024.

[30] Z. Miao, Q. Sun, J. Wang, Y. Gong, Y. Zheng, S. Li, and R. Zhang, "Easy Dataset: A Unified and Extensible Framework for Synthesizing LLM Fine-Tuning Data from Unstructured Documents," *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations.* pp. 960-968.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition." pp. 770-778.

[32] A. Howard, M. Sandler, B. Chen, W. Wang, L. C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, "Searching for MobileNetV3." pp. 1314-1324.

[33] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, "Transformer in transformer," in Proceedings of the 35th International Conference on Neural Information Processing Systems, 2024, pp. Article 1217.

[34] X. Chu, Z. Tian, B. Zhang, X. Wang, and C. Shen, "Conditional positional encodings for vision transformers," *arXiv preprint arXiv:2102.10882*, 2021.

[35] W. Wang, E. Xie, X. Li, D. P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions." pp. 548-558.

[36] X. Su, S. You, J. Xie, M. Zheng, F. Wang, C. Qian, C. Zhang, X. Wang, and C. Xu, "ViTAS: Vision Transformer Architecture Search." pp. 139-157.

[37] B. Chen, P. Li, C. Li, B. Li, L. Bai, C. Lin, M. Sun, J. Yan, and W. Ouyang, "GLiT: Neural Architecture Search for Global and Local Image Transformer." pp. 12-21.

[38] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of Tricks for Image Classification with Convolutional Neural Networks." pp. 558-567.

[39] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z. Jiang, F. E. H. Tay, J. Feng, and S. Yan, "Tokens-to-Token ViT: Training Vision Transformers from Scratch on ImageNet." pp. 538-547.