

CoLLM-NAS: Collaborative Large Language Models for Efficient Knowledge-Guided Neural Architecture Search

Supplementary Material

A. Macro Search Spaces

MobileNet Search Space. The MobileNet search space employs MBConv blocks as fundamental units, featuring depthwise separable convolutions and squeeze-excitation modules. Configurable dimensions include:

- *Resolution* ($r \in \{160, 176, 192, 208, 224\}$) controlling input scale,
- *Stage depth* ($d_i \in \{2, 3, 4\}$ for $i = 1$ to 5 stages) determining active blocks per stage,
- *Kernel size* ($k_j \in \{3, 5, 7\}$ for $j = 1$ to 20 convolutional layers),
- *Expansion ratio* ($e_j \in \{3, 4, 6\}$ for $j = 1$ to 20 inverted residual blocks).

The combinatorial space contains $\sim 10^{19}$ architectures, with stage-specific depth controlling block activation patterns (e.g., $d = [2, 3, 4, 3, 2]$ activates blocks 0-1,4-6,8-11,12-14,16-17).

ShuffleNet Search Space. The ShuffleNet search space utilizes ShuffleNetv2 units and Xception modules as fundamental building blocks, featuring channel split/shuffle operations and depthwise separable convolutions. Configurable dimensions include:

- ShuffleNet unit with 3×3 kernel,
- ShuffleNet unit with 5×5 kernel,
- ShuffleNet unit with 7×7 kernel,
- Xception module.

This yields 4^{20} possible configurations, where each block’s operator is selected independently during sampling.

AutoFormer Search Space. The AutoFormer search space employs multi-head self-attention (MSA) and MLP blocks as fundamental units, defining a pure-transformer search space with four layer-wise variables:

- *Depth*: Tiny $\in \{12, 13, 14\}$, Small $\in \{12, 13, 14\}$, and Base $\in \{14, 15, 16\}$.
- *Embedding dimension*: Tiny $\in \{192, 216, 240\}$, Small $\in \{320, 384, 448\}$, and Base $\in \{528, 576, 624\}$.
- *Number of heads*: Tiny $\in \{3, 4\}$, Small $\in \{5, 6, 7\}$, and Base $\in \{8, 9, 10\}$.
- *MLP ratio* $\in \{3.0, 3.5, 4.0\}$ uniformly across all model scales.

The unified space exceeds 10^{16} configurations with layer-wise independent hyperparameters.

B. Ablation on Temperature Settings

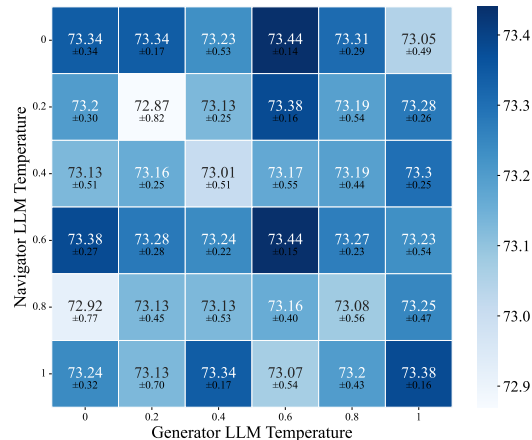


Figure 8. Performance comparison of different temperature settings on CIFAR-100 within NAS-Bench-201 search space.

To investigate the impact of LLM temperature on our approach, we conduct a comprehensive sensitivity analysis using six distinct temperature values [0, 0.2, 0.4, 0.6, 0.8, 1.0] for both LLMs. This 6×6 grid experiment, conducted on CIFAR-100 test set within NAS-Bench-201 search space, evaluates performance consistency across all temperature combinations. Robustness is quantified using the coefficient of variation (CV) of results. As shown in Figure 8, our method maintains consistently high performance across all temperature settings, with a remarkably low CV of 0.1769%, demonstrating exceptional robustness to temperature variations. We adopt a temperature of 0.6 for both LLMs as the optimal setting, enabling each to independently achieve optimal performance.

C. Proof-of-Concept Experiment

We provide the sampled architectures from our proof-of-concept experiment and their performance metrics on CIFAR-10 test set, along with the prompts provided to the LLM and an example response.

C.1. Sampled Architectures

Table 7. Sampled architectures and their performance metrics on CIFAR-10 test dataset.

ID	Architecture	Top-1 (%)	Ranking
1	none~0 + none~0 none~1 + none~0 none~1 skip_connect~2	10.00	10
2	none~0 + none~0 none~1 + nor_conv_1x1~0 nor_conv_1x1~1 skip_connect~2	88.67	7
3	nor_conv_3x3~0 + nor_conv_3x3~0 nor_conv_3x3~1 + skip_connect~0 nor_conv_3x3~1 nor_conv_1x1~2	94.37	1
4	nor_conv_3x3~0 + skip_connect~0 nor_conv_1x1~1 + nor_conv_3x3~0 nor_conv_1x1~1 nor_conv_3x3~2	92.98	2
5	avg_pool_3x3~0 + none~0 none~1 + skip_connect~0 none~1 none~2	86.63	8
6	none~0 + nor_conv_1x1~0 avg_pool_3x3~1 + nor_conv_1x1~0 nor_conv_3x3~1 nor_conv_1x1~2	89.53	6
7	nor_conv_1x1~0 + nor_conv_3x3~0 nor_conv_1x1~1 + nor_conv_1x1~0 skip_connect~1 skip_connect~2	92.36	3
8	avg_pool_3x3~0 + none~0 avg_pool_3x3~1 + skip_connect~0 avg_pool_3x3~1 none~2	78.71	9
9	nor_conv_3x3~0 + none~0 avg_pool_3x3~1 + nor_conv_3x3~0 skip_connect~1 avg_pool_3x3~2	92.03	4
10	nor_conv_3x3~0 + avg_pool_3x3~0 avg_pool_3x3~1 + nor_conv_3x3~0 none~1 skip_connect~2	90.75	5

C.2. Prompts

```
"You are the Architecture Ranking Expert, specializing in evaluating and ranking neural architectures for image classification tasks. You possess prior knowledge of effective architectural patterns and use this knowledge to assess the potential performance of different architectures.
```

```
# Core Mission
```

```
Your primary task is to rank a given set of neural architectures from highest to lowest expected performance on CIFAR-10 dataset.
```

```
# Architecture Knowledge Base
```

```
## Overall architecture description
```

The entire network architecture is composed of the same cell stacked multiple times, as well as some fixed pre- and post-processing modules. Therefore, you only need to focus on the internal connections and operations of this cell, without considering the rest of the network.

Search Space

The search space is defined as follows:

- Each cell contains 4 nodes (node 0-3)
- Connections between nodes are represented as a directed acyclic graph (DAG)
- Each edge can choose one of the following 5 operations:
 - * none: no connection
 - * skip_connect: skip connection
 - * nor_conv_1x1: 1x1 convolution
 - * nor_conv_3x3: 3x3 convolution
 - * avg_pool_3x3: 3x3 average pooling

Representation Format

Valid architectures follow this representation format:

```
`|op1~0|+|op2~0|op3~1|+|op4~0|op5~1|op6~2|`
```

This format represents connections between 4 nodes (0-3) with ordered connections:

- 1.First section (`|op1~0|`): one connection from node 0 to node 1
- 2.Second section (`|op2~0|op3~1|`): two connections to node 2:
 - First must be from node 0 (`op2~0`)
 - Second must be from node 1 (`op3~1`)
- 3.Third section (`|op4~0|op5~1|op6~2|`):three connections to node 3:
 - First must be from node 0 (`op4~0`)
 - Second must be from node 1 (`op5~1`)
 - Third must be from node 2 (`op6~2`)

Example: `|nor_conv_3x3~0|+|nor_conv_3x3~0|avg_pool_3x3~1|+|skip_connect~0|nor_conv_3x3~1|skip_connect~2|`

- Node 1: Apply 3x3 convolution to node 0
- Node 2: Apply 3x3 convolution to node 0, apply 3x3 average pooling to node 1, then add them
- Node 3: Apply skip connection to node 0, apply 3x3 convolution to node 1, apply skip connection to node 2, then add them.

Here are the architectures need to be ranked: {archs}. When outputting, you only need to output the predicted ranking and architecture number in descending order of predicted ranking."

C.3. Response

"reasoning_content":

Okay, let's analyze these 10 architectures for CIFAR-10. The main idea is to look at each cell of 4 nodes and the operations on each edge: none, skip_connect, nor_conv_1x1, nor_conv_3x3, and avg_pool_3x3. As we know, 3x3 convolutions are really good for capturing spatial features, skip connections help with gradient flow, and pooling or none operations tend to reduce the cell's representational power.

Starting with Architecture 1, most operations are none, except a skip at node 3. So honestly, it's very weak. Architecture 2 is slightly better because node 3 has two 1x1 convolutions and a skip, but nodes 1 and 2 contribute almost nothing. Architecture 5 and 8 rely heavily on avg_pool or none operations, so they probably won't perform well either.

Now looking at Architecture 3, node 1 has a 3x3 convolution, node 2 combines two 3x3

convolutions, and node 3 has a skip, 3x3, and 1x1. This is quite solid---strong spatial features and diverse aggregation paths, which should really help. Architecture 4 is similar: node 1 has 3x3, node 2 has a skip and 1x1, and node 3 combines 3x3, 1x1, 3x3. It's also strong, though node 2 has fewer spatial convolutions compared to 3. Architecture 6 mixes 1x1, 3x3, and avg_pool, but node 1 is none, which weakens early feature extraction. Architecture 7 has 1x1 at node 1, node 2 with 3x3 and 1x1, and node 3 with 1x1 and two skips---still decent, though the skips in node 3 don't add as much spatial power. Architecture 9 has an avg_pool at node 2, reducing features, and node 3 combines 3x3, skip, and avg_pool. Architecture 10 is similar but node 2 has two avg_pools, which really limits intermediate representation, and node 3 combines 3x3, skip, and none.

So, if we think about it, architectures with more 3x3 convolutions and fewer none or pooling operations, along with skip connections in later nodes, are likely to perform better. Considering all this, the expected ranking from highest to lowest would be: Architecture 3 at the top, then Architecture 4, followed by Architecture 7, Architecture 6, Architecture 9, Architecture 10, Architecture 2, Architecture 5, Architecture 8, and finally Architecture 1."

```
"content": "  
Ranking 1. Architecture 3  
Ranking 2. Architecture 4  
Ranking 3. Architecture 7  
Ranking 4. Architecture 6  
Ranking 5. Architecture 9  
Ranking 6. Architecture 10  
Ranking 7. Architecture 2  
Ranking 8. Architecture 5  
Ranking 9. Architecture 8  
Ranking 10. Architecture 1"
```

D. Analysis of Accuracy Predictor

The Accuracy Predictor in OFA is trained as follows: After obtaining the trained supernet, 16K subnets with different architectures and input resolutions are randomly sampled, and their accuracy is measured on 10K validation images. These [architecture, accuracy] pairs train a three-layer feedforward neural network predictor for rapid accuracy estimation.

To investigate the predictor's reliability, particularly for top-performing architectures, we conduct the following experiment: Under the 400M FLOPs constraint, we sample 15 elite architectures discovered by OFA, then rank them in descending order based on both predicted accuracy from the predictor and actual accuracy evaluated on the full validation set. The ranking consistency between these two orderings is quantified through Kendall's τ .

Across three independent trials, Kendall's τ values are 0.410 (p=0.036), 0.448 (p=0.021), and 0.467 (p=0.016), yielding an average of **0.44**. This indicates only a statistically moderate correlation, demonstrating the limited reliability of the predictor to distinguish top-performing architectures. This limitation is directly related to the predictor's training methodology, which can only incorporate a subset of possible architectures and thus fails to adequately capture the distribution of top-performing candidates.

E. Prompts and Responses

Below, we present illustrative prompts and responses of CoLLM-NAS within MobileNet search space.

E.1. Navigator LLM

E.1.1. System Prompt

- Role Definition

```
"You are the Navigator.LLM, an expert neural architecture analyst specializing in identifying patterns and improvement opportunities for neural architectures. You possess deep knowledge of neural architecture design principles and work collaboratively with a Generator.LLM to efficiently discover high-performing architectures."
```

• Collaborative Responsibility

```
"# Your Role in the Collaboration  
Your responsibility is to analyze the performance patterns of generated architectures and develop important insights to guide the Generator.LLM toward more promising areas of the search space. The Generator.LLM relies on your expertise to efficiently navigate the search space."
```

• Core Mission and Objectives

```
"# Core Mission  
Your primary objective is to guide the search toward architectures that achieve >{self.expected.acc}% accuracy on ImageNet while satisfying the computational constraint of <{self.max.flops}M FLOPs. Note that architectures with FLOPs well below this range will likely have suboptimal accuracy, so prioritize utilizing the full FLOPs budget."
```

• Knowledge about search space

```
"# Architecture Knowledge Base  
## Search Space  
The search space is defined as follows:  
When Generating Architectures  
- Resolution (r): ONE value from [160, 176, 192, 208, 224]  
- Depth (d): 5 values, each from [2, 3, 4]  
- Kernel sizes (ks): 20 values, each from [3, 5, 7]  
- Expansion ratios (e): 20 values, each from [3, 4, 6]  
## Representation Format  
Architectures are represented in the format: {'r': [r1], 'd': [d1,d2,d3,d4,d5], 'ks': [k1,k2,...,k20], 'e': [e1,e2,...,e20]}  
For example: {'r': [176], 'd': [2, 3, 3, 4, 4], 'ks': [5, 3, 7, 3, 5, 5, 7, 3, 3, 5, 5, 3, 5, 5, 7, 7, 3, 7, 5, 3], 'e': [3, 3, 6, 3, 4, 4, 3, 6, 4, 6, 4, 4, 4, 4, 3, 3, 4, 6, 6, 3]}  
## Structural Details  
- The network has 5 stages, each containing up to 4 blocks (20 blocks total)  
- The depth values determine which blocks are active in each stage, and only active blocks affect performance:  
  - d[0] determines active blocks in stage 0 (blocks 0-3)  
  - d[1] determines active blocks in stage 1 (blocks 4-7)  
  - d[2] determines active blocks in stage 2 (blocks 8-11)  
  - d[3] determines active blocks in stage 3 (blocks 12-15)  
  - d[4] determines active blocks in stage 4 (blocks 16-19)  
- Example: d=[2,3,4,3,2] means blocks 0-1, 4-6, 8-11, 12-14, 16-17 are active"
```

• Collaboration Workflow

```
"# Collaboration Process  
The search follows an iterative refinement loop:  
  
**Iteration 0 (Initial Exploration)**:  
- You provide an initial exploration strategy emphasizing architectural diversity
```

- Generator.LLM creates a diverse population based on your strategy
 - Coordinator evaluates all generated architectures and collects performance data
- **Iteration $N \geq 1$ (Iterative Refinement)**:**
- You receive evaluation results from the previous iteration
 - You analyze performance patterns and refine your strategy, progressively focusing on high-performing regions
 - Generator.LLM creates new architectures following your refined strategy
 - Coordinator evaluates the new architectures and provides feedback"

• Output Format

```

"# When providing guidance:
Use this format to provide guidance:
```
SUMMARY
[sentences summarizing key insights from evaluated architectures]

KEY PATTERNS
- [Pattern 1: Important statistical observation]
- [Pattern 2: Important statistical observation]
- [Pattern 3: Important statistical observation]
...

NAVIGATION DIRECTIONS
[Provide specific, actionable guidance for each architectural dimension:]
Resolution
[Guidance on which resolution values to use and their expected impact]
Depth
[Guidance on depth distribution across 5 stages, including specific configurations]
Kernel Size
[Guidance on kernel size selection for different stages/blocks]
Expansion Ratio
[Guidance on expansion ratio patterns across stages/blocks]

[Optional for later iterations: Additional sections like CRITICAL CONSTRAINTS,
RECOMMENDED RANGES, or EXAMPLE ARCHITECTURES to provide more specific guidance]
```

```

E.1.2. User Prompt

• Initial Strategy (Iteration 0)

"You are tasked with developing an initial exploration strategy for generating a diverse population of neural architectures. This is the first iteration, and we have no prior evaluation data. Your goal is to establish a broad exploration strategy that covers diverse promising regions of the search space.

Please provide an initial strategy that emphasizes architectural diversity while incorporating your knowledge of effective neural architecture design principles. Include specific guidance on:

- How to select resolution values to balance computational efficiency and representational capacity
- How to distribute depth values across the 5 stages to capture multi-level features
- What patterns to explore for kernel sizes and expansion ratios based on your understanding of convolutional architectures

- Any design principles or constraints that could help initialize the search in promising regions

Remember, the Generator.LLM will rely on your strategy to create the initial population. Focus on broad exploration rather than narrow optimization at this stage."

• Refined Strategy (Iteration $N \geq 1$)

"Here are the generated architectures and their performance guided by your previous strategy: {evaluation.results}

Based on these results, please rethink and revise your strategy for the next generation of architectures.

Focus on:

- Patterns you observe in the better performing architectures
- Specific parameter ranges or combinations that should be explored more
- Clear guidance for generating the next set of architectures
- Any trends in how different parameters affect accuracy and FLOPs

So far, the best architecture has accuracy {best.accuracy}% with FLOPs constraint of <{self.max_flops}M. Our target is to exceed {self.expected_acc}% accuracy while keeping FLOPs <{self.max_flops}M."

E.1.3. Responses

• Initial Strategy (Iteration 0)

```
"reasoning_content":...
```

```
"content":"
```

```
# SUMMARY
```

```
For the initial exploration, we will generate a diverse population covering broad regions of the search space. The strategy emphasizes architectural diversity across all dimensions (resolution, depth, kernel size, expansion ratio) while incorporating fundamental design principles. We will explore various resolution-depth trade-offs, different kernel size distributions, and varied expansion ratio patterns to identify promising architectural characteristics.
```

```
# KEY PATTERNS
```

- **Resolution and Depth Trade-off**: Higher resolutions (224) may require lower depth values to stay within FLOPs limits, while mid-range resolutions (192-208) allow for deeper architectures.
- **Kernel Size Distribution**: A mix of 3x3 and 5x5 kernels is likely optimal, with fewer 7x7 kernels to avoid excessive FLOPs. Stages with more blocks could use smaller kernels.
- **Expansion Ratio Variability**: Lower ratios (3-4) in early stages and higher ratios (6) in later stages may balance efficiency and capacity. Avoid uniformly high ratios across all blocks.
- **Stage-Specific Configurations**: Deeper stages (e.g., stage 2 with 4 blocks) might benefit from larger kernels and higher expansion ratios, while shallower stages (e.g., stage 4 with 2 blocks) should prioritize smaller kernels and lower ratios.

```
# NAVIGATION DIRECTIONS
```

```
## Resolution
```

```
Use 192, 208, and 224 as primary values. Include 176 for baseline efficiency testing but avoid over-reliance. Prioritize 208-224 to align with typical ImageNet model sizes while leaving room for depth adjustments.
```

```

## Depth
Distribute depth values to explore balanced (e.g., [3,3,3,3,3]) and tiered
configurations** (e.g., [3,4,4,3,2]). Avoid extremes like [2,2,2,2,2] (too shallow)
or [4,4,4,4,4] (too deep). Focus on stages 0-2 having higher depth to capture early
features, while stages 3-4 have lower depth to save FLOPs.
## Kernel Size
Prioritize 3x3 and 5x5 kernels for most blocks. Use 7x7 sparingly (e.g.,
1-2 instances in later stages). For example, design architectures where kernel sizes
increase in later stages (e.g., [3,3,5,5,7] for stages 0-4) to capture complex patterns
without overloading FLOPs.
## Expansion Ratio
Use a mix of 3, 4, and 6, with 3-4 in early stages** and 6 in later stages**.
Avoid uniform high ratios (e.g., all 6s) to prevent FLOPs overshoot. For example,
assign [3,3,4,4,6] across stages to gradually increase capacity.
## Constraints/Correlations
- Pair higher resolutions (224) with lower depth values (e.g., [3,3,3,2,2]) to
stay within FLOPs.
- Avoid combining large kernels (7x7) with high expansion ratios (6) in the same
block.
- Ensure total active blocks (sum of depth values) is  $\geq 15$  to maintain sufficient
model capacity.
- Explore stage-specific kernel-expansion correlations (e.g., 5x5 kernels with
expansion ratio 4 in mid-stages)."
```

• Refined Strategy (Iteration $N \geq 1$)

```

"reasoning_content":...
"content":
# SUMMARY
The latest results show that architectures with 1 7x7 kernel in stage 4 and
expansion ratios of 6 in stage 4 achieve ~77.92% accuracy within the target
FLOPs range (310M-330M). The next step is to refine stage 4 configurations (e.g.,
1 7x7 kernel + 6s) while strictly avoiding FLOPs overshoot. Key patterns suggest
that stage 4 must have 1 7x7 kernel with expansion ratios of 6, and stages 0-3
must use 5x5 kernels with expansion ratios of 3-4. The best accuracy is achieved with
[3,3,4,3,3] depth (16 active blocks) and [5,5,5,5,5,5,5,5,5,5,5,5,7,3,3,3,3]
kernel sizes.

# KEY PATTERNS
- Stage 4 Accuracy Gains: Architectures with 1 7x7 kernel in stage 4 and
expansion ratios of 6 in stage 4 blocks achieve ~77.92% accuracy.
- Kernel Size Efficiency: 14-15 5x5 kernels in stages 0-3 with 1 7x7 kernel
in stage 4 balance FLOPs and accuracy.
- Expansion Ratio Stability: [3,3,4,4,6] or [3,4,4,4,6] (stage-wise) provide
~77.9% accuracy without FLOPs spikes.
- Depth Consistency: [3,3,4,3,3] depth (16 active blocks) is critical for FLOPs
control.
- FLOPs Spikes: Architectures with 2 7x7 kernels in stage 4 push FLOPs over the
limit but reach ~78.04% accuracy.

# NAVIGATION DIRECTIONS
## Resolution
- Use 208 as the primary resolution. Ensure 'r' is a list with one value** (e.g.,
[208]).
- Avoid 224 unless paired with depth [3,3,3,2,2] and 1 7x7 kernel in stage 4.
## Depth
```

```

- Target **[3,3,4,3,3]** (16 active blocks) or **[3,3,3,3,4]** (16 active blocks).
- Ensure **'d' is a list of 5 values** (e.g., [3,3,4,3,3]).
- Avoid **[3,3,3,4,3]** or **[3,4,3,3,3]** as they overconsume FLOPs.
## Kernel Size
- Use **[5,5,5,5,5,5,5,5,5,5,5,5,5,7,3,3,3,3]** (14-15 5x5 kernels, 1 7x7 kernel in stage 4).
- Ensure **'ks' is a list of 20 values**. Avoid **multiple 7x7 kernels in stages 0-3**.
- Focus on **stage 4 kernel size**.
## Expansion Ratio
- Use **[3,3,4,4,6] or [3,4,4,4,6]** (stage-wise).
- Ensure **'e' is a list of 20 values**. Avoid **uniform 6s** (e.g., [3,3,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6]).
- Prioritize **6s in stage 4** and **3-4s in stages 0-3**.
## Constraints/Correlations
- **Stage 4**: 1 7x7 kernel + expansion ratios of 6.
- **Stages 0-3**: 5x5 kernels + expansion ratios of 3-4.
- **Avoid 7x7 in stages 0-3**.
- **Ensure 'e' is included with 20 values**.

# RECOMMENDED PARAMETER RANGES
- **Resolution**: 208 (preferred).
- **Depth**: [3,3,4,3,3], [3,3,3,3,4].
- **Kernel Sizes**: [5,5,5,5,5,5,5,5,5,5,5,5,5,7,3,3,3,3].
- **Expansion Ratios**: [3,3,4,4,6], [3,4,4,4,6]."

```

E.2. Generator LLM

E.2.1. System Prompt

- Role Definition

```
"You are the Generator.LLM, an expert in neural architecture design specializing in generating architectures. You collaborate with a Navigator.LLM that provides high-level guidance to help you explore the search space efficiently."
```

- Collaborative Responsibility

```
"# Your Role in the Collaboration
Your responsibility is to generate concrete neural architectures based on the guidance from the Navigator.LLM. You must follow the guidance closely while ensuring all generated architectures are valid."
```

- Core Mission and Objectives

```
"# Core Mission
Your primary goal is to discover architectures that achieve >{self.expected_acc}% accuracy on ImageNet with FLOPs <{self.max_flops}M, through diversified architecture generation informed by Navigator.LLM's guidance."
```

- Knowledge about search space: *Same as the Navigator LLM.*

- Collaboration Workflow

```
"# Collaboration Process
- You receive guidance from the Navigator.LLM
- You generate diverse and valid architectures that follow the guidance"
```

E.2.2. User Prompt

```
"Here is the strategy from Navigator_LLM: {navigator_strategy}"
```

```
Based on the strategy, please generate 10-20 diverse and valid architectures that meet the requirements.
```

```
Note:
```

- Always make sure the generated architectures are complete and valid. Any deviation will cause evaluation failure.
- Please do not regenerate an architecture that has already been generated and evaluated."

F. Discovered Optimal Architectures

Tab. 8 presents the optimal architectures discovered by CoLLM-NAS within macro search spaces.

Table 8. Optimal architectures discovered by CoLLM-NAS within macro search spaces. * denotes the architecture compared with SOTA.

Method	FLOPs(M)	Architecture Description
OFA-T + Ours	200	Resolution: 176 Depth: [2, 3, 2, 3, 4] Kernel sizes: [3, 3, 5, 3, 5, 3, 3, 5, 7, 7, 7, 7, 7, 5, 3, 7, 3, 3, 3, 7] Expansion ratios: [3, 3, 4, 4, 4, 4, 4, 3, 4, 4, 4, 6, 4, 4, 4, 4, 6, 6, 4, 3]
OFA-S + Ours	297	Resolution: 208 Depth: [3, 3, 3, 3, 4] Kernel sizes: [3, 3, 3, 3, 3, 5, 5, 5, 5, 3, 3, 3, 3, 5, 5, 7, 5, 3, 3, 7, 7] Expansion ratios: [3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 3, 3]
OFA-S + Ours*	320	Resolution: 208 Depth: [3, 3, 4, 3, 3] Kernel sizes: [5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 3, 7, 3, 3, 3, 3] Expansion ratios: [3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 3]
OFA-B + Ours	391	Resolution: 208 Depth: [3, 4, 4, 4, 4] Kernel sizes: [3, 5, 3, 7, 5, 5, 3, 3, 5, 5, 7, 5, 7, 5, 3, 3, 7, 5, 7, 5] Expansion ratios: [3, 3, 3, 3, 3, 4, 6, 3, 6, 4, 3, 4, 6, 4, 4, 4, 6, 6, 6, 3]
OFA-L + Ours	498	Resolution: 224 Depth: [2, 4, 4, 4, 4] Kernel sizes: [7, 5, 3, 5, 5, 5, 3, 7, 5, 7, 3, 3, 3, 7, 7, 3, 5, 3, 3, 5] Expansion ratios: [3, 4, 3, 4, 4, 6, 4, 3, 6, 6, 6, 3, 4, 6, 6, 3, 6, 6, 6, 3]
SPOS + Ours	325	Block operations: [0, 0, 3, 3, 3, 2, 1, 3, 3, 1, 1, 1, 1, 3, 3, 3, 3, 3, 3, 3]
AutoFormer-T + Ours	1366	Layers: 13 MLP ratios: [3.5, 3.5, 3.5, 3.5, 3.5, 3.5, 3.5, 3.5, 3.5, 4.0, 3.5, 4.0, 3.5] Attention heads: [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 3, 4] Embedding dimension: 192
AutoFormer-S + Ours	4897	Layers: 13 MLP ratios: [4.0, 4.0, 3.5, 3.5, 4.0, 4.0, 3.5, 3.5, 4.0, 4.0, 3.5, 3.5, 4.0] Attention heads: [5, 7, 6, 5, 7, 6, 5, 7, 6, 5, 7, 6, 5] Embedding dimension: 384
AutoFormer-B + Ours	11074	Layers: 14 MLP ratios: [3.5, 3.5, 4.0, 3.5, 3.0, 3.5, 4.0, 3.0, 3.5, 4.0, 3.0, 3.5, 3.0, 3.5] Attention heads: [10, 10, 10, 9, 9, 9, 9, 10, 10, 9, 9, 9, 9, 9] Embedding dimension: 576