# On Computational Limits of FlowAR Models: Expressivity and Efficiency

**Yang Cao**[♦] **Chengyue Gong**[♥] **Yekun Ke**[♢] **Xiaoyu Li**[△] **Yingyu Liang**[♣]
**Zhizhou Sha**[♥] **Zhenmei Shi**[♡] **Zhao Song**[♠]
[♥] University of Texas at Austin [♦] Wyoming Seminary [♣] The University of Hong Kong
[♡] University of Wisconsin-Madison [♠] Simons Institute, UC Berkeley
[♢] Independent Researcher [△] University of New South Wales

## Abstract

The expressive power and computational complexity of deep visual generative models, such as flow-based and autoregressive (AR) models, have gained considerable interest for their wide-ranging applications in generative tasks. However, the theoretical characterization of their expressiveness through the lens of circuit complexity remains underexplored, particularly for the state-of-the-art architecture like FlowAR proposed by [Ren et al., 2024], which integrates flow-based and autoregressive mechanisms. This gap limits our understanding of their inherent computational limits and practical efficiency. In this study, we address this gap by analyzing the circuit complexity of the FlowAR architecture. We demonstrate that when the largest feature map produced by the FlowAR model has dimensions $n \times n \times c$, the FlowAR model is simulable by a family of threshold circuits $\mathsf{TC}^0$, which have constant depth $O(1)$ and polynomial width $\mathrm{poly}(n)$. This is the first study to rigorously highlight the limitations in the expressive power of FlowAR models. Furthermore, we identify the conditions under which the FlowAR model computations can achieve almost quadratic time. To validate our theoretical findings, we present efficient model variant constructions based on low-rank approximations that align with the derived criteria. Our work provides a foundation for future comparisons with other generative paradigms and guides the development of more efficient and expressive implementations.

Proceedings of the 29[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. 

## 1 INTRODUCTION

Visual generation has become a transformative force in artificial intelligence, reshaping capabilities in creative design, media synthesis, and digital content creation. Advances in deep generative models, such as Generative Adversarial Networks (GANs) (Goodfellow et al., 2020), Variational Autoencoders (VAEs) (Doersch, 2016), diffusion models (Ho et al., 2020; Song et al., 2020) and flow-based model (Kingma and Dhariwal, 2018), have enabled the synthesis of high-fidelity images, videos, and 3D assets with unprecedented diversity and realism. The introduction of the visual autoregressive model (VAR) (Tian et al., 2024) represents a significant shift in the paradigm in the visual generation field. The VAR model adopts a coarse-to-fine Scale-wise prediction to replace the traditional autoregressive image generation techniques. This innovative technique enables the VAR model to effectively capture visual distributions while outperforming diffusion transformers in image generation benchmarks.

Recently, the introduction of FlowAR (Ren et al., 2024) has further advanced the field of autoregressive visual generation. Specifically, FlowAR streamlines the scale design of VAR, improving generalization for predictions at the next scale and enabling seamless integration with the Flow Matching model (Liu et al., 2023) for high-quality image generation. It is worth noting that FlowAR has achieved cutting-edge results in multiple empirical studies of visual generation.

As the visual generation model architectures grow increasingly sophisticated to meet the demands of high-resolution and photorealistic generation, critical questions arise regarding their computational efficiency and intrinsic expressive power. While empirical im-

provements in generation quality dominate the current discourse, comprehending the theoretical foundations of these models continues to be a pivotal challenge. To tackle the challenge mentioned above, some prior researchers have made significant contributions. For example, (Merrill and Sabharwal, 2024) show that $\mathsf{DLOGTIME}$-uniform $\mathsf{TC}^0$ circuits can simulate softmax-attention transformers; later, (Chen et al., 2024a) show that the introduction of RoPE will not enhance the express power of transformer; (Ke et al., 2025b) present the circuit complexity for the VAR model. Up to now, the expressiveness from a circuit complexity perspective of the FlowAR model remains unexplored. This gap raises an important question:

*Does the Flow Matching architecture enhance the expressive power of the VAR Model?*

This study seeks to explore this question through the lens of circuit complexity. First, we provide a model formulation for each module of FlowAR. Our insight is that using circuit complexity theory, we prove that each module of FlowAR, including the Attention Layer, Flow-Matching Layer, and others, can be simulated by a constant-depth, polynomial-size $\mathsf{TC}^0$ circuit. Ultimately, the combined result shows that the entire FlowAR architecture can be simulated by a constant-depth, polynomial-size $\mathsf{TC}^0$ circuit. Therefore, our conclusion offers a negative response to the question: despite the inclusion of the flow-matching mechanism, the expressive power of FlowAR, in terms of circuit complexity, is on par with that of the VAR model.

In addition, we explored the runtime of the FlowAR model inference process and potential efficient algorithms. Specifically, we analyzed the runtime of each module in the FlowAR model and found that the bottleneck affecting the overall runtime originates from the computation of the attention mechanism. As a result, we accelerated the original attention computation using low-rank approximation, which makes the overall runtime of the FlowAR model almost quadratic.

The primary contributions of our work are summarized below:

- **Circuit Complexity Bound:** FlowAR model can be simulated by a $\mathsf{DLOGTIME}$-uniform $\mathsf{TC}^0$ family. (Theorem 4.9)
- **Provably Efficient Criteria:** Suppose the largest feature map produced by the FlowAR model has dimensions $n \times n \times c$ and $c = O(\log n)$. We prove that the time complexity of the FlowAR model architecture is $O(n^{4+o(1)})$. By applying low-rank approximation to the Attention module within FlowAR, we obtain a FlowAR model with an almost quadratic runtime. Explicitly, we demonstrate that the FlowAR model variant's time complexity in realistic settings is $O(n^{2+o(1)})$. (Theorem 5.8)

**Roadmap.** The paper's organizational structure is outlined as follows: Section 2 synthesizes key academic contributions in the domain. Section 3 then elucidates foundational circuit complexity principles essential for subsequent analysis. Subsequent sections progress systematically, with Section B detailing mathematical formalizations for all FlowAR modules. Section 4 outlines our principal findings. Section 5 presents provably efficient criteria of the fast FlowAR model. In Section 7, we conclude our paper.

## 2 RELATED WORK

**Flow-based and diffusion-based models.** Another line of work focuses on flow-based and diffusion-based models for image and video generation (Ho et al., 2020; Hoogeboom et al., 2023; Li et al., 2024b). The latent diffusion model (LDM) (Rombach et al., 2022) transforms image generation from pixel space to latent space, reducing the computational cost of diffusion-based generative models. This transformation enables these models to scale to larger datasets and model parameters, contributing to the success of LDM. Subsequent works, such as U-ViT (Bao et al., 2023) and DiT (Peebles and Xie, 2023), replace the U-Net architecture with Vision Transformers (ViT) (Dosovitskiy, 2020), leveraging the power of Transformer architectures for image generation. Later models like SiT (Atito et al., 2021) incorporate flow-matching into the diffusion process, further enhancing image generation quality. Many later studies (Esser et al., 2024; Jin et al., 2024; Wang et al., 2024b, 2023, 2024a) have pursued the approach of integrating the strengths of both flow-matching and diffusion models to develop more effective image generation techniques. More related works on flow models and diffusion models can be found in (Hu et al., 2022; Song et al., 2025; Liang et al., 2024d; Li et al., 2024a; Hu et al., 2024b).

**Circuit complexity.** Circuit complexity is a key field in theoretical computer science that explores the computational power of Boolean circuit families. Different circuit complexity classes are used to study machine learning models, aiming to reveal their computational constraints. A significant result related to machine learning is the inclusion chain $\mathsf{AC}^0 \subset \mathsf{TC}^0 \subseteq \mathsf{NC}^1$, although it is still unresolved whether $\mathsf{TC}^0 = \mathsf{NC}^1$ (Vollmer, 1999a; Arora and Barak, 2009). The analysis of circuit complexity limitations has served as a valuable methodology for evaluating the computational ca-

pabilities of diverse neural network structures. Recent investigations have particularly focused on Transformers and their two principal derivatives: Average-Head Attention Transformers (AHATs) and SoftMax-Attention Transformers (SMATs). Research has established that non-uniform threshold circuits operating at constant depth (within $\mathsf{TC}^0$ complexity class) can effectively simulate AHAT implementations (Merrill et al., 2022), with parallel studies demonstrating similar computational efficiency achieved through L-uniform simulations for SMAT architectures (Liu et al., 2022). Subsequent theoretical developments have extended these investigations, confirming that both architectural variants can be effectively approximated using $\mathsf{DLOGTIME}$-uniform $\mathsf{TC}^0$ circuit models (Merrill and Sabharwal, 2024). In addition to standard Transformers, circuit complexity analysis has also been applied to various other frameworks (Chen et al., 2024c; Ke et al., 2025b). Other works related to circuit complexity can be referenced in (Chen et al., 2024a; Li et al., 2024c).

# 3 PRELIMINARY

All notations employed throughout this paper are present in Section 3.1. Section 3.2 introduces circuit complexity axioms. In Section 3.3, we define floating-point numbers and establish the complexity bounds of their operations.

## 3.1 Notations

Given a matrix $X \in \mathbb{R}^{hw \times d}$, we denote its tensorized form as $\mathsf{X} \in \mathbb{R}^{h \times w \times d}$. Additionally, we define the set $[n]$ to represent $\{1, 2, \cdots, n\}$ for any positive integer $n$. We define the set of natural numbers as $\mathbb{N} := \{0, 1, 2, \dots\}$. Let $X \in \mathbb{R}^{m \times n}$ be a matrix, where $X_{i,j}$ refers to the element at the $i$-th row and $j$-th column. When $x_i$ belongs to $\{0,1\}^*$, it signifies a binary number with arbitrary length. In a general setting, $x_i$ represents a length $p$ binary string, with each bit taking a value of either 1 or 0. Given a matrix $X \in \mathbb{R}^{n \times d}$, we define $\|X\|_\infty$ as the maximum norm of $X$. Specifically, $\|X\|_\infty = \max_{i,j} |X_{i,j}|$.

## 3.2 Circuit Complexity Class

Firstly, we present the definition of the boolean circuit.

**Definition 3.1** (Boolean Circuit, (Arora and Barak, 2009))**.** *A Boolean circuit $C_n : \{0,1\}^n \to \{0,1\}$ is formally specified through a directed acyclic graph (DAG) where: Part 1. Nodes represent logic gates from the basis $\{\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}\}$. Part 2. Source nodes (in degree 0) correspond to input Boolean variables $x_1, \dots, x_n$. Part 3. Each non-source gate computes its output by applying its designated Boolean operation to values received via incoming edges.*

Then, we proceed to show the definition of languages related to a specific Boolean circuit.

**Definition 3.2** (Languages, page 103 of (Arora and Barak, 2009))**.** *A language $L \subseteq \{0,1\}^*$ is recognized by a Boolean circuit family $\mathcal{C} = \{C_n\}_{n \in \mathbb{N}}$ if:*

- *The family is parameterized by input length: $C_n$ operates on $n$ Boolean variables.*
- *Membership equivalence: $\forall x \in \{0,1\}^*, C_{|x|}(x) = 1 \iff x \in L$.*
- *Circuit existence: For every string length $n \in \mathbb{N}$, $\mathcal{C}$ contains an appropriate circuit $C_n$.*

Then, we present different language classes that can be recognized by different circuit families. Firstly, we introduce the $\mathsf{NC}^i$ class.

**Definition 3.3** ($\mathsf{NC}^i$ Complexity Class, (Arora and Barak, 2009))**.** *The complexity class $\mathsf{NC}^i$ comprises all languages recognized by Boolean circuit families $\{C_n\}$ satisfying: $\mathsf{Size}(C_n) = O(\mathrm{poly}(n))$. $\mathsf{Depth}(C_n) = O((\log n)^i)$. Gate constraints: (1) $\mathsf{AND}, \mathsf{OR}$ gates have bounded fan-in (2) $\mathsf{NOT}$ gates have unit fan-in.*

$\mathsf{AC}^i$ circuits relax the gate fan-in restriction of $\mathsf{NC}^i$ circuits. We present the definition of $\mathsf{AC}^i$ as the following:

**Definition 3.4** ($\mathsf{AC}^i$ Complexity Class, (Arora and Barak, 2009))**.** *The complexity class $\mathsf{AC}^i$ comprises all languages recognized by Boolean circuit families $\{C_n\}$ satisfying: Part 1. $\mathsf{Size}(C_n) = O(\mathrm{poly}(n))$. Part 2. $\mathsf{Depth}(C_n) = O((\log n)^i)$. Part 3. Gate constraints: (1) $\mathsf{AND}, \mathsf{OR}$ gates have un-bounded fan-in (2) $\mathsf{NOT}$ gates have unit fan-in.*

$\mathsf{TC}^i$ introduces the $\mathsf{MAJORITY}$ gate on top of $\mathsf{AC}^i$. The $\mathsf{MAJORITY}$ gate outputs 1 if more than half of its inputs are 1 and outputs 0 otherwise.

**Definition 3.5** ($\mathsf{TC}^i$ Complexity Class, (Vollmer, 1999b))**.** *The complexity class $\mathsf{TC}^i$ comprises all languages recognized by Boolean circuit families $\{C_n\}$ satisfying: Part 1. $\mathsf{Size}(C_n) = O(\mathrm{poly}(n))$. Part 2. $\mathsf{Depth}(C_n) = O((\log n)^i)$. Part 3. Gate constraints: (1) $\mathsf{AND}, \mathsf{OR}, \mathsf{MAJORITY}$ gates have un-bounded fan-in (3) $\mathsf{NOT}$ gates have unit fan-in.*

In this paper, a boolean circuit that employs $\mathsf{MAJORITY}$ gate is referred to as a threshold circuit.

Then, we present two import definitions $\mathsf{L}$-uniformity and $\mathsf{DLOGTIME}$-uniformity.

**Definition 3.6** (L-uniformity, Definition 6.5 on page 104 of (Arora and Barak, 2009))**.** *Let $\mathcal{C}$ denote a circuit family (e.g., $\mathsf{NC}^i, \mathsf{AC}^i, \mathsf{TC}^i$) that decides a language $\mathsf{C}$. A language $L \subseteq \{0,1\}^*$ belongs to $\mathsf{L}$-uniform $\mathsf{C}$ if there exists a deterministic Turing machine that, on input $1^n$, outputs a circuit $C_n \in \mathcal{C}$ in $O(\log n)$ space, such that $C_n$ recognizes $L$.*

And we present the definition of DLOGTIME-uniformity.

**Definition 3.7** (DLOGTIME-uniformity, (Barrington and Immerman, 1994))**.** *Let $\mathcal{C}$ denote a circuit family (e.g., $\mathsf{NC}^i, \mathsf{AC}^i, \mathsf{TC}^i$) that decides a language $\mathsf{C}$. A language $L \subseteq \{0,1\}^*$ belongs to DLOGTIME-uniform $\mathsf{C}$ if there exists a random access Turing machine that, on input $1^n$, outputs a circuit $C_n \in \mathcal{C}$ in $O(\log n)$ time such that $C_n$ recognizes $L$.*

### 3.3 Circuit Complexity for Floating-Point Operations

In this section, we first introduce the key definitions of floating-point numbers.

**Definition 3.8** (Float point number, (Chiang, 2024))**.** *A $p$-bit floating-point number is a tuple $\langle a, b\rangle$ of integers where Part 1. The significand $a$ satisfies $a \in (-2^p, -2^{p-1}] \cup \{0\} \cup [2^{p-1}, 2^p)$. Part 2. The exponent $b$ lies in the interval $b \in [-2^p, 2^p)$. This represents the real number $a \cdot w^b$. The set of all $p$-bit floating-point numbers is denoted $\mathsf{F}_p$.*

Then, we can show the circuit complexity bounds of some float point number operations.

**Lemma 3.9** (Float point number operations in $\mathsf{TC}^0$, (Chiang, 2024))**.** *Assume the precision $p \leq \mathrm{poly}(n)$. The following hold:*

- **Basic Arithmetic:** *Addition, comparison and multiplication of two $p$-bit floating-point numbers are computable by uniform $\mathcal{TC}^0$ circuits (depth $O(1)$, size $\mathrm{poly}(n)$). Denote by $d_{\mathrm{std}}$ the circuit depth for these operations.*
- **Iterated Multiplication:** *The product of $n$ $p$-bit floating-point numbers is computable by uniform $\mathsf{TC}^0$ circuits (depth $O(1)$, size $\mathrm{poly}(n)$). Denote by $d_{\otimes}$ the required circuit depth.*
- **Iterated Addition:** *The sum of $n$ $p$-bit floating-point numbers is computable by uniform $\mathsf{TC}^0$ circuits (depth $O(1)$, size $\mathrm{poly}(n)$). Denote by $d_{\oplus}$ the required circuit depth.*

**Lemma 3.10** (Exponential Approximation in $\mathsf{TC}^0$, (Chiang, 2024))**.** *Let precision $p \leq \mathrm{poly}(n)$. For every $p$-bit floating-point number $x \in \mathsf{F}_p$, there exists a constant depth uniform $\mathsf{TC}^0$ circuit of size $\mathrm{poly}(n)$ that can compute $\exp(x)$ with relative error bounded by $2^{-p}$. Denote by $d_{\exp}$ the required circuit depth.*

**Lemma 3.11** (Square Root Approximation in $\mathsf{TC}^0$, (Chiang, 2024))**.** *Let precision $p \leq \mathrm{poly}(n)$. For every $p$-bit floating-point number $x \in \mathsf{F}_p$, there exists a constant depth uniform $\mathsf{TC}^0$ circuit of size $\mathrm{poly}(n)$ that can compute $\sqrt{x}$ with relative error bounded by $2^{-p}$. Denote by $d_{\mathrm{sqrt}}$ the required circuit depth.*

## 4 COMPLEXITY OF FLOWAR ARCHITECTURE

This section presents key results on the circuit complexity of fundamental modules in the FlowAR architecture. Section 4.1 analyzes matrix multiplication, while Section 4.2 examines the up-sampling and down-sampling functions. In Sections 4.3 and 4.4, we compute the circuit complexity of the MLP and FFN layers, respectively. Sections 4.5 and 4.6 focus on the single attention layer and layer normalization. Section B.4 addresses the flow-matching layer. Finally, Section 4 presents our main result, establishing the circuit complexity bound for the complete FlowAR architecture.

### 4.1 Computing Matrix Products in $\mathsf{TC}^0$

we demonstrate that matrix multiplication is computable in $\mathsf{TC}^0$, which will be used later.

**Lemma 4.1** (Matrix multiplication in $\mathsf{TC}^0$, (Chen et al., 2024a))**.** *Given the following:*

- *Let the precision $p \leq \mathrm{poly}(n)$.*
- *Let $X \in \mathsf{F}_p^{n_1 \times d}, Y \in \mathsf{F}_p^{d \times n_2}$ be matrices.*
- *Assume $n_1 \leq \mathrm{poly}(n), n_2 \leq \mathrm{poly}(n)$.*

*The matrix product $XY$ can be computed by a uniform $\mathsf{TC}^0$ circuit with:*

- *Size: $\mathrm{poly}(n)$.*
- *Depth: $d_{\mathrm{std}} + d_{\oplus}$. , where $d_{\mathrm{std}}$ and $d_{\oplus}$ are defined in Definition 3.9.*

### 4.2 Computing Down-Sampling and Up-Sampling in in $\mathsf{TC}^0$

In this section, we show that Up-Sampling can be efficiently computable by a uniform $\mathsf{TC}^0$ circuit.

**Lemma 4.2** (Up-Sampling computation in $\mathsf{TC}^0$)**.** *Let $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$ be the input tensor. Let $\phi_{\mathrm{up}}(X, r)$ :*

$\mathbb{R}^{h\times w\times c} \to \mathbb{R}^{(hr)\times(wr)\times c}$ *denote the bicubic up sample function defined in Definition B.1. Assume* $n = h = w$. *Assume* $r \leq n$. *Assume* $c \leq n$. *Assume* $p \leq \mathrm{poly}(n)$. *The linear up sample function can be computed by a uniform* $\mathsf{TC}^0$ *circuit with:*

- *Size:* $\mathrm{poly}(n)$.
- *Depth:* $2d_{\mathrm{std}}+d_{\oplus}$. *, where* $d_{\mathrm{std}}$ *and* $d_{\oplus}$ *are defined in Definition 3.9.*

*Proof.* For each $i \in [nr], j \in [nr], l \in [c]$, we need to compute $\phi_{\mathrm{up}}(\mathsf{X}, r)_{i,j,l} = \sum_{s=-1}^{2}\sum_{t=-1}^{2} W(s)\cdot W(t)\cdot \mathsf{X}_{\frac{i}{r}+s,\frac{j}{r}+s,l}$. We need a $2d_{\mathrm{std}}$ depth and $\mathrm{poly}(n)$ size circuit to compute $W(s)\cdot W(t)\cdot \mathsf{X}_{\frac{i}{r}+s,\frac{j}{r}+s,l}$ by Part 1 of Lemma 3.9 and for all $s,t \in \{-1,0,1,2\}$, these terms can be computed in parallel. Furthermore, by Part 3 of Lemma 3.9, we can need a $d_{\oplus}$ depth and $\mathrm{poly}(n)$ size circuit to compute $\sum_{s=-1}^{2}\sum_{t=-1}^{2} W(s)\cdot W(t)\cdot \mathsf{X}_{\frac{i}{r}+s,\frac{j}{r}+s,l}$. Since the computation of $\phi_{\mathrm{up}}(\mathsf{X}, r)_{i,j,l}$ needs a $2d_{\mathrm{std}} + d_{\oplus}$ depth and $\mathrm{poly}(n)$ size circuit.

Since for all $i \in [nr], j \in [nr], l \in [c]$, we can compute $\phi_{\mathrm{up}}(\mathsf{X}, r)_{i,j,l}$ in parallel, then the total depth of the circuit is $2d_{\mathrm{std}} + d_{\oplus}$ and size remains $\mathrm{poly}(n)$. □

Then, we move forward to consider the down-sampling function.

**Lemma 4.3** (Down-Sampling computation in $\mathsf{TC}^0$)**.** *Let* $\mathsf{X} \in \mathbb{R}^{h\times w\times c}$ *be the input tensor. Let* $\phi_{\mathrm{down}}(X, r)$ *be the linear down sample function from Definition B.2. Assume* $n = h = w$. *Assume* $r \leq n$. *Assume* $c \leq n$. *Assume* $p \leq \mathrm{poly}(n)$.

*The function* $\phi_{\mathrm{down}}$ *can be computed by a uniform* $\mathsf{TC}^0$ *circuit with:*

- *Size:* $\mathrm{poly}(n)$.
- *Depth:* $d_{\mathrm{std}} + d_{\oplus}$. *, where* $d_{\mathrm{std}}$ *and* $d_{\oplus}$ *are defined in Definition 3.9.*

*Proof.* By Definition B.2, we know that down-sampling computation is essentially matrix multiplication. Then, by Lemma 4.1, we can easily get the proof. □

### 4.3 Computing Multiple-layer Perceptron in $\mathsf{TC}^0$

We prove that MLP computation can be efficiently simulated by a uniform $\mathsf{TC}^0$ circuit.

**Lemma 4.4** (MLP computation in $\mathsf{TC}^0$, informal version of Lemma C.1)**.** *Given an input tensor* $\mathsf{X} \in \mathbb{R}^{h\times w\times c}$. *Let* $\mathsf{MLP}(\mathsf{X}, c, d)$ *be the MLP layer defined in Definition B.6. Under the following constraints:* **Part 1.** *Satisfy* $h = w = n$, **Part 2.** *Channel bounds:* $c, d \leq n$, **Part 3.** *Precision:* $p \leq \mathrm{poly}(n)$, *The* $\mathsf{MLP}(\mathsf{X}, c, d)$ *function can be computed by a uniform* $\mathsf{TC}^0$ *circuit with:*

- *Size:* $\mathrm{poly}(n)$.
- *Depth:* $2d_{\mathrm{std}} + d_{\oplus}$, *where* $d_{\mathrm{std}}$ *and* $d_{\oplus}$ *are defined in Definition 3.9.*

### 4.4 Computing Feed-Forward Layer in $\mathsf{TC}^0$

We also prove that feed-forward network computation can be simulated by a uniform $\mathsf{TC}^0$ circuit.

**Lemma 4.5** (FFN computation in $\mathsf{TC}^0$, informal version of Lemma C.2)**.** *Given an input tensor* $\mathsf{X} \in \mathbb{R}^{h\times w\times c}$. *Let* $\mathsf{FFN}(X) : \mathbb{R}^{h\times w\times c} \to \mathbb{R}^{h\times w\times c}$ *as defined in Definition B.7. Under the following constraints:* **Satisfy** $h = w = n$. **Channel bound***:* $c \leq n$. **Precision bound***:* $p \leq \mathrm{poly}(n)$.

*The* $\mathsf{FFN}(\mathsf{X})$ *layer can be computed by a uniform* $\mathsf{TC}^0$ *circuit with:*

- *Size:* $\mathrm{poly}(n)$.
- *Depth:* $6d_{\mathrm{std}} + 2d_{\oplus}$.

### 4.5 Computing Single Attention Layer in $\mathsf{TC}^0$.

We prove the single attention layer can be efficiently simulated by a uniform $\mathsf{TC}^0$ circuit.

**Lemma 4.6** (Attention layer computation in $\mathsf{TC}^0$, informal version of Lemma C.3)**.** *Given an input tensor* $\mathsf{X} \in \mathbb{R}^{h\times w\times c}$. *Let* $\mathsf{Attn}(X) : \mathbb{R}^{h\times w\times c} \to \mathbb{R}^{h\times w\times c}$ *as defined in Definition B.5. Under the following constraints:* **Satisfy** $h = w = n$. **Channel bound***:* $c \leq n$. **Precision bound***:* $p \leq \mathrm{poly}(n)$. *The* $\mathsf{Attn}(\mathsf{X})$ *layer can be computed by a uniform* $\mathsf{TC}^0$ *circuit with:*

- *Size:* $\mathrm{poly}(n)$.
- *Depth:* $6(d_{\mathrm{std}} + d_{\oplus}) + d_{\exp}$.

### 4.6 Computing Layer-wise Norm Layer in $\mathsf{TC}^0$.

We prove that the layer normalization layer can be efficiently simulated by a uniform $\mathsf{TC}^0$ circuit.

**Lemma 4.7** (Layer normalization layer computation in $\mathsf{TC}^0$, informal version of Lemma C.4)**.** *Given an input tensor* $\mathsf{X} \in \mathbb{R}^{h\times w\times c}$. *Let* $\mathsf{LN}(X) : \mathbb{R}^{h\times w\times c} \to \mathbb{R}^{h\times w\times c}$ *as defined in Definition B.8. Under the following constraints:* **Part 1.** *Satisfy* $h = w = n$, **Part 2.** *Channel bound:* $c \leq n$, **Part 3.** *Precision bound:* $p \leq \mathrm{poly}(n)$.

*The* $\mathsf{LN}(\mathsf{X})$ *layer can be computed by a uniform* $\mathsf{TC}^0$ *circuit with:*

- *Size:* $\mathrm{poly}(n)$.
- *Depth:* $5d_{\mathrm{std}} + 2d_{\oplus} + d_{\mathrm{sqrt}}$.

### 4.7 Computing Flow Matching Layer in $\mathsf{TC}^0$.

We prove that the flow-matching layer can be efficiently simulated by a uniform $\mathsf{TC}^0$ circuit.

**Lemma 4.8** (Flow matching layer computation in $\mathsf{TC}^0$)**.** *Given an input tensor* $\mathsf{X} \in \mathbb{R}^{h\times w\times c}$*. Let* $\mathsf{NN}(X)$ *denote the flow-matching layer defined in Definition B.12. Under the following constraints:*

- **Part 1.** *Satisfy* $h = w = n$,
- **Part 2.** *Channel bound:* $c \leq n$,
- **Part 3.** *Precision bound:* $p \leq \mathrm{poly}(n)$.

*The* $\mathsf{NN}(\cdot,\cdot,\cdot)$ *can be computed by a uniform* $\mathsf{TC}^0$ *circuit with*

- *Size:* $\mathrm{poly}(n)$.
- *Depth:* $26d_{\mathrm{std}} + 12d_{\oplus} + 2d_{\mathrm{sqrt}} + d_{\exp}$.

*with* $d_{\mathrm{std}}$ *and* $d_{\oplus}$ *defined in Definition 3.9,* $d_{\exp}$ *defined in Definition 3.10 and* $d_{\mathrm{sqrt}}$ *defined in Definition 3.11.*

*Proof.* **Considering Step 1 in the flow-matching layer:** By Lemma C.1, parameters $\alpha_1$, $\alpha_2$,$\beta_1$,$\beta_2$,$\gamma_1$,$\gamma_2$ are computed via a circuit with: **Part 1.** Depth: $2d_{\mathrm{std}} + d_{\oplus}$. **Part 2.** Size: $\mathrm{poly}(n)$

**Considering Step 2 in flow-matching layer:** By Lemma C.4, $\mathsf{LN}(\mathsf{F}_i^t)$ requires depth $5d_{\mathrm{std}} + 2d_{\oplus} + d_{\mathrm{sqrt}}$. By Lemma 3.9, $A_1 = \gamma_1 \circ \mathsf{LN}(\mathsf{F}_t) + \beta_1$ requires depth $2d_{\mathrm{std}}$. By Lemma C.3, $A_2 = \mathsf{Attn}(A_1)$ requires depth $6(d_{\mathrm{std}}+d_{\oplus})+d_{\exp}$. By Lemma 3.9 again, scaling $A_2 \circ \alpha_1$ requires depth $d_{\mathrm{std}}$. The total depth requires $14d_{\mathrm{std}} + 8d_{\oplus} + d_{\mathrm{sqrt}} + d_{\exp}$ for step 2.

**Considering Step 3 in flow-matching layer:** By Lemma C.4, $\mathsf{LN}(\mathsf{F}_i'^t)$ requires depth $5d_{\mathrm{std}}+2d_{\oplus}+d_{\mathrm{sqrt}}$. By Lemma 3.9, $A_3 = \gamma_2 \circ \mathsf{LN}(\widehat{\mathsf{F}}_t) + \beta_2$ requries depth $2d_{\mathrm{std}}$. By Lemma C.1, $A_4 = \mathsf{MLP}(A_3, c, c)$ requires depth $2d_{\mathrm{std}}+d_{\oplus}$. By Lemma 3.9 again, $A_4 \circ \alpha_2$ requires depth $d_{\mathrm{std}}$. The total depth requires $10d_{\mathrm{std}}+3d_{\oplus}+d_{\mathrm{sqrt}}$ for step 3..

Finally, combining the result above, we need a circuit with depth $26d_{\mathrm{std}}+12d_{\oplus}+2d_{\mathrm{sqrt}}+d_{\exp}$ and size $\mathrm{poly}(n)$ to simulate the flow-matching layer.

□

### 4.8 Circuit Complexity Bound for FlowAR Architecture

We present that the FlowAR Model can be efficiently simulated by a uniform $\mathsf{TC}^0$ circuit.

**Theorem 4.9** (FlowAR Model computation in $\mathsf{TC}^0$)**.** *Given an input tensor* $\mathsf{X} \in \mathbb{R}^{h\times w\times c}$*. Under the following constraints:*

- **Part 1.** *Satisfy* $h = w = n$,
- **Part 2.** *Channel bound:* $c \leq n$,
- **Part 3.** *Precision bound:* $p \leq \mathrm{poly}(n)$.
- **Part 4.** *Number of scales:* $K = O(1)$,
- **Part 5.** $d_{\mathrm{std}}, d_{\oplus}, d_{\mathrm{sqrt}}, d_{\exp} = O(1)$.

*Then, the FlowAR Model can be simulated by a uniform* $\mathsf{TC}^0$ *circuit family.*

*Proof.* For every $i \in [K]$, by Lemma 4.2, Lemma 4.3, Lemma 4.6, Lemma 4.5 and Lemma 4.8, we can simulate the $i$-th layer of FlowAR Model with a uniform $\mathsf{TC}^0$ circuit whose size is $\mathrm{poly}(n)$ and depth is $O(1)$. Since the total number of layers $K = O(1)$, the composition of all $K$ circuits yields a single uniform $\mathsf{TC}^0$ circuit with: **Part 1.** Size: $\mathrm{poly}(n)$. **Part 2.** Depth: $O(1)$. □

In Theorem 4.9, we establish that a FlowAR model with $\mathrm{poly}(n)$ precision, constant depth, and $\mathrm{poly}(n)$ size can be efficiently simulated by a $\mathsf{DLOGTIME}$-uniform $\mathsf{TC}^0$ circuit family. This indicates that while the flow-matching architecture enhances the capability of visual autoregressive models, the FlowAR architecture remains inherently limited in expressivity under circuit complexity theory.

## 5 PROVABLY EFFICIENT CRITERIA

### 5.1 Approximate Attention Computation

In this section, we introduce approximate attention computation, which can accelerate the computation of the attention layer.

**Definition 5.1** (Approximate Attention Computation $\mathsf{AAttC}(n, d, R, \delta)$, Definition 1.2 in (Alman and Song, 2023))**.** *Given an input sequence* $X \in \mathbb{R}^{n\times d}$ *and an approximation tolerance* $\delta > 0$*. Let* $Q, K, V \in \mathbb{R}^{n\times d}$ *be weigh matrices bounded such that*

$$\max\{\|Q\|_\infty, \|K\|_\infty, \|V\|_\infty\} \leq R.$$

*The* **Approximate Attention Computation** $\mathsf{AAttC}(n, d, R, \delta)$ *outputs a matrix* $N \in \mathbb{R}^{n \times d}$ *satisfying*

$$\|N - \mathsf{Attn}(X)\|_\infty \leq \delta$$

Next, we present a lemma that demonstrates the computational time cost of the AATTC method.

**Lemma 5.2** (Fast Attention via Subquadratic Computation, Theorem 1.4 of (Alman and Song, 2023))**.** *Let* $\mathsf{AAttC}$ *be formalized as in Definition 5.1. For parameter configurations: Part 1. Embedding dimension* $d = O(\log n)$*, Part 2.* $R = \Theta(\sqrt{\log n})$*, Part 3. Approximation tolerance* $\delta = 1/\operatorname{poly}(n)$*, the* $\mathsf{AAttC}$ *computation satisfies* $\mathcal{T}(n, n^{o(1)}, d) = n^{1+o(1)}$*, where* $\mathcal{T}$ *denotes the time complexity under these constraints.*

### 5.2 Fast FlowAR Architecture in the Inference Pipleline

Firstly, we define the fast flow-matching layer, where the $\mathsf{Attn}$ layers in the original flow-matching module are replaced with $\mathsf{AAttC}$ layers.

**Definition 5.3** (Fast Flow Matching Architecture)**.** *Given the following:*

- **Input tensor:** $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$.
- **Scales number:** $K \in \mathbb{N}$.
- **Token maps:** *For* $i \in [K]$, $\widehat{\mathsf{Y}}_i \in \mathbb{R}^{(h/r_i) \times (w/r_i) \times c}$ *denote the token maps generated by autoregressive transformer defined in Definition B.9.*
- **Interpolation Tokens:** *For* $i \in [K]$, $\mathsf{F}_i^t \in \mathbb{R}^{(h/r_i) \times (w/r_i) \times c}$ *denote interpolated input defined in Definition B.10.*
- **Time step:** *For* $i \in [K]$, $t_i \in [0, 1]$ *denotes timestep.*
- **Approximate Attention layer:** *For* $i \in [K]$, $\mathsf{AAttC}_i(\cdot) : \mathbb{R}^{h/r_i \times w/r_i \times c} \to \mathbb{R}^{h/r_i \times w/r_i \times c}$ *is defined in Definition B.5.*
- **MLP layer:** *For* $i \in [K]$, $\mathsf{MLP}_i(\cdot, c, d) : \mathbb{R}^{h/r_i \times w/r_i \times c} \to \mathbb{R}^{h/r_i \times w/r_i \times c}$ *is defined in Definition B.6.*
- **LN layer:** *For* $i \in [K]$, $\mathsf{LN}_i(\cdot) : \mathbb{R}^{h/r_i \times w/r_i \times c} \to \mathbb{R}^{h/r_i \times w/r_i \times c}$ *is defined in Definition B.8.*

*The computation steps of flow-matching layers are as follows:*

- **Time-conditioned parameter generation:**

$$\begin{aligned} &\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2 \\ :=\ &\mathsf{MLP}_i(\widehat{\mathsf{Y}}_i + t_i \cdot \mathbf{1}_{(h/r_i) \times (w/r_i) \times c}, c, 6c) \end{aligned}$$

- **Intermediate variable computation:**

$$\mathsf{F}_i'^{t_i} := \mathsf{AAttC}_i(\gamma_1 \circ \mathsf{LN}(\mathsf{F}_i^{t_i}) + \beta_1) \circ \alpha_1$$

  *with* $\circ$ *denoting Hadamard (element-wise) product.*

- **Final projection:**

$$\mathsf{F}_i''^{t_i} := \mathsf{MLP}_i(\gamma_2 \circ \mathsf{LN}(\mathsf{F}_i'^{t_i}) + \beta_2, c, c) \circ \alpha_2$$

*The operation is denoted as* $\mathsf{F}_i''^{t_i} := \mathsf{FNN}_i(\widehat{\mathsf{Y}}_i, \mathsf{F}_i^{t_i}, t_i)$

Next, we define the Fast FlowAR inference pipeline architecture, where all $\mathsf{Attn}$ layers in the original FlowAR architecture are replaced with $\mathsf{AAttC}$ layers.

**Definition 5.4** (Fast FlowAR Inference Architecture)**.** *Given the following:*

- **Scales number:** $K \in \mathbb{N}$.
- **Scale factor:** *For* $i \in [K]$, $r_i := a^{K-i}$ *where base factor* $a \in \mathbb{N}^+$.
- **Upsampling functions:** *For* $i \in [K]$, $\phi_{\mathrm{up},i}(\cdot, a) : \mathbb{R}^{(h/r_i) \times (w/r_i) \times c} \to \mathbb{R}^{(h/r_{i+1}) \times (w/r_{i+1}) \times c}$ *from Definition B.1.*
- **Approximate Attention layer:** *For* $i \in [K]$, $\mathsf{AAttC}_i(\cdot) : \mathbb{R}^{(\sum_{j=1}^i h/r_j \cdot w/r_j) \times c} \to \mathbb{R}^{(\sum_{j=1}^i h/r_j \cdot w/r_j) \times c}$ *which acts on flattened sequences of dimension defined in Definition 5.1.*
- **Feed forward layer:** *For* $i \in [K]$, $\mathsf{FFN}_i(\cdot) : \mathbb{R}^{(\sum_{j=1}^i h/r_j \cdot w/r_j) \times c} \to \mathbb{R}^{(\sum_{j=1}^i h/r_j \cdot w/r_j) \times c}$ *which acts on flattened sequences of dimension defined in Definition B.7.*
- **Fast flow-matching layer:** *For* $i \in [K]$, $\mathsf{FNN}_i(\cdot, \cdot, \cdot) : \mathbb{R}^{h/r_i \times w/r_i \times c} \times \mathbb{R}^{h/r_i \times w/r_i \times c} \times \mathbb{R} \to \mathbb{R}^{h/r_i \times w/r_i \times c}$ *denote the fast flow-matching layer defined in Definition 5.3.*
- **Initial condition:** $\mathsf{Z}_{\mathrm{init}} \in \mathbb{R}^{(h/r_1) \times (w/r_1) \times c}$ *denotes the initial token maps which encodes class information.*
- **Time steps:** *For* $i \in [K]$, $t_i \in [0, 1]$ *denotes time steps.*
- **Interpolated inputs:** *For* $i \in [K]$, $\mathsf{F}_i^{t_i} \in \mathbb{R}^{h/r_i \times w/r_i \times c}$ *defined in Definition B.10.*
- **Cumulative dimensions:** *We define* $\widetilde{h}_i := \sum_{j=1}^i h/r_j$ *and* $\widetilde{w}_i := \sum_{j=1}^i w/r_j$ *for* $i \in [K]$.

*The FlowAR model conducts the following recursive construction:*

- **Base case $i = 1$:**

$$\begin{aligned}\mathsf{Z}_1 &= \mathsf{Z}_{\text{init}},\\ \widehat{\mathsf{Y}}_1 &= \mathsf{FFN}_1(\mathsf{AAttC}_1(\mathsf{Z}_1)),\\ \widetilde{\mathsf{Y}}_1 &= \mathsf{FNN}_1(\widehat{\mathsf{Y}}_1, \mathsf{F}_1^{t_1}, t_1).\end{aligned}$$

- **Inductive step $i \geq 2$:**
  - *Spatial aggregation:*

$$\begin{aligned}&\mathsf{Z}_i\\ =\ &\mathsf{Concat}(\mathsf{Z}_{\text{init}}, \phi_{\text{up},1}(\widetilde{\mathsf{Y}}_{i-1}), \ldots, \phi_{\text{up},i-1}(\widetilde{\mathsf{Y}}_{i-1}))\end{aligned}$$

  - *Autoregressive transformer computation:*

$$\widehat{\mathsf{Y}}_i = \mathsf{FFN}_i(\mathsf{AAttC}_i(\mathsf{Z}_1))_{\widetilde{h}_{i-1}:\widetilde{h}_{i-1},\widetilde{w}_i:\widetilde{w}_i,0:c}$$

  - *Flow matching layer:*

$$\widetilde{\mathsf{Y}}_i = \mathsf{FNN}_i(\widehat{\mathsf{Y}}_i, \mathsf{F}_i^{t_i}, t_i)$$

  *The final output is $\widetilde{\mathsf{Y}}_K \in \mathbb{R}^{h \times w \times c}$.*

### 5.3 Running Time

In this section, we analyzed the running time required by the original FlowAR architecture and the running time required by the Fast FlowAR architecture. The results indicate that by adopting the Approximate Attention computation module, we can accelerate the running time of FlowAR to almost quadratic time.

First, we present the results of the running time analysis for the original FlowAR model.

**Lemma 5.5** (Inference Runtime of Original FlowAR Architecture, informal version of Lemma D.1)**.** *Consider the original FlowAR inference pipeline with the following parameters:*

- **Input tensor:** $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$. *Assume $h = w = n$ and $c = O(\log n)$.*
- **Number of scales:** $K = O(1)$.
- **Scale factor:** *For $i \in [K]$, $r_i := a^{K-i}$ where base factor $a \in \mathbb{N}^+$.*
- **Upsampling functions** *For $i \in [K]$, $\phi_{\text{up},i}(\cdot, a)$ from Definition B.1.*
- **Attention layer:** *For $i \in [K]$, $\mathsf{Attn}_i(\cdot)$ which acts on flattened sequences of dimension defined in Definition B.5.*
- **Feed forward layer:** *For $i \in [K]$, $\mathsf{FFN}_i(\cdot)$ which acts on flattened sequences of dimension defined in Definition B.7.*
- **Flow matching layer:** *For $i \in [K]$, $\mathsf{NN}_i(\cdot, \cdot, \cdot)$ denote the flow-matching layer defined in Definition B.12.*

*Under these conditions, the total inference runtime of FlowAR is bounded by*

$$O(n^{4+o(1)}).$$

Then, we present the results of the running time analysis for the fast FlowAR model.

**Lemma 5.6** (Inference Runtime of Fast FlowAR Architecture, informal version of Lemma D.2)**.** *Consider the original FlowAR inference pipeline with the following parameters:*

- **Input tensor:** $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$. *Assume $h = w = n$ and $c = O(\log n)$.*
- **Number of scales:** $K = O(1)$.
- **Scale factor:** *For $i \in [K]$, $r_i := a^{K-i}$ where base factor $a \in \mathbb{N}^+$.*
- **Upsampling functions** *For $i \in [K]$, $\phi_{\text{up},i}(\cdot, a)$ from Definition B.1.*
- **Approximate Attention layer:** *For $i \in [K]$, $\mathsf{AAttC}_i(\cdot)$ defined in Definition 5.1.*
- **Feed forward layer:** *For $i \in [K]$, $\mathsf{FFN}_i(\cdot)$ which acts on flattened sequences of dimension defined in Definition B.7.*
- **Fast flow-matching layer:** *For $i \in [K]$, $\mathsf{FNN}_i(\cdot, \cdot, \cdot)$ denote the fast flow-matching layer defined in Definition 5.3.*

*Under these conditions, the total inference runtime of FlowAR is bounded by*

$$O(n^{2+o(1)}).$$

### 5.4 Error Propagation Analysis

In this section, we present an error analysis introduced by the fast algorithm applied to the FlowAR model.

**Lemma 5.7** (Error Bound Between Fast FlowAR and FlowAR Outputs, informal version of Lemma D.8)**.** *Under certain conditions, the $\ell_\infty$ error between the final outputs is bounded by*

$$\|\widetilde{\mathsf{Y}}'_K - \widetilde{\mathsf{Y}}_K\|_\infty \leq 1/\operatorname{poly}(n).$$

### 5.5 Existence of Almost Quadratic Time Algorithm

This section presents a theorem proving the existence of a quadratic-time algorithm that speeds up the FlowAR architecture and guarantees a bounded additive error.

**Theorem 5.8** (Existence of Almost Quadratic Time Algorithm)**.** *Suppose $d = O(\log n)$ and $R = o(\sqrt{\log n})$. There is an algorithm that approximates the FlowAR architecture up to $1/\operatorname{poly}(n)$ additive error in $O(n^{2+o(1)})$ time.*

*Proof.* By combining the result of Lemma 5.6 and Lemma 5.7, we can easily derive the proof. □

Our Theorem 5.8 shows that we can accelerate FlowAR while only introducing a small error. Using the low-rank approximation in the attention mechanism is also used in previous works (Ke et al., 2025a; Liang et al., 2024a; Li et al., 2025; Liang et al., 2025; Chen et al., 2024b; Liang et al., 2024b,c; Alman and Song, 2024c,b,a; Hu et al., 2024a).

## 6 DISCUSSION

In this section, we provide a broader perspective on our theoretical results, addressing the implications for future architecture design and the relationship between circuit complexity and empirical performance.

**Theoretical Boundaries and Architectural Evolution**. Our characterization of FlowAR within $\mathsf{TC}^0$ establishes a rigorous computational boundary for constant depth generative models. While the integration of flow matching and autoregressive mechanisms significantly improves visual quality in practice, our analysis reveals that these additions do not shift the model into a more powerful complexity class such as $\mathsf{TC}^1$ or $\mathsf{NC}^1$. This implies that the current success of visual generative models may rely on the inherent flatness of visual data which can be processed within constant depth. Future research aiming to handle more complex, hierarchical, or highly sequential dependencies might require a fundamental shift toward architectures with dynamic depth or looped structures to break the $TC^0$ expressivity ceiling.

**The Scaling Impact of Multi-scale Structures.** The parameter $K$, representing the number of scales, plays a pivotal role in bridging complexity classes. In standard implementations where $K$ is a fixed hyperparameter, the model remains strictly within $\mathsf{TC}^0$. However, our analysis suggests a theoretical phase transition: if $K$ were to scale logarithmically with the input resolution $n$, the model would transition into $\mathsf{TC}^1$. This observation provides a principled roadmap for scaling laws in visual generation. It suggests that as we move toward ultra high resolution synthesis, maintaining constant depth may eventually limit the model's ability to capture long range global structures, necessitating a more flexible approach to architectural depth.

**Bridging Theoretical Efficiency and Practical Systems**. We have identified that low rank approximations can reduce the inference complexity of FlowAR to almost quadratic time while maintaining a provably small error bound. We acknowledge that a gap exists between these circuit theoretic efficiency bounds and current hardware performance. Modern GPUs are heavily optimized for dense matrix operations. Therefore, the practical realization of our $O(n^{2+o(1)})$ variant requires specialized software implementations or hardware accelerators capable of exploiting the sparsity and low rank structures inherent in our proposed approximations. Our work serves as a theoretical foundation for such future engineering efforts, focusing on the fundamental "attainability" of efficiency rather than specific hardware benchmarking.

## 7 CONCLUSION

In this work, we have addressed several fundamental questions about the FlowAR architecture, making significant contributions to both theoretical understanding and complexity efficiency. By rigorously analyzing the architecture of FlowAR, we demonstrated that despite its sophisticated integration of flow-based and autoregressive mechanisms, it resides within the complexity class $\mathsf{TC}^0$. Specifically, we proved that each module of FlowAR, including the attention and flow-matching layers, can be simulated by a constant-depth, polynomial-size circuit.

Beyond the circuit theoretical analysis, we identified the computational bottleneck in FlowAR's attention mechanism and developed an efficient variant using low-rank approximation techniques. This optimization achieves nearly quadratic runtime $O(n^{2+o(1)})$, a substantial improvement over the original $O(n^{4+o(1)})$ complexity, while maintaining an error bound of $1/\operatorname{poly}(n)$.

Our findings provide both a theoretical foundation for understanding the computational limits of FlowAR and practical guidelines for implementing more efficient variants, offering valuable insights for future development of generative architectures and establishing a framework for comparisons with other generative paradigms.

## References


Alman, J. and Song, Z. (2023). Fast attention requires bounded entries. *Advances in Neural Information Processing Systems*, 36:63117–63135.

Alman, J. and Song, Z. (2024a). Fast rope attention: Combining the polynomial method and fast fourier transform. *manuscript*.

Alman, J. and Song, Z. (2024b). The fine-grained complexity of gradient computation for training large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Alman, J. and Song, Z. (2024c). How to capture higher-order correlations? generalizing matrix softmax attention to kronecker computation. In *The Twelfth International Conference on Learning Representations*.

Arora, S. and Barak, B. (2009). *Computational complexity: a modern approach*. Cambridge University Press.

Atito, S., Awais, M., and Kittler, J. (2021). Sit: Self-supervised vision transformer. *arXiv preprint arXiv:2104.03602*.

Bao, F., Nie, S., Xue, K., Cao, Y., Li, C., Su, H., and Zhu, J. (2023). All are worth words: A vit backbone for diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22669–22679.

Barrington, D. M. and Immerman, N. (1994). Time, hardware, and uniformity. In *Proceedings of IEEE 9th Annual Conference on Structure in Complexity Theory*. IEEE.

Chen, B., Li, X., Liang, Y., Long, J., Shi, Z., and Song, Z. (2024a). Circuit complexity bounds for rope-based transformer architecture. *arXiv preprint arXiv:2411.07602*.

Chen, Y., Huo, J., Li, X., Liang, Y., Shi, Z., and Song, Z. (2024b). Fast gradient computation for rope attention in almost linear time. *arXiv preprint arXiv:2412.17316*.

Chen, Y., Li, X., Liang, Y., Shi, Z., and Song, Z. (2024c). The computational limits of state-space models and mamba via the lens of circuit complexity. *arXiv preprint arXiv:2412.06148*.

Chiang, D. (2024). Transformers in uniform $tc^0$. *arXiv:2409.13629*.

Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv:1606.05908*.

Dosovitskiy, A. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., et al. (2024). Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. *CACM*, 63(11).

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Hoogeboom, E., Heek, J., and Salimans, T. (2023). simple diffusion: End-to-end diffusion for high resolution images. In *International Conference on Machine Learning*, pages 13213–13232. PMLR.

Hu, H., Song, Z., Tao, R., Xu, Z., Yin, J., and Zhuo, D. (2022). Sublinear time algorithm for online weighted bipartite matching. *arXiv preprint arXiv:2208.03367*.

Hu, J. Y.-C., Su, M., Kuo, E.-J., Song, Z., and Liu, H. (2024a). Computational limits of low-rank adaptation (lora) fine-tuning for transformer models. In *The Thirteenth International Conference on Learning Representations*.

Hu, J. Y.-C., Wu, W., Lee, Y.-C., Huang, Y.-C., Chen, M., and Liu, H. (2024b). On statistical rates of conditional diffusion transformers: Approximation, estimation and minimax optimality. *arXiv preprint arXiv:2411.17522*.

Jin, Y., Sun, Z., Li, N., Xu, K., Jiang, H., Zhuang, N., Huang, Q., Song, Y., Mu, Y., and Lin, Z. (2024). Pyramidal flow matching for efficient video generative modeling. *arXiv preprint arXiv:2410.05954*.

Ke, Y., Li, X., Liang, Y., Sha, Z., Shi, Z., and Song, Z. (2025a). On computational limits and provably efficient criteria of visual autoregressive models: A fine-grained complexity analysis. *arXiv preprint arXiv:2501.04377*.

Ke, Y., Li, X., Liang, Y., Shi, Z., and Song, Z. (2025b). Circuit complexity bounds for visual autoregressive model. *arXiv preprint arXiv:2501.04299*.

Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. *NeurIPS*, 31.

Li, C., Liang, Y., Shi, Z., and Song, Z. (2024a). Exploring the frontiers of softmax: Provable optimization, applications in diffusion model, and beyond. *arXiv preprint arXiv:2405.03251*.

Li, C., Liang, Y., Shi, Z., and Song, Z. (2025). When can we solve the weighted low rank approximation

problem in truly subquadratic time? In *International Conference on Artificial Intelligence and Statistics*.

Li, T., Tian, Y., Li, H., Deng, M., and He, K. (2024b). Autoregressive image generation without vector quantization. *arXiv preprint arXiv:2406.11838*.

Li, X., Liang, Y., Shi, Z., Song, Z., and Wan, M. (2024c). Theoretical constraints on the expressive power of RoPE-based tensor attention transformers. *arXiv preprint arXiv:2412.18040*.

Liang, Y., Liu, H., Shi, Z., Song, Z., Xu, Z., and Yin, J. (2024a). Conv-basis: A new paradigm for efficient attention inference and gradient computation in transformers. *arXiv preprint arXiv:2405.05219*.

Liang, Y., Sha, Z., Shi, Z., Song, Z., and Zhou, Y. (2024b). Multi-layer transformers gradient can be approximated in almost linear time. *arXiv preprint arXiv:2408.13233*.

Liang, Y., Sha, Z., Shi, Z., Song, Z., and Zhou, Y. (2025). Looped relu mlps may be all you need as practical programmable computers. In *International Conference on Artificial Intelligence and Statistics*.

Liang, Y., Shi, Z., Song, Z., and Zhou, Y. (2024c). Tensor attention training: Provably efficient learning of higher-order transformers. *arXiv preprint arXiv:2405.16411*.

Liang, Y., Shi, Z., Song, Z., and Zhou, Y. (2024d). Unraveling the smoothness properties of diffusion models: A gaussian mixture perspective. *arXiv preprint arXiv:2405.16418*.

Liu, B., Ash, J. T., Goel, S., Krishnamurthy, A., and Zhang, C. (2022). Transformers learn shortcuts to automata. *arXiv preprint arXiv:2210.10749*.

Liu, X., Gong, C., and Liu, Q. (2023). Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*.

Merrill, W. and Sabharwal, A. (2024). A logic for expressing log-precision transformers. *Advances in Neural Information Processing Systems*, 36.

Merrill, W., Sabharwal, A., and Smith, N. A. (2022). Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, 10:843–856.

Peebles, W. and Xie, S. (2023). Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205.

Ren, S., Yu, Q., He, J., Shen, X., Yuille, A., and Chen, L.-C. (2024). Flowar: Scale-wise autoregressive image generation meets flow matching. *arXiv preprint arXiv:2412.15205*.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020). Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

Song, Z., Wang, W., Yin, C., and Yin, J. (2025). Fast and efficient matching algorithm with deadline instances. In *The Second Conference on Parsimony and Learning (Proceedings Track)*.

Tian, K., Jiang, Y., Yuan, Z., Peng, B., and Wang, L. (2024). Visual autoregressive modeling: Scalable image generation via next-scale prediction. *arXiv preprint arXiv:2404.02905*.

Vollmer, H. (1999a). *Introduction to circuit complexity: a uniform approach*. Springer Science & Business Media.

Vollmer, H. (1999b). *Introduction to circuit complexity: a uniform approach*. Springer Science & Business Media.

Wang, Y., Chen, Z., Zhong, L., Ding, Z., Sha, Z., and Tu, Z. (2023). Dolfin: Diffusion layout transformers without autoencoder. *arXiv preprint arXiv:2310.16305*.

Wang, Y., Xu, H., Zhang, X., Chen, Z., Sha, Z., Wang, Z., and Tu, Z. (2024a). Omnicontrolnet: Dual-stage integration for conditional image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7436–7448.

Wang, Z., Sha, Z., Ding, Z., Wang, Y., and Tu, Z. (2024b). Tokencompose: Text-to-image diffusion with token-level supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8553–8564.

## Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Not Applicable]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Not Applicable]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Not Applicable]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Not Applicable]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Not Applicable]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Not Applicable]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

# On Computational Limits of FlowAR Models: Expressivity and Efficiency
# Supplementary Materials

**Roadmap.** Section A presents all the notations of this paper. In Section B presents the formal definition of every module of FlowAR. In Section C, we present some missing proofs in Section 4. Section D presents provably efficient criteria of the fast FlowAR model.

## A NOTATIONS

Given a matrix $X \in \mathbb{R}^{hw \times d}$, we denote its tensorized form as $\mathsf{X} \in \mathbb{R}^{h \times w \times d}$. Additionally, we define the set $[n]$ to represent $\{1, 2, \cdots, n\}$ for any positive integer $n$. We define the set of natural numbers as $\mathbb{N} := \{0, 1, 2, \dots\}$. Let $X \in \mathbb{R}^{m \times n}$ be a matrix, where $X_{i,j}$ refers to the element at the $i$-th row and $j$-th column. When $x_i$ belongs to $\{0,1\}^*$, it signifies a binary number with arbitrary length. In a general setting, $x_i$ represents a length $p$ binary string, with each bit taking a value of either 1 or 0. Given a matrix $X \in \mathbb{R}^{n \times d}$, we define $\|X\|_\infty$ as the maximum norm of $X$. Specifically, $\|X\|_\infty = \max_{i,j} |X_{i,j}|$.

## B MODEL FORMULATION FOR FLOWAR ARCHITECTURE

In this section, we provide a mathematical definition for every module of FlowAR. Section B.1 provides the definition of up-sample and down-sample functions. In Section B.2, we mathematically define the VAE tokenizer. Section B.3 presents a mathematical formulation for every module in the autoregressive transformer in FlowAR. Section B.4 provides some important definitions of the flow-matching architecture. In Section B.5, we also provide a rigorous mathematical definition for the overall architecture of the FlowAR Model during the inference process.

### B.1 Sample Function

We define the bicubic upsampling function.

**Definition B.1** (Bicubic Upsampling Function)**.** *Given the following:*

- **Input tensor:** $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$ *where $h, w, c$ represent height, width, and the number of channels, respectively.*
- **Scaling factor:** *A positive integer $r \geq 1$.*
- **Bicubic kernel:** $W : \mathbb{R} \to [0, 1]$

*The bicubic upsampling function $\phi_{\mathrm{up}}(\mathsf{X}, r)$ computes an output tensor $\mathsf{Y} \in \mathbb{R}^{rh \times rw \times c}$. For every output position $i \in [rh], j \in [rw], l \in [c]$:*

$$\mathsf{Y}_{i,j,l} = \sum_{s=-1}^{2} \sum_{t=-1}^{2} W(s) \cdot W(t) \cdot \mathsf{X}_{\lfloor \frac{i}{r} \rfloor + s, \lfloor \frac{j}{r} \rfloor + t, l}$$

Next, we define the downsampling function.

**Definition B.2** (Linear Downsampling Function)**.** *Given the following:*

- **Input tensor:** $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$ *where $h, w, c$ represent height, width, and the number of channels, respectively.*

- **Scaling factor:** *A positive integer* $r \geq 1$.

*The linear downsampling function* $\phi_{\text{down}}(\mathsf{X}, r)$ *computes an output tensor* $\mathsf{Y} \in \mathbb{R}^{(h/r)\times(w/r)\times c}$. *Let* $\Phi_{\text{down}} \in \mathbb{R}^{(h/r\cdot w/r)\times hw}$ *denote a linear transformation matrix. Reshape* $\mathsf{X}$ *into the matrix* $X \in \mathbb{R}^{hw\times c}$ *by flattening its spatial dimensions. The output matrix is defined via:*

$$Y = \Phi_{\text{down}} X \in \mathbb{R}^{(h/r\cdot w/r)\times c},$$

*Then reshaped back to* $\mathsf{Y} \in \mathbb{R}^{(h/r)\times(w/r)\times c}$.

### B.2 Multi-Scale Downsampling Tokenizer

Given an input image, the FlowAR model will utilize the VAE to generate latent representation $\mathsf{X} \in \mathbb{R}^{h\times w\times c}$. To meet the requirements of Multi-Scale autoregressive image generation, FlowAR uses a Multi-Scale VAE Tokenizer to downsample $\mathsf{X}$ and generate Token Maps of different sizes.

**Definition B.3** (Multi-Scale Downsampling Tokenizer)**.** *Given the following:*

- **Latent representation tensor:** $\mathsf{X} \in \mathbb{R}^{h\times w\times c}$ *generated by VAE.*
- **Number of scales:** $K \in \mathbb{N}$.
- **Base scaling factor:** *positive integer* $a \geq 1$
- **Downsampling functions:** *For* $i \in [K]$, *define scale-specific factors* $r_i := a^{K-i}$ *and use the linear downsampling function* $\phi_{\text{down}}(\mathsf{X}, r_i)$ *from Definition B.2.*

*Then tokenizer outputs a sequence of token maps* $\{\mathsf{Y}^2, \mathsf{Y}^2, \ldots, \mathsf{Y}^K\}$, *where the* $i$*-th token map is*

$$\mathsf{Y}^i := \phi_{\text{down},i}(\mathsf{X}, r_i) \in \mathbb{R}^{(h/r_i)\times(w/r_i)\times c},$$

*Formally, the tokenizer is defined as*

$$\mathsf{TN}(\mathsf{X}) := \{\mathsf{Y}^1, \ldots, \mathsf{Y}^K\}.$$

**Remark B.4.** *In (Ren et al., 2024), the base factor is set to* $a = 2$, *resulting in exponentially increasing scales* $r_i = 2^{K-i}$ *for* $i \in [K]$.

### B.3 Autoregressive Transformer

The autoregressive transformer is a key module of the FlowAR model. We will introduce each layer of autoregressive transformer in this section.

**Definition B.5** (Attention Layer)**.** *Given the following:*

- **Input tensor:** $\mathsf{X} \in \mathbb{R}^{h\times w\times c}$ *where* $h, w, c$ *represent height, width, and the number of channels, respectively.*
- **Weight matrices:** $W_Q, W_K, W_V \in \mathbb{R}^{c\times c}$ *will be used in query, key, and value projection, respectively.*

*The attention layer* $\mathsf{Attn}(\mathsf{X})$ *computes an output tensor* $\mathsf{Y} \in \mathbb{R}^{h\times w\times c}$ *as follows:*

- **Reshape:** *Flatten* $\mathsf{X}$ *into a matrix* $X \in \mathbb{R}^{hw\times c}$ *with spatial dimensions collapsed.*
- **Compute attention matrix:** *For* $i, j \in [hw]$, *compute pairwise scores:*

$$A_{i,j} := \exp(X_{i,*} W_Q W_K^\top X_{j,*}^\top), \quad \text{for } i, j \in [hw].$$

- **Normalization:** *Compute diagnal matrix* $D := \mathrm{diag}(A\mathbf{1}_n) \in \mathbb{R}^{hw\times hw}$, *where* $\mathbf{1}_n$ *is the all-ones vector. And compute:*

$$Y := D^{-1} A X W_V \in \mathbb{R}^{hw\times c}.$$

*then reshape* $Y$ *to* $\mathsf{Y} \in \mathbb{R}^{h\times w\times c}$.

Then, we define the multiple-layer perception layer.

**Definition B.6** (MLP layer)**.** *Given the following:*

- **Input tensor:** $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$ *where* $h, w, c$ *represent height, width, and the number of channels, respectively.*
- **Weight matrices and bias vector:** $W \in \mathbb{R}^{c \times d}$ *and* $b \in \mathbb{R}^{1 \times d}$*.*

*The MLP layer computes an output tensor* $\mathsf{Y} \in \mathbb{R}^{h \times w \times d}$ *as follows:*

- **Reshape:** *Flatten* $\mathsf{X}$ *into a matrix* $X \in \mathbb{R}^{hw \times c}$ *with spatial dimensions collapsed.*
- **Affine transformation:** *For all* $j \in [hw]$*, compute*

$$Y_{j,*} = \underbrace{X_{j,*}}_{1 \times c} \cdot \underbrace{W}_{c \times d} + \underbrace{b}_{1 \times d}$$

  *Then reshape* $Y \in \mathbb{R}^{hw \times d}$ *into* $\mathsf{Y} \in \mathbb{R}^{h \times w \times d}$*.*

*The operation is denoted as* $\mathsf{Y} := \mathsf{MLP}(\mathsf{X}, c, d)$*.*

Next, we introduce the definition of the feedforward layer.

**Definition B.7** (Feed forward layer)**.** *Given the following:*

- **Input tensor:** $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$ *where* $h, w, c$ *represent height, width, and the number of channels, respectively.*
- **Weight matrices and bias vector:** $W_1, W_2 \in \mathbb{R}^{c \times d}$ *and* $b_1, b_2 \in \mathbb{R}^{1 \times d}$*.*
- **Activation:** $\sigma : \mathbb{R} \to \mathbb{R}$ *denotes the* $\mathsf{ReLU}$ *activation function which is applied element-wise.*

*The feedforward layer computes an output tensor* $\mathsf{Y} \in \mathbb{R}^{h \times w \times d}$ *as follows:*

- **Reshape:** *Flatten* $\mathsf{X}$ *into a matrix* $X \in \mathbb{R}^{hw \times c}$ *with spatial dimensions collapsed.*
- **Transform:** *For each* $j \in [hw]$*, compute*

$$Y_{j,*} = \underbrace{X_{j,*}}_{1 \times c} + \sigma(\underbrace{X_{j,*}}_{1 \times c} \cdot \underbrace{W_1}_{c \times c} + \underbrace{b_1}_{1 \times c}) \cdot \underbrace{W_2}_{c \times c} + \underbrace{b_2}_{1 \times c} \in \mathbb{R}^{1 \times c}$$

  *where* $\sigma$ *acts element-wise on intermediate results. Then reshape* $Y \in \mathbb{R}^{hw \times c}$ *into* $\mathsf{Y} \in \mathbb{R}^{h \times w \times c}$*.*

*The operation is denoted as* $\mathsf{Y} := \mathsf{FFN}(\mathsf{X})$*.*

To move on, we define the layer normalization layer.

**Definition B.8** (Layer Normalization Layer)**.** *Given the following:*

- **Input tensor:** $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$ *where* $h, w, c$ *represent height, width, and the number of channels, respectively.*

*The layer normalization computes* $\mathsf{Y}$ *through*

- **Reshape:** *Flatten* $\mathsf{X}$ *into a matrix* $X \in \mathbb{R}^{hw \times c}$ *with spatial dimensions collapsed.*
- **Normalize:** *For each* $j \in [hw]$*, compute*

$$Y_{j,*} = \frac{X_{j,*} - \mu_j}{\sqrt{\sigma_j^2}}$$

  *where*

$$\mu_j := \sum_{k=1}^{c} X_{j,k}/c, \quad \sigma_j^2 = \sum_{k=1}^{c} (X_{j,k} - \mu_j)^2/c$$

  *Then reshape* $Y \in \mathbb{R}^{hw \times c}$ *into* $\mathsf{Y} \in \mathbb{R}^{h \times w \times c}$*.*

*The operation is denoted as* $\mathsf{Y} := \mathsf{LN}(\mathsf{X})$.

Now, we can proceed to show the definition of the autoregressive transformer.

**Definition B.9** (Autoregressive Transformer)**.** *Given the following:*

- **Input tensor:** $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$ *where* $h, w, c$ *represent height, width, and the number of channels, respectively.*
- **Scales number:** $K \in \mathbb{N}$ *denote the number of total scales in FlowAR.*
- **Token maps:** *For* $i \in [K]$, $\mathsf{Y}_i \in \mathbb{R}^{(h/r_i) \times (w/r_i) \times c}$ *generated by the Multi-Scale Downsampling Tokenizer defined in Definition B.3 where* $r_i = a^{K-i}$ *with base* $a \in \mathbb{N}^+$.
- **Upsampling functions:** *For* $i \in [K]$, $\phi_{\mathrm{up},i}(\cdot, a) : \mathbb{R}^{(h/r_i) \times (w/r_i) \times c} \to \mathbb{R}^{(h/r_{i+1}) \times (w/r_{i+1}) \times c}$ *from Definition B.1.*
- **Attention layer:** *For* $i \in [K]$, $\mathsf{Attn}_i(\cdot) : \mathbb{R}^{(\sum_{j=1}^{i} h/r_j \cdot w/r_j) \times c} \to \mathbb{R}^{(\sum_{j=1}^{i} h/r_j \cdot w/r_j) \times c}$ *which acts on flattened sequences of dimension defined in Definition B.5.*
- **Feed forward layer:** *For* $i \in [K]$, $\mathsf{FFN}_i(\cdot) : \mathbb{R}^{(\sum_{j=1}^{i} h/r_j \cdot w/r_j) \times c} \to \mathbb{R}^{(\sum_{j=1}^{i} h/r_j \cdot w/r_j) \times c}$ *which acts on flattened sequences of dimension defined in Definition B.7.*
- **Initial condition:** $\mathsf{Z}_{\mathrm{init}} \in \mathbb{R}^{(h/r_1) \times (w/r_1) \times c}$ *denotes the initial token maps which encodes class information.*

*Then, the autoregressive processing is:*

1. **Initialization:** *Let* $\mathsf{Z}_1 := \mathsf{Z}_{\mathrm{init}}$.
2. **Iterative sequence construction:** *For* $i \geq 2$.
$$\mathsf{Z}_i := \mathsf{Concat}(\mathsf{Z}_{\mathrm{init}}, \phi_{\mathrm{up},1}(\mathsf{Y}^1, a), \ldots, \phi_{\mathrm{up},i-1}(\mathsf{Y}^{i-1}, a)) \in \mathbb{R}^{(\sum_{j=1}^{i} h/r_j \cdot w/r_j) \times c}$$
*where* $\mathsf{Concat}$ *reshapes tokens into a unified spatial grid.*
3. **Transformer block:** *For* $i \in [K]$,
$$\mathsf{TF}_i(\mathsf{Z}_i) := \mathsf{FFN_i}(\mathsf{Attn}_i(\mathsf{Z}_i)) \in \mathbb{R}^{(\sum_{j=1}^{i} h/r_j \cdot w/r_j) \times c}$$
4. **Output decomposition:** *Extract the last scale's dimension from the reshaped* $\mathsf{TF}_i(\mathsf{Z}_i)$ *to generate* $\widehat{\mathsf{Y}}_i \in \mathbb{R}^{(h/r_i) \times (w/r_i) \times c}$.

### B.4 Flow Matching

We begin by outlining the concept of velocity flow in the flow-matching architecture.

**Definition B.10** (Flow)**.** *Given the following:*

- **Input tensor:** $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$ *where* $h, w, c$ *represent height, width, and the number of channels, respectively.*
- **Scales number:** $K \in \mathbb{N}$.
- **Noise tensor:** *For* $i \in [K]$, $\mathsf{F}_i^0 \in \mathbb{R}^{(h/r_i) \times (w/r_i) \times c}$ *with every entry sampled from* $\mathcal{N}(0, 1)$.
- **Token maps:** *For* $i \in [K]$, $\widehat{\mathsf{Y}}_i \in \mathbb{R}^{(h/r_i) \times (w/r_i) \times c}$ *denote the token maps generated by autoregressive transformer defined in Definition B.9.*

*Then, the model does the following:*

- **Interpolation:** *For timestep* $t \in [0, 1]$ *and scale* $i$,
$$\mathsf{F}_i^t := t\widehat{\mathsf{Y}}_i + (1-t)\mathsf{F}_i^0$$
*defining a linear trajectory between noise* $\mathsf{F}_0^i$ *and target tokens* $\widehat{\mathsf{Y}}_i$.

- **Velocity Field:** *The time-derivative of the flow at scale $i$ is*

$$\mathsf{V}_i^t := \frac{\mathrm{d}\mathsf{F}_i^t}{\mathrm{d}t} = \widehat{\mathsf{Y}}_i - \mathsf{F}_i^0.$$

*constant across $t$ due to linear interpolation.*

To move forward, we propose an approach to enhance the performance of the flow-matching layer by replacing linear interpolation with a Quadratic Bézier curve.

**Definition B.11** (High Order Flow)**.** *Given the following:*

- **Input tensor:** $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$ *where $h, w, c$ represent height, width, and the number of channels, respectively.*
- **Scales number:** $K \in \mathbb{N}$.
- **Noise tensor:** *For $i \in [K]$, $\mathsf{F}_i^0 \in \mathbb{R}^{(h/r_i) \times (w/r_i) \times c}$ with every entry sampled from $\mathcal{N}(0,1)$.*
- **Token maps:** *For $i \in [K]$, $\widehat{\mathsf{Y}}_i \in \mathbb{R}^{(h/r_i) \times (w/r_i) \times c}$ denote the token maps generated by autoregressive transformer defined in Definition B.9.*

*Then, the model does the following:*

- **Interpolation:** *For timestep $t \in [0,1]$ and scale $i$,*

$$\mathsf{F}_i^t := (1-t)^2 \mathsf{F}_i^0 + 2t(1-t)\mathsf{C}_i + t^2 \widehat{\mathsf{Y}}_i$$

*defining a quadratic Bézier curve as the interpolation path between the initial noise and the target data. To be noticed, we take $\mathsf{C} = \frac{\mathsf{F}_i^0 + \widehat{\mathsf{Y}}_i}{2}$ as a control point that governs the curvature of the trajectory. This formulation replaces the standard linear interpolation with a higher-order flow, enabling a smoother and more flexible transition from noise to data in the flow-matching framework.*

- **Velocity Field:** *The time-derivative of the flow at scale $i$ is*

$$\begin{aligned} \mathsf{V}_i^t := & \frac{\mathrm{d}\mathsf{F}_i^t}{\mathrm{d}t} \\ = & -2(1-t)\mathsf{F}_i^0 + 2(1-2t)\mathsf{C}_i + 2t\widehat{\mathsf{Y}}_i \end{aligned}$$

*constant across $t$ due to linear interpolation.*

We are now able to define the flow-matching layer, which is integrated in the FlowAR model.

**Definition B.12** (Flow Matching Architecture)**.** *Given the following:*

- **Input tensor:** $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$ *where $h, w, c$ represent height, width, and the number of channels, respectively.*
- **Scales number:** $K \in \mathbb{N}$ *denote the number of total scales in FlowAR.*
- **Token maps:** *For $i \in [K]$, $\widehat{\mathsf{Y}}_i \in \mathbb{R}^{(h/r_i) \times (w/r_i) \times c}$ denote the token maps generated by autoregressive transformer defined in Definition B.9.*
- **Interpolation Tokens:** *For $i \in [K]$, $\mathsf{F}_i^t \in \mathbb{R}^{(h/r_i) \times (w/r_i) \times c}$ denote interpolated input defined in Definition B.10.*
- **Time step:** *For $i \in [K]$, $t_i \in [0,1]$ denotes timestep.*
- **Attention layer:** *For $i \in [K]$, $\mathsf{Attn}_i(\cdot) : \mathbb{R}^{h/r_i \times w/r_i \times c} \to \mathbb{R}^{h/r_i \times w/r_i \times c}$ is defined in Definition B.5.*
- **MLP layer:** *For $i \in [K]$, $\mathsf{MLP}_i(\cdot, c, d) : \mathbb{R}^{h/r_i \times w/r_i \times c} \to \mathbb{R}^{h/r_i \times w/r_i \times c}$ is defined in Definition B.6.*
- **LN layer:** *For $i \in [K]$, $\mathsf{LN}_i(\cdot) : \mathbb{R}^{h/r_i \times w/r_i \times c} \to \mathbb{R}^{h/r_i \times w/r_i \times c}$ is defined in Definition B.8.*

*The computation steps of flow-matching layers are as follows:*

- **Time-conditioned parameter generation:**

$$\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2 := \mathsf{MLP}_i(\widehat{\mathsf{Y}}_i + t_i \cdot \mathbf{1}_{(h/r_i)\times(w/r_i)\times c}, c, 6c)$$

- **Intermediate variable computation:**

$$\mathsf{F}_i'^{t_i} := \mathsf{Attn}_i(\gamma_1 \circ \mathsf{LN}(\mathsf{F}_i^{t_i}) + \beta_1) \circ \alpha_1$$

  *with $\circ$ denoting Hadamard (element-wise) product.*

- **Final projection:**

$$\mathsf{F}_i''^{t_i} := \mathsf{MLP}_i(\gamma_2 \circ \mathsf{LN}(\mathsf{F}_i'^{t_i}) + \beta_2, c, c) \circ \alpha_2$$

*The operation is denoted as* $\mathsf{F}_i''^{t_i} := \mathsf{NN}_i(\widehat{\mathsf{Y}}_i, \mathsf{F}_i^{t_i}, t_i)$

### B.5 Inference of FlowAR Architecture

The inference phase of the FlowAR model differs from the training phase. During inference, neither the VAE nor the Multi-Scale Downsampling layers are used. Instead, given an initial token map representing class embeddings, the model autoregressively generates token maps across scales.

**Definition B.13** (FlowAR Inference Architecture)**.** *Given the following:*

- **Scales number:** $K \in \mathbb{N}$ *denote the number of total scales in FlowAR.*
- **Scale factor:** *For $i \in [K]$, $r_i := a^{K-i}$ where base factor $a \in \mathbb{N}^+$.*
- **Upsampling functions:** *For $i \in [K]$, $\phi_{\mathrm{up},i}(\cdot, a) : \mathbb{R}^{(h/r_i)\times(w/r_i)\times c} \to \mathbb{R}^{(h/r_{i+1})\times(w/r_{i+1})\times c}$ from Definition B.1.*
- **Attention layer:** *For $i \in [K]$, $\mathsf{Attn}_i(\cdot) : \mathbb{R}^{(\sum_{j=1}^i h/r_j \cdot w/r_j)\times c} \to \mathbb{R}^{(\sum_{j=1}^i h/r_j \cdot w/r_j)\times c}$ which acts on flattened sequences of dimension defined in Definition B.5.*
- **Feed forward layer:** *For $i \in [K]$, $\mathsf{FFN}_i(\cdot) : \mathbb{R}^{(\sum_{j=1}^i h/r_j \cdot w/r_j)\times c} \to \mathbb{R}^{(\sum_{j=1}^i h/r_j \cdot w/r_j)\times c}$ which acts on flattened sequences of dimension defined in Definition B.7.*
- **Flow matching layer:** *For $i \in [K]$, $\mathsf{NN}_i(\cdot,\cdot,\cdot) : \mathbb{R}^{h/r_i \times w/r_i \times c} \times \mathbb{R}^{h/r_i \times w/r_i \times c} \times \mathbb{R} \to \mathbb{R}^{h/r_i \times w/r_i \times c}$ denote the flow-matching layer defined in Definition B.12.*
- **Initial condition:** $\mathsf{Z}_{\mathrm{init}} \in \mathbb{R}^{(h/r_1)\times(w/r_1)\times c}$ *denotes the initial token maps which encodes class information.*
- **Time steps:** *For $i \in [K]$, $t_i \in [0,1]$ denotes time steps.*
- **Interpolated inputs:** *For $i \in [K]$, $\mathsf{F}_i^{t_i} \in \mathbb{R}^{h/r_i \times w/r_i \times c}$ defined in Definition B.10.*
- **Cumulative dimensions:** *We define $\widetilde{h}_i := \sum_{j=1}^i h/r_j$ and $\widetilde{w}_i := \sum_{j=1}^i w/r_j$ for $i \in [K]$.*

*The FlowAR model conducts the following recursive construction:*

- **Base case $i = 1$:**

$$\begin{aligned}
\mathsf{Z}_1 &= \mathsf{Z}_{\mathrm{init}} \\
\widehat{\mathsf{Y}}_1 &= \mathsf{FFN}_1(\mathsf{Attn}_1(\mathsf{Z}_1)) \\
\widetilde{\mathsf{Y}}_1 &= \mathsf{NN}_1(\widehat{\mathsf{Y}}_1, \mathsf{F}_1^{t_1}, t_1)
\end{aligned}$$

- **Inductive step $i \geq 2$:**

– **Spatial aggregation:**

$$\mathsf{Z}_i = \mathsf{Concat}(\mathsf{Z}_{\text{init}}, \phi_{\text{up},1}(\widetilde{\mathsf{Y}}_{i-1}), \ldots, \phi_{\text{up},i-1}(\widetilde{\mathsf{Y}}_{i-1})) \in \mathbb{R}^{(\sum_{j=1}^{i} h/r_j \cdot w/r_j) \times c}$$

– **Autoregressive transformer computation:**

$$\widehat{\mathsf{Y}}_i = \mathsf{FFN}_i(\mathsf{Attn}_i(\mathsf{Z}_1))_{\widetilde{h}_{i-1}:\widetilde{h}_{i-1},\widetilde{w}_i:\widetilde{w}_i,0:c}$$

– **Flow matching layer:**

$$\widetilde{\mathsf{Y}}_i = \mathsf{NN}_i(\widehat{\mathsf{Y}}_i, \mathsf{F}_i^{t_i}, t_i)$$

*The final output is* $\widetilde{\mathsf{Y}}_K \in \mathbb{R}^{h \times w \times c}$.

# C SUPPLEMENTARY PROOF FOR SECTION ??

In this section, we present some missing proofs in Section 4.

## C.1 Computing Multiple-layer Perceptron in $\mathsf{TC}^0$

This section presents the detailed proof for Lemma 4.4.

**Lemma C.1** (MLP computation in $\mathsf{TC}^0$, formal version of Lemma 4.4)**.** *Given an input tensor* $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$. *Let* $\mathsf{MLP}(\mathsf{X}, c, d)$ *be the MLP layer defined in Definition B.6. Under the following constraints:*

- *Satisfy* $h = w = n$,
- *Channel bounds:* $c, d \leq n$,
- *Precision:* $p \leq \mathrm{poly}(n)$,

*The* $\mathsf{MLP}(\mathsf{X}, c, d)$ *function can be computed by a uniform* $\mathsf{TC}^0$ *circuit with:*

- *Size:* $\mathrm{poly}(n)$.
- *Depth:* $2d_{\text{std}} + d_{\oplus}$.

*with* $d_{\text{std}}$ *and* $d_{\oplus}$ *defined in Definition 3.9.*

*Proof.* For each $j \in [hw]$, by Lemma 4.1, compute $X_{j,*} \cdot W$ requires depth $d_{\text{std}} + d_{\oplus}$. By Part 1 of Lemma 3.9, compute $X_{j,*} \cdot W + b$ requires depth $d_{\text{std}}$. Since for all $j \in [hw]$, the computation $X_{j,*} \cdot W + b$ can be simulated in parallel. Hence the total depth remains $2d_{\text{std}} + d_{\oplus}$ and size is $\mathrm{poly}(n)$. □

## C.2 Computing Feed Forward Layer in $\mathsf{TC}^0$

This section presents the detailed proof for Lemma 4.5.

**Lemma C.2** (FFN computation in $\mathsf{TC}^0$, formal version of Lemma 4.5)**.** *Given an input tensor* $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$. *Let* $\mathsf{FFN}(X) : \mathbb{R}^{h \times w \times c} \to \mathbb{R}^{h \times w \times c}$ *as defined in Definition B.7. Under the following constraints:*

- *Satisfy* $h = w = n$,
- *Channel bound:* $c \leq n$,
- *Precision bound:* $p \leq \mathrm{poly}(n)$.

*The* $\mathsf{FFN}(\mathsf{X})$ *layer can be computed by a uniform* $\mathsf{TC}^0$ *circuit with:*

- *Size:* $\mathrm{poly}(n)$.
- *Depth:* $6d_{\mathrm{std}} + 2d_{\oplus}$.

*with $d_{\mathrm{std}}$ and $d_{\oplus}$ defined in Definition 3.9.*

*Proof.* For each $j \in [hw]$, by the proof of Lemma C.1, compute $X_{j,*} \cdot W_1 + b_1$ requires depth $2d_{\mathrm{std}} + d_{\oplus}$. By Lemma 3.9, compute $A_1 = \sigma(X_{j,*} \cdot W + b)$ requires depth $d_{\mathrm{std}}$. By applying Lemma C.1 again, compute $A_2 = A_1 \cdot W_2 + b_2$ requires depth $2d_{\mathrm{std}} + d_{\oplus}$. Lastly, by Part 1 of Lemma 3.9, compute $X_{j,*} + A_2$ requires depth $d_{\mathrm{std}}$.

Combing the result above, we can have that compute $Y_{j,*} = X_{j,*} + \sigma(X_{j,*} \cdot W_1 + b_1) \cdot W_2 + b_2$ requires depth $6d_{\mathrm{std}} + 2d_{\oplus}$.

Since for all $j \in [hw]$, the computation $Y_{j,*}$ can be simulated in parallel. Hence the total depth remains $6d_{\mathrm{std}}+2d_{\oplus}$ and size is $\mathrm{poly}(n)$. □

### C.3 Computing Attention Layer in $\mathsf{TC}^0$

This section presents the detailed proof for Lemma 4.6.

**Lemma C.3** (Attention layer computation in $\mathsf{TC}^0$, formal version of Lemma 4.6)**.** *Given an input tensor $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$. Let $\mathsf{Attn}(X) : \mathbb{R}^{h \times w \times c} \to \mathbb{R}^{h \times w \times c}$ as defined in Definition B.5. Under the following constraints:*

- *Satisfy $h = w = n$,*
- *Channel bound: $c \leq n$,*
- *Precision bound: $p \leq \mathrm{poly}(n)$.*

*The $\mathsf{Attn}(\mathsf{X})$ layer can be computed by a uniform $\mathsf{TC}^0$ circuit with:*

- *Size:* $\mathrm{poly}(n)$.
- *Depth:* $6(d_{\mathrm{std}} + d_{\oplus}) + d_{\exp}$.

*with $d_{\mathrm{std}}$ and $d_{\oplus}$ defined in Definition 3.9, $d_{\exp}$ defined in Definition 3.10.*

*Proof.* We analyze the $\mathsf{TC}^0$ simulation of the attention layer through sequential computation phases:

- **Key-Query Product**: Compute $W_Q W_K^\top$ vial Lemma 4.1 requires depth $d_{\mathrm{std}} + d_{\oplus}$.
- **Pairwise Score Computation**: Compute $s_{i,j} = X_{i,*} W_Q W_K^\top X_{j,*}^\top$ requires depth $2(d_{\mathrm{std}}+d_{\oplus})$ by Lemma 4.1. By Lemma 3.10, computing $A_{i,j} = \exp(s_{i,j})$ requires depth $d_{\exp}$.

Since all entries $A_{i,j}$ for $i, j \in [n]$ can be computed in parallel, the attention matrix $A$ computation requires depth $3(d_{\mathrm{std}} + d_{\oplus}) + d_{\exp}$.

Then keep on analyzing:

- **Row Nomalization:** Computing $D := \mathrm{diag}(A\mathbf{1}_n)$ requires depth $d_{\oplus}$ by Lemma 3.9. Computing $D^{-1}$ requires depth $d_{\mathrm{std}}$ by Lemma 3.9 .
- **Value Projection** Computing $AXW_V$ requires depth $2(d_{\mathrm{std}} + d_{\oplus})$ by applying Lemma 4.1. Compute $D^{-1} \cdot AXW_V$ requires $d_{\mathrm{std}}$.

Combing the result, we need a

$$d_{\mathrm{all}} = 6(d_{\mathrm{std}} + d_{\oplus}) + d_{\exp}$$

depth and size $\mathrm{poly}(n)$ uniform $\mathsf{TC}^0$ circuit to compute the attention layer. □

### C.4 Computing Layer-wise Norm Layer in $\mathsf{TC}^0$

This section presents the detailed proof for Lemma 4.7.

**Lemma C.4** (Layer-wise norm layer computation in $\mathsf{TC}^0$, formal version of Lemma 4.7)**.** *Given an input tensor $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$. Let $\mathsf{LN}(X) : \mathbb{R}^{h \times w \times c} \to \mathbb{R}^{h \times w \times c}$ as defined in Definition B.8. Under the following constraints:*

- *Satisfy $h = w = n$,*
- *Channel bound: $c \leq n$,*
- *Precision bound: $p \leq \mathrm{poly}(n)$.*

*The $\mathsf{LN}(\mathsf{X})$ layer can be computed by a uniform $\mathsf{TC}^0$ circuit with:*

- *Size:* $\mathrm{poly}(n)$.
- *Depth:* $5d_{\mathrm{std}} + 2d_{\oplus} + d_{\mathrm{sqrt}}$.

*with $d_{\mathrm{std}}$ and $d_{\oplus}$ defined in Definition 3.9, $d_{\mathrm{sqrt}}$ defined in Definition 3.11.*

*Proof.* By Part 1 and Part 3 of Lemma 3.9, computing mean vector $\mu_j$ requires depth $d_{\mathrm{std}} + d_{\oplus}$. By Part 1 and Part 3 of Lemma 3.9, computing mean vector $\sigma_i^2$ requires depth $2d_{\mathrm{std}} + d_{\oplus}$. By Lemma 3.9 and Lemma 3.11, computing $Y_{j,*}$ requires depth $2d_{\mathsf{std}} + d_{\oplus}$. So the process requires total depth $5d_{\mathrm{std}} + 2d_{\oplus} + d_{\mathrm{sqrt}}$ and $\mathrm{poly}(n)$ size. □

## D PROVABLY EFFICIENT CRITERIA

### D.1 Running Time Analysis for Inference Pipeline of Origin FlowAR Architecture

We proceed to compute the total running time for the inference pipeline of the origin FlowAR architecture.

**Lemma D.1** (Inference Runtime of Original FlowAR Architecture, formal version of Lemma 5.5)**.** *Consider the original FlowAR inference pipeline with the following parameters:*

- **Input tensor:** $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$. *Assume $h = w = n$ and $c = O(\log n)$.*
- **Number of scales:** $K = O(1)$.
- **Scale factor:** *For $i \in [K]$, $r_i := a^{K-i}$ where base factor $a \in \mathbb{N}^+$.*
- **Upsampling functions** *For $i \in [K]$, $\phi_{\mathrm{up},i}(\cdot, a)$ from Definition B.1.*
- **Attention layer:** *For $i \in [K]$, $\mathsf{Attn}_i(\cdot)$ which acts on flattened sequences of dimension defined in Definition B.5.*
- **Feed forward layer:** *For $i \in [K]$, $\mathsf{FFN}_i(\cdot)$ which acts on flattened sequences of dimension defined in Definition B.7.*
- **Flow matching layer:** *For $i \in [K]$, $\mathsf{NN}_i(\cdot, \cdot, \cdot)$ denote the flow-matching layer defined in Definition B.12.*

*Under these conditions, the total inference runtime of FlowAR is bounded by $O(n^{4+o(1)})$.*

*Proof.* **Part 1: Running time of bicubic up-sample Layer.** The $i$-th layer pf FlowAR model contains $\phi_{\mathrm{up},1}(\cdot, 2), \ldots, \phi_{\mathrm{up},i-1}(\cdot, 2)$. Considering $\phi_{\mathrm{up},i-1}(\cdot, 2)$, this operation needs $O(n^2 c / 2^{2(K-i)})$ time. Then the total time required for upsampling in the i-th layer of the FlowAR architecture is $O(n^2 c \cdot \frac{1}{2^{2K}} \cdot (1 - \frac{1}{4^i}))$ which is due to simple algebra. Hence, the total runtime for all bicubic up sample functions is

$$\mathcal{T}_{\mathrm{up}} = \sum_{i=1}^{K} O(n^2 c \cdot \frac{1}{2^{2K}} \cdot (1 - \frac{1}{4^i}))$$

$$= O(n^{2+o(1)})$$

where the first equation is derived from summing up all the running time of the up sample functions, the second step is due to simple algebra and $K = O(1)$ and $c = O(\log n)$.

**Part 2: Running time of Attention Layer.** The input size of the $i$-th attention layer $\mathsf{Attn}_i$ is $\sum_{j=1}^{i}(n/2^{K-j}) \times \sum_{j=1}^{i}(n/2^{K-j}) \times c$. So the time needed to compute the $i$-th attention layer is $O(n^4 c \cdot (2^i - 1)^4 / 2^{4K-4})$. Hence, the total runtime for all attention layers is

$$\begin{aligned} \mathcal{T}_{\mathrm{Attn}} = & \sum_{i=1}^{K} O(n^4 c \cdot (2^i - 1)^4 / 2^{4K-4}) \\ = & O(n^{4+o(1)}) \end{aligned}$$

The first equation is derived from summing up all the running time of the attention layer, the second step is due to simple algebra and $K = O(1)$ and $c = O(\log n)$.

**Part 3: Running time of FFN Layer.** The input size of the $i$-th FFN layer $\mathsf{FFN}_i$ is $\sum_{j=1}^{i}(n/2^{K-j}) \times \sum_{j=1}^{i}(n/2^{K-j}) \times c$. So by Definition B.7, we can easily derive that the time needed to compute the $i-$th FFN layer is $O(n^2 c^2 (2^i - 1)^2 / 2^{2K-2})$. Hence, the total runtime for all FFN layers is

$$\begin{aligned} \mathcal{T}_{\mathrm{FFN}} = & \sum_{i=1}^{K} O(n^2 c^2 (2^i - 1)^2 / 2^{2K-2}) \\ = & O(n^{2+o(1)}) \end{aligned}$$

The first step is derived from summing up all the running time of the FFN layer, and the second step is due to simple algebra and $K = O(1)$ and $c = O(\log n)$.

**Part 4: Running time of Flow Matching Layer.** The input size of the $i$-th flow-matching layer $\mathsf{NN}_i$ is $(n/2^{K-i}) \times (n/2^{K-i}) \times c$. It's trivially that the running time of the flow-matching layer will be dominated by the running time of the attention layer, which is $O(n^4 c / 2^{4(K-i)})$ (see Part 2 of Definition B.12). Hence, the total runtime for all flow-matching layers is

$$\begin{aligned} \mathcal{T}_{\mathsf{FM}} = & \sum_{i=1}^{K} O(n^4 c / 2^{4(K-i)}) \\ = & O(n^{4+o(1)}) \end{aligned}$$

The first step is derived from summing up all the running time of the origin flow-matching layer, and the second step is due to simple algebra and $K = O(1)$ and $c = O(\log n)$.

Then, by summing up Part 1 to Part 4, we can get the total running time for FlowAR architecture, which is

$$\begin{aligned} \mathcal{T}_{\mathrm{ori}} = & \mathcal{T}_{\mathrm{up}} + \mathcal{T}_{\mathrm{Attn}} + \mathcal{T}_{\mathrm{FFN}} + \mathcal{T}_{\mathsf{FM}} \\ = & O(n^{4+o(1)}) \end{aligned}$$

□

Lemma D.1 demonstrates the runtime required for the original FlowAR architecture, from which we can deduce that the dominant term in the runtime comes from the computation of the Attention Layer.

### D.2 Running Time Analysis for Inference Pipeline of Fast FlowAR Architecture

In this section, we apply the conclusions of (Alman and Song, 2023) to the FlowAR architecture, where all Attention modules in FlowAR are computed using the Approximate Attention Computation defined in Definition 5.1.

**Lemma D.2** (Inference Runtime of Fast FlowAR Architecture, formal version of Lemma 5.6)**.** *Consider the original FlowAR inference pipeline with the following parameters:*

- **Input tensor:** $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$. *Assume* $h = w = n$ *and* $c = O(\log n)$.
- **Number of scales:** $K = O(1)$.
- **Scale factor:** *For* $i \in [K]$, $r_i := a^{K-i}$ *where base factor* $a \in \mathbb{N}^+$.
- **Upsampling functions** *For* $i \in [K]$, $\phi_{\mathrm{up},i}(\cdot, a)$ *from Definition B.1.*
- **Approximate Attention layer:** *For* $i \in [K]$, $\mathsf{AAttC}_i(\cdot)$ *defined in Definition 5.1.*
- **Feed forward layer:** *For* $i \in [K]$, $\mathsf{FFN}_i(\cdot)$ *which acts on flattened sequences of dimension defined in Definition B.7.*
- **Fast flow-matching layer:** *For* $i \in [K]$, $\mathsf{FNN}_i(\cdot, \cdot, \cdot)$ *denote the fast flow-matching layer defined in Definition 5.3.*

*Under these conditions, the total inference runtime of FlowAR is bounded by* $O(n^{2+o(1)})$.

*Proof.* **Part 1: Running time of bicubic up-sample Layer.** The runtime of the upsample function in the fast FlowAR architecture is the same as that in the original FlowAR architecture, which is

$$\mathcal{T}_{\mathrm{up}} = O(n^{2+o(1)})$$

**Part 2: Running time of Attention Layer.** The input size of the $i$-th approximate attention computation layer $\mathsf{AAttC}_i$ is $\sum_{j=1}^{i}(n/2^{K-j}) \times \sum_{j=1}^{i}(n/2^{K-j}) \times c$. So the time needed to compute the $i$-th attention layer is $O(n^{2+o(1)} \cdot (2^i - 1)^4 / 2^{4K-4})$. Hence, the total runtime for all attention layers is

$$\begin{aligned} \mathcal{T}_{\mathrm{Attn}} = & \sum_{i=1}^{K} O(n^{2+o(1)} \cdot (2^i - 1)^4 / 2^{4K-4}) \\ = & O(n^{2+o(1)}) \end{aligned}$$

The first equation is derived from summing up all the running time of the approximate attention computation layer, and the second equation is due to basic algebra and $K = O(1)$.

**Part 3: Running time of FFN Layer.** The runtime of the FFN layer in the fast FlowAR architecture is the same as that in the original FlowAR architecture, which is

$$\mathcal{T}_{\mathrm{FFN}} = O(n^{2+o(1)})$$

**Part 4: Running time of Flow Matching Layer.** For each $i \in [K]$, the input size of the $i$-th fast flow-matching layer $\mathsf{FNN}_i$ is $(n/2^{K-i}) \times (n/2^{K-i}) \times c$. By Definition B.6, we can know that the total computational time for the MLP layer is $O(n^{2+o(1)})$, which is due to $c = O(\log n)$. Then by Lemma 5.2, we can speed up the attention computation from $O(n^{4+o(1)})$ to $O(n^{2+o(1)})$. Hence, the total runtime for all flow-matching layers is

$$\begin{aligned} \mathcal{T}_{\mathrm{Attn}} = & \sum_{i=1}^{K} O(n^{2+o(1)}) \\ = & O(n^{2+o(1)}) \end{aligned}$$

The equation is due to $K = O(1)$.

Then, by summing up Part 1 to Part 4, we can get the total running time for fast FlowAR architecture, which is

$$\begin{aligned} \mathcal{T}_{\mathrm{fast}} = & \mathcal{T}_{\mathrm{up}} + \mathcal{T}_{\mathrm{Attn}} + \mathcal{T}_{\mathrm{FFN}} + \mathcal{T}_{\mathsf{FM}} \\ = & O(n^{2+o(1)}) \end{aligned}$$

□

### D.3 Error Analysis of $\mathsf{MLP}(\mathsf{X}')$ and $\mathsf{MLP}(\mathsf{X})$

We conduct the error analysis between $\mathsf{MLP}(\mathsf{X}')$ and $\mathsf{MLP}(\mathsf{X})$ where $\mathsf{X}'$ is the approximation version of $\mathsf{X}$.

**Lemma D.3** (Error analysis of MLP Layer)**.** *If the following conditions hold:*

- *Let* $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$ *denote the input tensor.*
- *Let* $\mathsf{X}' \in \mathbb{R}^{h \times w \times c}$ *denote the approximation version of input tensor* $\mathsf{X}$*.*
- *Let* $\epsilon \in (0, 0.1)$ *denote the approximation error.*
- *Suppose we have* $\|\mathsf{X}' - \mathsf{X}\|_\infty \leq \epsilon$*.*
- *Let* $R > 1$*.*
- *Assume the value of each entry in matrices can be bounded by* $R$*.*
- *Let* $\mathsf{MLP}(\cdot, c, d)$ *denote the MLP layer defined in Definition B.6.*

*We can demonstrate the following*

$$\|\mathsf{MLP}(\mathsf{X}') - \mathsf{MLP}(\mathsf{X})\|_\infty \leq cR\epsilon$$

*Here, we abuse the* $\ell_\infty$ *norm in its tensor form for clarity.*

*Proof.* We can show that for $i \in [h], j \in [w], l \in [c]$, we have

$$\begin{aligned}
\|\mathsf{MLP}(\mathsf{X}', c, d)_{i,j,*} - \mathsf{MLP}(\mathsf{X}, c, d)_{i,j,*}\|_\infty &= \|\mathsf{X}'_{i,j,*} \cdot W - \mathsf{X}_{i,j,*} \cdot W\|_\infty \\
&\leq \| \underbrace{(\mathsf{X}'_{i,j,*} - \mathsf{X}_{i,j,*})}_{1 \times c} \cdot \underbrace{W}_{c \times d} \|_\infty \\
&\leq c \cdot \| \underbrace{(\mathsf{X}'_{i,j,*} - \mathsf{X}_{i,j,*})}_{1 \times c} \|_\infty \cdot \| \underbrace{W}_{c \times d} \|_\infty \\
&\leq c \cdot R \cdot \epsilon
\end{aligned}$$

The first equation is due to Definition B.6, the second inequality is derived from simple algebra, the third inequality is a consequence of basic matrix multiplication, and the last inequality comes from the conditions of this lemma.

Then by the definition of $\ell_\infty$ norm, we can easily get the proof. □

### D.4 Error Analysis of $\mathsf{AAttC}(\mathsf{X}')$ and $\mathsf{Attn}(\mathsf{X})$

We conduct the error analysis between $\mathsf{AAttC}(\mathsf{X}')$ and $\mathsf{Attn}(\mathsf{X})$ where $\mathsf{X}'$ is the approximation version of $\mathsf{X}$.

**Lemma D.4** (Error analysis of $\mathsf{AAttC}(X')$ and $\mathsf{Attn}(X)$, Lemma B.4 of (Ke et al., 2025a))**.** *If the following conditions hold:*

- *Let* $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$ *denote the input tensor.*
- *Let* $\mathsf{X}' \in \mathbb{R}^{h \times w \times c}$ *denote the approximation version of input tensor* $\mathsf{X}$*.*
- *Let* $\epsilon \in (0, 0.1)$ *denote the approximation error.*
- *Suppose we have* $\|\mathsf{X}' - \mathsf{X}\|_\infty \leq \epsilon$*.*
- *Let* $R > 1$*.*
- *Assume the value of each entry in matrices can be bounded by* $R$*.*
- *Let* $\mathsf{Attn}$ *denote the attention layer defined in Definition B.5.*

- *Let* $\mathsf{AAttC}$ *denote the approximated attention layer defined in Definition 5.1.*
- *Let* $U, V \in \mathbb{R}^{hw \times k}$ *be low-rank matrices constructed for polynomial approximation of attention matrix* $\mathsf{AAttC}(\mathsf{X})$.
- *Let* $f$ *be a polynomial with degree* $g$.

*We can demonstrate the following:*

$$\|\mathsf{AAttC}(\mathsf{X}') - \mathsf{Attn}(\mathsf{X})\|_\infty \leq O(kR^{g+1}c) \cdot \epsilon$$

*Here, we abuse the* $\ell_\infty$ *norm in its tensor form for clarity.*

### D.5 Error Analysis of $\mathsf{FFN}(\mathsf{X}')$ and $\mathsf{FFN}(\mathsf{X})$

In this section, we conduct the error analysis between $\mathsf{FFN}(\mathsf{X}')$ and $\mathsf{FFN}(\mathsf{X})$ where $\mathsf{X}'$ is the approximation version of $\mathsf{X}$.

**Lemma D.5** (Error analysis of $\mathsf{FFN}(\mathsf{X}')$ and $\mathsf{FFN}(\mathsf{X})$)**.** *If the following conditions hold:*

- *Let* $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$ *denote the input tensor.*
- *Let* $\mathsf{X}' \in \mathbb{R}^{h \times w \times c}$ *denote the approximation version of input tensor* $\mathsf{X}$.
- *Let* $\epsilon \in (0, 0.1)$ *denote the approximation error.*
- *Suppose we have* $\|\mathsf{X}' - \mathsf{X}\|_\infty \leq \epsilon$.
- *Let* $R > 1$.
- *Assume the value of each entry in matrices can be bounded by* $R$.
- *Let* $\mathsf{FFN}$ *denote the FFN layer defined in Definition B.7.*
- *Let the activation function* $\sigma(\cdot)$ *in* $\mathsf{FFN}$ *be the ReLU activation function.*

*We can demonstrate the following:*

$$\|\mathsf{FFN}(\mathsf{X}') - \mathsf{FFN}(\mathsf{X})\|_\infty \leq O(c^2R^2) \cdot \epsilon$$

*Here, we abuse the* $\ell_\infty$ *norm in its tensor form for clarity.*

*Proof.* Firstly we can bound that for $i \in [h], j \in [w]$

$$\begin{aligned}
\|(\mathsf{X}'_{i,j,*} \cdot W_1 + b_1) - (\mathsf{X}_{i,j,*} \cdot W_1 + b_1)\|_\infty &= \| \underbrace{(\mathsf{X}'_{i,j,*} - \mathsf{X}_{i,j,*})}_{1 \times c} \cdot \underbrace{W_1}_{c \times c} \|_\infty \\
&\leq c \cdot \|\mathsf{X}'_{i,j,*} - \mathsf{X}_{i,j,*}\|_\infty \|W_1\|_\infty \\
&\leq c \cdot \epsilon \cdot R \qquad (1)
\end{aligned}$$

The first equation comes from basic algebra, the second inequality is due to basic matrix multiplication, and the last inequality follows from the conditions of this lemma.

We can show that for $i \in [h], j \in [w]$,

$$\begin{aligned}
&\|\mathsf{FFN}(\mathsf{X}')_{i,j,*} - \mathsf{FFN}(\mathsf{X})_{i,j,*}\|_\infty \\
= &\|\mathsf{X}'_{i,j,*} - \mathsf{X}_{i,j,*} + \underbrace{(\sigma(\mathsf{X}_{i,j,*} \cdot W_1 + b_1) - \sigma(\mathsf{X}'_{i,j,*} \cdot W_1 + b_1))}_{1 \times c} \cdot \underbrace{W_2}_{c \times c} \|_\infty \\
\leq &\|\mathsf{X}'_{i,j,*} - \mathsf{X}_{i,j,*}\|_\infty + c \cdot \|W_2\|_\infty \cdot \|\sigma(\mathsf{X}_{i,j,*} \cdot W_1 + b_1) - \sigma(\mathsf{X}'_{i,j,*} \cdot W_1 + b_1)\|_\infty \\
\leq &\epsilon + cR \cdot \|(\mathsf{X}'_{i,j,*} W_1 + b_1) - (\mathsf{X}_{i,j,*} W_1 + b_1)\|_\infty
\end{aligned}$$

$$\begin{aligned}
&\leq \epsilon + c^2 R^2 \cdot \epsilon \\
&= O(c^2 R^2) \cdot \epsilon
\end{aligned}$$

The first equation is due to Definition B.7, the second step follows from triangle inequality and basic matrix multiplication, the third step follows from the property of ReLU activation function and basic algebra, the fourth step follows from Eq. (1), and the last step follows from simple algebra. □

### D.6 Error Analysis of $\phi_{\mathrm{up}}(\mathsf{X}')$ and $\phi_{\mathrm{up}}(\mathsf{X})$

In this section, we conduct the error analysis between $\phi_{\mathrm{up}}(\mathsf{X}')$ and $\phi_{\mathrm{up}}(\mathsf{X})$ where $\mathsf{X}'$ is the approximation version of $\mathsf{X}$.

**Lemma D.6** (Error Analysis of Up Sample Layer, Lemma B.5 of (Ke et al., 2025a))**.** *If the following conditions hold:*

- *Let $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$ denote the input tensor.*
- *Let $\mathsf{X}' \in \mathbb{R}^{h \times w \times c}$ denote the approximation version of input tensor $\mathsf{X}$.*
- *Let $a = 2$ denote a positive integer.*
- *Let $\phi_{\mathrm{up},i}(\cdot, a)$ be the bicubic up sample function defined in Definition B.1.*
- *Let $\epsilon \in (0, 0.1)$ denote the approximation error.*
- *Let $\|X - X'\|_\infty \leq \epsilon$.*

*Then we have*

$$\|\phi_{\mathrm{up}}(\mathsf{X}', a) - \phi_{\mathrm{up}}(\mathsf{X}, a)\|_\infty \leq O(\epsilon)$$

*Here, we abuse the $\ell_\infty$ norm in its tensor form for clarity.*

### D.7 Error Analysis of $\mathsf{FNN}(\mathsf{F}'^t, \mathsf{X}', t)$ and $\mathsf{NN}(\mathsf{F}^t, \mathsf{X}, t)$

In this section, we conduct the error analysis between $\mathsf{FNN}(\mathsf{F}'^t, \mathsf{X}', t)$ and $\mathsf{NN}(\mathsf{F}^t, \mathsf{X}, t)$ where $\mathsf{X}'$ is the approximation version of $\mathsf{X}$.

**Lemma D.7** (Error Analysis of Flow Matching Layer)**.** *If the following conditions hold:*

- *Let $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$ denote the input tensor.*
- *Let $\mathsf{X}' \in \mathbb{R}^{h \times w \times c}$ denote the approximation version of input tensor $\mathsf{X}$.*
- *Let $\mathsf{F}^t, \mathsf{FF}^t \in \mathbb{R}^{h \times w \times c}$ be the interpolated input defined in Definition B.10.*
- *Let $\mathsf{NN}(\cdot, \cdot, \cdot)$ denote flow-matching layer defined in Definition B.12.*
- *Let $\mathsf{FNN}(\cdot, \cdot, \cdot)$ denote fast flow-matching layer defined in Definition 5.3.*
- *Let $\mathsf{Attn}$ denote the attention layer defined in Definition B.5.*
- *Let $\mathsf{AAttC}$ denote the approximated attention layer defined in Definition 5.1.*
- *Let $R > 1$.*
- *Assume the value of each entry in matrices can be bounded by $R$.*
- *Let $U, V \in \mathbb{R}^{hw \times k}$ be low-rank matrices constructed for polynomial approximation of attention matrix $\mathsf{AAttC}(\mathsf{X})$.*
- *Let $f$ be a polynomial with degree $g$.*

- *Let* $\epsilon \in (0, 0.1)$ *denote the approximation error.*
- *Let* $\|\mathsf{X} - \mathsf{X}'\|_\infty \leq \epsilon$.
- *Let* $t \in [0, 1]$ *denote a time step.*
- *Assume that Layer-wise Norm layer* $\mathsf{LN}(\cdot)$ *defined in Definition B.8 does not exacerbate the propagation of errors, i.e., if* $\|X' - X\|_\infty \leq \epsilon$, *then* $\|\mathsf{LN}(X') - \mathsf{LN}(X)\|_\infty \leq \epsilon$.

*Then we have*

$$\|\mathsf{FNN}(\mathsf{FF}^t, \mathsf{X}', t) - \mathsf{NN}(\mathsf{F}^t, \mathsf{X}, t)\|_\infty \leq O(kR^{g+6}c^3) \cdot \epsilon$$

*Here, we abuse the* $\ell_\infty$ *norm in its tensor form for clarity.*

*Proof.* Firstly, we can show that

$$\|\mathsf{FF}^t - \mathsf{F}^t\|_\infty = \|t(\mathsf{X}' - \mathsf{X})\|_\infty \leq \epsilon$$

The inequality comes from $t \in [0, 1]$ and $\|\mathsf{X}' - \mathsf{X}\|_\infty \leq \epsilon$.

By **Step 1** of Definition B.12 and Definition 5.3, we need to compute

$$\begin{aligned}
\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2 &= \mathsf{MLP}(\mathsf{X} + t \cdot \mathbf{1}_{h \times w \times c}, c, 6c) \\
\alpha_1', \alpha_2', \beta_1', \beta_2', \gamma_1', \gamma_2' &= \mathsf{MLP}(\mathsf{X}' + t \cdot \mathbf{1}_{h \times w \times c}, c, 6c)
\end{aligned}$$

Then, we can show that

$$\|\alpha_1' - \alpha_1\|_\infty \leq cR\epsilon$$

where the step follows from Lemma D.3. The same conclusion holds for the intermediate parameter $\alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2$.

By **Step 2** of Definition B.12 and Definition 5.3, we need to compute

$$\begin{aligned}
\mathsf{F}'^t &= \mathsf{Attn}(\gamma_1 \circ \mathsf{LN}(\mathsf{F}^t) + \beta_1) \circ \alpha_1 \\
\mathsf{FF}'^t &= \mathsf{AAttC}(\gamma_1' \circ \mathsf{LN}(\mathsf{FF}^t) + \beta_1') \circ \alpha_1'
\end{aligned}$$

Then, we move forward to show that

$$\begin{aligned}
& \|\gamma_1' \circ \mathsf{LN}(\mathsf{FF}^t) + \beta_1' - \gamma_1 \circ \mathsf{LN}(\mathsf{F}^t) - \beta_1\|_\infty \\
\leq & \|\gamma_1' \circ \mathsf{LN}(\mathsf{FF}^t) - \gamma_1 \circ \mathsf{LN}(\mathsf{F}^t)\|_\infty + \|\beta_1' - \beta_1\|_\infty \\
\leq & \|\gamma_1' \circ (\mathsf{LN}(\mathsf{FF}^t) - \mathsf{LN}(\mathsf{F}^t))\| + \|(\gamma_1' - \gamma_1) \circ \mathsf{LN}(\mathsf{F}^t)\|_\infty + cR\epsilon \\
\leq & R \cdot \epsilon + R \cdot \epsilon + cR\epsilon \\
= & O(cR) \cdot \epsilon
\end{aligned} \tag{2}$$

where the first and second step follows from triangle inequality, the third step follows from conditions of this Lemma, and the last step follows from simple algebra.

Then we have

$$\begin{aligned}
\|\mathsf{AAttC}(\gamma_1' \circ \mathsf{LN}(\mathsf{FF}^t) + \beta_1') - \mathsf{Attn}(\gamma_1 \circ \mathsf{LN}(\mathsf{F}^t) + \beta_1)\|_\infty &\leq O(kR^{g+1}c) \cdot O(cR) \cdot \epsilon \\
&\leq O(kR^{g+2}c^2)\epsilon
\end{aligned} \tag{3}$$

where the first step follows from Lemma D.4 and Eq. (2) and the second step follows from simple algebra.

Now, we are able to show that

$$\begin{aligned}\|\mathsf{FF}'^t - \mathsf{F}'^t\|_\infty &= \|\mathsf{AAttC}(\gamma_1' \circ \mathsf{LN}(\mathsf{FF}^t) + \beta_1') \circ \alpha_1' - \mathsf{Attn}(\gamma_1 \circ \mathsf{LN}(\mathsf{F}^t) + \beta_1) \circ \alpha_1\|_\infty \\ &\leq \|\mathsf{AAttC}(\gamma_1' \circ \mathsf{LN}(\mathsf{FF}^t) + \beta_1') \circ (\alpha_1' - \alpha_1)\|_\infty \\ &+ \|\alpha_1 \cdot \mathsf{AAttC}(\gamma_1' \circ \mathsf{LN}(\mathsf{FF}^t) + \beta_1') - \mathsf{Attn}(\gamma_1 \circ \mathsf{LN}(\mathsf{F}^t) + \beta_1)\|_\infty \\ &\leq R \cdot cR\epsilon + R \cdot O(kR^{g+2}c^2)\epsilon \\ &= O(kR^{g+3}c^2)\epsilon \end{aligned} \tag{4}$$

where the first step follows from the definition of $\widehat{\mathsf{F}}'^t$ and $\widehat{\mathsf{F}}^t$, the second step follows from triangle inequality, the third step follows from Eq. (3) and the conditions of this lemma, and the last step follows from simple algebra.

By **Step 3** of Definition B.12 and Definition 5.3, we need to compute

$$\begin{aligned}\mathsf{F}''^t &= \mathsf{MLP}(\gamma_2 \circ \mathsf{LN}(\mathsf{F}'^t) + \beta_2, c, c) \circ \alpha_2 \\ \mathsf{FF}''^t &= \mathsf{MLP}(\gamma_2' \circ \mathsf{LN}(\mathsf{FF}'^t) + \beta_2', c, c) \circ \alpha_2'\end{aligned}$$

Then, we move forward to show that

$$\begin{aligned}&\|\gamma_2' \circ \mathsf{LN}(\mathsf{FF}'^t) + \beta_2' - \gamma_2 \circ \mathsf{LN}(\mathsf{F}'^t) - \beta_2\|_\infty \\ \leq\ &\|\gamma_2' \circ \mathsf{LN}(\mathsf{FF}'^t) - \gamma_2 \circ \mathsf{LN}(\mathsf{F}'^t)\|_\infty + \|\beta_1' - \beta_1\|_\infty \\ \leq\ &\|\gamma_2' \circ (\mathsf{LN}(\mathsf{FF}'^t) - \mathsf{LN}(\mathsf{F}'^t))\|_\infty + \|(\gamma_2' - \gamma_2) \circ \mathsf{LN}(\mathsf{F}'^t)\|_\infty + cR\epsilon \\ \leq\ &R \cdot O(kR^{g+3}c^2)\epsilon + cR\epsilon \cdot R + cR\epsilon \\ =\ &O(kR^{g+4}c^2) \cdot \epsilon \end{aligned} \tag{5}$$

where the first and the second steps follow from triangle inequality, the third step follows from Eq. (4) the conditions of this lemma, and the last step follows from simple algebra.

Then, we can show

$$\begin{aligned}\|\mathsf{MLP}(\gamma_2' \circ \mathsf{LN}(\mathsf{FF}'^t) + \beta_2') - \mathsf{MLP}(\gamma_2 \circ \mathsf{LN}(\mathsf{F}'^t) + \beta_2)\|_\infty &\leq cR \cdot O(kR^{g+4}c^2) \cdot \epsilon \\ &= O(kR^{g+5}c^3) \cdot \epsilon \end{aligned} \tag{6}$$

where the first step follows from Lemma D.3 and Eq. (5) and the second step follows from simple algebra.

Finally, we are able to show that

$$\begin{aligned}&\|\mathsf{FNN}(\mathsf{FF}^t, \mathsf{X}', t) - \mathsf{FN}(\mathsf{F}^t, \mathsf{X}, t)\|_\infty \\ =\ &\|\mathsf{MLP}(\gamma_2' \circ \mathsf{LN}(\mathsf{FF}'^t) + \beta_2', c, c) \circ \alpha_2' - \mathsf{MLP}(\gamma_2 \circ \mathsf{LN}(\mathsf{F}'^t) + \beta_2, c, c) \circ \alpha_2\|_\infty \\ \leq\ &\|(\mathsf{MLP}(\gamma_2' \circ \mathsf{LN}(\mathsf{FF}'^t) + \beta_2', c, c) - \mathsf{MLP}(\gamma_2 \circ \mathsf{LN}(\mathsf{F}'^t) + \beta_2, c, c)) \circ \alpha_2'\|_\infty \\ +\ &\|\mathsf{MLP}(\gamma_2 \circ \mathsf{LN}(\mathsf{F}'^t) + \beta_2, c, c) \circ (\alpha_2' - \alpha_2)\|_\infty \\ \leq\ &R \cdot O(kR^{g+5}c^3) \cdot \epsilon + R \cdot cR\epsilon \\ =\ &O(kR^{g+6}c^3) \cdot \epsilon \end{aligned}$$

where the step follows from the definition of output of $\mathsf{FFN}(\mathsf{F}'^t, \mathsf{X}', t)$ and $\mathsf{FN}(\mathsf{F}^t, \mathsf{X}, t)$, the second step follows from triangle inequality, the third step follows from Eq. (6) and conditions of this lemma, and the last step follows from simple algebra.

Then, we complete the proof. □

### D.8 Error Analysis of Fast FlowAR Architecture

Here, we proceed to present the error analysis of fast FlowAR Architecture.

**Lemma D.8** (Error Bound Between Fast FlowAR and FlowAR Outputs)**.** *Given the following:*

- **Input tensor:** $\mathsf{X} \in \mathbb{R}^{h \times w \times c}$.

- **Scales number:** $K = O(1)$.
- **Dimensions:** *Let* $h = w = n$ *and* $c = O(\log n)$. *Let* $\widetilde{h}_i := \sum_{j=1}^{i} h/r_j$ *and* $\widetilde{w}_i := \sum_{j=1}^{i} w/r_j$.
- **Bounded Entries:** *All tensors and matrices have entries bounded by* $R = O(\sqrt{\log n})$.
- **Layers:**
  - $\phi_{\mathrm{up},a}(\cdot)$ *: bicubic upsampling function (Definition B.1).*
  - $\mathsf{Attn}(\cdot)$*: attention layer (Definition B.5).*
  - $\mathsf{AAttC}(\cdot)$*: approximate attention layer (Definition 5.1)*
  - $\mathsf{NN}(\cdot,\cdot,\cdot)$*: flow-matching layer (Definition B.12)*
  - $\mathsf{FNN}(\cdot,\cdot,\cdot)$*: fast flow-matching layer (Definition 5.3)*
- **Input and interpolations:**
  - *Initial inputs:* $\mathsf{Z}_{\mathrm{init}} \in \mathbb{R}^{(h/r_1)\times(w/r_1)\times c}$.
  - $\mathsf{Z}_i$ : *Reshaped tensor of* $\mathsf{Z}_{\mathrm{init}}, \phi_{\mathrm{up},1}(\widetilde{\mathsf{Y}}_1), \ldots, \phi_{\mathrm{up},i-1}(\widetilde{\mathsf{Y}}_{i-1})$ *for FlowAR.*
  - $\mathsf{Z}'_i$ : *Reshaped tensor of* $\mathsf{Z}_{\mathrm{init}}, \phi_{\mathrm{up},1}(\widetilde{\mathsf{Y}}'_1), \ldots, \phi_{\mathrm{up},i-1}(\widetilde{\mathsf{Y}}'_{i-1})$ *for Fast FlowAR.*
  - $\mathsf{F}_i^{t_i} \in \mathbb{R}^{h/r_i \times w/r_i \times c}$ *be the interpolated value of FlowAR (Definition B.10).*
  - $\mathsf{FF}_i^{t_i} \in \mathbb{R}^{h/r_i \times w/r_i \times c}$ *be the interpolated value of Fast FlowAR (Definition B.10).*
- **Outputs:**
  - $\widetilde{\mathsf{Y}}_i \in \mathbb{R}^{h/r_i \times w/r_i \times c}$*: FlowAR output at layer i (Definition B.13)*
  - $\widetilde{\mathsf{Y}}'_i \in \mathbb{R}^{h/r_i \times w/r_i \times c}$*: Fast FlowAR output at layer i (Definition 5.4)*

*Under these conditions, the* $\ell_\infty$ *error between the final outputs is bounded by:*

$$\|\widetilde{\mathsf{Y}}'_K - \widetilde{\mathsf{Y}}_K\|_\infty \leq 1/\operatorname{poly}(n)$$

*Proof.* We can conduct math induction as the following.

Consider the first layer of fast FlowAR Architecture. Firstly, we can show that

$$\|\mathsf{AAttC}_1(\mathsf{Z}_1) - \mathsf{Attn}_1(\mathsf{Z}_1)\|_\infty \leq 1/\operatorname{poly}(n)$$

The inequality is derived Lemma 5.2.

Then, we have

$$\begin{aligned}\|\widehat{\mathsf{Y}}'_1 - \widehat{\mathsf{Y}}_1\|_\infty &= \|\mathsf{FFN}_1(\mathsf{AAttC}_1(\mathsf{Z}_1)) - \mathsf{FFN}_1(\mathsf{Attn}_1(\mathsf{Z}_1))\|_\infty \\ &\leq O(c^2R^2) \cdot 1/\operatorname{poly}(n) \\ &= 1/\operatorname{poly}(n)\end{aligned}$$

The first equation comes from the definition of $\widehat{Y}'_1$ and $\widehat{Y}_1$, the second inequality is due to Lemma D.5 and the last equation is due to $c = O(\log n)$ and $R = O(\sqrt{\log n})$.

Then, we can show that

$$\begin{aligned}\|\widetilde{\mathsf{Y}}'_1 - \widetilde{Y}_1\|_\infty &= \|\mathsf{FNN}_1(\mathsf{FF}_1^{t_1}, \widehat{Y}'_1, t_1) - \mathsf{NN}_1(\mathsf{F}_1^{t_1}, \widehat{Y}_1, t_1)\|_\infty \\ &\leq O(kR^{g+6}c^3) \cdot 1/\operatorname{poly}(n) \\ &= 1/\operatorname{poly}(n)\end{aligned}$$

The first equation is due to the definition of $\mathsf{Y}'_1$ and $\mathsf{Y}_1$, the second inequality comes from Lemma D.7, and the last step follows from $c = O(\log n)$ and $R = O(\sqrt{\log n})$.

Assume that the following statement is true for $k$-th iteration (where $k < K$):

$$\|\widetilde{\mathsf{Y}}'_k - \widetilde{Y}_k\|_\infty \leq 1/\operatorname{poly}(n)$$

Then, we can easily to bound

$$\|\mathsf{Z}'_{k+1} - \mathsf{Z}_{k+1}\|_\infty \leq 1/\operatorname{poly}(n)$$

The inequality is due to Lemma D.6 and Definition of $\mathsf{Z}'_{k+1}$ and $\mathsf{Z}_{k+1}$.

Then, we can show that

$$\begin{aligned}\|\mathsf{AAttC}_{k+1}(\mathsf{Z}'_{k+1}) - \mathsf{Attn}_{k+1}(\mathsf{Z}_{k+1})\|_\infty &\leq O(kR^{g+1}c) \cdot 1/\operatorname{poly}(n) \\ &= 1/\operatorname{poly}(n)\end{aligned}$$

The first inequality comes from Lemma D.4, and the second equation is due to $c = O(\log n)$ and $R = O(\sqrt{\log n})$.

Then we have

$$\begin{aligned}\|\widehat{\mathsf{Y}}'_{k+1} - \widehat{\mathsf{Y}}_{k+1}\|_\infty &= \|\mathsf{FFN}_{k+1}(\mathsf{AAttC}_{k+1}(\mathsf{Z}'_{k+1})) - \mathsf{FFN}_{k+1}(\mathsf{Attn}_{k+1}(\mathsf{Z}_{k+1}))\|_\infty \\ &\leq O(c^2R^2) \cdot 1/\operatorname{poly}(n) \\ &= 1/\operatorname{poly}(n)\end{aligned}$$

The first equation comes from the definition of $\widehat{Y}'_{k+1}$ and $\widehat{Y}_{k+1}$, the second inequality is due to Lemma D.5 and the third equation is due to $c = O(\log n)$ and $R = O(\sqrt{\log n})$.

Then, we can derive that

$$\begin{aligned}\|\widetilde{\mathsf{Y}}'_{k+1} - \widetilde{Y}_{k+1}\|_\infty &= \|\mathsf{FNN}_{k+1}(\mathsf{FF}_{k+1}^{t_{k+1}}, \widehat{Y}'_{k+1}, t_{k+1}) - \mathsf{NN}_{k+1}(\mathsf{F}_{k+1}^{t_{k+1}}, \widehat{Y}_k + 1, t_{k+1})\|_\infty \\ &\leq O(kR^{g+6}c^3) \cdot 1/\operatorname{poly}(n) \\ &= 1/\operatorname{poly}(n)\end{aligned}$$

The first equation comes from the definition of $\mathsf{Y}'_{k+1}$ and $\mathsf{Y}_{k+1}$, the second inequality is due to Lemma D.7 and the third equation is due to $c = O(\log n)$ and $R = O(\sqrt{\log n})$.

Then, by mathematical induction, we can get the proof. □