揭秘 Flux 架构

Demystifying Flux Architecture

Or Greenberg[1,2]

[1]Hebrew University of Jerusalem, Israel
[2]General Motors R&D, SDVR

图 1: Teaser image. 题图。

摘要

FLUX.1 is a diffusion-based text-to-image generation model developed by Black Forest Labs, designed to achieve faithful text-image alignment while maintaining high image quality and diversity. FLUX is considered state-of-the-art in text-to-image generation, outperforming popular models such as Midjourney, DALL·E 3, Stable Diffusion 3 (SD3), and SDXL. Although publicly available as open source, the authors have not released official technical documentation detailing the model's architecture or training setup. This report summarizes an extensive reverse-engineering effort aimed at demystifying FLUX's architecture directly from its source code, to support its adoption as a backbone for future research and

本文为 AI 辅助翻译版本，仅供学习交流；引用请以英文原文为准：arXiv:2507.09595。

development. This document is an *unofficial* technical report and is **not published or endorsed by the original developers or their affiliated institutions**.

FLUX.1 是 Black Forest Labs 开发的一款基于扩散的文生图模型，在保持高图像质量与多样性的同时，力求实现忠实的文本-图像对齐。FLUX 被视为文生图领域的最先进模型，性能超越 Midjourney、DALL·E 3、Stable Diffusion 3（SD3）、SDXL 等热门模型。尽管已作为开源公开，作者却始终未发布详述模型架构或训练配置的官方技术文档。本报告总结了一项系统的逆向工程工作，直接从源代码入手揭示 FLUX 的架构，以支持将其用作未来研究与开发的骨干网络。本文是一份非官方技术报告，**未经原开发者或其所属机构发表或认可**。

Contents

1 Background 背景

To better understand the foundations of FLUX, we briefly present several key models and methods that preceded it and influenced its design. This overview is intentionally high-level, aiming only to provide the essential background necessary to grasp the core principles behind FLUX. For readers interested in a deeper exploration of these foundational works, references to the original papers and technical reports are provided throughout this section.

为帮助读者理解 FLUX 的基础，我们简要介绍几种在它之前出现、并影响其设计的关键模型与方法。这里的梳理刻意保持在较高层面，只求提供把握 FLUX 核心原理所需的必要背景。若读者希望深入了解这些奠基性工作，本节各处均给出了原始论文与技术报告的引用。

1.1 ML-Based Image Generation 基于机器学习的图像生成

The task of ML-based image generation refers to a family of models designed to generate novel samples from a simple distribution that emulates a source dataset. Specifically, some models learn to map a dataset of images to a simple distribution (e.g., Gaussian), from which new images can be sampled. Several types of generative models have been developed in recent years, including—but not limited to—Variational Autoencoders (VAEs) [1], Generative Adversarial Networks (GANs) [2], Normalizing Flows (NFs) [3], and Diffusion Models (DMs) [4].

基于机器学习的图像生成，指的是一类模型：它们从某个模仿源数据集的简单分布中采样出新样本。具体而言，一些模型学习把图像数据集映射到某个简单分布（例如高斯分布），再从中采样得到新图像。近年来出现了多种生成模型，包括但不限于变分自编码器（VAE）[1]、生成对抗网络（GAN）[2]、标准化流（NF）[3] 以及扩散模型（DM）[4]。

Among these, text-to-image models (e.g., DALL·E [5], Stable Diffusion [6], FLUX [7]) have achieved state-of-the-art performance and are widely used in applications that generate high-quality images from textual instructions, commonly referred to as "prompts."

其中，文生图模型（如 DALL·E [5]、Stable Diffusion [6]、FLUX [7]）取得了最先进的性能，被广泛用于根据文本指令生成高质量图像的应用——这类文本指令通常称为“提示词”（prompt）。

1.2 Diffusion Models 扩散模型

Within this scope, Diffusion Models (DMs) [4] have recently emerged as the state-of-the-art (SoTA) approach for text-conditioned image generation. They work by learning to reverse a gradual noising process applied to training data. While early diffusion models relied on U-Net architectures composed of convolutional layers augmented with attention mechanisms for image–text alignment, recent models such as SD3 [8] and FLUX.1 [7] have transitioned to fully Transformer-based architectures for the denoising process, offering improved scalability and context modeling. During the training process, clean images are incrementally corrupted by Gaussian noise whose magnitude is determined by a timestep-specific schedule, and the model learns to iteratively denoise them.

在这一范畴内，扩散模型（DM）[4] 近来成为文本条件图像生成的最先进（SoTA）方法。其工作方式是学习逆转施加在训练数据上的逐步加噪过程。早期扩散模型依赖 U-Net 架构，由卷积层配合注意力机制来实现图像与文本的对齐；而 SD3 [8]、FLUX.1 [7] 等近期模型已将去噪过程转向完全基于 Transformer 的架构，从而获得更好的可扩展性与上下文建模能力。训练时，干净图像被高斯噪声逐步破坏，噪声幅度由与时

间步相关的调度决定，模型则学习对其逐步去噪。

In the most common version of DM optimization, the model is trained to predict the normalized added noise ϵ, and optimized using a reconstruction loss as formulated in 式 (1)

在最常见的扩散模型优化方案中，模型被训练来预测归一化的所加噪声 ϵ，并采用 式 (1) 所给出的重建损失进行优化

$$\mathcal{L}_\epsilon = \mathbb{E}_{x_t,\epsilon,t}\left[\|\epsilon_\theta(x_t,t)-\epsilon\|^2\right] \tag{1}$$

where $x_t = \sqrt{\alpha_t}x + \sqrt{1-\alpha_t}\epsilon$ represent the noisy version of the clean image x_0, corrupted by the gaussian noise $\epsilon \sim \mathcal{N}(0,1)$ to timestep t.

其中 $x_t = \sqrt{\alpha_t}x + \sqrt{1-\alpha_t}\epsilon$ 表示干净图像 x_0 的含噪版本，即在时间步 t 处被高斯噪声 $\epsilon \sim \mathcal{N}(0,1)$ 破坏后的结果。

At inference time, the model begins from pure Gaussian noise and iteratively denoises it to produce realistic images. Unlike earlier generative models such as GANs, which rely on adversarial training, diffusion models optimize a reconstruction-based objective. This leads to more stable training and superior output quality, particularly for high-resolution image synthesis. Moreover, they offer fine-grained control over the generation process, enabling powerful applications such as super-resolution, inpainting, and text-conditioned image generation.

推理时，模型从纯高斯噪声出发，通过迭代去噪生成逼真图像。与 GAN 等依赖对抗训练的早期生成模型不同，扩散模型优化的是基于重建的目标。这带来了更稳定的训练与更优的输出质量，在高分辨率图像合成上尤为明显。此外，扩散模型还能对生成过程进行细粒度控制，从而支撑起超分辨率、图像修复、文本条件图像生成等强大应用。

1.3 Stable Diffusion Stable Diffusion

Stable Diffusion is a family of latent diffusion models (LDMs) [6] designed to generate high-quality images conditioned on textual prompts. Unlike traditional pixel-space diffusion models, LDMs operate in a compressed latent space, significantly improving computational efficiency while maintaining visual fidelity. This is achieved by encoding input images into a lower-dimensional latent representation using a pre-trained autoencoder. The diffusion process is then applied in this latent space, and the final output is decoded back into pixel space.

Stable Diffusion 是一族潜在扩散模型（LDM）[6]，用于在文本提示词条件下生成高质量图像。与传统的像素空间扩散模型不同，LDM 在压缩后的潜在空间中运行，在保持视觉保真度的同时大幅提升计算效率。其做法是先用预训练的自编码器把输入图像编码为低维潜在表示，随后在该潜在空间中执行扩散过程，最终输出再解码回像素空间。

The 1.x versions of Stable Diffusion (1.0 to 1.5) introduced this efficient framework for open-domain image synthesis, using a frozen CLIP [9] text encoder for prompt conditioning and a U-Net architecture for latent denoising. These models, trained on subsets of LAION-2B [10], quickly gained popularity due to their open-source release, high versatility, and ease of fine-tuning.

Stable Diffusion 的 1.x 版本（1.0 至 1.5）为开放域图像合成引入了这一高效框架：用冻结的 CLIP [9]

文本编码器做提示词条件化，用 U-Net 架构做潜在去噪。这些模型在 LAION-2B [10] 的子集上训练，凭借开源发布、通用性强、易于微调等特点迅速流行。

Stable Diffusion v2.1 introduced several architectural upgrades, including a switch to OpenCLIP [11] and training on higher resolutions (768×768), improving image structure and fidelity. SDXL [12], a major advancement in the series, uses a two-stage architecture with separate base and refiner models, allowing better handling of complex prompts and achieving significantly improved realism and prompt alignment. SD-Turbo [13] further innovates by enabling near real-time image generation through a Adversarial-diffusion-distillation (ADD) process, where a student model trains to mimic the performance of a teacher model (SD2.1 or SDXL) using fewer inference steps.

Stable Diffusion v2.1 带来了若干架构升级，包括改用 OpenCLIP [11] 以及在更高分辨率（768×768）上训练，改善了图像结构与保真度。SDXL [12] 是该系列的一次重大进步，采用由独立的 base 模型与 refiner 模型构成的两阶段架构，能更好地处理复杂提示词，并显著提升真实感与提示词对齐。SD-Turbo [13] 则更进一步，通过对抗扩散蒸馏（ADD）实现近实时的图像生成：学生模型学习用更少的推理步数模仿教师模型（SD2.1 或 SDXL）的表现。

Most recently, Stable Diffusion 3 (SD3) [8] integrates a diffusion transformer backbone and adopts multimodal training (see 节 2.3 below), combining both text and image understanding for improved prompt adherence, compositional reasoning, and consistency. SD3 is designed for scalability and robustness, closing the gap between open models and proprietary systems like DALL·E 3 and Midjourney in terms of controllability and quality.

最近，Stable Diffusion 3（SD3）[8] 集成了扩散 Transformer 骨干，并采用多模态训练（见下文 节 2.3），将文本理解与图像理解结合起来，以提升提示词遵循度、组合推理能力与一致性。SD3 面向可扩展性与鲁棒性设计，在可控性与质量上缩小了开放模型与 DALL·E 3、Midjourney 等专有系统之间的差距。

1.4 Rectified Flows 修正流

Rectified Flow (RF) [14] is a recent generative modeling framework that simplifies and generalizes diffusion models by replacing stochastic denoising with a deterministic vector field, based on the Flow Matching principle introduced in [15]. While RF is a training paradigm rather than an architectural change, its relevance to FLUX stems from the fact that FLUX was trained using this method. In this report, we do not provide a detailed explanation of Rectified Flow. Instead, we briefly highlight the key distinction between standard diffusion model training and the approach used in FLUX. This divergence in training schemes likely contributes to differences in performance and convergence behavior. For a comprehensive understanding of Rectified Flow, we refer readers to the original works.

修正流（Rectified Flow，RF）[14] 是近期的一种生成建模框架，它以 [15] 提出的流匹配（Flow Matching）原理为基础，用确定性的向量场取代随机去噪，从而简化并推广了扩散模型。RF 是一种训练范式，而非架构上的改动；它与 FLUX 的相关性在于 FLUX 正是用这种方法训练的。本报告不对修正流展开详细讲解，而只简要指出标准扩散模型训练与 FLUX 所用方法之间的关键区别。训练方案上的这一分歧，很可能导致了性能与收敛行为的差异。若要全面理解修正流，请读者参阅原始工作。

While diffusion models (such as DDPMs [4]) learn to predict the noise (ϵ) added during a forward diffusion process (see 节 1.2), RFs train a model to predict a velocity vector v that directly points from a noisy point x_0 (not to be confused with DM notation, where x_0 denotes the "clean" image) back to the data point x_1, along a straight interpolation path.

扩散模型（如 DDPM [4]）学习预测前向扩散过程中所加的噪声（ϵ，见 节 1.2）；而 RF 训练模型去预

测一个速度向量 v，它沿一条直线插值路径，直接从含噪点 x_0（切勿与扩散模型记号混淆——在扩散模型中 x_0 表示“干净”图像）指向数据点 x_1。

In diffusion models, the network is trained using the ϵ-objective, which minimizes the difference between the predicted noise $\epsilon_\theta(x_t, t)$ and true noise ϵ added to a sample (see 式 (1))

在扩散模型中，网络使用 ϵ 目标训练，即最小化预测噪声 $\epsilon_\theta(x_t, t)$ 与施加到样本上的真实噪声 ϵ 之间的差异（见 式 (1)）

In contrast, RF defines a deterministic interpolation between the data x_1 and noise x_0, and trains the model v_θ to predict the velocity vector between the two:

相比之下，RF 在数据 x_1 与噪声 x_0 之间定义一条确定性插值路径，并训练模型 v_θ 预测二者之间的速度向量：

$$\mathcal{L}_v = \mathbb{E}_{x_0, x_1, t}\left[\|v_\theta(x_t, t) - (x_1 - x_0)\|^2\right] \tag{2}$$

where $x_t = (1-t)x_0 + tx_1$ is a linear interpolation between $x_0 \sim \mathcal{N}(0, I)$ and $x_1 \sim p_{data}$, and $v(x_t, t)$ represents the target vector pointing from x_0 to x_1.

其中 $x_t = (1-t)x_0 + tx_1$ 是 $x_0 \sim \mathcal{N}(0, I)$ 与 $x_1 \sim p_{data}$ 之间的线性插值，$v(x_t, t)$ 表示由 x_0 指向 x_1 的目标向量。

Unlike diffusion models that rely on stochastic sampling, noise schedules, and denoising objectives, Rectified Flow leverages a velocity-based formulation that defines a deterministic transport path from noise to data. This approach eliminates the need for iterative noise perturbation and reverse denoising, enabling faster and more stable image synthesis. By solving a simple ODE with a learned velocity field, Rectified Flow achieves high-quality generation with significantly reduces complexity in both training and inference—making it a compelling and efficient alternative to traditional diffusion-based methods.

扩散模型依赖随机采样、噪声调度与去噪目标，修正流则不同：它采用基于速度的表述，定义了一条从噪声到数据的确定性传输路径。这一做法省去了迭代式的噪声扰动与反向去噪，使图像合成更快、更稳定。借助学到的速度场求解一个简单的常微分方程（ODE），修正流即可实现高质量生成，同时大幅降低训练与推理的复杂度——因而成为传统扩散方法一个颇具吸引力的高效替代方案。

1.5 Transformers Transformer

Transformers are a class of neural network architectures originally introduced for natural language processing (NLP) [16]. The transformer's core innovation is the self-attention mechanism, which enables the model to weigh relationships between all elements in an input sequence. This architecture has since become the foundation of many state-of-the-art models across modalities—including text, image, video, and multimodal systems.

Transformer 是一类神经网络架构，最初为自然语言处理（NLP）而提出 [16]。其核心创新是自注意力机制，使模型能够权衡输入序列中所有元素之间的关系。此后，这一架构成为诸多跨模态最先进模型的基础，涵盖文本、图像、视频以及多模态系统。

In each attention block, the model computes three matrices from the input embeddings: queries (Q), keys (K), and values (V), using learned linear projections. The attention scores are computed as:

在每个注意力块中，模型通过学到的线性投影，从输入嵌入计算出三个矩阵：查询（Q）、键（K）与值（V）。注意力分数按如下方式计算：

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}} \cdot V) \tag{3}$$

Here, d_k is the dimensionality of the key vectors, and the softmax operation ensures the attention weights sum to 1. This mechanism allows the model to assign different levels of importance to different tokens when computing a new representation for each token in the sequence. In practice, multi-head attention is used, where multiple sets of $Q/K/V$ projections are computed in parallel to capture diverse types of relationships.

其中 d_k 是键向量的维度，softmax 操作确保注意力权重之和为 1。该机制让模型在为序列中每个 token 计算新表示时，能对不同 token 赋予不同的重要程度。实践中会采用多头注意力，即并行计算多组 $Q/K/V$ 投影，以捕捉多种类型的关系。

For vision tasks, Vision Transformers (ViTs) [17] tokenize an image into fixed-size non-overlapping patches (e.g., 16×16 pixels), flatten each patch, and project them into an embedding space. These patch embeddings are then processed by a standard transformer encoder, often with added positional encodings to preserve spatial structure. This enables ViTs to model long-range dependencies across an image without convolutional inductive biases.

在视觉任务中，视觉 Transformer（ViT）[17] 把图像切分为固定大小、互不重叠的图块（patch，例如 16×16 像素），将每个图块展平后投影到嵌入空间。随后这些图块嵌入交由标准的 Transformer 编码器处理，通常还会加上位置编码以保留空间结构。由此，ViT 无需卷积的归纳偏置便能建模图像中的长程依赖。

While originally $Q/K/V$ represenations are all projected from the same input embedding in a process denotes *Self-Attention*, In text-to-image generation, transformers often use *Cross-Attention* to condition image synthesis on textual prompts. In this setup, the queries (Q) are projected from the image tokens, while the keys (K) and values (V) are projected from the text embeddings. This allows the model to modulate image generation based on semantic information from the prompt.

最初，$Q/K/V$ 表示都从同一个输入嵌入投影而来，这一过程称为自注意力；而在文生图生成中，Transformer 常使用交叉注意力，使图像合成以文本提示词为条件。在这种设置下，查询（Q）由图像 token 投影得到，而键（K）与值（V）由文本嵌入投影得到。这让模型能够依据提示词中的语义信息来调制图像生成。

2 FLUX.1

2.1 Introduction 引言

FLUX.1 [7] is a rectified flow transformer trained in the latent space of an image encoder, introduced by **Black Forest Labs** in August 2024.

FLUX.1 [7] 是一个修正流 Transformer，在图像编码器的潜在空间中训练，由 **Black Forest Labs** 于 2024 年 8 月推出。

The FLUX.1 models (see 节 3) demonstrate State-of-the-art (SoTA) performance for text-to-image tasks, in both terms of output quality and image-text alignment, as demonstrated in 图 2 和 3 using the ELO-score metric, which

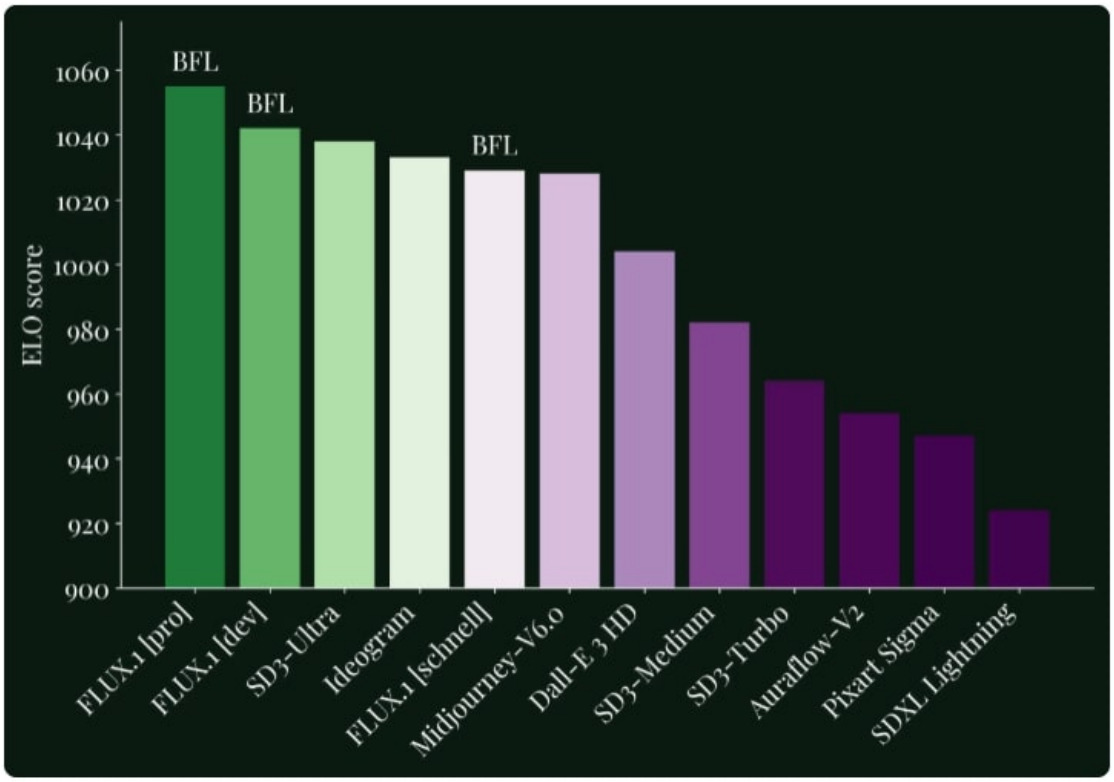


图 2: FLUX.1 defines a new state-of-the-art in image detail, prompt adherence, style diversity and scene complexity for text-to-image synthesis. Evaluation from [18] 在文生图合成中，FLUX.1 在图像细节、提示词遵循度、风格多样性与场景复杂度上重新定义了最先进水平。评测数据来自 [18]。

ranks image generation models based on human preferences in head-to-head comparisons. A qualitative illustration and comparison to SD is provided in Appendix A.

FLUX.1 系列模型（见 节 3）在文生图任务上展现出最先进（SoTA）的性能，无论是输出质量还是文本-图像对齐皆然，如 图 2 和 3 中基于 ELO 分数指标所示——该指标依据两两对比中的人类偏好对图像生成模型排名。与 SD 的定性图示与对比见 Appendix A。

While the model adheres to the Rectified Flow training paradigm (according to the developers statement), the exact details regarding the training setup —including the dataset, scheduling strategy, and hyperparameters— have not been publicly disclosed. However, the model's architecture and inference scheme can be reverse-engineered from the publicly available inference code.

尽管（据开发者声明）该模型遵循修正流训练范式，但训练配置的确切细节——包括数据集、调度策略与超参数——均未公开披露。不过，模型的架构与推理方案可以从公开的推理代码中逆向还原。

In this section, we outline the model's architecture to demystify its behavior at inference time. A top-view of the architecture is illustrated in 图 4, where text embeddings and latent image embeddings are iteratively processed via a series of attention blocks. In the following section we deep-dive into the different components of the model. We begin with a high-level overview of FLUX's sampling pipeline in 节 2.2, highlighting the key stages and the pre-trained components involved, followed by a deep-dive into the transformer's architecture in 节 2.3, where a detailed

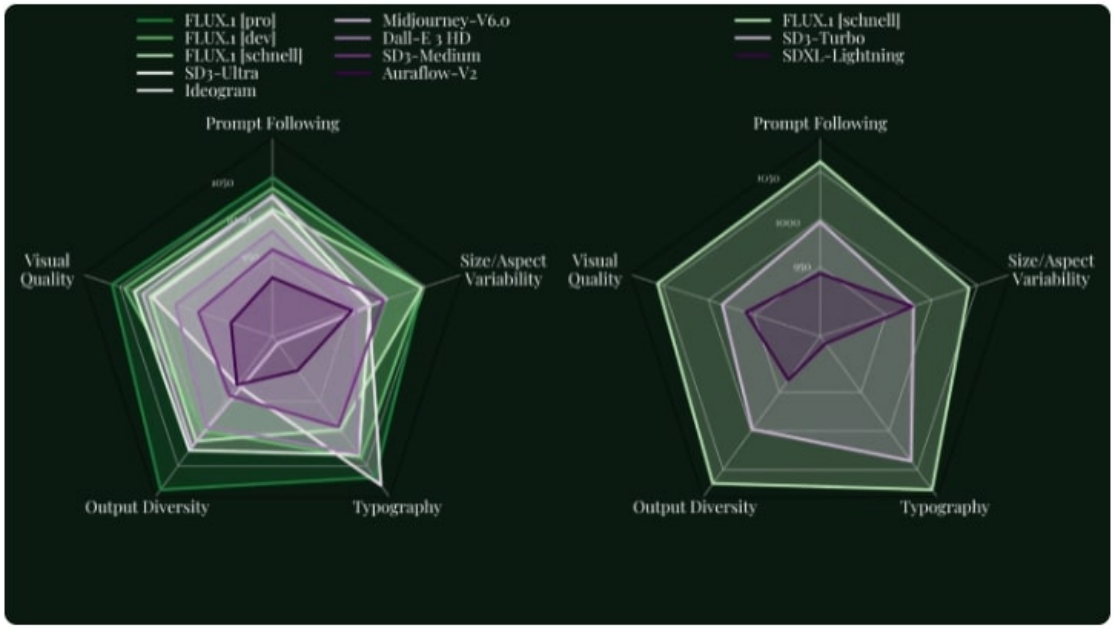


图 3: ELO scores for different aspects: Prompt Following, Size/Aspect Variability, Typography, Output Diversity, Visual Quality. Evaluation from [18] 各维度的 ELO 分数：提示词遵循、尺寸/宽高比可变性、排版文字、输出多样性、视觉质量。评测数据来自 [18]。

explanation is provided to the different stages and concepts used to construct it.

本节勾勒模型的架构，以揭示其在推理时的行为。架构的顶视图见 图 4，其中文本嵌入与潜在图像嵌入经由一系列注意力块被迭代处理。接下来我们深入模型的各个组成部分。首先在 节 2.2 中对 FLUX 的采样流水线做高层概览，突出其中的关键阶段与所涉及的预训练组件；随后在 节 2.3 中深入 Transformer 架构，对构建它所用的各个阶段与概念给出详细说明。

2.2 FLUX.1 Sampling Pipeline FLUX.1 采样流水线

In this section, we describe the sampling pipeline of FLUX.1. For simplicity, we refer to the text-to-image sampling process as being conditioned on a single prompt per sample.

本节描述 FLUX.1 的采样流水线。为简明起见，我们将文生图采样过程视为每个样本以单条提示词为条件。

Similar to LDM [6], FLUX operates in a latent space, where the final latent output is decoded to reconstruct the RGB image in pixel space. Following LDM's approach, the developers trained a convolutional autoencoder from scratch using an adversarial objective, but scaled up the latent representation from 4 channels (in LDM) to 16 channels.

与 LDM [6] 类似，FLUX 在潜在空间中运行，最终的潜在输出被解码以在像素空间重建 RGB 图像。沿用 LDM 的思路，开发者用对抗目标从零训练了一个卷积自编码器，但把潜在表示从（LDM 中的）4 通道扩大到 16 通道。

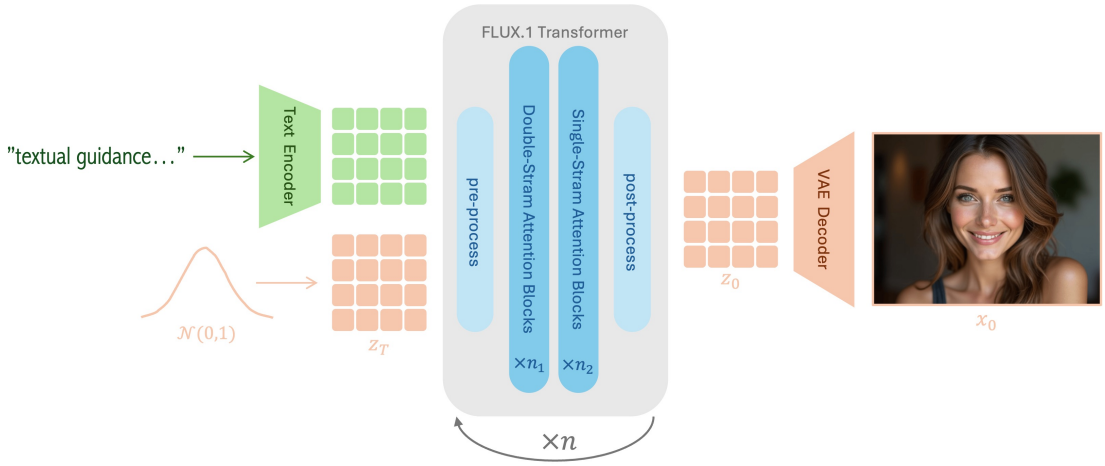


图 4: High-level overview of the FLUX.1 architecture. Text embeddings and latent image embeddings are iteratively processed through a series of attention blocks to generate a text-conditioned image. FLUX.1 架构的高层概览。文本嵌入与潜在图像嵌入经由一系列注意力块被迭代处理，从而生成文本条件下的图像。

The sampling pipeline consists of three phases: (1) *Initiation and Pre-processing*, (2) *Iterative Refinement*, and (3) *Post-processing*. An overview of these steps is illustrated in 图 5, where the notations follows the ones used in the official implementation of FLUX.1 pipeline in diffusers [19]. In the section below, we primarily focus on the *Initiation and Pre-processing* phase of the pipeline. A brief overview of the *Iterative Refinement* and *Post-processing* phases is provided at the end of this section. The *Iterative Refinement* phase, in which the FLUX transformer operates, is discussed in detail in 节 2.3.

采样流水线包含三个阶段：(1) 初始化与预处理，(2) 迭代精修，(3) 后处理。这些步骤的概览见 图 5，其中的记号沿用 diffusers [19] 中 FLUX.1 流水线官方实现所用的记号。下文主要聚焦流水线的初始化与预处理阶段。迭代精修与后处理阶段的简要概览放在本节末尾。FLUX Transformer 运作所在的迭代精修阶段，将在 节 2.3 中详细讨论。

2.2.1 Initiation and Pre-processing 初始化与预处理

In this phase, the model processes the user-provided inputs to prepare them for the main transformer's refinement stage. A key component of this stage is the text encoder, which processes the textual guidance supplied by the user. FLUX.1 utilizes two pre-trained text encoders:

在此阶段，模型处理用户提供的输入，为主 Transformer 的精修阶段做好准备。此阶段的一个关键组件是文本编码器，负责处理用户给出的文本引导。FLUX.1 使用两个预训练文本编码器：

CLIP Text Encoder. CLIP (Contrastive Language–Image Pretraining) [9] is a foundation model developed by OpenAI, designed to bridge vision and language understanding. CLIP's text-encoder transforms natural language prompts into class level or dense, high-dimensional embeddings that can be directly compared with image embeddings in a shared latent space. Trained on hundreds of millions of image–text pairs, the CLIP text encoder captures rich semantic information, enabling models to align textual descriptions with corresponding visual content effectively, making it widely used in text-to-image generation tasks.

CLIP 文本编码器。CLIP（对比语言-图像预训练，Contrastive Language–Image Pretraining）[9] 是 OpenAI 开发的一个基础模型，旨在打通视觉与语言的理解。CLIP 的文本编码器把自然语言提示词转换为类别级或稠密的高维嵌入，这些嵌入可在共享的潜在空间中与图像嵌入直接比较。CLIP 文本编码器在数亿图文对上训练，捕捉了丰富的语义信息，使模型能有效地将文本描述与相应的视觉内容对齐，因而在文生图任务中被广泛使用。

T5 Text Encoder. T5 (Text-To-Text Transfer Transformer) [20] is a versatile language model developed by Google that frames all NLP tasks (e.g., translation, summarization, and question answering) as text-to-text problems. Its text encoder converts input text into contextualized embeddings using a Transformer-based architecture trained over massive language corpora. Unlike CLIP's text encoder, which is trained jointly with an image encoder to produce embeddings aligned with visual features for contrastive learning, T5 is trained purely on textual data and optimized for language understanding and generation. This makes T5 well-suited for providing rich, token-level semantic representations even for long and complex textual prompmts.

T5 文本编码器。 T5（文本到文本迁移 Transformer，Text-To-Text Transfer Transformer）[20] 是 Google 开发的一个通用语言模型，它把所有 NLP 任务（如翻译、摘要、问答）都表述为文本到文本的问题。其文本编码器采用基于 Transformer 的架构，在海量语言语料上训练，把输入文本转换为带上下文的嵌入。CLIP 的文本编码器与图像编码器联合训练，以产生与视觉特征对齐、用于对比学习的嵌入；T5 则不同，它纯粹在文本数据上训练，为语言理解与生成而优化。这使得 T5 即便面对冗长复杂的文本提示词，也能很好地提供丰富的、token 级的语义表示。

Below, we outline all the required inputs and describe how the model processes them in preparation for the refinement stage.

下面，我们列出所有必需的输入，并描述模型如何处理它们，为精修阶段做准备。

The pipeline's required inputs are:

流水线所需的输入包括：

- **text:** textual prompt guides the image generation process to enforce specified features or qualities. 文本提示词，引导图像生成过程以体现指定的特征或品质。
- **guidance_scale:** controls the strength of conditioning. 控制条件化的强度。
- **num_inference_steps:** how many iterative sampling steps should be performed. 需要执行多少个迭代采样步。
- **resolution:** Specifies the spatial resolution (height and width) of the generated image. 指定生成图像的空间分辨率（高与宽）。

Once initiated with those inputs, they are being preprocessed as follows:

一旦以这些输入完成初始化，它们便按如下方式被预处理：

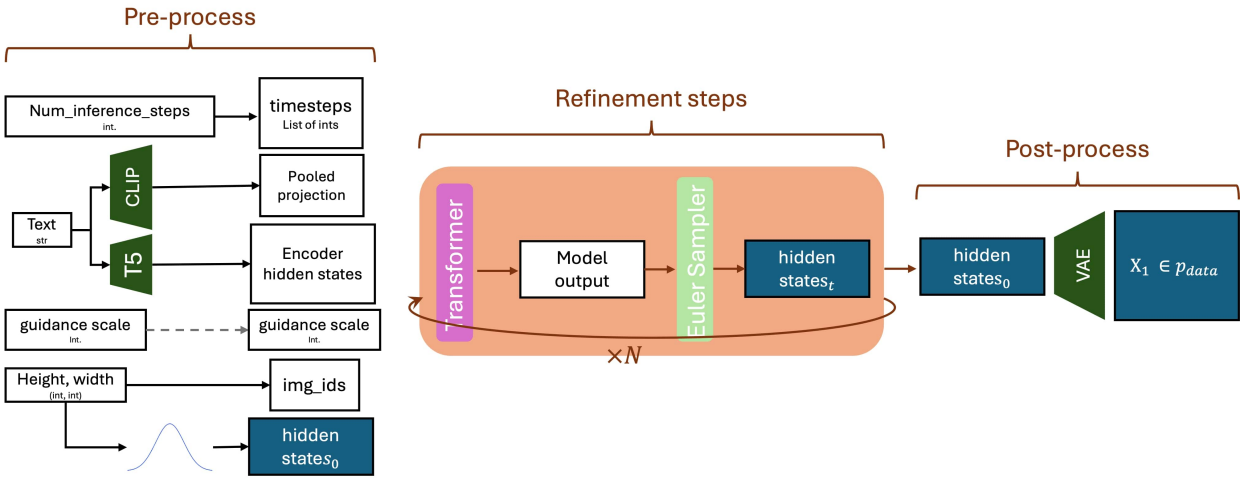


图 5: Flux.1 sampling pipeline. Just like in Diffusion Models, after pre-processing the noisy latent z_t (denoted $hidden\_state_t$) is being iteratively refined in the latent space. The final refined z_0 ($=hidden\_states_0$) is decoded into an RGB image using a pre-trained VAE decoder. Flux.1 采样流水线。与扩散模型一样，预处理之后，含噪潜在变量 z_t（记为 $hidden\_state_t$）在潜在空间中被迭代精修。最终精修得到的 z_0（$=hidden\_states_0$）经预训练的 VAE 解码器解码为 RGB 图像。

- **text:** The textual prompt is being encoded using two pre-trained text encoders: 文本提示词使用两个预训练文本编码器进行编码：
  - T5: provides dense (per token) embeddings, denoted *encoder hidden states* T5：提供稠密的（逐 token）嵌入，记为 *encoder hidden states*
  - CLIP: provides pooled embeddings (one embedding for the whole prompt, like CLS embedding), denoted *pooled projection* CLIP：提供池化嵌入（整条提示词一个嵌入，类似 CLS 嵌入），记为 *pooled projection*
- **num_inference_steps:** Specifies the total number of sampling steps used during inference. It determines a subset of timesteps from the full diffusion range $t \in [0:T]$, where typically $t = 1000$. Iterating over these selected timesteps defines the sampling trajectory. 指定推理时采样步的总数。它从完整的扩散区间 $t \in [0:T]$（通常 $t = 1000$）中确定一个时间步子集。在这些选定的时间步上迭代，即定义了采样轨迹。
- **resolution:** The desired resolution determines the spatial dimensions of the initial (latent) noise sample $z_0 \sim \mathcal{N}(0,1)$. it is also used to define the img_ids- a set of per-token indicators, pointing at the token's spatial location on a 2D grid. Given a target resolution (H,W) in pixel space, the corresponding latent dimensions (h, w) are computed as $(h = H//VAE_{scale})$ and $(w = W//VAE_{scale})$ where $VAE_{scale} = 8$. The image-token grid is further downsampled to dimensions $(h//2, w//2)$, and each token is assigned a unique identifier of the form $(t, \hat{h}, \hat{w})$, where $\hat{h} \in [0:h-1]$ and $\hat{w} \in [0, w-1]$, indicating the token's location on a $2D$ spatial grid. text_ids are initiated using the same structure of img_ids, but with $t = h = w = 0$ for all tokens. Formally, text_ids $= n \cdot (0,0,0)$ where n is the maximal number of tokens in T5 $(= 512)$. 期望的分辨率决定了初始（潜在）噪声样本 $z_0 \sim \mathcal{N}(0,1)$ 的空间维度。它还用于定义 img_ids——一组逐 token 的指示符，指向

token 在二维网格上的空间位置。给定像素空间中的目标分辨率 (H, W)，对应的潜在维度 (h, w) 计算为 $(h = H//VAE_{scale})$ 与 $(w = W//VAE_{scale})$，其中 $VAE_{scale} = 8$。图像 token 网格进一步下采样到 $(h//2, w//2)$ 的尺寸，每个 token 被赋予一个形如 $(t, \hat{h}, \hat{w})$ 的唯一标识符，其中 $\hat{h} \in [0 : h-1]$、$\hat{w} \in [0, w-1]$，表示该 token 在二维空间网格上的位置。text_ids 采用与 img_ids 相同的结构初始化，但所有 token 的 $t = h = w = 0$。形式上，text_ids $= n \cdot (0, 0, 0)$，其中 n 是 T5 中的最大 token 数 $(= 512)$。

2.2.2 Iterative Refinement and Post-processing 迭代精修与后处理

Let $\Phi =$ *[encoder_hidden_states, pooled_projection, guidance, img_ids, text_ids]* be a set the pre-processed pipeline's inputs. Same as in Diffusion Models, $[z_t, t, \Phi]$, are iteratively fed into the transformer during inference sampling, s.t.:

令 $\Phi =$ *[encoder_hidden_states, pooled_projection, guidance, img_ids, text_ids]* 为流水线经预处理后的输入集合。与扩散模型一样，推理采样时 $[z_t, t, \Phi]$ 被迭代地送入 Transformer，使得：

$$\forall\, t \in \text{timesteps}: \quad z_{t+\Delta t} = \text{Samp}(v_\theta(z_t, t, \Phi)) \tag{4}$$

Where v_θ is the trainable network that estimates the velocity vector (see 1.4) and *Samp(·)* refers to the Flow-Matching Euler Discrete sampler [15]. Pay attention to the notation that differs from the one used in Diffusion Models. Here *timesteps* ranges between 0 to 1, with z_1 the clear image and z_0 the pure Gaussian noise. In 节 2.3 we explore the architecture of v_θ.

其中 v_θ 是估计速度向量的可训练网络（见 1.4），*Samp(·)* 指流匹配欧拉离散（Flow-Matching Euler Discrete）采样器 [15]。请留意这里的记号与扩散模型所用的不同：此处 *timesteps* 取值在 0 到 1 之间，z_1 是清晰图像，z_0 是纯高斯噪声。我们将在 节 2.3 中探讨 v_θ 的架构。

After the iterative refinement, the final clean latent z_1 is decoded via pre-trained VAE model to get the final image x_1.

迭代精修之后，最终的干净潜在变量 z_1 经预训练的 VAE 模型解码，得到最终图像 x_1。

2.3 Transformer Transformer

The core component of FLUX.1's synthesis pipeline is the velocity predictor v_θ, which is optimized to estimate the velocity vector along the sampling trajectory (see 节 1.4). Similar to SD3 [8], FLUX.1 replaces the conventional *U-Net* architecture with a fully transformer-based design. A high-level overview of the transformer's operations at each sampling step is provided in Figure 6 using the notation defined in the official implementation by diffusers [19]. On every iteration, the inputs are preprocessed before they ate fed into a sequence of transformer blocks. There are two types of transformer blocks: (1) *Double-Stream* (see 节 2.3.2), and (2) *Single-Stream* (see 节 2.3.3). Both blocks employ a *Multi-Modal Attention*: joint self-attention mechanism to process concatenated text and image tokens in a unified attention operation, enabling bidirectional interactions that capture both self-information and cross-modal information between text and image modalities. Below we provide a step-by-step guide to the per-iteration pre-process (节 2.3.1), and the *Double-Stream* (节 2.3.2) and *Single-Stream* (节 2.3.3) attention blocks.

FLUX.1 合成流水线的核心组件是速度预测器 v_θ，它经优化以估计采样轨迹上的速度向量（见 节 1.4）。

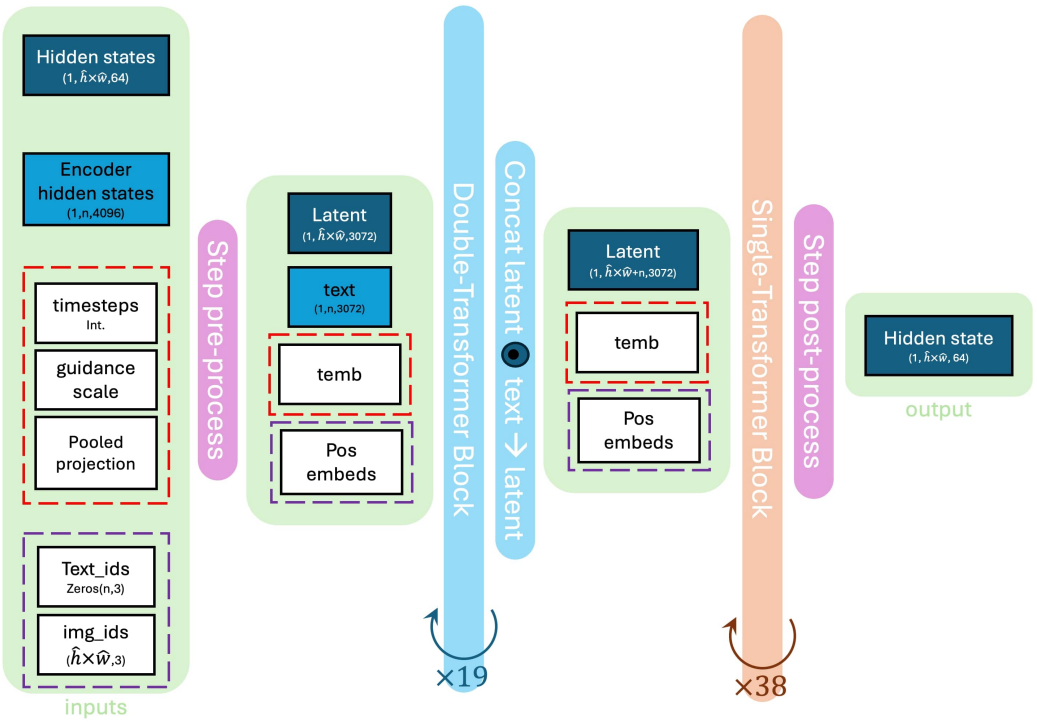


图 6: Flux Transformer: high-level overview Flux Transformer：高层概览

与 SD3 [8] 类似，FLUX.1 用完全基于 Transformer 的设计取代了传统的 *U-Net* 架构。Transformer 在每个采样步的操作概览见图 6，其中的记号采用 diffusers [19] 官方实现中定义的记号。每次迭代中，输入先经预处理，再送入一串 Transformer 块。Transformer 块有两种类型：(1) 双流（Double-Stream，见 节 2.3.2），(2) 单流（Single-Stream，见 节 2.3.3）。两种块都采用多模态注意力：一种联合自注意力机制，在统一的注意力操作中处理拼接后的文本 token 与图像 token，从而实现双向交互，既捕捉自身信息，也捕捉文本与图像两个模态间的跨模态信息。下面我们逐步讲解每次迭代的预处理（节 2.3.1），以及双流（节 2.3.2）和单流（节 2.3.3）注意力块。

2.3.1 Per-Iteration Pre-process 每次迭代的预处理

Along the sampling trajectory, the transformer is called multiple times, once at each sampling iteration. In every call, it is provided with an updated set of parameters:

沿采样轨迹，Transformer 被多次调用，每个采样迭代调用一次。每次调用都会向它提供一组更新后的参数：

- the guiding parameter *timestep* is iterated from the pre-calculated list of values between *1* to *0*. 引导参数 *timestep*，从预先计算好的、取值介于 *1* 到 *0* 之间的列表中逐个取用。
- z_t (refers as hidden_state in the official implementation by diffusers [19]) is the latent representation which is (iteratively) refined from Gaussian noise into a clear image. It is being updated along the sampling trajectory. z_t（在 diffusers [19] 官方实现中称为 hidden_state）是潜在表示，它（迭代地）从高斯噪声精修为清

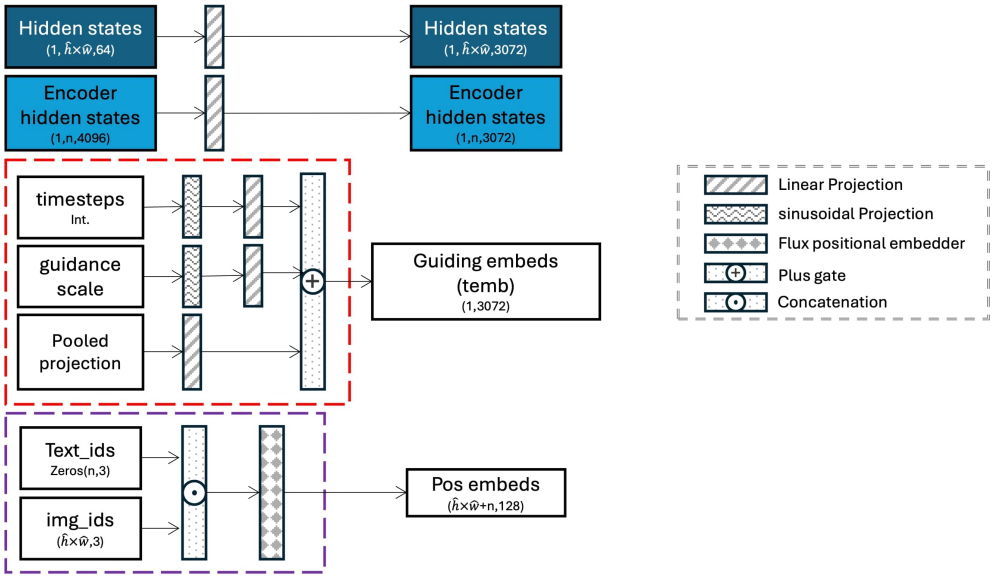


图 7: Flux Transformer: per-iteration inputs pre-process Flux Transformer：每次迭代的输入预处理

晰图像，并沿采样轨迹不断更新。

These parameters join the "constant" inputs (which remain constant across different iterations) pre-calcualted in the pipeline's pre-process stage (节 2.2.1). The transformer pre-processes it's inputs internally as described in 图 7. Formally, the following steps are taken:

这些参数与流水线预处理阶段（节 2.2.1）预先算好的"常量"输入（在不同迭代间保持不变）汇合。Transformer 在内部按 图 7 所述对其输入进行预处理。形式上，执行以下步骤：

- use of per-domain linear layers to bring the latent embeddings and dense-prompt-embeddings (T5) into a shared dimensionality of 3072 features per token. 使用各域独立的线性层，把潜在嵌入与稠密提示词嵌入（T5）投影到共享维度，即每 token 3072 个特征。

- construction of guiding embeds. Unlike Diffusion Models, that encode only the "temporal" information (i.e, timestep) in this phase, FLUX.1 uses both the timestep and the pooled prompt embeds (from CLIP) as guiding embeddings, yet keeps the traditional notation *temb*. It uses sinusoidal projection (As is [4]) to embed the integer values of the timestep and guidance, than applies dedicated linear projection layers to each component to bring them to the shared dimensionality of 3072 features. Finally, the 3 projected embeddings are summed to create the final *temb* (marked in a red block in 图 7) 构建引导嵌入。扩散模型在此阶段只编码"时间"信息（即 timestep），FLUX.1 则不同：它同时用 timestep 和池化提示词嵌入（来自 CLIP）作为引导嵌入，但仍沿用传统记号 *temb*。它用正弦投影（如 [4] 所做）来嵌入 timestep 与 guidance 的整数值，再对每个分量施加专门的线性投影层，把它们投影到 3072 特征的共享维度。最后，这 3 个投影后的嵌入相加，构成最终的 *temb*（在 图 7 中以红色块标出）。

- the img_ids and text_ids (see 节 2.2) are concatenated, than used to extract per-token positional embeds. To extract positional embeddings from 3D token indices (t,h,w), each axis is first converted to a continuous

embedding using axis-specific sinusoidal frequencies scaled by a constant θ. The cosine and sine of the resulting frequency-position products are computed and interleaved to form real-valued vectors. These are then concatenated across all axes to produce the final positional embedding as a pair of tensors (cosine and sine), ready to be used in rotary positional encoding (the process os composed in the purple block in 图 7). Note that this process is not influenced neither by *timestep* nor *hidden_states*, hence the pos_embeds are constant across different sampling steps. img_ids 与 text_ids（见 节 2.2）被拼接起来，用于提取逐 token 的位置嵌入。为从三维 token 索引 (t, h, w) 提取位置嵌入，先用各轴专属、由常数 θ 缩放的正弦频率，把每个轴转换为连续嵌入。再计算由此得到的频率-位置乘积的余弦与正弦，并交错排列构成实数向量。随后将所有轴上的结果拼接起来，得到最终的位置嵌入，形式为一对张量（余弦与正弦），可直接用于旋转位置编码（该过程在 图 7 中由紫色块构成）。注意这一过程既不受 *timestep* 影响，也不受 *hidden_states* 影响，因此 pos_embeds 在不同采样步之间保持不变。

The *pos_embeds* and *temb* parameters are used to support the attention mechanism along the step, while *hidden_states* and *encoder_hidden_states* are being processed and refined along the step.

pos_embeds 与 *temb* 参数用于在每一步中支撑注意力机制，而 *hidden_states* 与 *encoder_hidden_states* 则在每一步中被处理与精修。

2.3.2 Double-Stream Transformer Block 双流 Transformer 块

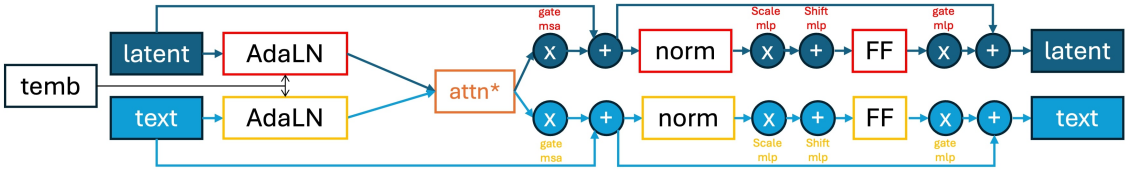
After pre-process, a series of *19* Double-Stream Transformer blocks are applied. Those blocks employ separate weights for image and text tokens. Multi-modality is achieved by applying the attention operation over the concatenation of the tokens (see 图 8b).

预处理之后，会施加一串共 *19* 个双流 Transformer 块。这些块对图像 token 与文本 token 采用各自独立的权重。多模态性通过在拼接后的 token 上执行注意力操作来实现（见 图 8b）。

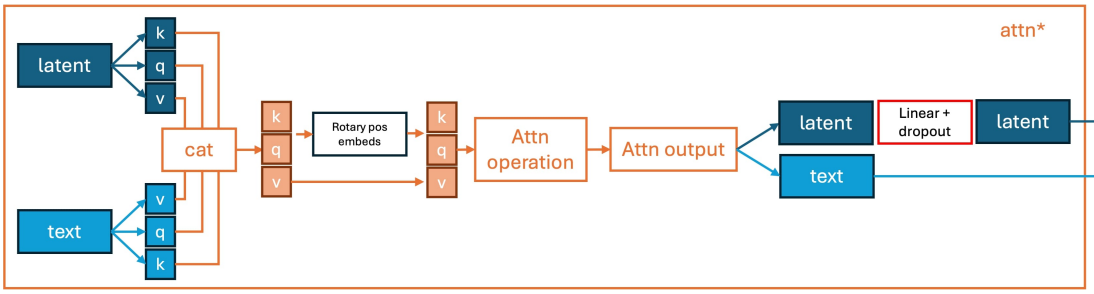
Each stream (latent and prompt) uses Adaptive Layer Normalization (AdaLN) [21] layers for normalization and modulation (see 图 9 and 图 8a). AdaLN is a conditioning mechanism used in Transformer-based models [22, 23] to modulate intermediate activations based on external input, such as text or image embeddings, based on the Adaptive Instance Normalization (AdaIN) mechanism proposed in [24]. Unlike standard Layer Normalization, which applies fixed scaling and shifting parameters, AdaLN dynamically generates these parameters as functions of a conditioning vector (see 图 9). This allows the model to adapt its behavior at each layer according to the input prompt or guidance signal.

每条流（潜在流与提示词流）都使用自适应层归一化（AdaLN）[21] 层来做归一化与调制（见 图 9 与图 8a）。AdaLN 是一种用于基于 Transformer 的模型 [22, 23] 的条件化机制，它依据外部输入（如文本或图像嵌入）来调制中间激活，其基础是 [24] 提出的自适应实例归一化（AdaIN）机制。标准层归一化施加固定的缩放与平移参数，AdaLN 则不同：它把这些参数动态生成为条件向量的函数（见 图 9）。这让模型能在每一层根据输入提示词或引导信号调整自身行为。

The normalized tensors are used to extract the K, Q, and V matrices for each domain, which are then concatenated for mixed attention. The concatenated K and Q matrices are rotated using precomputed Rotary Positional Embeddings. Rotary Positional Embeddings (RoPE) [25] are a method for injecting positional information into



(a) Double-Stream Attention Block: latent and prompt embeddings are processed separately, in a standard multi-modal attention scheme. 双流注意力块：潜在嵌入与提示词嵌入分别处理，采用标准的多模态注意力方案。



(b) The attention operation is applied over the concatenation of the tokens. 注意力操作施加于拼接后的 token 上。

图 8: Double-Stream Block. 双流块。

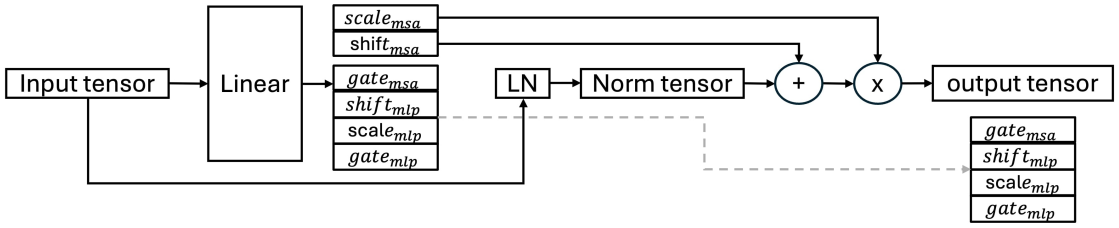


图 9: AdaLN layer, where MSA (Multi-head Self Attention) and MLP (Multi-Layer Processor) modulation parameters are computed based on the input tensor. In Single-Stream block (see 节 2.3.3), MLP modulation is not computed. AdaLN 层，其中 MSA（多头自注意力，Multi-head Self Attention）与 MLP（多层处理器，Multi-Layer Processor）的调制参数根据输入张量计算。在单流块（见 节 2.3.3）中不计算 MLP 调制。

Transformer models by rotating the query and key vectors in multi-head self-attention according to their token positions. Unlike traditional sinusoidal or learned positional embeddings that are **added** to input tokens, RoPE applies a **rotation** in the complex plane that preserves relative positional relationships across sequences. This approach allows the model to generalize better to unseen sequence lengths and supports extrapolation beyond training data. RoPE has become popular in recent vision-language models, where understanding spatial relationships between tokens, especially when re-purposing textual positions for image patches, is crucial.

归一化后的张量用于为每个域提取 K、Q、V 矩阵，随后将它们拼接以做混合注意力。拼接后的 K、Q 矩阵用预先计算好的旋转位置编码进行旋转。旋转位置编码（RoPE）[25] 是一种向 Transformer 模型注入位置信息的方法：它在多头自注意力中，依据 token 位置对查询向量与键向量施加旋转。传统的正弦或可学习位置嵌入是**加**到输入 token 上的，RoPE 则不同：它在复平面上施加一次**旋转**，从而在序列间保留相对位置关系。这一做法让模型能更好地泛化到未见过的序列长度，并支持超出训练数据范围的外推。RoPE 在近期的视觉-语言模型中已相当流行——在这些模型中，理解 token 之间的空间关系至关重要，尤其是把文本位置改用于图像图块时。

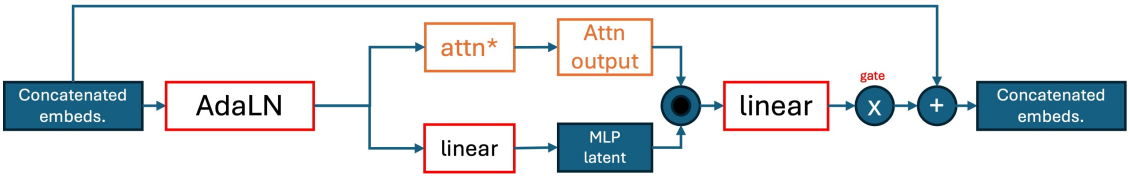
The rotated K and Q matrices are passed through the mixed attention operation. The outputs are separated back into their respective streams and alpha-blended with the residual input using the $gate_{\text{msa}}$ parameter, extracted by the AdaLN layer. The results are further normalized via a LayerNorm (LN) layer and modulated using the $scale_{\text{mlp}}$ and $shift_{\text{mlp}}$ parameters, also extracted by the AdaLN layer, before being passed to the feedforward layer. Finally, the output is alpha-blended with the unnormalized input using the $gate_{\text{mlp}}$ parameter from the AdaLN layer. The entire process is illustrated in 图 8a.

旋转后的 K、Q 矩阵经过混合注意力操作。其输出被拆分回各自的流，并用 AdaLN 层提取的 $gate_{\text{msa}}$ 参数与残差输入做 alpha 混合。所得结果再经一个 LayerNorm（LN）层归一化，并用同样由 AdaLN 层提取的 $scale_{\text{mlp}}$ 与 $shift_{\text{mlp}}$ 参数调制，然后送入前馈层。最后，输出用 AdaLN 层的 $gate_{\text{mlp}}$ 参数与未归一化的输入做 alpha 混合。整个过程如 图 8a 所示。

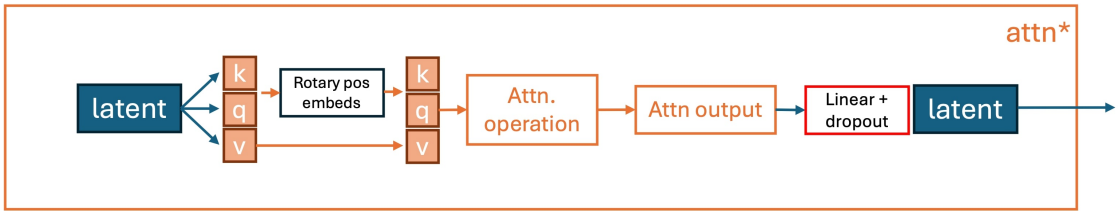
2.3.3 Single-Stream Transformer Block 单流 Transformer 块

After the series of Double-Stream blocks is applied, the processed latent and prompt embeddings are concatenated and fed through a series of Single-Stream blocks. While the Double-Stream blocks apply different weights for prompt and latent embeddings, the Single-Stream blocks use a single set of weights to process a concatenated tensor or latent and text embeddings (see 图 10). In addition, the Single-Stream blocks replace the standard sequential attention block (where an MLP, like Feed-Forward is applied after the attention step), with a parallel mechanism where the attention block and an MLP are computed simultaneously from the same input (see 图 10a). This should not be confused with the multi-modal attention mechanism, which is used in both Double-Stream and Single-Stream blocks, where self-attention and cross-attention are computed jointly in parallel.

施加完这一串双流块之后，处理过的潜在嵌入与提示词嵌入被拼接起来，送入一串单流块。双流块对提示词嵌入与潜在嵌入使用不同的权重，单流块则用同一组权重来处理由潜在嵌入与文本嵌入拼接而成的张量（见 图 10）。此外，标准的顺序式注意力块是在注意力步之后再施加一个 MLP（如前馈层），单流块则用并行机制取而代之：注意力块与 MLP 从同一输入同时计算（见 图 10a）。这不应与多模态注意力机制混淆——后者在双流块与单流块中都会用到，指自注意力与交叉注意力被联合地并行计算。



(a) Single-Stream Attention Block: latent and prompt embeddings are processed together, in a simultaneous attention scheme. 单流注意力块：潜在嵌入与提示词嵌入一起处理，采用同时进行的注意力方案。



(b) The K, Q and V metrices are computed directly from the concatenated representation. K、Q、V 矩阵直接从拼接后的表示计算得到。

图 10: Single-Stream Block. 单流块。

A comparison between the Double-Stream and Single-Stream block is summarized in 表 1:

双流块与单流块的对比总结于 表 1：

属性 Property	**双流块 Double-Stream**	**单流块 Single-Stream**
权重共享	在注意力层与前馈层中，对文本 token 与潜在 token 使用各自独立的权重。	在注意力层与前馈层中，对文本 token 与潜在 token 使用共享权重。
计算方式	注意力层与前馈（MLP）层顺序施加，注意力的输出即前馈层的输入。	注意力层与前馈层并行计算，二者使用同一输入。

表 1: Comparison between Double-Stream and Single-Stream blocks in FLUX.1 FLUX.1 中双流块与单流块的对比

Specifically, the Double-Stream and Single-Stream blocks differ in two main attributes: weight sharing between text and latent tokens (shared vs. not shared), and computation style (sequential vs. parallel). While the exact motivation behind the FLUX.1 developers choice of this specific architecture is not documented, the following is a brief comparison of these attributes, highlighting the possible pros and cons of each, potentially shedding light on the reasoning behind including both block types in FLUX.1's architecture.

具体而言，双流块与单流块在两个主要属性上有所不同：文本 token 与潜在 token 之间的权重共享（共享 vs. 不共享），以及计算方式（顺序 vs. 并行）。FLUX.1 开发者选择这一特定架构的确切动机并无记载，下面对这些属性做简要对比，突出各自可能的优缺点，或许能为在 FLUX.1 架构中同时纳入两种块类型的理由提供一些线索。

Weight Sharing (Shared vs. Not Shared) **权重共享（共享 vs. 不共享）**

Weight sharing refers to whether the same attention and feedforward parameters are used for both text and latent tokens. In the Single-Stream block, shared weights enable more efficient parameter usage and promote tighter integration between modalities, which may help the model generalize better across domains. However, this comes at the cost of reduced flexibility, as both token types must be processed identically despite potentially having very different characteristics. In contrast, the Double-Stream block avoids weight sharing, allowing the model to specialize its representations for text and latent tokens independently. This specialization can improve performance when domain-specific distinctions are critical, though it increases model size and computational overhead. The inclusion of both strategies in FLUX.1 suggests a deliberate balance between efficiency and specialization.

权重共享指的是文本 token 与潜在 token 是否使用相同的注意力与前馈参数。在单流块中，共享权重让参数使用更高效，并促进模态间更紧密的融合，这可能有助于模型跨域泛化。但代价是灵活性下降：尽管两类 token 可能具有截然不同的特性，却必须被同样地处理。相比之下，双流块回避了权重共享，让模型能分别为文本 token 与潜在 token 特化各自的表示。当域间的区别至关重要时，这种特化能提升性能, 但也会增大模型规模与计算开销。FLUX.1 同时纳入两种策略, 暗示了在效率与特化之间刻意求取的平衡。

Computation Style (Sequential vs. Parallel) **计算方式（顺序 vs. 并行）**

The key distinction in computation style lies in how the attention and feedforward (MLP) layers are applied within a block. In the Double-Stream block, the computation is sequential: the input is first normalized, passed through an attention layer, then normalized again before being fed into the feedforward layer. This means the output of the attention block defines the input to the MLP feedforward, enabling tightly coupled, stage-wise processing that allows each layer to build upon the previous one. In contrast, the Single-Stream block follows a parallel design. The input is normalized once, and the resulting representation is simultaneously passed through both the attention and MLP layers independently. Their outputs are then combined downstream. This parallelism increases efficiency and allows for broader representation capacity per block, but it may limit the depth of inter-layer interaction present in sequential designs. FLUX.1's use of both may reflect a trade-off between expressive sequential processing and the speed or simplicity of parallel computation.

计算方式上的关键区别，在于一个块内注意力层与前馈（MLP）层如何施加。在双流块中，计算是顺序的：输入先被归一化，经过一个注意力层，再次归一化后才送入前馈层。这意味着注意力块的输出决定了 MLP 前馈层的输入，形成紧耦合的、分阶段的处理，使每一层都能在前一层的基础上继续构建。相比之下，单流块采用并行设计：输入只归一化一次，所得表示同时独立地经过注意力层与 MLP 层，二者的输出再在下游合并。这种并行提高了效率，并让每个块具备更大的表示容量，但也可能限制顺序设计所具有的层间交互深度。FLUX.1 同时使用两者,或许反映了在富有表现力的顺序处理与并行计算的速度或简洁之间的一种取舍。

In summary, Single-Stream blocks emphasize efficiency and simplicity through parallel computation and shared weights, while Double-Stream blocks favor specialization and expressiveness with sequential flow and separate weights. While the Double-Stream blocks follow the *mm-DiT* design previously used in *SD3* [8], the addition of Single-Stream blocks may reflect the FLUX authors' intention to expand the model's capacity in a relatively lightweight and efficient manner.

总之，单流块通过并行计算与共享权重强调效率与简洁，双流块则以顺序流与独立权重偏向特化与表现力。双流块沿用了先前 *SD3* [8] 所用的 *mm-DiT* 设计，而单流块的加入，可能反映了 FLUX 作者意在以相对轻

量、高效的方式扩展模型容量。

3 Models and Tools 模型与工具

3.1 text-to-image 文生图

The FLUX.1 suite of text-to-image models comes in three variants, all share the same architecture of 12B parameters, striking a balance between accessibility and model capabilities [18].

FLUX.1 文生图模型套件提供三个变体，它们共享同一套 12B 参数的架构，在易获取性与模型能力之间取得平衡 [18]。

- **FLUX.1[pro]** This the highest-performing variant of the FLUX.1 family, offering superior prompt alignment, visual detail, and output diversity. It is designed for commercial use and is accessible only through licensed API endpoints such as Replicate or Fal.ai. The model weights are not publicly released, making it a hosted-only solution. This version is ideal for production environments where top-tier image generation quality is critical and commercial licensing is feasible. 这是 FLUX.1 家族中性能最高的变体，提供更优的提示词对齐、视觉细节与输出多样性。它面向商业用途设计，仅可通过 Replicate、Fal.ai 等获授权的 API 端点访问。模型权重未公开发布，因此只能作为托管式方案使用。该版本非常适合对顶级图像生成质量要求严苛、且商业授权可行的生产环境。

- **FLUX.1[dev]** provides an open-weight alternative to Pro, guidance-distilled [26] from it. It is licensed strictly for non-commercial use, making it accessible to researchers, developers, and hobbyists for experimentation and academic work. Unlike Pro, Dev can be downloaded and run locally, offering full control and flexibility for non-commercial applications. The FLUX.1 [dev] model is available via huggingface can be directly tried out on Replicate or fal.ai. It should be noted that the Dev version might require more sampling steps than the pro version to achieve high-level performance. 提供了 Pro 的开放权重替代版本，由 Pro 经引导蒸馏 [26] 而来。它的授权严格限定为非商业用途，供研究者、开发者与爱好者用于实验与学术工作。与 Pro 不同，Dev 可下载并在本地运行，为非商业应用提供了完全的控制与灵活性。FLUX.1[dev] 模型可经 huggingface 获取，也可直接在 Replicate 或 fal.ai 上试用。需要注意的是，Dev 版本可能需要比 Pro 版本更多的采样步才能达到高水平性能。

- **FLUX.1[schnell]** is a speed-optimized, distilled variant of FLUX.1 Pro, designed to significantly reduce inference time while retaining reasonable image quality. It shares the same 12B architecture but is trained via timesteps-distillation [27] from the Pro model, meaning it learns to mimic Pro's outputs using fewer sampling steps, up to a single step, by optimizing for speed and efficiency. This results in faster generations at the cost of some prompt fidelity and fine detail. Released under the permissive Apache 2.0 license, Schnell allows unrestricted commercial use, making it ideal for real-time, low-latency applications and lightweight deployments where speed and open licensing outweigh the need for peak visual fidelity. 是 FLUX.1 Pro 的一个速度优化、蒸馏而来的变体，旨在大幅缩短推理时间，同时保留合理的图像质量。它共享同一套 12B 架构，但由 Pro 模型经时间步蒸馏 [27] 训练而来——也就是说，它通过面向速度与效率的优化，学习用更少的采样步（少至单步）模仿 Pro 的输出。这带来了更快的生成，代价是牺牲一部分提示词保真度与精细细

节。Schnell 以宽松的 Apache 2.0 许可发布，允许不受限的商业使用，因而非常适合实时、低延迟的应用，以及那些速度与开放许可胜过极致视觉保真需求的轻量部署。

- **FLUX1.1[pro]** is an enhanced version of the FLUX.1[pro] model, capable of faster image generation while also improving image quality, prompt adherence, and diversity. The model introduces new modes—Ultra, enabling 4x higher resolution without compromising speed, and Raw, producing hyper-realistic, candid-style images. Additionally, a prompt upsampling feature leverages large language models (LLMs) to expand and enrich user prompts, enhancing creative outputs. While not covered in this report, FLUX 1.1 Pro is available via API on platforms like Replicate and Fal.ai, with commercial licensing required. 是 FLUX.1[pro] 模型的增强版本，能更快地生成图像，同时提升图像质量、提示词遵循度与多样性。该模型引入了新模式——Ultra，可在不牺牲速度的前提下实现 4 倍分辨率；Raw，可生成超写实、抓拍风格的图像。此外，提示词上采样功能借助大语言模型（LLM）来扩写并丰富用户提示词，从而增强创意产出。FLUX 1.1 Pro 不在本报告的讨论范围内，它可通过 Replicate、Fal.ai 等平台的 API 使用，并需商业授权。

3.2 tools 工具

Originally developed for text-to-image generation, FLUX.1 has since been extended by Black Forest Labs with a suite of tools supporting a range of use cases beyond its original purpose. This report does not explore each tool in depth as it does for the FLUX.1 model itself. Below is a list of publicly available tools, current as of June 5th, 2025:

FLUX.1 最初为文生图生成而开发，此后 Black Forest Labs 为它扩展了一套工具，支持超出其最初用途的一系列使用场景。本报告不像对 FLUX.1 模型本身那样深入探讨每个工具。下面列出截至 2025 年 6 月 5 日公开可用的工具：

- **FLUX-Fill** is an inpainting and outpainting models, enabling editing and expansion of real and generated images given a text description and a binary mask. 是图像内补与外扩模型，可在给定文本描述与二值掩码的情况下，对真实图像与生成图像进行编辑与扩展。
- **FLUX-Canny** enables structural guidance based on canny edges extracted from an input image and a text prompt. 基于从输入图像提取的 Canny 边缘与文本提示词，实现结构性引导。
- **FLUX-Depth** enables structural guidance based on a depth map extracted from an input image and a text prompt. 基于从输入图像提取的深度图与文本提示词，实现结构性引导。
- **FLUX-Redux** is an adaptor that aligns dense (per-token) image embeddings extracted by SigLIP image encoder [28] with the T5 text-embedding's space for image-conditioned generation with a pre-traine FLUX.1 base model. While limited for image variation in the Dev version, It integrates into more complex workflows unlocking image restyling via prompt when using the FLUX1.1 [pro] Ultra model, allowing for combining input images and text prompts. 是一个适配器，它把 SigLIP 图像编码器 [28] 提取的稠密（逐 token）图像嵌入对齐到 T5 文本嵌入空间，从而配合预训练的 FLUX.1 base 模型进行图像条件下的生成。在 Dev 版本中它仅限于图像变体，但接入更复杂的工作流后，配合 FLUX1.1[pro] Ultra 模型可解锁经提示词的图像风格重塑，允许将输入图像与文本提示词结合。

- **FLUX-Kontext** extends the text-to-image capabilities of the base FLUX model into powerful in-context generation and editing workflows by enabling multimodal conditioning—allowing users to guide generation using both text and reference images. While *FLUX-Redux* had already introduced basic multimodal conditioning for image variation and prompt-driven restyling, *FLUX-Kontext* unlocks advanced functionality such as localized edits, style transfer, and character or scene consistency across images. With models like Kontext [pro] and [max], the focus shifts toward fast, iterative generation and high-precision editing. As of June 5th, 2025, the *FLUX-Kontext* models are not yet publicly released, and therefore could not be directly explored or evaluated in this report. 通过启用多模态条件化，把 base FLUX 模型的文生图能力扩展为强大的上下文内生成与编辑工作流——让用户能同时用文本与参考图像来引导生成。*FLUX-Redux* 已引入用于图像变体与提示词驱动的风格重塑的基础多模态条件化，*FLUX-Kontext* 则进一步解锁了局部编辑、风格迁移，以及跨图像的角色或场景一致性等高级功能。借助 Kontext[pro]、Kontext[max] 等模型，重点转向了快速的迭代式生成与高精度编辑。截至 2025 年 6 月 5 日，*FLUX-Kontext* 模型尚未公开发布，因此本报告无法对其直接探讨或评测。

参考文献

[1] D. P. Kingma, M. Welling *et al.*, "Auto-encoding variational bayes," 2013.

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[3] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference," *Journal of Machine Learning Research*, vol. 22, no. 57, pp. 1–64, 2021.

[4] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[5] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in *International conference on machine learning*. Pmlr, 2021, pp. 8821–8831.

[6] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[7] B. F. Labs, "Flux," https://github.com/black-forest-labs/flux, 2024.

[8] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel *et al.*, "Scaling rectified flow transformers for high-resolution image synthesis," in *Forty-first international conference on machine learning*, 2024.

[9] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PmLR, 2021, pp. 8748–8763.

[10] C. Schuhmann, R. Beaumont, R. Vencu, C. W. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, P. Schramowski, S. R. Kundurthy, K. Crowson, L. Schmidt, R. Kaczmarczyk, and J. Jitsev, "LAION-5b: An open large-scale dataset for training next generation image-text models," in *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. [Online]. Available: https://openreview.net/forum?id=M3Y74vmsMcY

[11] G. Ilharco, M. Wortsman, R. Wightman, C. Gordon, N. Carlini, R. Taori, A. Dave, V. Shankar, H. Namkoong, J. Miller, H. Hajishirzi, A. Farhadi, and L. Schmidt, "Openclip," Jul. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5143773

[12] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach, "Sdxl: Improving latent diffusion models for high-resolution image synthesis," *arXiv preprint arXiv:2307.01952*, 2023.

[13] A. Sauer, D. Lorenz, A. Blattmann, and R. Rombach, "Adversarial diffusion distillation," in *European Conference on Computer Vision*. Springer, 2024, pp. 87–103.

[14] X. Liu, C. Gong, and Q. Liu, "Flow straight and fast: Learning to generate and transfer data with rectified flow," *arXiv preprint arXiv:2209.03003*, 2022.

[15] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," *arXiv preprint arXiv:2210.02747*, 2022.

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

[17] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[18] "Black-forest-labs official flux.1 announcement," https://bfl.ai/announcements/24-08-01-bfl.

[19] P. von Platen, S. Patil, A. Lozhkov, P. Cuenca, N. Lambert, K. Rasul, M. Davaadorj, D. Nair, S. Paul, W. Berman, Y. Xu, S. Liu, and T. Wolf, "Diffusers: State-of-the-art diffusion models," https://github.com/huggingface/diffusers, 2022.

[20] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.

[21] F. E. Keddous, A. Llanza, N. Shvai, and A. Nakib, "Vision transformers inference acceleration based on adaptive layer normalization," *Neurocomputing*, vol. 610, p. 128524, 2024.

[22] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen, "Glide: Towards photorealistic image generation and editing with text-guided diffusion models," *arXiv preprint arXiv:2112.10741*, 2021.

[23] A. Sauer, T. Karras, S. Laine, A. Geiger, and T. Aila, "Stylegan-t: Unlocking the power of gans for fast large-scale text-to-image synthesis," in *International conference on machine learning*. PMLR, 2023, pp. 30 105–30 118.

[24] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1501–1510.

[25] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, p. 127063, 2024.

[26] C. Meng, R. Rombach, R. Gao, D. Kingma, S. Ermon, J. Ho, and T. Salimans, "On distillation of guided diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 14 297–14 306.

[27] A. Sauer, F. Boesel, T. Dockhorn, A. Blattmann, P. Esser, and R. Rombach, "Fast high-resolution image synthesis with latent adversarial diffusion distillation," in *SIGGRAPH Asia 2024 Conference Papers*, 2024, pp. 1–11.

[28] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, "Sigmoid loss for language image pre-training," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 11 975–11 986.

Appendices

A Qualitative Evaluation 定性评测

In this section, we provide a qualitative comparison of FLUX.1's performance against Stable Diffusion (SD) 2.1, which remains widely used in various data generation projects across R&D and Engineering. We use FLUX.1*[dev]* with 50 sampling steps and a guidance scale of 2.5. For SD 2.1, we use 50 sampling steps and a guidance scale of 7.5. All comparisons are conducted at a resolution of 512×512, which is the recommended resolution for SD 2.1. We use the publicly available version of the two models without any additional finetuning.

本节给出 FLUX.1 与 Stable Diffusion（SD）2.1 性能的定性对比；后者在研发与工程的各类数据生成项目中至今仍被广泛使用。我们对 FLUX.1*[dev]* 采用 50 个采样步、引导尺度 2.5；对 SD 2.1 采用 50 个采样步、引导尺度 7.5。所有对比均在 512×512 分辨率下进行，这是 SD 2.1 的推荐分辨率。我们使用两个模型的公开版本，未做任何额外微调。

We compare the models' performance on text-to-image generation of automotive scenarios across the following aspects:

我们从以下几个维度，比较两个模型在汽车场景文生图生成上的表现：

- Overall image quality (图 1) 整体图像质量（图 1）
- text-image alignment (图 2) 文本-图像对齐（图 2）
- Alignment to sparse prompts: text descriptions that are minimal or loosely defined. (图 3) 对稀疏提示词的对齐：即极简或宽泛定义的文本描述。（图 3）
- Synthesis of rare concepts (图 4) 罕见概念的合成（图 4）
- Adherence to complex prompts containing multiple descriptive elements. (图 5) 对包含多个描述性元素的复杂提示词的遵循。（图 5）

In all examples, even if not specifically noted, the models were prompted to generate images as if captured from a vehicle's front dashcam. While SD 2.1 struggled to maintain the specifically requested viewing angle, FLUX.1 consistently remained faithful to it across different tasks.

在所有示例中，即便未特别说明，模型都被要求生成仿佛由车辆前置行车记录仪拍摄的图像。SD 2.1 难以维持所指定的视角，FLUX.1 则在不同任务中始终忠实于该视角。

We also demonstrate FLUX.1 capability to generate images in various resolutions and aspect ratios (图 6).

我们还展示了 FLUX.1 在多种分辨率与宽高比下生成图像的能力（图 6）。

All in all, FLUX.1 brings state-of-the-art genrative capabilities in all aformentioned aspects, compared to previous models, and particularly compared to SD2.1. These gaps become more pronaunced when relates to complex generative tasks, such as the automotive case, where FLUX's rich voucabulry and heavy pre-training enable hin to deal with generative tasks that are way out of distribution for previos models. It also handels high-resolution images

图 1: Overall image quality comparison. FLUX.1 generates high quality automotive scenes with different attributes (here: weather and lighting) without additional optimization. 整体图像质量对比。FLUX.1 无需额外优化即可生成具有不同属性（此处为天气与光照）的高质量汽车场景。

图 2: FLUX.1 is sensitive to subtle differences in textual guidance- for example, distinguishing between light and heavy snow. It also handles complex attributes, such as in the prompt: "A sunny day after a snowy period". FLUX.1 对文本引导中的细微差异很敏感——例如能区分小雪与大雪。它还能处理复杂属性，如提示词"雪后的一个晴天"。

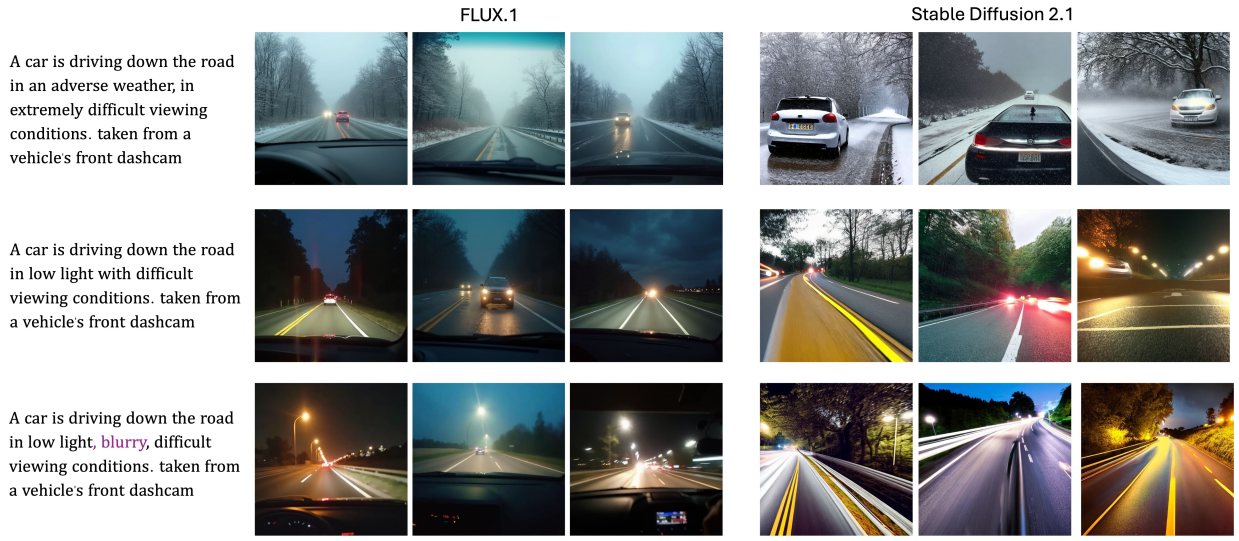


图 3: FLUX.1 is capable of interpreting textual guidance that is minimal or loosely defined, such as 'adverse weather" "difficult viewing conditions" while remaining sensitive to direct requests like "blurry". FLUX.1 能够理解极简或宽泛定义的文本引导，如“恶劣天气”“不良视野条件”，同时对“模糊”这类直接要求仍保持敏感。

图 4: FLUX.1's rich vocabulary and extensive pre-training enable it to handle rare concepts such as "boat-trailer" and "upside-down crashed car". FLUX.1 丰富的词汇与充分的预训练，使它能够处理“拖船挂车”“底朝天的撞毁汽车”等罕见概念。

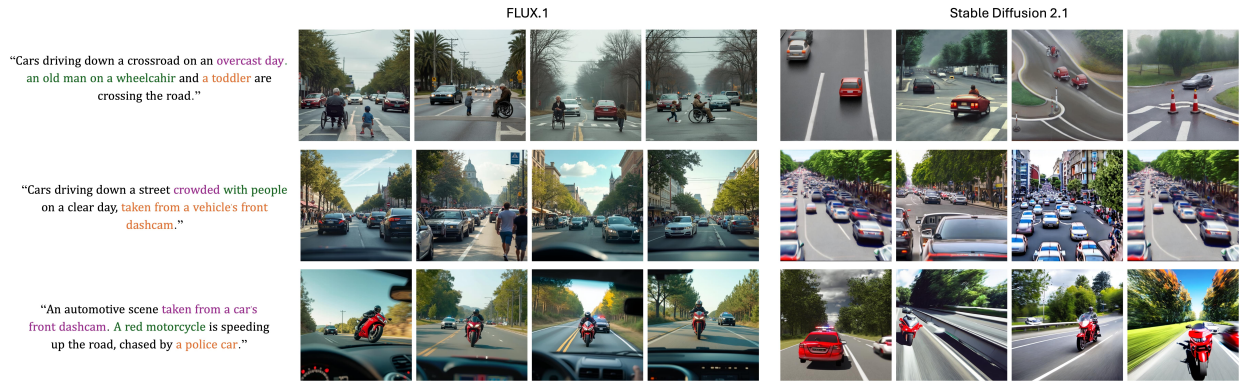


图 5: The dense prompt embeddings produced by the T5 text encoder enable FLUX.1 to adhere to complex prompts containing multiple requests. T5 文本编码器产生的稠密提示词嵌入，使 FLUX.1 能够遵循包含多个要求的复杂提示词。

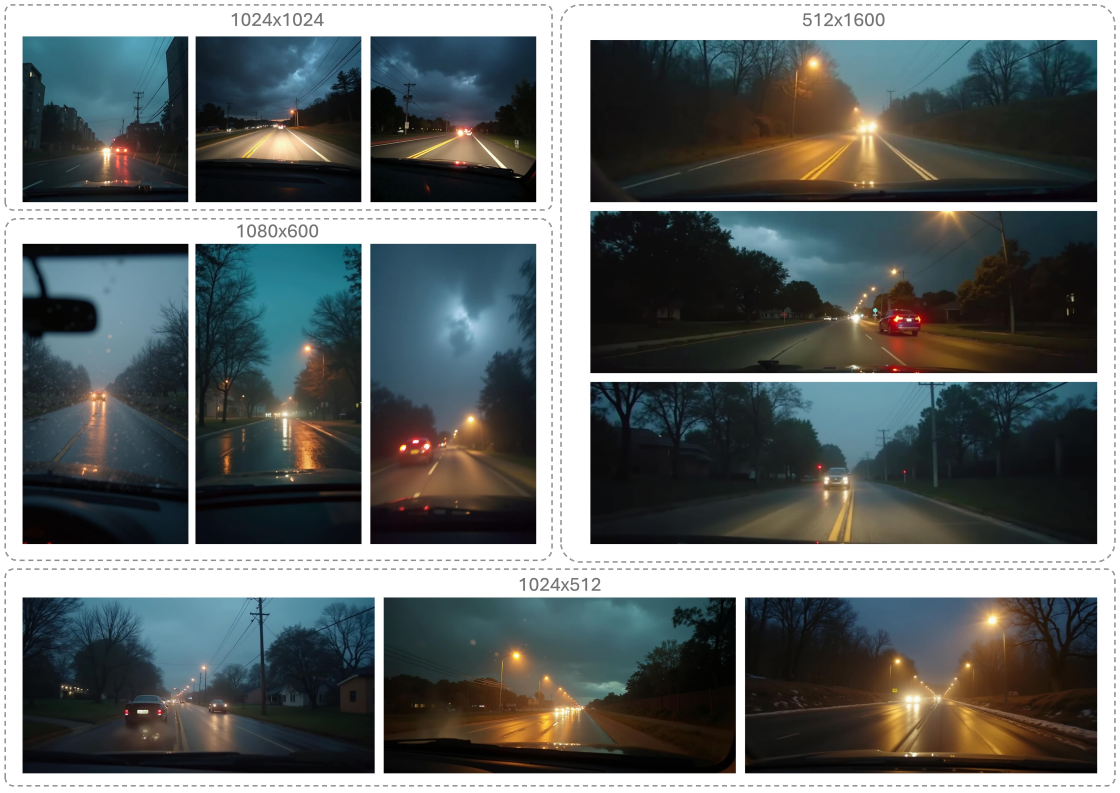


图 6: While existing U-Net–based diffusion models are typically limited to a specific or fixed set of resolutions, FLUX.1 can operate across a wide range of resolutions and aspect ratios. 现有基于 U-Net 的扩散模型通常受限于某个特定或固定的分辨率集合，FLUX.1 则能在广泛的分辨率与宽高比范围内工作。

with various aspect ratios. These capabilities may become handy in automotive-related augmentation tasks like Adverse viewing conditions and rare-concpts generation/inpaining, as well as in data restoration and enhancement.

总而言之，在上述所有维度上，FLUX.1 相较以往模型都带来了最先进的生成能力，相较 SD2.1 尤为明显。当涉及复杂生成任务（如汽车场景）时，这些差距更加显著：FLUX 丰富的词汇与充分的预训练，使它能够应对那些对以往模型而言严重超出分布的生成任务。它还能处理各种宽高比的高分辨率图像。这些能力在与汽车相关的数据增广任务中或许颇为实用，例如恶劣视野条件与罕见概念的生成/修复，以及数据修复与增强。