Deblurring Text Images with Generative Models

Bernardo Perrone Ribeiro

1 Introduction

Image deblurring is a core task in computer vision and image processing, focused on restoring clarity to blurred images. Common types of blur include motion blur, caused by camera or object movement, and defocus blur, resulting from improper focus. A notable subfield is text image deblurring, crucial for applications like document analysis, OCR, license plate recognition, and image-based search.

This project addresses single-image text deblurring: recovering a sharp text image from a blurred input. We explored two generative approaches: Denoising Diffusion Implicit Models [1] and Conditional Flow Matching [2], both state-of-the-art in image restoration. The models were trained on a custom dataset of blurred text images created by applying motion, Gaussian, and defocus blur to clean text samples. The dataset includes diverse fonts, styles, and sizes to promote generalization. The code is available at https://github.com/b-rbmp/DeblurCFM.

2 Related Works

The field of image deblurring has advanced significantly, with early work in text-specific deblurring relying on tailored priors. A key example is Pan et al.'s method [3], which introduced an effective L_0-regularized prior based on the unique intensity and gradient characteristics of text images to guide kernel estimation and restoration.

The advent of deep learning marked a turning point for general image deblurring. Nah et al. [4] pioneered this shift with a multi-scale convolutional neural network, demonstrating the effectiveness of end-to-end learning on realistic blurry images without explicit kernel estimation.

Generative models soon emerged as powerful tools for perceptual restoration. DeblurGAN [5] was among the first to use conditional GANs for deblurring, achieving state-of-the-art results. Its successor, DeblurGAN-v2 [6], further improved performance by integrating a Feature Pyramid Network and a relativistic conditional GAN, offering better speed and quality.

More recently, diffusion models have shown strong potential in image restoration. HI-Diff [7] performs diffusion in latent space to generate priors for a regression-based model, reducing computational cost and improving distortion metrics. DeblurDiff [8] leverages pre-trained Stable Diffusion models guided by a Latent Kernel Prediction Network (LKPN) to enhance structure and detail preservation in real-world deblurring.

Despite progress in general deblurring, text-specific methods using modern deep learning remain limited. One such effort is DeepDeblur [9], which applied deep networks to approximate blind deconvolution for text images, exploring various architectures tailored for this task.

This project builds on recent advances in generative models to tackle the specific challenges of single-image text deblurring.

3 Background

In this section, we provide a more theoretical overview of the two main generative models used in this project: Denoising Diffusion Implicit Models (DDIMs) and Conditional Flow Matching (CFM).

3.1 Conditional DDIM

Denoising Diffusion Implicit Models [1] (DDIMs) are a class of generative models that, like Denoising Diffusion Probabilistic Models [10] (DDPMs), learn to reverse a forward noising process. DDIMs introduce a non-Markovian reverse process, enabling deterministic sampling and faster generation, while sharing the same training objective and forward diffusion mechanism as DDPMs.

The forward process q gradually adds Gaussian noise to a data sample $x_0 \sim q(x_0)$ over T discrete timesteps as a Markov chain:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

with variance schedule $\{\beta_t\}_{t=1}^T$. Direct sampling from x_0 is given by:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$. As $t \to T$, $x_T \sim \mathcal{N}(0, I)$.

The key distinction of DDIMs lies in the reverse process p_θ, which is non-Markovian. It transforms $x_T \sim \mathcal{N}(0, I)$ back to x_0, potentially conditioned on some input y. The reverse step is defined as:

$$\begin{aligned} x_{t-1} = \sqrt{\bar{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(x_t, t, y)}{\sqrt{\bar{\alpha}_t}} \right) \\ + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} \cdot \epsilon_\theta(x_t, t, y) + \sigma_t \epsilon_t \end{aligned} \tag{1}$$

where $\epsilon_\theta(x_t, t, y)$ estimates the noise added to x_0. Setting $\sigma_t = 0$ for all t yields a fully deterministic process, enabling efficient sampling using a reduced timestep set $\{\tau_1, \ldots, \tau_S\} \subset \{1, \ldots, T\}$, with $S < T$.

The model, typically a U-Net, is trained to predict ϵ from noisy input:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$$

using a mean squared error objective between the true and predicted noise:

$$\mathcal{L}_{\text{DDIM}}(\theta) = \mathbb{E}_{t,x_0,\epsilon,y}\left[\|\epsilon - \epsilon_\theta(x_t, t, y)\|^2\right] \tag{2}$$

where $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$, and y is the conditioning variable.

3.2 Conditional Flow Matching

Conditional Flow Matching (CFM), introduced by Tong et al. [11] and building on Lipman et al. [2], offers an alternative framework for training Continuous Normalizing Flows (CNFs) without simulating full probability paths. CNFs learn an invertible mapping $x_1 = \Phi(x_0)$ from a simple prior $p_0(x_0)$ (e.g., Gaussian) to a complex target distribution $p_1(x)$, defined via the ODE:

$$\frac{dx_t}{dt} = v_t(x_t)$$

The evolution of the density $p_t(x)$ is governed by the continuity equation:

$$\frac{d}{dt}p_t(x) + \nabla \cdot (p_t(x)v_t(x)) = 0$$

The standard Flow Matching (FM) objective minimizes the difference between a learnable vector field $v_t(x;\theta)$ and a target field $u_t(x)$:

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t\sim U[0,1], x\sim p_t}\left[\|v_t(x;\theta) - u_t(x)\|^2\right] \tag{3}$$

However, since $p_t(x)$ and $u_t(x)$ are often intractable, CFM reformulates this using conditional paths $p_t(x|x_0, x_1)$ and fields $u_t(x|x_0, x_1)$, where $x_0 \sim p_0(x)$ and $x_1 \sim q(x)$. These paths interpolate from x_0 to x_1, and their average recovers the desired marginal path $p_t(x)$.

The CFM objective is:

$$\begin{aligned}\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t\sim U[0,1], x_0\sim p_0, x_1\sim q, x\sim p_t(x|x_0,x_1)} \\ \left[\|v_t(x;\theta) - u_t(x|x_0, x_1)\|^2\right]\end{aligned} \tag{4}$$

This formulation is tractable and only requires sampling from known distributions and computing simple vector fields.

This work adopts Independent Conditional Flow Matching (I-CFM), a variant where x_0 and x_1 are sampled independently. The conditional paths are Gaussian:

$$p_t(x|x_0, x_1) = \mathcal{N}(x \mid \mu_t(x_0, x_1), \sigma_t^2 I)$$

with

$$\mu_t(x_0, x_1) = (1 - t)x_0 + tx_1, \quad \sigma_t = \sigma_{\text{const}}$$

The corresponding conditional vector field is:

$$u_t(x|x_0, x_1) = x_1 - x_0$$

This time-independent and state-independent vector field results in a straight-line interpolation between samples, contributing to stable and efficient training.

I-CFM enables simulation-free training of CNFs with high efficiency, offering benefits such as faster convergence, reduced complexity, and strong performance in generative modeling tasks.

4 Methodology

This section outlines the methodology used in this work, covering dataset generation, the conditional diffusion and flow matching models, as well as implementation and evaluation details.

4.1 Dataset Generation

We constructed a custom dataset of paired sharp and blurred text images to train our models. The generation pipeline was designed to ensure diversity across text content, styles, and degradation types.

Text Corpus. Text was sourced from Project Gutenberg [12] using the `generate_corpora.py` script, which aggregated content from various ebook IDs into a single corpus (`corpus.txt`).

Fonts. We collected 3626 fonts from the Google Fonts GitHub repository [13], providing broad typographic variability.

Dataset Generation. The core image generation process was handled by `generate_dataset_DB.py`. For each sample, the following steps were applied:

1. **Text Selection:** Random lines were selected from the corpus (1 to 4 lines, up to 3 words per line).
2. **Font Styling:** A random font and size (18 to 72 px) were chosen, with RGB text color sampled from $(0, 0, 0)$ to $(60, 60, 60)$.
3. **Background Generation:** Backgrounds were randomly selected as:
   - Solid color: RGB $(200, 200, 200)$ to $(255, 255, 255)$ (40% of the time),
   - Noise (40% of the time),
   - Texture from a set of 67 images [14] (20% of the time).
4. **Rendering:** The text was rendered onto the background using the chosen style, producing the sharp image.
5. **Blurring:** One of the following blur types was applied:
   - **Gaussian blur** (40% of the time): Radius 1.0-5.0.
   - **Motion blur** (30% of the time): 1D kernel with length 6-40 px and angle 0-360°, using `scipy.signal.convolve2d`.
   - **Defocus blur** (30% of the time): Disk kernel with radius 2-10 px.
6. **Final Degradations:** Additive Gaussian noise (intensity 2-10) and JPEG compression (quality 40-85) were applied for realism.
7. **Output:** The final sharp and blurred image pair was saved.

This pipeline produced a dataset of 100,000 samples with resolution 480×270 pixels, capturing a broad range of text densities, fonts, backgrounds, and degradation patterns to support robust model training.

4.2 Conditional DDIM

For our Conditional DDIM model, we adopted a U-Net architecture adapted from OpenAI's Guided Diffusion repository [15]. U-Net's hierarchical encoder-decoder design with skip connections is particularly effective for diffusion-based image restoration, as it captures multi-scale context while preserving fine details.

DDIM Training Procedure

The model ϵ_θ is trained to predict the noise added to a clean image x_0 to produce a noisy version x_t, conditioned on the corresponding blurred image y. We use $T = 1000$ diffusion steps with a linear noise schedule: $\beta_{\text{start}} = 10^{-4}$, $\beta_{\text{end}} = 2 \times 10^{-2}$.

The training process is as follows:

1. **Data Loading:** Sharp-blurred image pairs (x_0, y) are loaded from the dataset. Both are resized and normalized to $[-1, 1]$.
2. **Forward Diffusion:** For each pair:
   - Sample timestep $t \sim U[1, T]$ and noise $\epsilon \sim \mathcal{N}(0, I)$.
   - Compute noisy image:
   $$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon$$
3. **Noise Prediction:** The model receives x_t, timestep t (via embedding), and blurred image y as inputs. These are concatenated channel-wise and passed through the U-Net, producing $\epsilon_\theta(x_t, t, y)$.
4. **Loss Computation:** The objective is the MSE between the true and predicted noise:
   $$\mathcal{L} = \|\epsilon - \epsilon_\theta(x_t, t, y)\|^2$$
5. **Optimization:** Parameters θ are updated via Adam optimizer using gradients of the loss.

DDIM Sampling Procedure

Once trained, the U-Net model ϵ_θ is used to generate a sharp image from a blurred input y via the DDIM sampling process, which reverses the forward diffusion using a reduced set of steps $S < T$, enabling faster inference compared to DDPMs.

The sampling procedure proceeds as follows:

1. **Initialization:** Begin with $x_S \sim \mathcal{N}(0, I)$ and the conditioning blurred image y. Select a sequence of S linearly spaced timesteps $\{\tau_S, \tau_{S-1}, \ldots, \tau_1\} \subset \{1, \ldots, T\}$.
2. **Iterative Denoising:** For each step $i = S$ down to 1:
   - Predict the noise: $\epsilon_\theta(x_t, t, y)$.
   - Estimate the clean image:
   $$\hat{x}_0 = \frac{x_t - \sqrt{1 - \bar{\alpha}_t}\, \epsilon_\theta(x_t, t, y)}{\sqrt{\bar{\alpha}_t}}$$
   - Compute the next sample using:
   $$x_{t'} = \sqrt{\bar{\alpha}_{t'}} \hat{x}_0 + \sqrt{1 - \bar{\alpha}_{t'} - \sigma_t^2} \cdot \epsilon_\theta(x_t, t, y) + \sigma_t \epsilon$$
   where $\epsilon \sim \mathcal{N}(0, I)$, and σ_t controls the stochasticity. Setting $\sigma_t = 0$ yields a deterministic process.
3. **Final Output:** After S steps, the final x_0 is returned as the deblurred image.

In this work, we use $S = 100$ steps and set $\sigma_t = 0.0$ for deterministic sampling. This iterative refinement gradually transforms Gaussian noise into a sharp image, conditioned on the blurred input.

4.3 Conditional Flow Matching (CFM)

We adopt Independent Conditional Flow Matching (I-CFM) [11] using the same U-Net architecture employed in Conditional DDIM. Unlike DDIM, the network v_θ is trained to predict the time-dependent vector field $v_t(x; \theta)$, which transports samples from a prior distribution p_0 to the data distribution p_1, conditioned on a blurred input y.

I-CFM Training Procedure

The model learns to align its predicted vector field $v_\theta(x_t, t, y)$ with the target conditional vector field $u_t(x_t|x_0, x_1)$, following the I-CFM objective (Equation 4).

1. **Path Sampling:** For each sharp-blurred pair (x_1, y):
   - Sample noise image $x_0 \sim \mathcal{N}(0, I)$ and time $t \sim U[0, 1]$.
   - Compute the intermediate point:
   $$x_t = (1-t)x_0 + t x_1 + \sigma \epsilon', \quad \epsilon' \sim \mathcal{N}(0, I),\ \sigma = 0.01$$
   - Define the target vector field:
   $$u_t(x_t|x_0, x_1) = x_1 - x_0$$
2. **Vector Field Prediction:** The U-Net v_θ receives x_t, time t (as a continuous embedding), and conditioning image y, concatenated along the channel dimension. It outputs the predicted vector field $v_\theta(x_t, t, y)$.
3. **Loss Computation:** The objective is the mean squared error between the predicted and target vector fields:
   $$\mathcal{L}_{\text{I-CFM}} = \mathbb{E}_{t, x_0, x_1, y}\left[\|v_\theta(x_t, t, y) - (x_1 - x_0)\|^2\right]$$
4. **Optimization:** Gradients are computed and parameters θ are updated using the Adam optimizer.

I-CFM Sampling Procedure

After training the U-Net v_θ to predict the conditional vector field, image generation is performed by solving an Ordinary Differential Equation (ODE) conditioned on a blurred input y:

1. **Initialization:** Sample the initial noise $x_0 \sim \mathcal{N}(0, I)$.
2. **ODE Integration:** The sharp image x_1 is obtained by solving the initial value problem:
   $$\frac{dx_t}{dt} = v_\theta(x_t, t, y), \quad \text{with } x(0) = x_0$$

Integration is performed over $t \in [0, 1]$ using the Dormand–Prince 5 (DOPRI5) Runge–Kutta solver [16] with adaptive step size and error tolerances of 1×10^{-3} (relative and absolute).

3. **Final Output:** The solution at $t = 1$, denoted x_1, is returned as the generated deblurred image.

4.4 Training Details

Both Conditional DDIM and Conditional Flow Matching models were trained on an NVIDIA A100 GPU using the same dataset of 100,000 samples, split into 80,000 for training and 20,000 for testing. To ensure comparability, shared training procedures and hyperparameters were applied where applicable.

Models were trained for up to 30 epochs with a batch size of 12, using the Adam optimizer with an initial learning rate of 1×10^{-4}. A learning rate scheduler reduced the rate by a factor of 0.5 if the Peak Signal-to-Noise Ratio (PSNR) on a fixed validation subset of 100 test samples did not improve for 5 consecutive epochs. The best-performing model (based on PSNR) was saved via checkpointing.

Since validation loss may plateau before perceptual quality improves in diffusion and flow-based models, PSNR was used as the primary metric for model selection. Gradient clipping with a maximum norm of 1.0 was applied throughout training to improve stability.

4.5 Evaluation Details

Model performance was assessed using Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM), computed between the generated deblurred images and their corresponding ground truth sharp images from the 20,000-sample test set.

- **PSNR** quantifies pixel-wise differences in intensity. Higher PSNR (in decibels) indicates greater fidelity to the original image.
- **SSIM** evaluates structural similarity, brightness, and contrast, aligning more closely with human perception. An SSIM value near 1 indicates high structural similarity.

The evaluation procedure involved:

1. **Image Generation:** For each batch of test samples, deblurred outputs were generated using the respective sampling procedures (Sections 4.2 and 4.3).
2. **Metric Computation:** PSNR and SSIM were calculated for each output using the `torchmetrics` library.
3. **Qualitative Inspection:** Selected triplets (blurred input, generated output, and ground truth) were saved for visual comparison.

5 Results

Table 1 reports the PSNR (in dB), SSIM, sampling time per image, and the number of training epochs required to reach the best PSNR for both models. As both methods use the same architecture and batch size, the epoch count provides a fair measure of convergence speed. Sampling times were measured on an NVIDIA RTX 4060 Laptop GPU.

Model	PSNR	SSIM	Epochs	Samp. Time (s)
DDIM	12.16	0.50	23	25
CFM	**25.93**	**0.72**	**14**	**20**

Table 1: Evaluation results of the Conditional DDIM and Conditional Flow Matching models on the test set. Best results are highlighted in bold.

The CFM model achieved its best PSNR after 14 epochs, compared to 23 epochs for DDIM, indicating faster convergence. It also outperformed DDIM in both PSNR and SSIM, while offering faster sampling. These results demonstrate the advantages of CFM in training efficiency and overall performance, despite both models sharing identical architectures and training conditions.

5.1 Qualitative Results

Additionally, qualitative results are shown for two samples in Figures 1 and 2.

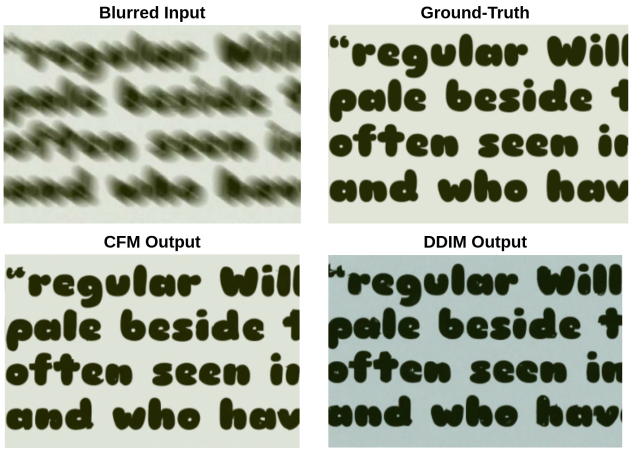

Figure 1: Qualitative results for a sample with strong motion and defocus blur and a simple background.

Figure 1 illustrates a case with strong motion and defocus blur on a simple background. Both models successfully reconstruct the text with high fidelity, preserving font and letter structure. However, DDIM slightly misrepresents the background color, while CFM reproduces it accurately.

Figure 2 presents a more challenging case with strong defocus blur and a textured background. CFM recovers most letters, albeit with some inaccuracies, whereas DDIM fails to generate legible text and also misrepresents the background.

5.2 Discussion

The results show that the CFM model outperforms DDIM in both PSNR and SSIM, while also offering faster sampling, consistent with findings by Lipman et al. [2]. Background mismatches observed in DDIM outputs could potentially be mitigated by training in a latent space, as proposed by

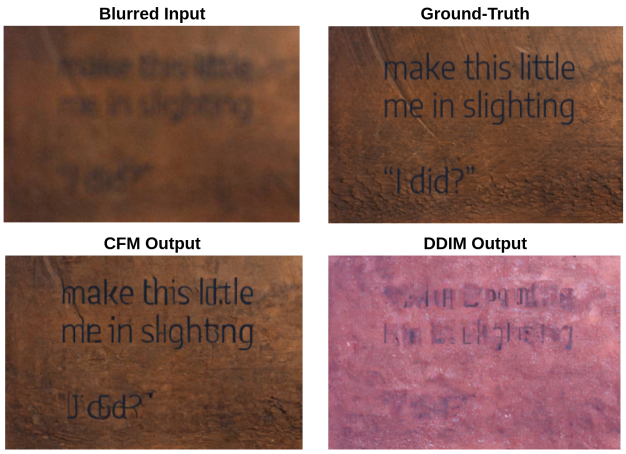

Figure 2: Qualitative results for a sample with strong defocus blur and a textured background.

Rombach et al. [17], where it would be necessary to train a VAE to encode the images into a latent space.

Overall, CFM meets the project's objective of recovering sharp images from blurred inputs. Future improvements could include training on a larger and more diverse dataset and leveraging latent-space modeling for greater computational efficiency.

6 Conclusion

This project tackled the task of single-image text deblurring by comparing two advanced generative methods: Conditional Denoising Diffusion Implicit Models (DDIM) and Conditional Flow Matching (CFM). Both models were trained and evaluated on a custom dataset capturing a wide range of text styles and blur types.

Beyond deblurring quality (measured by PSNR and SSIM), we assessed training efficiency and sampling speed. CFM outperformed DDIM across all metrics while converging faster and requiring less sampling time. These findings support the theoretical advantages of CFM, particularly its simulation-free training and more stable optimization.

Future work could explore the use of these models on real-world blurred text images and evaluate their effectiveness as pre-processing steps in Optical Character Recognition (OCR) pipelines. As discussed in Section 5.2, further gains may be achieved by training in latent space and expanding the dataset's size and diversity.

References

[1] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *ICLR*, 2021.

[2] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *ICLR*, 2023.

[3] J. Pan, Z. Hu, Z. Su, and M.-H. Yang, "Deblurring text images via l0-regularized intensity and gradient prior," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2901–2908.

[4] S. Nah, T. Hyun Kim, and K. Mu Lee, "Deep multi-scale convolutional neural network for dynamic scene deblurring," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[5] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, and J. Matas, "Deblurgan: Blind motion deblurring using conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8183–8192.

[6] O. Kupyn, T. Martyniuk, J. Wu, and Z. Wang, "Deblurgan-v2: Deblurring (orders-of-magnitude) faster and better," in *Proceedings of the IEEE/CVF*, 2019, pp. 8878–8887.

[7] Z. Chen, Y. Zhang, D. Liu, J. Gu, L. Kong, X. Yuan *et al.*, "Hierarchical integration diffusion model for realistic image deblurring," *NeurIPS*, vol. 36, 2024.

[8] L. Kong, J. Zhang, D. Zou, J. Ren, X. Wu, J. Dong, and J. Pan, "Deblurdiff: Real-world image deblurring with generative diffusion models," *arXiv preprint arXiv:2502.03810*, 2025.

[9] J. Mei, Z. Wu, X. Chen, Y. Qiao, H. Ding, and X. Jiang, "Deepdeblur: text image recovery from blur to sharp," *Multimedia Tools and Applications*, 2019.

[10] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *NeurIPS*, 2020.

[11] A. Tong, K. FATRAS, N. Malkin, G. Huguet, Y. Zhang, J. Rector-Brooks, G. Wolf, and Y. Bengio, "Improving and generalizing flow-based generative models with minibatch optimal transport," *TMLR*, 2024.

[12] Project Gutenberg Literary Archive Foundation, www.gutenberg.org, accessed 2025-05-09.

[13] Google Fonts, github.com/google/fonts, accessed 2025-05-09.

[14] TextureLabs, texturelabs.org/, accessed 2025-05-09.

[15] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *NeurIPS*, vol. 34, pp. 8780–8794, 2021.

[16] J. Dormand and P. Prince, "A family of embedded runge-kutta formulae," *Journal of Computational and Applied Mathematics*, vol. 6, no. 1, pp. 19–26, 1980.

[17] 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*, 2022, pp. 10 684–10 695.