JSON: Jigsaw Self-play Optimization for Normalizing Flows

Fengxiang Yang[1], Tianyi Zheng[1], Jinwei Chen[1], and Bo Li[1†]

[1]vivo BlueImage Lab, vivo Mobile Communication Co., Ltd., China
{yangfx, zhengtianyi, libra}@vivo.com

Abstract. Normalizing Flows (NFs) is a principle generative framework, which learns to generate images by establishing a bijective mapping trajectory between noise and data with likelihood maximization. In this paper, we leverage the self-play paradigm, a general trajectory fine-tuning method to improve generative capability of NFs without auxiliary fine-tuning data. The general self-play mechanism formulates competition between model and its historical counterpart in favoring real data trajectory (*i.e.*, "wining trajectory") while forgetting trajectory of self-generated synthesized data (*i.e.*, "losing trajectory"), improving model capability through preference alignment. However, NF-based self-play faces two challenges. (1) Enforcing NF to forget data synthesized with historical model contradicts with NF's original training goal, leading to potential training collapse. (2) Self-play is naturally plausible for generative model with pre-defined data trajectory (*e.g.*, diffusion models), but faces compatibility problem with NFs due to its implicit wining data trajectory definition. We thus propose jigsaw self-play optimization for NFs (JSON) to overcome these challenges. Our core contribution is the integration of a jigsaw puzzle reassembly task to serve as an intrinsic spatial coherence evaluator, identifying "winning" trajectories that exhibit superior spatial logic for self-play fine-tuning. We further propose a bounded self-play loss and a nested-loop optimization strategy to mitigate training collapse and ensure stable fine-tuning. By aligning the model with high-quality structural anchors mined during self-play, JSON achieves promising results on ImageNet-1K, providing a robust framework for high-fidelity NF-based generation.

Keywords: Normalizing Flows · Generative Models

1 Introduction

The rapid development of generative models have achieved remarkable success, driven by the advancements in Diffusion Models (DMs) [20, 37], Auto-Regressive models (ARs) [44, 53], and Normalizing Flows (NFs) [15, 56, 57]. Building upon "change of variable formula", modern NFs [15, 56] implicitly learn the dynamics

† Corresponding author.

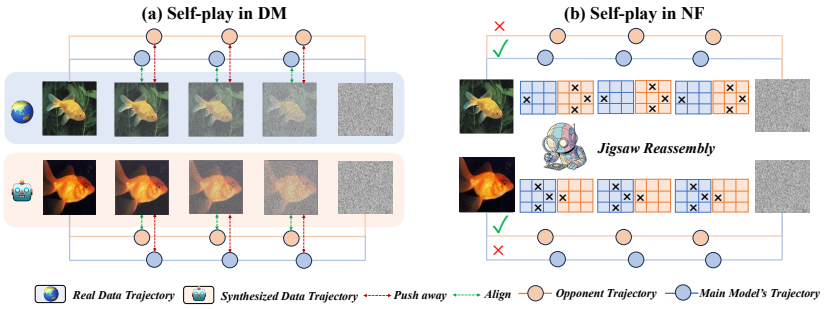


Fig. 1: Schematic illustration of DM-based self-play and our NF-based self-play. (a) DMs explicitly definite data-noise dynamics, enabling accurate preference alignment and self-play between main model and its competitor; (b) NFs implicitly achieve data-noise transformation, posing a challenge for "wining" and "losing" trajectory definition. We propose to integrate jigsaw reassembly into NF's self-play, providing a complementary way to define wining / losing trajectories and benefiting subsequent fine-tuning.

from standard Gaussian noise to real world data, without imposing time-scheduled trajectories [29, 32]. This unique trait eliminates the need for iterative denoising, facilitating efficient sampling compared to its continuous-time, flow-matching counterpart [4, 13]. However, NFs often require specialized layers to maintain strict invertibility and ensure tractable Jacobian determinant computation, constraining model's expressive capacity [24, 33, 56]. Consequently, scaling NFs to generate high-fidelity images without violating inherent constraints remains a challenge.

Generally speaking, enhancing generative capability relies on *advanced architecture design* [12, 34], or *trajectory fine-tuning (TFT)* [14, 27, 47, 54]. TarFlow [56] and StarFlow [15] establish modern NFs by introducing scalable and novel architecture. Although effective, NF's implicit training and under-constrained transformation trajectory hinder high-level semantic learning [25], increasing the training difficulty. As a representative method to improve model's generative capability, TFT has achieved a huge success in diffusion by requiring model to align with diffusion trajectories that finally leads to high quality images. The whole process requires manually assigning each trajectory with human [47] or AI preference [22, 30] to encourage the emergence of better noise-image transformation. For example, SimFlow-REPA-E [57] achieves this by incorporating pre-trained vision encoders [36] for feature (*i.e.*, trajectory) alignment, but it introduces additional overhead. Other solutions adopt PPO [1] or DPO-based [47] reinforcement learning to achieve this. However, TFT's application in NFs is largely hindered by the high cost of preference collection [38]. Recently, Large Language Models (LLMs) and DMs have successfully adopted self-play [2, 7, 43, 55] for more efficient training. As demonstrated in Fig. 1-(a), self-play TFT for DMs can be summarized as a competition between two models (main model and its competitor) for better alignment with real trajectory generated by diffusion dynamics (*e.g.*, DDIM [42]), where main model should achieve better / worse trajectory alignment than its competitor for real / synthesized data. It is worth noting that main model and

its competitor refer to the same model, but instantiated with parameters from different epochs. The intrinsic preference eliminates the requirement of preference set, showing promising results. However, self-play is not compatible with NFs due to two challenges. (1) Directly deploying self-play to under-fit synthesized data will contradict with its previous training goal, leading to the potential training collapse. (2) NFs lack the pre-defined real data trajectories, causing challenges in discerning wining and losing trajectories. Although likelihood itself could be used to define wining samples, concurrent works claim its deficiencies in terms of spatial unconscious [23] and low sensitivity [49]. Therefore, it is necessary to seek for a complementary metric.

In this paper, we introduce JSON (Jigsaw Self-play Optimization for Normalizing flows), effectively boosting the generative capacity of NFs. (1) Considering the unstable training with original self-play objective, we propose a bounded self-play that limits the unconstrained forgetting of synthesized data to avoid training collapse. (2) To formulate wining and losing trajectories, we use jigsaw puzzle reassembly as complementary task and organize our training into two stages, *i.e.*, NF pre-training and self-play fine-tuning. During NF pre-training, we deploy jigsaw head at intermediate NF block, feeding it with features that are deliberately random-shuffled at the spatial dimensions for the prediction of their original locations. The idea is built upon the hypothesis that high-fidelity visual representations must maintain spatial coherence and be friendly to jigsaw puzzle reassembly [3, 35]. At NF self-play stage, the trained jigsaw head provides spatial coherence evaluation as shown in Fig. 1-(b) for both real and synthesized data, judging whether synthesized trajectory should be focused by using bounded self-play or self-play solely with synthesized data. Based on the proposed two mechanism, we design a nested-loop (inner- & outer-loop) training pipeline. (1) At inner-loop self-play, the competitor model (snapshot model at last epoch) synthesize images based on the given condition, conducting self-play with main model, bounded self-play, and jigsaw puzzle reassembly. (2) At the outer-loop self-play, the competitor is updated with main model to track the learned evolving distribution. JSON bootstraps the generative performance of NFs by alternatively training between inner and outer loops. To sum up, our contributions can be summarized into three points:

- We introduce jigsaw puzzle reassembly as the auxiliary task for NF training, encouraging spatial coherence representation learning while establishing intrinsic trajectory evaluation.
- We formulate a novel self-play fine-tuning, termed as "JSON". The algorithm is built upon novel bounded self-play and inherent jigsaw evaluation, avoiding potential training collapse and improving NF generative capability.
- JSON achieves state-of-the-art performance on ImageNet-1K benchmark, demonstrating a novel and promising solution for NFs in image generation.

2 Related Work

Normalizing Flows. Normalizing Flows (NFs) is a family of likelihood-based generative models [10, 11, 15, 16, 19, 24, 40, 56], synthesizing images or videos from simple prior distribution (*e.g.*, standard Gaussian) with a series of inherent and invertible transformations. Building upon change of variable formula, NFs eliminates iterable denoising steps, improving generation efficiency and is thus different from typical diffusion models [13, 20]. Early NFs [11, 19, 24] are built upon restrictive architectures to meet the constraint of invertible transformation and easy determinant computation, limiting model's generative capacity. TarFlow [56] advances NFs by introducing causal transformer for model's scalability. StarFlow [15] diverts the training space from pixel to latent, devising deep-shallow architecture for more stable training and better sampling. Recently, SimFlow [57] proposes joint training pipeline for NF and further boosting its generative capability with pretrained vision encoder (*e.g.*, DINOv2 [36]) in REPA-like [54] manner. Different from previous works, we construct a self-play training pipeline, enabling NF to enhance generative capability without pre-trained vision encoders.

Trajectory Fine-tuning. With the success of preference alignment in LLMs [12, 45], trajectory fine-tuning is becoming the main-stream technology to scale language and vision models in an online [1, 12, 14] or offline reinforcement learning manner [38, 47, 55], which assigns high reward to trajectories that leads to high-quality results and encourages their emergence. Although effective, most research inject the preference through pre-trained reference model [31, 52], introducing high cost. Recently, the success of reinforcement learning with verifiable reward (RLVR) [7, 17, 21] in LLM and MLLM [50] has inspired the research of fine-tuning generative models with correct and easily obtained reward. However, most RL fine-tuning are restricted within diffusion [20]. Our paper mainly focuses on NFs and establishes a self-contained trajectory fine-tuning method by training jigsaw puzzle reassembly head at pre-training stage, which formulates subsequent RLVR paradigm for better generative capability.

3 Methodology

3.1 Preliminaries

Normalizing Flows. An NF model $f_\theta := f_{n-1} \circ f_{n-2} \circ \cdots \circ f_0$, parameterized by θ, is typically consists of n invertible blocks f_i $(0 \leq i < n)$ to establish the step-by-step bijective transformation between real data distribution $p(x_0)$ and a simple Gaussian prior $p(\mathbf{z})$. During the forward process, f_θ constructs model distribution p_{nf} through "change-of-variable formula", which could be formulated as:

$$p_{\text{nf}}(x;\theta) = p_0(\mathbf{z}) \prod_{i=0}^{n-1} |\det\Big(\frac{\partial f_i(x_i)}{\partial x_i}\Big)|, \tag{1}$$

where x_i is the input of i-th NF block. $\det(\cdot)$ is the determinant operator. $\mathbf{z} = f_\theta(x_0)$ is NF's final output, conditioned on NF trajectory $\mathcal{T} = \{x_i | 0 \leq i \leq n-1\}$. The training is thus performed by maximizing log-likelihood (MLE) over the given training data x_0:

$$\mathcal{L}_{\text{nf}}(x_0;\theta) = -\log p_{\text{nf}}(x;\theta) = -\log p_0(\mathbf{z}) - \sum_{i=0}^{n-1} \log|\det\Big(\frac{\partial f_i(x_i)}{\partial x_i}\Big)|. \tag{2}$$

Considering f_i's invertibility and Jacobian determinant's high computational cost, specially designed architectures like affine coupling [10, 11] or causal transformer-based auto-regressive flows [56, 57] are required. We take the latter for further demonstration. Under this scenario, the i-th NF block predicts the scaling α_i and shifting μ_i parameters conditioned on all previous input tokens to perform an affine transformation:

$$x_{i+1,m} = \begin{cases} x_{i,m} & m{=}0 \\ (x_{i,m} - \mu_{i,m}) \odot \exp(-\alpha_{i,m}) & \text{otherwise,} \end{cases} \tag{3}$$

where $0 \leq m \leq HW - 1$ and H, W are height and width of latents. As the Jacobian matrix of a causal transformer is lower triangular, the $\det(\cdot)$ term simplifies to the product of diagonal entries [56]:

$$\mathcal{L}_{\text{nf}}(x_0;\theta) = \frac{1}{2}||x_n||_2^2 + \sum_{i=0}^{n-1}\sum_{m=0}^{HW-1}\sum_{d=0}^{D-1}\alpha_{i,m,d}, \tag{4}$$

where D is the feature dimension. Once the training is finished, its inverse function $f_\theta^{-1} := f_0^{-1} \circ f_1^{-1} \circ ... \circ f_{n-1}^{-1}$ estimates α_i and μ_i with next-token prediction, enabling kv-cache [45] for faster inference and achieving generation from Gaussian noise $\epsilon \sim \mathcal{N}(0, 1)$ by reversing Eq. 3. Our baseline follows this causal transformer paradigm [56] but operates within a latent space [15, 57], where input images $\mathbf{I}$ are first encoded with a VAE-encoder ψ to obtain latent x_0 for subsequent training, while generated $\hat{x}$ are decoded with VAE-decoder ϕ to obtain images.

Self-play in NF. Trajectory Fine-tuning (TFT) encourages model to mimic trajectories that leads to high-quality images while suppresses the emergence of low-quality ones. Previous works have demonstrated its effectiveness on DMs through feature alignment [54], online [1, 14, 30], or offline [47] reinforcement learning. However, highly-reliable vision encoder [36], reward model [51] or manually annotating the preference set are costly and may introduce additional overhead. Self-play [55] emerges as a remedy by constraining the current model θ to compete with its predecessor θ_k in capturing pre-determined real data dynamics. As NFs also establish a noise-data evolving trajectory, a similar paradigm can be adapted to enhance their generative capability.

Generally speaking, for the given real data x and their condition c, we denote x' as the synthesized images conditioned on c with θ_k. The objective for self-play

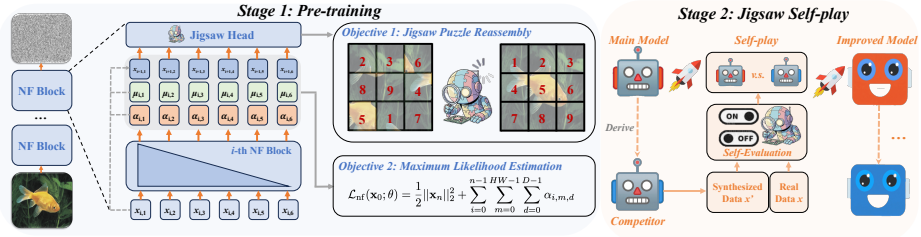


Fig. 2: Overall pipeline of our jigsaw self-play. We formulate our overall training into "stage 1: pre-training" and "stage 2: jigsaw self-play". At stage 1, we insert additional jigsaw head to manually perform jigsaw puzzle reassembly task. At stage 2, we formulate the self-play between current model (main model) and its competitor (model at last epoch) to improve NF generative capability. This is achieved by first synthesizing data and conducting self-evaluation with frozen jigsaw head to discern wining and losing trajectory for trajectory alignment (Eq. 8).

is defined with Bradly-Terry model [2]:

$$\begin{aligned}\mathcal{L}_{\text{sp}}(x, x', \theta_k; \theta) &= \mathbb{E}\Big[\sigma(\log \frac{p_\theta(x)}{p_{\theta_k}(x)} - \log \frac{p_\theta(x')}{p_{\theta_k}(x')})\Big] \\ &= \sigma(-\mathcal{L}_{\text{nf}}(x;\theta) + \mathcal{L}_{\text{nf}}(x;\theta_k) + \mathcal{L}_{\text{nf}}(x';\theta) - \mathcal{L}_{\text{nf}}(x';\theta_k)),\end{aligned} \tag{5}$$

where $\sigma(\cdot)$ is a monotonically decreasing function (*e.g.*, $\sigma(x) = -\log(1/(1+\exp(-x)))$). $p_\theta(\cdot)$ is the predicted marginal distribution, which has various definitions over different generative models. By substituting $p_\theta(\cdot)$ and $p_{\theta_k}(\cdot)$ with Eq. 1 we formulate a basic self-play algorithm (B-NSP) under the context of NF, which intuitively encourages θ / θ_k to achieve smaller loss than θ_k / θ on real data x / synthesized data x'. This imposes a generative constraint such that $p_\theta(x) > p_{\theta_k}(x)$ and $p_\theta(x') < p_{\theta_k}(x')$.

3.2 Jigsaw Self-play Optimization

Limitations of B-NSP. Despite the intuitive design, B-NSP suffers from two fundamental issues. (1) Eq. 5 enforces θ to "under-fit" x'. However, as x' is generated by θ_k, the snapshot of θ at last epoch, these samples inherently reside within the model's high-likelihood region. Penalizing them creates a direct conflict with the NF's foundational MLE objective, leading to the potential training collapse. (2) NF training relies on maximum likelihood estimation (MLE), prioritizing global density matching without anchoring trajectories obtained through diffusion dynamics [20]. Consequently, B-NSP blindly follows MLE objective while ignoring potentially valuable information within x'. Moreover, concurrent works [23, 49] note the limitations of likelihood modeling within generative models, which further demonstrates the necessity of introducing trajectory-level constrains.

Bounded Self-play. Considering the limitation-(1) of B-NSP, we propose a revised preference relation for fine-tuning, termed as "bounded self-play". Specifi-

cally, we constraint $p_\theta(x')$ to $p_{\theta_k}(x)$ and change self-play objective to:

$$\mathcal{L}_{\text{bsp}}(x, x', \theta_k; \theta) = \mathbb{E}\Big[\sigma(\log \frac{p_\theta(x)}{p_{\theta_k}(x)}) - \sigma(\log \frac{p_{\theta_k}(x)}{p_\theta(x')}) - \sigma(\log \frac{p_\theta(x')}{p_{\theta_k}(x')})\Big]$$
$$= \sigma(\mathcal{L}_{\text{nf}}(x; \theta_k) - \mathcal{L}_{\text{nf}}(x; \theta)) - \sigma(\mathcal{L}_{\text{nf}}(x'; \theta) - \mathcal{L}_{\text{nf}}(x; \theta_k)) - \sigma(\mathcal{L}_{\text{nf}}(x'; \theta_k) - \mathcal{L}_{\text{nf}}(x'; \theta)). \tag{6}$$

The revision avoids potential training collapse by modifying the generative constraint to $p_{\theta_k}(x) < p_\theta(x') < p_{\theta_k}(x')$. Therefore, θ tries to strike the balance between "forgetting synthesized trajectories" ($p_\theta(x') < p_{\theta_k}(x')$) and "staying at θ_k's high-likelihood region" ($p_{\theta_k}(x) < p_\theta(x')$).

Jigsaw Puzzle Reassembly for Spatial Coherence. As demonstrated in B-NSP's limitation-(2), lacking pre-defined diffusion dynamics increases the difficulty of defining wining and losing trajectories for self-play fine-tuning. Therefore, B-NSP fails to learn potentially valuable information within trajectories of synthesized data. We thus integrate a specialized jigsaw head into Normalizing Flow (NF) block during NF pre-training, regularizing the model to perform spatial reassembly tasks [35]. Specifically, a jigsaw head g_i is deployed at the output of each invertible block f_i. For the intermediate feature $o_{i+1} = f_i(x_i) \in \mathbb{R}^{HW \times D}$, we patchify it with patch size p and assign ground-truth spatial index labels to each patchified token along the HW dimension in a zigzag scanning order. These tokens are then stochastically permuted to form $\tilde{o}_{i+1}$, which serves as the input of $g_i(\cdot)$. The optimization objective for the jigsaw head is to predict the original spatial coordinates of each shuffled token:

$$\mathcal{L}_{\text{jig},i}(x; \theta) = -\sum_m \mathcal{Y}_m \odot \log[\text{softmax}(g(\tilde{o}_{i+1,m}))], \tag{7}$$

where $\mathcal{Y}_m \in \mathbb{R}^{HW/p^2}$ is the one-hot encoded ground-truth location of the m-th ($1 \leq m \leq HW/p^2$) permuted token. *As jigsaw head is not initially introduced in NF, we organize our training as two-stage training pipeline.* As illustrated in Fig. 2, we adopt jigsaw head and Eq. 7 during NF pre-training (Fig. 2-"Stage 1") to equip NF with spatial evaluation capability, while freezing NF heads during subsequent self-play stage. On the one hand, $\mathcal{L}_{\text{jig}}$ explicitly requires NF to learn spatially coherent features from real data x, benefiting NF to find optimal affine parameters; on the other hand, frozen jigsaw heads enable us to explicitly assess transformation trajectories of synthesized data x' during self-play.

Jigsaw Self-play Optimization for NFs. Building upon the synergy between bounded self-play and jigsaw-based feedback, we formulate our "Jigsaw Self-play Optimization for NFs" (JSON) algorithm. JSON is used after NF pre-training, where jigsaw heads are already well-trained and frozen. As illustrated in Fig. 2-"Stage 2", our core strategy involves dynamically analyzing optimal training objectives based on the spatial coherence evaluated by jigsaw heads. Specifically, for the given condition c, we first synthesize x' with θ_k. The synthesized data x', coupled with real data x, are forwarded to θ to obtain jigsaw reassembly loss for spatial coherence evaluation. (1) When the trajectory of x' exhibits inferior spatial coherence to x (as evidenced by a higher $\mathcal{L}_{\text{jig}}$), we adopt our

Algorithm 1 Overall Process of JSON.

Inputs: Training images $\mathbf{I}$ from ImageNet training set $\mathcal{D}$, NF model θ, VAE-decoder ϕ, encoder ψ, total training epochs E
Outputs: NF and VAE model.

for e in E **do**
  if $e \leq 155$ **then**
    Setting $\lambda_1 = 1, \lambda_2 = 0$;
    while $\mathcal{D}$ is not Empty **do**
      Sampling images and class labels from dataset, *i.e.*, $\mathbf{I}$, $c \leftarrow \mathcal{D}$;
      Compute Eq. 10, optimize θ, ψ, ϕ;
    end while
  else
    Initialize competitor model θ_k with θ;
    Setting $\lambda_1 = 0, \lambda_2 = 1$;
    while $\mathcal{D}$ is not Empty **do**
      Sampling images and class labels from dataset, *i.e.*, $\mathbf{I}$, $c \leftarrow \mathcal{D}$;
      Synthesize x' with θ_k and c;
      Conducting Self-evaluation for x and x' based on Eq. 7 and jigsaw head;
      Compute Eq. 10, optimize θ, ψ, ϕ;
    end while
    Update θ_k with θ;
  end if
end for
Return θ, ψ, and ϕ.

bounded self-play loss $\mathcal{L}_{\text{bsp}}$ to encourage learning from real data for better spatial coherence. (2) When the trajectory of x' exhibits superior spatial coherence to that of the real data trajectory (as evidenced by a lower $\mathcal{L}_{\text{jig}}$), it indicates that real data trajectory should not be favored. Considering both constraints, we propose the following alternative self-play loss:

$$\mathcal{L}_{\text{jsp}}(x, x', \theta_k; \theta) = \begin{cases} \mathcal{L}_{\text{bsp}}(x, x', \theta_k; \theta) & \mathcal{L}_{\text{jig}}(x; \theta) \leq \mathcal{L}_{\text{jig}}(x'; \theta) \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

where $\mathcal{L}_{\text{jig}} = \sum_i \mathcal{L}_{\text{jig,i}}$ denotes the cumulative jigsaw loss across all NF blocks. In our design, we inherently evaluate the spatial coherence of real and generated images and assign optimal training objective to improve NF generative capability based on the feedback from the frozen jigsaw head.

Joint Optimization. Considering the computational efficiency, we restrict our NF training at latent space while taking standard VAE optimization into training [57].

$$\mathcal{L}_{\text{vae}}(\mathbf{I}; \psi, \phi) = ||\mathbf{I} - q_\phi(\hat{\mathbf{I}}|x)||_2^2 + \text{KL}(q_\psi(x|\mathbf{I})||p(x)), \tag{9}$$

where ψ and ϕ are VAE encoder and decoder, respectively. $\hat{\mathbf{I}}$ are the reconstructed images, $p(x)$ denotes standard Gaussian. The overall loss for our NF training

could be formulated as:

$$\mathcal{L}_{\text{total}}(\mathbf{I};\psi,\phi,\theta,\theta_k) = \mathcal{L}_{\text{vae}}(\mathbf{I};\psi,\phi) + \mathcal{L}_{\text{nf}}(x;\theta) + \lambda_1\mathcal{L}_{\text{jig}}(x;\theta) + \lambda_2\mathcal{L}_{\text{jsp}}(x,x';\theta,\theta_k), \tag{10}$$

where $x \sim q_\psi(\cdot|\mathbf{I})$ is the encoded real data while x' is synthesized data with model at last epoch. Crucially, our self-play objective ($\mathcal{L}_{\text{jsp}}$) requires an on-the-fly inference step to synthesize x' using θ_k during each training iteration. Given the non-trivial computational overhead introduced by this, we conduct two-stage training as demonstrated at "Jigsaw Puzzle Reassembly for Spatial Coherence". During the pre-training stage, the model is trained with $\mathcal{L}_{\text{vae}}$, $\mathcal{L}_{\text{nf}}$, and $\mathcal{L}_{\text{jig}}$ to establish a stable latent manifold. During the self-play stage, we conduct nested-loop self-play training. Specifically, we clone NF model (main model) after pre-training as its competitor. (1) At inner-loop self-play, competitor and main model perform jigsaw self-play by setting $\lambda_1 = 0, \lambda_2 = 1$ and freeze jigsaw heads for fine-tuning. (2) After finishing one training epoch, we perform outer-loop self-play, updating competitor with the latest NF model to track learned evolving distribution. The nested-loop training strategy allows NF to refine its generative fidelity using high-quality synthetic data without significantly increasing the cumulative training time. We summarize the overall training of JSON at Alg. 1.

4 Experiment

Experiment Setup & Evaluation Protocol. We conduct experiments on the large-scale ImageNet-1K benchmark at resolutions of 256×256 [8]. Following the established protocols in previous Normalizing Flow (NF) [15, 56, 57], we evaluate generative performance using four standard metrics: FID [18], IS, Precision, and Recall [26]. All metrics are calculated based on 50,000 generated samples.

Implementation Details. Our NF architecture follows the state-of-the-art deep-shallow design [15], comprising 5 shallow layers (each containing 2 NF blocks) and 1 deep layer consisting of 46 NF blocks. The VAE used in our method is MAR-VAE [28], which is the same as previous state-of-the-art SimFlow [57] for a fair comparison. Regarding the jigsaw head architecture, we adopt conv-based implementation to maintain the spatial information within NF features. NF model is trained with a global batch size of 160 over $E = 160$ training epochs, with patch size $p = 1$. We set pre-training epochs to 155 while self-play epochs as 5, indicating that $\lambda_1 = 1, \lambda_2 = 0$ for the initial 155 epochs while $\lambda_1 = 0, \lambda_2 = 1$ for the last 5 epochs. The overall training could be conducted on 16 NVIDIA H20 GPUs. During inference, we apply classifier free guidance (CFG) proposed in StarFlow [15] and set CFG value to 1.1. We use the minus log-sigmoid function to implement $\sigma(\cdot)$, *i.e.*, $\sigma(x) = -\log(1/(1+\exp(-x)))$.

4.1 Comparison with State-of-the-arts

We compare our method with previous Diffusion, AR, and NF models under both pixel and latent scenario. As shown in Tab. 1, we obtain the following conclusions.

Table 1: Comparison with state-of-the-art. "Blue background" : using external pre-trained vision encoder for feature alignment. "P": Precision, "R": Recall.

Methods	Epochs	#Params	w/ Guidance				w/o Guidance			
Pixel-Level Diffusion			FID	IS	P	R	FID	IS	P	R
ADM [9]	400	554M	3.94	215.8	0.83	0.53	10.94	101.0	0.69	0.63
DiP [6]	160	631M	2.16	271.8	0.82	0.61	-	-	-	-
PixelFlow [5]	320	667M	1.98	282.1	0.81	0.60	-	-	-	-
PixelNerd [48]	160	700M	2.15	297.0	0.79	0.59	-	-	-	-
Pixel-Level NF										
TarFlow [56]	320	1.4B	4.69	-	-	-	-	-	-	-
JetFormer [46]	500	2.8B	6.64	-	0.69	0.56	-	-	-	-
FARMER [59]	320	1.9B	3.60	269.2	0.81	0.51	-	-	-	-
Latent-Level Diffusion										
DiT [37]	1400	675M	2.27	278.2	0.83	0.57	9.62	121.5	0.67	0.67
SiT [34]	1400	675M	2.06	270.3	0.82	0.59	8.61	131.7	0.68	0.67
MaskDiT [60]	1600	675M	2.28	276.6	0.80	0.61	5.69	177.9	0.74	0.60
REPA [54]	800	675M	1.29	306.3	0.79	0.64	5.78	158.3	0.70	0.68
REPA-E [27]	800	675M	1.12	302.9	0.79	0.66	1.69	219.3	0.77	0.67
RAE [58]	800	839M	1.13	262.6	0.78	0.67	1.51	242.9	0.79	0.63
Latent-Level AR										
VAR [44]	350	2B	1.73	350.2	0.82	0.60	1.92	323.1	0.82	0.59
MAR [28]	800	943M	1.55	303.7	0.81	0.62	2.35	227.8	0.79	0.62
xAR [39]	800	1.1B	1.24	301.6	0.83	0.64	-	-	-	-
Latent-Level NF										
StarFlow [15]	320	1.4B	2.40	-	-	-	-	-	-	-
BiFlow [33]	160	133M	2.39	303.0	-	-	31.88	-	-	-
SimFlow [57]	160	1.4B	2.15	276.8	0.83	0.57	13.72	105.2	0.67	0.62
JSON	160	1.45B	2.10	306.2	0.82	0.58	11.68	118.4	0.68	0.64

(1) Our method achieves state-of-the-art performance for NF models under both pixel and latent scenario without additional prior knowledge. Specifically, our method achieves 2.10 FID and 306.2 IS scores, which outperforms previous best pixel-level NF [59] and latent-level NF model [57]. These results demonstrate the effectiveness of our method in improving NF generative capability without external supervision. (2) Our method is also comparable to other modern generative architectures. Specifically, JSON outperforms the representative diffusion model DiT [37] in terms of FID, and popular AR model MAR [28] in terms of IS, demonstrating its effectiveness in generating images with high fidelity.

4.2 Ablation Study

We summarize the main contributions of our JSON into "spatial-aware jigsaw head" and "jigsaw self-play". To study the effectiveness of each component, we

Table 2: Ablation Study. We conduct our ablation study at the resolution of 256×256. "N/A": training collapsed.

No.	Jigsaw Head	$\mathcal{L}_{\text{sp}}$	$\mathcal{L}_{\text{bsp}}$	$\mathcal{L}_{\text{jsp}}$	FID	IS	P	R
1	×	×	×	×	2.15	276.80	0.83	0.57
2	✓	×	×	×	2.37	284.27	0.79	0.58
3	✓	✓	×	×	N/A			
4	×	×	✓	×	2.46	311.64	0.81	0.58
5	✓	×	✓	×	2.19	291.85	0.82	0.56
6	✓	×	×	✓	2.10	306.20	0.82	0.58

conduct ablation study and report the results at Tab. 2. The included variants are: (1) "Baseline": a simple NF model without any proposed element, which is the same as SimFlow [57] and its results are shown at "No.1"; (2) "Jigsaw NF": integrating jigsaw head into SimFlow and its results are shown in "No.2"; (3) "B-NSP": NF with original self-play. We report its results at "No.3"; (4)"Bounded Self-play": the results of using bounded self-play without jigsaw head ("No.4") and with jigsaw head ("No.5"). **It should be noted $\mathcal{L}_{\text{bsp}}$ itself does not necessarily require jigsaw head**. We thus remove all jigsaw heads and conduct experiment "No.4" by using the same architecture, but with different loss ($\mathcal{L}_{\text{bsp}}$) with SimFlow; (5) "JSON": the results when using all proposed components ("No. 6").

Effectiveness of Introducing Spatial Coherence Constraint. By comparing "No. 1" and "No. 2" of Tab. 2, we conclude that introducing spatial constraints into NF training is beneficial to improving generative capability. Specifically, our jigsaw head is a stack of convolutional layers to keep NF features' original spatial cue. The architecture is proven to be valid in previous generative research [41] while supervising NF training with $\mathcal{L}_{\text{jig}}$ improves IS from 276.80 to 284.27. Although "No. 2" achieves worse FID score than "No. 1", the result is comparable and better than previous state-of-the-art BiFlow (FID=2.39). These results demonstrate the effectiveness of introducing additional spatial coherence constraint.

Effectiveness of Bounded Self-play. In Tab. 2:"No. 3"-"No.5", we compare the results of proposed bounded self-play with original self-play. As analyzed in previous sections, the original self-play poses unlimited penalization on synthesized data, creating a direct conflict with the NF's foundational MLE objective. The contradictory causes training collapse of NF model and leads to NAN losses. Distinct from B-NSP, our bounded self-play ensures stable training and yields consistent improvements in IS, regardless of whether the jigsaw head is integrated. Moreover, $\mathcal{L}_{\text{bsp}}$ achieves comparable FID scores with SimFlow when jigsaw head is involved, demonstrating the necessity of using jigsaw head when joint optimizing model with our $\mathcal{L}_{\text{bsp}}$.

Effectiveness of JSON. At "No.6" of Tab. 2, we report the results of using $\mathcal{L}_{\text{jsp}}$, which mines valuable trajectories of synthesized data for further training. The NF generative capabilities are further improved, demonstrated by both better FID and IS scores than all previous variants. The results prove the necessity of each proposed component.

Table 3: Architectural Design of Jigsaw Head.

Method	FID	IS
MLP	2.28	294.03
Conv Proj.	2.10	306.20

Table 4: Sensitivity to Patch Size p.

p	FID	IS
4	2.47	290.54
2	2.18	301.38
1	2.10	306.20

Table 5: Training Speed (ms / batch) Comparison.

Method	#Params	Speed
SimFlow	1.4 B	580.6
JigsawNF	1.45 B	614.9
Ours	1.45 B	1263.2

Architectural Design of Jigsaw Head. The default architecture of our jigsaw head is the stacked convolutional projection layers to maintain the spatial information within NF features. We also explore other architectural design of jigsaw head at Tab. 3. The included architectures are: (1) "MLP": the commonly used 3-layer MLP projection to map NF features for jigsaw puzzle reassembly. As pooling is used before MLP mapping, the spatial information within NF features are largely diluted, which may lead to potential performance degradation. (2) "Conv Proj.": default setting in our experiments, which adopt stacked convolutional layers to maintain the spatial information before jigsaw puzzle reassembly. As demonstrated in Tab. 3, the "Conv Proj." variant achieves best results, indicating the necessity of keeping spatial information within NF features for subsequent jigsaw self-play. The finding also align with recent research about representation alignment [41]. Different from their work, this work mainly adopt "Conv Proj." for better spatial information preservation.

Training Efficiency. We also compare the training speed with "SimFlow" and our "Jigsaw NF" in Tab. 5. We conclude that introducing additional jigsaw head does not significantly slow down the training speed. However, after integrating the training with self-play, the overall training speed is reduced due to the on-the-fly inference to obtain synthesized data x'. However, we constrain the self-play training at the final 5 training epochs, partially alleviating the relatively slow training speed problem and strike a balance between generative capability and training efficiency.

4.3 Sensitivity Analysis

Sensitivity to Patch Size. We also explore the sensitivity of our method to different patch sizes p. For input images with size of 256×256, the encoded latent size at spatial dimension is 16×16. We thus change p from 1 to 4 with step size of 2 and report the experimental results at Tab. 4. We conclude that smaller patch size leads to better generative capability of NF models. As jigsaw heads post additional puzzle reassembly task on NF features, using small p enables fine-level constraint for NF to learn feature spatial coherence, thus improves generative capability.

Sensitivity to CFG. We change CFG value from 0.5 to 2 and visualize FID of "Jigsaw NF" and our method in Fig. 4. The visualization demonstrates our method achieves optimal FID when setting CFG value to 1.1. Moreover, our jigsaw self-play outperforms "Jigsaw NF" with most CFG values, showing its effectiveness in improving NF generative capability.

Fig. 3: Qualitative evaluation of our jigsaw self-play. (a) We visualize the change of generated samples among all self-play training epochs for two given classes ("Norwegian Elkhound" and "Beaver"); (b) Visualization of synthesized data that achieves lower jigsaw losses than their real-world counterparts; (c) More visualized results.

4.4 Visualization

Qualitative Evaluation. We choose "Norwegian Elkhound" and "Beaver" as the given class condition and visualize generated images at different training epochs at Fig. 3-(a) to illustrate the qualitative impact of our JSON. As shown in the visualization, we note improvement on image quality with our jigsaw self-play. The qualitative assessment underscores our method's effectiveness in improving NF generative capability without using external supervision. More visualized results are shown in Fig. 3-(c).

Visualization of Samples with lower $\mathcal{L}_{\text{jig}}$. In our jigsaw self-play, $\mathcal{L}_{\text{jig}}$ plays an important role in assigning optimal training objective. We thus visualize some synthesized images that achieves lower $\mathcal{L}_{\text{jig}}$ values than their real-world counterparts to qualitatively assess the effectiveness of jigsaw head. The paired samples are shown in Fig. 3-(b). The first row are synthesized samples that achieves lower $\mathcal{L}_{\text{jig}}$ than real data at the second row. We conclude that synthesized

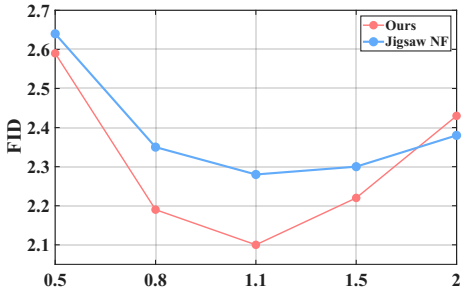


Fig. 4: Sensitivity to CFG.

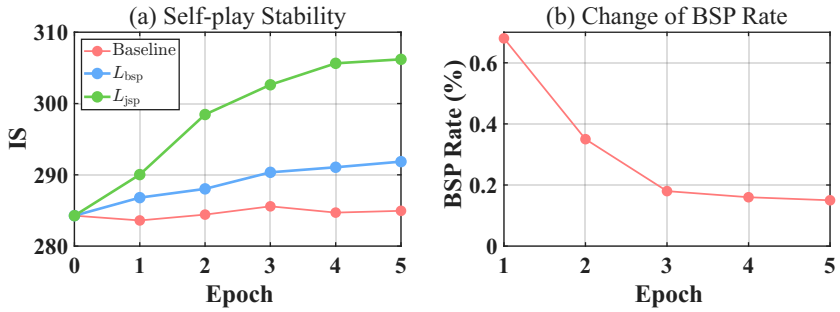


Fig. 5: Further analysis for our jigsaw self-play. (a) Jigsaw self-play's stability, "Baseline": model with jigsaw heads and vanilla NF optimization. "$\mathcal{L}_{\text{bsp}}$": NF with our bounded self-play. "$\mathcal{L}_{\text{jsp}}$": NF with our jigsaw self-play. (b) Change of BSP Rate.

samples with lower $\mathcal{L}_{\text{jig}}$ have the following traits. (1) they usually contain full semantic information for the given condition. For example, as shown in the first column, synthesized image achieves lower $\mathcal{L}_{\text{jig}}$ than its real-world companion as the synthesized images presents a complete bottle. This enables NF model to learn the intact semantic information of "Beer Bottle" without being negatively affected by cropping; (2) they may present better image quality than real data. As shown at the second column, synthesized data present better image quality than real data for the given condition "Pembroke Welsh Corgi". To sum up, jigsaw head enables useful information mining from synthesized data, which further enhances the generative capability of NF model.

Jigsaw Self-play's Stability. Our jigsaw self-play relies on alternative training between "bounded self-play loss" and "no fine-tuning" based on inherent spatial evaluation (Eq. 8). We thus show the change of IS over 5 epochs of jigsaw self-play training and compare the results with "bounded self-play only" and "jigsaw head only" baseline. The results are visualized in Fig. 5-(a). We conclude that our jigsaw self-play, achieves consistent improvement and higher IS scores over the 5 epochs training, demonstrating the effectiveness of our alternative self-play strategy.

Further Experiments on BSP Rate during Jigsaw Self-play. We also analyze the percentage of using bounded self-play part in Eq. 8, denoted as "BSP rate", for the whole 5 self-play training epochs (*i.e.*, last 5 epochs). BSP rate roughly reflects the contribution of synthesized data in improving final generative capability. Higher BSP rate indicates more fine-tuning steps with synthesized data. As shown in Fig. 5-(b), synthesized data plays an important role during the initial stages of self-play fine-tuning, demonstrating the necessity of integrating synthesized data into NF training for better generative capability. At the end of jigsaw self-play, BSP rate has reduced and remains stagnant, indicating the convergence of our jigsaw self-play. Our model finally obtains better generative capability than the baseline with the help of synthesized data.

Further Discussion on Defining Wining Trajectories. There are alternative methods to define wining and losing trajectories. For example, preferring trajectory with high likelihood or using MLP-probing. Previous methods [23, 49]

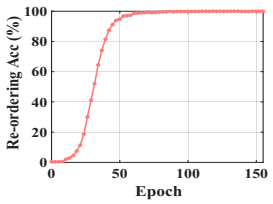
demonstrates the deficiency of using likelihood to define wining trajectory. Moreover, training accurate MLP for MLP-probing requires binary labels (win/lose) that are often noisy or heuristically defined. Compared to these solutions, jigsaw task provides deterministic and verifiable ground truth, ensuring accurate supervision for better generative capabilities.

Jigsaw Accuracies. We visualize the change of reordering accuracies at Fig. 6. As demonstrated in the figure, the jigsaw reordering accuracy increases with training going on, which confirms the compatibility of Jigsaw objective with the standard MLE.

Table 6: Results of different self-play strategies.

Method	FID	IS
SPIN [7]	Collapsed	
SPACE [49]	2.25	286.2
Ours	2.10	306.2

Fig. 6: Reordering acc.



Comparison with Different Self-play Strategies. We also note alternative solution in recent studies. For example, SPACE [49] stabilizes self-play by separately optimizes real and synthesized responses. We conduct experiments by using their strategy and report the results in Tab. 6. We note little improvement from SPACE. We speculate that introducing additional task like jigsaw puzzle benefits wining sample mining and outperforms concurrent self-play strategies.

5 Conclusion

In this paper, we devise a self-play paradigm to improve NF generative capability without auxiliary fine-tuning data. The main challenges of self-play in NFs are twofold, encompassing unstable training due to the contradictory of original self-play objective, and lack of pre-defined data trajectory for wining and losing trajectory mining. To alleviate these problems, we propose jigsaw self-play optimization for NFs (JSON), which integrates jigsaw puzzle reassembly task to serve as an intrinsic spatial coherence evaluator, identifying "winning" trajectories that exhibit superior spatial logic for self-play fine-tuning. Based on this contribution, we further propose a bounded self-play loss and a nested-loop optimization strategy to ensure stable fine-tuning. Extensive experiments on ImageNet-1K benchmark demonstrate our method's effectiveness.

Acknowledgment. We acknowledge the support from Shanghai Municipal Commission of Economy and Informatization, under Grant No. 2024-GZL-RGZN-01008.

References

1. Black, K., Janner, M., Du, Y., Kostrikov, I., Levine, S.: Training diffusion models with reinforcement learning. In: International Conference on Learning Representation (2023)
2. Bradley, R.A., Terry, M.E.: Rank analysis of incomplete block designs: I. the method of paired comparisons. Biometrika **39**(3/4), 324–345 (1952)
3. Carlucci, F.M., D'Innocente, A., Bucci, S., Caputo, B., Tommasi, T.: Domain generalization by solving jigsaw puzzles. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 2229–2238 (2019)
4. Chen, R.T., Rubanova, Y., Bettencourt, J., Duvenaud, D.K.: Neural ordinary differential equations. In: Advances in neural information processing systems (2018)
5. Chen, S., Ge, C., Zhang, S., Sun, P., Luo, P.: Pixelflow: Pixel-space generative models with flow. arXiv preprint arXiv:2504.07963 (2025)
6. Chen, Z., Zhu, J., Chen, X., Zhang, J., Hu, X., Zhao, H., Wang, C., Yang, J., Tai, Y.: Dip: Taming diffusion models in pixel space. arXiv preprint arXiv:2511.18822 (2025)
7. Chen, Z., Deng, Y., Yuan, H., Ji, K., Gu, Q.: Self-play fine-tuning converts weak language models to strong language models. In: International Conference on Machine Learning (2024)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)
9. Dhariwal, P., Nichol, A.: Diffusion models beat gans on image synthesis. In: Advances in neural information processing systems (2021)
10. Dinh, L., Krueger, D., Bengio, Y.: Nice: Non-linear independent components estimation. arXiv preprint arXiv:1410.8516 (2014)
11. Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using real nvp. In: Proceedings of International Conference in Learning Representation (2017)
12. Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al.: The llama 3 herd of models. arXiv e-prints pp. arXiv–2407 (2024)
13. Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., et al.: Scaling rectified flow transformers for high-resolution image synthesis. In: Proceedings of International Conference on Machine Learning (2024)
14. Fan, Y., Watkins, O., Du, Y., Liu, H., Ryu, M., Boutilier, C., Abbeel, P., Ghavamzadeh, M., Lee, K., Lee, K.: Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. In: Advances in Neural Information Processing Systems (2023)
15. Gu, J., Chen, T., Berthelot, D., Zheng, H., Wang, Y., Zhang, R., Dinh, L., Bautista, M.A., Susskind, J., Zhai, S.: Starflow: Scaling latent normalizing flows for high-resolution image synthesis. In: Advances in Neural Information Processing Systems (2025)
16. Gu, J., Shen, Y., Chen, T., Dinh, L., Wang, Y., Bautista, M.A., Berthelot, D., Susskind, J., Zhai, S.: Starflow-v: End-to-end video generative modeling with normalizing flow. arXiv preprint arXiv:2511.20462 (2025)
17. Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al.: Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948 (2025)

18. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: NeurIPS. pp. 6626–6637 (2017)
19. Ho, J., Chen, X., Srinivas, A., Duan, Y., Abbeel, P.: Flow++: Improving flow-based generative models with variational dequantization and architecture design. In: International conference on machine learning. pp. 2722–2730. PMLR (2019)
20. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Advances in Neural Information Processing Systems (2020)
21. Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., et al.: Openai o1 system card. arXiv preprint arXiv:2412.16720 (2024)
22. Jiang, D., Guo, Z., Zhang, R., Zong, Z., Li, H., Zhuo, L., Yan, S., Heng, P.A., Li, H.: T2i-r1: Reinforcing image generation with collaborative semantic-level and token-level cot. In: Advances in neural information processing systems (2025)
23. Kamb, M., Ganguli, S.: An analytic theory of creativity in convolutional diffusion models. In: International Conference on Machine Learning (2024)
24. Kingma, D.P., Dhariwal, P.: Glow: Generative flow with invertible 1×1 convolutions. In: Advances in neural information processing systems (2018)
25. Kirichenko, P., Izmailov, P., Wilson, A.G.: Why normalizing flows fail to detect out-of-distribution data. In: Advances in neural information processing systems (2020)
26. Kynkäänniemi, T., Karras, T., Laine, S., Lehtinen, J., Aila, T.: Improved precision and recall metric for assessing generative models. In: NeurIPS. pp. 3929–3938 (2019)
27. Leng, X., Singh, J., Hou, Y., Xing, Z., Xie, S., Zheng, L.: Repa-e: Unlocking vae for end-to-end tuning with latent diffusion transformers. In: Proceedings of Intenational Conference on Computer Vision (2025)
28. Li, T., Tian, Y., Li, H., Deng, M., He, K.: Autoregressive image generation without vector quantization. Advances in Neural Information Processing Systems (2024)
29. Lipman, Y., Chen, R.T., Ben-Hamu, H., Nickel, M., Le, M.: Flow matching for generative modeling. In: Proceedings of International Conference in Learning Representation (2023)
30. Liu, J., Liu, G., Liang, J., Li, Y., Liu, J., Wang, X., Wan, P., Zhang, D., Ouyang, W.: Flow-grpo: Training flow matching models via online reinforcement learning. arXiv preprint arXiv:2505.05470 (2025)
31. Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Jiang, Q., Li, C., Yang, J., Su, H., et al.: Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In: Proceedings of European Conference in Computer Vision (2024)
32. Liu, X., Gong, C., Liu, Q.: Flow straight and fast: Learning to generate and transfer data with rectified flow. In: Proceedings of International Conference in Learning Representation (2023)
33. Lu, Y., Sun, Q., Wang, X., Jiang, Z., Zhao, H., He, K.: Bidirectional normalizing flow: From data to noise and back. arXiv preprint arXiv:2512.10953 (2025)
34. Ma, N., Goldstein, M., Albergo, M.S., Boffi, N.M., Vanden-Eijnden, E., Xie, S.: Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In: European Conference on Computer Vision. pp. 23–40 (2024)
35. Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: European conference on computer vision. pp. 69–84 (2016)
36. Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al.: Dinov2: Learning robust visual features without supervision. arXiv preprint arXiv:2304.07193 (2023)

37. Peebles, W., Xie, S.: Scalable diffusion models with transformers. In: Proceedings of the International Conference on Computer Vision. pp. 4195–4205 (2023)
38. Rafailov, R., Sharma, A., Mitchell, E., Manning, C.D., Ermon, S., Finn, C.: Direct preference optimization: Your language model is secretly a reward model. In: Advances in neural information processing systems (2023)
39. Ren, S., Yu, Q., He, J., Shen, X., Yuille, A., Chen, L.C.: Beyond next-token: Next-x prediction for autoregressive visual generation. arXiv preprint arXiv:2502.20388 (2025)
40. Rezende, D., Mohamed, S.: Variational inference with normalizing flows. In: International conference on machine learning. pp. 1530–1538. PMLR (2015)
41. Singh, J., Leng, X., Wu, Z., Zheng, L., Zhang, R., Shechtman, E., Xie, S.: What matters for representation alignment: Global information or spatial structure? In: Proceedings of International Conference on Learning Representation (2026)
42. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. In: Proceedings of International Conference in Learning Representation (2021)
43. Tesauro, G., et al.: Temporal difference learning and td-gammon. Communications of the ACM **38**(3), 58–68 (1995)
44. Tian, K., Jiang, Y., Yuan, Z., Peng, B., Wang, L.: Visual autoregressive modeling: scalable image generation via next-scale prediction. In: Advances in neural information processing systems (2024)
45. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al.: Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)
46. Tschannen, M., Pinto, A.S., Kolesnikov, A.: Jetformer: An autoregressive generative model of raw images and text. In: Proceedings of International Conference in Learning Representation (2024)
47. Wallace, B., Dang, M., Rafailov, R., Zhou, L., Lou, A., Purushwalkam, S., Ermon, S., Xiong, C., Joty, S., Naik, N.: Diffusion model alignment using direct preference optimization. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 8228–8238 (2024)
48. Wang, S., Gao, Z., Zhu, C., Huang, W., Wang, L.: Pixnerd: Pixel neural field diffusion. arXiv preprint arXiv:2507.23268 (2025)
49. Wang, Y., Chen, Q.G., Xu, Z., Luo, W., Zhang, K., Zhang, L.: Space: Noise contrastive estimation stabilizes self-play fine-tuning for large language models. In: Advances in Neural Information Processing Systems (2025)
50. Wang, Z., Zhu, J., Tang, B., Li, Z., Xiong, F., Yu, J., Blaschko, M.B.: Jigsaw-r1: A study of rule-based visual reinforcement learning with jigsaw puzzles. Transactions on Machine Learning Research (2025)
51. Xu, J., Liu, X., Wu, Y., Tong, Y., Li, Q., Ding, M., Tang, J., Dong, Y.: Imagereward: learning and evaluating human preferences for text-to-image generation. In: Proceedings of International Conference on Neural Information Processing Systems. pp. 15903–15935 (2023)
52. Xu, J., Liu, X., Wu, Y., Tong, Y., Li, Q., Ding, M., Tang, J., Dong, Y.: Imagereward: Learning and evaluating human preferences for text-to-image generation. In: Advances in Neural Information Processing Systems (2024)
53. Yu, J., Xu, Y., Koh, J.Y., Luong, T., Baid, G., Wang, Z., Vasudevan, V., Ku, A., Yang, Y., Ayan, B.K., et al.: Scaling autoregressive models for content-rich text-to-image generation. Transactions on Machine Learning Research (2022)

54. Yu, S., Kwak, S., Jang, H., Jeong, J., Huang, J., Shin, J., Xie, S.: Representation alignment for generation: Training diffusion transformers is easier than you think. In: International Conference on Learning Representations (2025)
55. Yuan, H., Chen, Z., Ji, K., Gu, Q.: Self-play fine-tuning of diffusion models for text-to-image generation. In: Advances in Neural Information Processing Systems (2024)
56. Zhai, S., Zhang, R., Nakkiran, P., Berthelot, D., Gu, J., Zheng, H., Chen, T., Bautista, M.A., Jaitly, N., Susskind, J.: Normalizing flows are capable generative models. In: Proceedings of the International Conference on Machine Learning (2024)
57. Zhao, Q., Zheng, G., Yang, T., Zhu, R., Leng, X., Gould, S., Zheng, L.: Simflow: Simplified and end-to-end training of latent normalizing flows. arXiv preprint arXiv:2512.04084 (2025)
58. Zheng, B., Ma, N., Tong, S., Xie, S.: Diffusion transformers with representation autoencoders. arXiv preprint arXiv:2510.11690 (2025)
59. Zheng, G., Zhao, Q., Yang, T., Xiao, F., Lin, Z., Wu, J., Deng, J., Zhang, Y., Zhu, R.: Farmer: Flow autoregressive transformer over pixels. arXiv preprint arXiv:2510.23588 (2025)
60. Zheng, H., Nie, W., Vahdat, A., Anandkumar, A.: Fast training of diffusion models with masked transformers. Transactions on Machine Learning Research (2024)