Study Notes: DiffusionNFT (Online Diffusion Reinforcement with Forward Process)

Qiang Liu

Reference. These notes summarize the algorithmic structure of [1].

1 Problem Setup (high-level)

We have a pre-trained diffusion/flow model that induces a sampling policy (a black-box sampler / ODE solver) producing samples $x \sim \pi_\theta$ (e.g., images). We are also given a reward function $r(x)$ (possibly black-box) and want to adapt the sampler/model to increase expected reward while retaining the base model's prior knowledge and sample quality.

2 Key Idea of DiffusionNFT

DiffusionNFT proposes an *online* reinforcement-learning loop for diffusion models that performs policy optimization *directly on the forward diffusion process*. This yields three practical properties emphasized by the authors:

- **Solver-agnostic**: data collection can use any black-box sampler/solver.
- **Forward-consistent & memory-efficient**: updates are done with clean images (and their forward-noised versions), without storing full reverse trajectories.
- **Compatible**: the update is built on the standard *flow-matching* training objective.

3 Algorithm (step-by-step)

3.1 Inputs and state

- A base (pretrained) diffusion/flow model parameterized by θ.
- A reward function $r(x)$.
- A sampler $\mathsf{Sample}(\theta)$ (e.g., an ODE solver) that returns i.i.d. samples x from the current policy induced by θ.
- A forward noising process $q_t(x_t \mid x)$ consistent with the training objective (e.g., the one used by flow matching / diffusion training).

3.2 Loop

Repeat for iterations $k = 0, 1, 2, \ldots$:

1. **Data collection (on-policy).** Sample a batch of clean images from the current policy:

$$x^{(i)} \sim \mathsf{Sample}(\theta_k), \qquad i = 1, \ldots, B,$$

   and compute rewards $r^{(i)} := r(x^{(i)})$.

2. **(Conceptual) positive/negative split.** Based on rewards, define higher-reward samples as "positive" and lower-reward samples as "negative" (this can be done by thresholding, ranking, or any monotone transform). Denote the resulting sign/weight information abstractly by a scalar weight $w^{(i)}$ derived from $r^{(i)}$.

3. **Forward process optimization (reward-weighted flow matching).** Train an updated policy v_θ on *noised versions* of the collected clean images:

   - sample a time $t \sim \mathrm{Unif}[0, 1]$ (or the schedule used in training);
   - sample $x_t^{(i)} \sim q_t(\cdot \mid x^{(i)})$ via the forward noising process;
   - compute the standard flow-matching target (oracle) vector field $u(x_t^{(i)}, t; x^{(i)})$ used by the base training recipe;
   - minimize a reward-weighted regression loss of the form

$$\min_\theta \sum_{i=1}^{B} w^{(i)}\, \mathbb{E}_t \Big[\big\| v_\theta(t, x_t^{(i)}) - u(x_t^{(i)}, t; x^{(i)}) \big\|^2 \Big],$$

     where the weights $w^{(i)}$ are chosen so that high-reward samples pull the model toward their corresponding flow-matching targets and low-reward samples are downweighted (or contribute with opposite sign, depending on the exact design).

4. **Update.** Set $\theta_{k+1} \leftarrow \theta$ (possibly with a trust-region / regularization / LoRA adapter, depending on implementation).

Notes.

- The critical implementation move is that the optimization happens on *forward-noised* versions of *clean* on-policy samples, avoiding the need to store reverse trajectories.
- The sampler used in Step 1 can be changed without changing the training loop, hence "solver-agnostic".

4 What to check when implementing

- **Reward normalization**: choose a stable mapping $r \mapsto w$ (clipping, temperature, rank transform) to avoid collapse.

- **Regularization**: keep updates close to the pretrained model (e.g., LoRA, KL/score regularization, or mixing with the original flow-matching loss).
- **Evaluation protocol**: measure both reward metrics and general image quality metrics to detect mode collapse.

References

[1] Zheng, K., Chen, H., Ye, H., Wang, H., Zhang, Q., Jiang, K., Su, H., Ermon, S., Zhu, J., and Liu, M.-Y. (2025). Diffusionnft: Online diffusion reinforcement with forward process. *arXiv preprint*.