Deriving DT Loss with Simple Example

1 Simple 2 Frame Setup

Let's derive DT loss with simple example:

We have a two frame video: X_1, X_2

Each frame has a noise level: $K \sim \text{Uniform}(0, K)$ where 0 is no noise, K is full.

So $X_1^{K_1}, X_2^{K_2}$ are the two frames with their noise levels K_1, K_2 respectively.

2 Diffusion Forcing Objective

We derive a more general view of diffusion forcing objective by deriving the ELBO for the 2 frame case.

Let us define our goal:

$$\arg\max_\theta \mathbb{E}_{X_1^0, X_2^0 \sim p_{\text{data}}} \left[\ln(p_\theta(X_1^0, X_2^0))\right] \tag{1}$$

We derive a surrogate objective via the ELBO:

$$\mathbb{E}_{X_1^0, X_2^0} \left[\ln(p_\theta(X_1^0, X_2^0))\right] \tag{2}$$

$$= \mathbb{E}_{X_1^0, X_2^0} \left[\ln \int p_\theta(X_1^0, X_2^0, X_1^{i:K}, X_2^{i:K}) dX_1^{i:K} dX_2^{i:K}\right] \tag{3}$$

Let's assume a Gaussian noising process:

$$q(X_i^k | X_i^0) \sim \alpha_K X_i^0 + \beta_K \epsilon, \quad \epsilon \sim \mathcal{N}(0, I) \tag{4}$$

$$\sim \mathcal{N}(\alpha_K X_i^0, \beta_K I) \tag{5}$$

$$= \mathbb{E}_{X_1^0, X_2^0} \left[\ln \int p(X_1^{0:K}, X_2^{0:K}) \cdot \frac{q(X_1^{1:K}, X_2^{1:K} | X_1^0, X_2^0)}{q(X_1^{1:K}, X_2^{1:K} | X_1^0, X_2^0)} dX_1^{1:K} dX_2^{1:K}\right] \tag{6}$$

$$= \mathbb{E}_{X_1^0, X_2^0 \sim p_{\text{data}}} \left[\ln \mathbb{E}_{X_1^{1:K}, X_2^{1:K} \sim q(\cdot | X_1^0, X_2^0)} \left[\frac{p(X_1^{0:K}, X_2^{0:K})}{q(X_1^{1:K}, X_2^{1:K} | X_1^0, X_2^0)}\right]\right] \tag{7}$$

$$\geq \mathbb{E}_{X_1^0, X_2^0 \sim p_{\text{data}}, X_1^{1:K}, X_2^{1:K} \sim q(\cdot | X_1^0, X_2^0)} \left[\ln \left[\frac{p(X_1^{0:K}, X_2^{0:K})}{q(X_1^{1:K}, X_2^{1:K} | X_1^0, X_2^0)}\right]\right] \tag{8}$$

$$p(X_1^{0:K}, X_2^{0:K}) = p(X_1^K, X_2^K) \prod_{i \in \tau} p(T_{i-1} | T_i), \quad \tau \sim \text{Traj}(2, K) \tag{9}$$

where T_i is the ith state in a trajectory, but always: $T_0 = (X_1^0, X_2^0)$, $T_K = (X_1^K, X_2^K)$.

$\text{Traj}(2, K)$ is the set of all trajectories on a 2-dimensional cube with K delimiters. From 0 vertex to the opposite vertex. Note that we can construct the joint distribution in any order according to the denoising trajectories we allow, and we can assign any probability distribution over these trajectories. This flexibility allows us to model different generative processes by choosing different trajectory distributions. In the general case beyond the 2-frame example, we would replace 2 with n to get $\text{Traj}(n, K)$, representing trajectories on an n-dimensional cube.

$$= \mathbb{E}_{X_1^{0:K}, X_2^{0:K}} \left[\ln p(X_1^K, X_2^K) + \sum_{T_i \in \text{Traj}(2,K)} \ln \left(\frac{p(T_{i-1}|T_i)}{q(T_i|T_{i-1})} \right) \right] \tag{10}$$

$$= \mathbb{E}_{X_1^{0:K}, X_2^{0:K}} \left[\ln p(X_1^K, X_2^K) \right] + \sum_{T_i \in \text{Traj}(2,K)} \mathbb{E}_{T_{i-1}, T_i} \left[\ln \left(\frac{p(T_{i-1}|T_i) \cdot q(T_{i-1})}{q(T_{i-1}|T_i) \cdot q(T_i)} \right) \right] \tag{11}$$

We take all constants w.r.t p out:

$$\mathbb{E}_{X_1^K, X_2^K} \left[\ln p(X_1^K, X_2^K) \right] + \sum_{T_i \in \text{Traj}(2,K)} \mathbb{E}_{T_{i-1}, T_i} \left[\ln \left(\frac{p(T_{i-1}|T_i)}{q(T_{i-1}|T_i)} \right) \right] \tag{12}$$

$$= \mathbb{E}_{X_1^K, X_2^K} \left[\ln p(X_1^K, X_2^K) \right] - \sum_{T_i \in \text{Traj}(2,K)} D_{KL}(q(T_{i-1}|T_i) || p(T_{i-1}|T_i)) \tag{13}$$

Note that $q(X_1^K, X_2^K) = p(X_1^K, X_2^K) = \mathcal{N}(0, I)$, and our model parameters typically do not change $p(X_1^K, X_2^K)$, so we can discard the first term. This gives us a simple sum of KL divergences over trajectories sampled from our trajectory distribution:

$$\mathcal{L} = \sum_{\tau \sim \text{Traj}(2,K)} \sum_{i \in \tau} D_{KL}(q(T_{i-1}|T_i) || p(T_{i-1}|T_i)) \tag{14}$$

The probabilities assigned to each trajectory should correspond to the importance of each trajectory when sampling from the model. Recall that each trajectory can be decomposed as a sequence of transitions from T_i to T_{i-1} (denoising steps). Our loss can also be written as an expectation over trajectory states:

$$\mathcal{L} = \mathbb{E}_{\tau \sim \text{Traj}(2,K)} \left[\mathbb{E}_{T_i \in \tau} \left[D_{KL}(q(T_{i-1}|T_i) || p(T_{i-1}|T_i)) \right] \right] \tag{15}$$

Or equivalently as a weighted sum over all possible trajectories:

$$= \sum_{\tau \in \text{Traj}(2,K)} p(\tau) \sum_{T_i \in \tau} D_{KL}(q(T_{i-1}|T_i) || p(T_{i-1}|T_i)) \tag{16}$$

If we instead write this as probability distributions over atomic transitions marginalized over the distribution of trajectories, we get:

$$= \sum_{T_i \to T_{i-1} \in \text{Traj}(2,K)} P(T_i \to T_{i-1}) D_{KL}(q(T_{i-1}|T_i) || p(T_{i-1}|T_i)) \tag{17}$$

3 Generalizing to the n-frame case

To generalize to the n-frame case, we just change 2 to n.

$$\sum_{T_i \in \text{Traj}(n,K)} P(T_i \to T_{i-1}) D_{KL}(q(T_{i-1}|T_i)||p(T_{i-1}|T_i)) \tag{18}$$

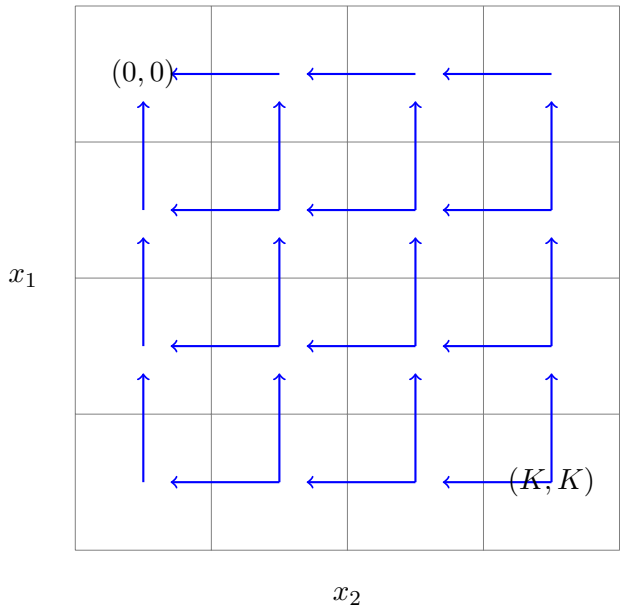
In DF paper it applies equal probability to each transition:

$$P(T_{i-1} \to T_i) = P(T_{j-1} \to T_j) \tag{19}$$

but this should not be the case if we never double back and assign equal probability to each trajectory (you expect transitions of states near the diagonal to be weighted more by the binomial Thm).

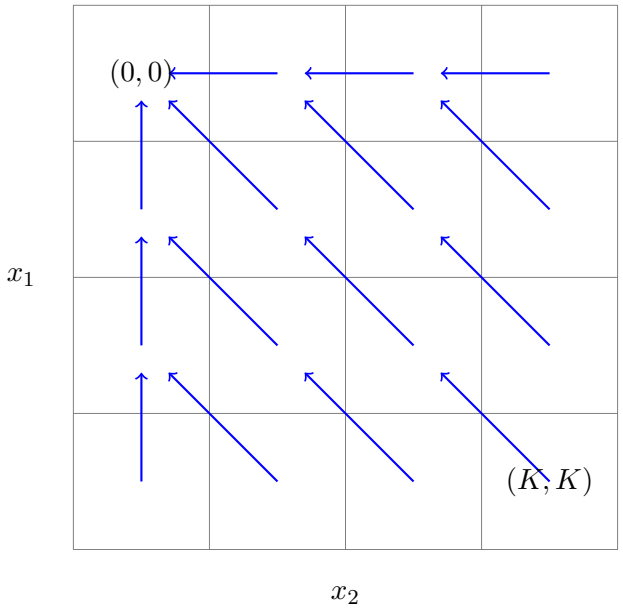
Practically, this should not matter too much but good to keep in mind that our prior over Trajectories is not uniform (but over transitions is).

Also, in an architecture like the Transformer, our state transitions decrease the noise level of each frame by 1, so the space of trajectories looks more like a diagonal pattern, but this proof still holds as we did not make any assumptions about the transitions allowed as long as they they are steadily denoising the frames.



RNN transitions:

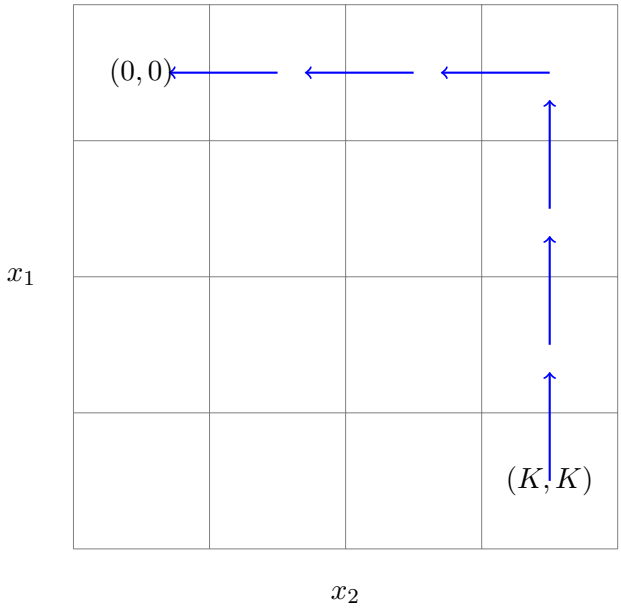
each step reduces noise by 1 in one frame.



Transformer transitions:

each step reduces noise by 1 in both frames.

This is for frames = 2 but for n frames we have an n-dim



Teacher Forcing transitions:

denoise one frame completely,

then move to the next frame.

TF is encapsulated in this framework too.

Like above, TF weights should be a constant traversing the edge of the cube from frame 1 $\rightarrow$ frame 2 $\rightarrow$ ... denoising which is:

$$\mathbb{I}\left(\left\{X_1^0, X_2^0, \ldots, X_i^{K_i}, X_{i+1}^K, \ldots, X_n^K\right\} \rightarrow \left\{X_1^0, X_2^0, \ldots, X_i^{K_i-1}, X_{i+1}^K, \ldots, X_n^K\right\}\right) \tag{20}$$

i.e. pick any frame i, every frame to the left should be clean, every frame to the right should be full noise, and the transition should be denoising the ith frame by 1 noise level.

4 Deriving the Self-Forcing Objective and Training Algorithm

Similarly, we can derive a more general view of self forcing with DMD by deriving the ELBO for the 2 frame case.

Let's derive the self-forcing objective and training algorithm from the following:
Instead of $\arg\max_\theta \mathbb{E}_{X\sim p_{\text{data}}}[\ln(p_\theta(X))]$,
we swap p_{data} and p_θ to get:

$$\arg\max_\theta \mathbb{E}_{X_1^0, X_2^0 \sim p_\theta}\left[\ln(p_{\text{data}}(X_1^0, X_2^0))\right] \tag{21}$$

$$= \arg\max_\theta \mathbb{E}_{X_1^0, X_2^0 \sim p_\theta}\left[\ln \int p_{\text{data}}(X_1^{0:K}, X_2^{0:K}) \cdot \frac{p_\theta(X_1^{1:K}, X_2^{1:K} | X_1^0, X_2^0)}{p_\theta(X_1^{1:K}, X_2^{1:K} | X_1^0, X_2^0)} dX_1^{1:K} dX_2^{1:K}\right] \tag{22}$$

$$= \arg\max_\theta \mathbb{E}_{X_1^0, X_2^0 \sim p_\theta,}\left[\ln \mathbb{E}_{X_1^{1:K}, X_2^{1:K} \sim p_\theta(\cdot | X_1^0, X_2^0)}\left[\frac{p_{\text{data}}(X_1^{0:K}, X_2^{0:K})}{p_\theta(X_1^{1:K}, X_2^{1:K} | X_1^0, X_2^0)}\right]\right] \tag{23}$$

$$\geq \arg\max_\theta \mathbb{E}_{X_1^{0:K}, X_2^{0:K} \sim p_\theta}\left[\ln\left(\frac{p_{\text{data}}(X_1^{0:K}, X_2^{0:K})}{p_\theta(X_1^{1:K}, X_2^{1:K} | X_1^0, X_2^0)}\right)\right] \tag{24}$$

From the inequality above we proceed by factorizing both the model and data joints along allowed denoising trajectories $\tau \in \text{Traj}(2, K)$:

$$p_\theta(X_1^{0:K}, X_2^{0:K}) = p_\theta(X_1^K, X_2^K) \prod_{T_i \in \tau} p_\theta(T_{i-1} \mid T_i), \tag{25}$$

$$p_{\text{data}}(X_1^{0:K}, X_2^{0:K}) = p_{\text{data}}(X_1^K, X_2^K) \prod_{T_i \in \tau} p_{\text{data}}(T_{i-1} \mid T_i), \tag{26}$$

where $T_0 = (X_1^0, X_2^0)$ and $T_K = (X_1^K, X_2^K)$.

Plugging these factorizations into the ELBO form and collecting the terms that depend on θ gives

$$\arg\max_\theta \ \mathbb{E}_{X_1^{0:K}, X_2^{0:K} \sim p_\theta}\left[\ln p_\theta(X_1^0, X_2^0) + \sum_{T_i \to T_{i-1} \in \text{Traj}(2,K)} \ln\left(\frac{p_{\text{data}}(T_{i-1} \mid T_i)}{p_\theta(T_{i-1} \mid T_i)}\right)\right]. \tag{27}$$

Separating expectations yields

$$= \underbrace{\mathbb{E}_{X_1^0, X_2^0 \sim p_\theta}[\ln p_\theta(X_1^0, X_2^0)]}_{-H(p_\theta(X_1^0, X_2^0))} - \sum_{T_i \to T_{i-1} \in \text{Traj}(2,K)} D_{\text{KL}}\big(p_\theta(T_{i-1} \mid T_i) \,\big\|\, p_{\text{data}}(T_{i-1} \mid T_i)\big) \tag{28}$$

$$+ \underbrace{\ln \frac{p_{\text{data}}(X_1^K, X_2^K)}{p_\theta(X_1^K, X_2^K)}}_{\text{independent of } \theta \text{ when } X^K \sim \mathcal{N}(0,I) \text{ for data and model}}. \tag{29}$$

Forward terminal state. We assume a Gaussian forward noising kernel so that the terminal distribution is fixed to standard normal:

$$p_\theta(X_1^K, X_2^K) = p_{\text{data}}(X_1^K, X_2^K) = \mathcal{N}(0, I). \tag{30}$$

Hence the terminal-state KL is zero and does not affect optimization.

Equivalent minimization problem. Maximizing the display above is equivalent to minimizing the positive terms:

$$\arg\min_{\theta} \; H(p_\theta) + \sum_{T_i \to T_{i-1} \in \mathrm{Traj}(2,K)} D_{\mathrm{KL}}\big(p_\theta(T_{i-1} \mid T_i) \,\big\|\, p_{\mathrm{data}}(T_{i-1} \mid T_i)\big)\,. \tag{31}$$

Averaging over trajectories drawn from any prior over paths $\tau \sim \mathrm{Traj}(2, K)$ gives the equivalent expectation form:

$$\arg\min_{\theta} \; H(p_\theta) + \mathbb{E}_{\tau \sim \mathrm{Traj}(2,K)} \left[\sum_{T_i \to T_{i-1} \in \tau} D_{\mathrm{KL}}\big(p_\theta(T_{i-1} \mid T_i) \,\big\|\, p_{\mathrm{data}}(T_{i-1} \mid T_i)\big) \right]. \tag{32}$$

Connection to DMD. Define the *distribution matching distillation* objective

$$\mathcal{L}_{\mathrm{DMD}} := \mathbb{E}_{\tau \sim \mathrm{Traj}(2,K)} \left[\sum_{T_i \to T_{i-1} \in \tau} D_{\mathrm{KL}}\big(p_\theta(T_{i-1} \mid T_i) \,\big\|\, p_{\mathrm{data}}(T_{i-1} \mid T_i)\big) \right].$$

Then (31) becomes simply

$$\arg\min_{\theta} \; H(p_\theta) + \mathcal{L}_{\mathrm{DMD}}. \tag{33}$$

Note. Trajectories are sampled from the model prior over paths; the variables along a path are distributed according to the model.

SF+DMD = Reverse KL objective The formulation of the DMD loss exactly mirrors the training method of self-forcing: at each training step, we randomly sample a transition state ($T_i \to T_{i-1}$) from a trajectory $\tau \sim \mathrm{Traj}(n, K)$, and if using the DMD loss, calculate the KL divergence $D_{\mathrm{KL}}\big(p_\theta(T_{i-1} \mid T_i) \,\big\|\, p_{\mathrm{data}}(T_{i-1} \mid T_i)\big)$ for that transition. This Monte Carlo sampling over transitions provides an unbiased estimate of the full trajectory loss.

DF vs. SF viewpoints

A convenient way to see Diffusion-Forcing (DF) and Self-Forcing (SF) is through the two KL directions:

$$\text{DF:} \quad \min_{\theta} \; D_{\mathrm{KL}}\big(p_{\mathrm{data}} \,\big\|\, p_\theta\big)\,, \qquad \text{SF:} \quad \min_{\theta} \; D_{\mathrm{KL}}\big(p_\theta \,\big\|\, p_{\mathrm{data}}\big)\,. \tag{34}$$

Using KL divergence decomposition

$$D_{\mathrm{KL}}\big(p_\theta \,\big\|\, p_{\mathrm{data}}\big) = \mathbb{E}_{x \sim p_\theta}[\ln p_{\mathrm{data}}(x)] - H(p_\theta)\,, \tag{35}$$

we see that SF encourages *entropy maximization* of p_θ while also minimizing the cross-entropy term $\mathbb{E}_{p_\theta}[\ln p_{\mathrm{data}}(x)]$. This partly explains the mode-seeking behavior of the reverse KL.

"Fixing" SF by removing entropy. If we wish to keep only the cross-entropy term and remove the entropy pressure in (35), we can optimize

$$\arg\max_{\theta} \; \mathbb{E}_{x \sim p_\theta}[\ln p_{\mathrm{data}}(x)] \quad \equiv \quad \arg\min_{\theta} \; D_{\mathrm{KL}}\big(p_\theta \,\big\|\, p_{\mathrm{data}}\big) - H(p_\theta)\,. \tag{36}$$

This isolates the data-fitting term without explicitly encouraging high model entropy. In practice, model entropy is intractable, so has to be estimated.