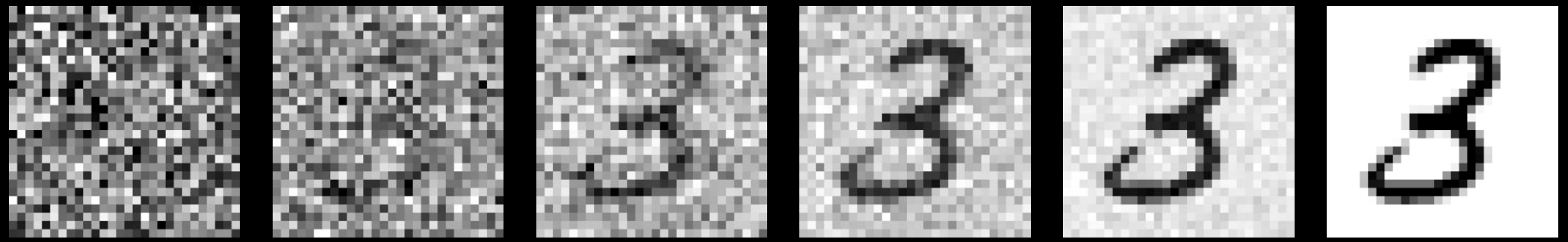
generation



corruption

In *flow matching*, generation is deterministic (from an initial noisy seed)

We'll now study *diffusion modelling*, in which generation is probabilistic

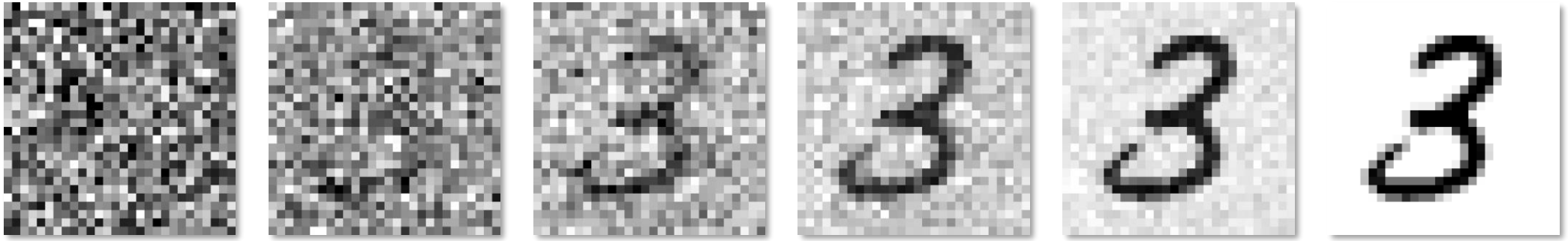
Why?

- it's a better fit for discrete data
- it's all just KL minimization

and anyway, this is a false dichotomy

Diffusion models are Markov generation processes, with a little noise at each step

$X_{0:1} = (X_0, X_\Delta, \dots, X_{m\Delta} = X_1)$ is a process of $m+1$ snapshots, $\Delta = 1/m$; X_0 is the initial noisy state and $X \coloneqq X_1$ the data.



Example: the Gaussian diffusion model. This model will let us do, in closed form, many of the calculations we'll need.

Generation process p^θ:

$$X_0 \underset{\sim N(0,I)}{\longrightarrow} \cdots \quad X_t \longrightarrow X_{t+\Delta} \underset{\sim N(\mu_\theta(X_t,t),\ s_t^2 I)}{\longrightarrow} \cdots \ X_1$$

Corruption process q:

$$X_0 \cdots \quad \underset{\sim N(a_t X_t,\ b_t^2 I)}{\longleftarrow} X_{t-\Delta} \longleftarrow X_t \cdots \quad \longleftarrow X_1 \underset{\sim q_X}{}$$

a_t, b_t are chosen so that $(X_t \mid X) \sim N(tX,\ (1-t)^2 I)$

Ho, Jain and Abbeel 2020, Denoising diffusion probabilistic models

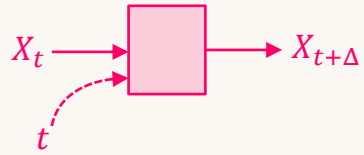
It's all just KL

The fitting objective. Our objective is, as usual, $\mathrm{KL}\left(q_X \parallel p_X^\theta\right)$. But this is a tough integral.
Instead, let's score how well the entire generation process fits, $\mathrm{KL}(q_{X_{0:1}} \parallel p_{X_{0:1}}^\theta)$.
This whole-process score bounds our objective, since

$$\mathrm{KL}(q_{X_{0:1}} \parallel p_{X_{0:1}}^\theta) = \mathrm{KL}(q_{X_1} \parallel p_{X_1}^\theta) + \mathbb{E}_{X_1 \sim q_{X_1}} \mathrm{KL}(q_{X_{<1}|X_1} \parallel p_{X_{<1}|X_1}^\theta) \;\geq\; \mathrm{KL}(q_X \parallel p_X^\theta).$$

This is just a collection of supervised learning problems:

$$\mathrm{KL}(q_{X_{0:1}} \parallel p_{X_{0:1}}^\theta) = \mathbb{E}_q \log \frac{q_X(X)\, q_{X_{<1}|X}(X_{<1})}{p_{X_0}(X_0) \prod_{t<1} p_{X_{t+\Delta}|X_t}^\theta (X_{t+\Delta})} = \text{const} - \sum_{t<1} \mathbb{E}_q \underbrace{\log p_{X_{t+\Delta}|X_t}^\theta (X_{t+\Delta})}_{\text{supervised autoregressive learning of the corrupted data process}}$$



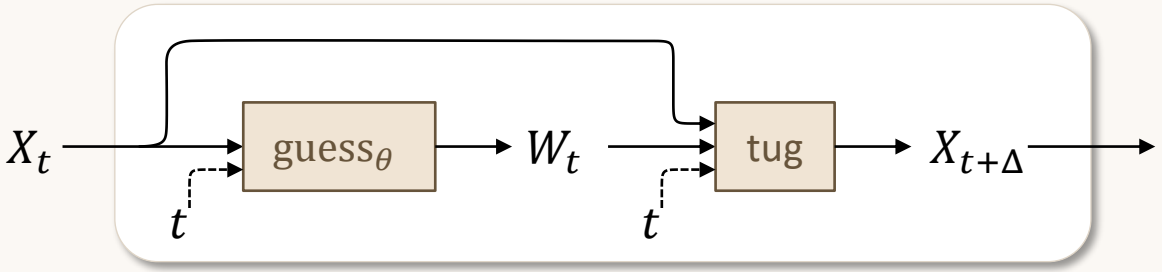
Equivalently, we want to match the corruption's step-by-step transitions:

$$\mathrm{KL}(q_{X_{0:1}} \parallel p_{X_{0:1}}^\theta) = \mathbb{E}_q \log \left(\underbrace{\text{const}'(X_{0:1})}_{\text{i.e. no } \theta} \times \frac{q_{X_0}(X_0) \prod_{t<1} q_{X_{t+\Delta}|X_t}(X_{t+\Delta})}{p_{X_0}(X_0) \prod_{t<1} p_{X_{t+\Delta}|X_t}^\theta (X_{t+\Delta})} \right) = \text{const}'' + \sum_{t<1} \mathbb{E}_{X_t \sim q} \mathrm{KL}\left(q_{X_{t+\Delta}|X_t} \parallel p_{X_{t+\Delta}|X_t}^\theta\right)$$

Training: minimize this bound over θ. We can estimate it using samples of the corruption process

Evaluation: simply compute this bound on holdout data

A slightly better generator. Generation has many steps, and errors accumulate.
Let's change to a generator-step mechanism that (hopefully) has an inductive bias to counter this, inspired by ResNet.



generate a latent variable W_t, trying to guess the real datapoint $(X_1|X_t)$

the trainable bit of the generator

generate $X_{t+\Delta} \sim q_{X_{t+\Delta}|X_t, X_1=W_t}$ (which can be calculated exactly for the Gaussian diffusion model)

tugs X_t a little bit towards the guessed destination W_t

Estimation. We can save a little bit of sampling when estimating the log likelihood:

$$\sum_{t<1} \mathbb{E}_{X_t, X_{t+\Delta} \sim q} \log p^\theta_{X_{t+\Delta}|X_t}(X_{t+\Delta}) = \sum_{t<1} \mathbb{E}_{X_1 \sim q} \left[\mathbb{E}_{X_t, X_{t+\Delta}|X_1 \sim q} \log p^\theta_{X_{t+\Delta}|X_t}(X_{t+\Delta}) \right] = \sum_{t<1} \mathbb{E}_{X_1, X_t \sim q} \left[\mathbb{E}_{X_{t+\Delta}|X_t, X_1 \sim q} \log p^\theta_{X_{t+\Delta}|X_t}(X_{t+\Delta}) \right]$$

A closed-form solution may be available. If so, use it! It reduces the variance of the estimator, if we don't have to sample.

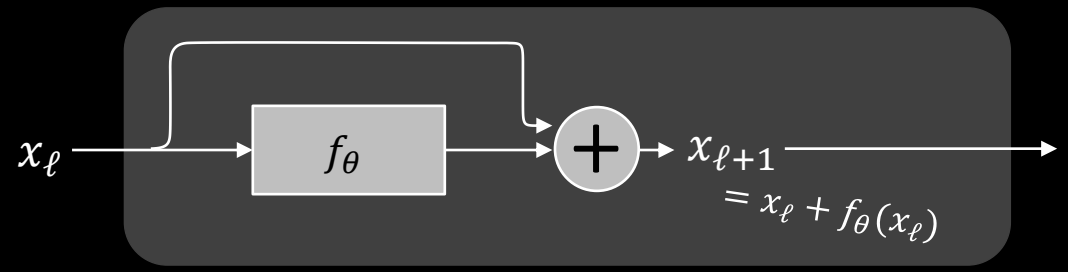
$$= m \, \mathbb{E}_T \, \mathbb{E}_{X_1, X_T \sim q}[\cdots] \quad \text{where } T \sim \text{Unif}\{0, \Delta, \ldots, 1-\Delta\}$$

The Gaussian diffusion model lets us draw (X_1, X_T) in a single shot. This saves us from crunching through the entire corruption process.

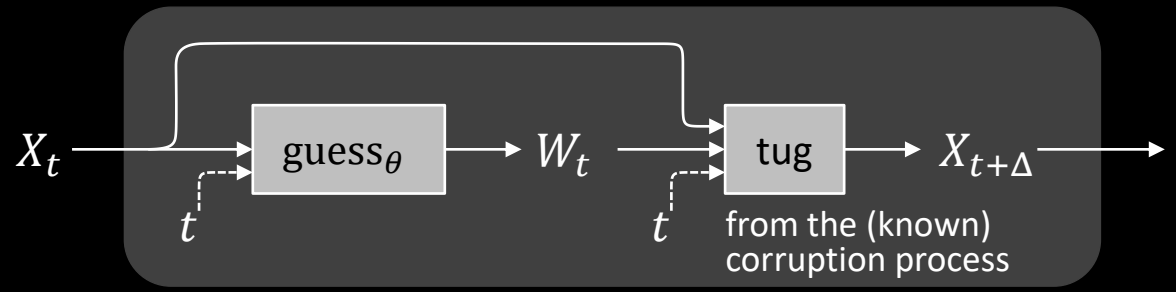
The KL equivalent of this is Ho et al. equation (5)

$$\text{KL}(q_X \parallel p^\theta_X) \leq \text{const}''' + \sum_{t<1-\Delta} \mathbb{E}_q \, \text{KL}\left(q_{X_{t+\Delta}|X_t, X_1} \parallel p^\theta_{X_{t+\Delta}|X_t}\right) - \mathbb{E}_q \log p^\theta_{X_1|X_{1-\Delta}}(X_1)$$

The ResNet trick



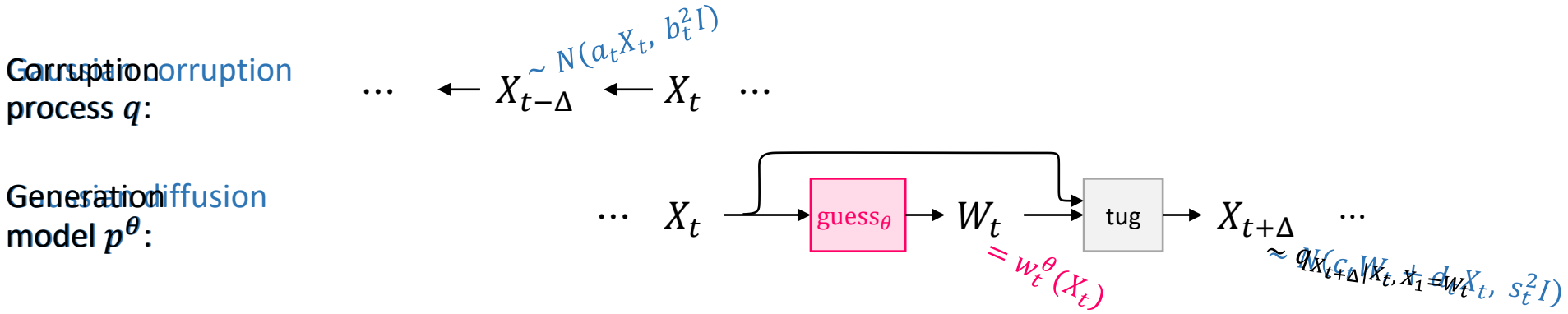
ResNet was the first successful network to stack hundreds of layers. It achieved this by using *residual blocks*, in which the network is asked to learn a difference against the 'structural default' $x_{\ell+1} \approx x_\ell$.



We're giving our diffusion model full freedom to make whatever guess it wants. We then wrap its guess up with the 'structural default' — i.e. the small step consistent with that guess.

If the network were fully expressive, and training perfect, we wouldn't need this ResNet trick.
But for finite-capacity networks, many of them stacked in sequence, it seems to be useful.

There's a lovely link between the Gaussian diffusion model and flow matching: they're both fitting to the same objective.



The KL bound for the Gaussian diffusion model has an explicit form:

$$\begin{aligned}
\mathrm{KL}(q_X \parallel p_X^\theta) &\le \text{const} + \sum_{t<1-\Delta} \mathbb{E}_q\ \mathrm{KL}\big(q_{X_{t+\Delta}|X_t,X} \,\|\, p^\theta_{X_{t+\Delta}|X_t}\big) + \text{last term} \\
&= \text{const} + \sum_{t<1-\Delta} \mathbb{E}_q\ \frac{\big\|(c_t X + d_t X_t) - \big(c_t\, w_t^\theta(X_t) + d_t X_t\big)\big\|^2}{2s_t^2} + \text{last term} \\
&= \text{const} + \sum_{t<1-\Delta} \frac{c_t^2}{2s_t^2}\ \mathbb{E}_q \big\|w_t^\theta(X_t) - X\big\|^2 + \text{last term} \\
&\approx \text{const} + m\,\mathbb{E}_T\ \frac{c_T^2}{2s_T^2}\ \mathbb{E}_{X,X_T} \big\|w_T^\theta(X_T) - X\big\|^2, \qquad T \sim \mathrm{Unif}\{0, \Delta, \ldots, 1-\Delta\}
\end{aligned}$$

Compare this to flow-matching, which seeks a velocity field $\tilde{v}^\theta$ to minimize the loss

$$\mathbb{E}_{X,Z,T} \big\|\tilde{v}^\theta(T, X_T) - \dot{X}_T\big\|^2 = \mathbb{E}_{X,Z,T} \left\|\tilde{v}^\theta(T, X_T) - \frac{X - X_T}{1-T}\right\|^2 = \mathbb{E}_T\ \frac{1}{(1-T)^2}\ \mathbb{E}_{X,X_T} \big\|\underbrace{X_T + (1-T)\,\tilde{v}^\theta(T, X_T)}_{\text{Our velocity field implies a guess about the final destination}} - X\big\|^2$$

It’s the modelling, stupid!

Flow matching and diffusion are SOTA (for images) — they both have the same training objective.

- This leaves a design space: deterministic versus stochastic generation steps
- And another design space: the relative weighting of the generator’s steps
- It’s good to run these generators in latent space (so they make the big-picture decisions)

(And the computation too.)

- For text, diffusion promises many tokens in one go, but quality and caching aren’t there yet
- For images, there is work on distilling the multi-step generation into a handful of steps