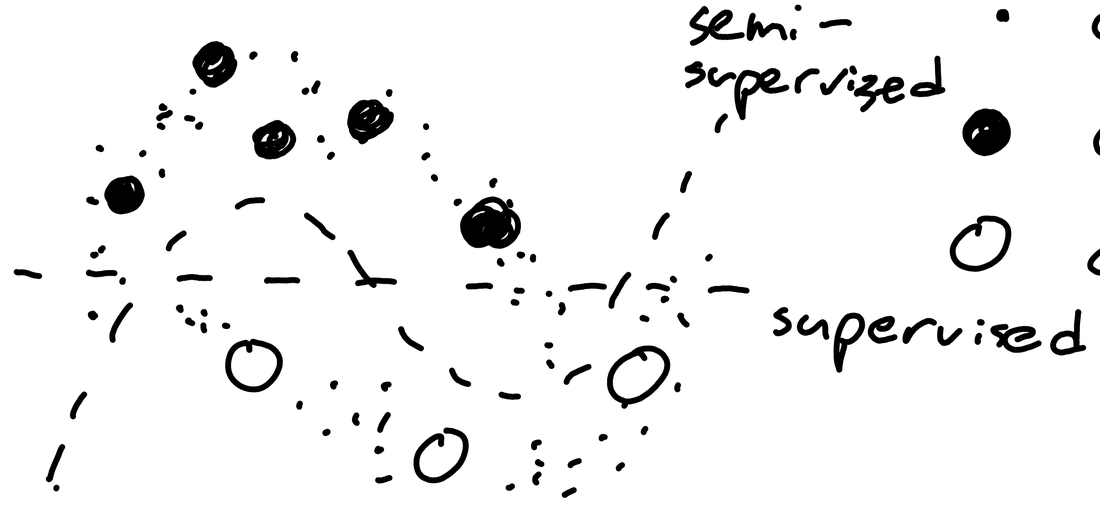
Semi-Supervised Learning

Supervised learning : $x, y \sim p(x, y)$

Semi-supervised : $x \sim p(x)$ also

unlabeled examples



Simplest approach: (self-training, pseudo-labeling)
Train the model to predict its predictions:

$$L = -\arg\max_y [p_\theta(y|x)] \log p_\theta(y|x)$$

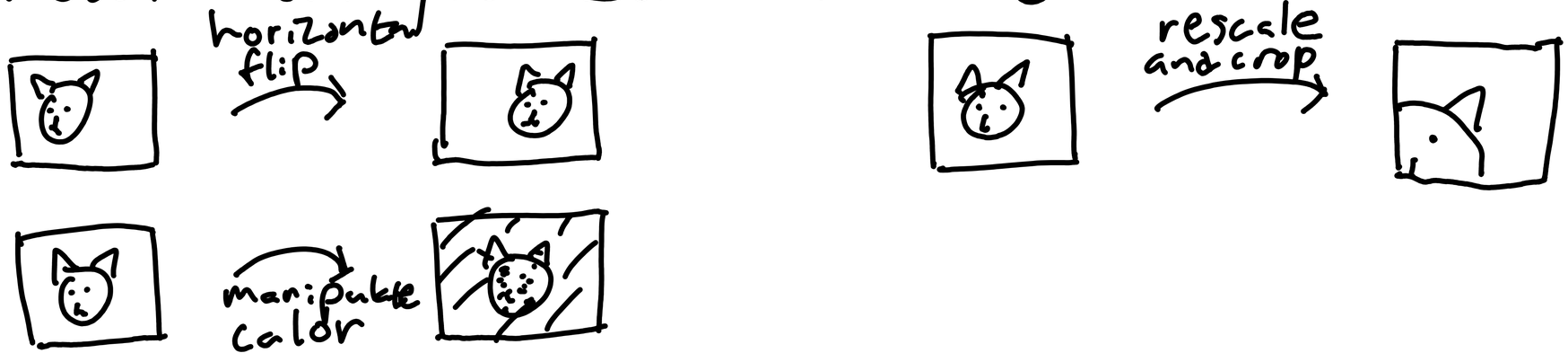
Problem: Model can reinforce (get more confident about) its own predictions.

Consistency regularization:
Model's output should be the same when the input is perturbed.

$$L = -\sum_y p_\theta(y|x) \log p_\theta(y|x')$$

(x: original x; x': perturbed x)

How to perturb x?
Simple: add a little noise.
Better: use data augmentation.
↳ modify x so that it's still "realistic" but the class is retained.
Allows us to bake in domain knowledge.



Self-supervised learning

We have $x \sim p(x)$ and can hand-design a function that gives a y for a given x.

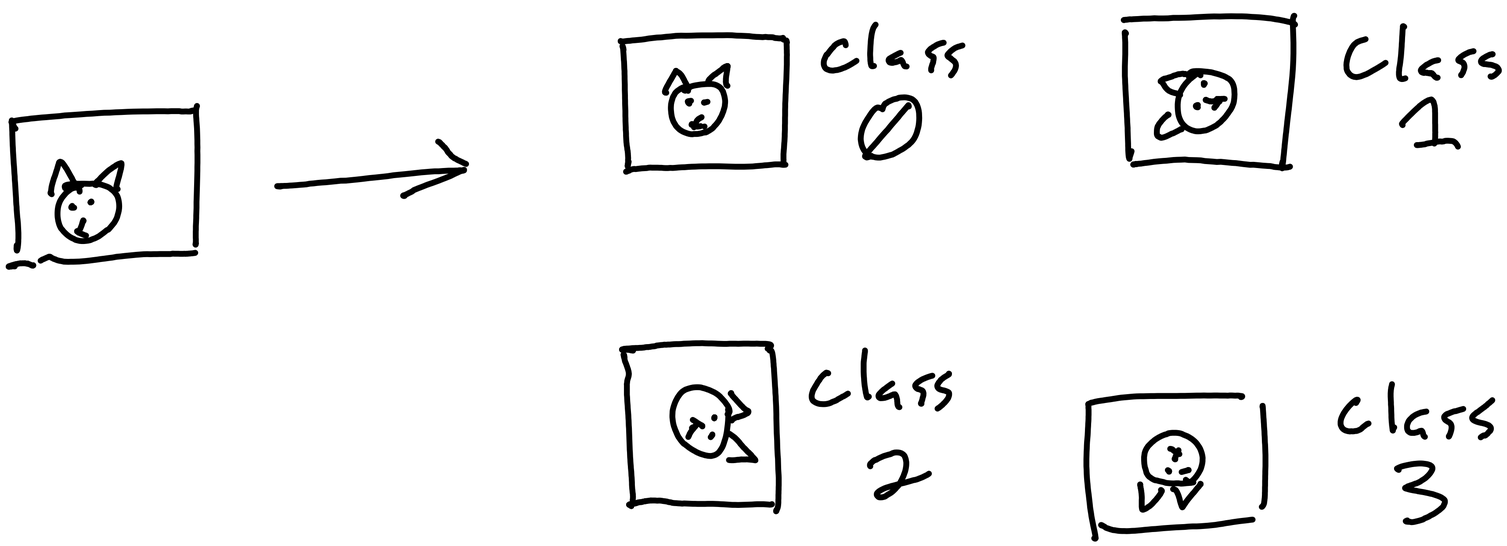
ex/ Masked language modeling

sentence → masked sentence → sentence

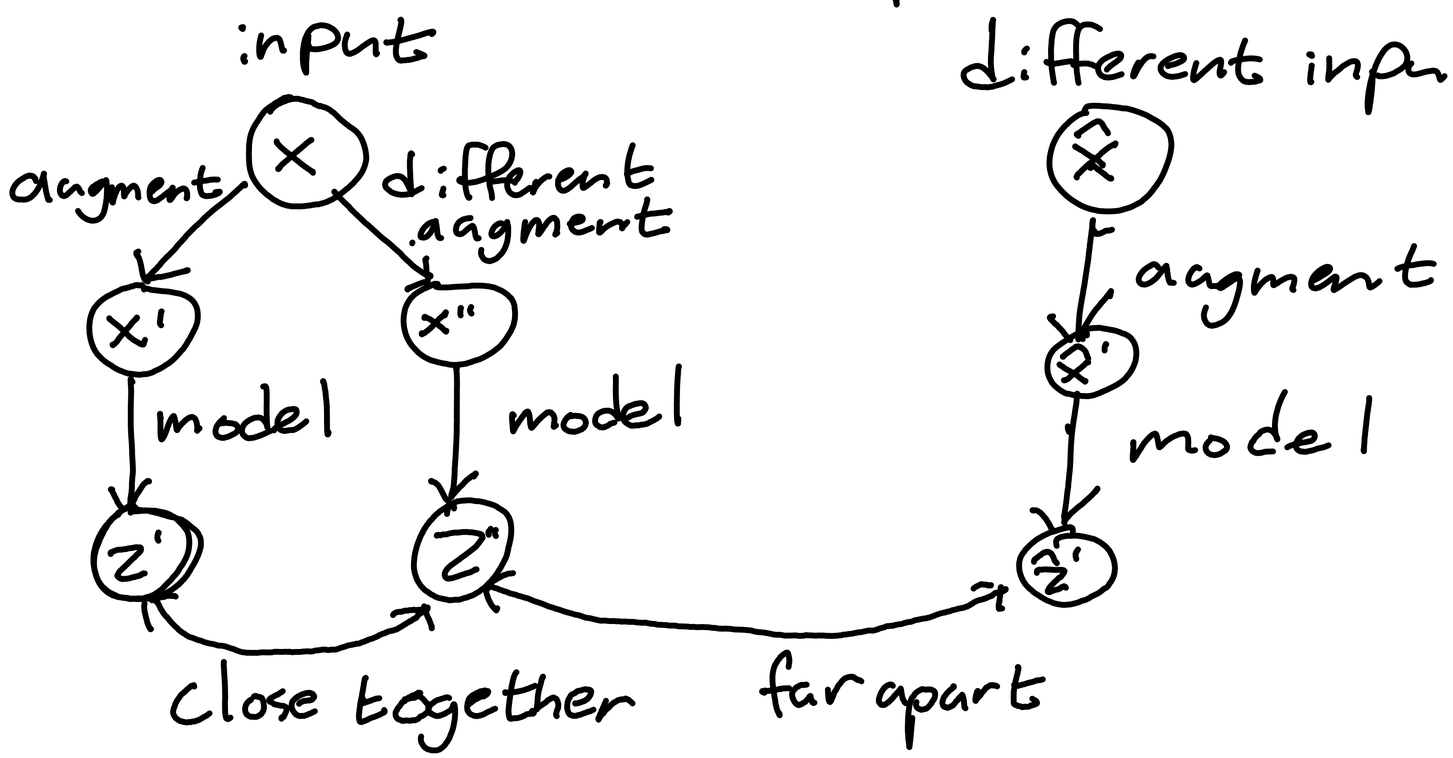
automatic

need no labelling

ex/ Predicting rotations



Contrastive self-supervised learning: (SimCLR)



$$L = -\log \frac{\exp(\mathrm{sim}(z', z'')/\tau)}{\sum_{b=1}^{B} \exp(\mathrm{sim}(z', \hat{z}_b')/\tau)}$$

↳ hyperparameter

↳ batch

↳ e.g. cosine distance

Generative Modeling

Generative model: Model $p(x)$ with $q_\theta(x)$ based on $x \sim p(x)$

Why? 1. If $q_\theta(x)$ approximates $p(x)$ well, we can sample $x \sim q_\theta(x)$ and x will be "realistic".

2. We can use $q_\theta(x)$ to approximate the probability of x.

3. Learning $q_\theta(x)$ can "help" model $p(y, x)$ (transfer learning).

Generative Adversarial Networks:

$$\min_{\theta} \max_{\phi} \mathop{\mathbb{E}}_{\substack{x \sim p(x) \\ \hat{x} \sim q_\theta(x)}} V(f_\phi(x), f_\phi(\hat{x}))$$

- V: loss function
- f_ϕ: neural network "discriminator"
- $q_\theta(x)$: neural net "generator"

"adversarial divergence"

Advantages:

- f_ϕ is a neural network, so we can give our divergence an inductive bias.
- Change V to change behavior.
- You can use anything you want for $q_\theta(x)$ as long as you can $\hat{x} \sim q_\theta(x)$ and it's differentiable.

Disadvantages:

- Train with alternating gradients, unstable.
- Can't actually evaluate $q_\theta(x)$

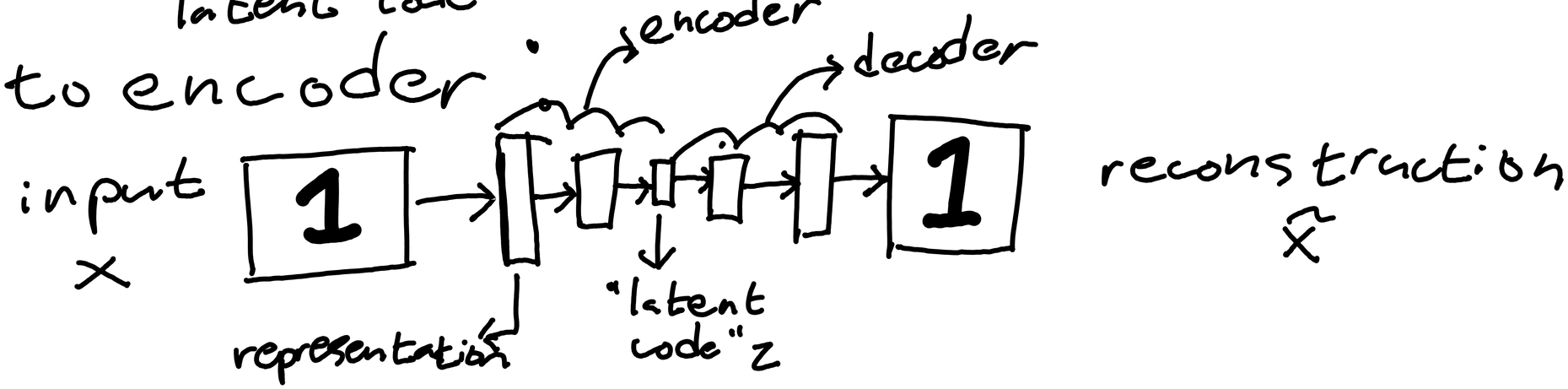
Variational Autoencoder

Generative process:

$$z \sim p(z) \qquad x \sim p(x|z)$$

prior on "latent code"

Autoencoder:



Train using a reconstruction loss

e.g. $\|x - \tilde{x}\|_2^2$

If $\dim(z) < \dim(x)$, we are (maybe) doing compression.

Problem: z is continuous → it can store infinite information.

VAE: Regularize the information in Z.

Objective = reconstruction + regularization

$$= \mathbb{E}_{z \sim q_\phi(z|x)} \log p_\theta(x|z) - KL(q_\phi(z|x) \,||\, p(z))$$

z is our latent code (stochastic).

$q_\phi(z|x)$ is "encoder", maps input → latent

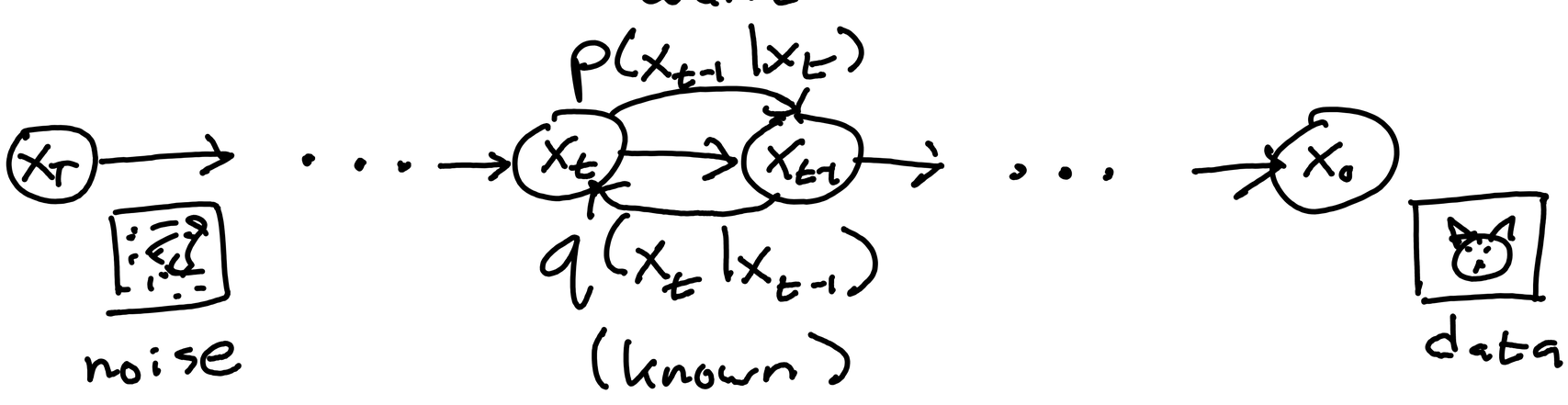
$p_\theta(x|z)$ is "decoder", maps latent → reconstruction

$p(z)$ is prior over latents.

KL term measures information in Z.

Diffusion models

Given $x \sim p(x)$, define a "forward" diffusion process that goes from x to noise.



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

↳ noise schedule ↲

β_t small

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

$$\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i, \quad \alpha_i = 1 - \beta_i$$

Learn a neural network $p_\theta(x_{t-1}|x_t)$

$$p(x_{t-1}|x_t) = \mathcal{N}(\mu_\theta(x_t, t), \beta_t I)$$

↳ optimal under certain assumptions

(If β_t is small enough, $q(x_{t-1}|x_t)$ will be Gaussian)

Now we just need to train μ_θ to predict

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\alpha_t}}\,\epsilon_t\right)$$

↳ noise applied at step t known during training

$$L_t = \mathop{\mathbb{E}}_{t\sim[1,T]} \|\epsilon_t - \epsilon_\theta(x_t, t)\|_2^2$$

$$= \mathop{\mathbb{E}}_{t\sim[1,T]} \|\epsilon_t - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\,x_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon_t, t)\|_2^2$$