Generative Models

Distribution learning

Training Data(CelebA)

Model Samples (Karras et.al., 2018)

4 years of progression on Faces

Brundage et al., 2017

Image credits to Andrej Risteski

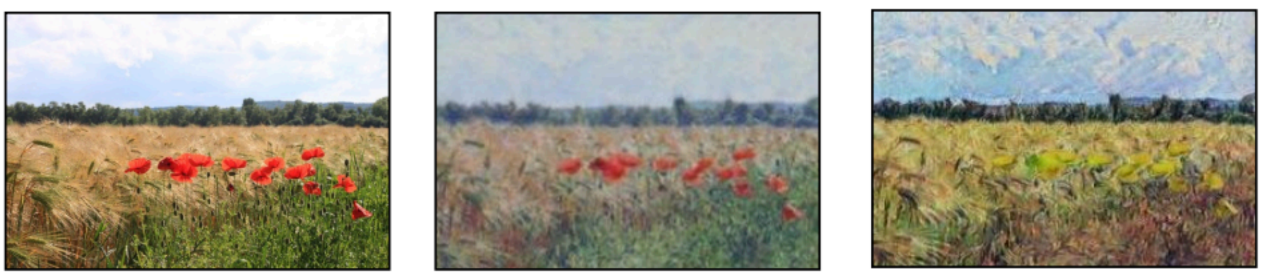
Distribution learning

BigGAN, Brock et al '18

Distribution learning

Conditional generative model P(zebra images | horse images)

Style Transfer



Input Image Monet Van Gogh

Image credits to Andrej Risteski

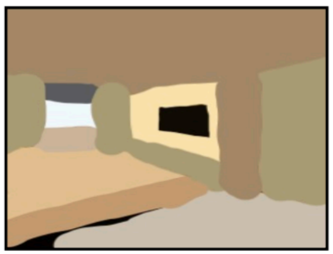
Distribution learning

Source actor

Target actor

Real-time reenactment

Generative model



Generative model
of realistic images

Generate

Stroke paintings to realistic images
[Meng, He, Song, et al., ICLR 2022]

"Ace of Pentacles"

Generative model
of paintings

Generate

Language-guided artwork creation
https://chainbreakers.kath.io @RiversHaveWings

Slides credit to Yang Song

Generative model

High probability

Generative model of traffic signs

Low probability

Outlier detection

[Song et al., ICLR 2018]

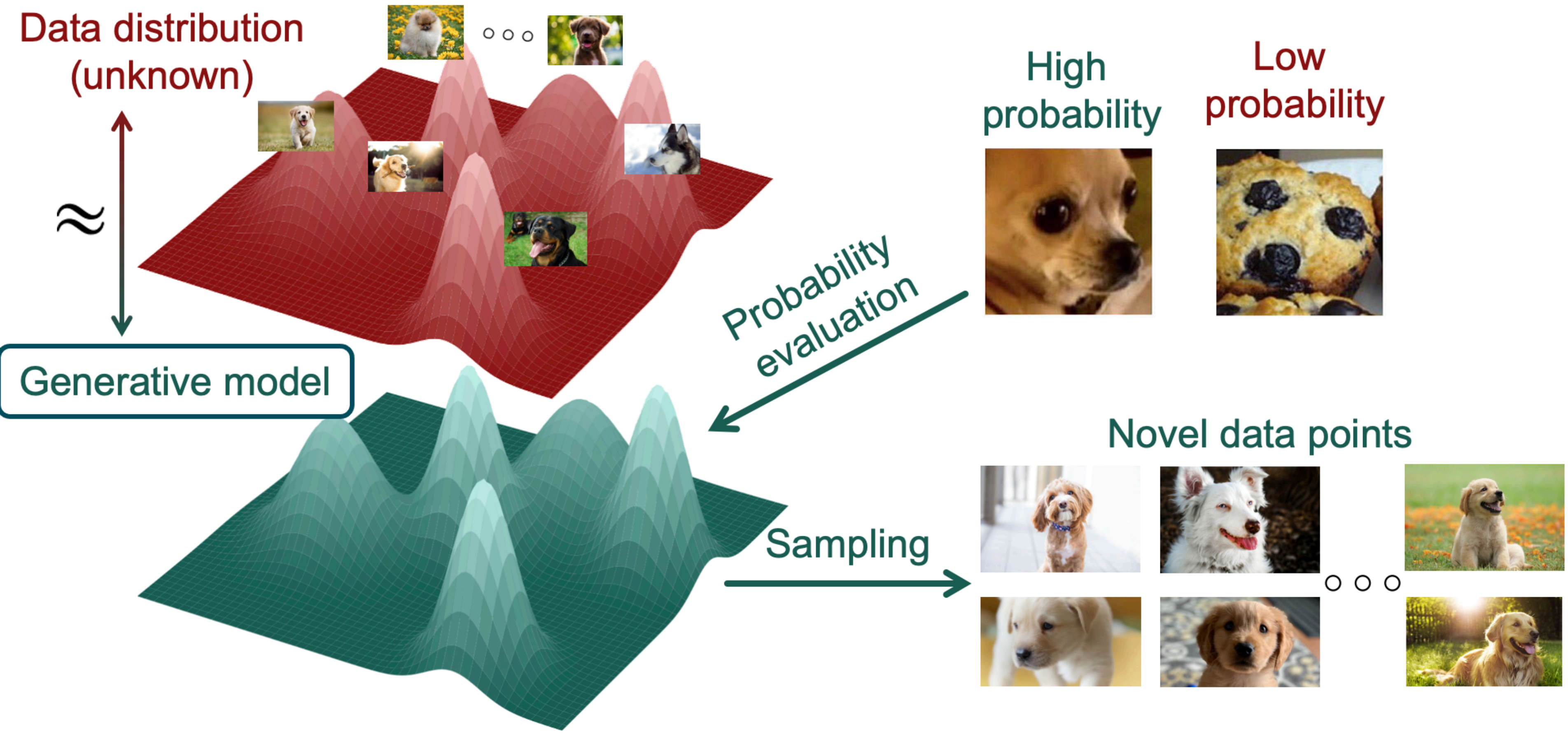
Desiderata for generative models

- **Probability evaluation**: given a sample, it is computationally efficient to evaluate the probability of this sample.

- **Flexible model family:** it is easy to incorporate any neural network models.

- **Easy sampling:** it is computationally efficient to sample a data from the probabilistic model.

Desiderata for generative models



Slide credit to Yang Song

Taxonomy of generative models

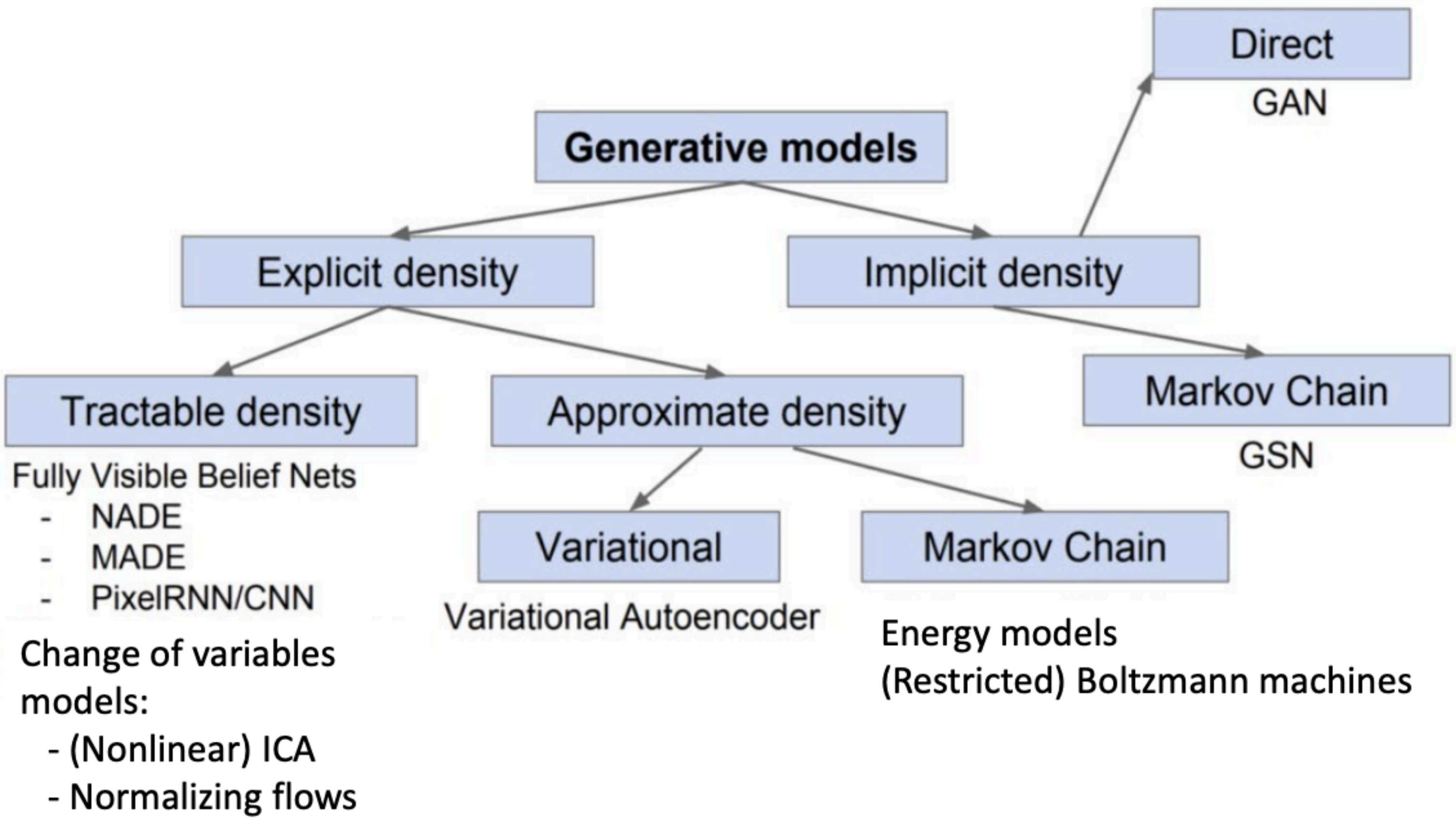
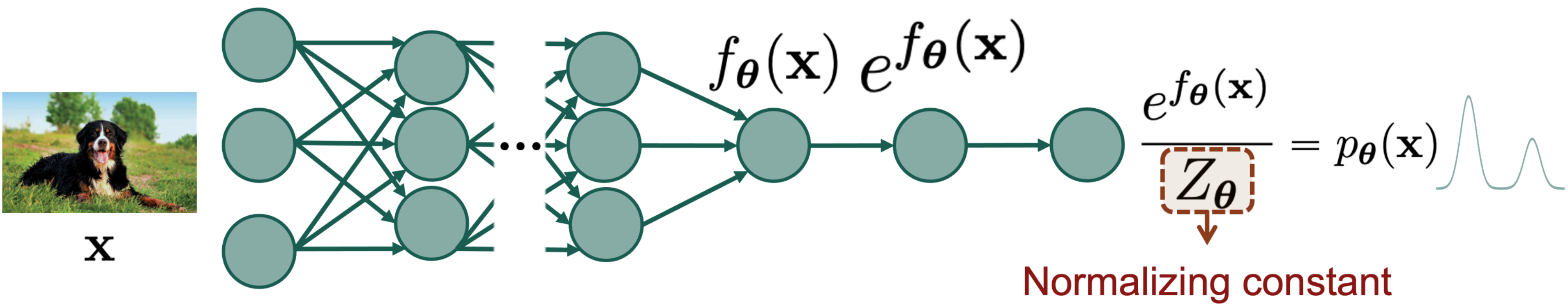


Image credits to Andrej Risteski

Key challenge for building generative models



Slide credit to Yang Song

Key challenge for building generative models

Approximating the normalizing constant

- Variational auto-encoders [Kingma & Welling 2014, Rezende et al. 2014]
- Energy-based models [Ackley et al. 1985, LeCun et al. 2006]

❌ Inaccurate probability evaluation

Using restricted neural network models

- Autoregressive models [Bengio & Bengio 2000, van den Oord et al. 2016]
- Normalizing flow models [Dinh et al. 2014, Rezende & Mohamed 2015]

❌ Restricted model family

Generative adversarial networks (GANs)

- Model the generation process, not the probability distribution [Goodfellow et al. 2014]

❌ Cannot evaluate probabilities

Slide credit to Yang Song

Training generative models

- **Likelihood-based:** maximize the likelihood of the data under the model (possibly using advanced techniques such as variational method or MCMC):

$$\max_{\theta} \sum_{i=1}^{n} \log p_{\theta}(x_i)$$

- Pros:
  - **Easy training**: can just maximize via SGD.
  - **Evaluation**: evaluating the fit of the model can be done by evaluating the likelihood (on test data).

- Cons:
  - **Large models needed**: likelihood objectve is hard, to fit well need very big model.
  - **Likelihood entourages averaging:** produced samples tend to be blurrier, as likelihood encourages "coverage" of training data.

Training generative models

- **Likelihood-free:** use a **surrogate loss** (e.g., GAN) to train a discriminator to differentiate real and generated samples.

- Pros:
  - **Better objective, smaller models needed:** objective itself is learned - can result in visually better images with smaller models.

- Cons:
  - **Unstable training:** typically min-max (saddle point) problems.
  - **Evaluation:** no way to evaluate the quality of fit.

Generative Adversarial Nets

Implicit Generative Model

- **Goal:** a sampler $g(\cdot)$ to generate images
- A simple generator $g(z;\theta)$:
  - $z \sim N(0,I)$
  - $x = g(z;\theta)$ deterministic transformation

- Likelihood-free training:
  - Given a dataset from some distribution p_{data}
  - Goal: $g(z;\theta)$ defines a distribution, we want this distribution $\approx p_{data}$
  - Training: minimize $D(g(z;\theta), p_{data})$
    - D is some distance metric (not likelihood)
  - Key idea: ***Learn* a differentiable** D

GAN (Goodfellow et al., '14)

- Parameterize the discriminator $D(\cdot\,;\phi)$ with parameter ϕ

- **Goal:** learn ϕ such that $D(x;\phi)$ measures how likely x is from p_{data}
  - $D(x,\phi) = 1$ if $x \sim p_{data}$
  - $D(x,\phi) = 0$ if $x! \sim p_{data}$
  - a.k.a., a binary classifier
- GAN: use a neural network for $D(\cdot\,;\phi)$

- **Training:** need both negative and positive samples
  - Positive samples: just the training data
  - Negative samples: use our sampler $g(\cdot\,;z)$ (can provide infinite samples).

- **Overall objectives:**
  - Generator: $\theta^* = \max_{\theta} D(g(z;\theta);\phi)$
  - Discriminator uses MLE Training:

    $$\phi^* = \max_{\phi} \mathbb{E}_{x\sim p_{data}}[\log D(x;\phi)] + \mathbb{E}_{\hat{x}\sim g(\cdot)}[\log(1 - D(\hat{x};\phi))]$$

GAN (Goodfellow et al., '14)

- Generator $G(z;\theta)$ where $z \sim N(0,I)$
  - Generate realistic data
- Discriminator $D(x;\phi)$
  - Classify whether the data is real (from p_{data}) or fake (from G)
- Objective function:

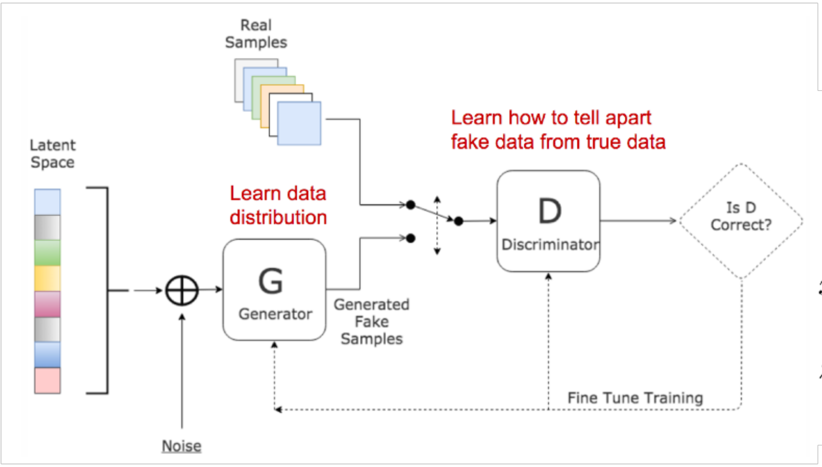
$$L(\theta,\phi) = \min_{\theta} \max_{\phi} \mathbb{E}_{x \sim p_{data}}\left[\log D(x;\phi)\right] + \mathbb{E}_{\hat{x} \sim G}\left[\log(1 - D(\hat{x};\phi))\right]$$

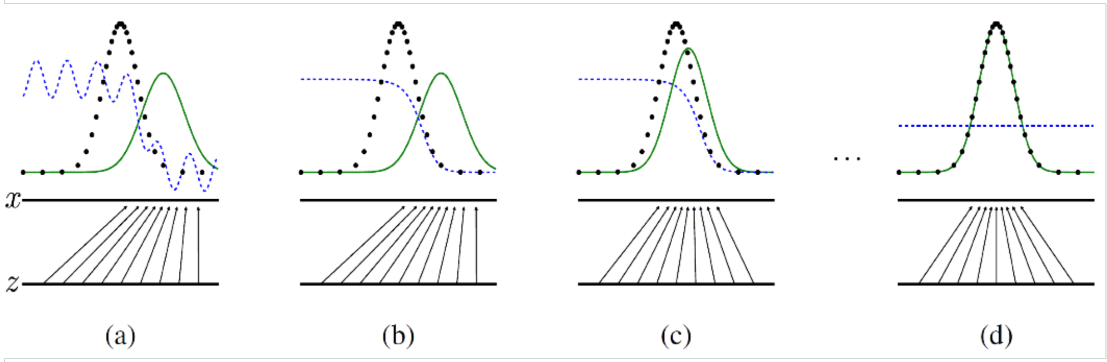
- Training procedure:
  - Collect dataset $\{(x,1) \mid x \sim p_{data}\} \cup \{(\hat{x},0) \sim g(z;\theta)\}$
  - Train discriminator
    $D : L(\phi) = \mathbb{E}_{x \sim p_{data}}\left[\log D(x;\phi)\right] + \mathbb{E}_{\hat{x} \sim G}\left[\log(1 - D(\hat{x};\phi))\right]$
  - Train generator $G : L(\theta) = \mathbb{E}_{z \sim N(0,I)}\left[\log D(G(z;\theta),\phi)\right]$
  - Repeat

GAN (Goodfellow et al., '14)

- Objective function:

$$L(\theta, \phi) = \min_{\theta} \max_{\phi} \mathbb{E}_{x \sim p_{data}} \left[\log D(x; \phi) \right] + \mathbb{E}_{\hat{x} \sim G} \left[\log(1 - D(\hat{x}; \phi)) \right]$$

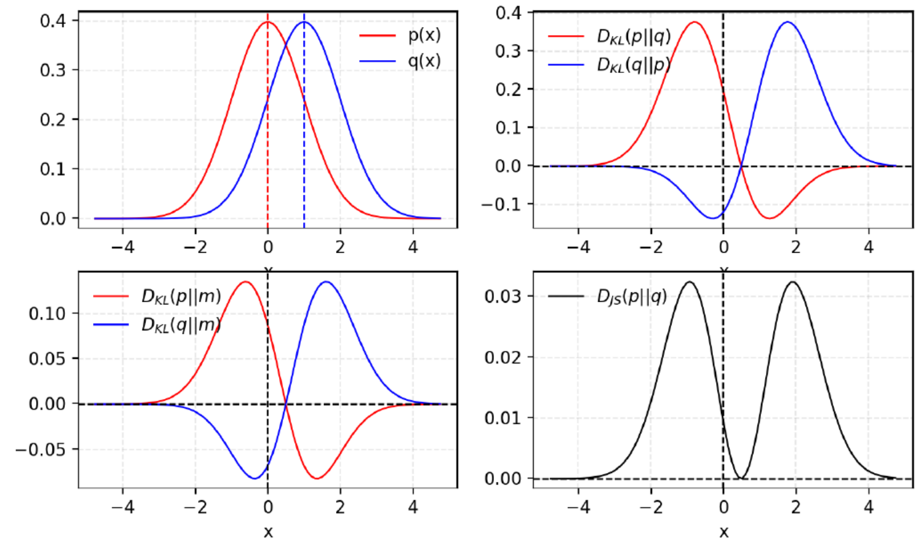




Math Behind GAN

Math Behind GAN

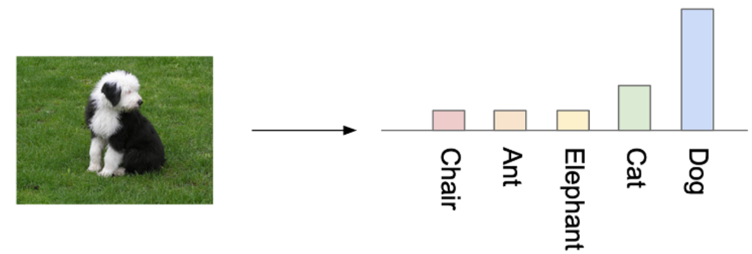
KL-Divergence and JS-Divergence

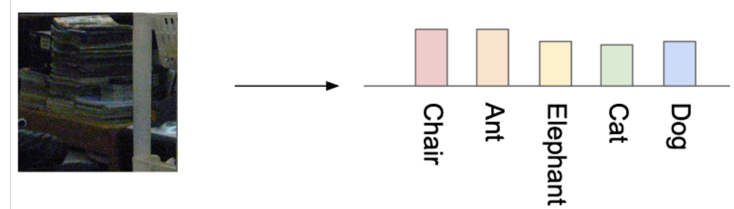


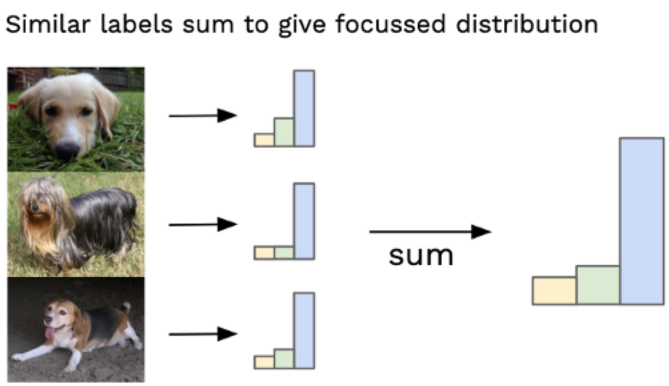
Math Behind GAN

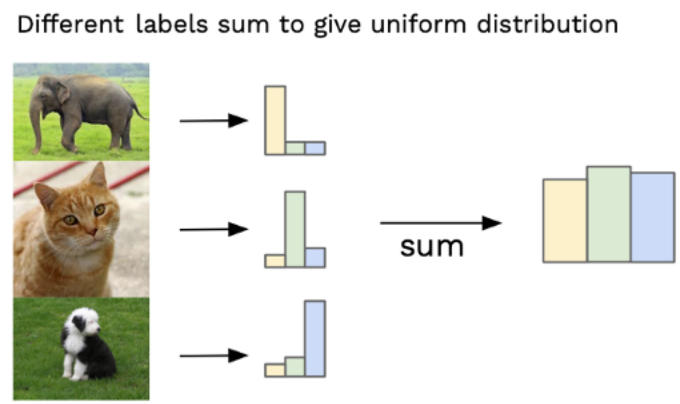
Evaluation of GAN

- No $p(x)$ in GAN.
- Idea: use a trained classifier $f(y \mid x)$:
- If $x \sim p_{data}$, $f(y|x)$ should have low entropy
  - Otherwise, $f(y \mid x)$ close to uniform.
- Samples from G should be diverse:
  - $p_f(y) = \mathbb{E}_{x \sim G}[f(y|x)]$ close to uniform.



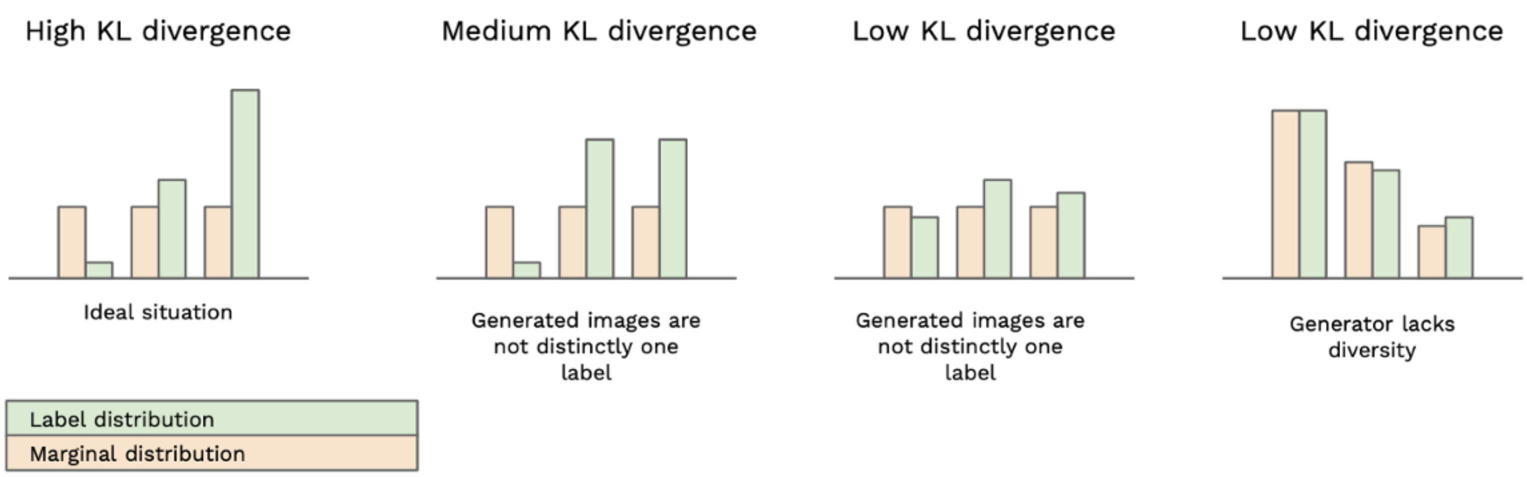






Evaluation of GAN

- **Inception Score** (IS, Salimans et al. ’16)
  - Use Inception V3 trained on ImageNet as $f(y\,|\,x)$
  - $IS = \exp\left(\mathbb{E}_{x\sim G}\left[KL(f(y\,|\,x)\,||\,p_f(y)))\right]\right)$
  - Higher the better



Comments on GAN

- Other evaluation metrics:
  - Fréchet Inception Distance (FID): Wasserstein distance between Gaussians

- Mode collapse:
  - The generator only generate a few type of samples.
  - Or keep oscillating over a few modes.

- Training instability:
  - Discriminator and generator may keep oscillating
  - Example: $-xy$, generator x, discriminator$y$. NE: $x = y = 0$ but GD oscillates.
  - No stopping criteria.
  - Use Wsserstein GAN (Arjovsky et al. '17):
    $$\min_{G} \max_{f:\text{Lip}(f)\leq 1} \mathbb{E}_{x\sim p_{data}}\left[f(x)\right] - \mathbb{E}_{\hat{x}\sim p_G}[f(\hat{x})]$$
  - And need many other tricks…