Generative Adversarial Networks

Gautam Kamath

Recapping Generative Modelling

- Given $X_1, \ldots, X_n \sim p$, generate more data from (a distribution close to) p
- But p may be complex…
- Solution: Use a NN to map samples from $N(0, I)$ to samples from p
- That is, if $z \sim N(0, I)$, then $x = T_\theta(z) \sim p$ (draw “Generator”)
- How to do this?
- VAE: Optimize mapping data distribution to $N(0, I)$ and then map samples from $N(0, I)$ back to data distribution
- GAN: Ensure samples from $T_\theta(z)$ are indistinguishable from real data

GAN Ideas

- Use another NN to classify real versus fake samples (“discriminator”)
- (Draw three networks:
  - Generator $T_\theta: z \sim N(0, I) \rightarrow \tilde{x}$ (“fake” sample)
  - Real data: box that outputs x
  - Discriminator $S_\phi: x$ or $\tilde{x} \rightarrow$ fake or real?)
- Goal: Distinguish between $T_\theta(z)$ (fake samples) versus D (real samples)
- How to formalize?
- First, a mathematical interlude…

Fenchel Conjugate

- Let $f(x) : \mathbf{R} \to \mathbf{R}$ be some function. The Fenchel conjugate of f is $f^*(x) = \max_y\big(xy - f(x)\big)$
- Example: $\boldsymbol{f(x) = x \log x}$
- $f^*(x) = \max_y[xy - y\log y]. \frac{d}{dy}[xy - y\log y] = x - \log y - 1 = 0$
    - $\log y = x - 1$, and thus $y = \exp(x-1)$
    - $f^*(x) = x\exp(x-1) - (x-1)\exp(x-1) = \exp(x-1)$
- $f^{**}(x) = \max_y[xy - \exp(y-1)]. \frac{d}{dy}[xy - \exp(y-1)] = x - \exp(y-1) = 0$
    - $x = \exp(y-1)$, and thus $\log x = y - 1$ and $y = 1 + \log x$
    - $f^{**}(x) = x(1+\log x) - \exp(1 + \log x - 1) = x\log x + x - x = \boldsymbol{x\log x} = \boldsymbol{f(x)}$
- Claim: f is convex iff $f = f^{**}$ (also needs some other technical conditions)
- Deep concept with many other connections and properties…

F-Divergences

- $D_f(p \parallel q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx$, where f is strictly convex and $f(1) = 0$
- Example: $f(t) = t \log t$
- $D_f(p \parallel q) = \int q(x) \frac{p(x)}{q(x)} \log\left(\frac{p(x)}{q(x)}\right) dx = \int p(x) \log\left(\frac{p(x)}{q(x)}\right) dx \triangleq KL(p \parallel q)$
- Claim: $D_f(p \parallel q) \geq 0$, with equality iff $p = q$
- $D_f(p \parallel q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx \geq f\left(\int q(x) \frac{p(x)}{q(x)} dx\right) = f(1) = 0$
- If $p = q$, then $D_f(p \parallel q) = \int q(x) f(1) dx = \int q(x) \cdot 0 dx = 0$

Back to GANs

- Goal: Get density q_θ which is $\approx p$
  - $q_\theta(x)$ is the density fn at x of the following random variable: sample $z \sim N(0, I)$, and output $T_\theta(z)$ where T_θ is some neural network
- Specifically: $\min_\theta D_f\big(p(x) \parallel q_\theta(x)\big)$ (for some f)
  - Note that if we use the KL divergence, this is essentially maximum likelihood
    - $\arg\min_\theta KL\,(p(x) \parallel q_\theta(x)) = \arg\min_\theta \int p(x) \log p(x)/q_\theta(x)\, dx = \arg\min_\theta -\int p(x) \log q_\theta(x)\, dx \approx \arg\max_\theta \frac{1}{n} \sum \log q_\theta(x_i)$
  - In words: come up with a good generator which matches the data distribution
  - Nothing to do with any sort of “discriminator”… but we’ll derive one

Deriving the GAN loss

$$D_f\left(p(x) \parallel q_\theta(x)\right) = \int q_\theta(x) f\left(\frac{p(x)}{q_\theta(x)}\right) dx$$

$$= \int q_\theta(x)\left(\max_{S(x)\in\mathbf{R}} S(x)\,\frac{p(x)}{q_\theta(x)} - f^*\big(S(x)\big)\right) dx$$ (using $f^{**} = f$)

$$= \max_{S\in\mathbf{R}^d\to\mathbf{R}} \int p(x)S(x)dx - \int q_\theta(x) f^*\big(S(x)\big)dx$$

$$= \max_{S\in\mathbf{R}^d\to\mathbf{R}} E_{x\sim p}[S(x)] - E_{x\sim q_\theta}\left[f^*\big(S(x)\big)\right]$$

Deriving the GAN loss

$$\arg\min_{\theta} D_f\big(p(x) \parallel q_\theta(x)\big)$$

$$\approx \min_{\theta}\max_{\phi}\left[\int p(x)S_\phi(x)dx - \int q_\theta(x)f^*\Big(S_\phi(x)\Big)\,dx\right]$$

$$\approx \min_{\theta}\max_{\phi}\left[\frac{1}{n}\sum_{i=1}^{n} S_\phi(x_i) - \frac{1}{m}\sum_{j=1}^{m} f^*\Big(S_\phi\big(T_\theta(z_j)\big)\Big)\right]$$

T_θ: generator network, S_ϕ: discriminator network

x_i's are real data, $T_\theta(z_j)$'s are "fake" data. $z_j \sim N(0, I)$ for $j = 1$ to m

Jensen-Shannon GAN

- Use Jensen-Shannon divergence
  - $D_{JS}(p \parallel q) = KL\left(p \parallel \frac{p+q}{2}\right) + KL\left(q \parallel \frac{p+q}{2}\right)$
- Claim: $f_{JS}^{*} = -\log(1 - \exp(t)) \cancel{- \log 4}$
- Also reparametrize: $S \leftarrow \log S$

$$\min_{T_\theta} \max_{S_\phi} \frac{1}{n} \sum \log S_\phi(x_i) + \frac{1}{m} \sum \log\left(1 - S_\phi\left(T_\theta(z_j)\right)\right)$$

Fix T_θ, then the maximization problem is roughly a cross-entropy loss

Fix S_ϕ, optimizing T_θ tries to “fool” discriminator into being wrong

(Draw Real data vs Fake data fed into Discriminator, has to guess 0 or 1)

After, can throw out discriminator, just use generator

Optimizing a GAN

- Have to update two parameters at once… tougher than before

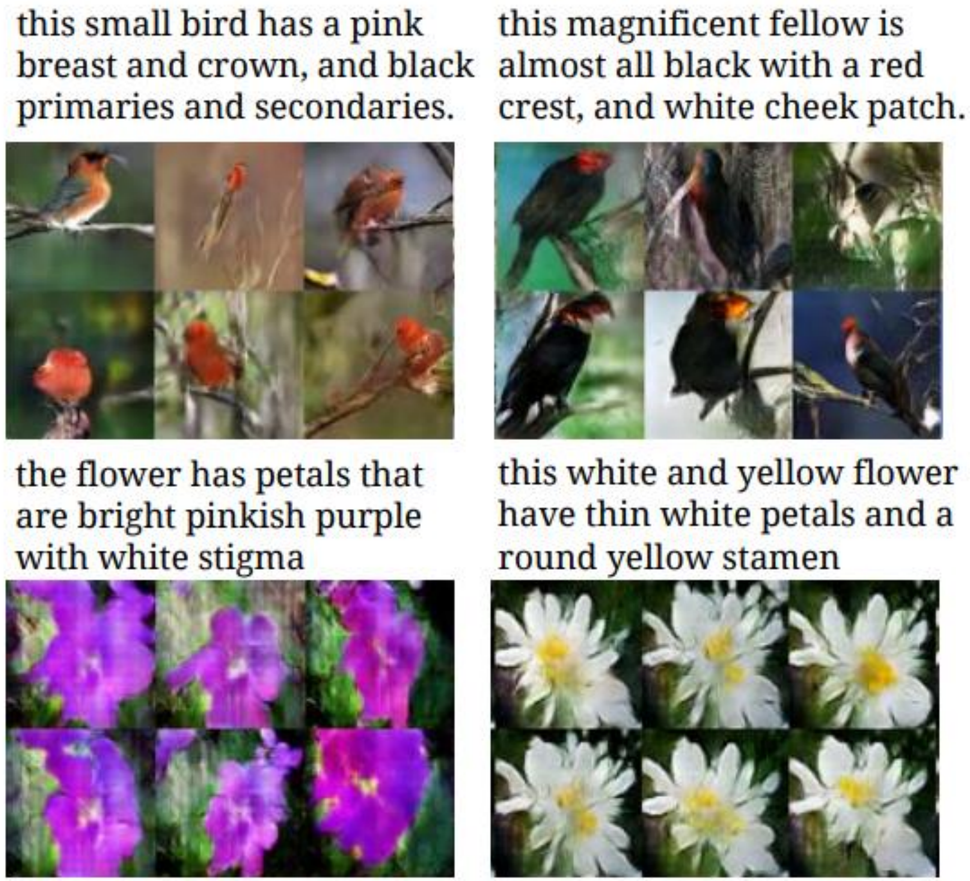
$$\phi^{(t+1)} \leftarrow \phi^{(t)} + \eta_\phi \nabla_\phi \left[\frac{1}{n}\sum \log S_{\phi^{(t)}}(x_i) + \frac{1}{m}\sum \log\left(1 - S_{\phi^{(t)}}\left(T_{\theta^{(t)}}(z_j)\right)\right)\right]$$

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta_\theta \nabla_\theta \left[\frac{1}{m}\sum \log\left(1 - S_{\phi^{(t)}}\left(T_{\theta^{(t)}}(z_j)\right)\right)\right]$$

- Take step on both parameters at the same time
  - Can also take alternating steps, multiple steps on one parameter and then one on the other, etc.
- GANs can be notoriously difficult to optimize
  - Sensitive to hyperparameters

Generating Faces

Text to image



Superresolution

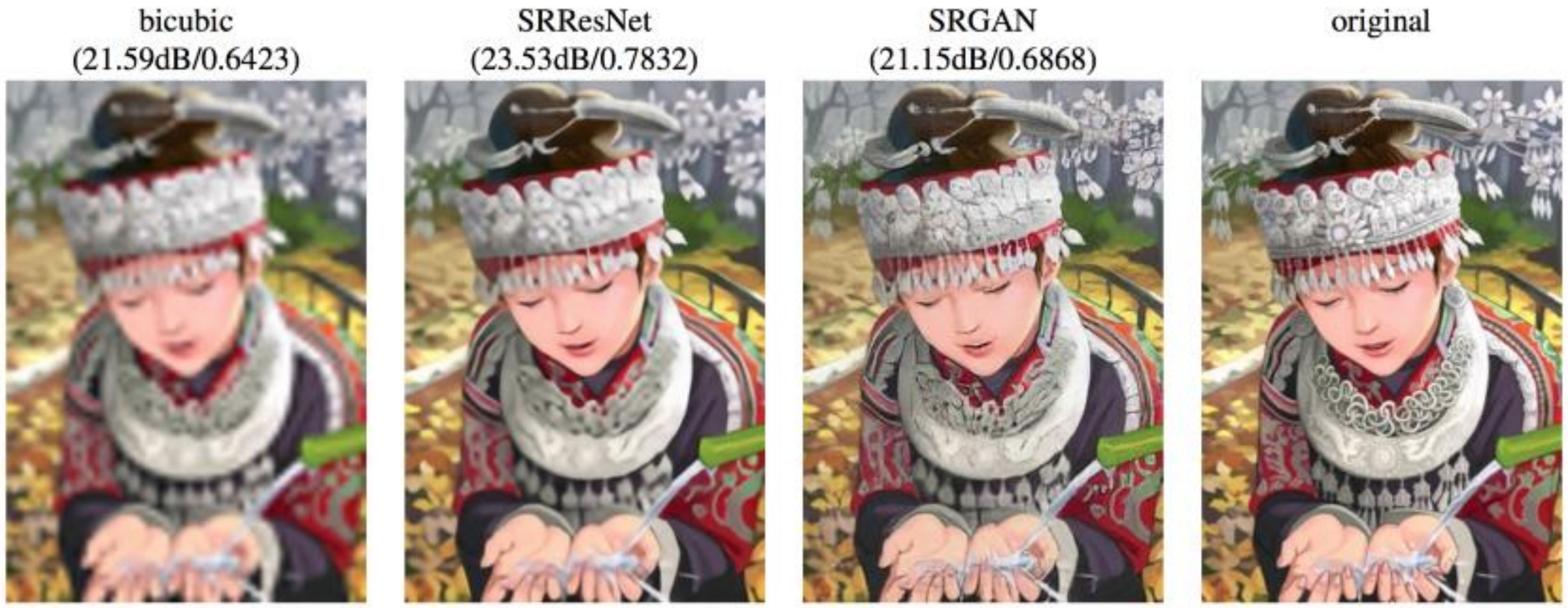
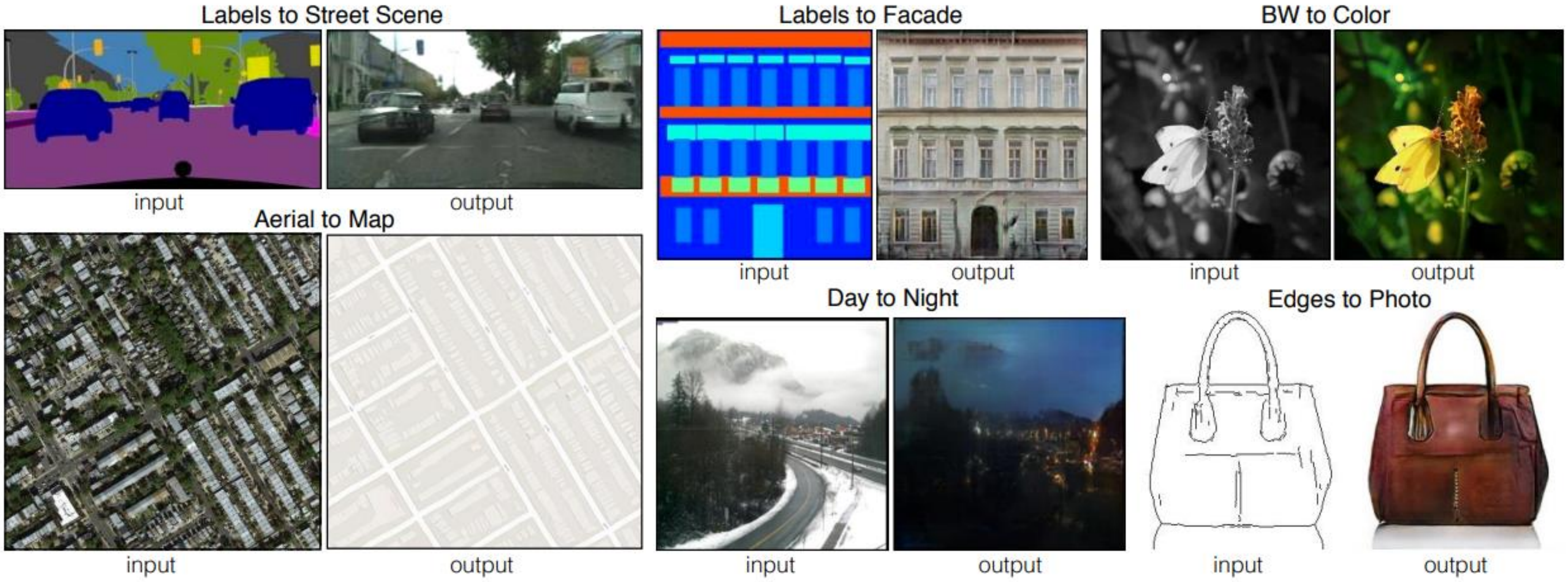


Image-to-Image Translation



GAN Arithmetic