Recent Generative Models

What is a generative model?

↳ In ML, we see a lot of assigning class probability to outcomes

conditional $p(y|x) \longrightarrow$ given image, the probability it is a cat, dog, etc

Generative model says: $\overset{\text{joint}}{p(x,y)} \rightarrow$ what's the probability of the image and the label?

↳ what is the distribution that generates the data? → unsupervised, just $p(\vec{x})$ ↑ no relational mapping

- more powerful learned representation. Can use it to estimate $p(y|x)$

$$p(x,y) = p(x|y)p(y) \propto p(y|x)$$

- can use to generate synthetic data → can check in advance if model can explain any data.

- The issue is that it is generally difficult to estimate the most general of parametric distributions. Estimating likelihood $p(\vec{x})$ is generally intractable☆. New models have clever ways of circumventing this issue.

☆ when you assume hidden representation. See PixelRNN for counter ex.

Circumstance

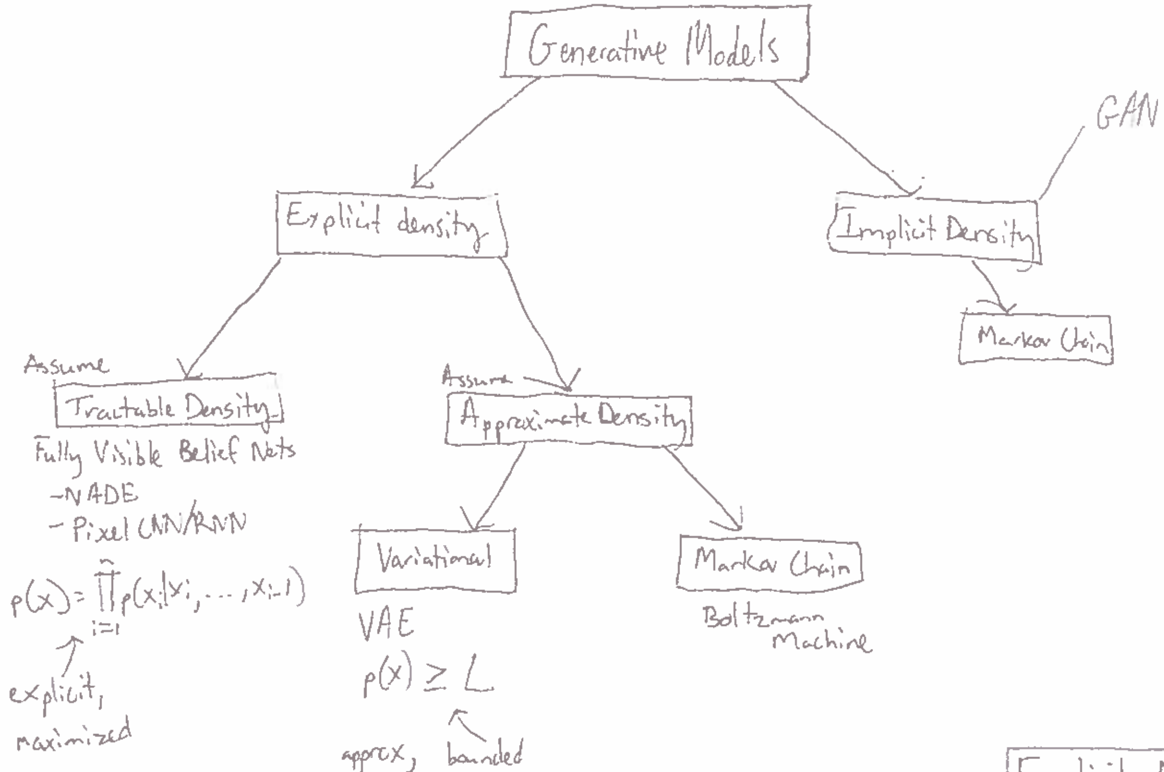
- Say we are given some $X = \{\vec{x}_i\}_{i=1}^{N}$ for some discrete or continuous.
- Say X are actually generated from some hidden variable $\vec{z} \sim p(\vec{z})$
- We want to make inference on the data and its generating process, but all the components to do this seem intractable to compute, namely:

marginal likelihood $p(\vec{x}) = \overset{\text{hard}}{\int} p(\vec{x}|\vec{z})p(\vec{z})d\vec{z}$ posterior density $p(\vec{z}|\vec{x}) = \dfrac{p(\vec{x}|\vec{z})\,p(\vec{z})}{p(\vec{x})}$ ← hard

- w/ the nonlinearities of NNs and large datasets now used.

This is the circumstance some new models try to circumvent.

Generative Model Taxonomy



Explicit Models

PixelRNN, for example, by van der Oord et al. 2016

- Dependency on previous pixels modeled using an RNN (LSTM). They work very well
- Drawback → Sequential Generation can be slow for many pixels

PixelCNN, van der Oord et al 2016

- Previous pixel dependency modeled w/CNN over neighboring region. Could be interesting for 2D grids
- No longer looking at "previous" but rather proximal.
- Softmax at each pixel location, on images for [0, ..., 255] possible values
- faster training b/c conv, but still slow generation

$$\underbrace{}$$

- explicit p(x) → good evaluation metric, good samples
- slow

VAE | Kingma, Welling 2013

- Main idea: if you can't maximize $p(x)$ b/c of this intractability, can you at least put a lower bound on it?

We want to be able to sample from $p(z)$ and use that sample to generate a sample from $p(x|z)$ ⟵ this will help us generate new samples that approximately come from original distribution

We will work with 2 conditional distributions: $p(x|z)$ modeled by $p_\theta(x|z)$

$q(z|x)$ modeled by $q_\phi(z|x)$

where θ, ϕ are the set of parameters for 2 neural networks

We use these to try and maximize our estimate of the evidence of the data $p_\theta(x)$:

$$\ln p_\theta(x) = E_{z \sim q_\phi(z|x_i)}\left[\ln p_\theta(x_i)\right]$$

$$= E_z\left[\ln \frac{p_\theta(x|z)\, p_\theta(z)}{p_\theta(z|x)} \frac{q_\phi(z|x)}{q_\phi(z|x)}\right]$$

$$= E_z\left[\ln p_\theta(x|z)\right] - E_z\left[\ln \frac{q_\phi(z|x)}{p_\theta(z)}\right] + E_z\left[\ln \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

$$= \underbrace{E_z\left[\ln p_\theta(x|z)\right] - D_{KL}\left(q_\phi(z|x) \,||\, p_\theta(z)\right)}_{\text{maximize the lower bound of the evidence}} + D_{KL}\left(q_\phi(z|x) \,||\, p_\theta(z|x)\right)$$

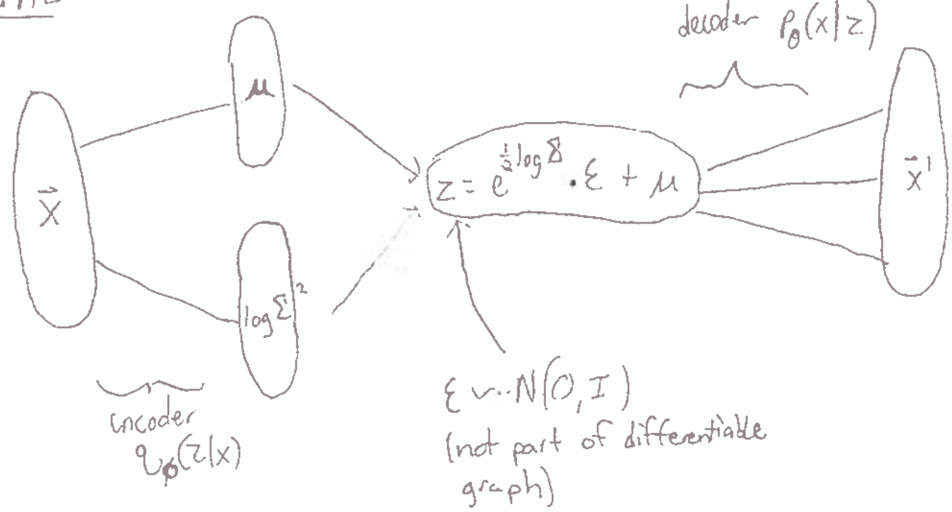
1 estimate through sampling, reconstruction

2 analytically calculable if we assume $p(z)$ is an easily parametrized dist like $p(z) \sim N(0, I)$

3 posterior intractable as seen earlier, but $D_{KL} \geq 0$!!!

What does this look like in practice? Turn the page

Training a VAE



- Feed $\vec{x}$ to $\vec{\mu}$, $\overrightarrow{\log \Sigma^2}$ layers $\longrightarrow$ use to generate some $\vec{z}$ using a random $\varepsilon \sim N(0, I)$ that doesn't require differentiation.
- Optimize reconstruction with cross-entropy or MSE
- Optimize $D_{KL}(q_\phi(z|x) \| P(z)) \longleftarrow$ can do directly with μ, Σ

↳ we know $q(z|x)$ goes like $\sim N(\mu, \Sigma)$ from our model, where μ, Σ are functions of X:

$$D_{KL}(q_\phi(z|x) \| P(z)) = D_{KL}(N(\mu, \Sigma) \| N(0, I))$$

$$= \frac{1}{2}\left(tr(\Sigma) + \mu^T\mu - k - \log\det(\Sigma)\right) \quad (\Sigma \text{ already diagonal})$$

where k is dimension of Gaussian:

treating diagonals of Σ as a vector

$$= \frac{1}{2}\left(\sum_k \Sigma + \sum_k \mu^2 - \sum_k 1 - \log \prod_k \Sigma\right)$$

$$= \frac{1}{2}\left(\sum_k \Sigma_i + \sum_k \mu^2 - \sum_k 1 - \sum_k \log \Sigma_i\right)$$

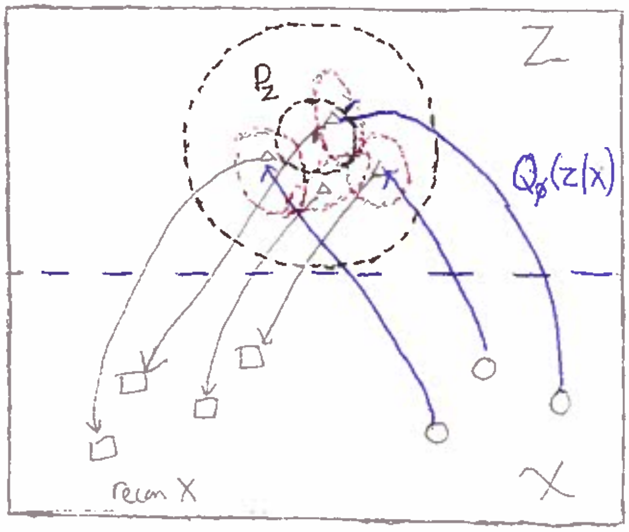
$$= \frac{1}{2}\sum_k \left(\Sigma + \mu^2 - 1 - \log \Sigma\right) \quad \checkmark$$

Overall

- Reconstruction requires that encoded z's correspond to decoded real samples
  ↳ enc & dec approximate inverses of each other
- KL latent distribution mapping ensures continuous, navigable latent space, ideally s.t. navigating through latent space corresponds to navigating through space of samples

Wasserstein Autoencoders improve this.

Visualizing VAE Latent Space Correspondence



- When well trained, sample μ, Σ from encoder are very near Gaussian/whatever prior is chosen.
- Including reconstruction error forces them to not all be exact/overlapping
- stable training (Pro), but limited resolution (Con) because of approx. likelihood max and reconstruction error.

Con
- multiple original x's run the risk of being most similar the same reconstructed x' b/c of intersection of latent embedd gaussians

↓

Wasserstein Autoencoder improves this by making $q(z|x) = \int q_\phi(z|x=x)\,dP_x$ match $P(z)$, which is a continuous mixture

What else can you do with them?

- improve interpolation in the latent space by learning independent features → minimizing Total Correlati

↓

- can be done with adversary or independence criteria
- implicitly what β-VAE does.

Avoiding explicit likelihood estimation with GAN (Generative Adversarial Networks) Goodfellow 2014

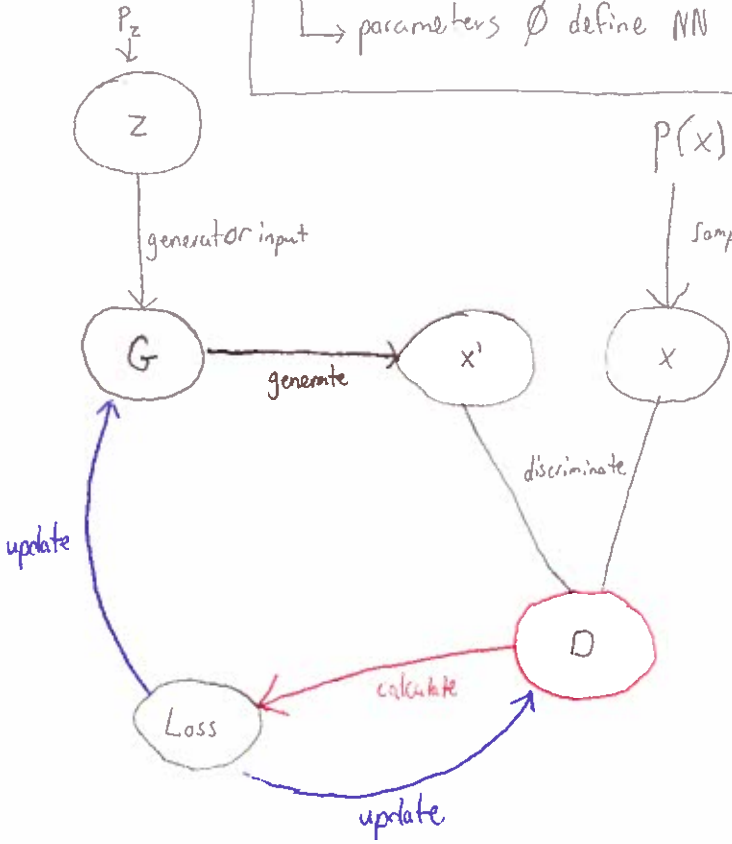
- This time, no explicit estimate of density $p(x)$, which is complex, high-dimensional
- Two networks are jointly optimized in tandem with one another → one that learns to generate samples from an approximate distribution $p_\theta(x) \approx p(x)$ and one that tries to discriminate samples from p_θ from those of p.*
- two-player game between generator and discriminator to transform samples from latent $z \sim p(z)$ to $z \rightarrow x \sim p_\theta(x)$

  ↳ parameters θ define neural network which transforms z in z-space to some x in x-space $\equiv \boxed{G(\vec{z};\theta)}$

  ↳ parameters ϕ define NN which outputs probability that a given x or x' is real $\equiv \boxed{D(x \text{ or } x';\phi)}$

*In reality, we don't have access to the true p, just a set of finite samples from it.

Model:



The loss function

- Generator and Discriminator participate in min-max game for value function V:

$$\min_G \max_D V(D,G) = E_{x\sim p(x)}\left[\ln D_\phi(x)\right] + E_{z\sim p(z)}\left[\ln\left(1 - D(G(z))\right)\right]$$

Loss function analysis of original GAN framework

- What are optimal D, G?
- Let's look at case of optimal D.
- Using $\int$ instead of expectation b/c dealing w/ complete distributions

So $V(D, G) = \int_x p_{real}(x) \log D(x)\, dx + \int_z p_z(z) \log(1 - D(G(z)))\, dz$

$$= \int_x \underbrace{p_{real} \log D(x)\, dx + p_g(x) \log(1 - D(x))\, dx}_{\text{Optimal discriminator to maximize this is: } D^*(x) = \frac{p_{real}(x)}{p_{real}(x) + p_g(x)}}$$

where $p_g(x)$ is probability that generator produces x

Using this, we rewrite

$$V^* = \min_G V(D^*, G) = E_x[\log D^*(x)] + E_{x \sim p_g}[\log(1 - D^*(x))]$$

$$= E_x\left[\log \frac{p_{real}(x)}{p_{real}(x) + p_g(x)}\right] + E_{x \sim p_g}\left[\log \frac{p_g}{p_{real} + p_g}\right]$$

when $p_g = p_{real}$, $D^* = \frac{1}{2}$. $p_g = p_{real}$ is optimal scenario. In this case

$$E_x\left[\log \frac{1}{2}\right] + E_{x \sim p_g}\left[\log \frac{1}{2}\right] = -\log 4 \text{ is lowest value}$$

Factoring this from the RHS of V^* will give us the minimum plus whatever needs to be minimized to 0 to achieve that minimum:

$$V^* = -\log 4 + E_x\left[\log \frac{p_{real}}{(p_{real} + p_g)/2}\right] + E_{x \sim p_g}\left[\log \frac{p_g}{(p_{real} + p_g)/2}\right]$$

$$= -\log 4 \; D_{KL}\left(p_{real} \,\Big\|\, \frac{p_{real} + p_g}{2}\right) + D_{KL}\left(p_g \,\Big\|\, \frac{p_{real} + p_g}{2}\right)$$

$$= -\log 4 + D_{JS}(p_{real} \| p_g)$$

$\therefore$ Finding optimal value function sol'n is equivalent to minimizing the Jensen-Shannon Divergence.

GAN perks

- highly parameterized replacement for likelihood means potential for diverse and expressive samples
- Theoretically motivated/sound optimal equilibrium for <u>distribution matching</u>

Original GAN problems

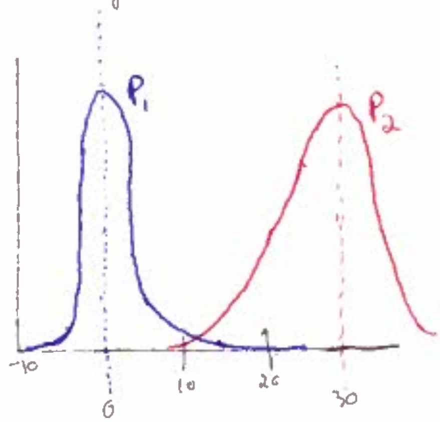
- In practice, hard to minimize this D_{JS}
- Generator/Discriminator can outperform one another. Training "gets stuck"
  - ↳ This is symptomatic of:

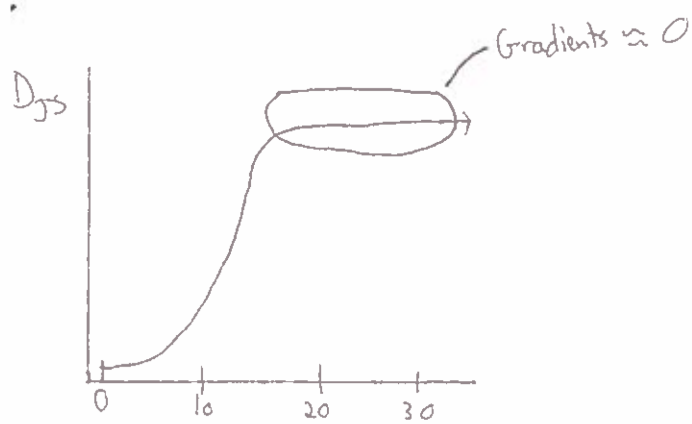
- Overfitting
- Density misspecification
  - → no parameters for which P_g and P_{real} are similar enough
- Dimensional misspecification
  - → real data might be on a lower dim. manifold

And evidenced by:

- mode collapse and/or diminished gradients
- non-convergence:
  - → oscillating parameters, no stability in training

Imagine comparing 2 distributions using D_{JS}:





Two Common Solutions to GAN Problems: Wasserstein GAN and GAN w/ Discriminator Gradient Penalty

Wasserstein GAN: Replace f-divergence with integral probability metric (WGAN-GP) Arjovsky 201 Gulrajani 201

(and improvements to it)

- question of optimal transport (OT) → what is the best (minimum cost) measure of how much density of the model distribution needs to be moved to make distributions equal?

Wasserstein Distance: $W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} E_{(x,y)\sim\gamma}\left[\|x-y\|\right]$ aka Earth Mover distance

- $\Pi(P_r, P_g)$ is set of all joint distributions $\gamma(x,y)$ whose marginals are P_r, P_g
- infimum is greatest lower bound for any transport plan

☆ • this equation is intractable to estimate, but can be expressed under Lipschitz constraint:

$$W(P_r, P_g) = \sup_{\|f\|_L \leq 1} E_{x\sim P_r}\left[f(x)\right] - E_{x\sim P_\theta}\left[f(x)\right]$$

- 1-Lipschitz functions are just functions that slopes/gradients ≤ 1 for all points $\frac{|f(x)-f(y)|}{|x-y|} \leq 1$
- This simplification is derived via the Kantarovich-Rubinstein duality

- New objective: The discriminator learns this supremal function $f(x)$ and the distance is minimized
  - Discriminator no longer a discriminator ⟶ now a critic
    - used to calculate a metric of distributional distance

1 Problem

- hard to maintain Lipschitz constraint when learning $f(x)$
- can penalize model if gradient becomes >1:

$$L = E_{\tilde{x}\sim P_\theta}\left[D(\tilde{x})\right] - E_{x\sim P_r}\left[D(x)\right] + \lambda E_{\hat{x}\sim P_{\hat{x}}}\left[\left(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1\right)^2\right]$$

where $\hat{x} = t\tilde{x} + (1-t)x$ w/ t uniformly sampled b/w 0 and 1

GAN w/ Discriminator Gradient Penalty (GAN-DP) Roth 2017

- similar gradient penalty to improving WGAN can be used to improve original GAN training.
- this method is locally convergent in general case, wheras WGAN is not
- $$L = E_x\left[\ln D_\phi(x)\right] + E_z\left[\ln\left(1 - D(G(z))\right)\right] + \underbrace{\frac{\gamma}{2} E_x\left[\|\nabla D_\phi(x)\|^2\right]}_{\text{regularizer that prevents Discriminator from deviating from Nash Equilibrium}}$$
- Stability studies on next page. WGAN-GP only stable in this context at very low learning rate $lr = 10^{-5}$

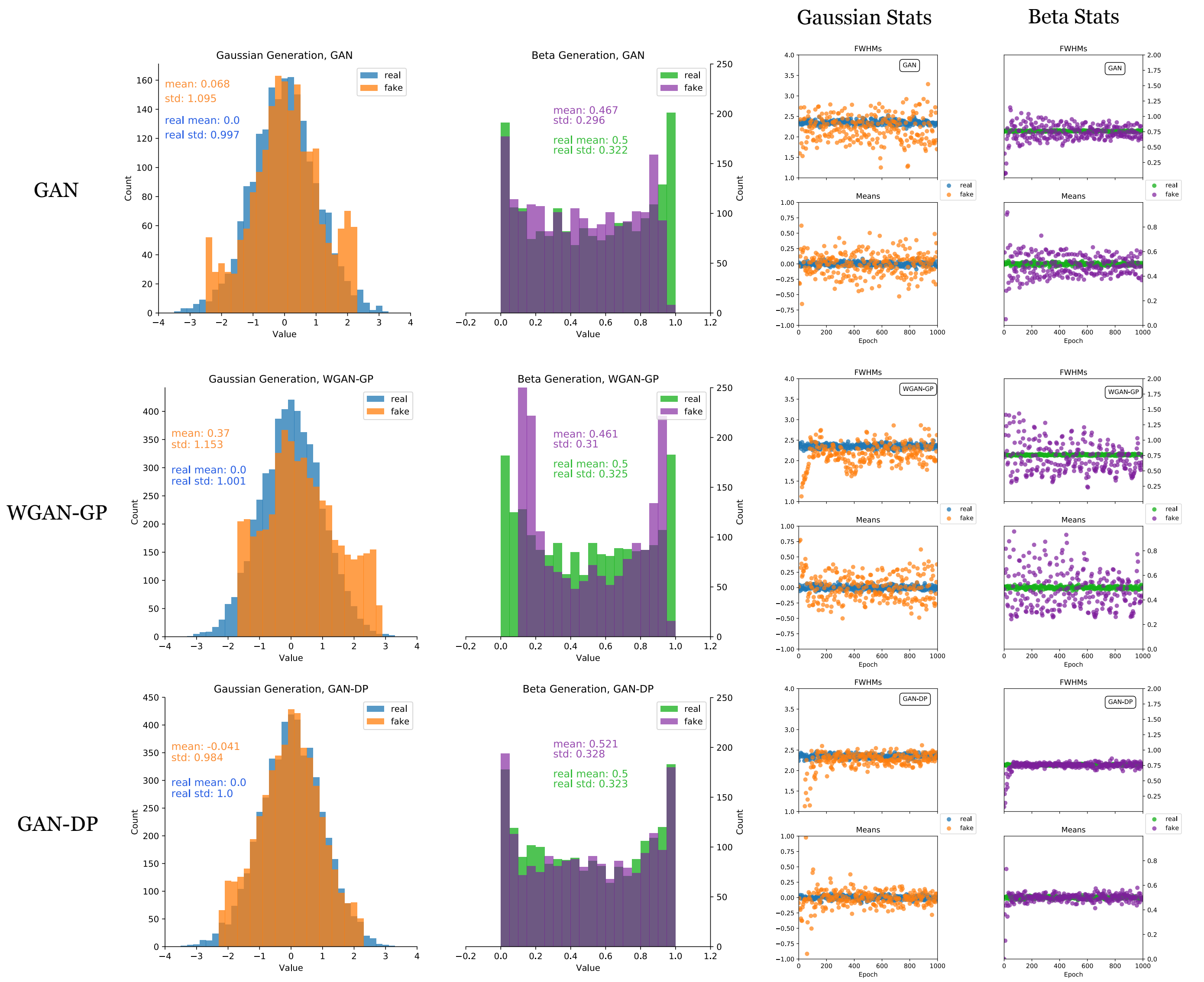


Figure 2.3: Top Row: Generation of Gaussian and Beta distributions using original GAN loss. Middle Row: Generation of Gaussian and Beta distributions using WGAN-GP loss. Bottom Row: Generation of Gaussian and Beta distributions using GAN-DP loss. The right half contains values of the sample mean and standard widths of the two generated distributions across epochs compared to ground truth.

accuracy, both due to my selecting of sample distributions to display and due to the stochasticity of sample generation. Yet, the continuous comparison of some metrics that characterize the Gaussian and Beta distributions across training epochs allows for more formative evaluation between the models. The WGAN-GP and unregularized GAN showed evidence of deviating from expected characteristics even after approximating them better in previous epochs, while the GAN-DP does not show such inconsistencies. Upon closer inspection of the WGAN-GP results, in which the training in the Beta case seems to be converging but at a much slower rate, another test was done with minute learning rate of 10^{-5} which was outside of the original hyperparameter search. Results show that the WGAN-GP could move down the loss surface more stably if trained with very small gradient updates, as shown in Figure 2.4.[3] Moreover, a comparison of the KL-divergences between estimates of the probability distributions from the generated samples and the true samples corroborates these claims. The average KL-divergences over 100 samples of 10000 values each were lowest for the GAN-DP and higher for the unregularized GAN and WGAN-GP models on both the Gaussian and Beta tests, as shown in Table 2.2. The WGAN-GP value improved significantly to 0.024 when training was locally convergent in the slow learning rate case. The KL-divergence was computed by creating normalized histograms of the 10000 values so that there could be a density estimates $p_{GAN}(\mathbf{x})$ and $q_{true}(\mathbf{x})$ at each bin. A Kernel Density Estimation was also tested to compare to the normalized histogram binning technique and it yielded similar results.

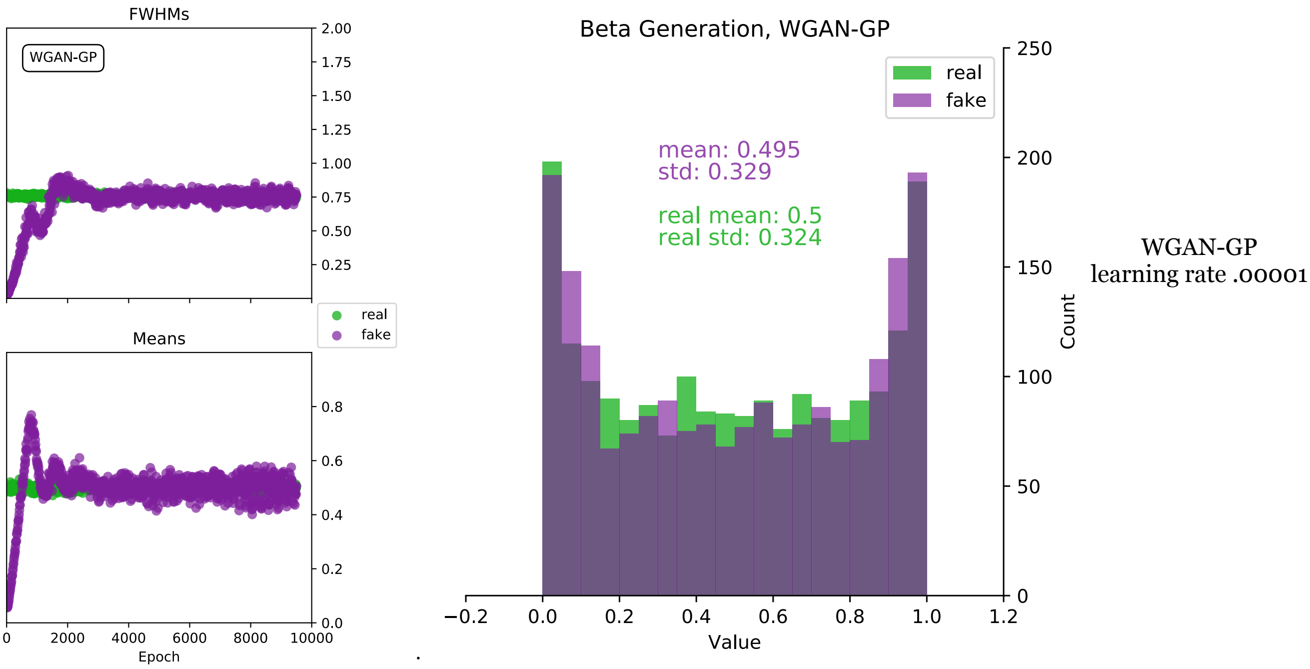


Figure 2.4: WGAN-GP training only achieves accuracy with small learning rate and many epochs.

Overall, the GAN-DP method could accurately model both the Gaussian and Beta distributions, capturing natural parameters of the distributions. It empirically struggles to generate values

[3]See Appendix B.1 for comparison of local stability across learning rates.