10-708: Probabilistic Graphical Models, Spring 2020

13 : Deep Generative Models II

Lecturer: Eric P. Xing *Scribe: Cheng Cheng, Chiyu Wu, Xinhe Zhang*

1 Generative Adversarial Networks (GANs)

1.1 Vanilla GAN [GPAM$^+$14]

The goal of Vanilla GAN in general is to generate new images via learning from the original image data. The model primarily consists of two parts, i.e. a generator which produces image from noise z and a discriminator which tries to distinguish between z and the true images x. The Vanilla GAN is performing an minimax game between a generator G and a discriminator D. G tries to generate samples as close to the true data as possible whereas D tries to distinguish them apart. The loss objective is therefore formulated as:

$$\min_G \max_D V(D,G) = E_{x\sim p_x(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

where $z \sim p_z(z)$ is a random noise (usually standard Gaussian) and $x \sim p_x(x)$ is the true data distribution. Images generated from GAN are pretty good without any traditional feature engineering.

1.2 Wasserstein GAN [GAA$^+$17]

Vanilla GAN relies on KL-divergence which causes instability during training. For instance, if the data and model's manifolds are different, there may exist x such that $P_g(x) = 0$ but $P_d(x) > 0$, in which the KL divergence is infinite.

In order to solve this problem, Wasserstein GAN (WGAN) is proposed. WGAN relies on Wasserstein distance from the optimal transport literature. It measures the minimum transportation cost for transforming one distribution into another distribution.

For WGAN to work, the Lipschitz continuity property $|D|_L \leq K$ need to be ensured. The loss term is therefore formulated as:

$$W(p_d,p_g) = \frac{1}{K}\sup_{||D||_L\leq K} E_{x\sim p_d}[D(x)] - E_{x\sim p_g}[D(x)]$$

Compared to vanilla GAN, WGAN provides more stable gradients at the place where vanilla GAN has vanishing gradients (Fig.1).

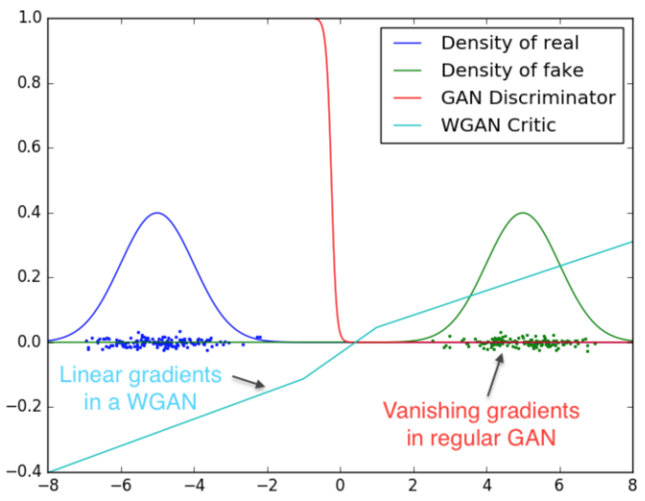

Figure 1: WGAN vs. Vanilla GAN on Gaussian data [GAA+17]

1.3 Progressive GAN [KALL18]

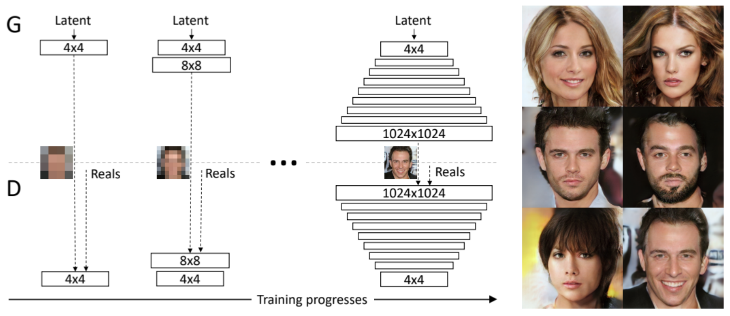

Figure 2: Progressive GAN for generating facial images [KALL18]

The classic GAN-based methods can only generate images with low resolution. Therefore, the progressive GAN is proposed to enhance the quality of training and generating large images.

Intuitively, progressive GAN works to first extract structure of the image using lower layers and then attend to details on the image. Therefore during training, low resolution images are first generated, and then their resolutions are increased by adding additional layers.

Progressive GAN is shown to generate high resolution facial images (Fig.2).

1.4 Big GAN [BDS19]

The authors of BigGAN finds that by up-scaling the Vanilla GAN, significantly better learning results could be achieved. In the paper, $2x - 4x$ more parameters as well as $8x$ larger batch size are experimented. Also, model collapse is avoided by utilizing a strong discriminator at the initial training stage, and then gradually relax it. High quality images are generated by Big GAN (Fig.3)

Figure 3: Images generated by Big GAN [BDS19]

2 Normalizing Flow (NF)

2.1 Overview

Normalizing Flows can transform a simple distribution into a complex one by applying sequence of invertible transformation functions. Through a chain of transformation, we replaced the current variable by the new one and eventually obtain a probability distribution for the target variable. Figure 4 shows the whole chain of the normalizing flow.

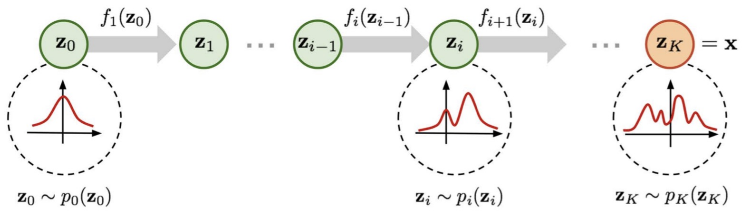
$f_1(\mathbf{z}_0)$
$\mathbf{z}_0$
$\mathbf{z}_1$
$\mathbf{z}_{i-1}$
$f_i(\mathbf{z}_{i-1})$
$\mathbf{z}_i$
$f_{i+1}(\mathbf{z}_i)$
$\mathbf{z}_K = \mathbf{x}$
$\mathbf{z}_0 \sim p_0(\mathbf{z}_0)$
$\mathbf{z}_i \sim p_i(\mathbf{z}_i)$
$\mathbf{z}_K \sim p_K(\mathbf{z}_K)$

Figure 4: The process of a normalizing flow, transforming a simple distribution $\mathbf{p}_0(\mathbf{z}_0)$ to a complex distribution $\mathbf{p}_K(\mathbf{z}_K)$. Figure courtesy: Lilian Weng

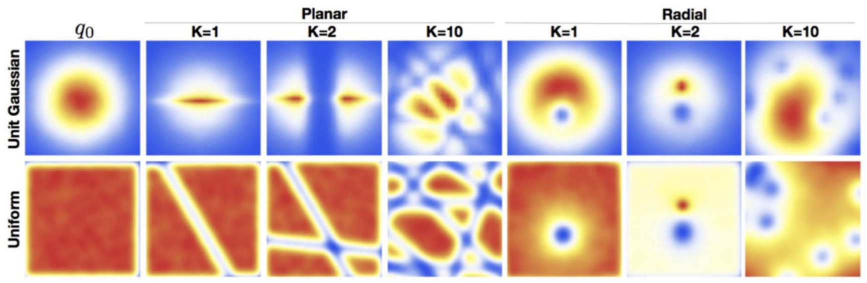

Figure 5: Effect of normalizing flow on two distributions.

Figure 5 shows the effect of the normalizing flows. We can see that a spherical Gaussian distribution can be transformed into a bimodal distribution through two successive transformations [RM15].

Now given a random variable $\mathbf{z}$ from a simple distribution $\mathbf{p}(\mathbf{z})$, we apply a invertible transformation function

$\mathbf{f}$ to obtain a new variable $\mathbf{x}$ from a more complex distribution.

$$\begin{aligned}\mathbf{z} &\sim \mathbf{p}(\mathbf{z})\\ \mathbf{x} &= \mathbf{f}(\mathbf{z})\\ \mathbf{z} &= \mathbf{f}^{-1}(\mathbf{x})\end{aligned}$$

According to Change of Variable Theorem, we have:

$$\begin{aligned}\mathbf{p}(\mathbf{x}) &= \mathbf{p}(\mathbf{z})\left|\det \frac{\mathbf{dz}}{\mathbf{dx}}\right|\\ &= \mathbf{p}(\mathbf{f}^{-1}(\mathbf{x}))\left|\det \frac{\mathbf{df}^{-1}}{\mathbf{dx}}\right|\end{aligned}$$

where $\left|\det \frac{\mathbf{df}^{-1}}{\mathbf{dx}}\right|$ is the Jacobian determinant of the function $\mathbf{f}^{-1}$.

Now considering the normalizing flow in the figure 4, we need to obtain $\mathbf{z}_K$ through chain of transformations $\mathbf{f}_1$, $\mathbf{f}_2$, ..., $\mathbf{f}_K$. That is,

$$\begin{aligned}\mathbf{z}_0 &\sim \mathbf{p}(\mathbf{z}_0)\\ \mathbf{x} = \mathbf{z}_K &= \mathbf{f}_K \cdot \mathbf{f}_{K-1} \ldots \mathbf{f}_0(\mathbf{z}_0)\\ \mathbf{z}_i &= \mathbf{f}_i^{-1}(\mathbf{z_{i-1}})\\ \mathbf{p}(\mathbf{z_i}) &= \mathbf{p}(\mathbf{z_{i-1}})\left|\det \frac{\mathbf{dz_{i-1}}}{\mathbf{dz_i}}\right|\end{aligned}$$

Then the training objective is to maximize the data log-likelihood:

$$\log \mathbf{p}(\mathbf{x}) = \log \mathbf{p}(\mathbf{z}_0) + \sum_{i=1}^{K} \log \left|\det \frac{\mathbf{dz_{i-1}}}{\mathbf{dz_i}}\right|$$

2.2 Case Study: GLOW

GLOW is a flow based generative model, consisting of a series of steps of flow and combined in a multi-scale architecture [KD18], as shown in figure 6. Each step of flow consists of actnorm, an invertible 1×1 convolution and a coupling layer.

For the actnorm layer, it is similar to batch normalization [IS15]. Any channel after the actnorm activation layer will have zero mean and unit variance. The key difference between actnorm layer and batch normalization layer is that actnorm can work reasonably well when batch size is only 1. 1×1 convolutional layer is used to perform permutation operation. And LU Decomposition is applied to reduce the computational cost of calculating the determinant of weight matrix $\det(W)$. The affine Coupling Layer is a powerful reversible transformation.

GLOW demonstrates a significant improvement performance in terms of log-likelihood on standard image modeling benchmarks.

3 Integrating Domain Knowledge into Deep Learning

Deep learning has been proven very successful in lots of areas. However, deep learning still suffers from several limitations. It heavily rely on massive labeled data which is really expensive. Also, deep network can

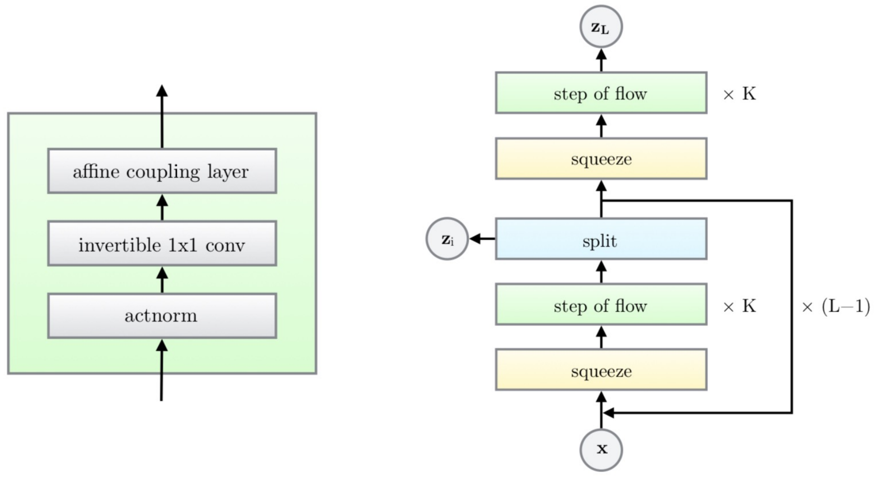

Figure 6: The architecture of GLOW

be regarded as a black-box, trained end-to-end, which is uninterpretable. Furthermore, it is hard to encode human intention and domain knowledge into the deep neural network. For human, we not only learn from concrete examples as deep neural networks, but also learn from abstract knowledge such as logic rules.

Now we hope to integrate domain knowledge into deep neural network. Consider a statistical model $\mathbf{p}_\theta(\mathbf{x})$ and a constraint function $\mathbf{f}_\phi(\mathbf{x})$:

$$\mathbf{x} \sim \mathbf{p}_\theta(\mathbf{x})$$
$$\mathbf{f}_\phi(\mathbf{x}) \in \mathbf{R}$$

where higher $\mathbf{f}_\phi(\mathbf{x})$ value means our generated x is better with regard to the prior knowledge.

Let's consider a image generation problem, as shown in Figure 7. Here we hope to generate images which has the consistent pose with the input pose template. We first use $\mathbf{p}_\theta$ as our generative model, which has two inputs, one is the source image, and the other is the target pose template. Then we apply a constraint function $\mathbf{f}_\phi$ to ensure the generated image having the consistent structure with the true target. Here $\mathbf{f}_\phi$ can be regarded as a human part parser, able to extract poses from an image. By this way, we can generate new images with desired structures.

4 Learning with Constraints

Just like what we do with GAN training, we fold the constraint function $f_\phi(x)$ under expectation of the generation distribution p_θ.

4.1 Objective

$$\min_\theta \mathcal{L}(\theta) - \alpha \mathbb{E}_{p_\theta}[f_\phi(x)]$$

Where the $\mathcal{L}(\theta)$ is the regular objective, such as the cross-entropy loss, etc. and $\alpha \mathbb{E}_{p_\theta}[f_\phi(x)]$ is the regularization, i.e. the imposed constraints, which is difficult to compute because when taking the derivative of

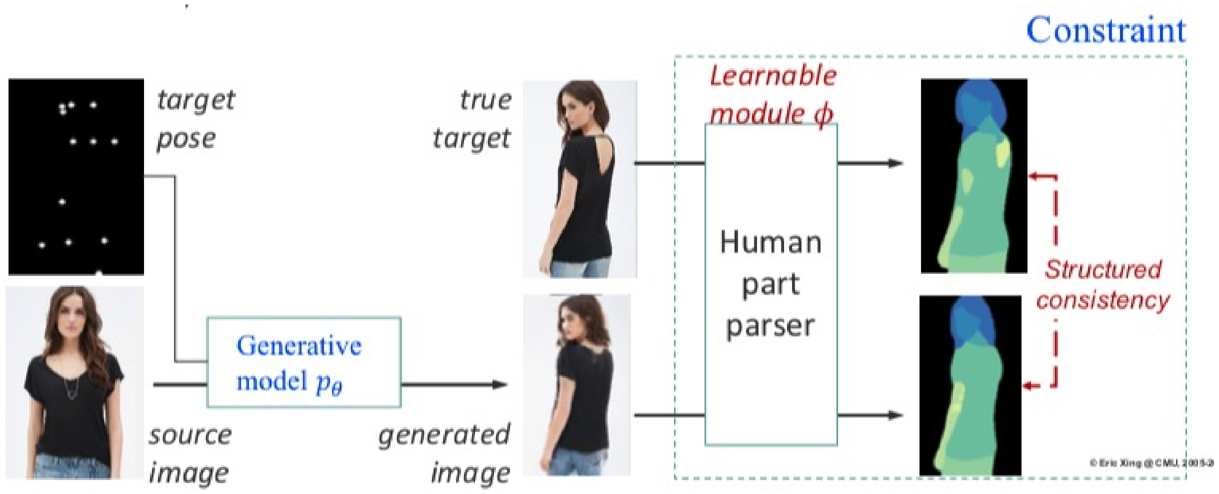

Figure 7: Pose guided person image generation

the expectation, we use the log probability trick. However, in this case, the magnitude of the log term will explode as explained in the weak-sleep algorithm.

Due to the difficulty of computing the regularization, we introduce a variational approximation q to the true distribution. We have a minimax situation: on the one hand, we would love to maximize the loss under the approximating distribution; on the other hand, we want to minimize the discrepancy between the true distribution and the approximation. As a result, we compute the KL divergence term minus the expected loss instead:

$$\mathcal{L}(\theta, q) = \mathrm{KL}(q(x) \| p_\theta(x)) - \lambda \mathbb{E}_q[f_\phi(x)]$$

This method is addressed as the posterior regularization method [GGT+10].

We then introduce a scaling term α to control the relative contribution of the two parts, our revised objective is:

$$\min_{\theta, q} \mathcal{L}(\theta) + \alpha \mathcal{L}(\theta, q)$$

4.2 Learning

An EM-like approach is applied.

E-step:

$$q^\star(x) = p_\theta(x) \exp\{\lambda f_\phi(x)\}/Z$$

M-step:

$$\min_\theta \mathcal{L}(\theta) - \mathbb{E}_{q^\star}[\log p_\theta(x)]$$

4.3 Logical Rule Constraints

Putting everything together, the set up now is we consider a supervised learning $p_\theta(y|x)$, our input-targe space is (X, Y), and our first-order logic rules (r, λ).

Given l rules, we can train the model by alternatively training for the variational distribution $q^\star(y|x)$ and the true generative model:

1. E-step

$$q^\star(y|x) = p_\theta(y|x)\text{exp}\left\{\sum_l \lambda_l r_l(y,x)\right\}/Z$$

2. M-step

$$\min_\theta \mathcal{L}(\theta) - \mathbb{E}_{q^\star}[\log p_\theta(y|x)]$$

Note in the M-step we are using the variational which is easier to compute.

4.4 Rule Knowledge Distillation

Instead of learning a difficult target $p_\theta(y|x)$one-shot, we iterate with an auxiliary $q(y|x)$, which is called the "teacher" and it is often an ensemble. The target $p_\theta(y|x)$ is called the student. We match the soft predictions of the teacher network and the student network. This interaction between the teacher(s) and the student will ultimately get the student closer to the teachers.

With our setup, at each iteration t, we update the $\theta^{(t+1)}$ with:

$$\arg\min_{\theta\in\Theta} \frac{1}{N}\sum_{n=1}^{N}(1-\pi)\ell(y_n, \sigma_\theta(x_n)) + \pi\ell(s_n^{(t)}, \sigma_\theta(x_n))$$

where π is the balancing parameter, y_n is the true hard label, $\sigma_\theta(x_n)$ is the soft prediction of $p_\theta(y|x)$ and $s_n^{(t)}$ is the soft prediction of the teacher network.

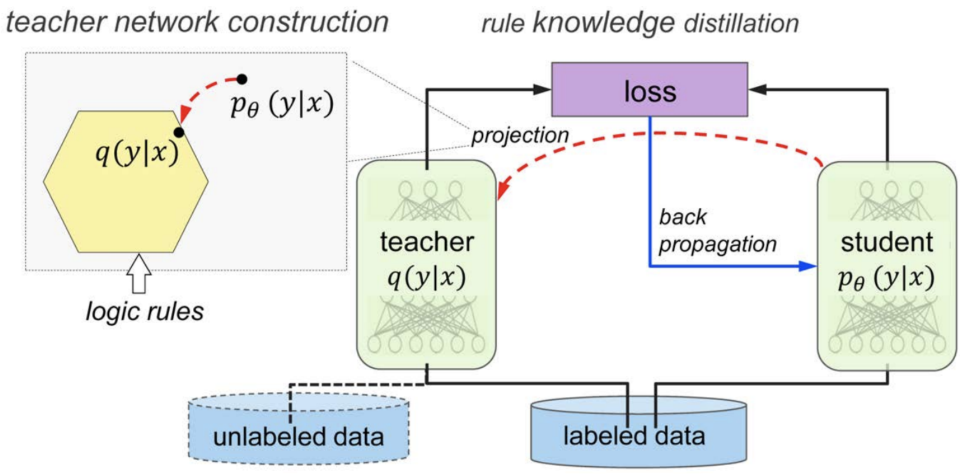

Figure 8: Graphical illustration of knowledge distillation.

There are a few catches with this approach:

1. The rule function f required a few properties: many occasions required the functions to be differentiable.
2. How to expand the input space beyond the logical rules is interesting. For example, if we phrase the reward function in reinforcement learning as a teacher model, then reinforcement learning is an instance of our approach as well.
3. The architecture could be modified and this approach could be extended to different domains of tasks as well. For instance, we could involve LSTM and solve language problems.

References

[BDS19] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

[GAA+17] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017.

[GGT+10] Kuzman Ganchev, Jennifer Gillenwater, Ben Taskar, et al. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11(Jul):2001–2049, 2010.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[KALL18] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

[KD18] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018.

[RM15] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.