Learning Implicit Generative Models by Matching Perceptual Features[1]

May 14, 2020

[1] "Learning Implicit Generative Models by Matching Perceptual Features" by Santos et al.

Overview

- Authors train a "non-standard" generative model by matching perceptual features:
  1. Define a generator $G(z)$ with $z \sim \mathcal{N}(0, I)$
  2. Take an ImageNet-pretrained classifier $E(x)$
  3. Compute data mean μ_{data} and covariance Σ_{data}
  4. Optimize $G(z)$ in such a way that generated mean and covariance is close to real ones
- They propose ADAM moving average of the moments to make the statistics more robust
- They perform some theoretical exploration of the approach
- Their model achieves quite good results on CIFAR-10 and STL-10

Maximum Mean Discrepancy (informally)

- Imagine, that we have a kernel function $k(x, x)$ for $x \in \mathcal{X}$. We define *Reproducing Kernel Hilbert Space (RKHS)* $\mathcal{H}_k$: a set of all real-valued functions on $\mathcal{X}$ (considering that $\mathcal{X}$ is a "good" space).
- $\mathcal{H}_k$ can have an inner product, for example:

$$\langle f, g \rangle_{\mathcal{H}_k} = \int_X f(x)\overline{g(x)}d\mu(x)$$

- This allows us to compute norms and distances between functions.
- *Maximum Mean Discrepancy (MMD)* is defined between 2 probability distributions p and q as:

$$\text{MMD}_k(p, q) = \|\mu_p - \mu_q\|_{\mathcal{H}_k},$$

where $\mu_p = \underset{p(x)}{\mathbb{E}}[k_x(.)]$ is the expectation of functions $k_x(.) = k(x, .)$.

MMD and generative modeling

- If the kernel $k(x, x)$ is *universal* (approximates functions well), then $\text{MMD}_k(p, q) = 0$ iff $p = q$.
- In theory, this gives us a way to build a generative model: just minimize $\text{MMD}_k(p_{\text{data}}, q_\theta)$ where q_θ is your neural network
- In practice, this gives us nothing: μ_p and μ_q are infinite-dimensional so there is no practical way we can optimize them directly.
- Luckily, we can rewrite MMD using the kernel:

$$\begin{aligned}\|\mu_p - \mu_q\|_{\mathcal{H}}^2 &= \langle \mu_p - \mu_q, \mu_p - \mu_q \rangle \\ &= \mathop{\mathbb{E}}_{p,p}\left[\langle k_x, k_{x'} \rangle\right] - 2\mathop{\mathbb{E}}_{p,q}\left[\langle k_x, k_y \rangle\right] + \mathop{\mathbb{E}}_{q,q}\left[\langle k_y, k_{y'} \rangle\right] \\ &= \mathop{\mathbb{E}}_{p,p}\left[k(x, x')\right] - 2\mathop{\mathbb{E}}_{p,q}\left[k(x, y)\right] + \mathop{\mathbb{E}}_{q,q}\left[k(y, y')\right]\end{aligned}$$

- But this is not a way authors build their model

MMD and feature maps

- One can define a kernel by using *feature maps*: some functions $\phi_j : \mathcal{X} \to \mathbb{R}$ which takes an image x and produce some scalar
- Several feature maps $\phi_1, ..., \phi_n, ...$ give us embedding $\Phi(x)$
- Now we define kernel $K_\phi(x, y) = \langle \Phi(x), \Phi(y) \rangle$
- If the set of feature maps is universal (approximates any function well), then the resulted kernel is universal as well.
- ImageNet-pretrained encoder E which outputs an embedding of size 512 can be seen as a set of 512 individual feature maps $\phi_1, ..., \phi_{512}$
- And it gives us a way to minimize $\text{MMD}_{K_\Phi}(p_{\text{data}}, q_\theta)$ with:

$$\langle \Phi(x), \Phi(y) \rangle = \Phi(x)^\top \Phi(y)$$

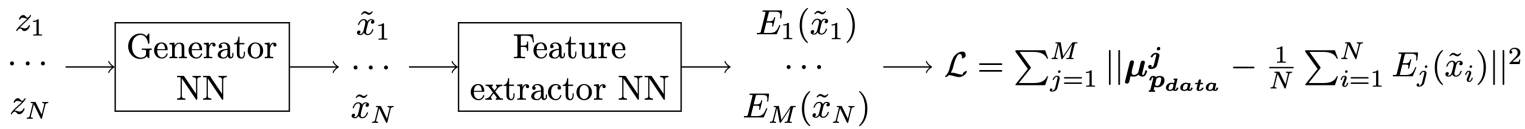
- These feature maps must be kept fixed during optimization
- As far as I understand, feature maps for a neural network cannot be universal...

Generative Feature Matching Network (GFMN)

- Authors propose to optimize MMD using ImageNet-pretrained encoder as a feature embedder $\Phi(x)$.
- In this case MMD is formulated as:

$$\mathrm{MMD}^2_{K_\Phi}(p, q) = \| \mathop{\mathbb{E}}_{x \sim p} [\Phi(x)] - \mathop{\mathbb{E}}_{x \sim q} [\Phi(x)] \|^2 \tag{1}$$

- Which is basically matching mean features between real and fake distributions:



- To theoretically justify matching covariances one need to introduce new feature maps of the kind $\phi_i(x)\phi_j(x)$

GFMN: what happens in practice

In practice, authors optimize the following loss:

$$\min_{\theta} \sum_{j=1}^{M} \left\| \mu^j_{p_{\text{data}}} - \mu^j_{p_G}(\theta) \right\|^2 + \left\| \sigma^j_{p_{\text{data}}} - \sigma^j_{p_G}(\theta) \right\|^2, \tag{2}$$

where M is the number of layers we gather the features from.
Here σ is a diagonal covariance since matching full covariances is too expensive:

$$\begin{aligned} \sigma^j_{p_{\text{data}},\ell} &= \mathop{\mathbb{E}}_{x \sim p_{\text{data}}} \left[E_{j,\ell}(x)^2 \right] - \left(\mu^{j,\ell}_{p_{\text{data}}} \right)^2, \ell = 1 \dots d_j \\ \sigma^j_{p_G,\ell}(\theta) &= \mathop{\mathbb{E}}_{z} \left[E_{j,\ell}(G(z;\theta))^2 \right] - \left(\mu^{j,\ell}_{p_G} \right)^2, \ell = 1 \dots d_j \end{aligned} \tag{3}$$

ADAM Moving Average (AMA)

- We can rewrite MMD for means as:

$$\|\mu_p - \mu_q\|^2 = (\mu_p - \mu_q)^\top (\mu_p - \mu_q) = \Delta^\top (\mu_p - \mu_q)$$

- Where $\mu_q = \mathop{\mathbb{E}}_{z}[E(G(z)]$, and we can approximate it on a minibatch of size N: $\hat{\mu}_q = \frac{1}{N} \sum_i E(G(z_i))$
- To make the estimate more robust authors propose to use a moving average of Δ:

$$\|\mu_p - \mu_q\|^2 \approx v^\top (\mu_p - \mu_q)$$

  where v is a moving average of Δ:

$$v_{\text{new}} = (1 - \alpha) v_{\text{old}} + \alpha \Delta$$

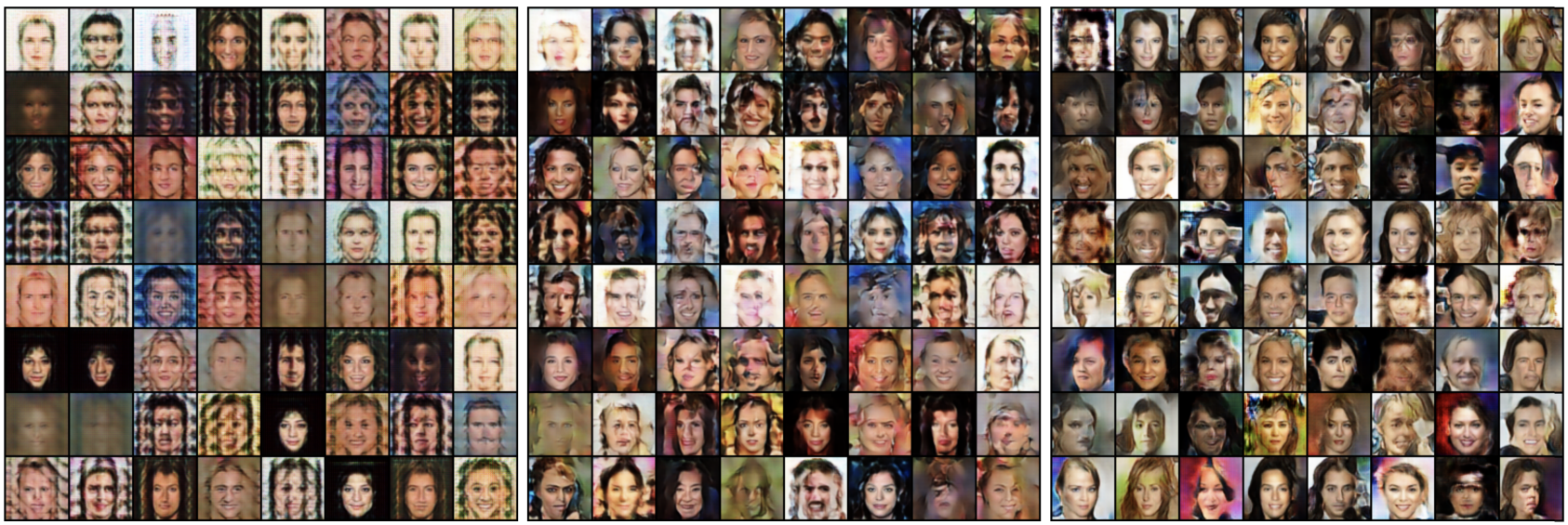
- This formula is equivalent to a gradient step towards minimizing

$$L(v) = \frac{1}{2} \|v - \Delta_j\|^2$$

- Authors propose to update v by minimizing $L(v)$ with Adam optimizer.

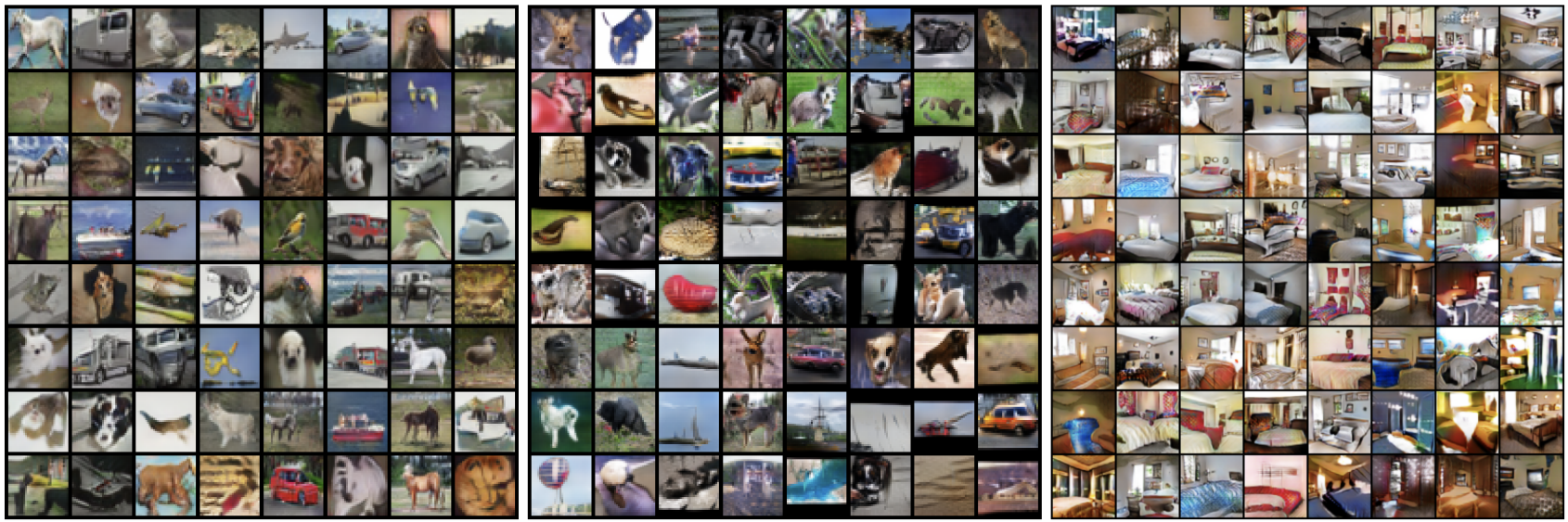
Using AMA turend out to be very important

Apparently, even batch size of 512 is not enough to compute a good estimate:



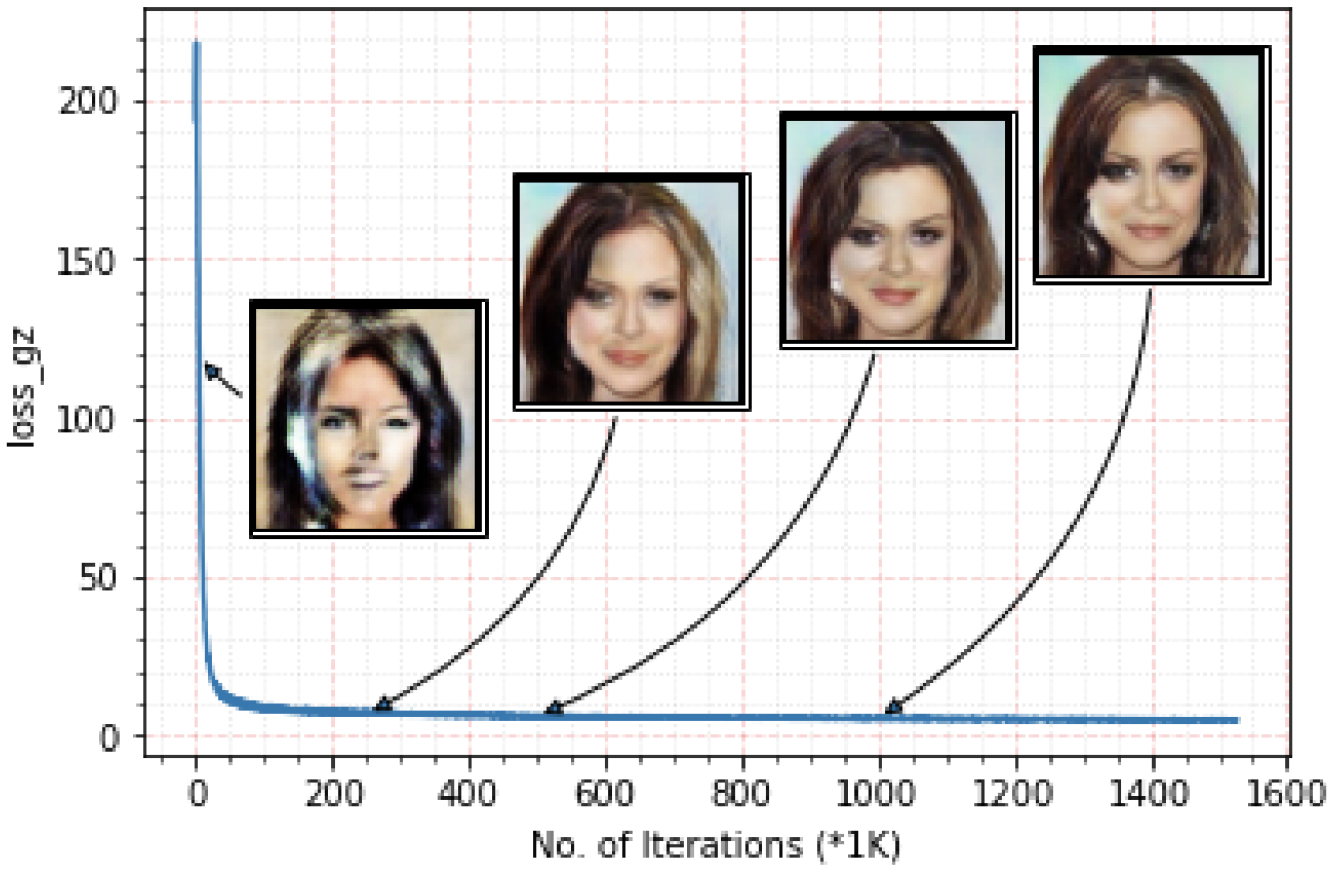
(a) MA - mbs 64 (b) MA - mbs 512 (c) AMA - mbs 64

Results



(a) CIFAR10 (b) STL10 (c) LSUN

Loss function value is correlated well with sample quality



Quantitive results

Table 4: Inception Score and FID of different generative models for CIFAR10 and STL10.

Model	**CIFAR 10**		**STL 10**	
	IS	**FID (5K / 50K)**	**IS**	**FID (5K / 50K)**
Real data	11.24±.12	7.8 / 3.2	26.08±.26	8.08 / 4.0
No Adversarial Training				
GMMN [21]	3.47±.03			
GMMN+AE [21]	3.94±.04			
(ours) GFMN$^{\text{VGG+Resnet}}$	8.08 ± 0.08	25.5 / 20.9	8.57 ± 0.08	34.2 / 17.2
(ours) GFMN$^{\text{VGG+Resnet}}$ (Resnet G)	**8.27 ± 0.09**	**18.1 / 13.5**	**9.12 ± 0.09**	**31.6 / 13.9**
Adversarial Training & Online Moment Learning Methods (Unsupervised)				
MMD GAN [21]	6.17±.07			
MMD$_{rq}$ GAN [3]	6.51±.03	39.9 / -		
WGAN-GP [27]	6.68±.06	40.2 / -	8.42±.13	55.1 / -
SN-GANs [27]	7.58±.12	25.5 / -	8.79±.14	43.2 / -
MoLM-1024 [33]	7.55±.08	25.0 / 20.3		
GAN-DFM [39]	7.72±.13			
MoLM-1536 [33]	7.90±.10	23.3 / 18.9		
Adversarial Training (Supervised)				
Impr. GAN [36]	8.09±.07			
FisherGAN (Resnet G) [28]	8.16±.12			
WGAN-GP (Resnet G) [13]	8.42±.10			

Conclusion

- Benefit 1: the loss function is directly correlated with the generated image quality
- Benefit 2: no mode collapse
- Benefit 3: one can use several encoders to improve scores and this encoders are cross-domain
- In some way their objective is quite close to FID, so in some sense they optimize a final evaluation metric directly