KDGAN: Knowledge Distillation with Generative Adversarial Networks

Xiaojie Wang
University of Melbourne
xiaojiew94@gmail.com

Rui Zhang*
University of Melbourne
rui.zhang@unimelb.edu.au

Yu Sun
Twitter Inc.
ysun@twitter.com

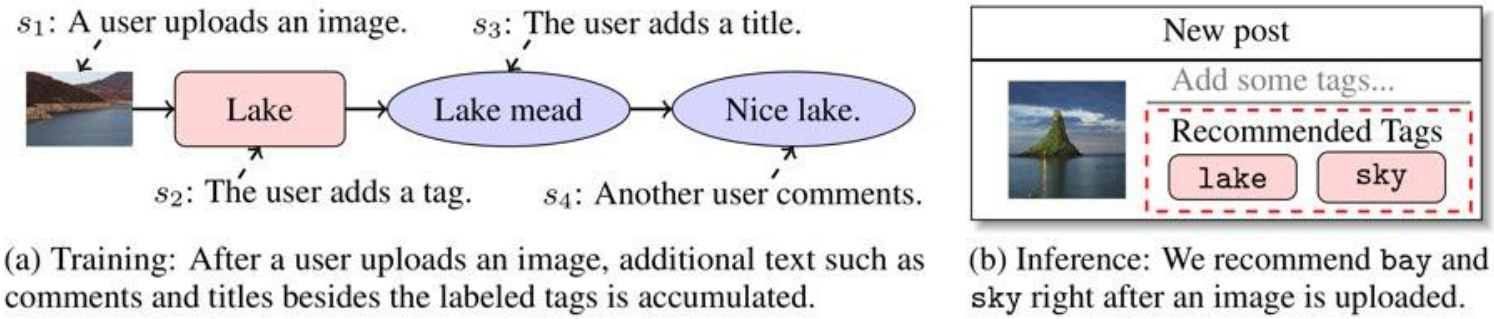
Jianzhong Qi
University of Melbourne
jianzhong.qi@unimelb.edu.au

김 성 철

Contents

Introduction

- **Terminologies**
  - ***Privileged** provision* : input features or computational resources → **Resources are only accessible for training!**
  - ***Privileged** information* : input features
- **Example about <u>the privileged provision</u> between training and inference**
  - Image tag recommendation / face recognition



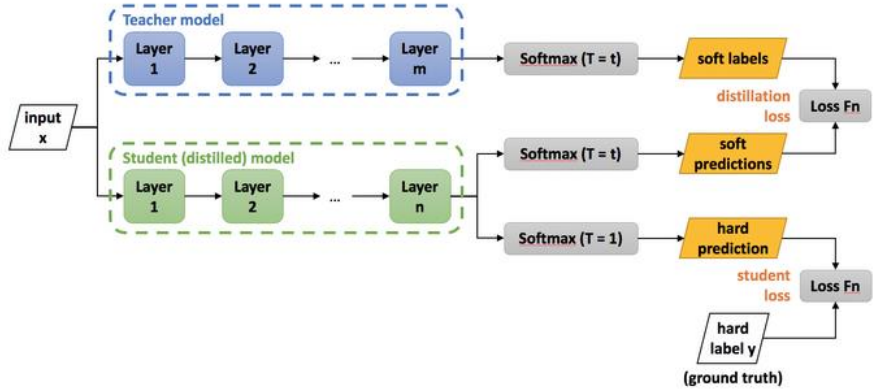
(a) Training: After a user uploads an image, additional text such as comments and titles besides the labeled tags is accumulated.

(b) Inference: We recommend `bay` and `sky` right after an image is uploaded.

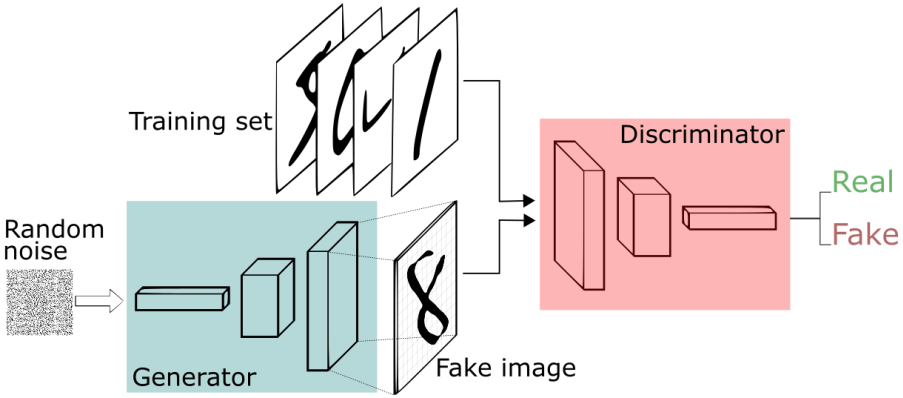
Figure 1: Image tag recommendation where the additional text is only available for training.

Introduction

- **Knowledge Distillation (KD)**
  - Classifier (student) / Teacher



- **Generative Adversarial Networks (GAN)**
  - Classifier (Generator) / Discriminator



https://light-tree.tistory.com/196
https://pathmind.com/kr/wiki/generative-adversarial-network-gan

Introduction

- **Knowledge Distillation with Generative Adversarial Networks（KDGAN)**
  - KD to train a lightweight classifier.
  - GAN to teach the classifier the true data distribution.

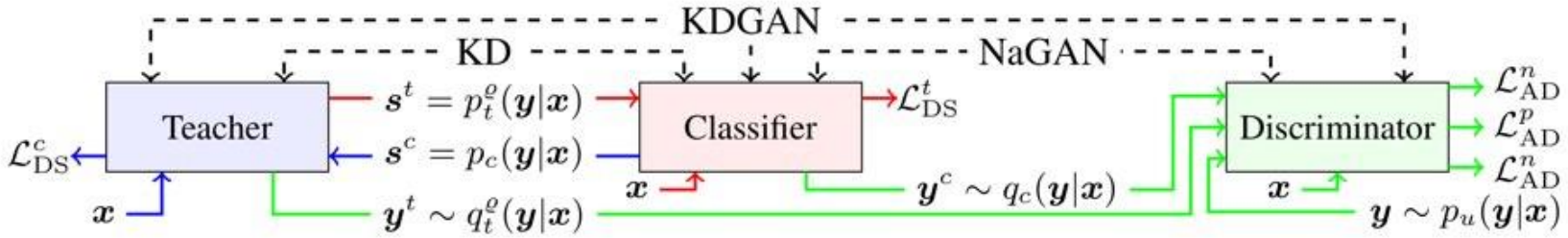


Figure 2: Comparison among KD, NaGAN, and KDGAN. The classifier (C) and the teacher (T) learn discrete categorical distributions $p_c(\boldsymbol{y}|\boldsymbol{x})$ and $p_t^\varrho(\boldsymbol{y}|\boldsymbol{x})$; $\boldsymbol{y}$ is a true label generated from the true data distribution $p_u(\boldsymbol{y}|\boldsymbol{x})$; $\boldsymbol{y}^c$ and $\boldsymbol{y}^t$ are continuous samples generated from concrete distributions $q_c(\boldsymbol{y}|\boldsymbol{x})$ and $q_t^\varrho(\boldsymbol{y}|\boldsymbol{x})$; $\boldsymbol{s}^c$ and $\boldsymbol{s}^t$ are soft labels produced by C and T; $\mathcal{L}_{\mathrm{DS}}^c$ and $\mathcal{L}_{\mathrm{DS}}^t$ are distillation losses for C and T; $\mathcal{L}_{\mathrm{AD}}^p$ and $\mathcal{L}_{\mathrm{AD}}^n$ are adversarial losses for positive and negative feature-label pairs.

Methods

- **Naïve GAN (NaGAN)**

$$\min_{c} \max_{d} V(c,d) = \mathbb{E}_{\boldsymbol{y}\sim p_u}[\log p_d^{\varrho}(\boldsymbol{x},\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{y}\sim p_c}[\log(1 - p_d^{\varrho}(\boldsymbol{x},\boldsymbol{y}))]. \quad (1)$$

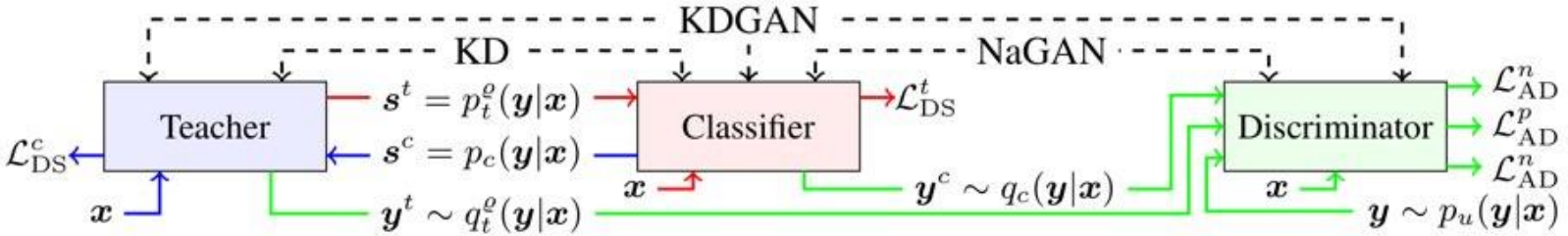
Let $h(\boldsymbol{x},\boldsymbol{y})$ and $g(\boldsymbol{x},\boldsymbol{y})$ be the scoring functions for C and D. We define $p_c(\boldsymbol{y}|\boldsymbol{x})$ and $p_d^{\varrho}(\boldsymbol{x},\boldsymbol{y})$ as

$$p_c(\boldsymbol{y}|\boldsymbol{x}) = \mathrm{softmax}(h(\boldsymbol{x},\boldsymbol{y})) \quad \text{and} \quad p_d^{\varrho}(\boldsymbol{x},\boldsymbol{y}) = \mathrm{sigmoid}(g(\boldsymbol{x},\boldsymbol{y})). \quad (2)$$

- **The advantages and disadvantages of KD and NaGAN**
  - KD : requires a small number of training instances and epochs, but ~.
  - NaGAN : ensures the equilibrium where $p_c(\boldsymbol{y}|\boldsymbol{x}) = p_u(\boldsymbol{y}|\boldsymbol{x})$, but ~.

Methods

- **KDGAN**
  - a classifier C, a teacher T, a discriminator D with privileged provision ϱ
  - D : maximize the probability of correctly distinguishing the true and pseudo labels.
  - C, T : minimize the probability that D rejects their generated pseudo labels.
  - C : learn from T by mimicking the learned distribution of T.
  - **T also learn from C, because a teacher's ability can also be enhanced by interacting with students. → reduce the probability of generating different pseudo labels.**



Methods

- KDGAN

Algorithm 1: Minibatch stochastic gradient descent training of KDGAN.

Pretrain a classifier C, a teacher T, and a discriminator D with the training data $\{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$.
for *the number of training epochs* **do**
  for *the number of training steps for the discriminator* **do**
    Sample labels $\{\boldsymbol{y}_1, ..., \boldsymbol{y}_k\}$, $\{\boldsymbol{y}_1^c, ..., \boldsymbol{y}_k^c\}$, and $\{\boldsymbol{y}_1^t, ..., \boldsymbol{y}_k^t\}$ from $p_u(\boldsymbol{y}|\boldsymbol{x})$, $q_c(\boldsymbol{y}|\boldsymbol{x})$, and $q_t^\varrho(\boldsymbol{y}|\boldsymbol{x})$.
    Update D by ascending along its gradients
    $\frac{1}{k}\sum_{i=1}^{k}\big(\nabla_d \log p_d^\varrho(\boldsymbol{x}, \boldsymbol{y}_i) + \alpha\nabla_d \log(1 - p_d^\varrho(\boldsymbol{x}, \boldsymbol{z}_i^c)) + (1-\alpha)\nabla_d \log(1 - p_d^\varrho(\boldsymbol{x}, \boldsymbol{z}_i^t))\big)$.
  for *the number of training steps for the teacher* **do**
    Sample labels $\{\boldsymbol{y}_1^t, ..., \boldsymbol{y}_k^t\}$ from $q_t^\varrho(\boldsymbol{y}|\boldsymbol{x})$ and update the teacher by descending along its gradients
    $\frac{1}{k}\sum_{i=1}^{k}(1-\alpha)\nabla_t \log q_t^\varrho(\boldsymbol{y}_i^t|\boldsymbol{x}) \log(1 - p_d^\varrho(\boldsymbol{x}, \boldsymbol{z}_i^t)) + \gamma\nabla_t \mathcal{L}_{\mathrm{DS}}^t(p_t^\varrho(\boldsymbol{y}|\boldsymbol{x}), p_c(\boldsymbol{y}|\boldsymbol{x}))$.
  for *the number of training steps for the classifier* **do**
    Sample labels $\{\boldsymbol{y}_1^c, ..., \boldsymbol{y}_k^c\}$ from $q_c(\boldsymbol{y}|\boldsymbol{x})$ and update C by descending along its gradients
    $\frac{1}{k}\sum_{i=1}^{k}\alpha\nabla_c \log q_c(\boldsymbol{y}_i^c|\boldsymbol{x}) \log(1 - p_d^\varrho(\boldsymbol{x}, \boldsymbol{z}_i^c)) + \beta\nabla_c \mathcal{L}_{\mathrm{DS}}^c(p_c(\boldsymbol{y}|\boldsymbol{x}), p_t^\varrho(\boldsymbol{y}|\boldsymbol{x}))$.

Methods

- **KDGAN**
  - The value function $U(c,t,d)$ $(\alpha \in (0,1), \beta \in (0,+\infty), \gamma \in (0,+\infty))$

$$\min_{c,t}\max_{d} U(c,t,d) = \mathbb{E}_{\boldsymbol{y}\sim p_u}[\log p_d^{\varrho}(\boldsymbol{x},\boldsymbol{y})] + \alpha\mathbb{E}_{\boldsymbol{y}\sim p_c}[\log(1-p_d^{\varrho}(\boldsymbol{x},\boldsymbol{y}))] + (1-\alpha)\mathbb{E}_{\boldsymbol{y}\sim p_t^{\varrho}}[\log(1-p_d^{\varrho}(\boldsymbol{x},\boldsymbol{y}))] + \beta\mathcal{L}_{\mathrm{DS}}^{c}(p_c(\boldsymbol{y}|\boldsymbol{x}), p_t^{\varrho}(\boldsymbol{y}|\boldsymbol{x})) + \gamma\mathcal{L}_{\mathrm{DS}}^{t}(p_t^{\varrho}(\boldsymbol{y}|\boldsymbol{x}), p_c(\boldsymbol{y}|\boldsymbol{x})), \quad (3)$$

Adversarial losses **Distillation losses**

    - Distillation losses : L2 loss, Kullback-Leibler divergence, …
    - $\mathcal{L}_{DS}^{c}, \mathcal{L}_{\mathrm{DS}}^{\mathrm{t}}$ are used to train the classifier and the teacher, respectively.

Methods

- **KDGAN**
  - The classifier perfectly learns the true data distribution.

    → $p_\alpha^\varrho(\boldsymbol{y}|\boldsymbol{x}) = \alpha p_c(\boldsymbol{y}|\boldsymbol{x}) + (1-\alpha)p_t^\varrho(\boldsymbol{y}|\boldsymbol{x})$

$$
\begin{aligned}
&\alpha\mathbb{E}_{\boldsymbol{y}\sim p_c}[\log(1-p_d^\varrho(\boldsymbol{x},\boldsymbol{y}))] + (1-\alpha)\mathbb{E}_{\boldsymbol{y}\sim p_t^\varrho}[\log(1-p_d^\varrho(\boldsymbol{x},\boldsymbol{y}))] \\
&= \alpha\textstyle\sum_{\boldsymbol{y}} p_c(\boldsymbol{y}|\boldsymbol{x})\log(1-p_d^\varrho(\boldsymbol{x},\boldsymbol{y})) + (1-\alpha)\sum_{\boldsymbol{y}} p_t^\varrho(\boldsymbol{y}|\boldsymbol{x})\log(1-p_d^\varrho(\boldsymbol{x},\boldsymbol{y})) \\
&= \textstyle\sum_{\boldsymbol{y}} \left(\alpha p_c(\boldsymbol{y}|\boldsymbol{x}) + (1-\alpha)p_t^\varrho(\boldsymbol{y}|\boldsymbol{x})\right)\log(1-p_d^\varrho(\boldsymbol{x},\boldsymbol{y})) \\
&= \mathbb{E}_{\boldsymbol{y}\sim p_\alpha^\varrho}[\log(1-p_d^\varrho(\boldsymbol{x},\boldsymbol{y}))].
\end{aligned}
\tag{4}
$$

$$
\begin{aligned}
&= \sum\nolimits_{\boldsymbol{y}} p_u(\boldsymbol{y}|\boldsymbol{x})\log p_d^\varrho(\boldsymbol{x},\boldsymbol{y}) + \sum\nolimits_{\boldsymbol{y}} p_\alpha^\varrho(\boldsymbol{y}|\boldsymbol{x})\log(1-p_d^\varrho(\boldsymbol{x},\boldsymbol{y})) \\
&= F(p_d^\varrho(\boldsymbol{x},\boldsymbol{y})).
\end{aligned}
$$

The function $F(p_d^\varrho(\boldsymbol{x},\boldsymbol{y}))$ achieves the maximum if and only if the distribution of the discriminator is equivalent to $p_d^\varrho(\boldsymbol{x},\boldsymbol{y}) = {p_u(\boldsymbol{y}|\boldsymbol{x})}/{p_u(\boldsymbol{y}|\boldsymbol{x}) + p_\alpha^\varrho(\boldsymbol{y}|\boldsymbol{x})}$, completing the proof. □

Methods

- **KDGAN**
  - $\mathcal{L}_{MD} = \beta \mathcal{L}_{DS}^{c}\left(p_c(\boldsymbol{y}|\boldsymbol{x}), p_t^{\varrho}(\boldsymbol{y}|\boldsymbol{x})\right) + \gamma \mathcal{L}_{DS}^{t}\left(p_t^{\varrho}(\boldsymbol{y}|\boldsymbol{x}), p_c(\boldsymbol{y}|\boldsymbol{x})\right)$
  - $\mathcal{L}_{JS}$: the Jensen-Shannon divergence

    * $\mathrm{JSD}(p,q) = \frac{1}{2} D_{KL}(p||\frac{p+q}{2}) + \frac{1}{2} D_{KL}(q||\frac{p+q}{2})$

$$\min_{\alpha} \max_{d} \mathbb{E}_{\boldsymbol{y}\sim p_u}[\log p_d^{\varrho}(\boldsymbol{x},\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{y}\sim p_\alpha^{g}}[\log(1 - p_d^{\varrho}(\boldsymbol{x},\boldsymbol{y}))] + \mathcal{L}_{\mathrm{MD}} \tag{5}$$

$$= \min_{\alpha} 2\mathcal{L}_{\mathrm{JS}}(p_u(\boldsymbol{y}|\boldsymbol{x})||p_\alpha^{g}(\boldsymbol{y}|\boldsymbol{x})) + \beta \mathcal{L}_{\mathrm{DS}}^{c}(p_c(\boldsymbol{y}|\boldsymbol{x}), p_t^{\varrho}(\boldsymbol{y}|\boldsymbol{x})) + \gamma \mathcal{L}_{\mathrm{DS}}^{t}(p_t^{\varrho}(\boldsymbol{y}|\boldsymbol{x}), p_c(\boldsymbol{y}|\boldsymbol{x})) - \log(4).$$

Methods

- KDGAN

$$
\begin{aligned}
&\min_{s,t} U(c,t,d) \\
&= \sum\nolimits_{\boldsymbol{y}} p_u(\boldsymbol{y}|\boldsymbol{x}) \log \frac{p_u(\boldsymbol{y}|\boldsymbol{x})}{p_u(\boldsymbol{y}|\boldsymbol{x}) + p_\alpha^\varrho(\boldsymbol{y}|\boldsymbol{x})} + \sum\nolimits_{\boldsymbol{y}} p_\alpha^\varrho(\boldsymbol{y}|\boldsymbol{x}) \log(1 - \frac{p_u(\boldsymbol{y}|\boldsymbol{x})}{p_u(\boldsymbol{y}|\boldsymbol{x}) + p_\alpha^\varrho(\boldsymbol{y}|\boldsymbol{x})}) + \mathcal{L}_{\mathrm{MD}} \\
&= \sum\nolimits_{\boldsymbol{y}} p_u(\boldsymbol{y}|\boldsymbol{x}) \log \frac{p_u(\boldsymbol{y}|\boldsymbol{x})}{p_u(\boldsymbol{y}|\boldsymbol{x}) + p_\alpha^\varrho(\boldsymbol{y}|\boldsymbol{x})} + \sum\nolimits_{\boldsymbol{y}} p_\alpha^\varrho(\boldsymbol{y}|\boldsymbol{x}) \log \frac{p_\alpha^\varrho(\boldsymbol{y}|\boldsymbol{x})}{p_u(\boldsymbol{y}|\boldsymbol{x}) + p_\alpha^\varrho(\boldsymbol{y}|\boldsymbol{x})} + \mathcal{L}_{\mathrm{MD}} \\
&= -\log(4) + \mathcal{L}_{\mathrm{KL}}(p_u(\boldsymbol{y}|\boldsymbol{x}) || \frac{p_u(\boldsymbol{y}|\boldsymbol{x}) + p_\alpha^\varrho(\boldsymbol{y}|\boldsymbol{x})}{2}) + \mathcal{L}_{\mathrm{KL}}(p_\alpha^\varrho(\boldsymbol{y}|\boldsymbol{x}) || \frac{p_u(\boldsymbol{y}|\boldsymbol{x}) + p_\alpha^\varrho(\boldsymbol{y}|\boldsymbol{x})}{2}) + \mathcal{L}_{\mathrm{MD}} \\
&= -\log(4) + 2\mathcal{L}_{\mathrm{JS}}(p_u(\boldsymbol{y}|\boldsymbol{x}) || p_\alpha^\varrho(\boldsymbol{y}|\boldsymbol{x})) + \beta \mathcal{L}_{\mathrm{DS}}^c(p_c(\boldsymbol{y}|\boldsymbol{x}), p_t^\varrho(\boldsymbol{y}|\boldsymbol{x})) + \gamma \mathcal{L}_{\mathrm{DS}}^t(p_t^\varrho(\boldsymbol{y}|\boldsymbol{x}), p_c(\boldsymbol{y}|\boldsymbol{x})).
\end{aligned}
$$

Methods

- **KDGAN**
  - $\mathcal{L}_{JS}$ reaches the minimum if and only if $p_\alpha^\varrho(\boldsymbol{y}|\boldsymbol{x}) = p_u(\boldsymbol{y}|\boldsymbol{x})$
  - $\mathcal{L}_{DS}^c$ (or $\mathcal{L}_{DS}^t$) reaches the minimum if and only if $p_c(\boldsymbol{y}|\boldsymbol{x}) = p_t^\varrho(\boldsymbol{y}|\boldsymbol{x})$

    →KDGAN equilibrium

    $\boldsymbol{p_c(y|x) = p_t^\varrho(y|x) = p_u(y|x)}$ **(C learns the true data distribution.)**

$$\min_\alpha \max_d \mathbb{E}_{\boldsymbol{y}\sim p_u}[\log p_d^\varrho(\boldsymbol{x},\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{y}\sim p_\alpha^\varrho}[\log(1 - p_d^\varrho(\boldsymbol{x},\boldsymbol{y}))] + \mathcal{L}_{\mathrm{MD}} \tag{5}$$
$$= \min_\alpha 2\mathcal{L}_{\mathrm{JS}}(p_u(\boldsymbol{y}|\boldsymbol{x})||p_\alpha^\varrho(\boldsymbol{y}|\boldsymbol{x})) + \beta\mathcal{L}_{\mathrm{DS}}^c(p_c(\boldsymbol{y}|\boldsymbol{x}), p_t^\varrho(\boldsymbol{y}|\boldsymbol{x})) + \gamma\mathcal{L}_{\mathrm{DS}}^t(p_t^\varrho(\boldsymbol{y}|\boldsymbol{x}), p_c(\boldsymbol{y}|\boldsymbol{x})) - \log(4).$$

Methods

- **KDGAN Training**
  - The training speed is closely related to the variance of gradients.
  - The KDGAN framework by design

    → can **reduce the variance of gradients**!!

    → **use a continuous space** by relaxing the discrete samples!!

Methods

- **KDGAN Training**

First, we show how KDGAN reduces the variance of gradients. As discussed above, C only receives gradients $\nabla_c V$ from D in NaGAN while it receives gradients $\nabla_c U$ from both D and T in KDGAN:

$$\nabla_c V = \nabla_c \mathcal{L}^n_{\mathrm{AD}}, \quad \nabla_c U = \lambda \nabla_c \mathcal{L}^n_{\mathrm{AD}} + (1-\lambda)\nabla_c \mathcal{L}^c_{\mathrm{DS}}, \tag{6}$$

where $\lambda \in (0,1)$, $\nabla_c \mathcal{L}^n_{\mathrm{AD}}$ and $\nabla_c \mathcal{L}^c_{\mathrm{DS}}$ are gradients from D and T, respectively. Consistent with the findings in existing work [23, 39], we also observe that $\nabla_c \mathcal{L}^c_{\mathrm{DS}}$ usually has a lower variance than $\nabla_c \mathcal{L}^n_{\mathrm{AD}}$ (see Figure 7 in Appendix D for empirical evidence that the variance of $\nabla_c \mathcal{L}^c_{\mathrm{DS}}$ is smaller than that of $\nabla_c \mathcal{L}^n_{\mathrm{AD}}$ during the training process). Hence, it can be easily shown that the gradients w.r.t. C in KDGAN have a lower variance than that in NaGAN (refer to Lemma 4.3):

$$\mathrm{Var}(\nabla_c \mathcal{L}^c_{\mathrm{DS}}) \leq \mathrm{Var}(\nabla_c \mathcal{L}^n_{\mathrm{AD}}) \Rightarrow \mathrm{Var}(\nabla_c U) \leq \mathrm{Var}(\nabla_c V). \tag{7}$$

Proof. Given $\mathrm{Var}(X) \leq \mathrm{Var}(Y)$, the covariance $\mathrm{Cov}(X,Y)$ is less than or equal to $\mathrm{Var}(Y)$ because

$$\mathrm{Cov}(X,Y) \leq |\mathrm{Cov}(X,Y)| \leq \sqrt{\mathrm{Var}(X)\,\mathrm{Var}(Y)} \leq \sqrt{\mathrm{Var}(Y)\,\mathrm{Var}(Y)} \leq \mathrm{Var}(Y).$$

According to the properties of the variance, for all $\lambda \in (0,1)$, we have

$$\begin{aligned}
\mathrm{Var}(Z) &= \lambda^2 \mathrm{Var}(X) + 2\lambda(1-\lambda)\,\mathrm{Cov}(X,Y) + (1-\lambda)^2 \mathrm{Var}(Y) \\
&\leq \lambda^2 \mathrm{Var}(Y) + 2\lambda(1-\lambda)\,\mathrm{Cov}(X,Y) + (1-\lambda)^2 \mathrm{Var}(Y) \\
&\leq \lambda^2 \mathrm{Var}(Y) + 2\lambda(1-\lambda)\,\mathrm{Var}(Y) + (1-\lambda)^2 \mathrm{Var}(Y) \\
&= \mathrm{Var}(Y),
\end{aligned}$$

Methods

- KDGAN Training

Next, we further reduce the variance of gradients with a reparameterization trick, in particular, the Gumbel-Max trick [20, 30]. The essence of the Gumbel-Max trick is to reparameterize generating discrete samples into a differentiable function of its parameters and an additional random variable of a Gumbel distribution. To perform the Gumbel-Max trick on generating discrete samples from the categorical distribution $p_c(\boldsymbol{y}|\boldsymbol{x})$, a concrete distribution [25, 31] can be used. We use a concrete distribution $q_c(\boldsymbol{y}|\boldsymbol{x})$ to generate continuous samples and use the continuous samples to compute the gradients $\nabla_c \mathcal{L}_{\mathrm{AD}}^n$ of the adversarial loss w.r.t. the classifier as

$$\nabla_c \mathcal{L}_{\mathrm{AD}}^n = \nabla_c \mathbb{E}_{\boldsymbol{y}\sim p_c}[\log(1 - p_d^\varrho(\boldsymbol{x}, \boldsymbol{y}))] = \mathbb{E}_{\boldsymbol{y}\sim q_c}[\nabla_c \log q_c(\boldsymbol{y}|\boldsymbol{x}) \log(1 - p_d^\varrho(\boldsymbol{x}, \boldsymbol{z}))]. \quad (8)$$

Here, $\boldsymbol{z} = \mathrm{onehot}(\mathrm{argmax}\, \boldsymbol{y})$ is a discrete pseudo label where $\boldsymbol{y} \sim q_c(\boldsymbol{y}|\boldsymbol{x})$. We define $q_c(\boldsymbol{y}|\boldsymbol{x})$ as

$$q_c(\boldsymbol{y}|\boldsymbol{x}) = \mathrm{softmax}\left(\frac{\log p_c(\boldsymbol{y}|\boldsymbol{x}) + \boldsymbol{g}}{\tau}\right), \quad \boldsymbol{g} \sim \mathrm{Gumbel}(0, 1). \quad (9)$$

Here, $\tau \in (0, +\infty)$ is a temperature parameter and $\mathrm{Gumbel}(0, 1)$ is the Gumbel distribution[2] [31]. We leverage the temperature parameter τ to control the variance of gradients over the training. With a high temperature, the samples from the concrete distribution are smooth, which give low-variance gradient estimates. Note that a disadvantage of the concrete distribution is that with a high temperature, it becomes a less accurate approximation to the original categorical distribution, which causes biased gradient estimates. We will discuss how to tune the temperature parameter in Section 4.

Methods

- **KDGAN Training**
  - Gumbel-Max trick

Gumbel-max trick(stochastic을 따로 뺌)

categorical한 variable을 reparametrization함. 요걸 쓰면 categorical에서 sample한 것과 비슷한 효과를 낸다고한다.

$x \sim \mathcal{C}at(\pi_\phi)$를 discrete categorical variable이라 해보자.

$\epsilon_k \sim \mathcal{G}umbel(0,1)$를 가지고 Reparametrization하면

$$x = \arg\max_k(\epsilon_k + \log \pi_k) \triangleq g(\phi, \epsilon)$$

로 쓸 수 있다. 요것이 Gumbel-max trick

- 하지만 아직도 ϕ에 대해 discrete하며 미분이 불가능하다.

Gumbel-softmax(continuous)

Gumbel-max의 argmax를 softmax로 바꾼 녀석. 이제 미분이 가능해진다.

$$(x_k)_{1\le k\le K} = \mathbf{softmax}((\epsilon_k + \log \pi_k)_k) \Leftrightarrow x_k = \frac{\exp((\log \pi_k + \epsilon_k)/\tau)}{\sum_j \exp((\log \pi_j + \epsilon_j)/\tau)}$$

특성들

- 여기서 τ는 temporature parameter인데
  - 0에 가까워지면 one-hot
  - 무한대로가면 uniform distribution을 갖는다.
- $p(x_k = \max_i x_i) = \pi_k$

https://hulk89.github.io/machine%20learning/2017/11/20/reparametrization-trick/
http://openresearch.ai/t/categorical-reparameterization-with-gumbel-softmax/386

Results

- **Experiments**
  - α in [0.0, 1.0]
  - β in [0.001, 1000]
  - γ in [0.0001, 100]
  - The temperature parameter τ

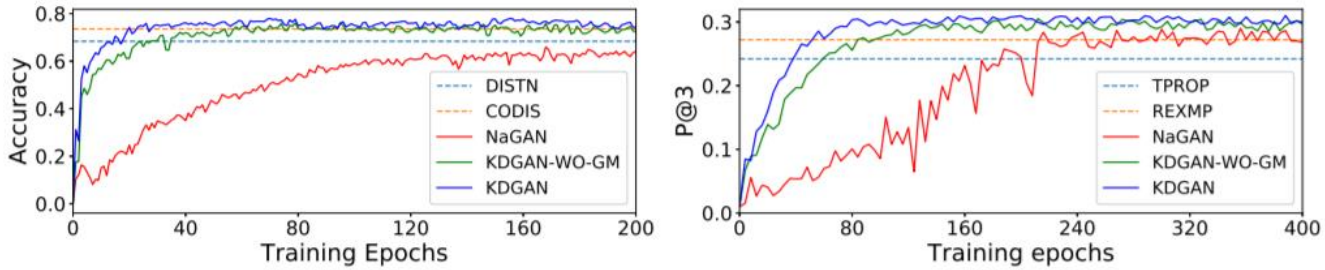
    → start with a large value (1.0) and exponentially decay it to a small value (0.1)

Results

- Deep Model Compression

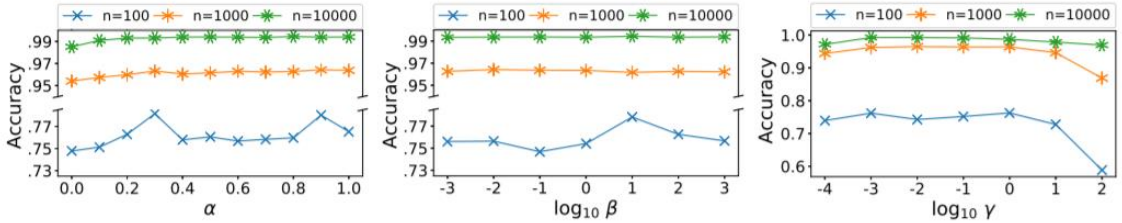
Table 1: Average accuracy over 10 runs in model compression (n is the number of training instances).

Method	MNIST			CIFAR-10		
	$n = 100$	$n = 1,000$	$n = 10,000$	$n = 500$	$n = 5,000$	$n = 50,000$
CODIS	74.02 ± 0.13	95.77 ± 0.10	98.89 ± 0.08	54.17 ± 0.20	77.82 ± 0.14	85.12 ± 0.11
DISTN	68.34 ± 0.06	93.97 ± 0.08	98.79 ± 0.07	50.92 ± 0.18	76.59 ± 0.15	83.32 ± 0.08
NOISY	66.53 ± 0.18	93.45 ± 0.11	98.58 ± 0.11	50.18 ± 0.28	75.42 ± 0.19	82.99 ± 0.12
MIMIC	67.35 ± 0.15	93.78 ± 0.13	98.65 ± 0.05	51.74 ± 0.23	75.66 ± 0.17	84.33 ± 0.10
NaGAN	64.90 ± 0.31	93.60 ± 0.22	98.95 ± 0.19	46.29 ± 0.32	76.11 ± 0.24	85.34 ± 0.27
KDGAN	$\mathbf{77.95} \pm 0.05$	$\mathbf{96.42} \pm 0.05$	$\mathbf{99.25} \pm 0.02$	$\mathbf{57.56} \pm 0.13$	$\mathbf{79.36} \pm 0.04$	$\mathbf{86.50} \pm 0.04$



(a) Deep model compression over MNIST. (b) Image tag recommendation on YFCC100M.

Figure 3: Training curves of the classifier in the proposed NaGAN and KDGAN.



(a) Effect of varying α (b) Effect of varying β (c) Effect of varying γ

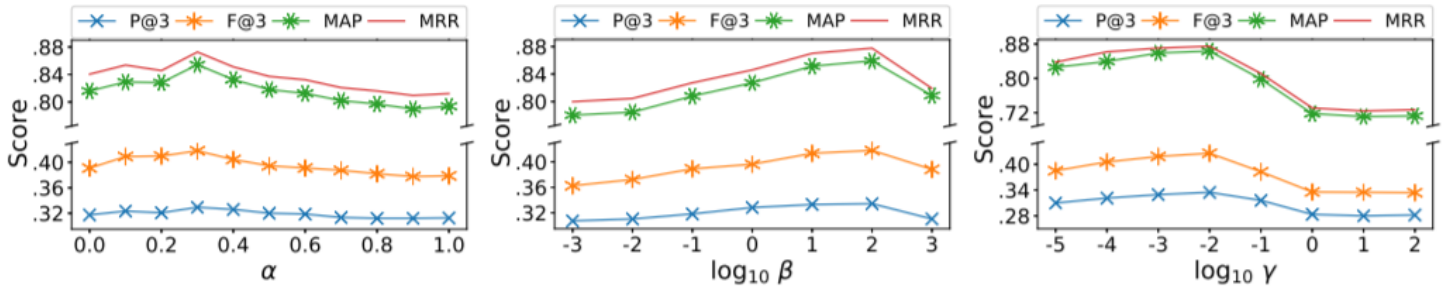
Figure 4: Effects of hyperparameters in KDGAN on MNIST for deep model compression.

Results

- Image Tag Recommendation

Table 2: Performance of various methods on the YFCC100M dataset in tag recommendation.

Method	Most Popular Tags						Randomly Sampled Tags					
	P@3	P@5	F@3	F@5	MAP	MRR	P@3	P@5	F@3	F@5	MAP	MRR
KNN	.2320	.1680	.2339	.1633	.5755	.5852	.1623	.1198	.1575	.1088	.3970	.4092
TPROP	.2420	.1636	.2811	.1949	.6177	.6270	.1883	.1372	.1810	.1252	.4512	.4636
TFEAT	.2560	.1752	.2871	.1999	.6417	.6503	.2002	.1420	.2195	.1495	.5149	.5309
REXMP	.2720	.1800	.3324	.2295	.7015	.7122	.2228	.1378	.2427	.1669	.5205	.5331
NaGAN	.2892	.1880	.3516	.2352	.7432	.7555	.2415	.1495	.2693	.1867	.5791	.5911
KDGAN	**.3047**	**.1968**	**.3678**	**.2526**	**.7787**	**.7905**	**.2572**	**.1666**	**.2946**	**.2009**	**.6302**	**.6452**



(a) Effect of varying α (b) Effect of varying β (c) Effect of varying γ

Figure 5: Effects of hyperparameters in KDGAN on YFCC100M for image tag recommendation.

감 사 합 니 다