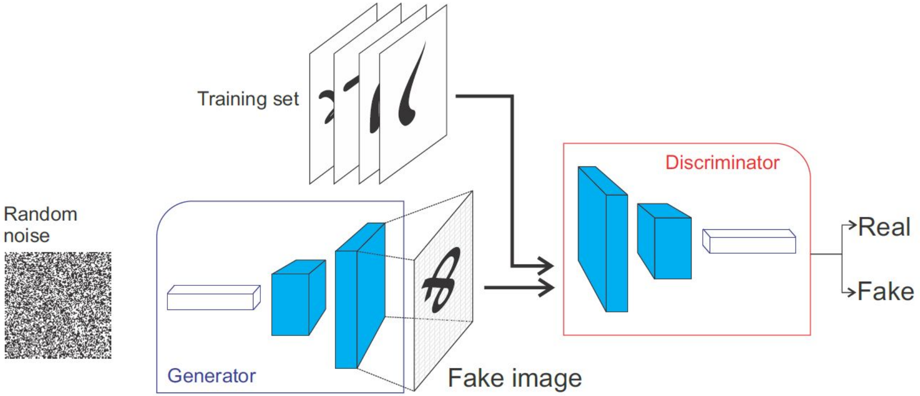
Generative Cooperative Networks for Natural Language Generation

Sylvain Lamprier, Thomas Scialom, Antoine Chaffin, Vincent Claveau,
Ewa Kijak, Benjamin Piwowarski, Jacopo Staiano

Language GANs fall short

- **GANs are good for approximating continuous data distributions:**

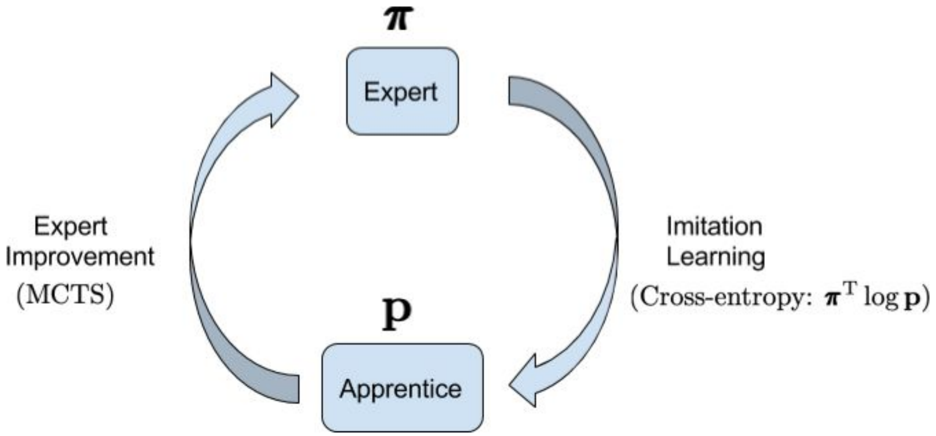


- **GANs for discrete data as text:**
  - ➢ No backpropagation from the discriminator to the generator :
    - **Reinforcement Learning** with Discriminator scores as Rewards
      - **Noisy**, **Sparse** and **Moving** Rewards
    - Existing language GANs are known to **fall short** (Caccia et al, 2020)

Cooperative Decoding

- ➔ Use of the **discriminator *D* cooperatively with the generator *p*** for sampling texts
  - In Beam Search: DAS [Scialom et. al, 2020b], Discriminative EBM [Ranzato et al., 2019]
  - In MCTS: SelfGAN [Scialom et. al, 2021]

- ➔ SelfGAN: Cooperative decoding can be useful for training via Expert Iteration



But unstable even at the optimum !

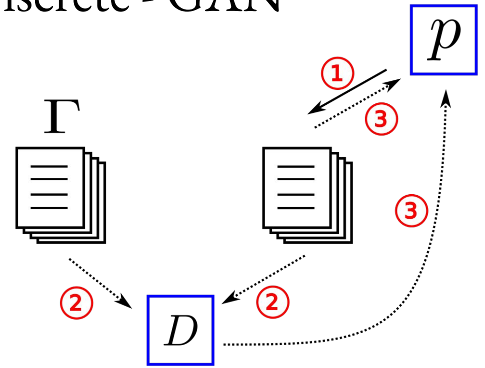
- Can diverge or oscillate
- Requires a LR scheduler

➔ We show that sampling from $q(\tau) \propto p(\tau)D(\tau)$ can allow to ensure convergence

(under usual assumptions and a **Reward-augmented Maximum Likelihood** process (RML) [Norouzi et al., 2016])

GAN vs RML-GAN

Discrete - GAN



1. Sample M documents from generator *p*

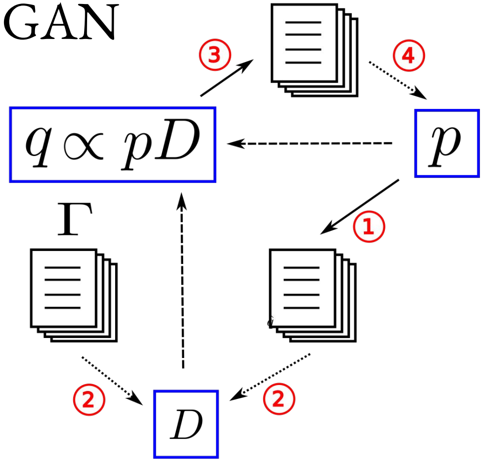
$$y^i \sim p_\theta(y^i)$$

2. Train the discriminator D_t

3. Train *p* with rewards from discriminator *D* on generated samples (policy gradient)

$$\theta \leftarrow \theta + \epsilon \frac{1}{M} \sum_{i=1}^{M} D_t(y^i) \nabla_\theta \log p_\theta(y^i)$$

RML-GAN



1. Sample documents from generator *p*
2. Train the discriminator *D*
3. Generate M samples from $q \propto pD$

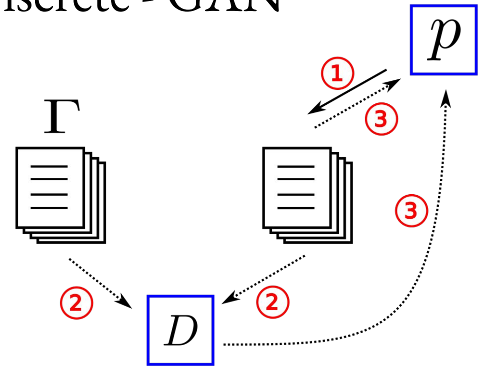
$$y^i \sim q(y^i)$$

4. Train *p* using samples from *q*

$$\theta \leftarrow \theta + \epsilon \frac{1}{M} \sum_{i=1}^{M} \nabla_\theta \log p_\theta(y^i)$$

GAN vs RML-GAN

Discrete - GAN



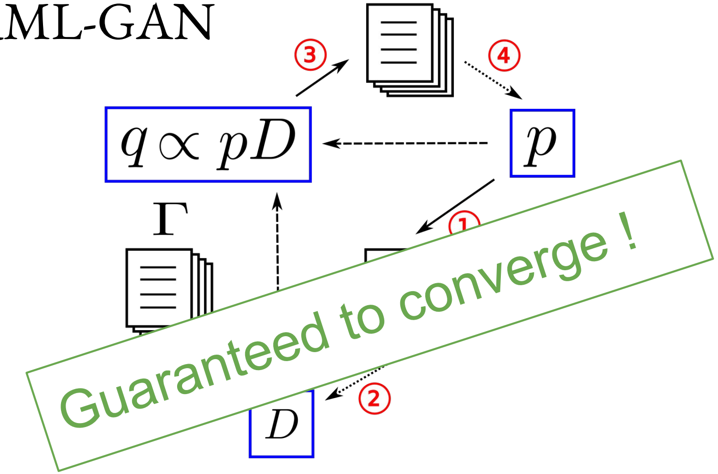
1. Sample M documents from generator *p*

$$y^i \sim p_\theta(y^i)$$

2. Train the discriminator D_t
3. Train *p* with rewards from discriminator *D* on generated samples (policy gradient)

$$\theta \leftarrow \theta + \epsilon \frac{1}{M} \sum_{i=1}^{M} D_t(y^i) \nabla_\theta \log p_\theta(y^i)$$

RML-GAN



1. Sample documents from generator *p*
2. Train the discriminator *D*
3. Generate M samples from $q \propto pD$

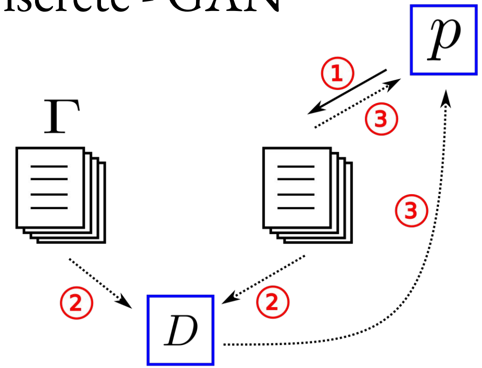
$$y^i \sim q(y^i)$$

4. Train *p* using samples from *q*

$$\theta \leftarrow \theta + \epsilon \frac{1}{M} \sum_{i=1}^{M} \nabla_\theta \log p_\theta(y^i)$$

GAN vs RML-GAN

Discrete - GAN



1. Sample M documents from generator p

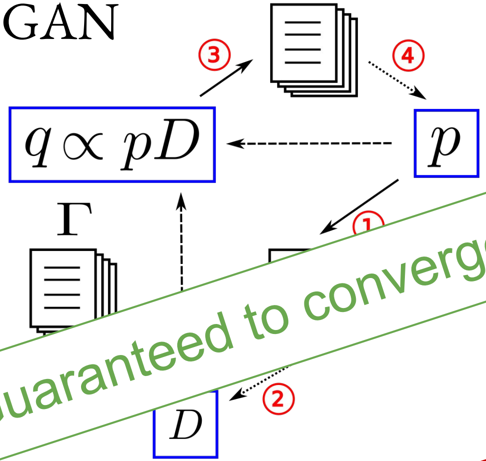
$$y^i \sim p_\theta(y^i)$$

2. Train the discriminator D_t

3. Train p with rewards from discriminator D on generated samples (policy gradient)

$$\theta \leftarrow \theta + \epsilon \frac{1}{M} \sum_{i=1}^{M} D_t(y^i) \nabla_\theta \log p_\theta(y^i)$$

RML-GAN



Guaranteed to converge !

1. Sample documents f

2. Train the

3. m $q \propto pD$

$$\sim q(y^i)$$

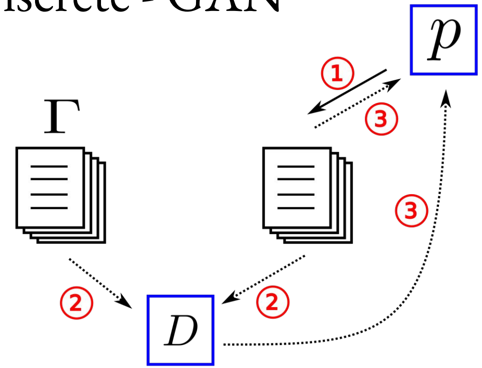
rain p using samples from q

$$\theta \leftarrow \theta + \epsilon \frac{1}{M} \sum_{i=1}^{M} \nabla_\theta \log p_\theta(y^i)$$

But we do not know sampling from q !

GAN vs GCN

Discrete - GAN



1. Sample M documents from generator *p*

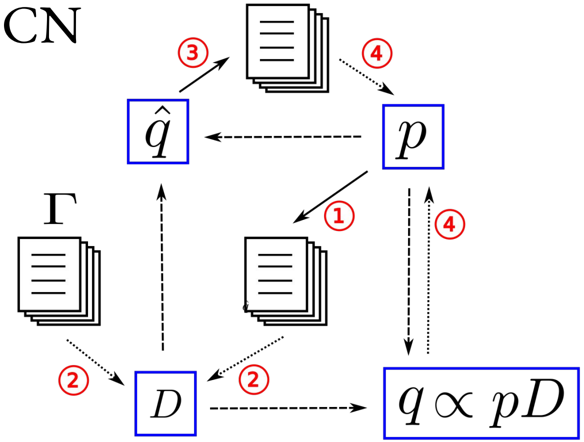
$$y^i \sim p_\theta(y^i)$$

2. Train the discriminator D_t

3. Train *p* with rewards from discriminator *D* on generated samples (policy gradient)

$$\theta \leftarrow \theta + \epsilon \frac{1}{M} \sum_{i=1}^{M} D_t(y^i) \nabla_\theta \log p_\theta(y^i)$$

GCN



1. Sample documents from generator *p*
2. Train the discriminator *D*
3. Generate M samples from $\hat{q}$

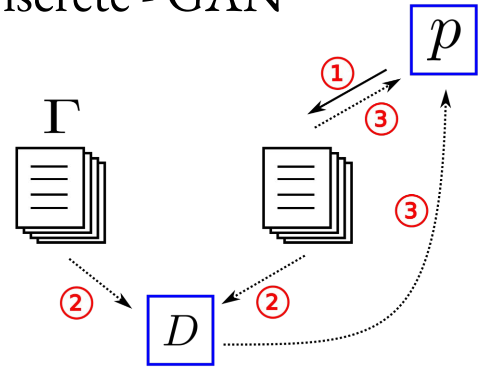
$$y^i \sim \hat{q}(y^i)$$

4. Train p using weighted importance sampling

$$\theta \leftarrow \theta + \epsilon \frac{1}{\sum_{i=1}^{M} w^i} \sum_{i=1}^{M} w^i \nabla_\theta \log p_\theta(y^i) \quad \text{with: } w^i = \frac{q(y^i)}{\hat{q}(y^i)}$$

GAN vs GCN

Discrete - GAN



1. Sample M documents from generator *p*

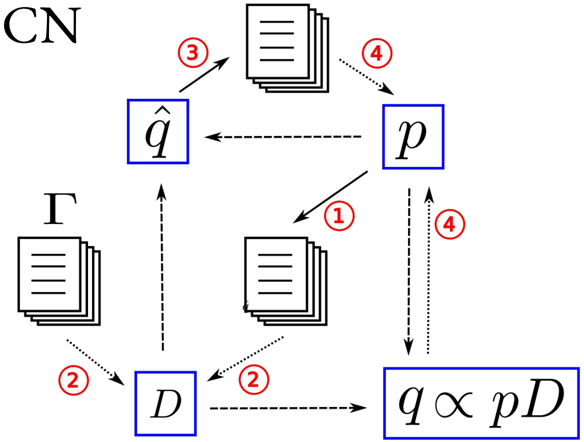
$$y^i \sim p_\theta(y^i)$$

2. Train the discriminator D_t

3. Train *p* with rewards from discriminator *D* on generated samples (policy gradient)

$$\theta \leftarrow \theta + \epsilon \frac{1}{M} \sum_{i=1}^{M} D_t(y^i) \nabla_\theta \log p_\theta(y^i)$$

GCN



1. Sample documents from generator *p*
2. Train the discriminator *D*
3. Generate M samples from $\hat{q}$

$$y^i \sim \hat{q}(y^i)$$

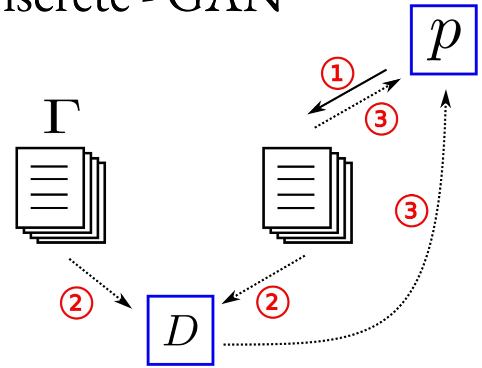
4. Train p using weighted importance sampling

$$\theta \leftarrow \theta + \epsilon \frac{1}{\sum_{i=1}^{M} w^i} \sum_{i=1}^{M} w^i \nabla_\theta \log p_\theta(y^i) \quad \text{with: } w^i = \frac{q(y^i)}{\hat{q}(y^i)}$$

⇒ With $\hat{q} = p_\theta$, we have: $\theta \leftarrow \theta + \epsilon \frac{1}{Z_t} \sum_{i=1}^{M} D_t(y^i) \nabla_\theta \log p_\theta(y^i)$, where $Z_t = \sum_{i=1}^{M} p_\theta(y^i) D_t(y^i)$

GAN vs GCN

Discrete - GAN



1. Sample M documents from generator *p*

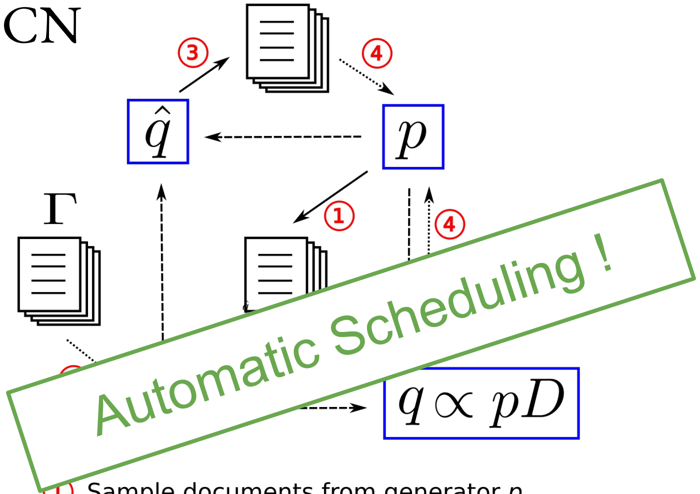
$$y^i \sim p_\theta(y^i)$$

2. Train the discriminator D_t

3. Train *p* with rewards from discriminator *D* on generated samples (policy gradient)

$$\theta \leftarrow \theta + \epsilon \frac{1}{M} \sum_{i=1}^{M} D_t(y^i) \nabla_\theta \log p_\theta(y^i)$$

GCN



1. Sample documents from generator *p*
2. Train the discriminator *D*
3. Generate M samples from $\hat{q}$

$$y^i \sim \hat{q}(y^i)$$

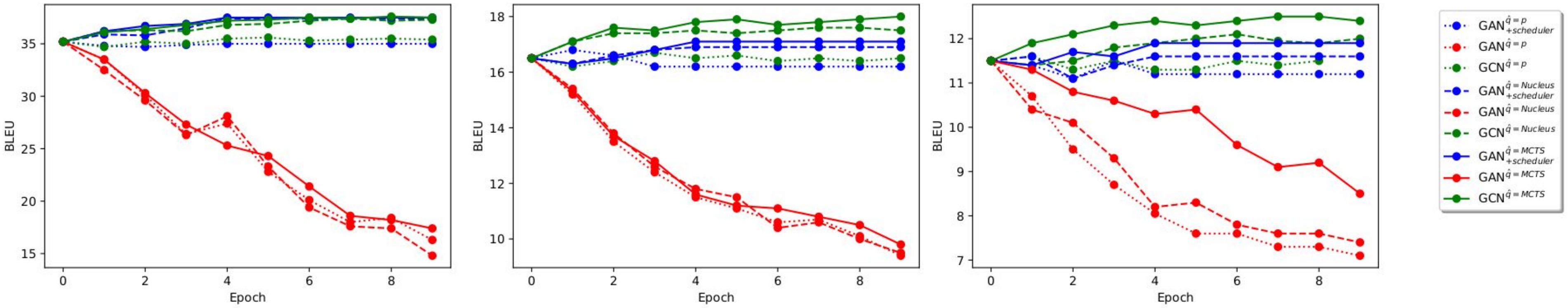
4. Train p using weighted importance sampling

$$\theta \leftarrow \theta + \epsilon \frac{1}{\sum_{i=1}^{M} w^i} \sum_{i=1}^{M} w^i \nabla_\theta \log p_\theta(y^i) \quad \text{with: } w^i = \frac{q(y^i)}{\hat{q}(y^i)}$$

⇒ With $\hat{q} = p_\theta$, we have: $\theta \leftarrow \theta + \epsilon \frac{1}{Z_t} \sum_{i=1}^{M} D_t(y^i) \nabla_\theta \log p_\theta(y^i), \text{ where } Z_t = \sum_{i=1}^{M} p_\theta(y^i) D_t(y^i)$

GAN vs GCN

Unconditional NLG | Question Generation | Summarization



➔ No scheduler required

➔ Sampling closer to q allows to still improve results (state-of-the-art) !

- Use of Monte-Carlo Tree Search guided by $p_\theta(y)D(y)$

Thank you for your attention !

Please check the paper for more details