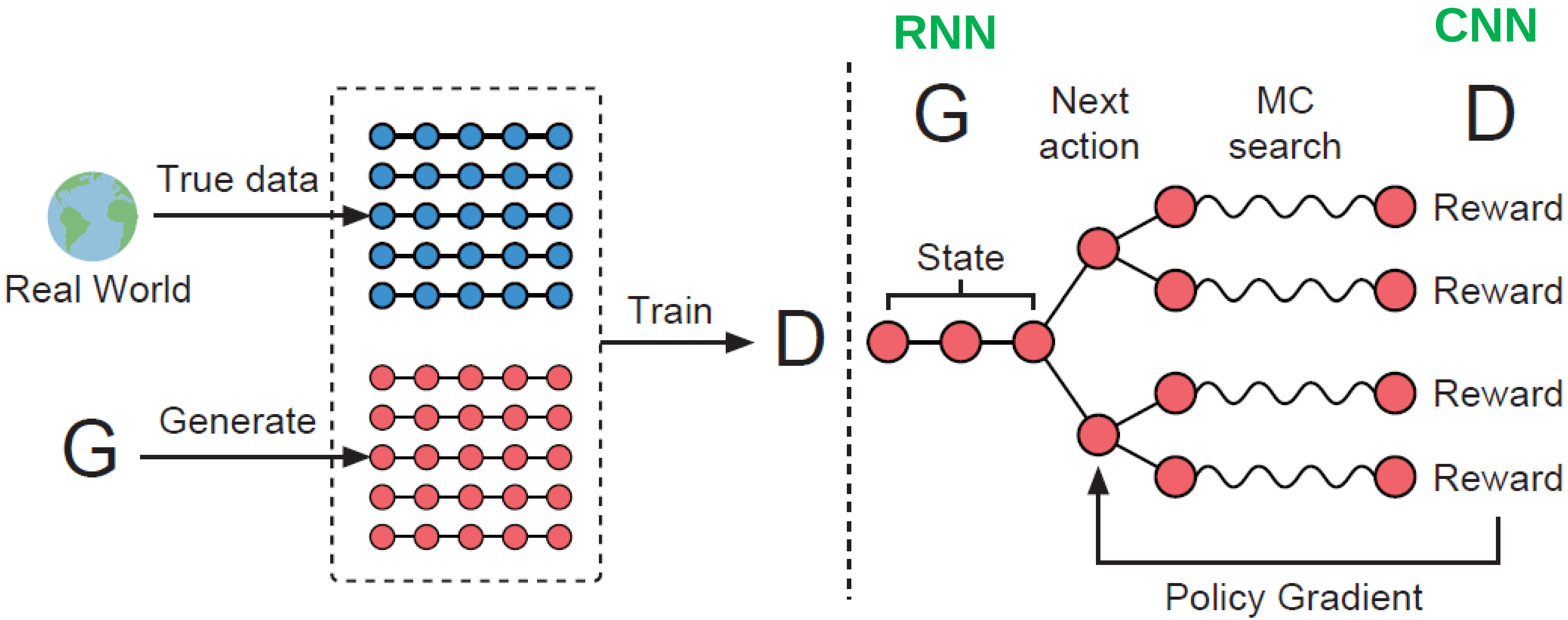
Using Wasserstein GAN (with RL) for Natural Language Generation

Guan-Yuan Chen

SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient

AAAI-17

Lantao Yu[†], Weinan Zhang[†*], Jun Wang[‡], Yong Yu[†]

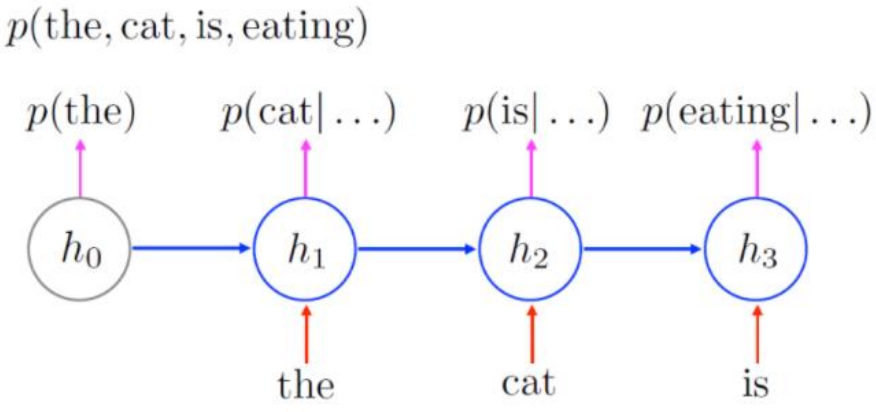


The problem of GAN on NLG

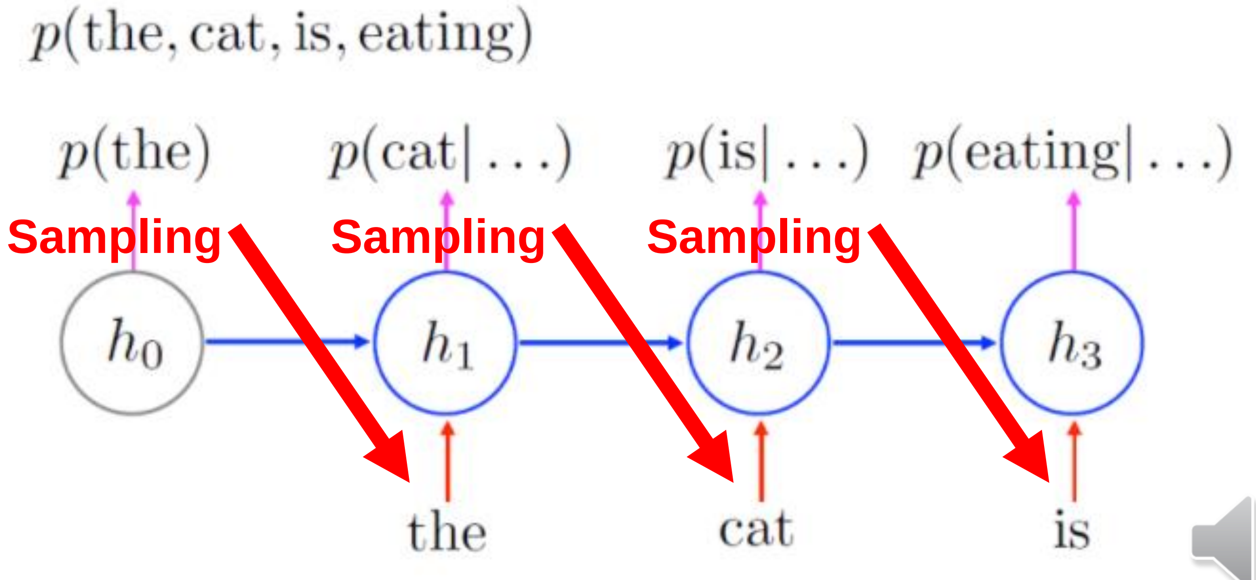
A major reason lies in that the discrete outputs from the generative model make it difficult to pass the gradient update from the discriminative model to the generative model.

The problem of GAN on RNN sampling

Training



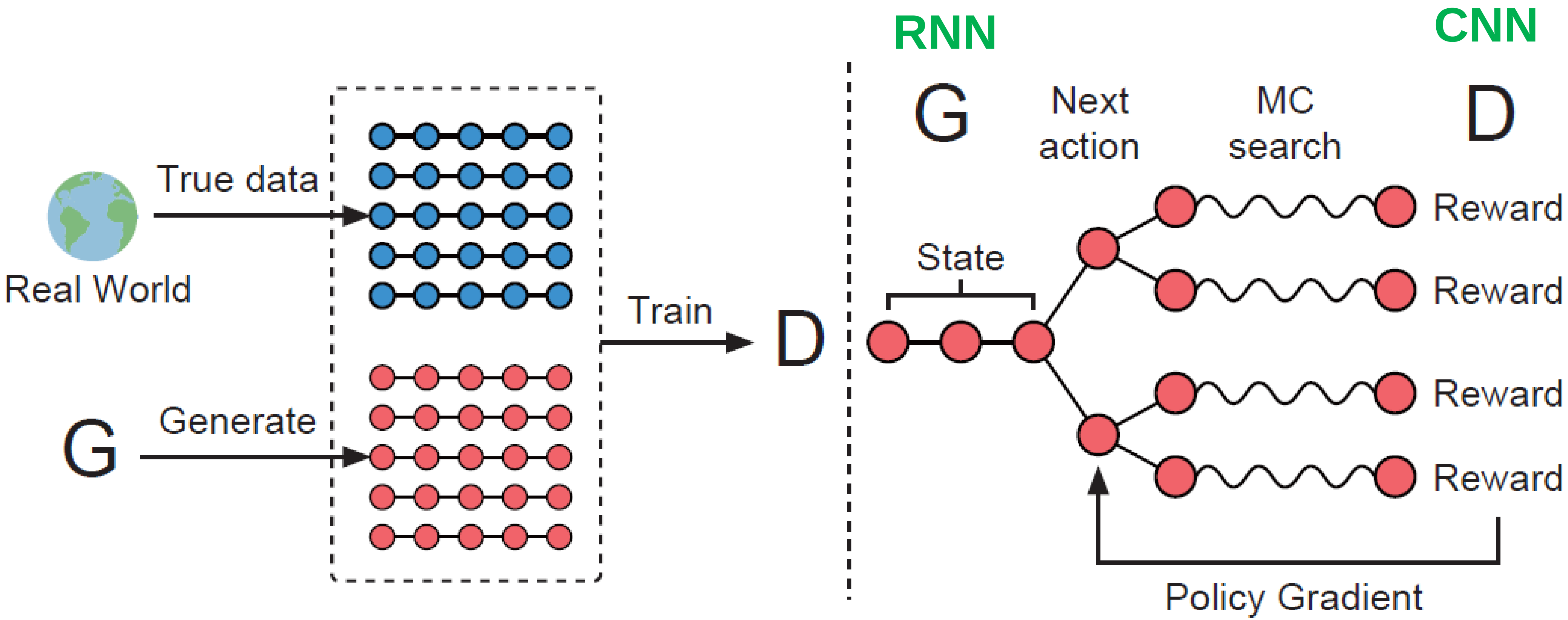
Sampling



SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient

AAAI-17

Lantao Yu[†], Weinan Zhang[†*], Jun Wang[‡], Yong Yu[†]



Algorithm 1 Sequence Generative Adversarial Nets

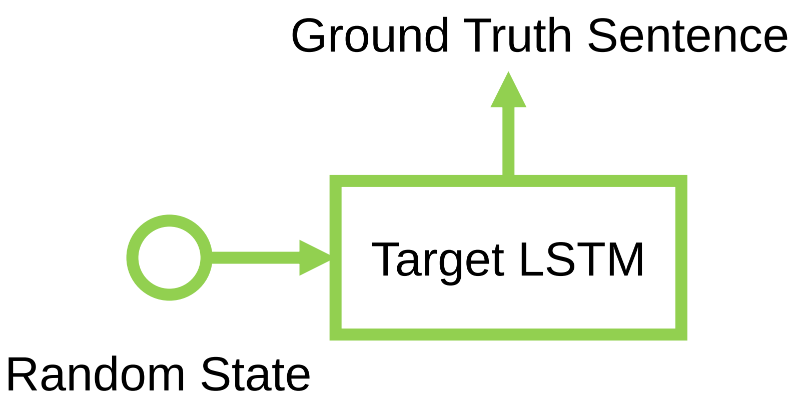
```
Require: generator policy G_θ; roll-out policy G_β; discriminator
    D_φ; a sequence dataset S = {X_{1:T}}
Initialize G_θ, D_φ with random weights θ, φ.
Pre-train G_θ using MLE on S
β ← θ
Generate negative samples using G_θ for training D_φ
Pre-train D_φ via minimizing the cross entropy
repeat
    for g-steps do
        Generate a sequence Y_{1:T} = (y_1, ..., y_T) ~ G_θ
        for t in 1 : T do
            Compute Q(a = y_t; s = Y_{1:t-1}) by Eq. (4)
        end for
        Update generator parameters via policy gradient Eq. (8)
    end for
    for d-steps do
        Use current G_θ to generate negative examples and com-
        bine with given positive examples S
        Train discriminator D_φ for k epochs by Eq. (5)
    end for
    β ← θ
until SeqGAN converges
```

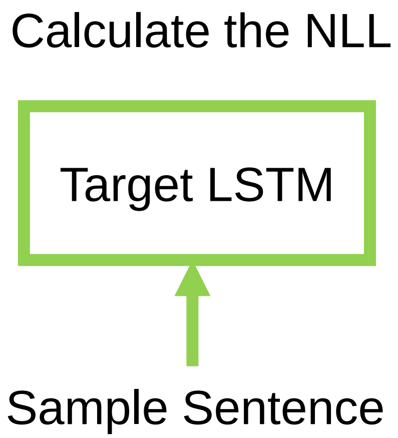
$$Q_{D_\phi}^{G_\theta}(s = Y_{1:t-1}, a = y_t) = \tag{4}$$
$$\begin{cases} \frac{1}{N}\sum_{n=1}^{N} D_\phi(Y_{1:T}^n), \; Y_{1:T}^n \in \mathrm{MC}^{G_\beta}(Y_{1:t}; N) & \text{for} \quad t < T \\ D_\phi(Y_{1:t}) & \text{for} \quad t = T, \end{cases}$$

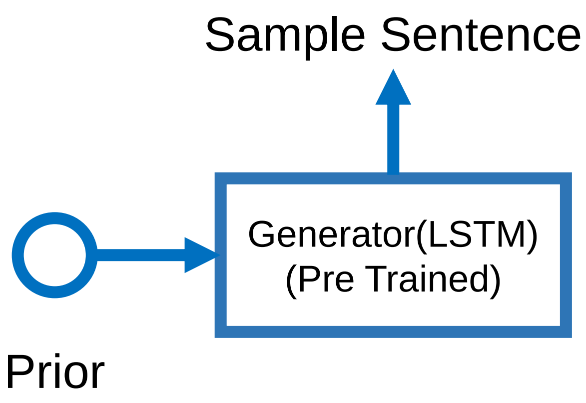
$$\nabla_\theta J(\theta) = \mathbb{E}_{Y_{1:t-1} \sim G_\theta}\Big[\sum_{y_t \in \mathcal{Y}} \nabla_\theta G_\theta(y_t | Y_{1:t-1}) \cdot Q_{D_\phi}^{G_\theta}(Y_{1:t-1}, y_t)\Big]. \tag{6}$$

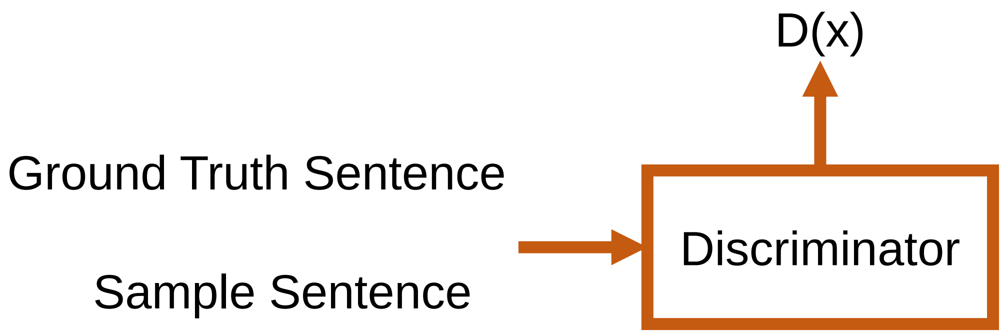
$$\theta \leftarrow \theta + \alpha_h \nabla_\theta J(\theta), \tag{8}$$

$$\min_\phi -\mathbb{E}_{Y \sim p_{\text{data}}}[\log D_\phi(Y)] - \mathbb{E}_{Y \sim G_\theta}[\log(1 - D_\phi(Y))]. \tag{5}$$



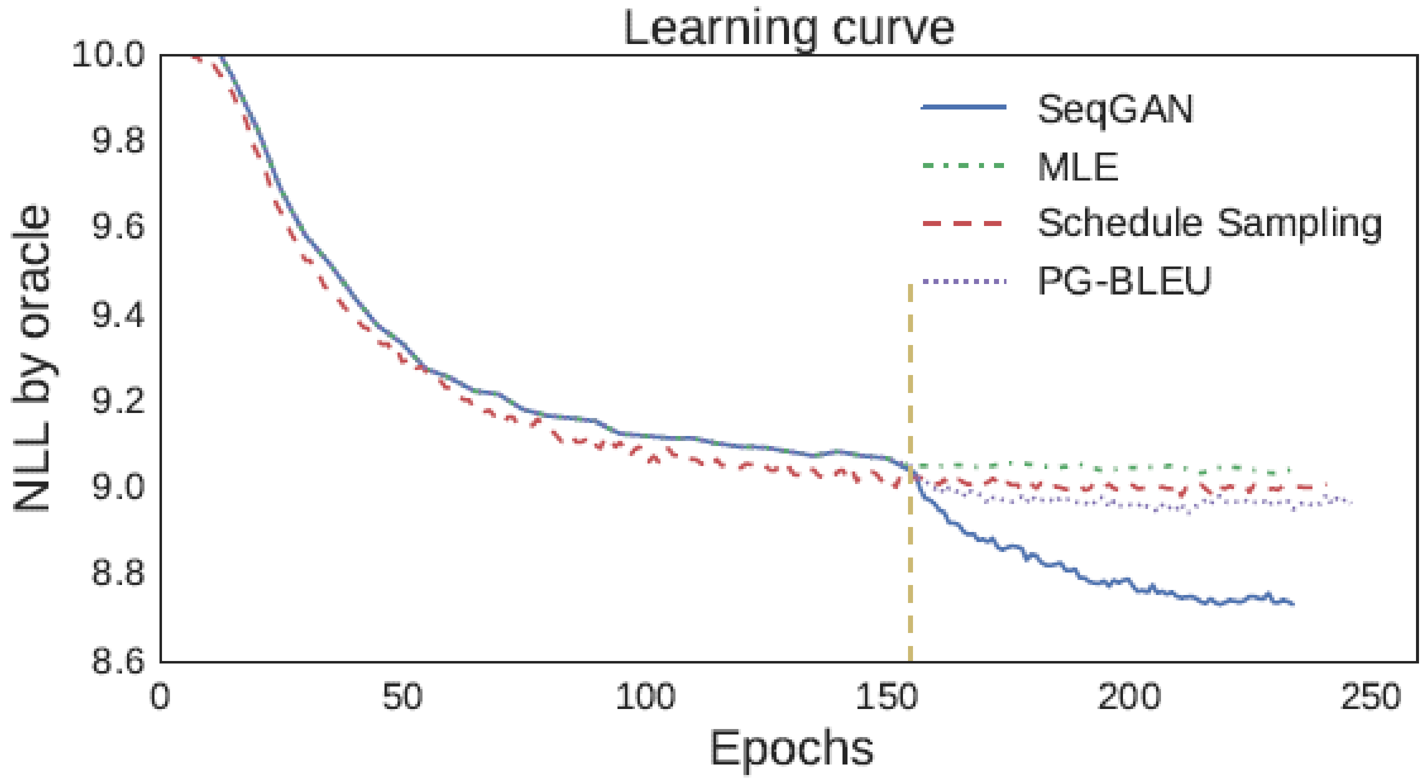






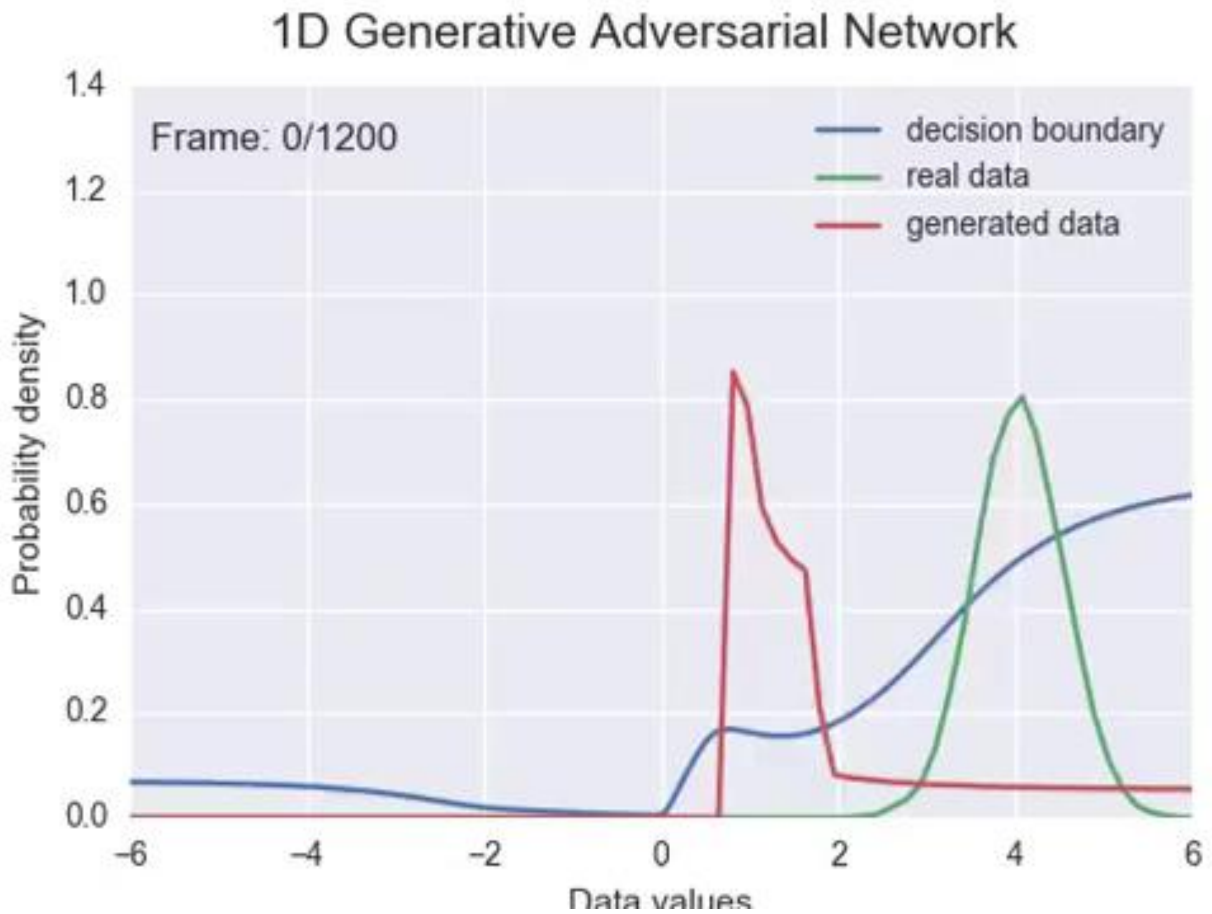
Adversarial Training Step:
1. Update D
2. Copy the Generator be the Rollout LSTM
3. Update G by Policy Gradient
4. Calculate the NLL by Target LSTM

Algorithm	Random	MLE	SS	PG-BLEU	SeqGAN
NLL	10.310	9.038	8.985	8.946	**8.736**
p-value	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	



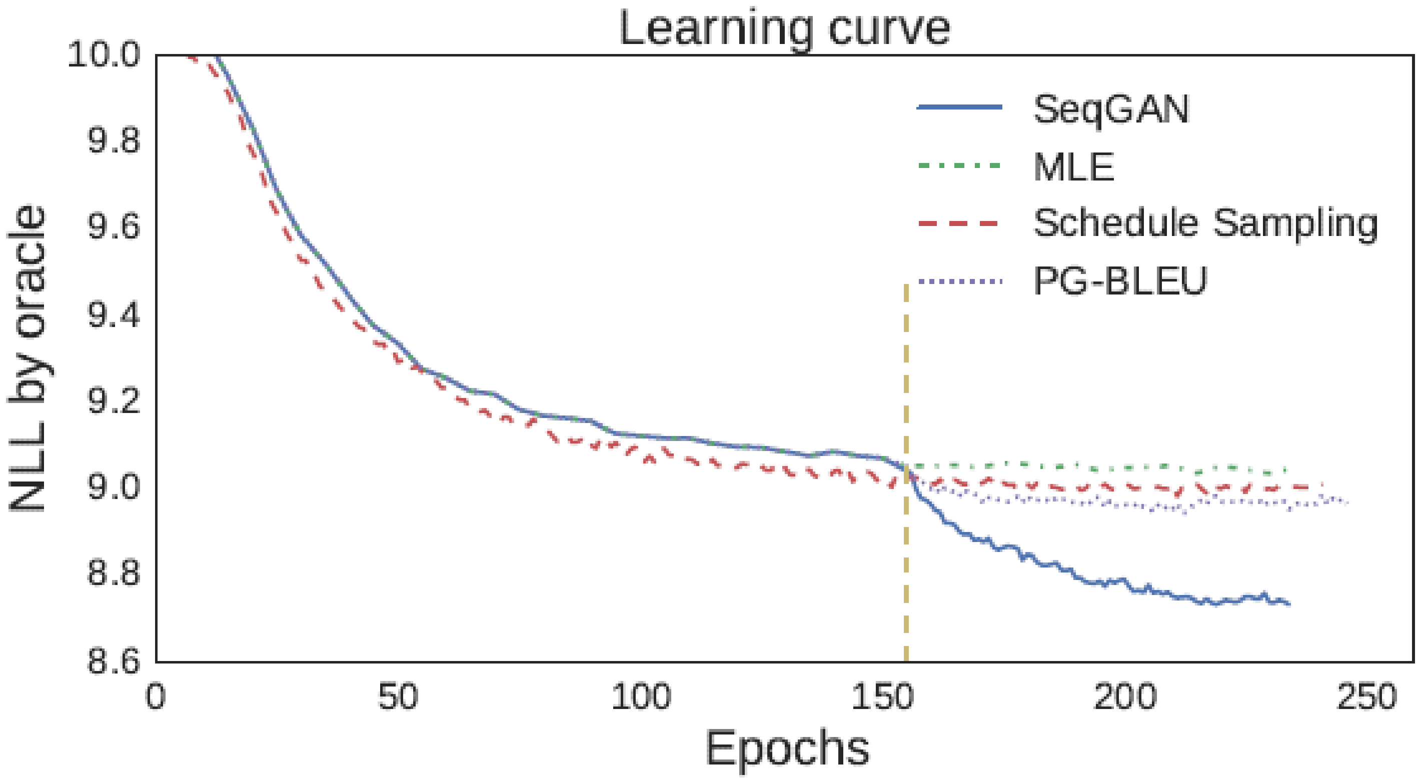
Problems of GAN:

Mode collapse problem,
The loss of G and D represent nothing (about the convergence)



http://blog.aylien.com/introduction-generative-adversarial-networks-code-tensorflow/

Algorithm	Random	MLE	SS	PG-BLEU	SeqGAN
NLL	10.310	9.038	8.985	8.946	**8.736**
p-value	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	



$$\mathrm{JSD}(P \parallel Q) = \frac{1}{2}D(P \parallel M) + \frac{1}{2}D(Q \parallel M)$$

where $M = \frac{1}{2}(P + Q)$

Earth Mover's Distance

Wasserstein GAN

Martin Arjovsky[1], Soumith Chintala[2], and Léon Bottou[1,2]

2017

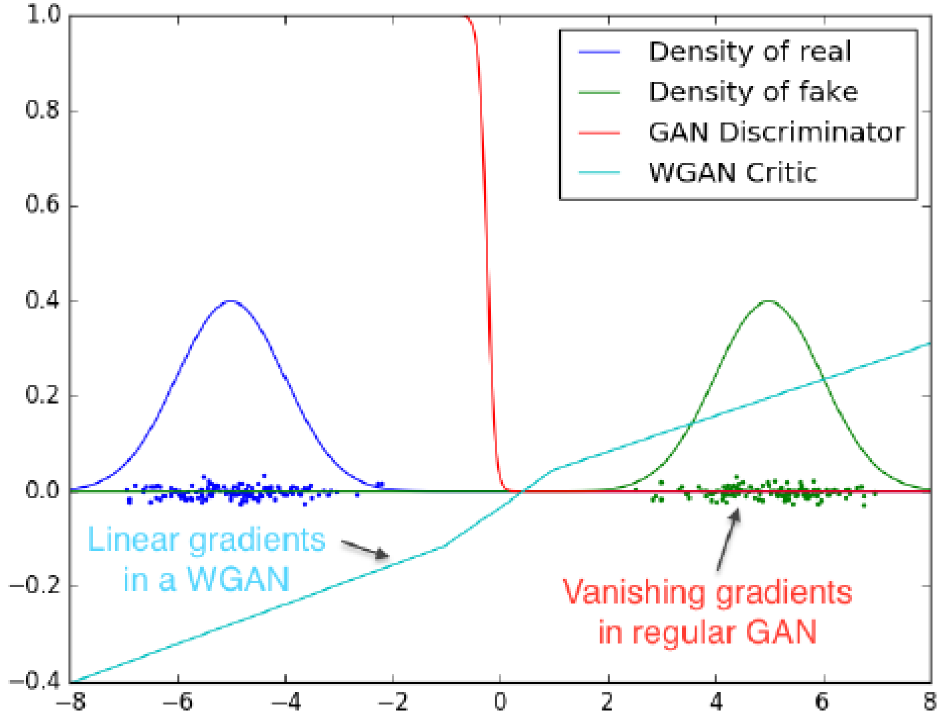
Wasserstein GAN

Generative Adversarial Nets

$$\max_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

$$\nabla_\theta W(\mathbb{P}_r, \mathbb{P}_\theta) = -\mathbb{E}_{z\sim p(z)}[\nabla_\theta f(g_\theta(z))]$$



Wasserstein GAN

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**

  for $t = 0, \ldots, n_{\text{critic}}$ **do**

    Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.

    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.

    $g_w \leftarrow \nabla_w \boxed{\left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]}$

    $w \leftarrow w + \alpha \cdot \boxed{\text{RMSProp}(w, g_w)}$

    $\boxed{w \leftarrow \text{clip}(w, -c, c)}$

  end for

  Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.

  $g_\theta \leftarrow \boxed{-\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))}$

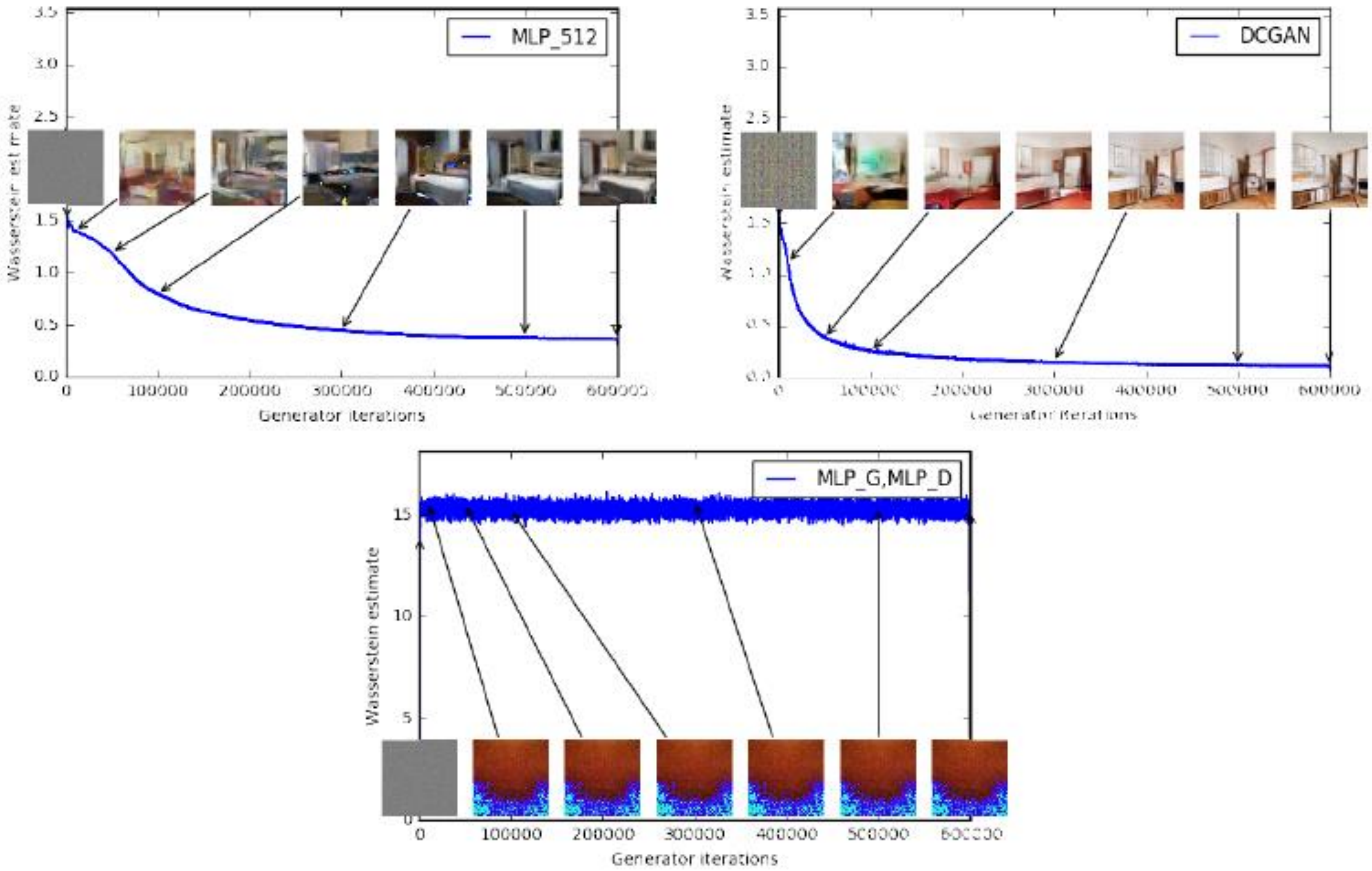
  $\theta \leftarrow \theta - \alpha \cdot \boxed{\text{RMSProp}(\theta, g_\theta)}$

end while

Improved Training of Wasserstein GANs

Ishaan Gulrajani[1], Faruk Ahmed[1], Martin Arjovsky[2]

2017



MLP_512
Wasserstein estimate
Generator iterations
0.0
0.5
1.0
1.5
2.5
3.0
3.5
0
100000
200000
300000
400000
500000
600000
DCGAN
MLP_G,MLP_D
0
5
10
15

Using Wasserstein GAN (with RL) for Natural Language Generation

方法:

以character-based的方式建構language generator(GRU)。

以RNN(GRU)的方式建構discriminator 。

分別以Original GAN and Wasserstein GAN with Policy Gradient 之方式，訓練discriminator與generator。

Dataset: 7200 abstracts of Arxiv papers about neural networks (from Stanford CS 20SI)

$$J^{(D)} = -E_{x \sim p_{data}}[D(x)] + E_{z \sim p_z}[D(G(z))] \quad (1)$$

$$\theta_d = \theta_d - \eta RMSProp(\theta_d, g_{\theta_d}), \quad \theta_d \leftarrow clip(\theta_d, c, -c) \quad (2)$$

其中，θ_d 為 discriminator 的參數。另定義 a prior on input noise variables $p_z(z)$ (from a random normal distribution)，作為 generator function 初始的 input value。細部的 procedure 與 algorithm 之定義如下:

Algorithm Wasserstein GAN with RL

Require: generator G_{θ_g}; discriminator D_{θ_d}; a sequence dataset $S = \{X_{1:T}\}$; roll-out policy G_ϕ

Initialize G_{θ_g}, D_{θ_d} with random weights θ_g, θ_d;

$\phi \leftarrow \theta_g$

Generate negative samples using G_{θ_g} for training D_{θ_d}

repeat

  Pre-train G_{θ_g} using MLE on S

  for g-steps **do**

    Generate a sequence $Y_{1:T} = (y_1, \ldots, y_T) \sim G_{\theta_g}$

    for t in 1 : T **do**

      Compute $Q(a = y_t; s = Y_{1:t-1})$ by Eq. (4) in [2]

    end for

    Update generator parameters via policy gradient Eq. (8) in [2]

  end for

  for d-steps **do**

    Use current G_{θ_g} to generate negative examples and combine with given positive example

    Train discriminator D_{θ_d} for 5 epochs by Eq. (1) and (2)

  end for

  $\phi \leftarrow \theta_g$

until D_{θ_d} converges

Result

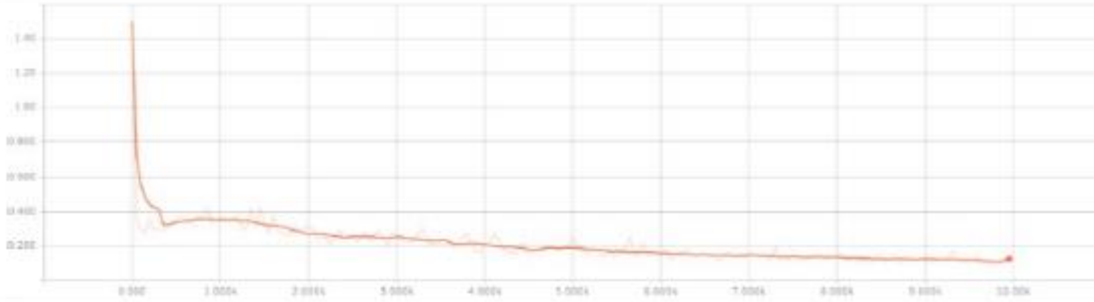
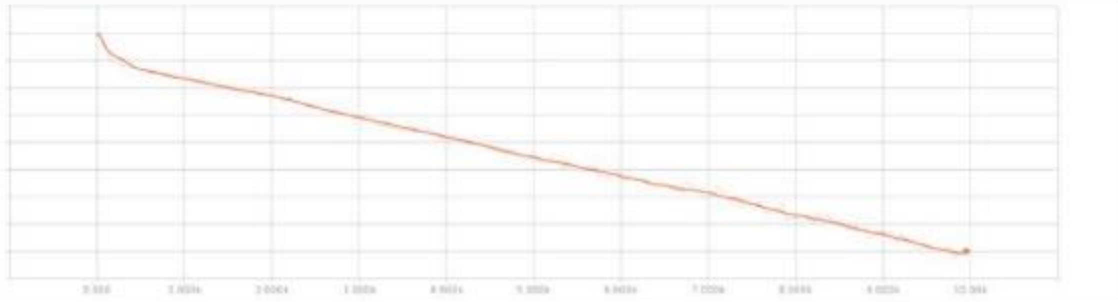
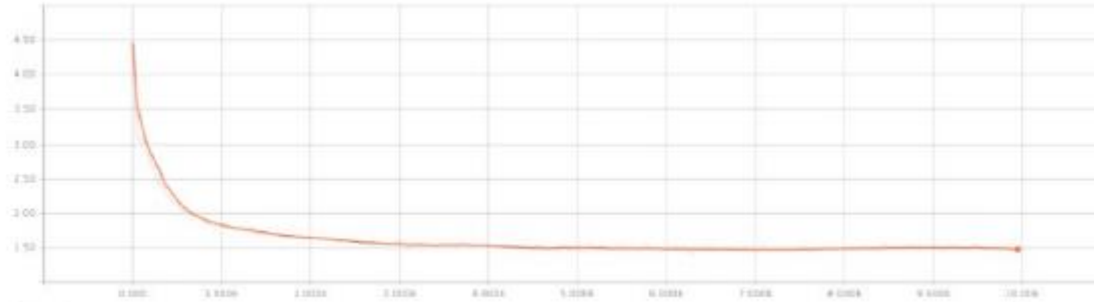
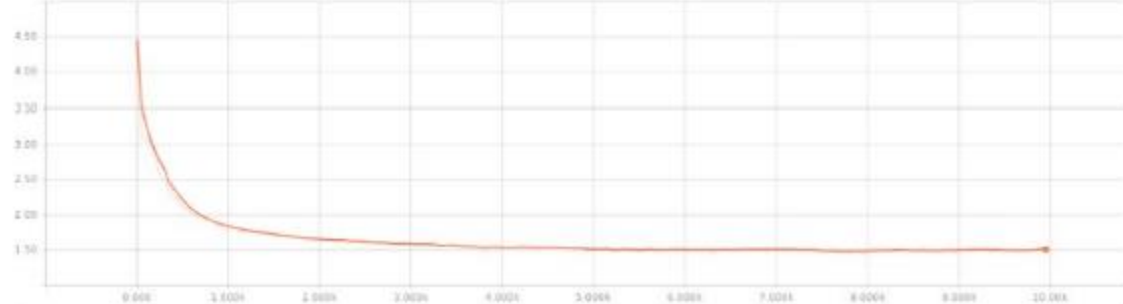
RNN Stack: 3, Hidden Size of G: 200, Hidden Size of D: 200	
SeqGAN	**WGAN with PG, clipping value: 5**
==	==
the activation of the network are a difficulties of the network are a difficulties of the network are a difficulties of the network are a difficulties of the network are a difficulties of the network are a difficulties of the network are a difficulties of the network are a difficulties of the network are a difficulties of the network are a difficulties of the network are a difficulties of the network are a difficulties of the network are a difficulties	in neural networks in machine learning. One of techniques have yielded recognition, type. We study the conventional layer-wise pre-training to the individual models and deep neural networks as part of a method for stochastic gradient and analyzing algorithms. This parallel learning architectures are generic and deep neural network architectures. The gradients. In this paper, we propose a novel approach for nonlinear unit, to the recently proposed deep RNNs benefit
==	==

Result

RNN Stack: 3, Hidden Size of G: 200, Hidden Size of D: 200	
SeqGAN	WGAN with PG, clipping value: 5

SeqGAN	WGAN with PG, clipping value: 5
==	==
the activation of the network are a difficulties of the network are a difficulties of the network are a difficulties of the network are a difficulties of the network are a difficulties of the network are a difficulties of the network are a difficulties of the network are a difficulties of the network are a difficulties of the network are a difficulties of the network are a difficulties of the network are a difficulties of the network are a difficulties of the network are a difficulties	a fixed-point of computational complexity. Thus, we propose a new synchronization protocol that can be model distributed operation recent to train leventigation and dimensionality reduction in a data of noness of our methods on the same datasets in the relevant features of acoustic framework for example, representations of the approaches with a computationally expensive to train and deep learning architecture can be trained using standard points that by
==	==

Result

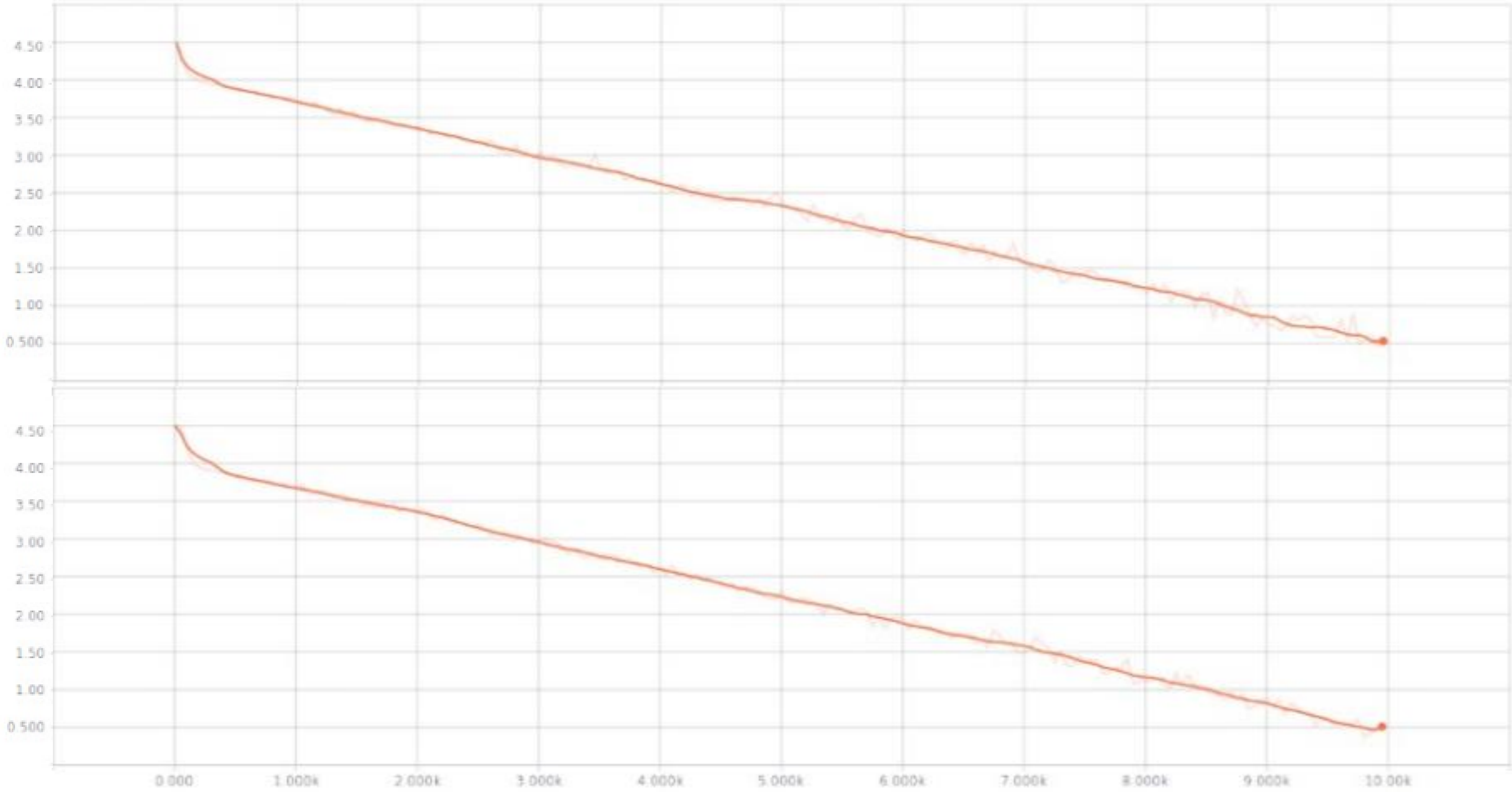
<table>
<tr><td colspan="2">RNN Stack: 3, Hidden Size of G: 200, Hidden Size of D: 200</td></tr>
<tr><td>SeqGAN</td><td>WGAN with PG, clipping value: 5</td></tr>
<tr><td colspan="2">Discriminator Loss</td></tr>
<tr><td></td><td></td></tr>
<tr><td colspan="2">Mean NLL</td></tr>
<tr><td></td><td></td></tr>
<tr><td>
</td><td>
</td></tr>
</table>

Result: Discriminator loss

RNN Stack 3, Hidden size of G 200, D 200

Adam

RMSprop



Result: Mean NLL

RNN Stack 3, Hidden size of G 200, D 200

Adam

RMSprop

References :

[1] Martin Arjovsky, Soumith Chintala, and Leon Bottou
Wasserstein GAN
2017
[2] Lantao Yuy, Weinan Zhangy, Jun Wangz, Yong Yu
SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient
2017
[3] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron Courville
Improved Training of Wasserstein GANs
2017
[4] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza
Generative Adversarial Nets
2014
[5] https://vincentherrmann.github.io/blog/wasserstein/
Wasserstein GAN and the Kantorovich-Rubinstein Duality
[6] https://github.com/LantaoYu/SeqGAN
Implementation of Sequence Generative Adversarial Nets with Policy Gradient
[7] http://blog.aylien.com/introduction-generative-adversarial-networks-code-tensorflow/
An introduction to Generative Adversarial Networks (with code in TensorFlow)
[8] http://web.stanford.edu/class/cs20si/
CS 20SI: Tensorflow for Deep Learning Research