An Online Learning Approach to Generative Adversarial Networks

Zhongkai Shangguan

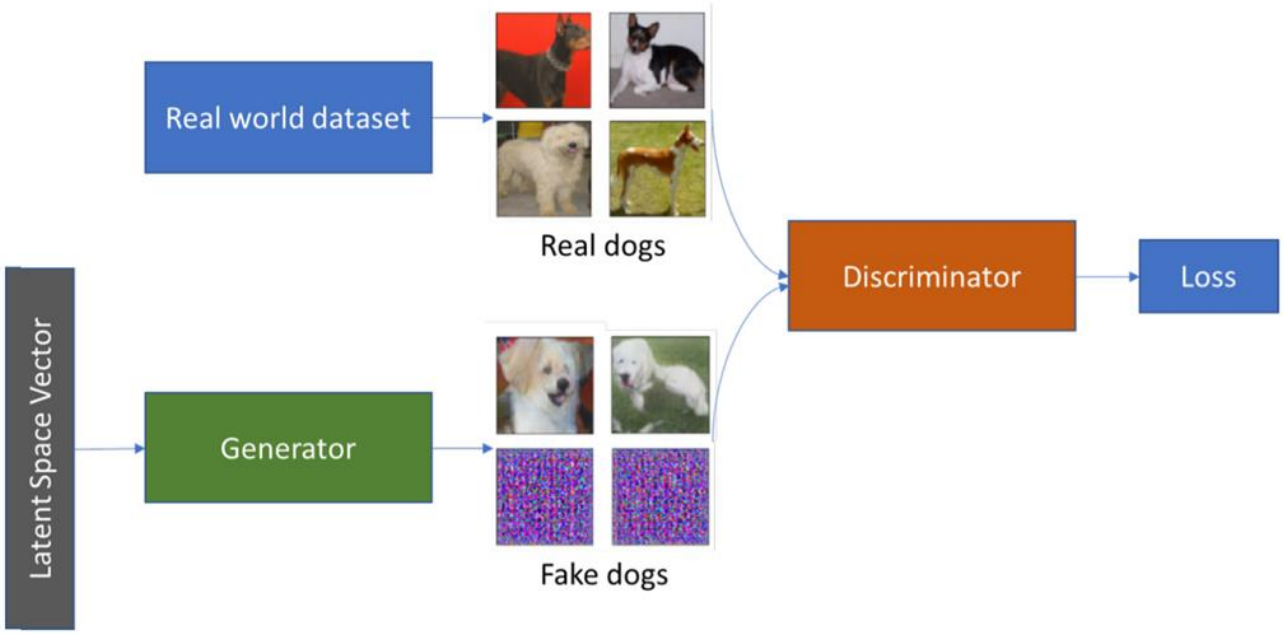
Dec. 8

Boston University

Outline

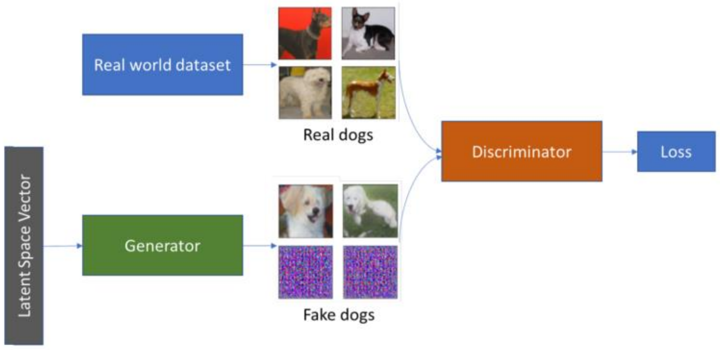
- Background of Generative Adversarial Networks (GAN).
- Mixed Nash Equilibrium (MNE).
- Theoretical algorithm: CHEKHOV GAN.
- Proof of the convergence.
- Practical algorithm and Experiments.

Generative Adversarial Network (GAN)



Generative Adversarial Network (GAN)

$$M(G,D) = \max_{D}\min_{G} L(D,G) = \frac{1}{N}\sum_{i=1}^{N} logD(x_i) + log(1 - D(G(z_i)))$$



Goal of Generator

$$\min_{u\in K_1} \boxed{\max_{v\in K_2} M(u,v)}$$

Goal of Discriminator

$$\max_{v\in K_2} \boxed{\min_{u\in K_1} M(u,v)}$$

	Rock	*Paper*	*Scissors*
Rock	*0*	*1*	*-1*
Paper	*-1*	*0*	*1*
Scissors	*1*	*-1*	*0*

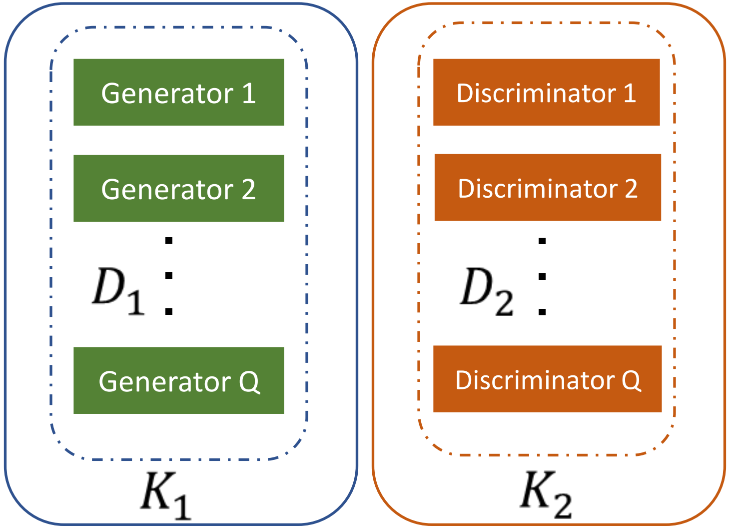
Pure Nash Equilibrium

$$M(u^*,v^*) \le \min_{u\in K_1} M(u,v^*) \qquad M(u^*,v^*) \ge \max_{v\in K_2} M(u^*,v)$$

Does a Pure Nash Equilibrium always exist?

Mixed Nash Equilibrium (MNE)

$$\mathbb{E}_{(\mathbf{u},\mathbf{v})\sim\mathcal{D}_1\times\mathcal{D}_2}[M(\mathbf{u},\mathbf{v})] \le \min_{\mathbf{u}\in\mathcal{K}_1} \mathbb{E}_{\mathbf{v}\sim\mathcal{D}_2}[M(\mathbf{u},\mathbf{v})], \ \& \ \mathbb{E}_{(\mathbf{u},\mathbf{v})\sim\mathcal{D}_1\times\mathcal{D}_2}[M(\mathbf{u},\mathbf{v})] \ge \max_{\mathbf{v}\in\mathcal{K}_2} \mathbb{E}_{\mathbf{u}\sim\mathcal{D}_1}[M(\mathbf{u},\mathbf{v})]$$



Definition 1.1. *Let $\varepsilon > 0$, the two distributions D_1, D_2 are called $\varepsilon - MNE$ if*

$$\mathbb{E}_{(u,v)\sim K_1\times K_2}[M(\boldsymbol{u},\boldsymbol{v})] \le \min_{\boldsymbol{u}\in K_1} \mathbb{E}_{v\sim D_2}[M(\boldsymbol{u},\boldsymbol{v})] + \varepsilon$$

$$\mathbb{E}_{(u,v)\sim K_1\times K_2}[M(\boldsymbol{u},\boldsymbol{v})] \ge \min_{\boldsymbol{v}\in K_2} \mathbb{E}_{u\sim D_1}[M(\boldsymbol{u},\boldsymbol{v})] - \varepsilon$$

$$if\ \varepsilon\ is\ sub - linear\ to\ T$$

$$\varepsilon = O(1/\sqrt{T})$$

Published as a conference paper at ICLR 2018

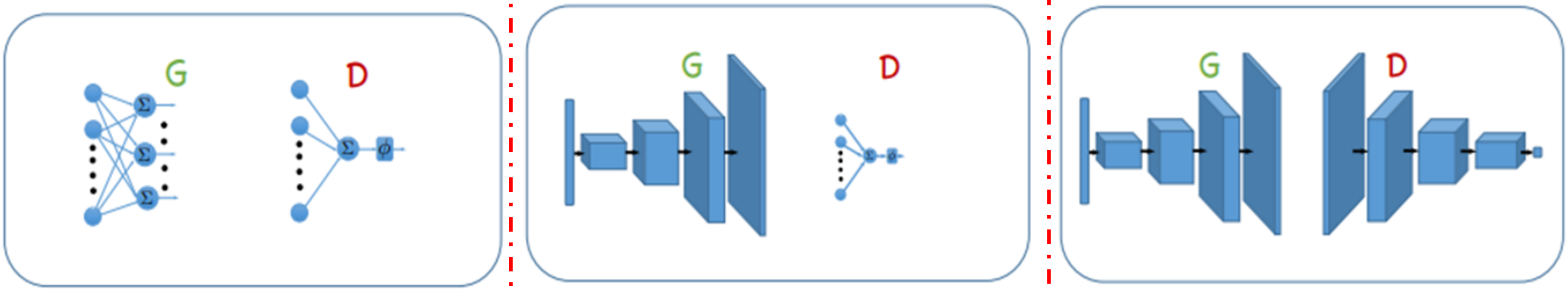
AN ONLINE LEARNING APPROACH TO GENERATIVE ADVERSARIAL NETWORKS

Paulina Grnarova, Kfir Y. Levy, Aurelien Lucchi, Thomas Hofmann, Andreas Krause
ETH Zürich
{paulina.grnarova,yehuda.levy,aurelien.lucchi,thomas.hofmann}@inf.ethz.ch
krausea@ethz.ch

- Proposed an algorithm that guarantees finding a MNE for GAN (Semi-Shallow).
- Designed a new GAN training algorithm that applies in practice (Deep).

G: Convex
D: Arbitrary

Three Different Types of GAN

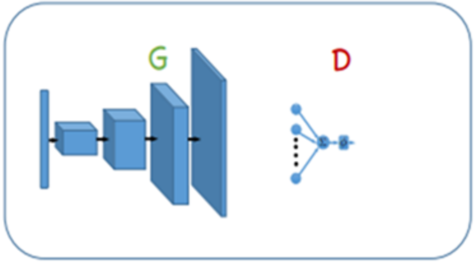


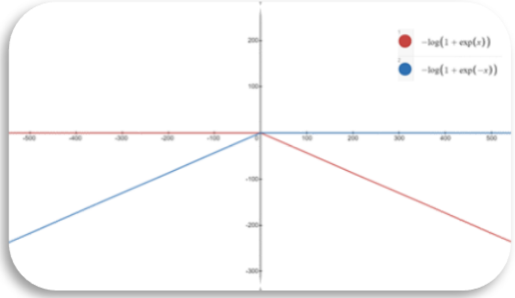
(a) G: Convex D: Concave

(b) G: Arbitrary D: Concave

(c) G: Arbitrary D: Arbitrary

Concavity of the Discriminator





$$F(a) = -log(1 + exp(-a))$$

Proposition 2.1. *Consider the GAN objective in Equation (1) and assume that the discriminator is a single-layer network with a sigmoid activation function, meaning* $D_v(x) = 1/(1 + exp(-v^T x))$, *where* $v \in R^n$. *Then the GAN objective is concave in* v.

$$M(G, D) = \max_D \min_G L(D, G) = \frac{1}{N} \sum_{i=1}^{N} logD(x_i) + log(1 - D(G(z_i)))$$

$$logD_v(x_i) = log(1) - log(1 + exp(-v^T x_i)) = -log(1 + exp(-v^T x_i))$$

$$\begin{aligned} log(1 - D_v(G_u(z_i))) &= log(1 - 1/(1 + exp(-v^T(G_u(z_i))))) \\ &= log(\frac{exp(-v^T(G_u(z_i)))}{(1 + exp(-v^T(G_u(z_i))))}) \\ &= log(1) - log(1 + 1/exp(-v^T G_u(z_i))) \\ &= -log(1 + exp(v^T G_u(z_i))) \end{aligned}$$

Algorithm 1 CHEKHOV GAN [4]

Input: Number of steps: T; Game objective: $M(\cdot, \cdot)$

for $t = 1$ `to` T **do**

Calculate:

$\boxed{\mathbf{u}_t \leftarrow \arg\min_{u \in K_1} \sum_{\tau=0}^{t-1} f_\tau(\mathbf{u})}$ $\triangleright A_1$

$\boxed{\mathbf{v}_t \leftarrow \arg\max_{v \in K_2} \sum_{\tau=0}^{t-1} \nabla g_\tau(\mathbf{v}_\tau)^T \mathbf{v} - \frac{\sqrt{T}}{2\eta_0} ||\mathbf{v}||^2}$ $\triangleright A_2$

Update:

$f_t(\cdot) = M(\cdot, \mathbf{v}_t)$; $g_t(\cdot) = M(\mathbf{u}_t, \cdot)$

end for

Output: $D_1 \sim Uni\{\mathbf{u}_1, ..., \mathbf{u}_T\}$; $D_2 \sim Uni\{\mathbf{v}_1, ..., \mathbf{v}_T\}$

Theorem 3.1. *Let K_2 be a convex set, M be a semi-concave zero-sum game, and assume M is L-Lipschitz continuous. Then the mixed strategies (D_1, D_2) produced by Algorithm 1 are $\varepsilon - MNE$, where $\varepsilon = \mathcal{O}(1/\sqrt{T})$.*

Proof Sketch

Theorem 3.1. *Let K_2 be a convex set, M be a semi-concave zero-sum game, and assume M is L-Lipschitz continuous. Then the mixed strategies (D_1, D_2) produced by Algorithm 1 are $\varepsilon - MNE$, where $\varepsilon = \mathcal{O}(1/\sqrt{T})$.*

Theorem 3.2. *The mixed strategies (D_1, D_2) that Algorithm 1 outputs are $\varepsilon - MNE$, where $\varepsilon := (B_T^{A_1} + B_T^{A_2})/T$ and $B_T^{A_1}$, $B_T^{A_2}$ are bounds on the regret of A_1, A_2.*

$B_T^{A_1}$ is $\mathcal{O}(\sqrt{T})$ $B_T^{A_2}$ is $\mathcal{O}(\sqrt{T})$

Proof of Theorem 3.2

Theorem 3.2. *The mixed strategies (D_1, D_2) that Algorithm 1 outputs are $\varepsilon - MNE$, where $\varepsilon := (B_T^{A_1} + B_T^{A_2})/T$ and $B_T^{A_1}$, $B_T^{A_2}$ are bounds on the regret of A_1, A_2.*

Proof. Let $f_t(\mathbf{u}) := \mathbf{M}(\mathbf{u}, \mathbf{v_t})$, $g_t(\mathbf{v}) := \mathbf{M}(\mathbf{u_t}, \mathbf{v})$. Recall that K_1, K_2 are domain of f, g; $B_T^{A_1}$, $B_T^{A_2}$ are bounds on the regret of A_1, A_2. Then

$$\sum_{t=1}^{T} M(\mathbf{u}_t, \mathbf{v}_t) - min_{\mathbf{u}\in K_1} \sum_{t=1}^{T} M(\mathbf{u}, \mathbf{v}_t) \le B_T^{A_1}$$

$$max_{\mathbf{v}\in K_2} \sum_{t=1}^{T} M(\mathbf{u}_t, \mathbf{v}) \ge \mathbb{E}_{v\sim K_2}[\sum_{t=1}^{T} M(\mathbf{u}_t, \mathbf{v})]$$

$$max_{\mathbf{v}\in K_2} \sum_{t=1}^{T} M(\mathbf{u}_t, \mathbf{v}) - \sum_{t=1}^{T} M(\mathbf{u}_t, \mathbf{v}_t) \le B_T^{A_2} \longrightarrow \mathbb{E}_{v\sim K_2}[\frac{1}{T}\sum_{t=1}^{T} M(\mathbf{u}_t, \mathbf{v})] - \frac{1}{T}\sum_{t=1}^{T} M(\mathbf{u}_t, \mathbf{v}_t) \le \frac{1}{T} B_T^{A_2}$$

Proof of Theorem 3.2

Theorem 3.2. *The mixed strategies* (D_1, D_2) *that Algorithm 1 outputs are* $\varepsilon - MNE$*, where* $\varepsilon := (B_T^{A_1} + B_T^{A_2})/T$ *and* $B_T^{A_1}$, $B_T^{A_2}$ *are bounds on the regret of* A_1, A_2.

$$\sum_{t=1}^{T} M(\mathbf{u}_t, \mathbf{v}_t) - min_{\mathbf{u}\in K_1} \sum_{t=1}^{T} M(\mathbf{u}, \mathbf{v}_t) \le B_T^{A_1}$$

$$\mathbb{E}_{v\sim K_2}[\frac{1}{T}\sum_{t=1}^{T} M(\mathbf{u}_t, \mathbf{v})] - \frac{1}{T}\sum_{t=1}^{T} M(\mathbf{u}_t, \mathbf{v}_t) \le \frac{1}{T} B_T^{A_2}$$

$$\mathbb{E}_{v\sim K_2}[\frac{1}{T}\sum_{t=1}^{T} M(\mathbf{u}_t, \mathbf{v})] - min_{\mathbf{u}\in K_1} \frac{1}{T}\sum_{t=1}^{T} M(\mathbf{u}, \mathbf{v}_t) \le \boxed{\frac{1}{T}(B_T^{A_1} + B_T^{A_2})} := \varepsilon$$

Definition 1.1. *Let* $\varepsilon > 0$*, the two distributions* D_1, D_2 *are called* $\varepsilon - MNE$ *if*

$$\mathbb{E}_{(u,v)\sim K_1\times K_2}[M(\boldsymbol{u}, \boldsymbol{v})] \le \min_{\boldsymbol{u}\in K_1} \mathbb{E}_{v\sim D_2}[M(\boldsymbol{u}, \boldsymbol{v})] + \varepsilon$$

$$\mathbb{E}_{(u,v)\sim K_1\times K_2}[M(\boldsymbol{u}, \boldsymbol{v})] \ge \min_{\boldsymbol{v}\in K_2} \mathbb{E}_{u\sim D_1}[M(\boldsymbol{u}, \boldsymbol{v})] - \varepsilon$$

$$\mathbb{E}_{(u,v)\sim K_1\times K_2}[M(\mathbf{u}, \mathbf{v})] \le \min_{\mathbf{u}\in K_1} \mathbb{E}_{v\sim D_2}[M(\mathbf{u}, \mathbf{v})] + \varepsilon$$

Proof of Regret Bounds

$B_T^{A_1}$ is $\mathcal{O}(\sqrt{T})$ *Let* d_2 *is the diameter of* K_2, M *is* $L-$ *Lipschitz.*

$$
\begin{aligned}
Regret_T^{A_1} &\leq \sum_{t=1}^{T}(f_t(u_t) - f_t(u_{t+1})) \\
&= \sum_{t=1}^{T-1}(f_t(u_t) - f_t(u_{t+1}) + f_{t+1}(u_{t+1}) - f_{t+1}(u_{t+1})) + (f_T(u_T) - f_T(u_{T+1})) \\
&= \sum_{t=1}^{T-1}(f_{t+1}(u_{t+1}) - f_t(u_{t+1})) + \sum_{t=1}^{T-1}(f_t(u_t) - f_{t+1}(u_{t+1})) + (f_T(u_T) - f_T(u_{T+1})) \\
&= \sum_{t=1}^{T-1}(M(u_{t+1}, v_{t+1}) - M(u_{t+1}, v_t)) + (f_1(u_1) - f_T(u_{T+1})) \\
&\leq Ld_2\sqrt{T}/\sqrt{2} + (f_1(u_1) - f_T(u_{T+1})) = Ld_2\sqrt{T}/\sqrt{2} + 2C
\end{aligned}
$$

Proof of Regret Bounds

$B_T^{A_2}$ is $\mathcal{O}(\sqrt{T})$ *Let* d_2 *is the diameter of* K_2, M *is* $L-$ *Lipschitz.*

$$Regret_T \leq 2\eta \sum_{t=1}^{T} || \nabla_{f_t} ||^2 + \frac{R(u) - R(x_1)}{\eta}$$

$$= 2\eta L^2 T + \frac{d^2}{\eta} \qquad \eta = \frac{d}{\sqrt{2T}L}$$

$$Regret_T^{A_2}(g_1, \ldots, g_T) \leq L d_2 \sqrt{2T}$$

Algorithm 1 CHEKHOV GAN [4]

Input: Number of steps: T; Game objective: $M(\cdot,\cdot)$
  for $t = 1$ `to` T **do**
    Calculate:
    $\mathbf{u}_t \leftarrow \arg\min_{u \in K_1} \sum_{\tau=0}^{t-1} f_\tau(\mathbf{u})$ $\triangleright A_1$
    $\mathbf{v}_t \leftarrow \arg\max_{v \in K_2} \sum_{\tau=0}^{t-1} \nabla g_\tau(\mathbf{v}_\tau)^T \mathbf{v} - \frac{\sqrt{T}}{2\eta_0} ||\mathbf{v}||^2$ $\triangleright A_2$
    Update:
    $f_t(\cdot) = M(\cdot, \mathbf{v}_t)$; $g_t(\cdot) = M(\mathbf{u}_t, \cdot)$
  end for
Output: $D_1 \sim Uni\{\mathbf{u}_1, ..., \mathbf{u}_T\}$; $D_2 \sim Uni\{\mathbf{v}_1, ..., \mathbf{v}_T\}$

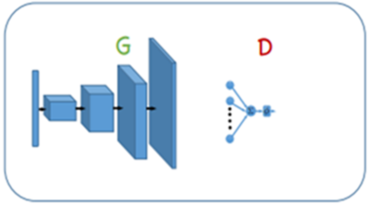
GAN is a minimax zero-sum game.

Standard method aim to find a Pure Nash Equilibrium.

Pure Nash Equilibrium not always exist

We aim to find a Mixed Nash Equilibrium

Finding a Mixed Nash Equilibrium guarantees convergence on Semi-Shallow GAN.



Algorithm 1 CHEKHOV GAN [4]

Input: Number of steps: T; Game objective: $M(\cdot, \cdot)$

for $t = 1$ to T **do**

Calculate:

$\mathbf{u}_t \leftarrow \arg\min_{u \in K_1} \sum_{\tau=0}^{t-1} f_\tau(\mathbf{u})$ $\triangleright A_1$

$\mathbf{v}_t \leftarrow \arg\max_{v \in K_2} \sum_{\tau=0}^{t-1} \nabla g_\tau(\mathbf{v}_\tau)^T \mathbf{v} - \frac{\sqrt{T}}{2\eta_0} ||\mathbf{v}||^2$ $\triangleright A_2$

Update:

$f_t(\cdot) = M(\cdot, \mathbf{v}_t)$; $g_t(\cdot) = M(\mathbf{u}_t, \cdot)$

end for

Output: $D_1 \sim Uni\{\mathbf{u}_1, ..., \mathbf{u}_T\}$; $D_2 \sim Uni\{\mathbf{v}_1, ..., \mathbf{v}_T\}$

Theoretical CHEKHOV GAN does not work in practice!

Practical CHEKHOV GAN

Algorithm 2 Practical CHEKHOV GAN [4], modified

Input: Number of steps: T; Game objective: $M(\cdot,\cdot)$; Number of past states K; Spacing m.
Initialize: Set loss/reward $f_0(\cdot)=0, g_0(\cdot)=0$, and $Q_1.insert(f_0), Q_2.insert(g_0)$

for $t = 1$ `to` T **do**
  $\mathbf{u}_{t+1} \leftarrow u_t - \eta_t \cdot \nabla_{u_t}(\frac{1}{|Q_1|}\sum_{f\in Q_1} f(u) + \frac{C}{\sqrt{t}}||u||^2)$
  $\mathbf{v}_{t+1} \leftarrow v_t - \eta_t \cdot \nabla_{v_t}(\frac{1}{|Q_2|}\sum_{g\in Q_2} g(v) - \frac{C}{\sqrt{t}}||v||^2)$
  Calculate: $f_t(\cdot) = M(\cdot, v_t)$; $g_t(\cdot) = M(u_t, \cdot)$
  Update Q_1 and Q_2:
  if t mod $m == 0$ and $|Q| == K$ **then** $\triangleright\ |Q| = |Q_1| = |Q_2|$
    Q_1.remove_last(); Q_2.remove_last().
    Q_1.insert(f_t); Q_2.insert(g_t).
  else if $|Q| < K$ **then**
    Q_1.insert(f_t); Q_2.insert(g_t).
  else
    Q_1.replace_first(f_t); Q_2.replace_first(g_t).
  end if
end for

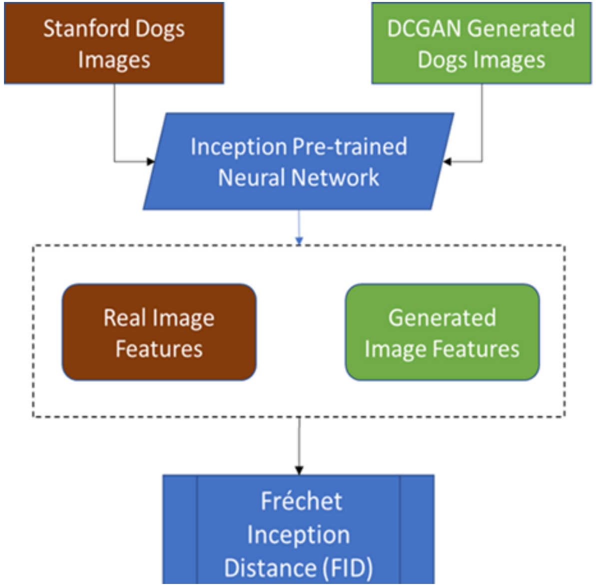
Output: $D_1 \sim Uni\{\mathbf{u}_1, ..., \mathbf{u}_K \in Q_1\}$; $D_2 \sim Uni\{\mathbf{v}_1, ..., \mathbf{v}_K \in Q_2\}$

Experiments

Data set:
Cifar-10 (paper)
Stanford Dogs Images (mine)

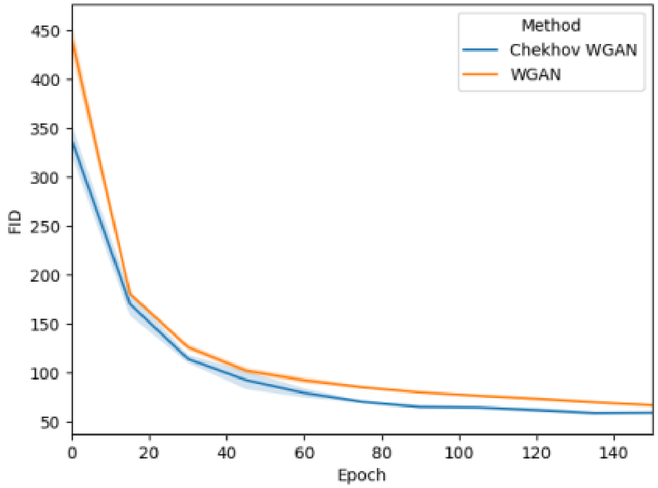
Evaluation metrics:
Fréchet Inception Distance

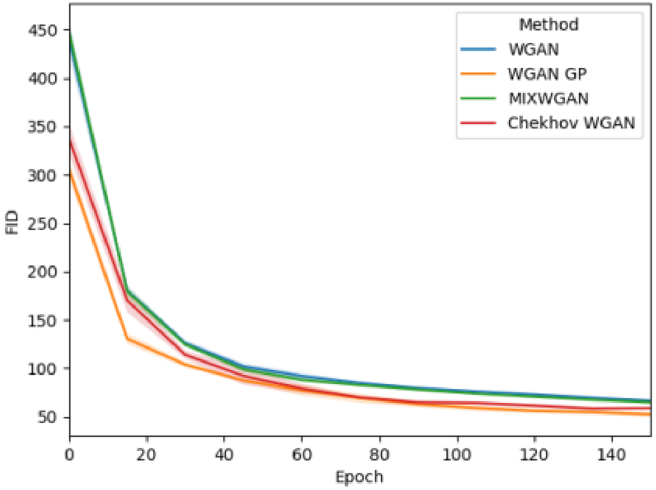
$$FID = \|\mu_r - \mu_s\|^2 + Tr\left(\Sigma_r + \Sigma_g - 2\sqrt{\Sigma_r\Sigma_g}\right)$$



Experiments

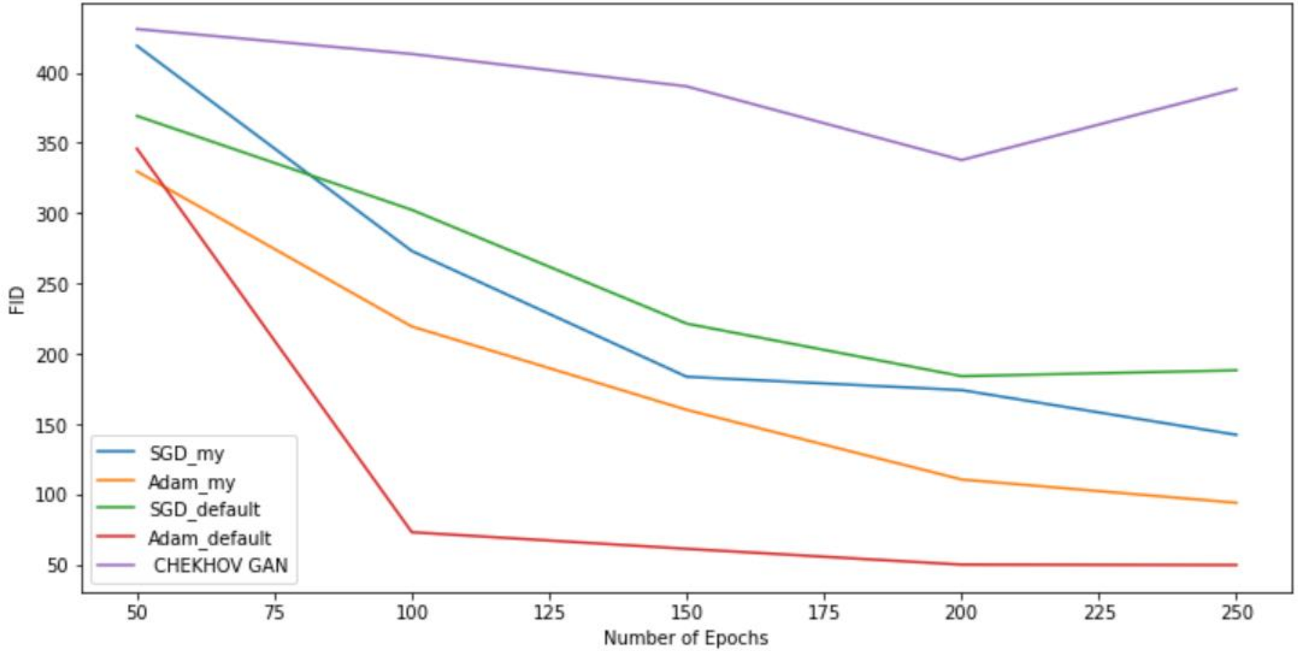
Cifar-10 (paper)





Experiments

Stanford Dogs Images (mine)



Discussion

- Converge but the speed?
- Experiments on Semi-Shallow GAN that show standard GAN training method does not converge but CHEKHOV GAN converges.
- The practical Chekhov GAN is using gradients and is very different from the theoretical Chekhov GAN, so does the algorithm still guarantee convergence?
- The proposed algorithm requires significantly large GPU memory and is also computational.
- The experiments in the original paper do not show many advantages of Checkhov GAN, though Chekhov WGAN consistently outperforms WGAN by achieving better FID scores.
- Proof convergence of Deep GAN architecture.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pages 214–223. PMLR, 06–11 Aug 2017.

[2] Jason Brownlee. How to implement the frechet inception distance (fid) for evaluating gans, 2019. [Online]. Available: https://machinelearningmastery.com/how-to-implement-the-frechet inception-distance-fid-from-scratch/.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. Advances in neural information processing systems, 27, 2014.

[4] Paulina Grnarova, Kfir Y Levy, Aurelien Lucchi, Thomas Hofmann, and Andreas Krause. An online learning approach to generative adversarial networks. In International Conference on Learning Representations, 2018.

[5] Elad Hazan. Introduction to online convex optimization. Foundations and Trends in Optimization, 2(3-4):157–325, 2016.

[6] Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. Journal of Computer and System Sciences, 71(3):291–307, 2005.

[7] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[8] John F Nash et al. Equilibrium points in n-person games. Proceedings of the national academy of sciences, 36(1):48–49, 1950.

[9] Francesco Orabona. A modern introduction to online learning. arXiv preprint arXiv:1912.13213, 2019.

[10] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. Advances in neural information processing systems, 29:2234–2242, 2016.

[11] Kaggle Team. Generative dog images - experiment with creating puppy pics | kaggle., 2020. [Online]. Available: https://www.kaggle.com/c/generative-dog-images/overview/evaluation.

Thanks!

Questions?

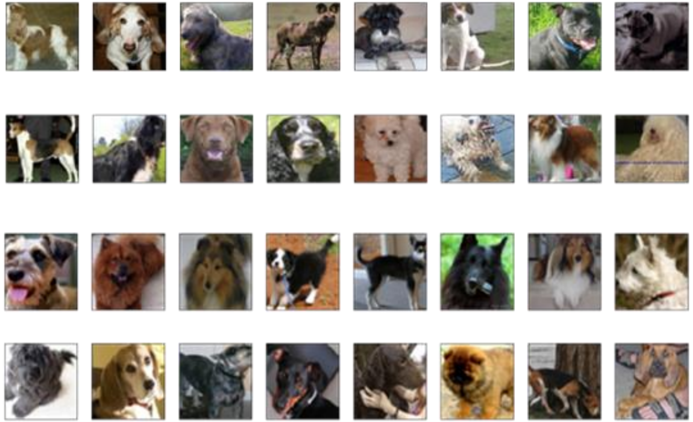


Fig.1 Image samples from Stanford Dog Dataset.

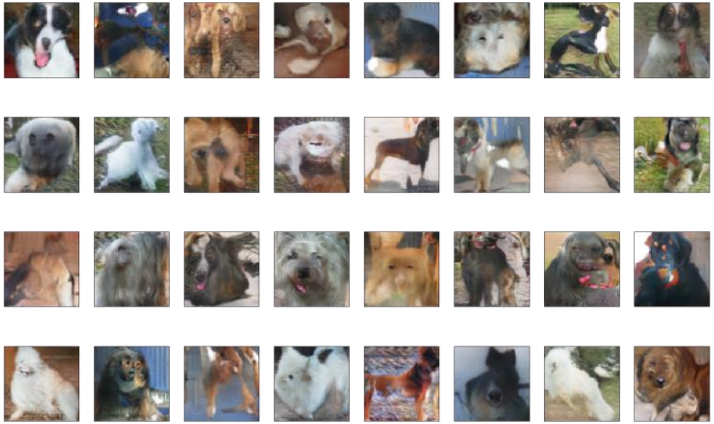


Fig.2 Generated using my implementation of Adam.

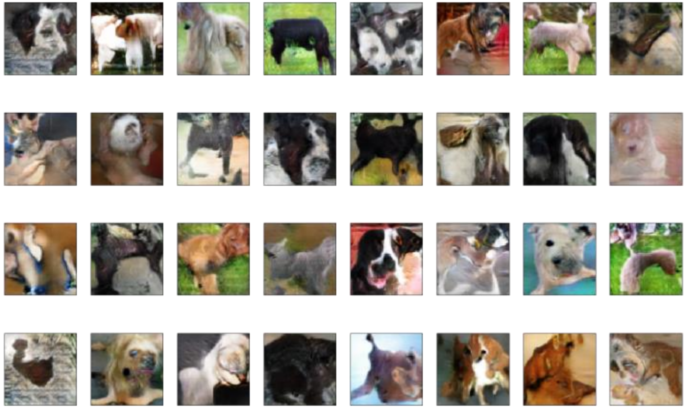


Fig.3 Generated using my implementation of SGD.

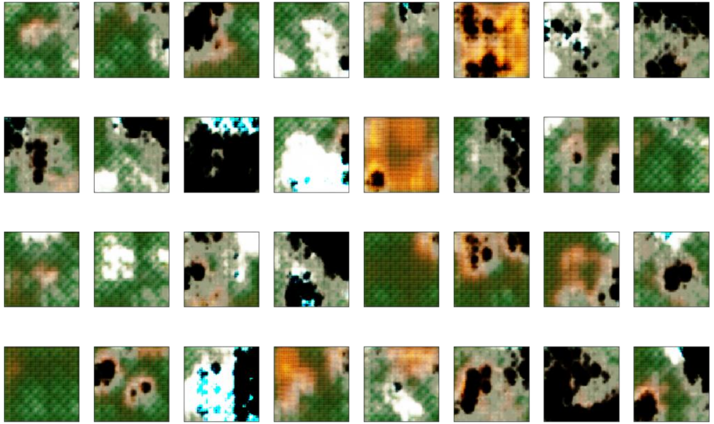


Fig.4 Generated using my implementation of CHECKOV GAN.

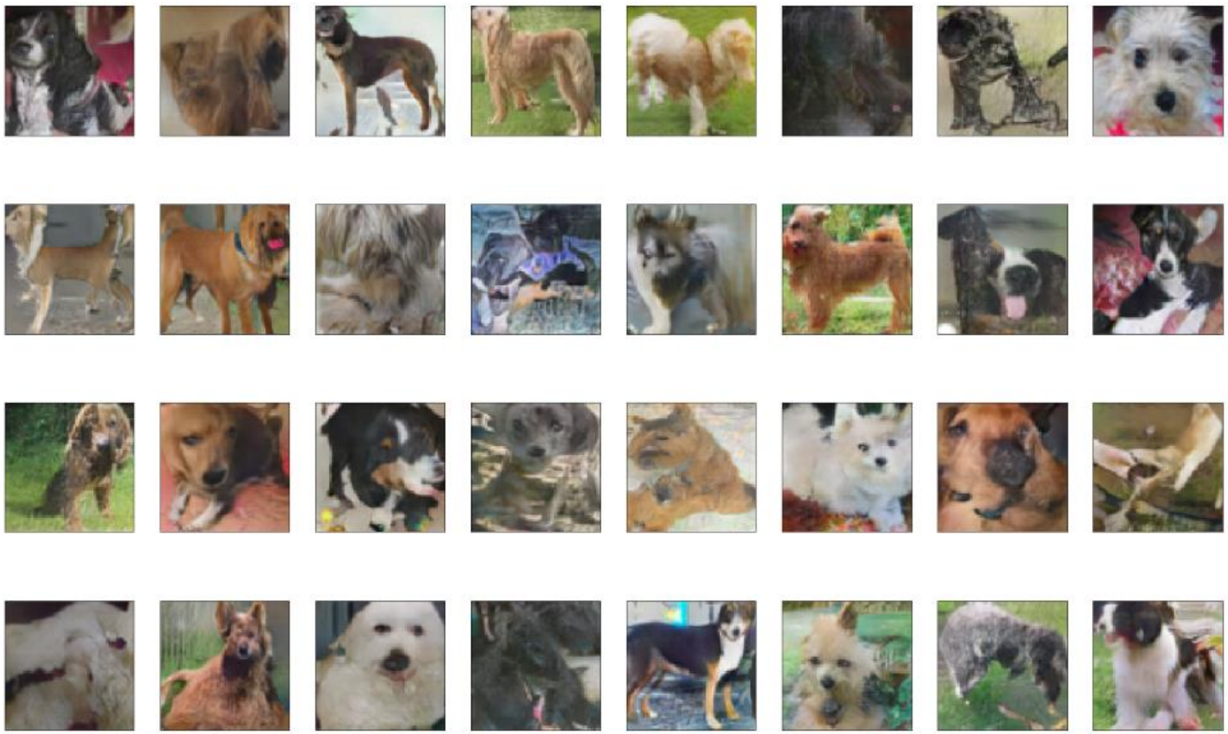


Figure 5: Result of DCGAN training using default Adam optimizer.

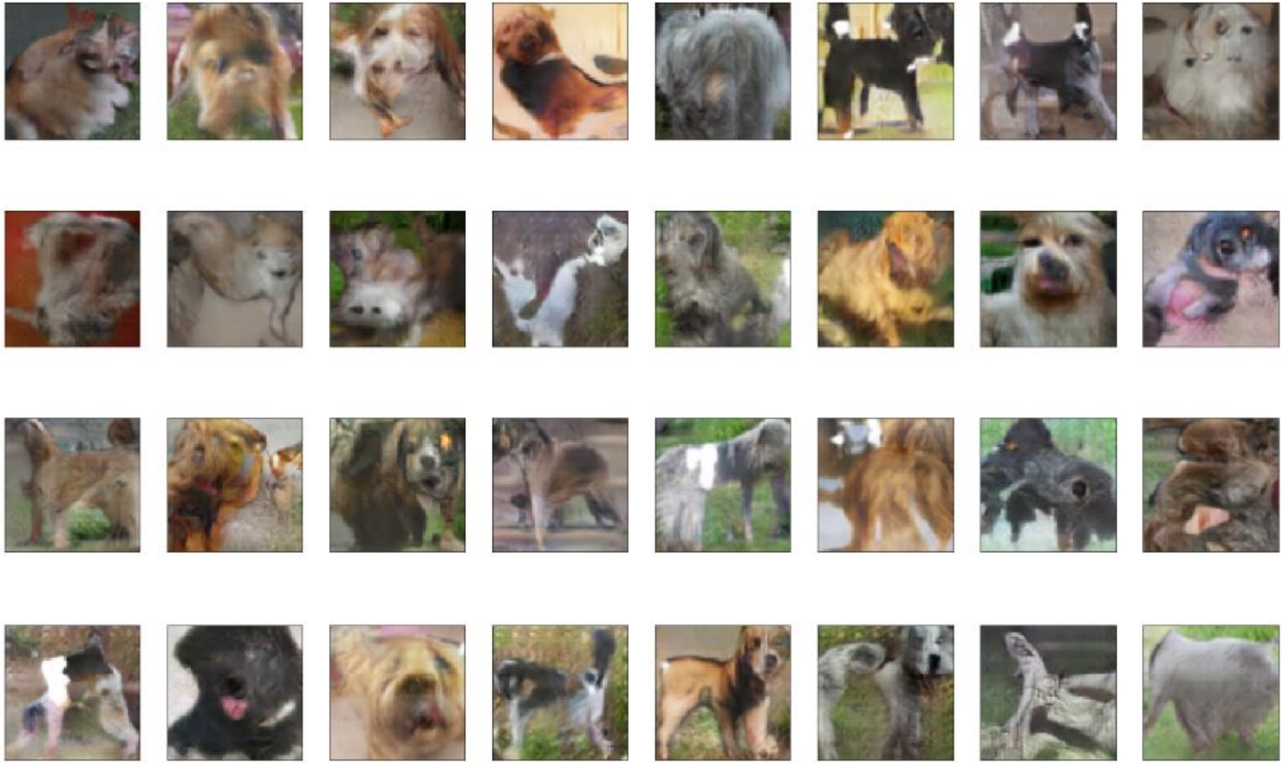
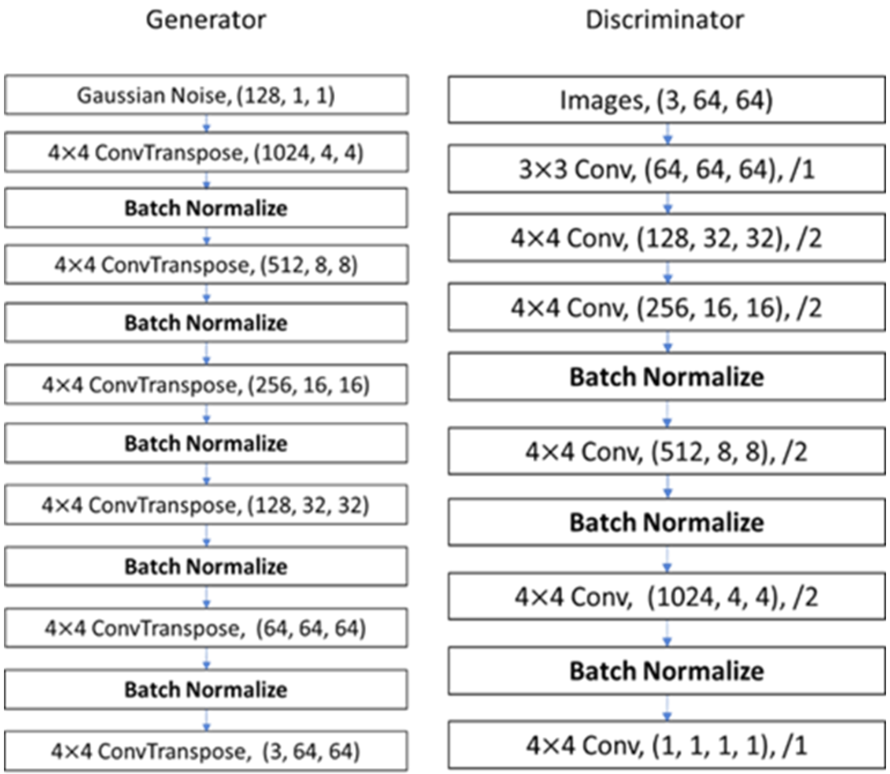


Figure 7: Result of DCGAN training using default SGD optimizer.

Thanks!

Questions?


Generator
Gaussian Noise, (128, 1, 1)
4×4 ConvTranspose, (1024, 4, 4)
Batch Normalize
4×4 ConvTranspose, (512, 8, 8)
Batch Normalize
4×4 ConvTranspose, (256, 16, 16)
Batch Normalize
4×4 ConvTranspose, (128, 32, 32)
Batch Normalize
4×4 ConvTranspose, (64, 64, 64)
Batch Normalize
4×4 ConvTranspose, (3, 64, 64)
Discriminator
Images, (3, 64, 64)
3×3 Conv, (64, 64, 64), /1
4×4 Conv, (128, 32, 32), /2
4×4 Conv, (256, 16, 16), /2
Batch Normalize
4×4 Conv, (512, 8, 8), /2
Batch Normalize
4×4 Conv, (1024, 4, 4), /2
Batch Normalize
4×4 Conv, (1, 1, 1, 1), /1