Progressive Growth of Generative Adversarial Networks

Stanisław Wilczyński

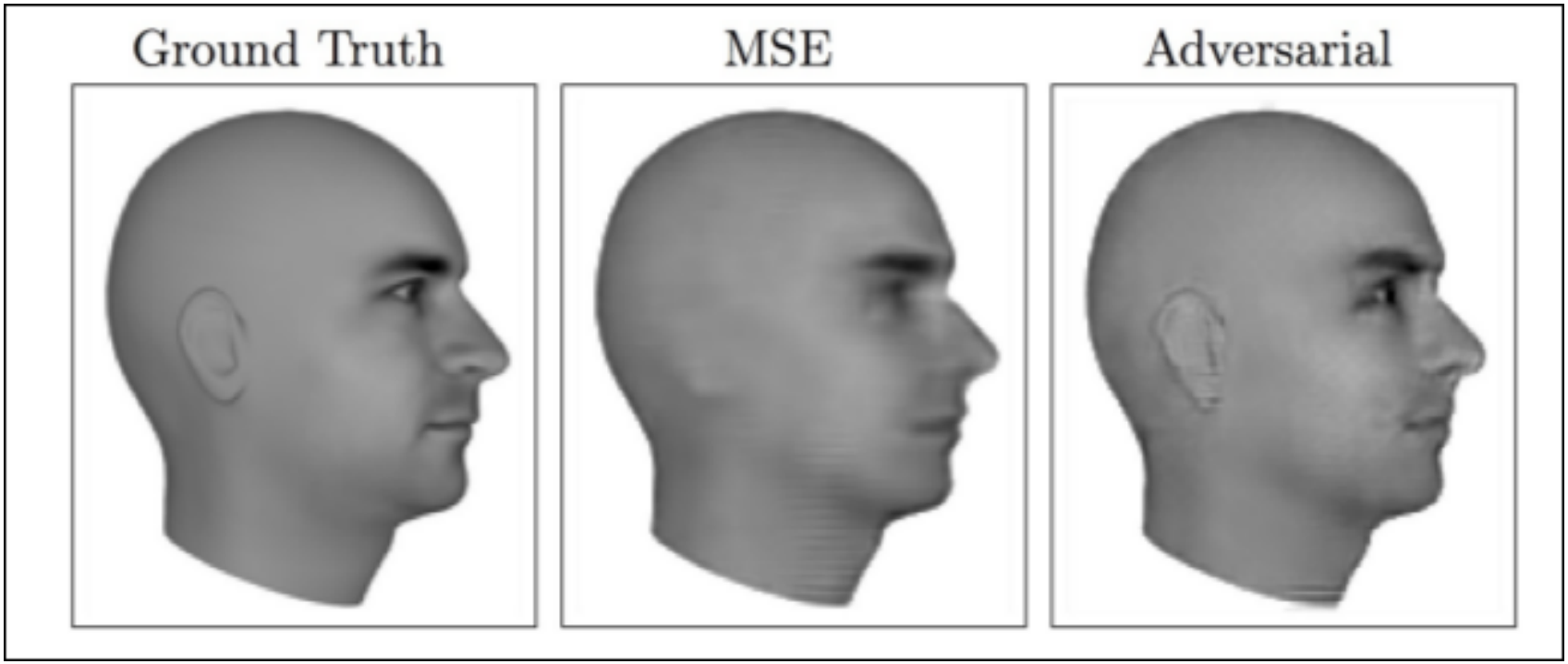
Uniwersytet Wrocławski

09.03.2018

GAN - przypomnienie

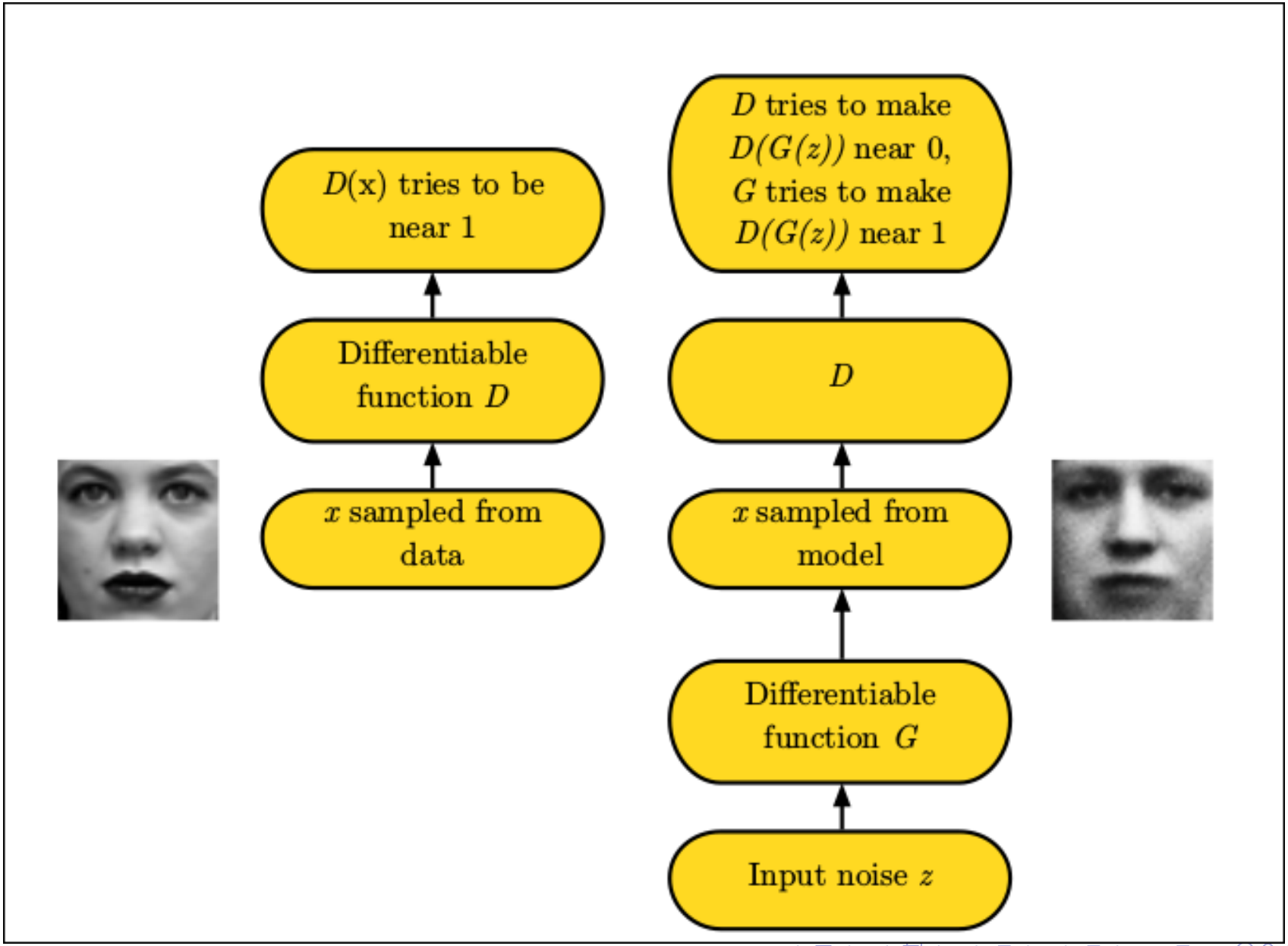
- Modelowanie wysoko-wymiarowych rozkładów prawdopodobieństwa
- Uczenie ze wzmocnieniem - symulowanie przyszłości dla naszego agenta
- Brakujące dane i semi-supervised learning
- Rozkłady wielomodalne - wiele możliwych poprawnych odpowiedzi
- Zwiększanie rozdzielczości zdjęć
- Generowanie zawartości - zdjęcia, muzyka, mowa

Przykład 1 - generowanie klatek



Przykład 2 - wysoka rozdzielczość

Dyskryminator i generator



Kara, minimax i pierwsze ulepszenia

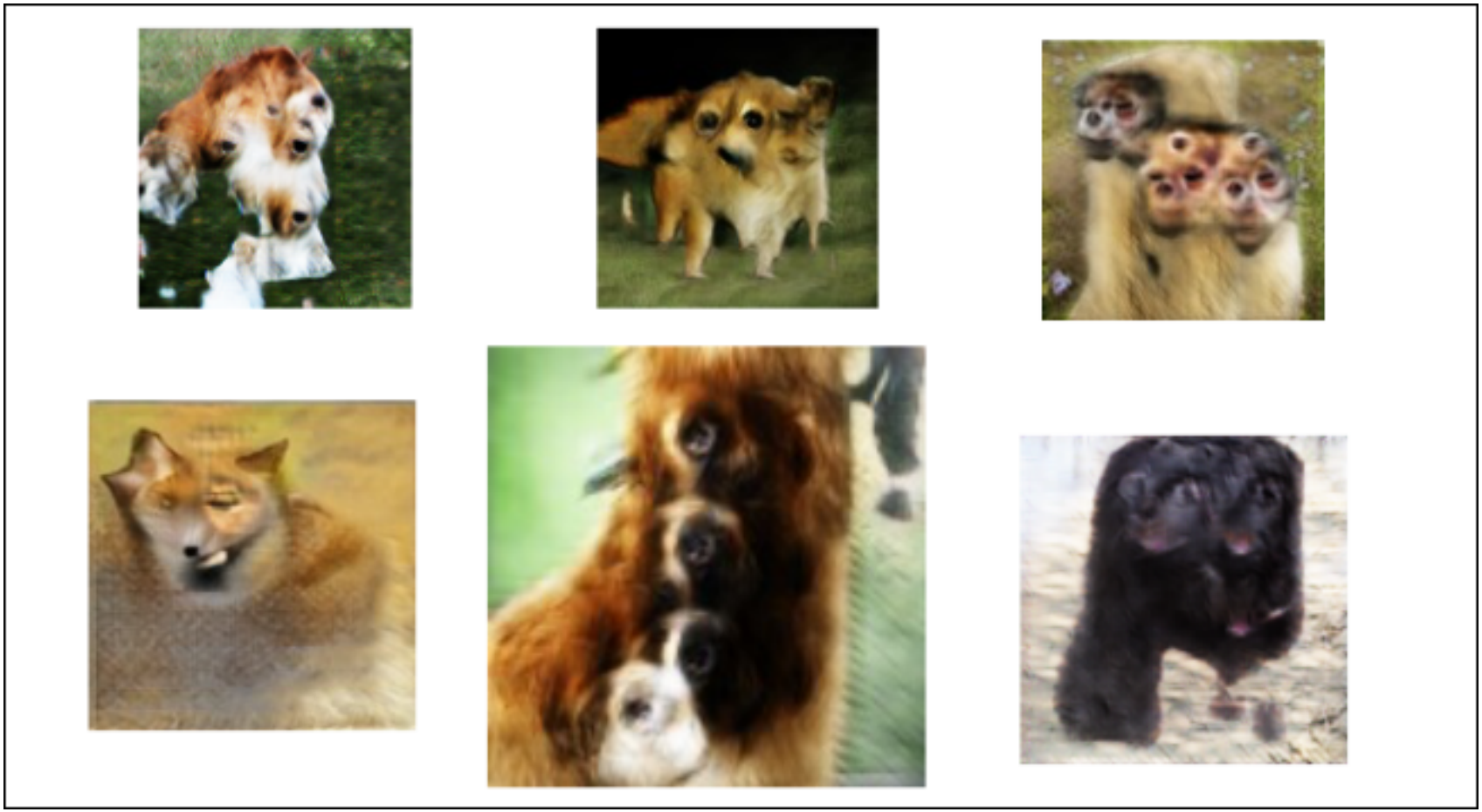
$$J^{(D)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2}\mathbf{E}_{x\sim p_{data}} \log D(x) - \frac{1}{2}\mathbf{E}_z \log(1 - D(G(z))$$

$$J^{(G)} = -J^{(D)}$$

$$J^{(G)} = -\frac{1}{2}\mathbf{E}_z \log D(G(z))$$

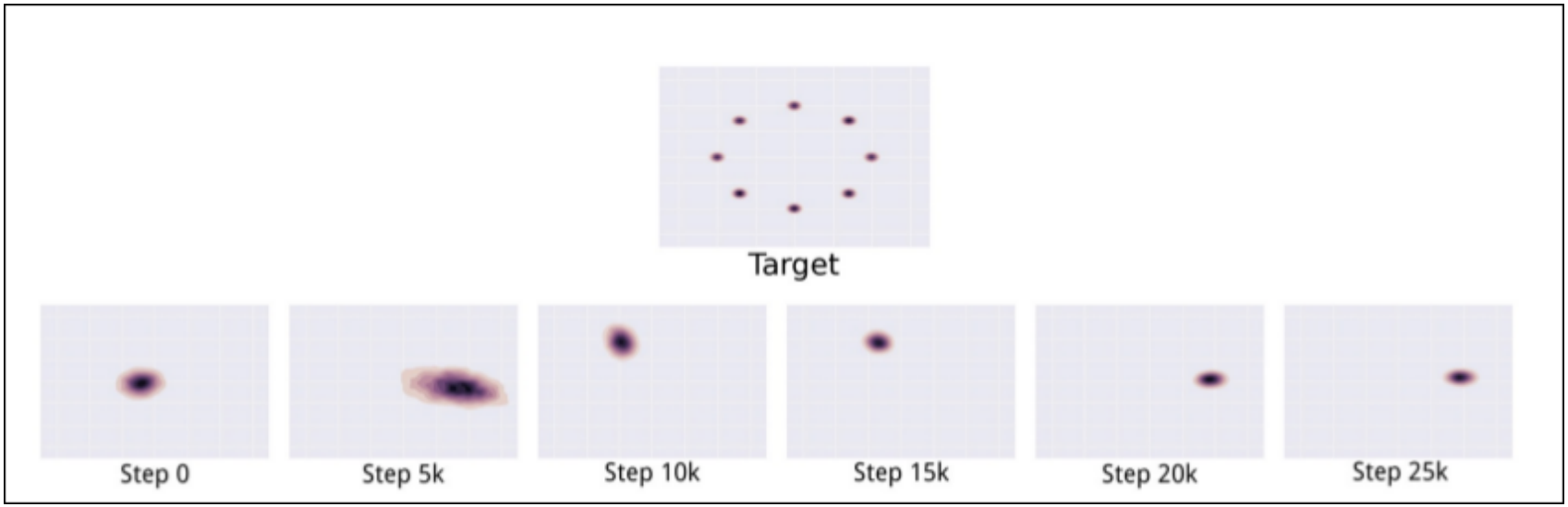
- Ostatnie równanie to próba pozbycia się problemu znikającego gradientu kiedy dyskryminator jest zbyt dobry
- Równe minibatche
- Kroki gradientowe wykonujemy na zmianę

Problem 1 - liczba oczu



Problem 2 - struktura globalna

Problem 3 - przeskakiwanie między modami



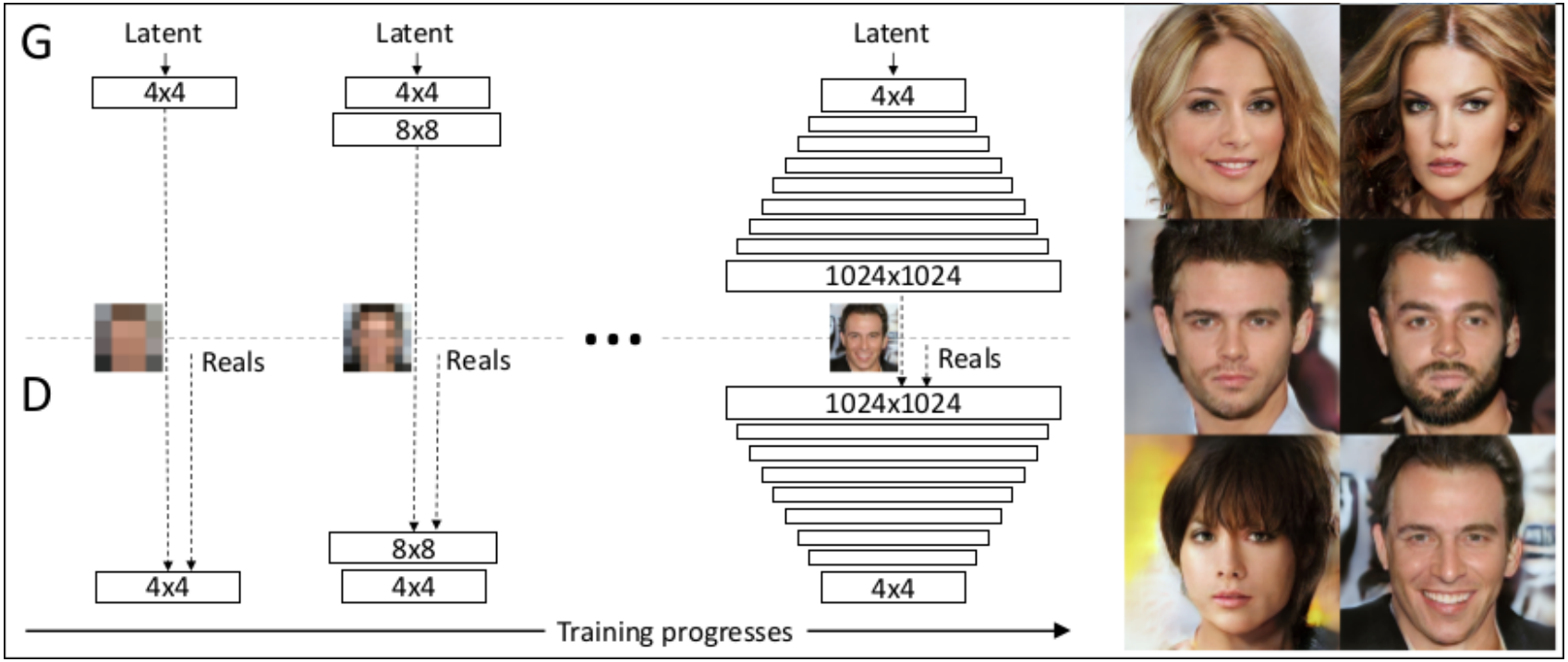
Pozostałe problemy

- Zbyt konkretne odpowiedzi generatora
- Generowanie obrazów w wysokiej rozdzielczości (łatwiej poznać fałszywkę)
- Mała różnorodność wyjść z generatora (problem z modami)
- GAN są wrażliwe na duże sygnały (przy ReLU)

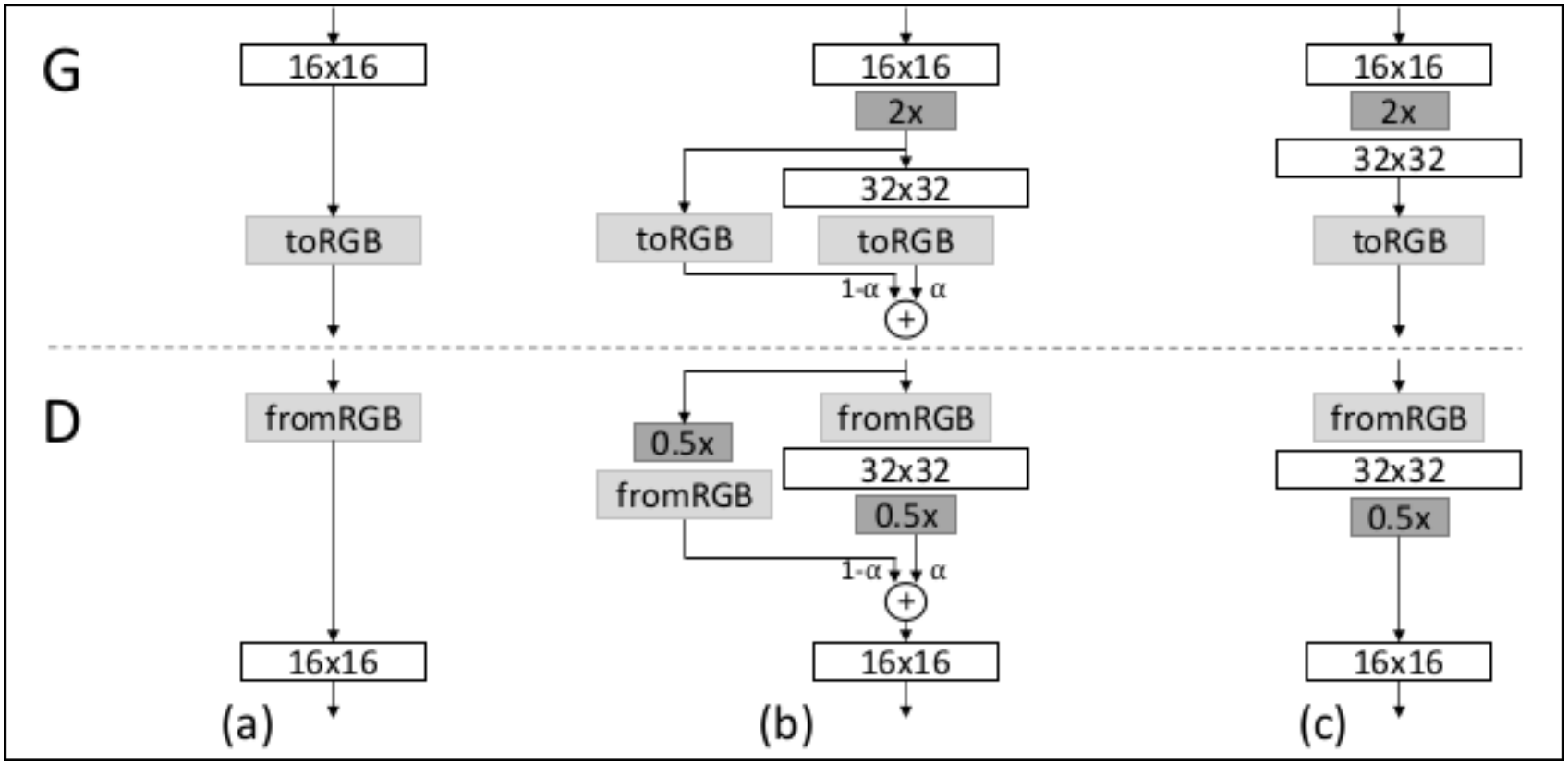
Co potrafią ulepszone GANy?

- Generowanie twarzy celebrytów w wysokiej rozdzielczości
- Realistyczne zdjęcia z CIFAR10, LSUN
- Przyspieszenie treningu

Stopniowy wzrost



Gładkie wprowadzenie warstwy



Zwiększenie różnorodności

- Idea: feature mapy ze statystykami z całego minibatcha
- Uzasadnienie: odróżnianie modów
- Poprzednie prace: dodatkowa warstwa z nauczalnymi wagami do obliczania statystyk
- Tutaj: liczymy odchylenie standardowe z każdej cechy (piksela) w każdej lokalizacji i bierzemy średnią ze wszystkich do S, a nowa feature mapa to macierz wypełniona S

Normalizacja

- Ochrona przed zbyt dużym sygnałem (ReLU) - standardowo batch normalization i odpowiednia incjalizacja
- Wyrównany learning rate - zamiast inicjalizować uważnie wagi, losujemy z $N(0,1)$. Następnie, zawsze gdy używamy danej wagi skalujemy ją $\hat{w}_i = \frac{w_i}{c}$
- c różne dla każdej warstwy
- Efekt: Odpowiednio przeskalowany update, lepsza szybkość uczenia
- Normalizacja cech per piksel w generatorze
- Wzór

$$b_{x,y} = a_{x,y} / \sqrt{\frac{1}{N} \sum_{j=1}^{N} (a^j_{x,y})^2 + \epsilon}$$

Ocenianie skuteczności generatora - MS-SSIM

- Multi-Scale Structural Similarity (MS-SSIM)
- Porównywanie statystyk między obrazkami
- Dobrze radzi sobie z wykrywaniem braku modów
- Słabo z wykrywaniem różnorodności tekstur i kolorów

Ocenianie skuteczności generatora - SWD

- Dystans Wassersteina - jak duży jest koszt przeniesienia jednej chmury punktów na drugą

$$W(X,Y)^2 = \min_{\sigma} \sum_{i=1}^{N} ||X_i - Y_{\sigma(i)}||^2$$

- Sliced Wasserstein Distance (SWD) - w celu zredukowania kosztów obliczeń, aproksymujemy dystans Wassersteina rzutami na wektory długości 1 (w jednowymiarowym przypadku jest wzór na najlepszą permutację) i bierzemy średnią

$$W(\hat{X},Y) = \int_{\theta\in\Theta} \min_{\sigma} \sum_{i=1}^{N} | < X_i - Y_{\sigma(i)}, \theta > |^2 d\theta$$

Ocenianie skuteczności generatora - piramida Laplace'a

- Piramida Laplace'a - poziomy rozdzielczości zdjęcia (od 16×16 do 128×128)
- Przebig oceny
  1. Losujemy 16384 obrazków i dla każdego z nich i każdego poziomu piramidy bierzemy 128 jego deskryptorów (wycinki wielkości $7 \times 7 \times 3$)
  2. Normalizujemy deskryptory przez średnią i odchylenie standardowe na każdym kolorze
  3. Obliczamy $SWD(\{x_i\}, \{y_i\})$

Użyte kary

- Przyczyna: zredukowanie szansy na znikanie gradientu
- Least Squares GAN - logarytm w karze zastępujemy kwadratem
- Wasserstein GAN - Gradient Penalty - szukanie dyskryminatora odpowiedniej postaci, a dodatkowa kara do WGAN, żeby tą postać wymusić

Twarze celebrytów

- CelebA - nieustandaryzowany zbiór obrazów celebrytów (wiele osób, rożne profile, szumy)

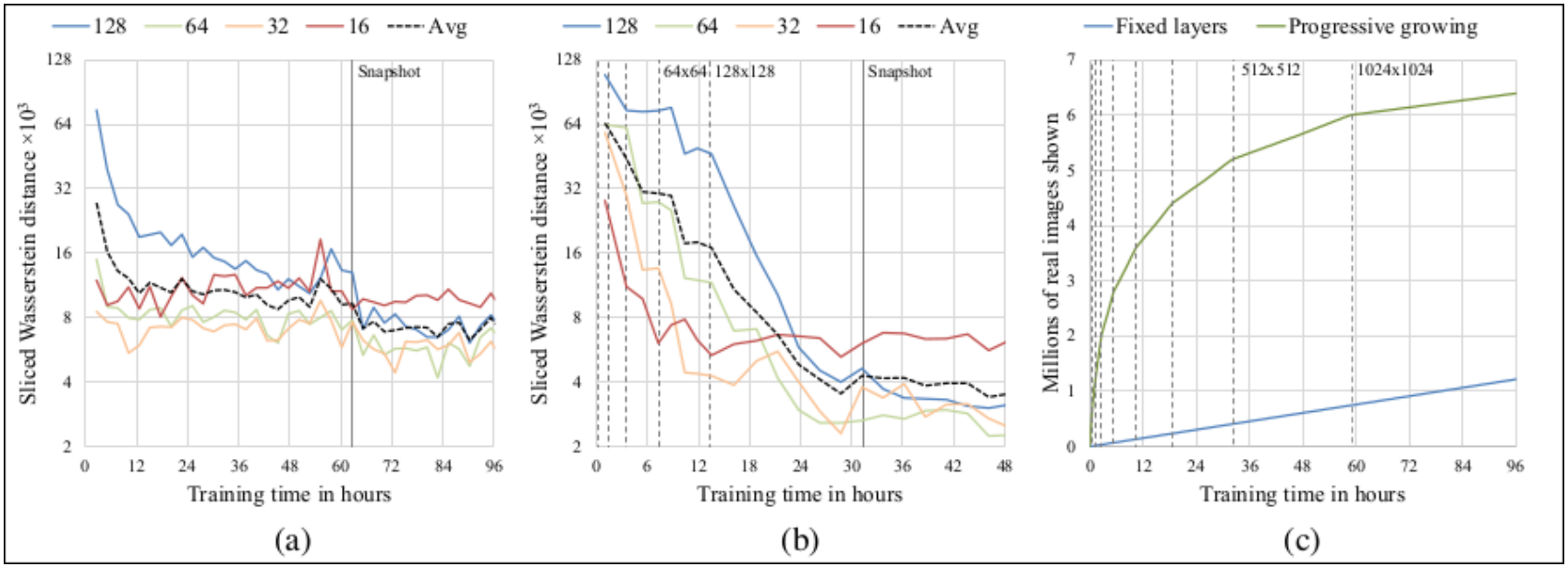
Generator	Act.	Output shape	Params
Latent vector	–	512 × 1 × 1	–
Conv 4 × 4	LReLU	512 × 4 × 4	4.2M
Conv 3 × 3	LReLU	512 × 4 × 4	2.4M
Upsample	–	512 × 8 × 8	–
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Upsample	–	512 × 16 × 16	–
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Upsample	–	512 × 32 × 32	–
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Upsample	–	512 × 64 × 64	–
Conv 3 × 3	LReLU	256 × 64 × 64	1.2M
Conv 3 × 3	LReLU	256 × 64 × 64	590k
Upsample	–	256 × 128 × 128	–
Conv 3 × 3	LReLU	128 × 128 × 128	295k
Conv 3 × 3	LReLU	128 × 128 × 128	148k
Upsample	–	128 × 256 × 256	–
Conv 3 × 3	LReLU	64 × 256 × 256	74k
Conv 3 × 3	LReLU	64 × 256 × 256	37k
Upsample	–	64 × 512 × 512	–
Conv 3 × 3	LReLU	32 × 512 × 512	18k
Conv 3 × 3	LReLU	32 × 512 × 512	9.2k
Upsample	–	32 × 1024 × 1024	–
Conv 3 × 3	LReLU	16 × 1024 × 1024	4.6k
Conv 3 × 3	LReLU	16 × 1024 × 1024	2.3k
Conv 1 × 1	linear	3 × 1024 × 1024	51
Total trainable parameters			**23.1M**

Discriminator	Act.	Output shape	Params
Input image	–	3 × 1024 × 1024	–
Conv 1 × 1	LReLU	16 × 1024 × 1024	64
Conv 3 × 3	LReLU	16 × 1024 × 1024	2.3k
Conv 3 × 3	LReLU	32 × 1024 × 1024	4.6k
Downsample	–	32 × 512 × 512	–
Conv 3 × 3	LReLU	32 × 512 × 512	9.2k
Conv 3 × 3	LReLU	64 × 512 × 512	18k
Downsample	–	64 × 256 × 256	–
Conv 3 × 3	LReLU	64 × 256 × 256	37k
Conv 3 × 3	LReLU	128 × 256 × 256	74k
Downsample	–	128 × 128 × 128	–
Conv 3 × 3	LReLU	128 × 128 × 128	148k
Conv 3 × 3	LReLU	256 × 128 × 128	295k
Downsample	–	256 × 64 × 64	–
Conv 3 × 3	LReLU	256 × 64 × 64	590k
Conv 3 × 3	LReLU	512 × 64 × 64	1.2M
Downsample	–	512 × 32 × 32	–
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Downsample	–	512 × 16 × 16	–
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Downsample	–	512 × 8 × 8	–
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Downsample	–	512 × 4 × 4	–
Minibatch stddev	–	513 × 4 × 4	–
Conv 3 × 3	LReLU	512 × 4 × 4	2.4M
Conv 4 × 4	LReLU	512 × 1 × 1	4.2M
Fully-connected	linear	1 × 1 × 1	513
Total trainable parameters			**23.1M**

Wyniki na CelebA

Training configuration	CELEBA						LSUN BEDROOM					
	Sliced Wasserstein distance $\times 10^3$					MS-SSIM	Sliced Wasserstein distance $\times 10^3$					MS-SSIM
	128	64	32	16	Avg		128	64	32	16	Avg	
(a) Gulrajani et al. (2017)	12.99	7.79	7.62	8.73	9.28	0.2854	11.97	10.51	8.03	14.48	11.25	**0.0587**
(b) + Progressive growing	4.62	**2.64**	3.78	6.06	4.28	**0.2838**	7.09	6.27	7.40	9.64	7.60	0.0615
(c) + Small minibatch	75.42	41.33	41.62	26.57	46.23	0.4065	72.73	40.16	42.75	42.46	49.52	0.1061
(d) + Revised training parameters	9.20	6.53	4.71	11.84	8.07	0.3027	7.39	5.51	3.65	9.63	6.54	0.0662
(e*) + Minibatch discrimination	10.76	6.28	6.04	16.29	9.84	0.3057	10.29	6.22	5.32	11.88	8.43	0.0648
(e) Minibatch stddev	13.94	5.67	2.82	5.71	7.04	0.2950	7.77	5.23	3.27	9.64	6.48	0.0671
(f) + Equalized learning rate	4.42	3.28	2.32	7.52	4.39	0.2902	**3.61**	3.32	**2.71**	6.44	4.02	0.0668
(g) + Pixelwise normalization	**4.06**	3.04	**2.02**	**5.13**	**3.56**	0.2845	3.89	**3.05**	3.24	**5.87**	**4.01**	0.0640
(h) Converged	2.42	2.17	2.24	4.99	2.96	0.2828	3.47	2.60	2.30	4.87	3.31	0.0636

Czas treningu



Twarze celebrytów - CelebA-HQ

Pozostałe przykłady

- Zdjęcia z Appendixu H
- Film - prezentacja wyników

Bibliografia


Ian J. Goodfellow.
NIPS 2016 tutorial: Generative adversarial networks.
CoRR, abs/1701.00160, 2017.


Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen.
Progressive growing of gans for improved quality, stability, and variation.
CoRR, abs/1710.10196, 2017.


Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot.
Wasserstein barycenter and its application to texture mixing.
In Alfred M. Bruckstein, Bart M. ter Haar Romeny, Alexander M. Bronstein, and Michael M. Bronstein, editors, *SSVM*, volume 6667 of *Lecture Notes in Computer Science*, pages 435–446. Springer, 2011.