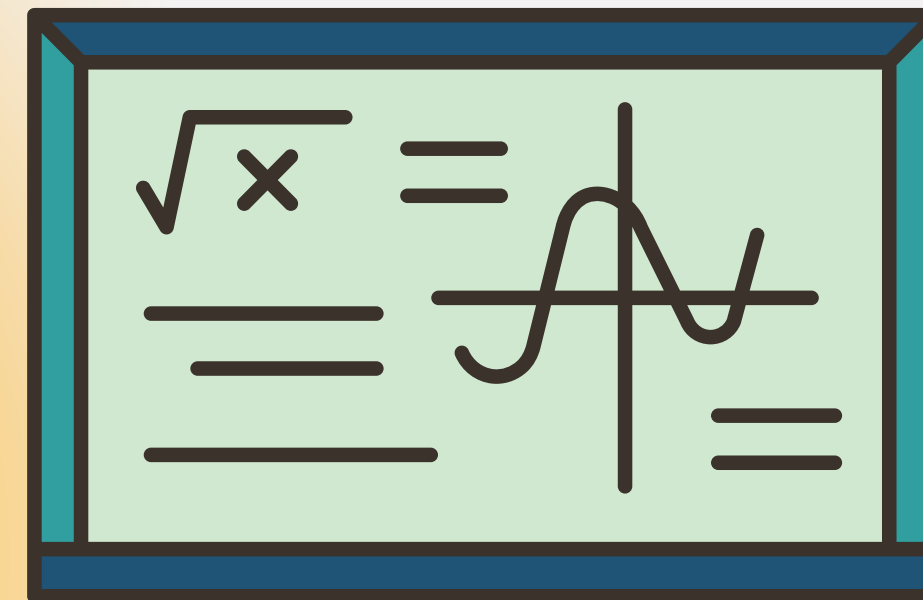




FaceFusion

Latent-Interpolations



Liza Kaladjian



Inhaltsverzeichnis

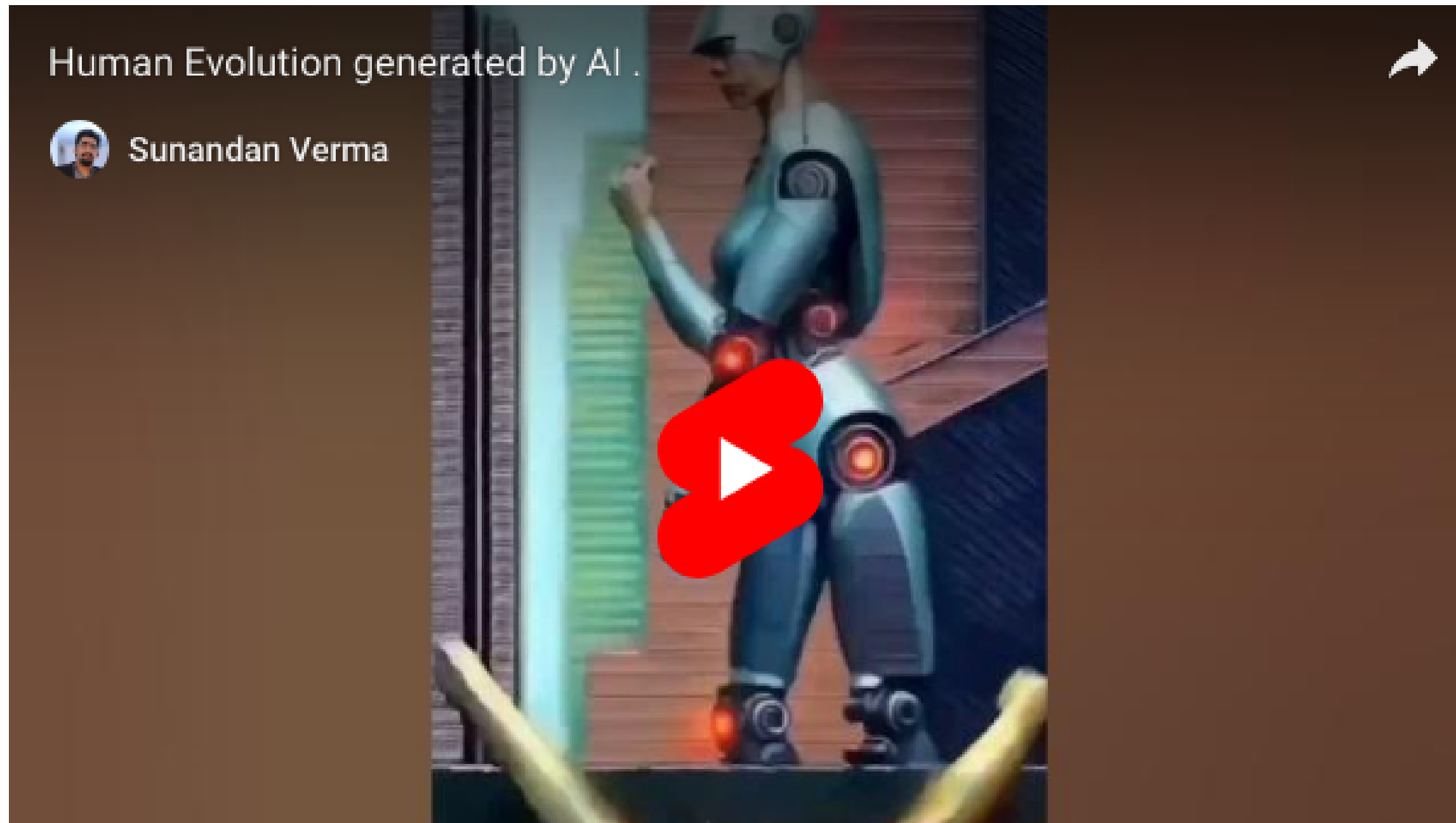
- 01 - Ein kleines Spiel
- 02 - Ziel und Motivation des Projekts
- 03 - Was sind GANs?
- 04 - Mathematische Grundlagen von GANs
- 05 - WGANs
- 06 - Training
- 07 - Live Demo

Ein kleines Spiel

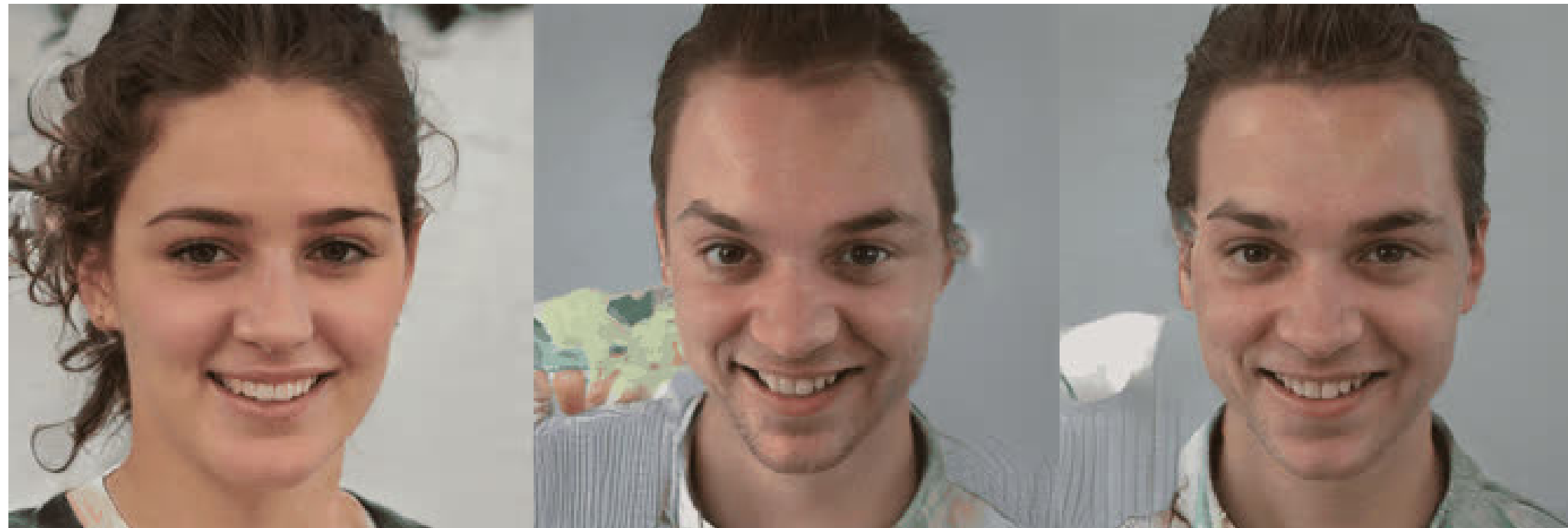
Let's Play



Ziel und Motivation des Projekts



Ziel und Motivation des Projekts

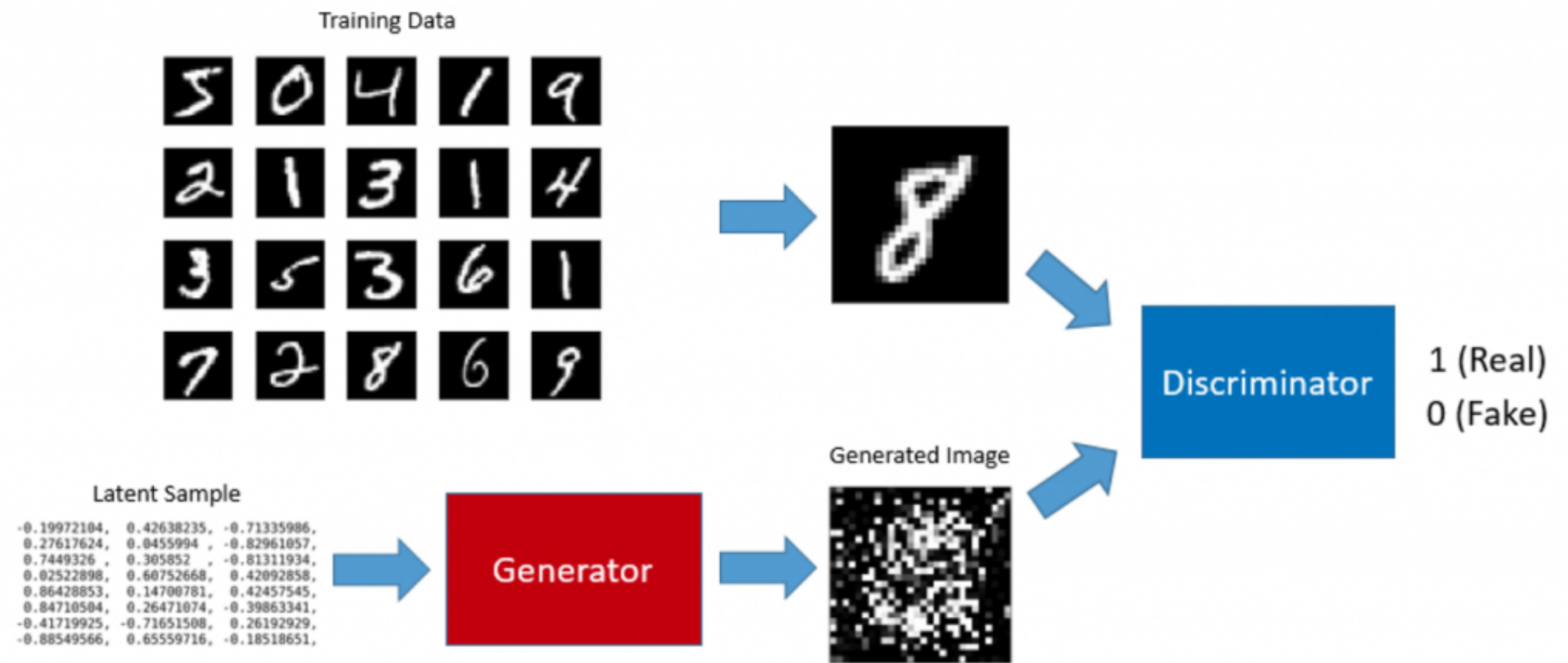


Was sind GANs?

- Generator erzeugt Bilder
- Diskriminator bewertet sie.



Beide Netzwerke
verbessern sich
gegenseitig.



https://www.researchgate.net/figure/Generated-images-by-GAN-and-WGAN-models-trained-on-MNIST-after-1-100k-500k-1000k_fig2_329705388

Mathematische Grundlagen von GANs

Diskriminatorverlust:

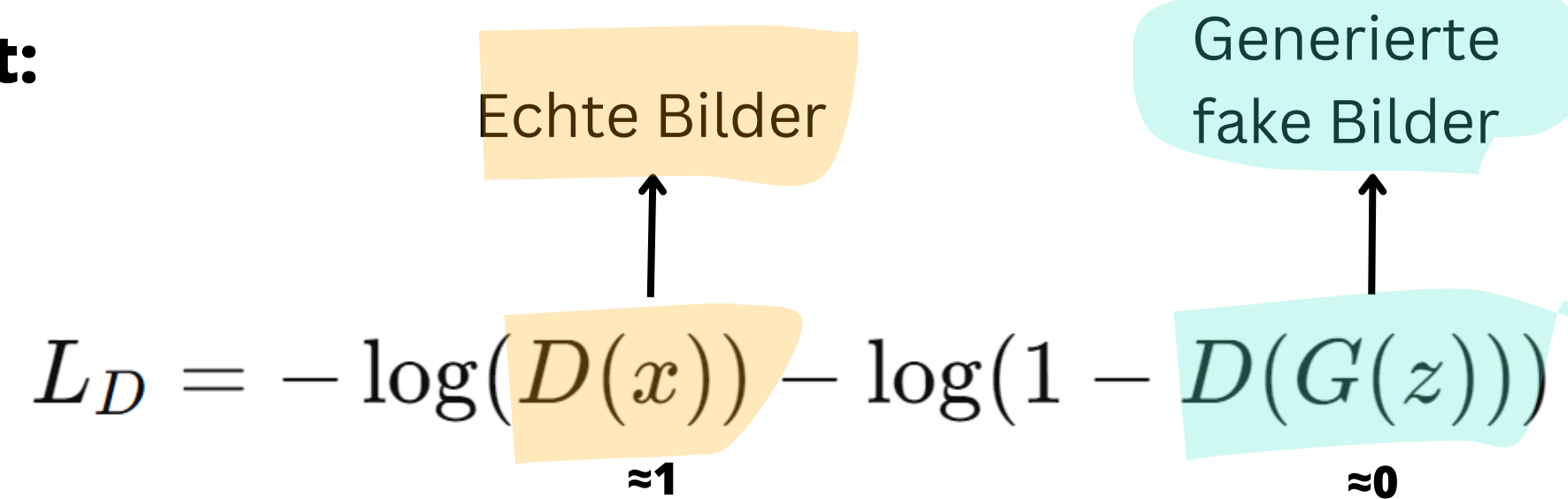
$$L_D = -\log(D(x)) - \log(1 - D(G(z)))$$

Generatorverlust:

$$L_G = -\log(D(G(z)))$$

Mathematische Grundlagen von GANs

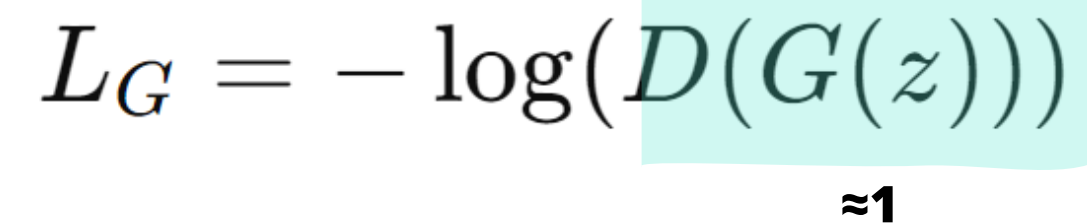
Diskriminatorverlust:



The diagram illustrates the components of the Discriminator Loss equation. The equation is $L_D = -\log(D(x)) - \log(1 - D(G(z)))$. The term $D(x)$ is highlighted in orange, with an arrow pointing to the label 'Echte Bilder' (Real Images) above it. Below $D(x)$ is the approximation ≈ 1 . The term $D(G(z))$ is highlighted in teal, with an arrow pointing to the label 'Generierte fake Bilder' (Generated fake Images) above it. Below $D(G(z))$ is the approximation ≈ 0 .

$$L_D = -\log(\underbrace{D(x)}_{\approx 1}) - \log(1 - \underbrace{D(G(z))}_{\approx 0})$$

Generatorverlust:

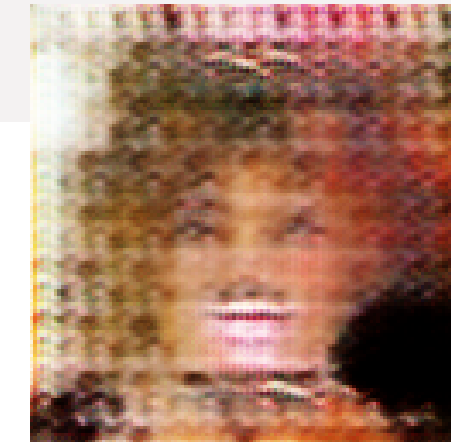


The diagram illustrates the Generator Loss equation. The equation is $L_G = -\log(D(G(z)))$. The term $D(G(z))$ is highlighted in teal, with an arrow pointing to the approximation ≈ 1 below it.

$$L_G = -\log(\underbrace{D(G(z))}_{\approx 1})$$

Mathematische Grundlagen von GANs

Diskriminatorverlust:



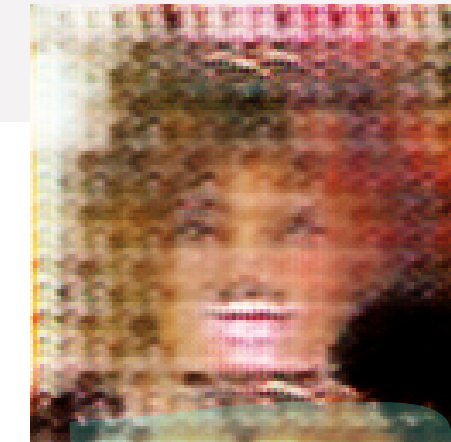
$$L_D = -\log(D(x)) - \log(1 - D(G(z)))$$

Generatorverlust:

$$L_G = -\log(D(G(z)))$$

Mathematische Grundlagen von GANs

Diskriminatorverlust:



$$L_D = -\log(D(x)) - \log(1 - D(G(z)))$$

$$L_D = -\log(0.9) - \log(1 - 0.2) = -\log(0.9) - \log(0.8)$$

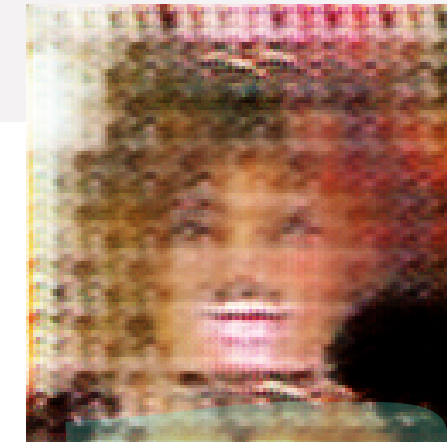
$$\log(0.9) \approx -0.045, \quad \log(0.8) \approx -0.097, \quad L_D = 0.142$$

Generatorverlust:

$$L_G = -\log(D(G(z)))$$

Mathematische Grundlagen von GANs

Diskriminatorverlust:



$$L_D = -\log(D(x)) - \log(1 - D(G(z)))$$

$$L_D = -\log(0.9) - \log(1 - 0.2) = -\log(0.9) - \log(0.8)$$

$$\log(0.9) \approx -0.045, \quad \log(0.8) \approx -0.097, \quad L_D = 0.142$$

Generatorverlust:

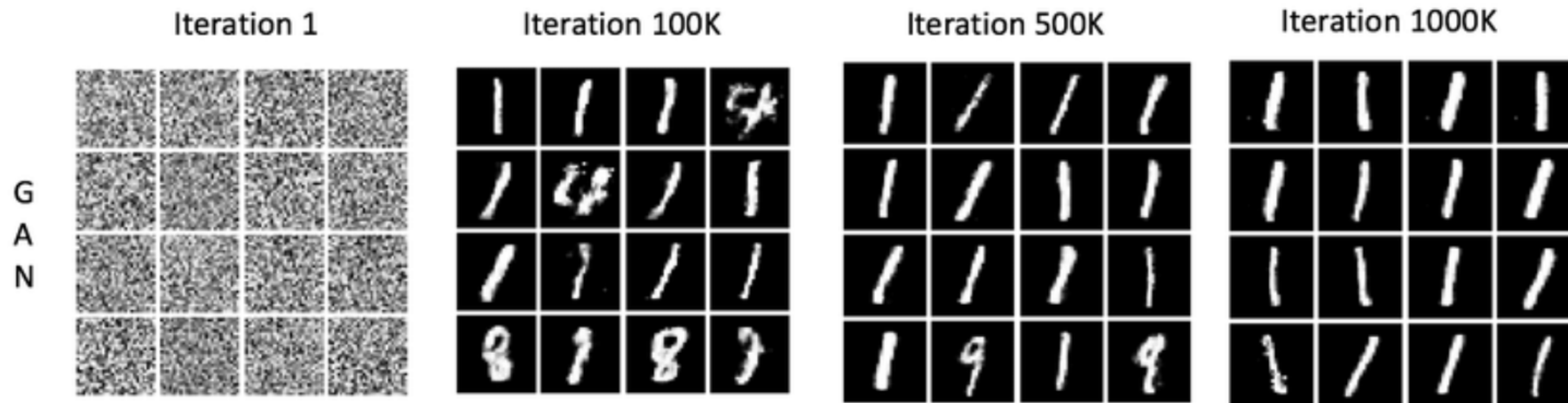
$$L_G = -\log(0.2)$$

$$L_G = -\log(D(G(z)))$$

$$\log(0.2) \approx -0.699, \quad L_G = 0.699$$

Zwei Hauptprobleme:

Mode Collapse



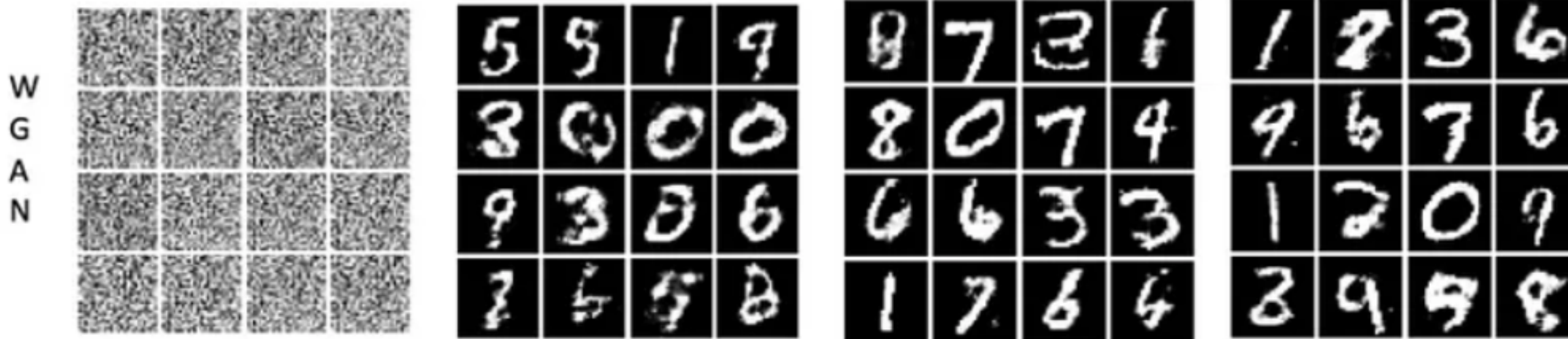
https://www.researchgate.net/figure/Generated-images-by-GAN-and-WGAN-models-trained-on-MNIST-after-1-100k-500k-1000k_fig2_329705388

Training Instabilität

WGANs

- verbesserte Version von GANs
- Lösen die Probleme
- "Kritiker" statt "Diskriminator"

$$\min_G \max_{D \in \mathcal{D}} \mathbb{E}_{x \sim \mathbb{P}_r} [D(x)] - \mathbb{E}_{\hat{x} \sim \mathbb{P}_g} [D(\hat{x})]$$



https://www.researchgate.net/figure/Generated-images-by-GAN-and-WGAN-models-trained-on-MNIST-after-1-100k-500k-1000k_fig2_329705388

WGANs

Generator Architektur:

Eingabe: Latenter Vektor (Größe: n_z)

↓
V
Layer 0: Linear -> BatchNorm -> ReLU (Ausgabe: $4 \times 4 \times 512$)
↓
V
Layer 1: TransConv -> BatchNorm -> ReLU (Ausgabe: $8 \times 8 \times n_{gf} \times 8$)
↓
V
Layer 2: TransConv -> BatchNorm -> ReLU (Ausgabe: $16 \times 16 \times n_{gf} \times 4$)
↓
V
Layer 3: TransConv -> BatchNorm -> ReLU (Ausgabe: $32 \times 32 \times n_{gf} \times 2$)
↓
V
Layer 4: TransConv -> BatchNorm -> ReLU (Ausgabe: $64 \times 64 \times n_{gf}$)
↓
V
Layer 5: TransConv -> BatchNorm -> ReLU (Ausgabe: $128 \times 128 \times n_{gf} / 2$)
↓
V
Final Conv: Conv -> Tanh (Ausgabe: $128 \times 128 \times n_c$)

Kritiker (Discriminator) Architektur:

Eingabe: Bild ($3 \times 128 \times 128$)

↓
V
Layer 0: Conv -> LeakyReLU (Ausgabe: $64 \times 64 \times n_{df}$)
↓
V
Layer 1: Conv -> BatchNorm -> LeakyReLU (Ausgabe: $32 \times 32 \times n_{df} \times 2$)
↓
V
Layer 2: Conv -> BatchNorm -> LeakyReLU (Ausgabe: $16 \times 16 \times n_{df} \times 4$)
↓
V
Layer 3: Conv -> BatchNorm -> LeakyReLU (Ausgabe: $8 \times 8 \times n_{df} \times 8$)
↓
V
Layer 4: Conv -> BatchNorm -> LeakyReLU (Ausgabe: $4 \times 4 \times n_{df} \times 16$)
↓
V
Final Layer: Linear -> Einzelner Ausgabewert

Training



Next Steps

- Erhöhen der Netzwerktiefe
- Progressive Growing of GANs
- Self-Attention GANs



Vielen Dank

Fragen?

