StyleGAN

A Preprint

1 Abstract

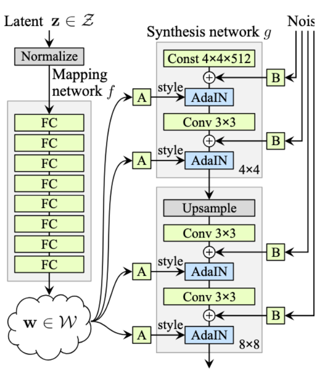
he new architecture leads to an automatically learned, unsupervised separation of high-level attributes (e.g., pose and identity when trained on human faces) and stochastic variation in the generated images (e.g., freckles, hair), and it enables intuitive, scale-specific control of the synthesis.

2 Generator

Basically in GAN architecture we have $z \sim N(0; \ldots)$ passed right to generator. However authors suggest different:

Mapping network: instead of going directly in generator z passes mapping network which is an 8-layer Multi-Layer Perceptron (MLP). Goal is to transform z from the input latent space Z into an intermediate latent space W. Then result is passed to synthesis network. It starts from a learned constant input (usually shape 4×4×512), then adds styles at each layer using AdaIN and also Injects noise at each layer for stochastic variation (e.g., texture, hair strands).Each layer of the generator doesn't use w directly, but instead passes it through a learned affine transformation: $style = A(w)$(its outputs Scale (for variance), Bias (for mean)).

$$ADAin(x; y) = y_{scale} \cdot \frac{x - \mu(x)}{\sigma(x)} + y_{bias}$$



(b) Style-based generator

3 Mixing regularization

The core idea is to improve the generator's ability to keep the effects of different "styles"(derived from the latent code w w) separate and localized to specific scales or parts of the image generation process.

During training we fix % of images for which we ll mix styles.

Algorithm 1 Style Mixing Pipeline (StyleGAN)

Input: Latent distribution $P(z)$, synthesis network g with N_{layers} blocks
Sample latent vectors: $z_1, z_2 \sim P(z)$
Compute intermediate latents: $w_1 = f(z_1)$, $w_2 = f(z_2)$
Randomly choose crossover point: $k \sim \mathcal{U}(1, N_{\text{layers}} - 1)$
Initialize input: $x_{\text{in}}^{(1)} = x_0$ (learned constant)
for $j = 1$ **to** N_{layers} **do**
    if $j \leq k$ **then**
        $w_{\text{current}} \leftarrow w_1$
    else
        $w_{\text{current}} \leftarrow w_2$
    end if
    Compute style: $(y_s^{(j)}, y_b^{(j)}) = A_j(w_{\text{current}})$
    Compute feature map: $h^{(j)} = \text{Conv}_j(\text{Upsample}_j(x_{\text{out}}^{(j-1)}))$
    Apply AdaIN per channel c:

$$h_c^{(j)\prime} = y_{s,c}^{(j)} \cdot \frac{h_c^{(j)} - \mu(h_c^{(j)})}{\sigma(h_c^{(j)})} + y_{b,c}^{(j)}$$

    Combine channels: $h^{(j)\prime} = \text{AdaIN}(h^{(j)}, y_s^{(j)}, y_b^{(j)})$
    Add noise: $x_{\text{out}}^{(j)} = h^{(j)\prime} + \text{ScaledNoise}^{(j)}$
end for
Final image: $I = \text{ToRGB}(x_{\text{out}}^{(N_{\text{layers}})})$

4 Perceptual Path Length (PPL)

PPL aims to measure the perceptual change in the image space for a small step in the latent space.

$$l_Z = \mathbb{E}_{z_1, z_2 \sim P(z), t \sim U(0;1)}[\frac{1}{\epsilon^2} d(G(slerp(z_1, z_2, t)), G(slerp(z_1, z_2, t + \epsilon)))]$$

For intermediate W z is replaced by f(z).

- **Spherical Linear Interpolation (SLERP):**
  Given two normalized latent vectors $z_1, z_2 \in \mathcal{Z}$ and interpolation factor $t \in [0, 1]$, the spherical interpolation is denoted as:
  $$\text{slerp}(z_1, z_2; t)$$
  This provides smooth interpolation on the hypersphere, suitable for unit-norm latent vectors z.
- **Generator:**
  The full generator is denoted by:
  $$G = g \circ f$$
  where f is the mapping network and g is the synthesis network (in style-based generators). For traditional GANs, G is a single monolithic generator network.
- **Perceptual Distance Metric:**
  Let $d(\cdot, \cdot)$ be a perceptual image distance metric. In the StyleGAN paper, this is implemented as a weighted ℓ_2 distance between deep feature embeddings extracted from a pretrained VGG16 network (as in Zhang et al. [?]).
- **Step Size ϵ:**
  A small constant step size, typically:
  $$\epsilon = 10^{-4}$$
- **Expectation $\mathbb{E}$:**
  Expectations are estimated via Monte Carlo sampling — that is, averaging over many samples.
- **Quadratic Metric Note:**
  Since $d(\cdot, \cdot)$ is based on a squared (quadratic) distance, terms involving it are divided by ϵ^2 when used in finite difference approximations.

5 Linear separatibiltiy

This metric quantifies how well a linear hyperplane can separate latent space points corresponding to a specific binary image attribute.

- Generate a large set of N images:

$$I_k = G(z_k) \quad \text{or} \quad I_k = g(f(z_k)), \quad \text{for } k = 1, \ldots, N$$

  Let x_k be the corresponding latent vectors (either z_k or $w_k = f(z_k)$).
- For each of the 40 binary attributes from CelebA (e.g., "Male", "Eyeglasses"), perform the following:
  1. Label the N generated images using a pre-trained auxiliary classifier C_{attr} specific to the attribute. This yields class labels:
  $$Y_k \in \{0, 1\}$$
  2. Sort the samples by the classifier's confidence score and select the top N_{label} most confident samples (e.g., $N_{\text{label}} = 100{,}000$).
  3. Train a linear Support Vector Machine (SVM) on the selected N_{label} latent vectors x_k to predict their corresponding labels Y_k. Let the SVM's predictions be denoted by X_k.
  4. Compute the conditional entropy $H(Y \mid X)$:
  $$H(Y \mid X) = -\sum_{i,j} P(Y = y_i, X = x_j) \log_2 P(Y = y_i \mid X = x_j)$$
  This measures how much additional information is needed to determine the true class Y given the SVM's prediction X.
- The final separability score is computed as:

$$\text{Separability} = \exp\left(\sum_{\text{attr}=1}^{40} H(Y_{\text{attr}} \mid X_{\text{attr}})\right)$$

  A lower score indicates better linear separability and hence a more disentangled representation of the attributes.