ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

Computer Vision Course Project Proposal

USING GAN TO GENERATE FACIAL IMAGES

1. REDIET FEREW TEKA UGR/1415/12
2. SAMUEL ABATNEH ENDALIE UGR/7229/12
3. SAMUEL GIRMA MEGRA UGR/6303/12
4. SEFINEH TESFA BEKELE UGR/2844/12

TO: DR. Fantahun Bogale June 18, 2024

Explore the use of GANs to generate realistic facial images

DCGAN, WGAN, StyleGAN, and CycleGAN for Facial Image Generation

Introduction

Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs), introduced by Ian Goodfellow and colleagues in 2014, have revolutionized the field of artificial intelligence, particularly in image generation. GANs consist of two neural networks: a generator and a discriminator, which are trained simultaneously through adversarial training. The generator creates images from random noise, while the discriminator attempts to distinguish between real and generated images. The goal of the generator is to produce images that are indistinguishable from real ones, while the discriminator aims to correctly classify images as real or fake. This adversarial process results in the generation of highly realistic images once the networks reach an equilibrium.

Deep Convolutional GAN (DCGAN)

DCGANs, introduced in 2015, represent a significant improvement over traditional GANs by incorporating deep convolutional neural networks in both the generator and discriminator components. This architectural enhancement allows for better capture of intricate facial features and spatial relationships, addressing the limitations of conventional GANs in terms of architectural constraints and training instability. DCGANs leverage convolutional layers to enhance feature extraction and spatial modeling, leading to superior performance in generating diverse and high-quality facial images. Their improved training stability and convergence have positioned DCGANs as a cornerstone in generating top-tier face images for applications such as facial recognition systems, virtual avatars, and artistic endeavors. It was chosen for this project for its robust architecture that enhances feature extraction and spatial modeling, making it suitable for generating high-quality, realistic facial images.

Wasserstein GAN (WGAN)

Wasserstein GAN (WGAN) was introduced to address some of the critical issues related to the training stability of GANs, particularly the problem of mode collapse. Traditional GANs use a binary

cross-entropy loss which can lead to vanishing gradients and unstable training. WGAN, on the other hand, employs the Wasserstein distance (or Earth Mover's distance) to measure the difference between the real and generated data distributions. This approach results in smoother gradients and more stable training dynamics. The use of the Wasserstein distance ensures that the generator receives more informative feedback, even when the discriminator is strong. This advancement significantly improves the robustness and quality of the generated images. It's selected for its stability in training and ability to produce high-quality images by addressing mode collapse and gradient issues inherent in traditional GANs. It is selected for this project for its stability in training and ability to produce high-quality images by addressing mode collapse and gradient issues inherent in traditional GANs.

StyleGAN

StyleGAN, introduced by NVIDIA researchers in 2018, represents a major leap in the generation of high-quality facial images. It introduces a novel architecture that separates the latent space (style) from the spatial resolution (structure), allowing for unprecedented control over the generated images. StyleGAN's generator network includes several intermediate latent spaces that enable nuanced control over the style at different levels of detail. This architecture allows for the generation of highly realistic images with detailed control over facial attributes such as age, gender, expression, and more. StyleGAN has set new standards in the field of image synthesis, making it a versatile tool for both research and practical applications. It was selected for this project for its ability to generate highly detailed and controllable facial images, providing fine-grained control over various facial attributes and ensuring high realism.

By leveraging these different GAN architectures, the project aims to explore a wide range of applications in AI-powered facial image creation, from generating new faces and manipulating facial attributes to creating artistic textures and styles. This comprehensive approach ensures that the project can address diverse challenges and applications within the field.

Dataset

In our project, we aimed to utilize a diversified dataset to generate realistic and varied facial images, including representations of Black individuals. Initially, we sought datasets that provided a wide range of ethnic diversity but encountered challenges in finding suitable sources. Ultimately, we opted for a dataset available on Kaggle, which includes images of celebrities. This dataset consists of 1,700 images of 17 celebrities, with each celebrity contributing 100 facial images. Notably, this dataset contains images of Black individuals, allowing us to generate diverse and inclusive facial images.

The dataset's characteristics are as follows:

- Total Images: 1,700
- Number of Celebrities: 17
- Images per Celebrity: 100
- Diversity: Includes images of Black individuals

The images in the dataset vary in pose, expression, and lighting, providing a comprehensive set of facial variations for training our GAN models. The diversity within this dataset helps ensure that the generated images reflect a wide range of facial features and characteristics.

Models

DCGAN

Dataset Preprocessing

The dataset was preprocessed to standardize the input images for training. The preprocessing steps included:

- Resizing: Images were resized to 64x64 pixels.
- Center Cropping: Ensured the facial features are maintained.
- Normalization: Pixel values were normalized to the range [-1, 1].

Model Architecture

DCGAN is a variant of GAN that uses deep convolutional layers in both the Generator and the Discriminator. This architecture improves the quality of generated images by leveraging the hierarchical feature learning capabilities of convolutional neural networks. The key distinctions of DCGANs include:
Convolutions without Pooling: Instead of pooling layers, DCGANs use strided convolutions (in the Discriminator) and fractional-strided convolutions (in the Generator) for downsampling and upsampling, respectively. This approach preserves spatial dimensions and details better.
Batch Normalization: Applied to stabilize the training process and normalize the activations in both the Generator and the Discriminator.
Leaky ReLU and ReLU: Uses Leaky ReLU activations in the Discriminator and ReLU activations in the Generator to improve gradient flow.

1. **Generator**

The Generator network is responsible for creating synthetic images from random noise vectors (latent vectors). It uses a series of transposed convolutional layers (also known as deconvolutional layers) to upsample the input noise vector to a full-sized image. The network architecture includes:

- Latent Vector Input: The input to the Generator is a latent vector of a predefined size (typically 100 dimensions), drawn from a standard normal distribution.
- Transposed Convolutions: These layers progressively increase the spatial dimensions of the input, starting from a small feature map and ending with an output image of the desired size (64x64 pixels in this case).
- Batch Normalization: Applied after each transposed convolutional layer to stabilize and accelerate training.
- Activation Functions: ReLU activations are used after each transposed convolutional layer, except the final layer which uses a Tanh activation function. The Tanh function ensures the output pixel values are in the range [-1, 1], matching the normalized input image range.

2. Discriminator

The Discriminator network is responsible for distinguishing between real and fake images. It uses a series of convolutional layers to downsample the input image to a single scalar output that indicates the probability of the input being a real image. The network architecture includes:

- Image Input: The input to the Discriminator is a 64x64 pixel image (either real or generated).
- Strided Convolutions: These layers progressively decrease the spatial dimensions of the input, capturing hierarchical features of the image.
- Batch Normalization: Applied after each convolutional layer (except the first) to stabilize and accelerate training.
- Activation Functions: Leaky ReLU activations are used after each convolutional layer to allow gradients to propagate through the network even for non-active units. The final layer uses a Sigmoid activation function to output a probability value between 0 and 1.

3. Weight Initialization

To improve training stability and convergence, weights of the convolutional and transposed convolutional layers were initialized to a normal distribution with mean 0 and standard deviation 0.02. This initialization method is crucial for DCGANs as it ensures that the gradients remain stable during the initial stages of training.

4. Loss Function and Optimizers

The loss function used for both the Generator and Discriminator is Binary Cross-Entropy Loss (BCELoss). This loss function measures the performance of the classification tasks of the Discriminator (real vs. fake) and the Generator (fooling the Discriminator). Both networks were optimized using the Adam optimizer with the following parameters:

- Learning Rate (lr): 0.0002
- Beta1: 0.5
- Beta2: 0.999

The choice of Beta1 and Beta2 helps in maintaining the stability of the training process and preventing the gradients from vanishing or exploding.

5. **Training Procedure**

The training involved alternating updates to the Discriminator and Generator. For each batch of real images:

Update Discriminator

- Real Images: The Discriminator processes real images from the dataset and updates its weights to maximize the probability of correctly classifying them as real.
- Fake Images: The Discriminator processes fake images generated by the Generator and updates its weights to maximize the probability of correctly classifying them as fake. The Generator's gradients are not updated in this step.

Update Generator:

- The Generator generates a batch of fake images from random noise vectors.
- These fake images are then passed through the Discriminator.
- The Generator updates its weights to maximize the probability of the Discriminator classifying the fake images as real. This step updates the Generator's gradients to produce more realistic images.

Result

The DCGAN model was trained for 500 epochs on the celebrity face image dataset. Over the course of training, significant progression in the quality of generated images was observed. Early in the training, the generated images were mostly indistinguishable noise, but as the training progressed, the Generator began to produce images with outlined facial structures and features. Although more epoch is required to generate more realistic images.

WGAN (Wasserstein Generative Adversarial Network)

Dataset Preprocessing

The dataset was preprocessed to standardize the input images for training. The preprocessing steps included:

- **Resizing**: Images were resized to 64x64 pixels.
- **Center Cropping**: Ensured the facial features are maintained.
- **Normalization**: Pixel values were normalized to the range [-1, 1].

Model Architecture

WGAN is a variant of GAN designed to address issues with training stability and mode collapse by using the Wasserstein distance (Earth-Mover's distance) as the loss metric. This approach improves the quality of generated images by providing a more meaningful loss landscape.

Key Distinctions of WGANs include:

- **Weight Clipping**: Instead of using Batch Normalization, WGANs use weight clipping to ensure the weights of the Discriminator lie within a compact space.
- **Gradient Penalty (in WGAN-GP)**: An alternative to weight clipping, which penalizes the norm of the gradient to enforce the Lipschitz constraint.
- **Different Loss Functions**: The Discriminator and Generator loss functions are based on Wasserstein distance rather than binary cross-entropy.

Generator

The Generator network is responsible for creating synthetic images from random noise vectors (latent vectors). It uses a series of transposed convolutional layers (also known as deconvolutional layers) to upsample the input noise vector to a full-sized image. The network architecture includes:

- **Latent Vector Input**: The input to the Generator is a latent vector of a predefined size (typically 100 dimensions), drawn from a standard normal distribution.
- **Transposed Convolutions**: These layers progressively increase the spatial dimensions of the input, starting from a small feature map and ending with an output image of the desired size (64x64 pixels in this case).
- **Batch Normalization**: Applied after each transposed convolutional layer to stabilize and accelerate training.
- **Activation Functions**: ReLU activations are used after each transposed convolutional layer, except the final layer which uses a Tanh activation function. The Tanh function ensures the output pixel values are in the range [-1, 1], matching the normalized input image range.

Discriminator

The Discriminator network is responsible for distinguishing between real and fake images. It uses a series of convolutional layers to downsample the input image to a single scalar output that indicates the critic score, which reflects the Wasserstein distance. The network architecture includes:

- **Image Input**: The input to the Discriminator is a 64x64 pixel image (either real or generated).
- **Strided Convolutions**: These layers progressively decrease the spatial dimensions of the input, capturing hierarchical features of the image.
- **Weight Clipping/Gradient Penalty**: Instead of Batch Normalization, weight clipping or gradient penalty is applied to enforce the Lipschitz constraint.

- **Activation Functions**: Leaky ReLU activations are used after each convolutional layer to allow gradients to propagate through the network even for non-active units. The final layer provides the critical score without a sigmoid activation, as WGAN does not require probabilities.

Weight Initialization

To improve training stability and convergence, weights of the convolutional and transposed convolutional layers were initialized to a normal distribution with mean 0 and standard deviation 0.02. This initialization method is crucial for WGANs as it ensures that the gradients remain stable during the initial stages of training.

Loss Function and Optimizers

The loss function used for both the Generator and Discriminator is based on the Wasserstein distance. Both networks were optimized using the RMSprop optimizer with the following parameters:

- **Learning Rate (lr)**: 0.00005

Training Procedure

The training involved alternating updates to the Discriminator and Generator. For each batch of real images:

Update Discriminator:

- **Real Images**: The Discriminator processes real images from the dataset and updates its weights to maximize the critic score for real images.
- **Fake Images**: The Discriminator processes fake images generated by the Generator and updates its weights to minimize the critic score for fake images. The Generator's gradients are not updated in this step.

Update Generator:

- The Generator generates a batch of fake images from random noise vectors.
- These fake images are then passed through the Discriminator.
- The Generator updates its weights to maximize the critic score for the fake images, effectively producing more realistic images.

Result

The WGAN's approach with the Wasserstein distance and weight clipping/gradient penalty results in a more stable training process and higher quality generated images compared to the traditional DCGAN. Unlike DCGAN, which may suffer from mode collapse and training instability, WGAN provides a smoother loss landscape, leading to more consistent improvements in image realism.

STYLE GAN

StyleGAN is a generative adversarial network (GAN) architecture developed by NVIDIA, which introduces a novel way of controlling the synthesis of high-quality images. It leverages a "style" vector to provide fine-grained control over different levels of detail in the generated images. This allows for the manipulation of specific features, such as facial attributes, hair styles, and backgrounds.

Key Features of StyleGAN:

1. **Mapping Network**:
    - **Purpose**: Transforms a latent vector zzz into an intermediate latent space www. This transformation helps in disentangling the latent space and makes it easier to control various aspects of the generated image.
    - **Structure**: The mapping network consists of multiple fully connected layers. Typically, it has 8 layers.
    - **Benefit**: By using a separate mapping network, StyleGAN can achieve better control over the generated images and create more detailed and varied outputs.
2. **Adaptive Instance Normalization (AdaIN)**:
    - **Purpose**: Modulates the feature maps based on the style vector www. This modulation controls the influence of styles at different layers in the generator network.
    - **Structure**: AdaIN adjusts the mean and variance of the activations in the generator network according to the style vector www.
    - **Benefit**: This allows for fine-grained control over different aspects of the image, such as textures and colors, by injecting style information at various levels of detail.
3. **Generator Architecture**:
    - **Purpose**: Generates images from the transformed latent vector www.
    - **Structure**: The generator starts from a learned constant input and progressively grows to higher resolutions using transposed convolutional layers. Each layer is modulated by the style vector www using AdaIN.
    - **Benefit**: This progressive growing approach allows the model to generate high-resolution images with detailed features.
4. **Discriminator Architecture**:
    - **Purpose**: Distinguishes between real and fake images.
    - **Structure**: Similar to traditional GAN discriminators but adapted to handle the progressive growing nature of the generator. It uses convolutional layers to downsample the image and outputs a scalar value indicating the likelihood of the image being real.
    - **Benefit**: Provides feedback to the generator to improve the quality of the generated images.

Dataset Preprocessing

1. **Resizing**: Images are resized to a fixed resolution (e.g., 64x64 pixels for WGAN and higher resolutions for StyleGAN).
2. **Center Cropping**: Ensures that key features (e.g., faces) are centered and consistent.

3. **Normalization**: Pixel values are normalized to the range [-1, 1] to match the activation function ranges used in the networks.

Weight Initialization:

To improve training stability and convergence, the weights of convolutional and transposed convolutional layers are initialized to a normal distribution with mean 0 and standard deviation 0.02. This initialization method is crucial as it ensures that the gradients remain stable during the initial stages of training.

Training Procedure:

1. **Discriminator Update**:
    - Processes real images from the dataset and updates its weights to maximize the critic score for real images.
    - Processes fake images generated by the generator and updates its weights to minimize the critic score for fake images. The generator's gradients are not updated in this step.
2. **Generator Update**:
    - Generates a batch of fake images from random noise vectors.
    - Passes these fake images through the discriminator.
    - Updates the generator's weights to maximize the critic score for the fake images, effectively producing more realistic images.

Result

StyleGAN's approach, with the use of the mapping network, AdaIN, and progressive growing, results in a more stable training process and higher quality generated images compared to traditional GANs and WGANs. Unlike traditional GANs, which may suffer from mode collapse and training instability, StyleGAN provides a smoother loss landscape, leading to more consistent improvements in image realism.

References:

- **StyleGAN Paper**: Karras, Tero, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019.
- **WGAN Paper**: Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein GAN." arXiv preprint arXiv:1701.07875 (2017).
- **Blog Resource**: Implementation and detailed explanation of StyleGAN can be found in the blog post.