StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks

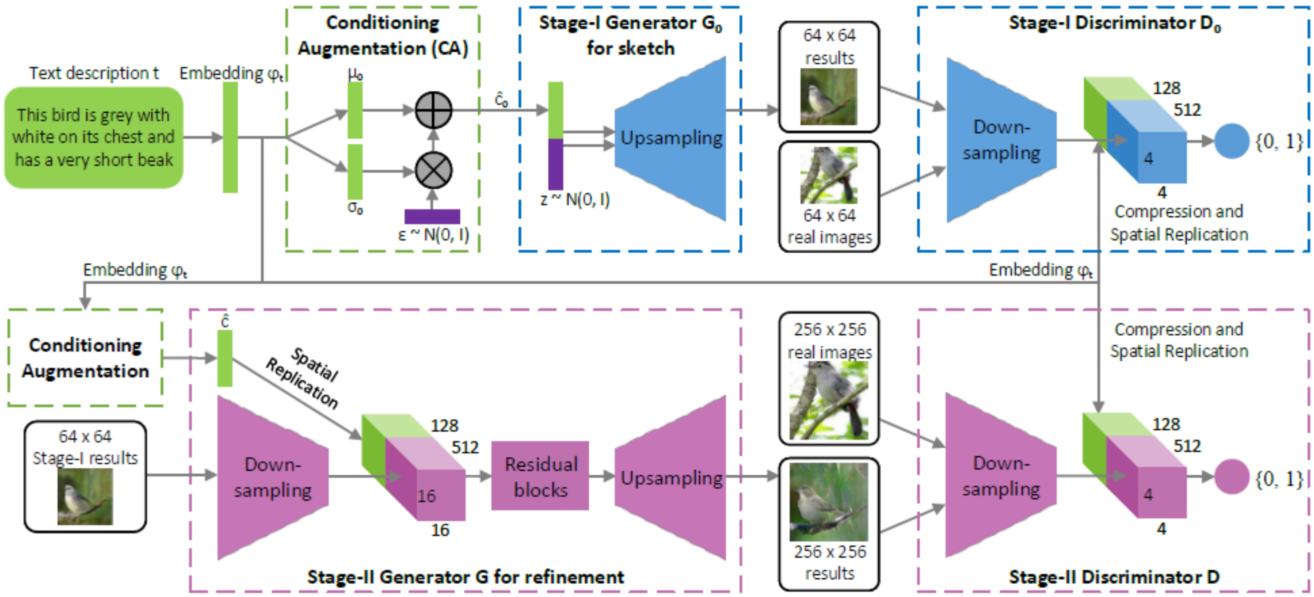

Figure 2. The architecture of the proposed StackGAN. The Stage-I generator draws a low-resolution image by sketching rough shape and basic colors of the object from the given text and painting the background from a random noise vector. Conditioned on Stage-I results, the Stage-II generator corrects defects and adds compelling details into Stage-I results, yielding a more realistic high-resolution image.

Text embedding:

1. Captures the **semantic meaning** of the text using pretrained NN
2. The output is a fixed-dimensional vector (e.g., 128 dimensions) representing the input text.
3. **Deterministic:** Each text description always maps to the same embedding.

Conditioning Augmentation

- Without CA: The model might generate one fixed image for this description, lacking diversity.
- With CA: The model generates varied instances of the bird (different poses, slight color variations, etc.) while remaining semantically consistent with the text.
- CA introduces **variability (for a same sentence we may able to generate diverse outputs)** to the text embedding by sampling from a Gaussian distribution around 't'.

Example

Conditioning Augmentation (train with KL divergence)

CA introduces variability to the text embedding by sampling from a Gaussian distribution around t′.

Step-by-Step Process

1. Text Embedding:
    - The fixed text embedding **t′=[0.9,0.3,−0.2]** is computed using a text encoder.
2. Parameterize a Gaussian Distribution:

- A neural network learns to generate the mean and standard deviation for a Gaussian distribution based on t′:

  μ = [0.9,0.3,−0.2], σ=[0.1,0.05,0.08]

3. **Sample the Conditioning Vector** (c):
    - During training, a random sample c is drawn from this distribution:

$$c = \mu + \sigma \odot \epsilon$$

Here, ϵ is random noise sampled from $N(0, I)$. For instance:

- If $\epsilon = [0.5, -0.3, 0.2]$,
- Then $c = [0.9, 0.3, -0.2] + [0.1, 0.05, 0.08] \odot [0.5, -0.3, 0.2]$,
- Resulting in $c = [0.95, 0.285, -0.184]$.

Use c in the Generator:

- The generator takes c as input along with noise z to create a variety of plausible images:
    - One c might produce an apple with a glossy surface.
    - Another c might generate an apple with a leaf or a slightly different background.

The **Stage-I Generator (G0)** in the StackGAN architecture:

It serves as the first step in generating an image from text descriptions. Its primary goal is to produce a **low-resolution "sketch" of the image**, capturing the basic shape, layout, and colors specified in the input text. Here's a breakdown:

Stage-I Generator

1. **Low-Resolution Output**:
   - G0 generates a coarse image, typically 64×64 pixels, as an initial approximation of the target image.
   - It focuses on capturing global structures and colors rather than fine details.
2. **Text-to-Image Mapping**:
   - The generator takes as input:
     - A **random noise vector** z for diversity (diversity in image generation).
     - A **conditioning vector** c (diversity in semantic meaning), derived from the text embedding t′ via **Conditioning Augmentation**.
   - The output is an image consistent with the semantic meaning of the text.
3. **Architecture**:

- **Concatenation**: c and z are concatenated to form the input to the generator.
- **Upsampling Layers**: The generator employs upsampling layers (e.g., transposed convolutions or nearest-neighbor upsampling followed by convolution) to transform the low-dimensional input into a 64×64 image.

4. **Training Objective**:
   - **Adversarial Loss**: Ensures the generated image looks realistic.
   - **KL Divergence Loss**: Regularizes the Conditioning Augmentation.

Architecture of D0

1. **Image Downsampling**: (extract features from image)
   - D0 processes the input image through a series of **downsampling layers** (e.g., convolutional layers with strides greater than 1).
   - These layers extract features from the image while reducing its spatial dimensions.
2. **Text Embedding (t′)**:

- The text embedding is processed through a fully connected layer to match the dimensionality of the image features.
- This transformed 't' is spatially replicated to create a tensor of the same shape as the image features (e.g., 4×4×128).

3. **Fusion with Image Features**:
   - The replicated 't' tensor is concatenated with the image feature map (e.g., 4×4×512 to produce a joint representation.
   - The combined tensor (e.g., 4×4×640) is further processed by convolutional layers to evaluate realism and semantic alignment.

Purpose of Stage-II Generator

1. **Enhancing Image Quality**:
   - Takes the 64×64 low-resolution image from Stage-I and transforms it into a high-resolution image (e.g., 256×256).
   - Adds fine details such as textures, edges, and complex patterns.
2. **Improving Semantic Alignment**:
   - Uses the input text description again to ensure the high-resolution image remains consistent with the semantics of the text.

- The combination of **downsampling** and **upsampling** in the Stage-II Generator ensures efficient feature extraction, refinement, and high-resolution output generation.
- Downsampling focuses on learning abstract features, while upsampling reconstructs them into a detailed, realistic image.
- As the Stage-II Generator refines high-resolution images, the network can become very deep. Residual blocks mitigate issues like vanishing gradients by preserving information through skip connections.

Results:

In **Figure 1** of the StackGAN paper, the "Vanilla GAN" approach to generating images from text uses a **single-stage GAN architecture without conditional augmentation**.