Implementing StackGANs using a Pokémon Dataset

IllusionCraft

June 13, 2024

Objective

Your task is to implement a StackGAN (Stacked Generative Adversarial Network) in Python. You will use the provided Pokémon dataset for training your model. The goal is to demonstrate a comprehensive understanding of the StackGAN architecture and its implementation.

Dataset

You will use the Pokémon dataset provided for this task. You can download the dataset from the following link:

Pokémon Dataset

Ensure you extract and preprocess the dataset appropriately for training your StackGAN model.

Key Requirements

1. **StackGAN Architecture Implementation:**
   - Implement the StackGAN architecture using a deep learning framework of your choice (e.g., TensorFlow, PyTorch).
   - The model should include:
     (a) A **Stage-I generator** network designed to produce low-resolution images.
     (b) A **Stage-I discriminator** network to evaluate the authenticity of low-resolution images.
     (c) A **Stage-II generator** network to refine the low-resolution images to high-resolution.
     (d) A **Stage-II discriminator** network to evaluate the authenticity of high-resolution images.
   - Ensure that the training process effectively uses the adversarial interplay between the generator and discriminator networks at both stages to enhance the quality of the generated images.
2. **Dataset Preprocessing:**
   - Download and extract the Pokémon dataset.
   - Pre-process the dataset to suit the StackGAN model requirements, such as normalization, resizing, and any other necessary transformations.
3. **Training Process:**
   - Train your StackGAN model on the Pokémon dataset.
   - Document the training process, including the hyperparameters used, training duration, and any challenges faced.
4. **Model Evaluation:**
   - Evaluate the quality of the generated images using the **Inception Score**.
   - Include sample outputs generated by the trained StackGAN to showcase its performance.

5. **Documentation and Code Submission:**
    - Submit all source code files, including scripts for training and testing the StackGAN.
    - Provide a README file with detailed instructions on how to run the model, including dependencies and installation steps.

Brownie Points

a. **API Integration:**
    - Develop an API to interface with your trained StackGAN model.
    - The API should allow users to generate images via a simple query, such as an HTTP request.
    - Consider using frameworks like Flask or FastAPI to build the API.

b. **User Interface:**
    - Implement a simple user interface or script to demonstrate the API's functionality.
    - Users should be able to generate new images at the click of a button or through a straightforward command.

c. **Deployment:**
    - Deploy the API to a cloud service (e.g., AWS, GCP) or provide a Docker container for easy deployment.

d. **Additional Features:**
    - Implement parameter customization in the API for generating images (e.g., different latent vectors).
    - Provide comprehensive documentation and well-organized code structure.

Submission Requirements

1. **Code and Documentation:**
    - Submit all source code files, including scripts for training and testing the StackGAN.
    - Provide a README file with detailed instructions on how to run the model, including dependencies and installation steps.
2. **Sample Outputs:**
    - Include a set of sample images generated by your StackGAN in your submission.
    - Provide any additional visualizations or metrics that illustrate the model's performance.
3. **API Demonstration (for Brownie Points):**
    - If you choose to implement an API, provide a live link to your deployed API, if possible.
    - Alternatively, provide clear instructions on how to locally set up and interact with the API.

Evaluation Criteria

- Correctness and robustness of the StackGAN implementation.
- Quality and realism of the generated images.
- Completeness and clarity of the documentation and code structure.
- Creativity in dataset preprocessing and additional features.
- Extra functionality and usability provided by the API, if implemented.
- Quality of the Inception Score evaluation and analysis.

Additional Resources

- StackGAN Paper - Original paper on Stacked Generative Adversarial Networks.
- Inception Score Implementation - A guide on calculating the Inception Score.
- Flask Documentation - For building the API.
- FastAPI Documentation - An alternative framework for building high-performance APIs.
- Streamlit Documentation - A framework for building rudimentary user-interfaces