GAN Assignment

PW SKILLS

Question 1: Use any GAN of your choice (preferably DCGAN) to generate images from noise. Perform the following experiments.

A. Use the CIFAR 10 database to learn the GAN network. Generate images once the learning is complete.

B. Plot generator and discriminator losses and show how can you ascertain the convergence of the GAN training process.

Question 2: Fine-tuning Take a ResNet50 model and the database to be used for this question is CIFAR-10. Remove its classification layer and place a 2-layer neural network followed by a Softmax layer. Calculate classification accuracy on a train set, test set, and plot accuracies over epochs when:

A. The complete network is trained from scratch (i.e, random weights)

B. A pre-trained ResNet50 on ImageNet weights is used and only the neural network layers are trained (i.e, weights of layers of ResNet50 are kept frozen and unchanged)

C. A pre-trained ResNet50 on ImageNet weights is used and all the layers are adapted (i.e, weights of layers of ResNet50 are also updated now)

D. Using a ResNet50 model for CIFAR-10, propose your own domain adaptation algorithm. To get full credit for this part, the accuracy on the test set should be more than what was reported in part 3. You may build upon part(3) to propose your own algorithm. Explain why your proposed algorithm is working better. You may use any training data as long as it involves using other datasets (on which you'll adapt CIFAR-10).

Question 3: Implement a gan from scratch using Keras to generate celebrity faces from noise using this data-: https://www.kaggle.com/datasets/jessicali9530/celeba-dataset

Use cases found for GAN:

- Super-resolution: increasing the resolution of input images
- Colorise blank and white images
- image inpainting - fill missing blocks in images
- Anime face generation
- font generation
- style transfer
- human face generation
- image to emoji
- GAN for data augmentation
- Face ageing GAN
- front facial view generation from images provided of different sides
- Photo blending- blending 2 images

Coding Questions

1. Data Augmentation Function for GAN Training

Write a Python function that generates augmented data for training a GAN. The function should take an image dataset as input and apply data augmentation techniques commonly used in GAN training, such as random rotation, flipping, and cropping. The function should return the augmented dataset.

You can use popular image-processing libraries like OpenCV or PIL to perform these augmentations. Ensure that the function allows customization of augmentation parameters, such as rotation angles, flip probability, and crop size

2.Create a simple discriminator model using tensorflow keras which can classify a image as real or fake. You can use random noise same size as the image to train the model.

3.Create a generator model with uses transpose convolution to generate 32 x 32 x 3 images from random noise. In this question you can just define the model architecture for the generator and make sure that the model is generating the desired image size , you can take a latent space dimension as a array of 100 float values.

4. Implementing a Minimax Loss Function for GANs

Write a Python function that calculates the Minimax loss for a GAN. The function should take as input the predictions (scores) from a discriminator and return the Minimax loss.

The Minimax loss function for GANs is typically defined as follows:

$$\text{Loss} = -\mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(D(x))] - \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

5.Use the models made in question 2,3 and make your own GAN model by connecting the generator and the discriminator to generate images from random noise , You can use CIFAR -10 dataset. Find some tips on creating your own GAN here :https://machinelearningmastery.com/how-to-code-generative-adversarial-network-hacks/

6. Transfer Learning with GANs on the CIFAR-10 Dataset

Transfer learning is commonly used to improve GAN performance. In this question, you should implement a GAN for image generation using transfer learning. The data source is the CIFAR-10 dataset, which is a dataset of 60,000 32x32 color images in 10 different classes.

Here are the steps you should follow:

- **Load a pre-trained convolutional neural network (CNN) model (e.g., VGG16 or ResNet) using a library like PyTorch or TensorFlow.**
- **Modify the model for GAN-based image generation by removing the fully connected layers.**
- **Implement a generator and discriminator network.**
- **Train the GAN using the pre-trained CNN as the feature extractor to improve the quality of generated images.**
- **Evaluate the performance of your GAN by generating sample images from random noise.**

7. **Can you create a Python function that implements a basic Generative Adversarial Network (GAN) for generating grayscale images resembling handwritten digits (0 to 9) from the MNIST dataset? Your function should include the generator and discriminator networks, as well as the training loop with appropriate loss functions and optimizers. Additionally, demonstrates the generation of a few sample images using the trained GAN model.**

8. **Create a Deep Convolutional Generative Adversarial Network (DCGAN) in TensorFlow/Keras to generate high-resolution images from low res images data set: https://www.kaggle.com/datasets/adityachandrasekhar/image-super-resolution . Describe the architectural choices you make and how they contribute to the model's performance.**

9. **Create a conditional GAN which can generate images based on input condition. The data set you would use in this case is fashion_mnist dataset from tensorflow data sets.**
Reference:https://machinelearningmastery.com/how-to-develop-a-conditional-generative-adversarial-network-from-scratch/

Submission Guidelines:

- Answer all the questions in a single Jupyter Notebook file (.ipynb).
- Include necessary code, comments, and explanations to support your answers and implementation.
- Ensure the notebook runs without errors and is well-organized.
- Create a GitHub repository to host your assignment files.
- Rename the Jupyter Notebook file using the format "date_month_topic.ipynb" (e.g., "21st_September_GAN.ipynb").
- Place the Jupyter Notebook file in the repository.
- Commit and push any additional files or resources required to run your code (if applicable) to the repository.
- Ensure the repository is publicly accessible.
- Submit the link to your GitHub repository as the assignment submission.

Grading Criteria:

- Understanding and completeness of answers: 40%
- Clarity and depth of explanations: 25%
- Correct implementation and evaluation of optimizer techniques: 15%
- Analysis and comparison of different optimizers: 10%
- Proper code implementation and organization: 10%

Note: Create your assignment in Jupyter notebook and upload it to GitHub & share that uploaded assignment file link through your dashboard. Make sure the repository is public.