Deep learning 2, big homework 2

Alexey Slizkov
18.12.2023

1 Diffusion

Run `python3 diffusion.py`, it will do something, but the quality is very poor, maybe due to insufficient computational resources.

The model is trained to remove noise from corrupted images by progressively refining predictions over a series of steps. The training process follows a diffusion method where the noise levels gradually increase and the model learns to denoise the images.

1.1 Dataset

The dataset used for training and testing the model consists of image files located in the specified directory. The dataset class (`Dataset`) loads the images and preprocesses them by converting them to tensors and normalizing their values. It randomly divides the dataset into training and testing sets based on a hash of the filename. This ensures that the training and testing sets have a similar distribution. The number of images in each set is printed for verification.

1.2 Model Architecture

The model architecture is defined by the `DenoisingGeniusModel` class. It consists of several downsampling and upsampling layers for feature extraction and reconstruction, respectively. The model uses convolutional and transposed convolutional layers along with batch normalization and ReLU activation functions.

The downsampling layers (denoted as `down1`, `down2`, `down3`, and `down4`) reduce the spatial dimension of the input while increasing the number of channels. The upsampling layers (denoted as `up4`, `up3`, `up2`, and `up1`) reverse this process by increasing the spatial dimension and reducing the number of channels. The `forward` method of the model performs the forward pass and returns the denoised image.

1.3 Training

The training process is performed over multiple epochs. In each epoch, the model is trained using the training dataset and the loss is computed. The Adam optimizer is used with a learning rate of 0.001. The learning rate is reduced exponentially using a scheduler. The mean squared error (MSE) loss function is used to compare the denoised image output with the noisy input.

During training, the noise levels are controlled using a variance schedule. The schedule consists of a series of beta values that gradually increase. These beta values are used to calculate alpha values, which determine the proportion

of noise and signal in the input. The alpha values are used to scale the noisy input and create a combination of noise and signal for training the model.

After each epoch, the model is evaluated using the testing dataset to compute the test loss. The test loss gives an indication of the model's performance on unseen data.

1.4 Visualization

During training, sample images are generated to visualize the progress of the denoising process. A set of random samples is generated and processed through the model for a specified number of steps. The resulting denoised images are saved as a grid of plots.

1.5 Usage

To run the diffusion model, execute the command `python3 diffusion.py`. However, note that the quality of the results may be poor due to insufficient computational resources.

2 FID and SSIM

I have spent many hours trying to compute piq.FID for just two tensors (1024, 3, 64, 64), what can be a more standard use case, but I was unable

SSIM is easier, can be computed by running eval.py: it is 0.0707 for my main VAE (anime).

The code description goes here:

First, the VAE model is loaded from the saved checkpoint. The model's parameters are printed to check the number of parameters.

Random latent vectors (latents) are gen-

erated for generating new images using the VAE model. These latent vectors are passed through the inference method of the model to generate the generated images.

A DatasetForFID class is defined to create a dataset from the generated images tensor. The DatasetForFID class stores the tensor of generated images as its attribute.

A DataLoaderForFID class is defined to create a DataLoader from the dataset. This class wraps around the original dataloader and preprocesses the images by scaling them to the range of -1 to 1.

Two instances of DataLoaderForFID are created: the first one with the generated images and the second one with the real images from the test dataset.

The FID metric and SSIM function from the piq library are imported. The FID metric is initialized, and the compute_feats method is used to compute the feature representations for the generated and real images.

The FID between the generated and real images is computed by passing the feature representations to the FID metric. The FID value is printed.

The SSIM between the generated and real images is computed using the ssim function from the piq library. The SSIM value is printed.

Note: The code for FID computation and SSIM computation has been separated into two sections for clarity.

3 Video

Is anime.mp4 here