Arbitrary-Scale Image Generation and Upsampling using Latent Diffusion Model and Implicit Neural Decoder

October 22, 2024

.

1 Implementation Details

The pipeline given in the paper integrates three major components:

- Pre-trained Autoencoder: Encodes input images into latent space without up-sampling layers.
- Latent Diffusion Model (LDM): Operates in a latent space and learns via denoising and alignment losses. The diffusion process in the latent space enables efficient high-quality image generation.
- Local Implicit Image Function (LIIF): Decodes images from the latent space and generates arbitrary-scale outputs by leveraging MLP-based architecture that decodes latent vectors to continuous pixel coordinates.

The paper is implemented using the publicly available code. Some suitable and required changes were made in the code for it's successful execution on the local machine. To train the implicit neural decoder and diffusion model, I used Adam optimizer with learning rates of 5e-5 and 1e 6, respectively. The network is implemented in two parts the first stage model and the latent diffusion model. First the first stage model is trained using the given dataset and the obtained checkpoints are stored which are then used for training the latent diffusion model.

2 Dataset Description

For the training purpose a dataset containing diverse set of high quality images with high resolution is used. The dataset comprises of 1000 images which are further splitted into training and validation and testing sets. But due to limited computing power available I used a subset of these images.

3 Results

The results were obtained only for the DIV2K dataset. Due to limited resources I was not able to run the code for multiple datasets. So only DIV2K dataset was used for training and evaluation. But due to not availability of the checkpoints of one of the model used for building the architecture I wasn't able to reproduce the results. I have successfully trained the first stage and obtained it's checkpoints for training the LDM but the later one require checkpoints of a SRDiffusion model which were not provided.

```
Validation DataLoader 0:   0% 0/50 [00:00<?, ?it/s]
Epoch 4:  99% 3020/3050 [24:57<00:14,  2.02it/s, loss=0.0801, v_num=0, rec_loss_step=0.0727, rec_loss_epoch=0.088]
Epoch 4: 100% 3040/3050 [25:02<00:04,  2.02it/s, loss=0.0801, v_num=0, rec_loss_step=0.0727, rec_loss_epoch=0.088]
Epoch 4: 100% 3050/3050 [25:05<00:00,  2.03it/s, loss=0.0801, v_num=0, rec_loss_step=0.0727, rec_loss_epoch=0.088]
Epoch 4: 100% 3050/3050 [25:05<00:00,  2.03it/s, loss=0.0801, v_num=0, rec_loss_step=0.0727, rec_loss_epoch=0.0853]
```

Figure 1: Training process