Report

312552052鄭博元

introduction

- DDPM通過逐步去噪的方式，從噪聲中生成清晰且高品質的圖像。相比傳統的生成方法，DDPM在生成質量和訓練穩定性上具有顯著的優勢。基於這些特點，我們這次要實現一個條件DDPM模型，該模型能夠根據多標籤條件生成對應的合成圖像，並通過預訓練的分類器評估生成圖像的準確性。

- 在這次作業中，我們將設計並訓練一個條件DDPM模型，選擇合適的噪聲調度、時間嵌入以及採樣方法。同時，我們將利用預訓練的評估器在訓練和採樣過程中進行指導，以提高模型的生成效果。不僅探討了DDPM在圖像生成中的優勢，還為未來的研究提供了可能的方向。

Implementation details

- 在class diffusion中先定義噪聲數,beta起始值結束值跟圖像大小以及設備,而prepare_noise_schedule則是線性調度方式,alpha是1-beta,而hat則是累積alpha值,
- noise_images部分把輸入圖像x加入noise並從alpha_hat獲取平方根,在計算1減去其值生成noise,而sample_timesteps隨機取時間步長做去noise,其中sample部分
- sample部分用torch.randn做生成標準正態分布之噪聲圖像,而for圈進行i步中的反覆取樣,通過模型predict條件label下的噪聲值,接著若>0做調整,然後做torch.lerp控制圖像對標籤的依賴程度,最後對x用反向擴散去除predict_noise然後再加入少量beta noise

Implementation details

- 而後進行訓練函數以及定義基本變數跟優化器等等,之後做epoch對每個數據前向傳播,loss計算,參數更新,在每個epochs結束時,用ema模型生成測試圖像並保存指定目錄,接著進行launch定義輸入的參數訓練
- 而evaluation_model部分用於評估生成圖像準確性,基於ResNet18並在最後一層使用24類全連階層,compute_acc對每張圖片比較預測的label與真實label算出準確率而eval就是用resnet18 model評估生成圖像,最後進行主程式的運行環節

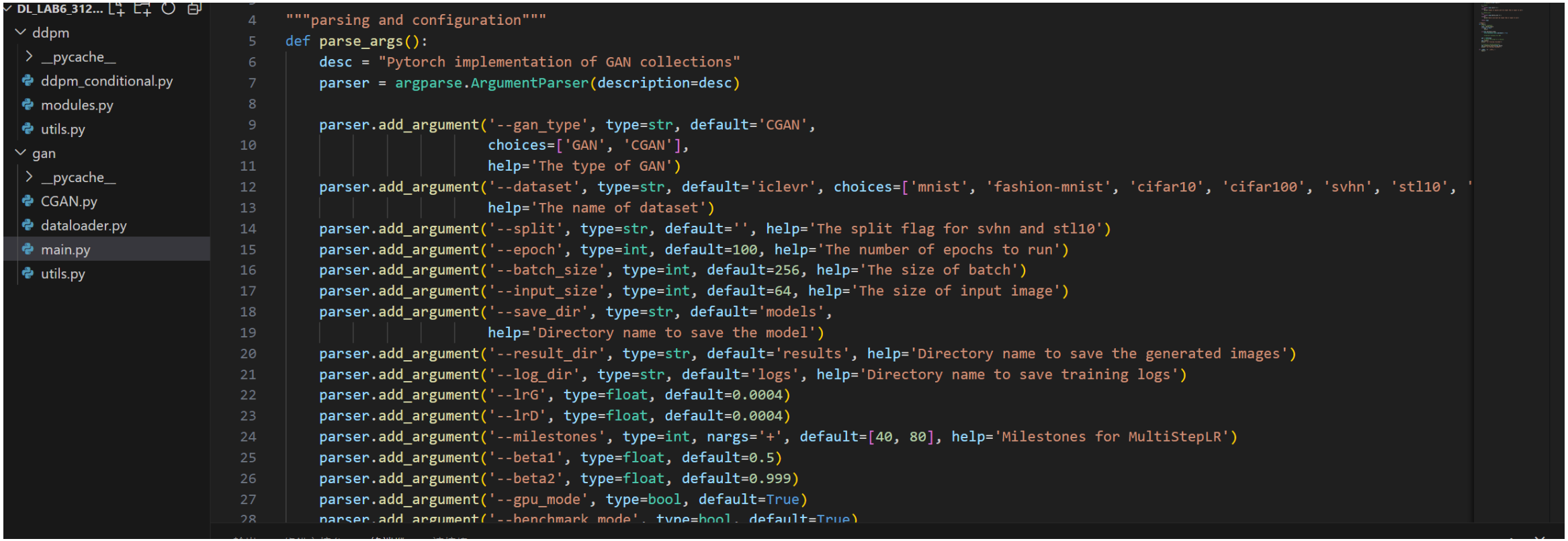
Results and discussion

- Result:skip
- 並未成功做出實驗結果

Results and discussion

- Disscussion
- 1. 額外設計GAN的實現
- **生成器（Generator）:** 生成器接收隨機噪聲向量和條件標籤作為輸入，並通過全連接層和轉置卷積層生成指定條件下的圖像。網絡架構包括一個全連接層將噪聲向量投影到高維空間，然後通過轉置卷積層逐步上採樣生成圖像。
- **判別器（Discriminator）:** 判別器接收生成的圖像和條件標籤作為輸入，通過卷積層和全連接層進行處理，最終輸出一個標量來判斷圖像的真實性。網絡架構與生成器相反，使用了卷積層來壓縮圖像信息，並通過全連接層輸出最終的判別結果。
- **損失函數:** 使用了二元交叉熵損失（BCE Loss）來訓練生成器和判別器，優化目標是讓生成器生成的圖像無法被判別器正確區分為假圖像。
- 2. 實驗設置與訓練
- **數據集與預處理:** 採用與DDPM相同的數據集，並進行了標準的圖像預處理（如歸一化和調整大小）。
- **訓練策略:** 為了確保訓練的穩定性，我在訓練過程中引入了學習率調度器（MultiStepLR），以逐步降低生成器和判別器的學習率。初始學習率設置較高，並在多個訓練階段逐步衰減，確保模型能夠逐步收斂到最優狀態。
- **固定噪聲與條件採樣:** 在訓練過程中，為了評估生成器的性能，我固定了噪聲向量和條件標籤，定期生成樣本並保存為圖像。通過這種方式，可以直觀地觀察生成器的學習進度和生成質量的提高。

implement


DL_LAB6_312...
ddpm
__pycache__
ddpm_conditional.py
modules.py
utils.py
gan
__pycache__
CGAN.py
dataloader.py
main.py
utils.py
"""parsing and configuration"""
def parse_args():
desc = "Pytorch implementation of GAN collections"
parser = argparse.ArgumentParser(description=desc)
parser.add_argument('--gan_type', type=str, default='CGAN',
choices=['GAN', 'CGAN'],
help='The type of GAN')
parser.add_argument('--dataset', type=str, default='iclevr', choices=['mnist', 'fashion-mnist', 'cifar10', 'cifar100', 'svhn', 'stl10',
help='The name of dataset')
parser.add_argument('--split', type=str, default='', help='The split flag for svhn and stl10')
parser.add_argument('--epoch', type=int, default=100, help='The number of epochs to run')
parser.add_argument('--batch_size', type=int, default=256, help='The size of batch')
parser.add_argument('--input_size', type=int, default=64, help='The size of input image')
parser.add_argument('--save_dir', type=str, default='models',
help='Directory name to save the model')
parser.add_argument('--result_dir', type=str, default='results', help='Directory name to save the generated images')
parser.add_argument('--log_dir', type=str, default='logs', help='Directory name to save training logs')
parser.add_argument('--lrG', type=float, default=0.0004)
parser.add_argument('--lrD', type=float, default=0.0004)
parser.add_argument('--milestones', type=int, nargs='+', default=[40, 80], help='Milestones for MultiStepLR')
parser.add_argument('--beta1', type=float, default=0.5)
parser.add_argument('--beta2', type=float, default=0.999)
parser.add_argument('--gpu_mode', type=bool, default=True)
parser.add_argument('--benchmark_mode', type=bool, default=True)