PyTorch에서의 image batch loading 방법들 (CIFAR 10을 기준으로)

최규형

Data loading

- for epoch in range(#epoch):
  - for i, data in enumerate(trainloader):
    - inputs, labels = data

- torchvision의 datasets.CIFAR10을 써서 한번에 memory에 올리기
- torchvision의 datasets.ImageFolder를 써서 training시 이미지 하나씩 읽기
- torch.utils.data.Dataset으로 한번에 memory에 올리기
- torch.utils.data.Dataset으로 training시 이미지 하나씩 읽기
- torch.utils.data.TensorDataset으로 한번에 memory에 올리기

- torchvision의 datasets.CIFAR10을 써서 한번에 memory에 올리기
- torchvision의 datasets.ImageFolder를 써서 training시 이미지 하나씩 읽기

torchvision

- PyTorch에서 제공하는 vision 문제에 특화된 라이브러리
- 유명한 image dataset들 (MNIST, CIFAR10, SHVN, ImageNet, COCO 등)을 바로 다운로드해서 data feeding 할 수 있게 해 줌
- 유명한 넷 아키텍처들 (AlexNet, VGG, ResNet 등)을 사용자가 따로 만들지 않고 바로 쓸 수 있게 해 줌.
- 일반적인 image 전처리나 data augmentation 등을 사용자가 쉽게 사용할 수 있게 해 줌.
- 학습 이미지 셋에서 적절하게 batch를 만들어서 training 모듈에 제공해 줌
- import torchvision

torchvision의 datasets.CIFAR10을 써서 한번에 memory에 올리기

- trainset = torchvision.datasets.CIFAR10(root='/data/', train=True, download=True, transform=…)
- testset = torchvision.datasets.CIFAR10(root='/data/', train=False, download=True, transform=…)
- trainloader = torch.utils.data.DataLoader(trainset, batch_size=…, shuffle=True, num_workers=…)
- testloader = torch.utils.data.DataLoader(testset, batch_size=…, shuffle=False, num_workers=…)

torchvision의 datasets.ImageFolder를 써서 training시 이미지 하나씩 읽기

- trainset = torchvision.datasets.ImageFolder('/data/train', …)
- testset = torchvision.datasets.ImageFolder('/data/test', …)
- trainloader = torch.utils.data.DataLoader(trainset, …)
- testloader = torch.utils.data.DataLoader(testset, …)

torchvision의 datasets.ImageFolder를 써서 training시 이미지 하나씩 읽기

- ImageFolder(path, …) 에서의 path의 하부 구조
  - Path/
    - Class_0/
      - Img001.[png/jpg/bmp…]
      - img002.[png/jpg/bmp…]
    - Class_1/
      - Img485.[png/jpg/bmp…]
      - img188.[png/jpg/bmp…]
    - …..
    - ……
    - Class_N/
      - Img523.[png/jpg/bmp…]
      - img086.[png/jpg/bmp…]

- for epoch in range(#epoch):
  - for i, data in enumerate(trainloader):
    - inputs, labels = data

- data/train/
  - dog/
    - Img34.png
    - img2568.png
  - monkey/
    - Image24.png
    - image555.png
  - …..
  - ……
  - tiger/
    - I523.png
    - i086.png

torch.utils.data.Dataset으로 한번에 memory에 올리기

- torch.utils.data.Dataset 을 상속받는 class 를 만들고, __init__, __getitem__, __len__ 세 함수를 자체적으로 만들어 줌.
- Class Cifar10CustomMemory(torch.utils.data.Dataset):
  - def __init__(self, path, …, data_transform, …):
    - Self.transform = data_transform
    - self.list_img_label = []
    - for each file under path
      - Read image as ‘img’
      - Get the ‘label’
      - Append (img, label) to self.list_img_label
  - def __getitem__(self, index):
    - Img, label = self.list_img_label[index]
    - Img = self.transform(img)
    - return img, label
  - __len__(self):
    - return len(self.list_img_label)

torch.utils.data.Dataset으로 training시 이미지 하나씩 읽기

- torch.utils.data.Dataset 을 상속받는 class 를 만들고, __init__, __getitem__, __len__ 세 함수를 자체적으로 만들어 줌.
- Class Cifar10CustomFile(torch.utils.data.Dataset):
  - def __init__(self, path, …, data_transform, …):
    - Self.transform = data_transform
    - self.list_fn_label = []
    - for each file under path
      - Get image file name ‘fn_img’
      - Get the label ‘label’
      - Append (fn_img, label) to self.list_fn_label
  - def __getitem__(self, index):
    - fn_ Img, label = self.list_fn_label[index]
    - Read image from ‘fn_ Img’ as ‘img’
    - Img = self.transform(img)
    - return img, label
  - __len__(self):
    - return len(self.list_img_label)

torch.utils.data.TensorDataset으로 한번에 memory에 올리기

- Image vector와 label vector를 만들고, torch.utils.data.TensorDataset에 넣어줌.
- Image vector는 torch.Tensor 이어야 하고, label vector는 torch.longTesnor이어야 함.
- ts_img, ts_label = torch.Tensor(), torch.LongTensor()
- 각 image에 대해
  - 이미지를 읽어 온다 (img)
  - Transform을 한다. (img_transformed)
  - 라벨을 읽어온다. (label)
  - ts_img = torch.cat(ts_img, img_transformed)
  - ts_label = torch.cat(ts_label, label)
- trainset = torch.utils.data.TensorDataset(ts_img, ts_label)
- trainloader = torch.utils.data.DataLoader(trainset, …)