1.Model\_squeeze

Đoạn mã này là mã Python sử dụng thư viện PyTorch để định nghĩa một kiến trúc mạng neural có tên là SqueezeNet. SqueezeNet là một mạng neural được thiết kế để có kích thước nhỏ gọn hơn so với các mạng neural khác nhưng vẫn giữ được độ chính xác tương đương trong việc phân loại ảnh. Kiến trúc mạng SqueezeNet bao gồm các module Fire, mỗi module Fire gồm có một lớp squeeze layer và hai lớp expand layer.

Module Fire được định nghĩa trong lớp Fire và kế thừa lớp nn.Module của PyTorch. Module này có các tham số đầu vào như sau:

inplanes: số kênh đầu vào.

squeeze\_planes: số kênh đầu ra của lớp squeeze layer.

expand1x1\_planes: số kênh đầu ra của lớp expand layer sử dụng convolution 1x1.

expand3x3\_planes: số kênh đầu ra của lớp expand layer sử dụng convolution 3x3.

Trong đó, squeeze layer là một lớp convolution 1x1 dùng để giảm số kênh đầu vào xuống còn số kênh đầu ra của squeeze layer. Expand layer được sử dụng để tăng số kênh của đầu ra của squeeze layer bằng cách sử dụng hai lớp convolution: một convolution 1x1 và một convolution 3x3.

Lớp SqueezeNet được định nghĩa để định nghĩa toàn bộ kiến trúc mạng neural SqueezeNet. Lớp này cũng kế thừa lớp nn.Module của PyTorch. Các tham số đầu vào cho lớp SqueezeNet bao gồm:

version: phiên bản của SqueezeNet, có thể là 1.0 hoặc 1.1.

num\_classes: số lượng lớp đầu ra.

Kiến trúc mạng SqueezeNet được định nghĩa trong phương thức init() của lớp SqueezeNet. Đối với phiên bản 1.0 của SqueezeNet, kiến trúc mạng bao gồm các lớp convolution, activation và các module Fire được xếp chồng lên nhau. Đối với phiên bản 1.1 của SqueezeNet, kiến trúc mạng được đơn giản hóa bằng cách giảm số lượng kênh trong các lớp convolution và sử dụng các module Fire có kích thước nhỏ hơn.

Phương thức forward() của lớp SqueezeNet được sử dụng để thực hiện truyền ngược dòng chảy (forward pass) trong kiến trúc mạng SqueezeNet. Trong phương thức này, các tensor được đưa vào kiến trúc mạng qua các lớp convolution, activation và module Fire để thực hiện việc tính toán và truyền dữ liệu từ đầu vào đến đầu ra. Cuối cùng, đầu ra sẽ được trả về.

Trong phương thức forward(), kiến trúc mạng SqueezeNet sử dụng các phương thức được định nghĩa trong các lớp như Conv2d, ReLU, MaxPool2d, AvgPool2d để thực hiện các thao tác convolution, activation và pooling. Các module Fire được sử dụng bằng cách khởi tạo một instance của lớp Fire và đưa tensor đầu vào qua các lớp squeeze layer và expand layer của module Fire.

Với đoạn mã này, ta có thể sử dụng kiến trúc mạng neural SqueezeNet được định nghĩa để thực hiện các tác vụ phân loại ảnh trên bộ dữ liệu của mình bằng PyTorch.

2.Train model

Code trên là một ví dụ về việc sử dụng PyTorch để xây dựng một mô hình học sâu (deep learning) để phân loại hình ảnh.

Đầu tiên, chương trình import các thư viện cần thiết như PyTorch, NumPy, Matplotlib. Sau đó, chương trình định nghĩa các siêu tham số (hyperparameters) cho mô hình, chẳng hạn như số lượng epoch (num\_epochs), số lượng lớp (num\_classes), kích thước batch (batch\_size), và tỷ lệ học (learning\_rate).

Tiếp theo, chương trình tải dữ liệu và tiền xử lý ảnh bằng các chuyển đổi như chuẩn hóa và chuyển đổi ảnh sang tensor.

Sau đó, chương trình tải một mô hình đã được huấn luyện trước đó (pretrained model) và thay đổi lớp đầu ra thành số lượng lớp mong muốn (num\_classes) bằng cách thay đổi kernel size trong lớp Conv2d. Chương trình cũng định nghĩa hàm mất mát CrossEntropyLoss để đánh giá độ chính xác của mô hình và tối ưu hóa Adam.

Tiếp theo, chương trình huấn luyện mô hình bằng cách lặp lại các bước lan truyền thuận (forward), lan truyền ngược (backward) và tối ưu hóa (optimize) trên dữ liệu huấn luyện (train\_loader). Ngoài ra, chương trình sử dụng tập kiểm tra (test\_loader) để đánh giá độ chính xác của mô hình trên dữ liệu không nhìn thấy trong quá trình huấn luyện.

Cuối cùng, chương trình vẽ đồ thị biểu diễn độ chính xác và độ mất mát của mô hình trên tập huấn luyện và tập kiểm tra sau mỗi epoch.

Đây chỉ là một ví dụ về việc sử dụng PyTorch để xây dựng một mô hình học sâu. Có thể có nhiều cách tiếp cận khác nhau để xây dựng mô hình và các phương pháp khác nhau để huấn luyện mô hình, tùy thuộc vào vấn đề được giải quyết.

#load model

model=models.squeezenet1\_0(pretrained=True)

#读取参数

model.load\_state\_dict(torch.load("squeezenet1\_0-a815701f.pth")) #load pre\_train model

model.classifier[1] = nn.Conv2d(in\_channels=512, out\_channels=num\_classes,

                                kernel\_size=1)

model.num\_classes = num\_classes

#model.cuda()

Trong đoạn code này, mô hình được sử dụng là SqueezeNet với phiên bản squeezenet1\_0. Mô hình đã được huấn luyện trước đó trên bộ dữ liệu ImageNet để nhận dạng hình ảnh với 1000 lớp khác nhau.

Đoạn code đầu tiên tải mô hình đã được huấn luyện trước đó bằng cách sử dụng tham số pretrained=True.

Tiếp theo, đoạn code tiến hành tải lại mô hình bằng hàm load\_state\_dict và cung cấp đường dẫn đến file chứa trọng số của mô hình đã được huấn luyện trước đó.

Sau đó, đoạn code điều chỉnh lớp đầu ra của mô hình bằng cách thay đổi lớp classifier[1] bằng một lớp Conv2d mới với số lượng kênh đầu ra bằng num\_classes.

Cuối cùng, đoạn code cập nhật thuộc tính num\_classes của mô hình để phù hợp với số lượng lớp mới được định nghĩa.

Trong đoạn code này, mô hình đã được sử dụng là SqueezeNet với phiên bản 1.0, được huấn luyện trước trên ImageNet dataset. SqueezeNet là một kiến trúc mạng nơ-ron sâu nhẹ, có khối lượng thấp và hiệu suất cao. Mô hình này đã được huấn luyện trên tập dữ liệu ImageNet với khoảng 1 triệu hình ảnh thuộc 1000 lớp khác nhau.

Trong đoạn code này, ta chỉ thay đổi lớp phân loại cuối cùng của mô hình bằng một lớp mới với số lượng lớp bằng với số lượng lớp của tập dữ liệu mà ta đang sử dụng (num\_classes). Điều này giúp mô hình có thể dự đoán chính xác lớp của ảnh từ tập dữ liệu của chúng ta.

Còn ảnh được sử dụng để huấn luyện và kiểm tra mô hình phụ thuộc vào tập dữ liệu mà bạn đang làm việc. Nếu bạn muốn huấn luyện mô hình với tập dữ liệu của riêng mình, bạn cần tải về tập dữ liệu của mình và thay đổi phần load ảnh của đoạn code để đọc dữ liệu từ tập dữ liệu của mình.