|  |  |  |
| --- | --- | --- |
|  | BAN CƠ YẾU CHÍNH PHỦ  “HỌC VIỆN KỸ THUẬT MẬT MÃ” | Mẫu 2 |

BÁO CÁO CHUYÊN ĐỀ SỐ 1.5

“Tìm hiểu về Torch”

NHIỆM VỤ: “Nghiên cứu và ứng dụng nền tảng học sâu để xây dựng hệ thống phát hiện mã độc trực tuyến”.

Mã số: 06/2022/CB.

Cơ quan chủ trì: Học viện Kỹ thuật Mật mã

Chủ nhiệm: ThS. Lê Đức Thuận

Hà Nội - 2022

|  |  |  |
| --- | --- | --- |
|  | BAN CƠ YẾU CHÍNH PHỦ  “HỌC VIỆN KỸ THUẬT MẬT MÃ” |  |

BÁO CÁO CHUYÊN ĐỀ SỐ 1.5

“Tìm hiểu về Torch”

NHIỆM VỤ: “Nghiên cứu và ứng dụng nền tảng học sâu để xây dựng hệ thống phát hiện mã độc trực tuyến”.

Mã số: 06/2022/CB.

Cơ quan chủ trì: Học viện Kỹ thuật Mật mã

Chủ nhiệm: ThS. Lê Đức Thuận

|  |  |
| --- | --- |
| **Người thực hiện chuyên đề** | **Cơ quan chủ trì** |
| *(Họ tên và chữ ký)* | *(Họ tên và chữ ký)* |

Hà Nội - 2022

MỤC LỤC

[MỤC LỤC 1](#_Toc115870598)

[TÌM HIỂU VỀ TORCH 2](#_Toc115870599)

[1.1. Giới thiệu về Torch 2](#_Toc115870600)

[1.2. Các API 2](#_Toc115870601)

[1.2.1. torch.nn.Dropout() 2](#_Toc115870602)

[1.2.2. torch.nn.Embedding() 3](#_Toc115870603)

[1.2.3. torch.nn.RNN 4](#_Toc115870604)

[1.2.4. torch.nn.Linear() 5](#_Toc115870605)

[1.2.5. torch.nn.init.uniform\_() 5](#_Toc115870606)

[1.2.6. torch.save() 5](#_Toc115870607)

[1.2.7. torch.load() 6](#_Toc115870608)

[1.2.9. torch.multiprocessing.spawm () 6](#_Toc115870609)

[1.2.10. torch.nn.parallel.DistributedDataParallel() 7](#_Toc115870610)

[1.2.11. torch.nn.DataParallel() 10](#_Toc115870611)

[1.2.12. torch.nn.CrossEntropyLoss() 10](#_Toc115870612)

[1.2.13. torch.utils.data.Subset() 12](#_Toc115870613)

[1.2.14. torch.nn.Conv2d() 12](#_Toc115870614)

[1.2.15. torch.utils.data.DataLoader() 13](#_Toc115870615)

# TÌM HIỂU VỀ TORCH

## 1.1. Giới thiệu về Torch

PyTorch là một framework được xây dựng dựa trên python cung cấp nền tảng tính toán khoa học phục vụ lĩnh vực Deep learning. Pytorch tập trung vào 2 khả năng chính:

* Một sự thay thế cho bộ thư viện numpy để tận dụng sức mạnh tính toán của GPU.
* Một platform Deep learning phục vụ trong nghiên cứu, mang lại sự linh hoạt và tốc độ.

## 1.2. Các API

### 1.2.1. torch.nn.Dropout()

torch.nn.Dropout(p=0.5, inplace=False)

Trong quá trình huấn luyện, lấy ngẫu nhiên một số phần tử của tensor đầu vào với xác suất pbằng cách sử dụng các mẫu từ phân phối Bernoulli. Mỗi kênh sẽ bị xóa độc lập trên mỗi cuộc gọi chuyển tiếp.

Điều này đã được chứng minh là một kỹ thuật hiệu quả để điều hòa và ngăn chặn sự đồng thích ứng của các tế bào thần kinh như được mô tả trong bài báo Cải thiện mạng lưới thần kinh bằng cách ngăn chặn sự đồng thích ứng của các bộ phát hiện tính năng .

Hơn nữa, kết quả đầu ra được chia tỷ lệ theo hệ số .Trong quá trình huấn luyện. Điều này có nghĩa là trong quá trình đánh giá, mô-đun chỉ cần tính một hàm nhận dạng.

* **Tham số**
* p: xác suất của một phần tử được làm bằng không. Mặc định: 0,5
* inplace: Nếu được đặt thành True, sẽ thực hiện thao tác này tại chỗ. Mặc định:False
* **Ví dụ:**

>>> m = nn.Dropout(p=0.2)

>>> input = torch.randn(20, 16)

>>> output = m(input)

### 1.2.2. torch.nn.Embedding()

torch.nn.Embedding(num\_embeddings, embedding\_dim, padding\_idx=None, max\_norm=None, norm\_type=2.0, scale\_grad\_by\_freq=False, sparse=False, \_weight=None, device=None, dtype=None)

Một bảng tra cứu đơn giản lưu trữ các bản nhúng của một từ điển và kích thước cố định.

Mô-đun này thường được sử dụng để lưu trữ các nhúng từ và truy xuất chúng bằng cách sử dụng các chỉ mục. Đầu vào cho mô-đun là danh sách các chỉ số và đầu ra là các từ nhúng tương ứng.

* **Tham số:**
* **num\_embeddings (**[***int***](https://docs.python.org/3/library/functions.html#int)**) -** kích thước của từ điển nhúng
* **embedding\_dim (**[***int***](https://docs.python.org/3/library/functions.html#int)**) -** kích thước của mỗi vectơ nhúng
* **padding\_idx (**[***int***](https://docs.python.org/3/library/functions.html#int)***, optional*) -** Nếu được chỉ định, các mục nhập tại padding\_idxkhông đóng góp vào gradient; do đó, vectơ nhúng tại padding\_idxkhông được cập nhật trong quá trình huấn luyện, tức là nó vẫn là một “vùng đệm” cố định. Đối với một Nhúng mới được xây dựng, vectơ nhúng padding\_idxsẽ mặc định là tất cả các số không, nhưng có thể được cập nhật thành một giá trị khác để được sử dụng làm vectơ đệm.
* **max\_norm (**[***float***](https://docs.python.org/3/library/functions.html#float)***, optional*) -** Nếu cho trước, mỗi vectơ nhúng có định mức lớn hơn max\_norm sẽ được chuẩn hóa lại để có định mức max\_norm.
* **Norm\_type (**[***float***](https://docs.python.org/3/library/functions.html#float)***, optional*) -** P của chỉ tiêu p để tính toán cho max\_normtùy chọn. Mặc định 2.
* **scale\_grad\_by\_freq ( *boolean , optional*) -** Nếu được cung cấp, điều này sẽ chia tỷ lệ gradient theo nghịch đảo tần số của các từ trong mini-batch. Mặc định False.
* **sparse**  **(**[***bool***](https://docs.python.org/3/library/functions.html#bool)***, optional*)** - Nếu True, ma trận wrt gradient weightsẽ là một tensor thưa thớt.

### 1.2.3. torch.nn.RNN

torch.nn.RNN(\*args, \*\*kwargs)

****Áp dụng Elman RNN nhiều lớp với tanh hoặc ReLU không tuyến tính đối với một chuỗi đầu vào.

Đối với mỗi phần tử trong chuỗi đầu vào, mỗi lớp tính toán chức năng sau:

* Với: ht là trạng thái ẩn tại thời điểm t
* x t là đầu vào tại thời điểm t
* ht-1 là trạng thái ẩn của lớp trước đó tại thời điểm t – 1 hoặc trạng thái ẩn ban đầu tại thời điểm 0 . Nếu nonlinearity có 'relu', thì ReLU được sử dụng thay vì tanh.
* **Tham số:**
* **input\_size** - Số lượng tính năng dự kiến ​​trong đầu vào *x*
* **hidden\_size** - Số lượng đối tượng ở trạng thái ẩn *h*
* **num\_layers** - Số lớp lặp lại. Ví dụ: cài đặt num\_layers=2 có nghĩa là xếp chồng hai RNN lại với nhau để tạo thành một *RNN xếp chồng lên nhau* , với RNN thứ hai nhận kết quả đầu ra của RNN đầu tiên và tính toán kết quả cuối cùng. Mặc định: 1
* **nonlinearity** - Tính phi tuyến tính để sử dụng. Có thể là một trong hai 'tanh'hoặc 'relu'. Mặc định:'tanh'
* **bias** - Nếu False, thì lớp không sử dụng trọng số thiên vị *b\_ih* và *b\_hh* . Mặc định:True
* **batch\_first** - Nếu True, thì các tensors đầu vào và đầu ra được cung cấp dưới dạng *(batch, seq, feature)* thay vì *(seq, batch, feature)* . Lưu ý rằng điều này không áp dụng cho các trạng thái ẩn hoặc ô. Xem phần Đầu vào / Đầu ra bên dưới để biết chi tiết. Mặc định:False
* **dropout** - Nếu khác 0, giới thiệu lớp *Bỏ học* trên đầu ra của mỗi lớp RNN ngoại trừ lớp cuối cùng, với xác suất bỏ học bằng dropout. Mặc định: 0
* **bidirectional** - Nếu True, trở thành RNN hai chiều. Mặc định:False

### 1.2.4. torch.nn.Linear()

torch.nn.Linear( in\_features , out\_features , bias = True , device = None , dtype = None )

* **Tham số:**
* **in\_features** - kích thước của mỗi mẫu đầu vào
* **out\_features** - kích thước của mỗi mẫu đầu ra
* **thiên vị** - Nếu được đặt thành False, lớp sẽ không học thiên vị phụ gia. Mặc định:True

### 1.2.5. torch.nn.init.uniform\_()

torch.nn.init.uniform\_(tensor, a=0.0, b=1.0)

Điền vào Tensor đầu vào với các giá trị được rút ra từ phân phối đồng đều

* **Tham số:**
* **tensor -** n-dimensional torch.Tensor
* **a** - giới hạn dưới của phân phối đồng đều
* **b** - giới hạn trên của phân phối đều

### 1.2.6. torch.save()

torch.save(obj, f, pickle\_module=pickle, pickle\_protocol=DEFAULT\_PROTOCOL, \_use\_new\_zipfile\_serialization=True)

Lưu một đối tượng vào tệp đĩa.

* **Tham số:**
* obj - đối tượng đã lưu
* f - một đối tượng giống tệp (phải thực hiện ghi và xả) hoặc một chuỗi hoặc đối tượng os.PathLike có chứa tên tệp
* pickle\_module - mô-đun được sử dụng để lấy siêu dữ liệu và đối tượng
* pickle\_protocol - có thể được chỉ định để ghi đè giao thức mặc định

### 1.2.7. torch.load()

torch.load(f, map\_location=None, pickle\_module=pickle, \*\*pickle\_load\_args)

Tải một đối tượng được lưu với torch.save() từ một tệp.

* **Tham số**
* f - một đối tượng giống tệp (phải triển khai read(), và ) readline(), hoặc một chuỗi hoặc đối tượng os.PathLike có chứa tên tệptell()seek()
* map\_location - một hàm, [torch.device](https://pytorch.org/docs/stable/tensor_attributes.html#torch.device)chuỗi hoặc một lệnh chỉ định cách ánh xạ lại các vị trí lưu trữ
* pickle\_module - mô-đun được sử dụng để giải nén siêu dữ liệu và đối tượng (phải khớp với pickle\_moduletệp được sử dụng để tuần tự hóa)
* pickle\_load\_args - (chỉ Python 3) các đối số từ khóa tùy chọn được chuyển đến pickle\_module.load()và pickle\_module.Unpickler(), ví dụ errors=...:.

### 1.2.9. torch.multiprocessing.spawm ()

torch.multiprocessing.spawn(fn, args=(), nprocs=1, join=True, daemon=False, start\_method='spawn')

Sinh nprocsra các quy trình chạy fnvới args.

Nếu một trong các quá trình thoát với trạng thái thoát khác 0, các quá trình còn lại sẽ bị giết và một ngoại lệ được đưa ra với nguyên nhân chấm dứt. Trong trường hợp một ngoại lệ được bắt gặp trong quy trình con, nó sẽ được chuyển tiếp và truy nguyên của nó được bao gồm trong ngoại lệ được nêu ra trong quy trình mẹ.

* **Tham số:**
* fn ( *function* ) -
* Chức năng được gọi là điểm đầu vào của quá trình sinh sản. Chức năng này phải được xác định ở cấp cao nhất của mô-đun để nó có thể được chọn và tạo ra. Đây là một yêu cầu được đặt ra bởi quá trình đa xử lý .
* Hàm được gọi là , ở đâu là chỉ mục của quá trình và được truyền qua nhiều đối số.fn(i, \*args)iargs
* args ( [*tuple*](https://docs.python.org/3/library/stdtypes.html#tuple) ) - Các đối số được chuyển đến fn.
* nprocs ( [*int*](https://docs.python.org/3/library/functions.html#int) ) - Số quá trình sinh ra.
* tham gia ( [*bool*](https://docs.python.org/3/library/functions.html#bool) ) - Thực hiện tham gia chặn trên tất cả các quy trình.
* daemon ( [*bool*](https://docs.python.org/3/library/functions.html#bool) ) - Cờ daemon của các tiến trình được tạo ra. Nếu được đặt thành True, các quy trình daemonic sẽ được tạo.
* start\_method ( *string* ) - (không dùng nữa) phương thức này sẽ luôn sử dụng spawn làm phương thức bắt đầu. Để sử dụng một phương pháp bắt đầu khác, hãy sử dụng start\_processes().

### 1.2.10. torch.nn.parallel.DistributedDataParallel()

torch.nn.parallel.DistributedDataParallel(module, device\_ids=None, output\_device=None, dim=0, broadcast\_buffers=True, process\_group=None, bucket\_cap\_mb=25, find\_unused\_parameters=False, check\_reduction=False, gradient\_as\_bucket\_view=False, static\_graph=False)

Thực hiện song song dữ liệu phân tán dựa trên torch.distributedgói ở cấp mô-đun.

* **Tham số:**
* **module** ( [*Module*](https://pytorch.org/docs/stable/generated/torch.nn.Module.html#torch.nn.Module) ) - mô-đun được song song hóa
* **device\_ids** ( *danh sách python: int hoặc*[*torch.device*](https://pytorch.org/docs/stable/tensor_attributes.html#torch.device) ) -
* Thiết bị CUDA. 1) Đối với mô-đun một thiết bị, device\_idscó thể chứa chính xác một id thiết bị, đại diện cho thiết bị CUDA duy nhất có mô-đun đầu vào tương ứng với quy trình này. Ngoài ra, device\_idscũng có thể được None. 2) Đối với mô-đun đa thiết bị và mô-đun CPU, device\_idsphải được None.
* Khi nào device\_idslà Nonecho cả hai trường hợp, cả dữ liệu đầu vào cho chuyển tiếp và mô-đun thực tế phải được đặt trên đúng thiết bị. (mặc định None:)
* **output\_device** ( [*int*](https://docs.python.org/3/library/functions.html#int)*hoặc*[*torch.device*](https://pytorch.org/docs/stable/tensor_attributes.html#torch.device) ) - Vị trí thiết bị của đầu ra cho các mô-đun CUDA một thiết bị. Đối với mô-đun đa thiết bị và mô-đun CPU, nó phải như vậy None, và chính mô-đun đó quyết định vị trí đầu ra. (mặc định: device\_ids[0] cho các mô-đun thiết bị đơn)
* **broadcast\_buffers** ( [*bool*](https://docs.python.org/3/library/functions.html#bool) ) - Cờ cho phép đồng bộ hóa (phát sóng) bộ đệm của mô-đun khi bắt đầu forward chức năng. (mặc định True:)
* **process\_group** - Nhóm quy trình được sử dụng để giảm tất cả dữ liệu phân tán. Nếu None, nhóm quy trình mặc định, được tạo bởi [torch.distributed.init\_process\_group()](https://pytorch.org/docs/stable/distributed.html#torch.distributed.init_process_group), sẽ được sử dụng. (mặc định None:)
* **bucket\_cap\_mb** - DistributedDataParallelsẽ gộp các tham số vào nhiều nhóm để giảm độ dốc của mỗi nhóm có thể trùng lặp với tính toán ngược. bucket\_cap\_mbkiểm soát kích thước nhóm tính bằng MegaBytes (MB). (mặc định: 25)
* **find\_unused\_parameters** ( [*bool*](https://docs.python.org/3/library/functions.html#bool) ) - Duyệt đồ thị autograd từ tất cả các tensor có trong giá trị trả về của forwardhàm của mô-đun được bao bọc. Các thông số không nhận được độ dốc như một phần của biểu đồ này được đánh dấu trước là sẵn sàng giảm bớt. Ngoài ra, các tham số có thể đã được sử dụng trong forward chức năng của mô-đun được bọc nhưng không phải là một phần của tính toán tổn thất và do đó cũng sẽ không nhận được các gradient được đánh dấu trước là sẵn sàng giảm. (mặc định False:)
* **check\_reduction** - Đối số này không được dùng nữa.
* **gradient\_as\_bucket\_view** ( [*bool*](https://docs.python.org/3/library/functions.html#bool) ) - Khi được đặt thành True, gradient sẽ là các khung nhìn trỏ đến các allreducenhóm giao tiếp khác nhau. Điều này có thể làm giảm mức sử dụng bộ nhớ tối đa, trong đó kích thước bộ nhớ đã lưu sẽ bằng tổng kích thước gradient. Hơn nữa, nó tránh được việc sao chép giữa các gradient và allreducenhóm giao tiếp. Khi gradient là dạng xem, detach\_()không thể được gọi trên gradient. Nếu gặp những lỗi như vậy, vui lòng sửa nó bằng cách tham khảo [zero\_grad()](https://pytorch.org/docs/stable/generated/torch.optim.Optimizer.zero_grad.html#torch.optim.Optimizer.zero_grad) hàm trong torch/optim/optimizer.pynhư một giải pháp. Lưu ý rằng các gradient sẽ là các khung nhìn sau lần lặp đầu tiên, vì vậy việc tiết kiệm bộ nhớ cao nhất nên được kiểm tra sau lần lặp đầu tiên.
* **static\_graph** ( [*bool*](https://docs.python.org/3/library/functions.html#bool) ) -
* Khi được đặt thành True, DDP biết đồ thị được đào tạo là tĩnh. Đồ thị tĩnh có nghĩa là 1) Tập hợp các tham số đã sử dụng và chưa sử dụng sẽ không thay đổi trong toàn bộ vòng huấn luyện; trong trường hợp này, việc người dùng có đặt hay không không quan trọng. 2) Cách đồ thị được huấn luyện sẽ không thay đổi trong toàn bộ vòng huấn luyện (nghĩa là không có luồng điều khiển phụ thuộc vào các lần lặp). Khi static\_graph được đặt thànhfind\_unused\_parameters = TrueTrue, DDP sẽ hỗ trợ các trường hợp không thể được hỗ trợ trong quá khứ: 1) Đăng nhập trở lại. 2) Kiểm tra kích hoạt nhiều lần. 3) Kiểm tra kích hoạt khi mô hình có các tham số không sử dụng. 4) Có các tham số mô hình nằm ngoài chức năng chuyển tiếp. 5) Có khả năng cải thiện hiệu suất khi có các tham số không sử dụng, vì DDP sẽ không tìm kiếm đồ thị trong mỗi vòng lặp để phát hiện các tham số không sử dụng khi static\_graph được đặt thành True. Để kiểm tra xem bạn có thể đặt static\_graph thành hay không True, một cách là kiểm tra dữ liệu ghi nhật ký ddp khi kết thúc khóa đào tạo mô hình trước đó của bạn, nếu hầu như bạn cũng có thể đặt .ddp\_logging\_data.get("can\_set\_static\_graph") == Truestatic\_graph = True

### 1.2.11. torch.nn.DataParallel()

torch.nn.DataParallel(module, device\_ids=None, output\_device=None, dim=0)

Thực hiện song song dữ liệu ở cấp độ mô-đun.

Vùng chứa này song song hóa ứng dụng của ứng dụng đã cho modulebằng cách tách đầu vào trên các thiết bị được chỉ định bằng cách phân khúc theo thứ nguyên lô (các đối tượng khác sẽ được sao chép một lần trên mỗi thiết bị). Trong chuyển tiếp, mô-đun được sao chép trên mỗi thiết bị và mỗi bản sao xử lý một phần của đầu vào. Trong quá trình vượt qua ngược lại, các gradient từ mỗi bản sao được tổng hợp vào mô-đun ban đầu.

Kích thước lô phải lớn hơn số lượng GPU được sử dụng.

* Tham số:
* module ( [*Module*](https://pytorch.org/docs/stable/generated/torch.nn.Module.html#torch.nn.Module) ) - mô-đun được song song hóa
* device\_ids ( *danh sách python: int hoặc*[*torch.device*](https://pytorch.org/docs/stable/tensor_attributes.html#torch.device) ) - Thiết bị CUDA (mặc định: tất cả các thiết bị)
* output\_device ( [*int*](https://docs.python.org/3/library/functions.html#int)*hoặc*[*torch.device*](https://pytorch.org/docs/stable/tensor_attributes.html#torch.device) ) - vị trí thiết bị của đầu ra (mặc định: device\_ids [0])

### 1.2.12. torch.nn.CrossEntropyLoss()

torch.nn.CrossEntropyLoss(weight=None, size\_average=None, ignore\_index=- 100, reduce=None, reduction='mean', label\_smoothing=0.0)

Tiêu chí này tính toán tổn thất entropy chéo giữa đầu vào và mục tiêu.

Nó rất hữu ích khi đào tạo một bài toán phân loại với các lớp C. Nếu được cung cấp, đối số tùy chọn weightphải là 1D Tensor gán trọng số cho mỗi lớp. Điều này đặc biệt hữu ích khi bạn tập luyện không cân bằng.

* **Tham số:**
* **weight**  ( [*Tensor*](https://pytorch.org/docs/stable/tensors.html#torch.Tensor)*, optional*) - trọng lượng thay đổi tỷ lệ thủ công được cung cấp cho mỗi loại. Nếu được cung cấp, phải là Tensor cỡ *C*
* size\_average ( [*bool*](https://docs.python.org/3/library/functions.html#bool)*, optional*) - Không được dùng nữa (xem reduction). Theo mặc định, các khoản lỗ được tính trung bình cho mỗi phần tử tổn thất trong lô. Lưu ý rằng đối với một số tổn thất, có nhiều yếu tố trên mỗi mẫu. Nếu trường size\_average được đặt thành False, thay vào đó, tổn thất sẽ được tổng hợp cho mỗi lô nhỏ. Bỏ qua khi reducelà False. Mặc định:True
* ignore\_index ( [*int*](https://docs.python.org/3/library/functions.html#int)*, optional*) - Chỉ định một giá trị đích bị bỏ qua và không đóng góp vào gradient đầu vào. Khi size\_averageđó True, tổn thất được tính trung bình trên các mục tiêu không bị bỏ qua. Lưu ý rằng ignore\_indexchỉ có thể áp dụng khi đích chứa các chỉ số lớp.
* **reduce** ( [*bool*](https://docs.python.org/3/library/functions.html#bool)*, optional*) - Không được dùng nữa (xem reduction). Theo mặc định, tổn thất được tính trung bình hoặc tổng hợp qua các lần quan sát cho mỗi lô nhỏ tùy thuộc vào size\_average. Thay vào reduceđó False, trả về một khoản lỗ cho mỗi phần tử lô và bỏ qua size\_average. Mặc định:True
* **reduction** ( *chuỗi , optional*) - Chỉ định giảm áp dụng cho đầu ra: 'none'| 'mean'| 'sum'. 'none': không áp dụng mức giảm 'mean',: giá trị trung bình có trọng số của đầu ra được lấy,: 'sum'đầu ra sẽ được tính tổng. Lưu ý: size\_average và reduceđang trong quá trình không được dùng nữa và trong thời gian chờ đợi, việc chỉ định một trong hai args đó sẽ ghi đè reduction. Mặc định:'mean'
* label\_smoothing ( [*float*](https://docs.python.org/3/library/functions.html#float)*, optional*) - Một float trong [0.0, 1.0]. Chỉ định mức làm mịn khi tính toán tổn thất, trong đó 0,0 có nghĩa là không làm mịn. Các mục tiêu trở thành một hỗn hợp của sự thật cơ bản ban đầu và sự phân bố đồng đều như được mô tả trong [Tư duy lại Kiến trúc Khởi đầu cho Thị giác Máy tính](https://arxiv.org/abs/1512.00567) . Mặc định:0,0.

### 1.2.13. torch.utils.data.Subset()

torch.utils.data.Subset(dataset, indices)

Tập hợp con của tập dữ liệu tại các chỉ số được chỉ định.

* **Tham số:**
* dataset ( [*Dataset*](https://pytorch.org/docs/stable/data.html#torch.utils.data.Dataset) ) - Toàn bộ tập dữ liệu
* indices ( *trình tự* ) - Các chỉ số trong toàn bộ tập hợp được chọn cho tập hợp con

### 1.2.14. torch.nn.Conv2d()

torch.nn.Conv2d(in\_channels, out\_channels, kernel\_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding\_mode='zeros', device=None, dtype=None)

Áp dụng tích chập 2D trên tín hiệu đầu vào bao gồm một số mặt phẳng đầu vào.

* **Tham số:**
* **in\_channels** ( [*int*](https://docs.python.org/3/library/functions.html#int) ) - Số kênh trong hình ảnh đầu vào
* **out\_channels** ( [*int*](https://docs.python.org/3/library/functions.html#int) ) - Số kênh được tạo ra bởi tích chập
* **kernel\_size** ( [*int*](https://docs.python.org/3/library/functions.html#int)*hoặc*[*tuple*](https://docs.python.org/3/library/stdtypes.html#tuple) ) - Kích thước của hạt nhân đang xoay
* **stride** ( [*int*](https://docs.python.org/3/library/functions.html#int)*hoặc*[*tuple*](https://docs.python.org/3/library/stdtypes.html#tuple)*, tùy chọn* ) - Bước của tích chập. Mặc định: 1
* **padding** ( [*int*](https://docs.python.org/3/library/functions.html#int)*,*[*tuple*](https://docs.python.org/3/library/stdtypes.html#tuple)*hoặc*[*str*](https://docs.python.org/3/library/stdtypes.html#str)*, tùy chọn* ) - Padding được thêm vào tất cả bốn mặt của đầu vào. Mặc định: 0
* **padding\_mode** ( *chuỗi, tùychọn* )  -  'zeros', 'reflect', 'replicate' or 'circular'. Mặc định: 'zeros'
* **dilation** ( [*int*](https://docs.python.org/3/library/functions.html#int)*hoặc*[*tuple*](https://docs.python.org/3/library/stdtypes.html#tuple)*, tùy chọn* ) - Khoảng cách giữa các phần tử kernel. Mặc định: 1
* **groups** ( [*int*](https://docs.python.org/3/library/functions.html#int)*, tùy chọn* ) - Số lượng kết nối bị chặn từ kênh đầu vào đến kênh đầu ra. Mặc định: 1
* **bias** ( [*bool*](https://docs.python.org/3/library/functions.html#bool)*, tùy chọn* ) - Nếu True, thêm thiên vị có thể học được vào đầu ra. Mặc định:True

### 1.2.15. torch.utils.data.DataLoader()

torch.utils.data.DataLoader(dataset, batch\_size=1, shuffle=None, sampler=None, batch\_sampler=None, num\_workers=0, collate\_fn=None, pin\_memory=False, drop\_last=False, timeout=0, worker\_init\_fn=None, multiprocessing\_context=None, generator=None, \*, prefetch\_factor=2, persistent\_workers=False, pin\_memory\_device='')

Bộ tải dữ liệu. Kết hợp một tập dữ liệu và một trình lấy mẫu, đồng thời cung cấp một tập dữ liệu có thể lặp lại trên tập dữ liệu đã cho.

Bộ DataLoaderdữ liệu này hỗ trợ cả tập dữ liệu kiểu bản đồ và kiểu có thể lặp lại với tính năng tải một hoặc nhiều quá trình, tùy chỉnh thứ tự tải và tự động chia lô (đối chiếu) và ghim bộ nhớ tùy chọn.

* **Tham số:**
* **dataset** ( [*Dataset*](https://pytorch.org/docs/stable/data.html#torch.utils.data.Dataset) ) - tập dữ liệu mà từ đó tải dữ liệu.
* **batch\_size** ( [*int*](https://docs.python.org/3/library/functions.html#int)*, tùy chọn* ) - tải bao nhiêu mẫu cho mỗi lô (mặc định: 1).
* **shuffle** ( [*bool*](https://docs.python.org/3/library/functions.html#bool)*, tùy chọn* ) - được đặt thành Truedữ liệu được cải tổ lại ở mọi thời điểm (mặc định: False).
* **sampler** ( [*Sampler*](https://pytorch.org/docs/stable/data.html#torch.utils.data.Sampler)*hoặc Iterable , tùy chọn* ) - xác định chiến lược để lấy mẫu từ tập dữ liệu. Có thể là bất kỳ Iterablevới \_\_len\_\_ thực hiện. Nếu được chỉ định, shufflekhông được chỉ định.
* **batch\_sampler** ( Trình lấy [*mẫu*](https://pytorch.org/docs/stable/data.html#torch.utils.data.Sampler)*hoặc Có thể lặp lại , tùy chọn* ) - giống như sampler, nhưng trả về một loạt chỉ số tại một thời điểm. Loại trừ lẫn nhau với batch\_size, shufflesampler, và  drop\_last
* **num\_workers** ( [*int*](https://docs.python.org/3/library/functions.html#int)*, tùy chọn* ) - có bao nhiêu quy trình con sử dụng để tải dữ liệu. 0có nghĩa là dữ liệu sẽ được tải trong tiến trình chính. (mặc định: 0)
* **collate\_fn** ( có *thể gọi , tùy chọn* ) - hợp nhất danh sách các mẫu để tạo thành một loạt nhỏ các Tensor. Được sử dụng khi sử dụng tải hàng loạt từ tập dữ liệu kiểu bản đồ.
* **pin\_memory** ( [*bool*](https://docs.python.org/3/library/functions.html#bool)*, tùy chọn* ) - Nếu True, bộ tải dữ liệu sẽ sao chép các Tensors vào bộ nhớ được ghim thiết bị / CUDA trước khi trả lại chúng. Nếu các phần tử dữ liệu của bạn là một loại tùy chỉnh hoặc của bạn collate\_fntrả về một lô là một loại tùy chỉnh, hãy xem ví dụ bên dưới.
* **drop\_last** ( [*bool*](https://docs.python.org/3/library/functions.html#bool)*, tùy chọn* ) - được đặt thành Trueloại bỏ lô chưa hoàn thành cuối cùng, nếu kích thước tập dữ liệu không chia hết cho kích thước lô. Nếu Falsevà kích thước của tập dữ liệu không chia hết cho kích thước lô, thì lô cuối cùng sẽ nhỏ hơn. (mặc định False)
* **timeout** ( *số , tùy chọn* ) - nếu dương, giá trị thời gian chờ để thu thập một lô từ công nhân. Nên luôn luôn không tiêu cực. (mặc định 0)
* **worker\_init\_fn** ( có *thể gọi , tùy chọn* ) - Nếu không None, điều này sẽ được gọi trên mỗi quy trình con worker với id worker (một int in [0, num\_workers - 1]) làm đầu vào, sau khi seeding và trước khi tải dữ liệu. (mặc định: None)
* **Trình tạo** ( [*torch.Generator*](https://pytorch.org/docs/stable/generated/torch.Generator.html#torch.Generator)*, tùy chọn* ) - Nếu không None, RNG này sẽ được RandomSampler sử dụng để tạo các chỉ mục ngẫu nhiên và đa xử lý để tạo *base\_seed* cho worker. (mặc định None)
* **prefetch\_factor** ( [*int*](https://docs.python.org/3/library/functions.html#int)*, tùy chọn , đối số chỉ từ khóa* ) - Số lô được tải trước bởi mỗi công nhân. 2có nghĩa là sẽ có tổng cộng 2 \* num\_workers lô được tải trước trên tất cả các công nhân. (mặc định 2)
* **persistent\_workers** ( [*bool*](https://docs.python.org/3/library/functions.html#bool)*, tùy chọn* ) - Nếu True, trình tải dữ liệu sẽ không tắt các quy trình công nhân sau khi tập dữ liệu đã được sử dụng một lần. Điều này cho phép duy trì các cá thể *Tập dữ liệu* công nhân còn sống. (mặc định False)
* **pin\_memory\_device** ( [*str*](https://docs.python.org/3/library/stdtypes.html#str)*, tùy chọn* ) - trình tải dữ liệu sẽ sao chép các Tensors vào bộ nhớ được ghim thiết bị trước khi trả về nếu pin\_memory được đặt thành true.