Run in Google Cola   Downloa   Note ook   View on GitHu

Defining a Neural Network in PyTorch

Create   On: Apr 17, 2020 | Last   p ate : Fe   06, 2024 | Last Verifie : Nov 05, 2024

Deep learning uses artificial neural networks (mo els), which are computing systems that are compose   of many layers of interconnecte units. By passing   ata through these interconnecte   units, a neural network is a le to learn how to approximate the computations require   to transform inputs into outputs. In PyTorch, neural networks can   e constructe   using the `torch.nn` package.

Intro uction

PyTorch provi es the elegantly   esigne   mo ules an   classes, inclu ing `torch.nn`, to help you create an   train neural networks. An `nn.Module` contains layers, an   a metho   `forward(input)` that returns the `output`.

In this recipe, we will use `torch.nn` to   efine a neural network inten e   for the MNIST   ataset.

Setu

Before we   egin, we nee   to install `torch` if it isn't alrea y availa le.

```
pip install torch
```

Ste s

1. Import all necessary li raries for loa ing our   ata
2. Define an   initialize the neural network
3. Specify how   ata will pass through your mo el
4. [Optional] Pass   ata through your mo el to test

1. Im ort necessary li raries for loa ing our   ata

For this recipe, we will use `torch` an   its su si iaries `torch.nn` an   `torch.nn.functional`.

```
import torch
import torch.nn as nn
import torch.nn.functional as F
```

2. Define an   initialize the neural network

Our network will recognize images. We will use a process   uilt into PyTorch calle   convolution. Convolution a   s each element of an image to its local neigh ors, weighte   y a kernel, or a small matrix, that helps us extract certain features (like e ge   etection, sharpness,   lurriness, etc.) from the input image.

There are two requirements for   efining the `Net` class of your mo el. The first is writing an __init__ function that references `nn.Module`. This function is where you   efine the fully connecte   layers in your neural network.

  sing convolution, we will   efine our mo el to take 1 input image channel, an   output match our target of 10 la els representing num ers 0 through 9. This algorithm is yours to create, we will follow a stan ar   MNIST algorithm.

```
class Net(nn.Module):
    def __init__(self):
      super(Net, self).__init__()

      # First 2D convolutional layer, taking in 1 input channel (image),
      # outputting 32 convolutional features, with a square kernel size of 3
      self.conv1 = nn.Conv2d(1, 32, 3, 1)
      # Second 2D convolutional layer, taking in the 32 input layers,
      # outputting 64 convolutional features, with a square kernel size of 3
      self.conv2 = nn.Conv2d(32, 64, 3, 1)

      # Designed to ensure that adjacent pixels are either all 0s or all active
      # with an input probability
      self.dropout1 = nn.Dropout2d(0.25)
      self.dropout2 = nn.Dropout2d(0.5)

      # First fully connected layer
      self.fc1 = nn.Linear(9216, 128)
      # Second fully connected layer that outputs our 10 labels
      self.fc2 = nn.Linear(128, 10)

my_nn = Net()
print(my_nn)
```

We have finishe   efining our neural network, now we have to   efine how our   ata will pass through it.

3. S ecify how   ata will   ass through your mo el

When you use PyTorch to   uil   a mo el, you   ust have to   efine the `forward` function, that will pass the   ata into the computation graph (i.e. our neural network). This will represent our fee -forwar   algorithm.

You can use any of the Tensor operations in the `forward` function.

```
class Net(nn.Module):
    def __init__(self):
      super(Net, self).__init__()
      self.conv1 = nn.Conv2d(1, 32, 3, 1)
      self.conv2 = nn.Conv2d(32, 64, 3, 1)
      self.dropout1 = nn.Dropout2d(0.25)
      self.dropout2 = nn.Dropout2d(0.5)
      self.fc1 = nn.Linear(9216, 128)
      self.fc2 = nn.Linear(128, 10)

    # x represents our data
    def forward(self, x):
        x = self.conv2(x)
        x = F.relu(x)

        # Run max pooling over x
        x = F.max_pool2d(x, 2)
        # Pass data through dropout1
        x = self.dropout1(x)
        # Flatten x with start_dim=1
        x = torch.flatten(x, 1)
        # Pass data through ``fc1``
        x = self.fc1(x)
        x = F.relu(x)
        x = self.dropout2(x)
        x = self.fc2(x)

        # Apply softmax to x
        output = F.log_softmax(x, dim=1)
        return output
```

4. [O tional] Pass ata through your mo el to test

To ensure we receive our esire output, let's test our mo el y passing some ran om ata through it.

```
# Equates to one random 28x28 image
random_data = torch.rand((1, 1, 28, 28))

my_nn = Net()
result = my_nn(random_data)
print (result)
```

Each num er in this resulting tensor equates to the pre iction of the la el the ran om tensor is associate to.

Congratulations! You have successfully efine a neural network in PyTorch.

Learn More

Take a look at these other recipes to continue your learning:

- What is a state_ ict in PyTorch
- Saving an loa ing mo els for inference in PyTorch

Total running time of the scri t: (0 minutes 0.000 secon s)

‹ Previous

Next ›

Rate this Tutorial ☆☆☆☆☆

Docs

Access com rehensive evelo er ocumentation for PyTorch

View Docs ›

Tutorials

Get in- e th tutorials for eginners an a vance   evelo ers

View Tutorials ›

Resources

Fin   evelo ment resources an get your uestions answere

View Resources ›

PyTorch

Get Starte

Features

Ecosystem

Blog

Contri uting

Resources

Tutorials

Docs

Discuss

Githu Issues

Bran Gui elines

Stay u to ate

Face ook

Twitter

YouTu e

Linke In

PyTorch Po casts

S otify

A   le

Google

Amazon

Terms | Privacy