Ứng dụng Trí tuệ nhân tạo trong Nuôi trồng thủy sản

NGUYỄN HẢI TRIỀU[1]

[1] Bộ môn Kỹ thuật phần mềm,

Khoa Công nghệ thông tin, Trường ĐH Nha Trang

NhaTrang, September 2024

Training Using GPUs (Graphical Processing Units)

Để **tăng tốc** và **huấn luyện mạng neural hiệu quả hơn**, chúng ta sẽ chuyển các tính toán **tensor từ CPU sang bộ nhớ GPU**. Các phép toán cho đại số tuyến tính như *dot product*, *matrix multiplication* đặc biệt hiệu quả khi tính toán song song.

Specifications	Intel® Core™ i9-11900KB Processor	NVIDIA GeForce® RTX™ 3080 Ti
Base Clock Frequency	3.3 GHz	1.37 GHz
Cores	16 (32 threads)	10240
Memory Bandwidth	45.8 GB/s	912.1 GB/s
Floating-Point Calculations	742 GFLOPS	34.10 TFLOPS
Cost	~ $540.00	~ $1200.00

GPUs are good for parallel computations (like matrix multiplications). *Source: Sebastian Raschka, Yuxi (Hayden) Liu, Vahid Mirjalili. Machine Learning with PyTorch and Scikit-Learn.*

Hiệu quả của việc tính toán trên GPU phần lớn là *do băng thông bộ nhớ* của GPU lớn hơn rất nhiều lần so với CPU. Bên cạnh đó số nhân tính toán của GPU cũng lớn hơn nhiều.

Specifications	Intel® Core™ i9-11900KB Processor	NVIDIA GeForce® RTX™ 3080 Ti
Base Clock Frequency	3.3 GHz	1.37 GHz
Cores	16 (32 threads)	10240
Memory Bandwidth	45.8 GB/s	912.1 GB/s
Floating-Point Calculations	742 GFLOPS	34.10 TFLOPS
Cost	~ $540.00	~ $1200.00

PyTorch supports NVIDIA(*cuda*), AMD, Apple Silicon(*mps*), INTEL GPUs(*xpu*–PyTorch 2.5). Trong môn học này, các code chủ yếu sử dụng *cuda* của NVIDIA GPUs. Kiểm tra máy tính có NVIDIA GPUs (lưu ý đã cài đặt đầy đủ driver), mở *terminal* gõ lệnh: *nvidia-smi*

CUDA *là một thư viện tính toán trên GPU dành riêng cho các GPU của Nvidia.* Trong PyTorch, chuyển một tensor từ CPU sang GPU rất đơn giản, ta sử dụng phương thức: .to('cuda')

Để huấn luyện hiệu quả, ta thực hiện load dữ liệu và model vào trong bộ nhớ GPU và thực hiện tính toán trên GPU.

```
[1]: import torch

     print(torch.cuda.is_available())

[1]: True

[2]: my_tensor = torch.tensor([1., 2., 3.])
     my_tensor

[2]: tensor([1., 2., 3.])

[3]: my_tensor = my_tensor.to('cuda')
     my_tensor

[3]: tensor([1., 2., 3.], device='cuda:0')

[4]: my_tensor = my_tensor.to('cpu')
     my_tensor

[4]: tensor([1., 2., 3.])
```

Hình 1: Ví dụ chuyển đổi load dữ liệu từ CPU vào GPU (*[3]*) và ngược lại (*[4]*). Lưu ý, phải kiểm tra trạng thái GPU là True khi dùng lệnh: *torch.cuda.is_available()*

Để huấn luyện mô hình sử dụng GPU, chúng ta cần chuyển data và model vào GPU. Ví dụ, *huấn luyện mạng MLP cho bài toán phân loại chữ số viết tay*, ta cần 3 thay đổi nhỏ so với code gốc chạy trên CPU

```
import torch.nn.functional as F

torch.manual_seed(1)

1. model = PyTorchMLP(num_features=784, num_classes=10)
   model.to("cuda")

optimizer = torch.optim.SGD(model.parameters(), lr=0.05)

num_epochs = 10

for epoch in range(num_epochs):

    model = model.train()
    for batch_idx, (features, labels) in enumerate(train_loader):

2.      features = features.to("cuda")
3.      labels = labels.to("cuda")

        logits = model(features)

        loss = F.cross_entropy(logits, labels)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

Transfer the model to the GPU

Transfer the data to the GPU

Sử dụng code MLP trên tập dữ liệu MNIST chạy trên GPU như sau

```
# -*- coding: utf-8 -*-
"""MINIST-GPU.ipynb

Automatically generated by Colab.

Original file is located at
  https://colab.research.google.com/drive/1vwKfNpkkymw-43lx_cCBcWEf9pz7v9FV
"""

from torch.utils.data import DataLoader
from torchvision import datasets, transforms

"""## Kiểm tra tồn tại GPU"""

if (torch.cuda.is_available()):
  print("Ton tai GPU cua NVIDIA")
  device = 'cuda'
else:
  device = 'cpu'
```

```
20
21  def compute_accuracy(model, dataloader):
22
23    model = model.eval()
24
25    correct = 0.0
26    total_examples = 0
27
28    for idx, (features, labels) in enumerate(dataloader):
29      features = features.to(device)
30      labels = labels.to(device)
31
32      with torch.no_grad():
33        logits = model(features)
34
35      predictions = torch.argmax(logits, dim=1)
36
37      compare = labels == predictions
38      correct += torch.sum(compare)
39      total_examples += len(compare)
40
41    return correct / total_examples
42
43  import torch.nn.functional as F
```

```
44
45  torch.manual_seed(1)
46  model = PyTorchMLP(num_features=784, num_classes=10)
47  model.to(device)
48
49  optimizer = torch.optim.SGD(model.parameters(), lr=0.05)
50
51  num_epochs = 10
52
53  loss_list = []
54  train_acc_list, val_acc_list = [], []
55  for epoch in range(num_epochs):
56
57    model = model.train()
58    for batch_idx, (features, labels) in enumerate(train_loader):
59      features = features.to(device)
60      labels = labels.to(device)
61      logits = model(features)
62
63      loss = F.cross_entropy(logits, labels)
64
65      optimizer.zero_grad()
66      loss.backward()
67      optimizer.step()
```

```
68
69      if not batch_idx % 250:
70        ### LOGGING
71        print(
72          f"Epoch: {epoch+1:03d}/{num_epochs:03d}"
73          f" | Batch {batch_idx:03d}/{len(train_loader):03d}"
74          f" | Train Loss: {loss:.2f}"
75        )
76      loss_list.append(loss.item())
77
78    train_acc = compute_accuracy(model, train_loader)
79    val_acc = compute_accuracy(model, val_loader)
80    print(f"Train Acc {train_acc*100:.2f}% | Val Acc {val_acc*100:.2f}%")
81    train_acc_list.append(train_acc)
82    val_acc_list.append(val_acc)
83
84  train_acc = compute_accuracy(model, train_loader)
85  val_acc = compute_accuracy(model, val_loader)
86  test_acc = compute_accuracy(model, test_loader)
87
88  print(f"Train Acc {train_acc*100:.2f}%")
89  print(f"Val Acc {val_acc*100:.2f}%")
90  print(f"Test Acc {test_acc*100:.2f}%")
```

Lưu ý

Khi tính toán trên GPU/CPU chúng ta sử dụng biến device để thiết lập chung.

```
1  if (torch.cuda.is_available()):
2    print("Ton tai GPU cua NVIDIA")
3    device = 'cuda'
4  else:
5    device = 'cpu'
```

Chi tiết code xem ở file "*Chuong_4.4_GPU_MNIST*"

Tài liệu tham khảo

Sebastian Raschka, Yuxi (Hayden) Liu, Vahid Mirjalili
Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python (2022). Published by Packt Publishing Ltd, ISBN 978-1-80181-931-2.

Sebastian Raschka
MACHINE LEARNING Q AND AI: 30 Essential Questions and Answers on Machine Learning and AI (2024). ISBN-13: 978-1-7185-0377-9 (ebook).

LightningAI
LightningAI: PyTorch Lightning (2024) .