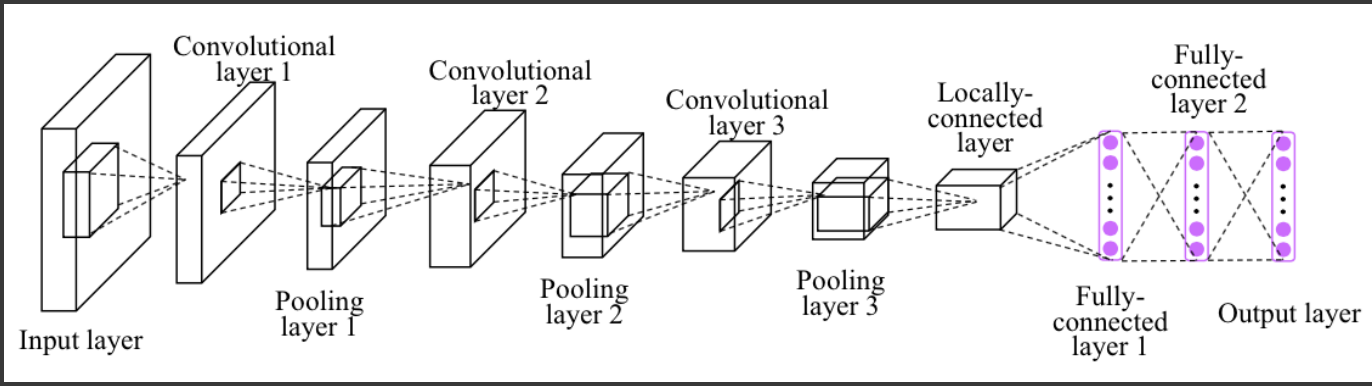
Bài tập về mạng tích chập

Trong bài này, chúng ta sẽ xây dựng một mạng tích chập sử dụng torch và thử train&test với tập MNIST nhé.

Tổng quan một mạng CNN cơ bản



MNIST dataset

Trong bài tập này, chúng ta sẽ sử dựng tập MNIST rất nổi tiếng về các chữ số viết tay từ 0->9. Tập dataset này bao gồm 60000 ảnh cho training và 10000 ảnh cho testing. Các bức ảnh này đều đã được căn giữa và chỉnh với kích thước cố định là 28x28.

Trong phần tiền xử lý, chúng ta sẽ cần chuẩn hóa các giá trị pixel của mỗi ảnh về khoảng [0,1], kiểu dữ liệu sẽ là float32

Chi tiết tại: http://yann.lecun.com/exdb/mnist/

Some configs

- Chúng ta sẽ setup một số hyper-parameters cũng như một số giá trị cần dùng theo hướng dẫn nhé
- Ở đây, mình muốn các bạn sử dụng Cuda, hãy vào runtime, rồi change the runtime type sang GPU nhé

```
# Trước hết, chúng ta import một số thư viện cần thiết đã
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchvision import datasets, transforms
import numpy as np
import matplotlib.pyplot as plt
import random
```

```
# Sô´classes trong tập MNIST
num_classes = None

# Sô´epoch
epochs = None

# Các tham sô´cần thiết cho quá trình traning.
learning_rate = None
batch_size = None
display_step = None

# Path lưu best model
checkpoint = None # có thể để dạng *.pth

# device chúng ta dùng cuda
device = 'cuda' if torch.cuda.is_available() else 'cpu'
assert device == 'cuda'
```

Dataloader

```
# Transform image
transform=transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
    ])

# load dataset từ torchvision.datasets
train_dataset = datasets.MNIST('../data', train=True, download=True,transform=None
test_dataset = datasets.MNIST('../data', train=False,transform=None)
train_loader = torch.utils.data.DataLoader(None,batch_size=None)
test_loader = torch.utils.data.DataLoader(None,batch_size=None)
```

Model

- Trong bài này, chúng ta sẽ định nghĩa một class Net, nó sẽ có cấu trúc như hình ở đầu notebook
- Bạn hãy chỉnh các tham số cho phù hợp nhé :)

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

    super(Net,self).__init__()
    self.dropout = nn.Dropout()
    self.relu = nn.ReLU()
    self.conv1 = nn.Conv2d(None,None,None,None)
    self.maxpool1 = nn.MaxPool2d(None)
    self.conv2 = nn.Conv2d(None,None,None,None)
    self.maxpool2 = nn.MaxPool2d(None)
```

```
    self.conv3 = nn.Conv2d(None,None,None,None)
    self.maxpool3 = nn.MaxPool2d(None)
    self.fc1 = nn.Linear(None,None)
    self.fc2 = nn.Linear(None,None)
    self.fc3 = nn.Linear(None,None)

  def forward(self,x):
    None # bạn hãy xem xet quá trình forward nhé
    return x
```

```
# call model, đừng quên set device nhé
model = None

# load lại pretrained model (nếu có)
try:
  None
except:
  print("!!! Hãy train để có checkpoint file")
```

```
criterion = None
optimizer = None
best_val_loss = 999

for epoch in range(1,2):
    # Quá trình training
    model.train()
    for batch_idx, (data, target) in enumerate(None):
        data, target = data.to(device), target.to(device)
        None # zero_grad()
        output = None
        loss = None
        None # backward()
        None # update
        if batch_idx % display_step == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tTrain Loss: {:.6f}'.format(
                epoch, batch_idx * len(data), len(train_loader.dataset),
                100. * batch_idx / len(train_loader), loss.item()))
    # Quá trình testing
    model.eval()
    test_loss = 0
    correct = 0
    # set no grad cho quá trình testing
    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(device), target.to(device)
            output = None
            output = None # log softmax dùng F, chú ý dim
            test_loss += None
            pred = None # argmax để lấy predicted label, chú ý dim, và keepdim = T
            correct += pred.eq(target.view_as(pred)).sum().item()
    test_loss /= len(None)
    if test_loss < best_val_loss:
      best_val_loss = test_loss
      None # lưu lại model
```

```
    print("***********    TEST_ACC = {}%    ***********".format(correct))
```

```
# load lại model đã train
None
# Set eval phase nhé bạn
None
```

```
item = iter(test_loader)
```

```
data,target = item.next() # lấy một batch ra
```

```
test_idx = random.choice(range(len(data))) # lấy index của một phần tử của một bat
```

```
data = data[test_idx]
target = target[test_idx]
assert data.shape == (1,28,28)
```

```
# thử predict

def plot(data,model):
  data = None # unsqueeze data, đồng thời set device
  output = None
  output = None # log softmax, chú ý dim
  pred = None # argmax, chú ý keepdim
  print("Predict Number : ", pred[0][0].detach().cpu().numpy())
  plt.imshow(data[0][0].detach().cpu().numpy(),cmap='gray')
  plt.show()
```

```
plot(data,model)
```

```
Predict Number :  2
```