0. Import Dependencies

In [1]:

```
from __future__ import print_function
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchvision import datasets, transforms
import matplotlib.pyplot as plt

import pandas as pd
import numpy as np
import scipy
import matplotlib.pyplot as plt

import torchvision.models as models
from ptflops import get_model_complexity_info
```

1. MNIST classification

main references:

```
 https://juejin.im/entry/5a6463a86fb9a01ca5609228
 https://github.com/Sylar257/My-data-science-tool-
kit/blob/458a96239a11145377710b90b455e8a81a365e7e/145_PyTorch_Tricks.ipynb
 https://github.com/sovrasov/flops-counter.pytorch
```

e) Download file YourEmail_in.csv from the course drive. File contains 5x5 array of integers X. Manualy compute valid convolution

of w and X and submit output in file named YourEmail_out.csv (note the difference between convolution and cross-correlation, see https://en.wikipedia.org/wiki/Convolution).

In [2]:

```
csv1=pd.read_csv('XZHANG6@TCD.IE_in.csv')
csv1.head(5)
```

Out[2]:

	214	54	89	143	169
0	4	38	72	87	74
1	66	90	88	104	242
2	103	210	51	141	145
3	76	179	19	211	80

In [3]:

```
input_data=[[214,54,89,143,169],
                 [4,38,72,87,74],
                 [66,90,88,104,242],
                 [103,210,51,141,145],
                 [76,179,19,211,80]]

weights_data=[[ 1, 0, 1],
                     [0, 0, 0],
                     [ -1,0, -1]]
```

In [4]:

```
weights_data1=weights_data[::-1]
print(weights_data1)
```

```
[[-1, 0, -1], [0, 0, 0], [1, 0, 1]]
```

In [5]:

```
def compute_conv(fm,kernel):
    h,w=fm.shape
    k,k=kernel.shape
    r=int(k/2)
    #define the map after no padding
    padding_fm=np.zeros([h,w],np.int)
    #Remain the calculating output
    rs=np.zeros([h-2,w-2],np.float32)
    #Assign the input result to the specified area, that is, the remaining area except 4 boundarie
s
    padding_fm[0:h,0:w]=fm
    #Traversing the area centered on each point
    for i in range(1,h-1):
        for j in range(1,w-1):
            #Take out the k*k area centered on the current point
            roi=padding_fm[i-r:i+r+1,j-r:j+r+1]
            #Calculate the convolution of the current point, multiply after k*k points
            rs[i-1][j-1]=np.sum(roi*kernel)

    return rs
```

In [6]:

```
def my_conv2d(input,weights):
    #shape=[h,w]
    h,w=input.shape
    #shape=[k,k]
    k,k=weights.shape
    outputs=np.zeros([h-2,w-2],np.int)
    f_map=input
    w=weights
    rs =compute_conv(f_map,w)
    outputs=outputs+rs

    return outputs
```

In [7]:

```
def main():
    input = np.asarray(input_data,np.int)
    weights =  np.asarray(weights_data1,np.int)
    rs = my_conv2d(input,weights)
    print(rs)
    final1=pd.DataFrame(rs)
    #final1.drop([0], axis=0)
    final1.to_csv('XZHANG6@TCD.IE_out22.csv',index=None,header=None)

if __name__=='__main__':
    main()
```

```
[[-149.   -3.   72.]
 [  78.  226.   50.]
 [ -59.  196. -231.]]
```

f) Download file YourEmail_network.csv from the course drive. File contains network definiton in the following format:

```
i) Convolution, N, K - convolution layer with N feature maps and KxK filters

ii) Maxpool, K - max pooling layer KxK pooling window

iii) Fully-connected, N - fully connected layer with N neurons
```

All convolutions use same padding that preserves input feature size. All convolutional and fully-connected layers use bias. The network is applied on 32x32 RGB images (3 input channels). Calculate the number of parameters and the number of FLOPS of this network and submit them in YourEmail_params.csv each of the 2 values in separate row (number of parameters in the first row).

In [8]:

```
csv2=pd.read_csv('XZHANG6@TCD.IE_network.csv')
csv2.head(10)
```

Out[8]:

	convolution	24	5
0	convolution	60	3.0
1	maxpool	2	NaN
2	convolution	72	3.0
3	maxpool	4	NaN
4	fully-connected	160	NaN
5	fully-connected	10	NaN

In [9]:

```
class CNN4(nn.Module):
    def __init__(self):
        super(CNN4, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=24,
                               kernel_size=5, stride=1, padding=2)
        self.conv2 = nn.Conv2d(in_channels=24, out_channels=60,
                               kernel_size=3, stride=1, padding=1)
        self.pool1 = nn.MaxPool2d(2)
        self.conv3 = nn.Conv2d(in_channels=60, out_channels=72,
                               kernel_size=3, stride=1, padding=1)
        self.pool2 = nn.MaxPool2d(4)
        self.fc1 = nn.Linear(in_features=1152, out_features=160)
        self.fc2 = nn.Linear(in_features=160, out_features=10)

        nn.init.kaiming_normal_(self.conv1.weight, nonlinearity='relu')
        nn.init.kaiming_normal_(self.conv2.weight, nonlinearity='relu')
        nn.init.kaiming_normal_(self.conv3.weight, nonlinearity='relu')
        nn.init.kaiming_normal_(self.fc1.weight, nonlinearity='relu')
        nn.init.kaiming_normal_(self.fc2.weight, nonlinearity='linear')

    def forward(self, x):
        x = self.conv1(x)
        x = F.relu(x)
        x = self.conv2(x)
        x = F.relu(x)
        x = self.pool1(x)
        x = self.conv3(x)
        x = F.relu(x)
        x = self.pool2(x)

        x = x.view(-1, 1152)
        x = self.fc1(x)
        x = F.relu(x)
        x = self.fc2(x)

        return F.log_softmax(x, dim=1)
```

In [10]:

```
use_cuda = torch.cuda.is_available()
device = torch.device("cuda" if use_cuda else "cpu")
model = CNN4().to(device)
flop,params=get_model_complexity_info(model, (3,32,32))
print(f'Params: {params}')
print(f'Flop: {flop}')
```

```
CNN4(
  0.025 GMac, 100.000% MACs,
  (conv1): Conv2d(0.002 GMac, 7.343% MACs, 3, 24, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2
))
  (conv2): Conv2d(0.013 GMac, 52.412% MACs, 24, 60, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
  (pool1): MaxPool2d(0.0 GMac, 0.242% MACs, kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  (conv3): Conv2d(0.01 GMac, 39.200% MACs, 60, 72, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
  (pool2): MaxPool2d(0.0 GMac, 0.072% MACs, kernel_size=4, stride=4, padding=0, dilation=1,
ceil_mode=False)
  (fc1): Linear(0.0 GMac, 0.725% MACs, in_features=1152, out_features=160, bias=True)
  (fc2): Linear(0.0 GMac, 0.006% MACs, in_features=160, out_features=10, bias=True)
)
Params: 239.89 k
Flop: 0.03 GMac
```

In [11]:

```
final2=pd.DataFrame(data={
        'parameters':'239.89k','FLOPS':'0.03GMac',},index=[0])
final2.to_csv('XZHANG6@TCD.IE_params.csv',index=None)
```