Cain's AI WP

历经折磨和kyr师傅的探讨，终于写完了整个AI分类，把我的WP供大家参考，欢迎找我交流捏~欢迎加入N0wayBack联合战队捏~
其他的所有题目WP都可以在队内全栈师傅Lazzzaro的博客找到(lazzzaro.github.io)

EZ_MLP

修复这个神经网络代码

```
# Hints:
#   > Fix the bug in the code to get the flag, only one line of code needs to be changed.
#   > Understand the code and the figure(Example.jpg) before flag submission.
#   > Example.jpg is only for tutorial and demonstration, no hidden information contained.
```

直接运行看看有啥不对

```
PS C:\Users\17845\Desktop\CTF\moectf2023\EZ_MLP\_Attachments> & C:/Users/17845/AppData/Local/Programs/Python/Python39/python.exe c:/Users/17845/Desktop/CTF
Traceback (most recent call last):
  File "c:\Users\17845\Desktop\CTF\moectf2023\EZ_MLP\_Attachments\run.py", line 25, in <module>
    y = forward(inputs[i])
  File "c:\Users\17845\Desktop\CTF\moectf2023\EZ_MLP\_Attachments\run.py", line 8, in forward
    z1 = fc(x, w1, b1)
  File "c:\Users\17845\Desktop\CTF\moectf2023\EZ_MLP\_Attachments\run.py", line 4, in fc
    return np.matmul(x, weight) + bias
ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with gufunc signature (n?,k),(k,m?)->(n?,m?) (size 4 is different from 1)
```

报错代码在np.matmul出问题了，百度查资料明白这个是矩阵乘法

报错信息提示维度有问题，在进行函数前一步添加调试代码看一下

```
def fc(x, weight, bias):
    return np.matmul(x, weight) + bias

def forward(x):
    print(x,'\n\n\n',w1)
    z1 = fc(x, w1, b1)
    z2 = fc(z1, w2, b2)
    y = fc(z2, w3, b3)
    return y
```

```
[[  8.08270469]
 [-12.91655347]
 [ -8.77295518]
 [  7.99327399]]


 [[-0.47944349 -0.60346999 -0.96774049 -1.77313615]
 [ 0.06037517 -0.44007958  0.42612877 -0.20750222]
 [ 0.84556701 -0.76486991  0.64091193 -1.1969599 ]
 [ 0.24803209 -0.43146066 -0.99032811 -1.28007669]]
  File "c:\Users\17845\Desktop\CTF\moectf2023\EZ_MLP\_Attachments\run.py", line 25, in <module>
    y = forward(inputs[i])
  File "c:\Users\17845\Desktop\CTF\moectf2023\EZ_MLP\_Attachments\run.py", line 8, in forward
    z1 = fc(x, w1, b1)
  File "c:\Users\17845\Desktop\CTF\moectf2023\EZ_MLP\_Attachments\run.py", line 4, in fc
    return np.matmul(x, weight) + bias
```

发现是 4*1 X 4*4 当然不行，矩阵乘法应该是k*n X n*j

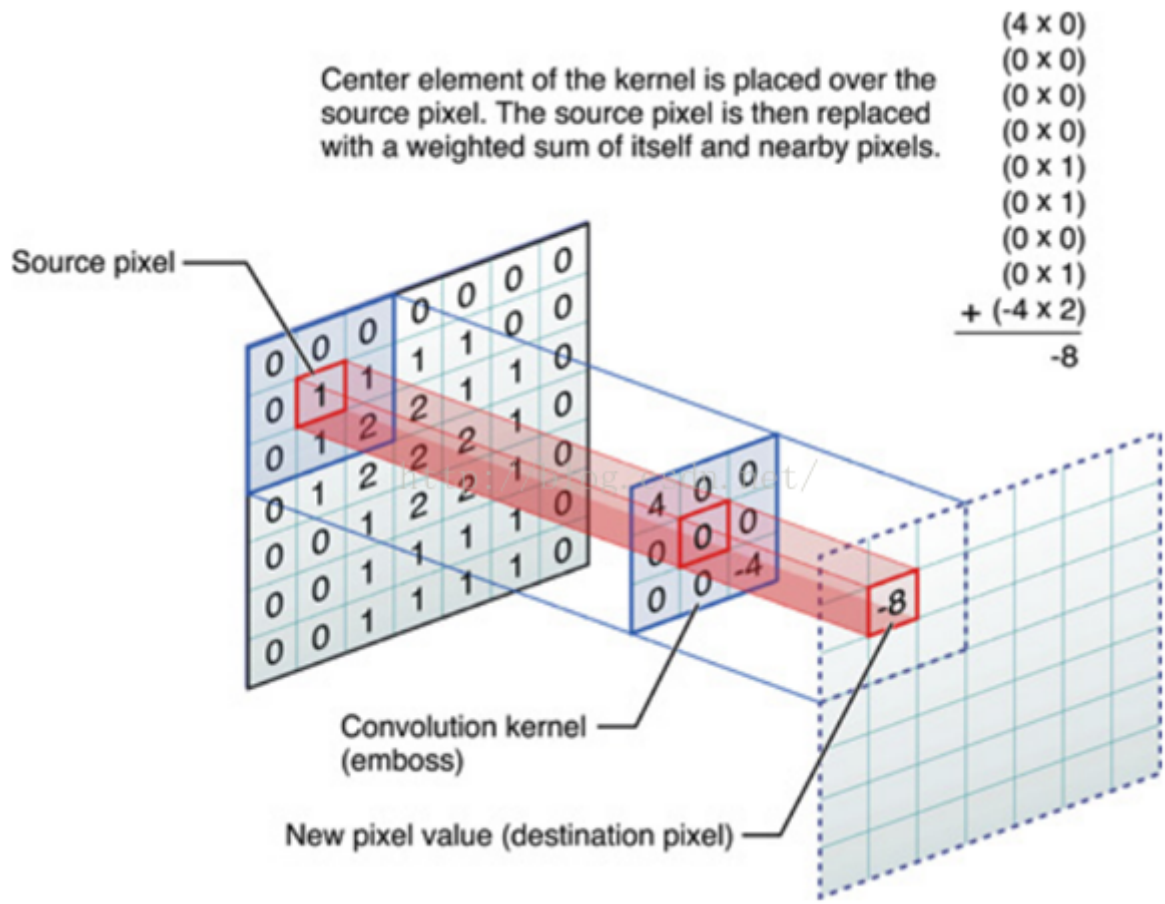
交换函数的参数位置

```
return np.matmul(weight,x) + bias
```

得到flag

EZ_Conv

要求完成一个卷积操作，卷积主要是将矩阵特征化



代码如下

```
from PIL import Image
import numpy as np
import torch
import torch.nn as nn

w = torch.tensor(np.load('npys/w.npy'))
b = torch.tensor(np.load('npys/b.npy'))
inp = torch.tensor(np.load('npys/inp.npy'))

conv_layer = nn.Conv2d(1, 25, kernel_size=5)
conv_layer.weight.data = w
conv_layer.bias.data = b

output = conv_layer(inp)

output = output.view(1, 25, 25)
output = (output * 255).byte()
output_np = output.cpu().numpy()
H, W = output_np.shape[1], output_np.shape[2]
```

```
image_data = output_np[0]

image = Image.fromarray(image_data)
image.save('output_image.png')
image.show()
```

在使用下面我自己写的conv进行操作时，内核本身没有问题,问题在于最后一步的压缩，view函数的压缩是展开为一维重新组织

```
def conv(x, w, b):
    output = np.zeros((25, 5, 5))
    for k in range(25):
        for i in range(5):
            for j in range(5):
                    window = x[:, i :i  + 5, j :j  + 5]
                    output[k, i, j] = np.sum(window * w[k, :, :, :]) + b[k]

    return output
```

classification

要求实现一个图像分类网络，最简单的ResNet即可实现，根据hint也只有34和50能完成

```
from torchvision import transforms,models
from PIL import Image
import numpy as np
import os

normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,
0.225])
transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    normalize,
])

model = models.resnet50(pretrained=True)
model.eval()
model.to('cpu')

flag = ""
salt = np.load("salt.npy")
labels = []
path = ".\\imgs\\"

for i in range(60):
    img_path = path + str(i) + ".png"
    img = Image.open(img_path)
    img = transform(img)
    img.unsqueeze_(0)
    label = model(img).argmax().item()
    labels.append(label)
```

```
print(labels)
print(salt)

for i in range(60):
    num = labels[i] - salt[i]
    if num<0:
        num += 1000

    if num>127:
        print(chr(32),end='')
    else:
        print(chr(num),end='')
    #labels.append((ord(flag[i]) + salt[i])%1000)


#Hint : 3 4 6 3
#ResNet34  : Great_ AHmi g_H _QOod_CV_M4N_1$9Bfkh! @UWF6vRxqw#LQ7z 2Jb8YD
#ResNet50  : Great_KAIming_H3_gOod_CV_M4N_1$9Bfkh!T@UWF6vRxqw#LQ7zO2Jb8YD
Bingo!
```

HappyMLP

进一个感知机的逆向，很简单，线代稍微学一下就行

```
import numpy as np
import torch
from model import Net

def chr2float(c):
    return ord(c) / 255. * 2. - 1

def float2chr(f):
    return chr(round((f + 1) / 2.0 * 255.0))

def verify(flag_path):

    def verify_flag(flag_tensor):
        checkpoint = torch.load('_Checkpoint.pth')
        net = Net()
        net.load_state_dict(checkpoint['model'])
        base_input = checkpoint['input']
        output = net(base_input + flag_tensor)
        return torch.equal(torch.round(output.detach(), decimals=5),
torch.tensor([2., 3., 3.]))

    def show_flag(flag_tensor):
        flag = ''
        for f in np.array(flag_tensor).ravel():
            flag += float2chr(f)
        print('moectf{' + flag + '}')

    try:
        flag_tensor = torch.load(flag_path).detach()
    except:
```

```
        print('Invalid flag path.')
        return
    if verify_flag(flag_tensor):
        print('You made this little MLP happy, here\'s his reward:')
        show_flag(flag_tensor)
    else:
        print('Sad :(')

def re_sigmoid(x):
    return torch.log(x / (1 - x))

def re_linear(x,w,b):
    return torch.matmul(x - b, torch.pinverse(w.t()))

def re_scale(x):
    scale = 40.0
    return (x + scale / 2.) / scale

if __name__ == '__main__':

    checkpoint = torch.load('_Checkpoint.pth')
    base_input = checkpoint['input']
    output = torch.tensor([2.0000000000, 3.00000000000, 3.000000000])

    output = re_linear(output, checkpoint['model']['fc3.weight'],
checkpoint['model']['fc3.bias'])
    output = re_scale(output)
    output = re_sigmoid(output)
    output = re_linear(output, checkpoint['model']['fc2.weight'],
checkpoint['model']['fc2.bias'])
    output = re_scale(output)
    output = re_sigmoid(output)
    output = re_linear(output, checkpoint['model']['fc1.weight'],
checkpoint['model']['fc1.bias'])
    output = output - base_input
    print(output)
    torch.save(output, "_Flag.pth")

    verify('_Flag.pth')
```

VisualHacker

msg使用ook解码可以看到相关hint

整体要求是一个完成一个L2CS网络的应用，然后根据应用场景实例化

```
from PIL import Image
import torch
from transform import transform
from model import L2CS
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import os
```

```
def judge_angle(x,a,b,len):
    if x > b:
        return judge_angle(x-len, a, b, len)
    if x < a:
        return judge_angle(x+len, a, b, len)
    return x

def InitModel():
    checkpoint = torch.load(".\\checkpoint.pkl")
    model = L2CS()
    model.load_state_dict(checkpoint)
    return model

def get_pitch_yaw(original_img, model):
    def raw_data_process(gaze_pitch, gaze_yaw):
        # gaze_pitch, gaze_yaw: torch.Tensor
        # return processed pitch, yaw
        idx_tensor = [idx for idx in range(90)]
        idx_tensor = torch.FloatTensor(idx_tensor)
        softmax = torch.nn.Softmax(dim=1)

        pitch_predicted = softmax(gaze_pitch)
        yaw_predicted = softmax(gaze_yaw)

        pitch_predicted = torch.sum(pitch_predicted * idx_tensor, 1).cpu() * 4 -
180
        yaw_predicted = torch.sum(yaw_predicted * idx_tensor, 1).cpu() * 4 - 180

        pitch_predicted = pitch_predicted*np.pi/180
        yaw_predicted = yaw_predicted*np.pi/180
        return pitch_predicted, yaw_predicted

    gaze_yaw, gaze_pitch = model(transform(original_img).unsqueeze_(0))
    return raw_data_process(gaze_pitch, gaze_yaw)

def grometry_transform(pitch, yaw):
    # Covert pitch, yaw to intersection point on the input screen
    gaze_gt = np.zeros([3])
    gaze_gt[0] = -torch.cos(yaw) * torch.sin(pitch)
    gaze_gt[1] = -torch.sin(yaw)
    gaze_gt[2] = -torch.cos(yaw) * torch.cos(pitch)
    return gaze_gt

model = InitModel()
for i in range(1,2):
    folder_path = f'.\\Captures\\{i}\\'
    files = os.listdir(folder_path)
    file_count = len(files)
    xl = []
    yl = []
    #zl = []
    plt.figure()
    for j in range(file_count):
        img_path = f".\\Captures\\{i}\\{j}.png"
```

```
        img = Image.open(img_path)
        pitch, yaw = get_pitch_yaw(img,model)
        x,y,z = grometry_transform(pitch, yaw)
        xl.append(x)
        yl.append(y)
        #zl.append(z)
        #print(j,gaze)
    plt.scatter(xl,yl,s=5)
    r = (-1.5,1.5)
    plt.xlim(r)
    plt.ylim(r)
    plt.savefig(f"fig_{i}.png")
    print(i)
    plt.show()
```

ABC

提示code在A*B*C中，那么就乘起来看下，但是根据第一问，我们知道要先看维度找顺序（然而并不用

```
a = np.load('A.npy')
b = np.load('B.npy')
c = np.load('C.npy')
print(len(a),len(a[0]),len(b),len(b[0]),len(c),len(c[0]))

flag = np.matmul(a,b)
flag = np.matmul(flag,c)
```

然后看看最后是个什么

```
print(len(flag),len(flag[0]))
print(flag)
#29 29
```

```
问题    输出    调试控制台    终端

  -1.  1.  1.  1. -1.  1. -1.  1.  1.  1. -1.]
 [-1. -1. -1. -1.  1.  1. -1. -1.  1.  1.  1.  1.  1. -1. -1. -1.  1.  1.
   1.  1.  1.  1. -1.  1.  1. -1.  1.  1.  1.]
 [ 1.  1.  1.  1. -1.  1.  1.  1. -1. -1.  1.  1.  1.  1.  1. -1. -1.  1.
   1. -1.  1. -1.  1. -1.  1. -1. -1. -1. -1.]
 [-1.  1.  1. -1. -1. -1. -1.  1.  1. -1. -1.  1.  1. -1. -1. -1. -1. -1.
   1.  1. -1. -1. -1. -1. -1.  1.  1. -1.  1.]
 [ 1.  1.  1.  1.  1.  1.  1.  1. -1.  1.  1.  1.  1. -1.  1. -1. -1.  1.
  -1. -1. -1.  1.  1.  1. -1.  1.  1.  1.  1.]
 [-1. -1. -1. -1. -1. -1. -1.  1.  1.  1. -1. -1.  1.  1.  1.  1.  1. -1.
   1.  1. -1.  1. -1.  1. -1.  1.  1. -1. -1.]
 [-1.  1.  1.  1.  1.  1. -1.  1. -1. -1.  1. -1.  1. -1.  1. -1. -1.  1.
  -1.  1. -1.  1.  1.  1. -1.  1. -1. -1. -1.]
 [-1.  1. -1. -1. -1.  1. -1.  1.  1. -1.  1.  1.  1. -1. -1. -1.  1. -1.
   1. -1. -1. -1. -1. -1. -1. -1.  1.  1.]
 [-1.  1. -1. -1. -1.  1. -1.  1. -1.  1. -1. -1. -1. -1.  1. -1.  1. -1.
   1. -1. -1.  1. -1. -1. -1. -1. -1. -1. -1.]
 [-1.  1. -1. -1. -1.  1. -1.  1.  1.  1. -1.  1. -1.  1.  1.  1. -1.  1.
```

得到了这样的一个29*29矩阵，里面非-1即1

很容易联想到二维码，画图（这个就别用我的代码了，Cain独特的画图代码

```
qr = np.zeros((29,29))
for i in range(len(flag)):
    for j in range(len(flag)):
        if(round(flag[i][j])==1):
            qr[i, j] = 255

Image.fromarray(qr).show()
```

最后的到二维码，扫一扫即可