

# CTF-Hgame\_accuracy\_writeup

原创

[involute](#) 于 2021-07-29 12:21:38 发布 80 收藏

分类专栏: [人工智能](#) [CTF](#) [算法](#) 文章标签: [人工智能](#) [深度学习](#) [信息安全](#) [python](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: [https://blog.csdn.net/qq\\_19917367/article/details/119208469](https://blog.csdn.net/qq_19917367/article/details/119208469)

版权



[人工智能](#) 同时被 3 个专栏收录

10 篇文章 1 订阅

订阅专栏



[CTF](#)

1 篇文章 0 订阅

订阅专栏



[算法](#)

12 篇文章 0 订阅

订阅专栏

把相关的代码记录一下, 方便以后使用

## accuracy

石碑上的文字, 究竟隐藏着怎样的秘密.....

## Write Up

题目给出两个文件, chars.csv dataset.csv,

简单查看之后, dataset是数据集, chars是测试集

```
df.head()
  label  pixel0  pixel1  pixel2  pixel3  pixel4  pixel5  pixel6  pixel7  pixel8  ...  pixel774  pixel775  pixel776  pixel777  pixel778  pixel779  pixel780  pixel781  pixel782  pixel783
0    10     0     0     0     0     0     0     0     0     0  ...     0     0     0     0     0     0     0     0     0     0
1    10     0     0     0     0     0     0     0     0     0  ...     0     0     0     0     0     0     0     0     0     0
2    10     0     0     0     0     0     0     0     0     0  ...     0     0     0     0     0     0     0     0     0     0
3    10     0     0     0     0     0     0     0     0     0  ...     0     0     0     0     0     0     0     0     0     0
4    10     0     0     0     0     0     0     0     0     0  ...     0     0     0     0     0     0     0     0     0     0
5 rows x 785 columns
```

数据集是由784个pixel, 即 28 \* 28

```
df.iloc[:,0].unique()
#array([10, 11, 12, 13, 14, 15, 1, 0, 4, 7, 3, 5, 8, 9, 2, 6])
```

查看标签, 共有16个, 结合数据集比对, 应该是一串十六进制的数

接下来就是训练 + 预测

```
import torch
import pytorch_lightning as pl
from torch import nn
from torch.utils.data import Dataset
from torch.nn import functional as F
from PIL import Image, ImageChops, ImageFilter, ImageEnhance
import os
from pytorch_lightning.core.lightning import LightningModule
from pytorch_lightning import Trainer
from torch.optim import Adam
from torchvision import datasets, transforms
from torch.utils.data import DataLoader, random_split
import matplotlib.pyplot as plt
import cv2
import numpy as np
```

## 网络

```
class LitMNIST(LightningModule):
    def training_step(self, batch, batch_idx):
        x, y = batch
        logits = self(x)
        loss = F.nll_loss(logits, y)
        self.log('my_loss', loss, on_step=True, on_epoch=True, prog_bar=True, logger=True)
        return loss

    def configure_optimizers(self):
        return Adam(self.parameters(), lr=1e-3)

    def __init__(self, input_num, hidden_num, output_num):
        super().__init__()
        self.fc1 = nn.Linear(input_num, hidden_num)
        self.fc2 = nn.Linear(hidden_num, output_num)
        self.relu = nn.ReLU()

    def forward(self, x):
        #batch_size, channels, width, height = x.size()
        # (b, 1, 28, 28) -> (b, 1*28*28)
        #x = x.view(batch_size, -1)
        x = x.view(x.shape[0], -1)
        x = self.fc1(x)
        x = self.relu(x)
        y = self.fc2(x)
        return F.log_softmax(y, dim=1)
```

## 加载数据

```

transform=transforms.Compose([
                                transforms.ToTensor()
                            ])
class MySet(Dataset):
    # 读取数据
    def __init__(self, df):
        self.df = df
    # 根据索引返回数据
    def __getitem__(self, index):
        image = Image.fromarray(np.uint8(self.df.iloc[index, 1:]).reshape(28,28))
        label = self.df.iloc[index,0]
        return transform(image),label
    # 返回数据集总长度
    def __len__(self):
        return self.df.shape[0]

```

```

train_data = MySet(df)
train = DataLoader(train_data, batch_size=64, shuffle=True)

```

## 训练

```

input_num = 784
hidden_num = 500
output_num = 16
#optimizer = Adam(LitMNIST(input_num, hidden_num, output_num).parameters(), lr=1e-2)
model = LitMNIST(input_num, hidden_num, output_num)
trainer = Trainer(gpus=0, max_epochs = )
trainer.fit(model, train)

```

## 预测

```

import numpy as np
ceshi = Image.open('./chars/87.png').convert('L')
ceshi = ImageChops.invert(ceshi)
#ceshi = Image.fromarray(np.uint8(df.iloc[1150, 1:]).reshape(28,28))
img = transform(ceshi)
img = img.unsqueeze(0) #图片扩展多一维,因为输入到保存的模型中是4维的[batch_size,通道,长,宽],而普通图片只有三维,[通道,长,宽]
#扩展后,为[1, 1, 28, 28]
print(img.shape)
output = model(img).detach().numpy()
pred = np.argmax(output) #选出概率最大的一个
print(pred.item())

```

```

file_dir = './chars/'
filenames = os.listdir(file_dir)
filenames = sorted(filenames, key = lambda x: int(x.strip('.png')))
dict_ = {0:0, 1:1, 2:2, 3:3, 4:4, 5:5, 6:6, 7:7, 8:8, 9:9, 10:'A', 11:'B', 12:'C', 13:'D', 14:'E', 15:'F'}
def predict(filename):
    ceshi = Image.open(file_dir + filename).convert('L')
    ceshi = ImageChops.invert(ceshi)
    img = transform(ceshi)
    img = img.unsqueeze(0)
    output = model(img).detach().numpy()
    pred = np.argmax(output) # 选出概率最大的一个
    return dict_[pred.item()]
result = ''
for filename in filenames:
    result += str(predict(filename))
print(result)

```

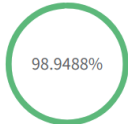
```

key 0E0FA00B0B7A5F5C0179C60C344E0053C4E0690010DF159047UC090B0004182B505AU/8809057/BC705250BEB7EED51744EA14E0950FD852I/430801767D054980E17B5370FD71E7957F5C00534A7
51F54C06BC14EE58BD760C562B5632165AD5F26542B6CEA66F45F2067009AD88D356D416D6A66488C22905397EB969451E04E167389697C8BD5624D6B4C548F7433740551A07EDD4E864E4C88635DF7
8C014E0888E98CE7D6E8368836164795C3D843D96EA832B832B53575317671D82B1672851708C014E0791CC51735C7166825FD85973513F6A21683758B64E0E56FD90FD62C55F53518D4EB2624B645
84E089EC46CB3843D65E59ED15C7198DE971C94C1753263E94EAE820D65E765F64E9188F3968D72EC5B644F3D7F574E714E164E395FC378A788406EE181547075601D53438F6C4E00610F8F85968B67
686CB35C217EB788C27EC8987B5165638C767E4E07660E73E062AC624B8D0540D5C06551068665219592990385174904498DE50AC82B14E00573A72B988B077086731621078A76DFB60C6600551F09
7F34E007FD60CA8083671D7EB265E05A577A7A789181EA4EDD8D1F50145F3A55104E0A5B985A49513F8C014E008EAB98CE96C55949541B738B63017D205FC36E05660E79E459294E0B624D90CE9047
540895267EE39EC47CB17EB77EAD4E2D97598D4F588B674E6E057167540E676567094E00751F660E65F76F31738956599994E4C6C5F65C16B4C9738738B7ED34E004EE36E0596C565ED53CC4E3E4E1
6886351A09F50671B8FBD84277EF0800C71D571D54EBE98DE600E7518968F6CE24F4E66027765776871D54E9167084E0A588B7A466842B2F18C019A6C8E0F684382B1753889D2957F541F91D19F1394
FF953558F8521F4E0D8BA94E07519B4E0052516321714C714C53438F7D8F89514953E44ECA76D87B75957F4F555E7867094F60767B575A51714E3E771E89DA660E79E6826F73898C0188404F5C80ED8
10257AA8D77592D77EB9F997FD47FE08896864E7B265CD175866E05585D5E8465877687540E8C0153C85B9A4E24671D98CE6CE2638C4E094E167684886351494ECA591C62114E3A4F608F7B53D95DFE
5E3C82AC82B36D696B4C4E0066F24F20999955314E0D588C8DC5B95660E5A9A521A5F3A767E4EE379FE534534379CB699C6837884C7ECF654568A64ECA53C8676588BF59299AD86D77961476DB4E1

```

上传

Accuracy



hgame{deep\_learnIng^and&AI\*1s\$amazing#r1ght?}

传入网站，得到flag

```
hgame{deep_learnIng^and&AI*1s$amazing#r1ght?}
```

```
ult += str(predict(filename))
print(result)
```

```

[外链图片转存中...(img-9F07m4qR-1627532437078)]

传入网站，得到f1ag

```flag
hgame{deep_learnIng^and&AI*1s$amazing#r1ght?}

```