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

本文链接：https://blog.csdn.net/qq_19917367/article/details/119208469

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 × 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)
```

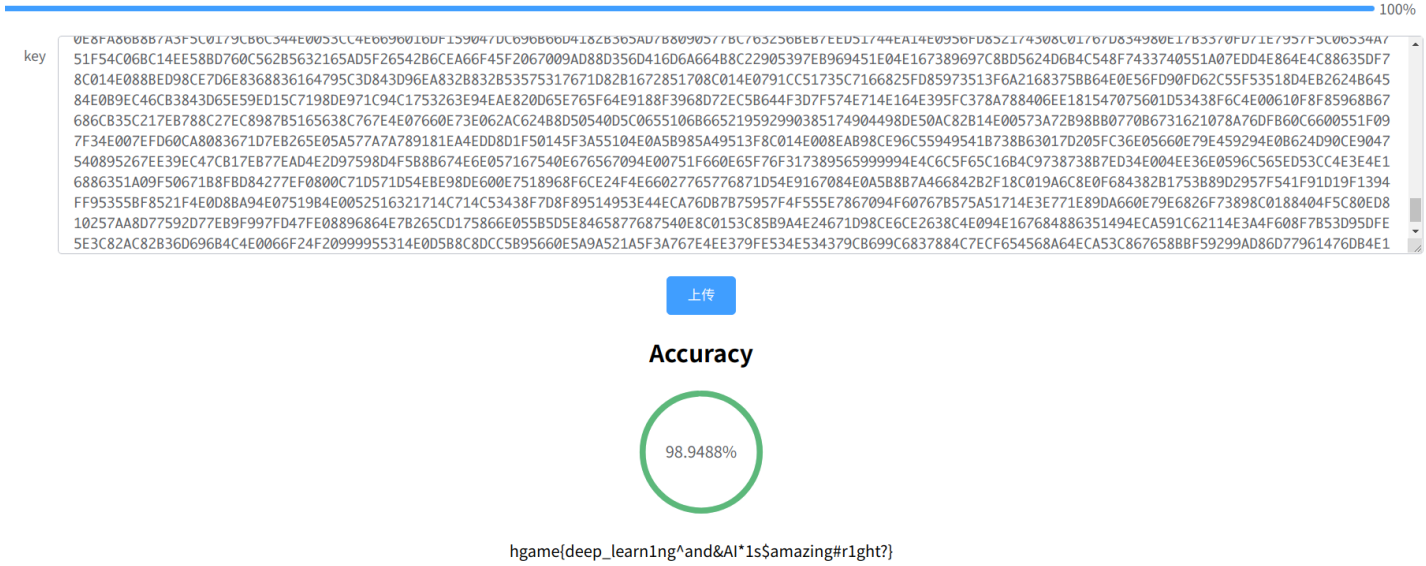
训练

```
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)
```



传入网站，得到flag

```
hgame{deep_learn1ng^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flag

```flag
hgame{deep_learn1ng^and&AI*1s$amazing#r1ght?}
```
```