

NYCU Pattern Recognition, Final Project

0810749, 張君實

Environment details

The version of common environment.

Python version==3.10.11

torch==1.13.1

torchaudio==0.13.1

torchvision==0.14.1

numpy==1.23.1

pandas==1.4.3

Pillow==9.2.0

The detailed of the environment is writing in requirements.txt.

By the way, if you want to support the GPU testing, please type

```
pip install torch==1.13.1+cu117 torchvision==0.14.1+cu117 torchaudio==0.13.1 --extra-index-url  
https://download.pytorch.org/whl/cu117
```

in the terminal/cmd.

The detailed of the troublesome issue can be referenced from the following article.

<https://github.com/AUTOMATIC111/stable-diffusion-webui/issues/7166>

Implementation details

Data preprocessing:

I had attempted to use the technique of data augmentation in this lab, including transforms.RandomResizedCrop(size = (224, 224),scale=(0.98, 1.0)) and transforms.RandomRotation(1).

However, it still reduces the accuracy (about 5%) of the model although I use the slight data augmentations. I guess that these symbols in the datasets are too strange to identify, especially rotating and clipping.

Thus, I ultimately only resize the images to 224*224 in order to meet the input size of the models.

Model architecture:

According to the different tasks, I independently trained different models in the different tasks.

I use resnet50 in this lab. The input of the model is 224*224, and the output of the model is the length of alphabet set multiplied by the word count.

```
class ResNet(nn.Module):
    def __init__(self, Layer=18, word_count=1, Pretrained=True):
        super(ResNet, self).__init__()
        self.word_count = word_count

        if Layer==18:
            self.classify = nn.Linear(512, alphabets_length * word_count)
        if Layer==50:
            self.classify = nn.Linear(2048, alphabets_length * word_count)

        pretrained_model = torchvision.models.__dict__['resnet{}'.format(Layer)](pretrained=Pretrained)
        self.conv1 = pretrained_model._modules['conv1']
        self.bn1 = pretrained_model._modules['bn1']
        self.relu = pretrained_model._modules['relu']
        self.maxpool = pretrained_model._modules['maxpool']

        self.layer1 = pretrained_model._modules['layer1']
        self.layer2 = pretrained_model._modules['layer2']
        self.layer3 = pretrained_model._modules['layer3']
        self.layer4 = pretrained_model._modules['layer4']

        self.avgpool = nn.AdaptiveAvgPool2d(1)

        del pretrained_model
```

```
def forward(self, x):
    x = self.conv1(x)
    x = self.bn1(x)
    x = self.relu(x)
    x = self.maxpool(x)

    x = self.layer1(x)
    x = self.layer2(x)
    x = self.layer3(x)
    x = self.layer4(x)

    x = self.avgpool(x)
    # print(x.shape)
    x = torch.flatten(x, start_dim=1)

    x = self.classify(x)

    return x
```

As for the loss function, in task1, I use nn.CrossEntropyLoss(). And I use nn.MultiLabelSoftMarginLoss() in task2 and task3, which are used in multi-labels tasks.

Bagging:

In every task, I train 3 models of Resnet50 to vote the ultimate result. Because there are 3 tasks in this lab, I totally train 3*3=9 models in this lab.



submission.csv
Complete · 5m ago · model_all3

0.966



submission.csv

Complete · 10h ago · task_model_all2

0.9628



submission.csv

Complete · 1d ago · task_model_all

0.966

As for the effect of the bagging technique, if I don't use the bagging, the accuracy of the prediction is respectively 0.9628, 0.966, and 0.966.

After using bagging techniques, the accuracy can reach 0.969. Maybe the technique is somewhat helpful.



submission2.csv

Complete · 4m ago · model_all1+model_all2+model_all3

0.969

Hyperparameters:

Epoch = 300

Learning Rate = $7e-4$

Batch Size = 32

Another detailed of training process:

The dataset what I use for training is all the images in the training dataset, in other words, I don't have dataset to validate whether the model is robust. Thus, I straightly save the model that the loss is minimum in the training process.

Additionally, I am worried that the Resnet50 cannot successfully converge, so I use the pretrained model afforded by torch libraries, and I fine-tuned it. At the beginning of the 10 epochs, I only train the last 2 layers.

Kaggle Submission

YOUR RECENT SUBMISSION

✓

submission2.csv

Submitted by ChunShihChang · Submitted 9 minutes ago

Score: 0.96900
Private score:

Jump to your leaderboard position

Search leaderboard

PublicPrivate

This leaderboard is calculated with approximately 50% of the test data. The final results will be based on the other 50%, so the final standings may be different.

#	Team	Members	Score	Entries	Last
1	310551171		0.98000	7	3d
2	311551089		0.97220	14	3d
3	0810749		0.96900	9	9m
<div><div></div><div>Your Best Entry! Your most recent submission scored 0.96900, which is an improvement of your previous score of 0.96600. Great job!</div><div>Tweet this</div></div>					
4	311551070		0.96680	16	5d