
猫狗识别小软件说明文档

JJ Yao

1 软件介绍

该软件的功能是识别输入的猫或者狗的图片，使用的是迁移学习，将 Resnet50 网络的最后一层即 $(fc) : Linear(in_features = 2048, out_features = 1000, bias = True)$ 改为了 $(fc) : Linear(in_features = 2048, out_features = 2, bias = True)$ ，并经过训练得到网络参数。数据集来自 [kaggle](#)，该数据集包含 12500 张猫图片和 12500 张狗图片组成的训练集以及 12500 为分类的图片作为测试集。

本软件在网络参数训练中将训练集以 1 : 4 划分为验证集和训练集。每一批次 16 个数据，每一轮是 1250 批次，一共训练了 5 轮。最后的结果为图1所示：

```
epoch 4/4
-----
training...
batch500,train loss:2.1734,train acc:96.1250
batch1000,train loss:2.1765,train acc:95.9813
train loss:2.1505 acc:96.0600%
validing...
valid loss:1.4812 acc:98.1200%
6862.861127138138
```

图 1: 训练结果

之后将已训练好的参数保存以供后序加载。

2 软件操作说明

打开软件会先有一个登录界面，如图2所示：该界面有登录和注册按钮，如果

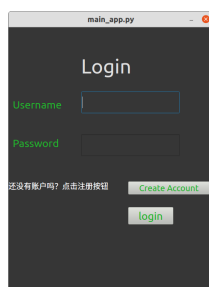


图 2: 登录界面

有账号，只需输入点击登录即可进入主界面，否则请点击注册按钮。点击注册按钮之后会打开一个注册界面，如图3所示：

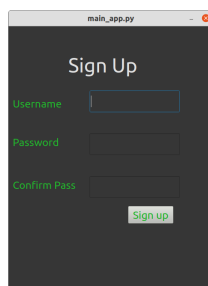


图 3: 注册界面

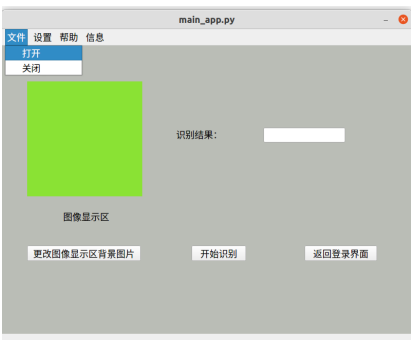
注册完之后在登录界面输入用户名和密码之后就可以、进入到主界面。主界面如图4所示：



图 4: 主界面

在主界面可以看到最上面一排是菜单栏，紧接着往下是图像显示区以及识别结果的显示框，最下面一行分别为更改图像显示区背景图片，开始识别按钮，返回登录按钮。

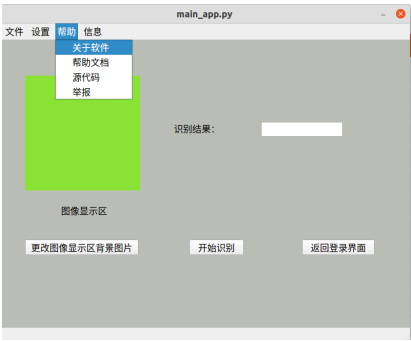
软件的菜单栏分别如图5a、图5b、图5c、图5d所示，可以看到界面与之前相比发生了变化，正是因为修改了一些设置所导致的。当然还有其它选项，比如源代码、关于软件等等。



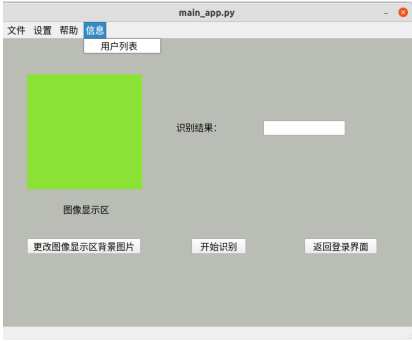
(a) 文件



(b) 设置



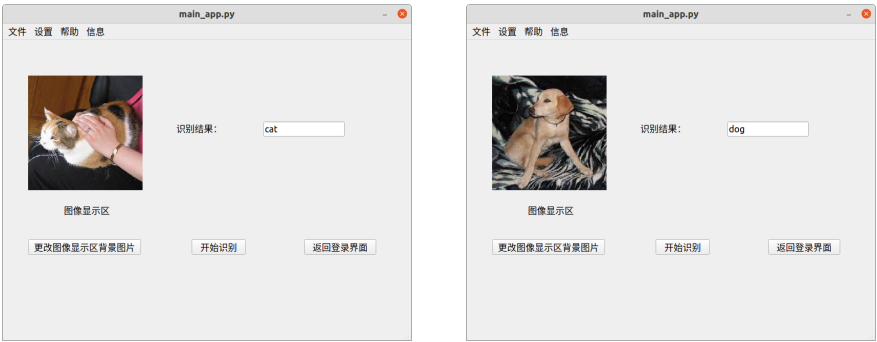
(c) 帮助



(d) 信息

图 5: 软件菜单栏图例

下面放两张识别案例，如图6a、图6b所示：



(a) 猫图片 (b) 狗图片

图 6: 软件实例

3 卷积神经网络模型

由于网络层数太多，无法画出，故给了一张卷积神经网络例图，如图7，并将网络结构于文尾给出。

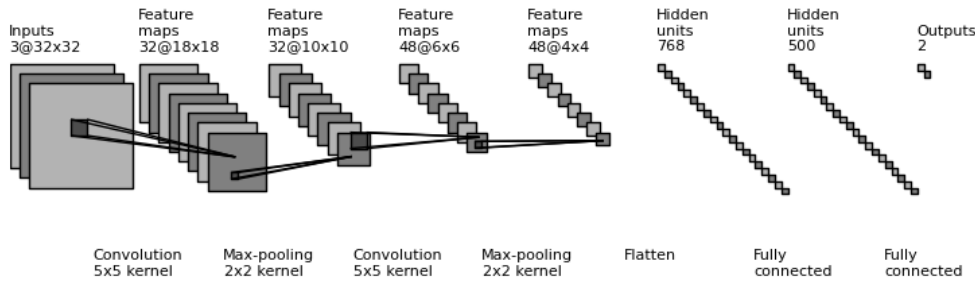


图 7: 卷积神经网络模型

网络结构如下所示，网络结构依次是图8a、图8b、图8c、图8d、图9。

```

1 ResNet
2 (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
3 (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
4 (relu): ReLU(inplace=True)
5 (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
6 (layer1): Sequential(
7   (0): Bottleneck(
8     (conv1): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
9     (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
10    (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
11    (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
12    (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
13    (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
14    (relu): ReLU(inplace=True)
15    (downsample): Sequential(
16      (0): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
17      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
18    )
19  )
20 (layer2): Sequential(
21   (0): Bottleneck(
22     (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
23     (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
24     (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
25     (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
26     (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
27     (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
28     (relu): ReLU(inplace=True)
29   )
30   (1): Bottleneck(
31     (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
32     (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
33     (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
34     (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
35     (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
36     (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
37     (relu): ReLU(inplace=True)
38   )
39 (layer3): Sequential(
40   (0): Bottleneck(
41     (conv1): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
42     (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
43     (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
44     (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
45     (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
46     (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
47     (relu): ReLU(inplace=True)
48     (downsample): Sequential(
49       (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
50       (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
51     )
52   )
53   (1): Bottleneck(
54     (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
55     (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
56     (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
57     (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
58     (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
59     (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
60     (relu): ReLU(inplace=True)
61   )
62   (2): Bottleneck(
63     (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
64     (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
65     (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
66     (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
67     (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
68     (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
69     (relu): ReLU(inplace=True)
70   )
71   (3): Bottleneck(
72     (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
73     (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
74     (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
75     (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
76     (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
77     (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
78     (relu): ReLU(inplace=True)
79   )
80 )
81 (layer4): Sequential(
82   (0): Bottleneck(
83     (conv1): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
84     (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
85     (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
86     (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
87     (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
88     (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
89     (relu): ReLU(inplace=True)
90     (downsample): Sequential(
91       (0): Conv2d(512, 1024, kernel_size=(1, 1), stride=(2, 2), bias=False)
92       (1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
93     )
94   )
95   (1): Bottleneck(
96     (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
97     (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
98     (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
99     (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
100    (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
101    (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
102    (relu): ReLU(inplace=True)
103  )
104   (2): Bottleneck(
105     (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
106     (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
107     (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
108     (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
109     (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
110     (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
111     (relu): ReLU(inplace=True)
112  )
113   (3): Bottleneck(
114     (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
115     (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
116     (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
117     (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
118     (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
119     (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
120     (relu): ReLU(inplace=True)
121  )
122 )
123 (4): Bottleneck(
124   (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
125   (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
126   (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
127   (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
128   (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
129   (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
130   (relu): ReLU(inplace=True)
131 )
132 (5): Bottleneck(
133   (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
134   (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
135   (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
136   (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
137   (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
138   (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
139   (relu): ReLU(inplace=True)
140 )
141 (layer5): Sequential(
142   (0): Bottleneck(
143     (conv1): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
144     (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
145     (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
146     (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
147     (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
148     (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
149     (relu): ReLU(inplace=True)
150     (downsample): Sequential(
151       (0): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(2, 2), bias=False)
152       (1): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
153     )
154   )
155   (1): Bottleneck(
156     (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
157     (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
158     (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
159     (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
160     (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)

```

(a) one

(b) two

(c) three

(d) four

图 8: 网络架构 1

```
161         (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
162         (relu): ReLU(inplace=True)
163     )
164     (2): Bottleneck(
165       (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
166       (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
167       (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
168       (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
169       (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
170       (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
171       (relu): ReLU(inplace=True)
172     )
173   )
174   (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
175   (fc): Linear(in_features=2048, out_features=2, bias=True)
176 )
```

图 9: 网络架构 2