技术路线

数据标注

↓

ResNet152

↓

Linear(2048, 512)

↓

Linear(512, 256)

↓Dropout

Linear(256, 1)

**导入包**

In [1]

# 导入包

import sys

sys.path.append('/home/aistudio/external-libraries')

import os

import cv2

import random

import numpy as np

import pandas as pd

from PIL import Image

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics.pairwise import euclidean\_distances

import matplotlib.pylab as plt

import paddle.nn.functional as F

import paddle

import paddle.nn as nn

import paddle.vision.models

from paddle.io import Dataset

import matplotlib.pyplot as plt

**定义网络与数据读取器**

In [2]

class Network(paddle.nn.Layer):

def \_\_init\_\_(self):

super(Network, self).\_\_init\_\_()

self.resnet = paddle.vision.models.resnet152(pretrained=True, num\_classes=0) # remove final fc 输出为[?, 2048, 1, 1]

self.flatten = paddle.nn.Flatten()

self.linear\_1 = paddle.nn.Linear(2048, 512)

self.linear\_2 = paddle.nn.Linear(512, 256)

self.linear\_3 = paddle.nn.Linear(256, 1)

self.relu = paddle.nn.ReLU()

self.dropout = paddle.nn.Dropout(0.2)

def forward(self, inputs):

# print('input', inputs)

y = self.resnet(inputs)

y = self.flatten(y)

y = self.linear\_1(y)

y = self.linear\_2(y)

y = self.relu(y)

y = self.dropout(y)

y = self.linear\_3(y)

# y = paddle.nn.functional.sigmoid(y)

return y

# 定义数据读取器

def load\_data(mylist,mode='train',image\_size=256,batchsize=32):

# 获得数据集长度

data\_length = len(mylist)

# 定义数据集每个数据的序号，根据序号读取数据

index\_list = list(range(data\_length))

# 定义数据生成器

def data\_generator():

if mode == 'train':

# 训练模式下打乱数据

random.shuffle(index\_list)

imgs\_list = []

labels\_list = []

for i in index\_list:

# 将数据处理成希望的类型，即处理成image\_size\*image\_size的向量

im = Image.open(mylist[i][0])

im = im.convert('RGB')

im = im.resize((image\_size, image\_size), Image.ANTIALIAS)

im = np.array(im).astype('float32')/255

im = im.transpose(2, 0, 1)

imgs\_list.append(im)

label = mylist[i][1]

labels\_list.append(label)

if len(imgs\_list) == batchsize:

# 获得一个batchsize的数据，并返回

yield np.array(imgs\_list), np.array(labels\_list).astype('float32')

# 清空数据读取列表

imgs\_list = []

labels\_list = []

# 如果剩余数据的数目小于BATCHSIZE，

# 则剩余数据一起构成一个大小为len(imgs\_list)的mini-batch

if len(imgs\_list) > 0:

yield np.array(imgs\_list), np.array(labels\_list).astype('float32')

return data\_generator

# 简易的预测器，使用一个list传入图片地址即可获得对应的list的卡路里，或者传入一张图片的地址可以获得对应的卡路里

def predict\_imgs(model,predict\_list,image\_size=256):

model.eval()

result=[]

if predict\_list is not list:

predict\_list=[predict\_list]

for item in predict\_list:

im = Image.open(item)

im = im.resize((image\_size, image\_size), Image.ANTIALIAS)

im = np.array(im).astype('float32')/255

im = im.transpose(2, 0, 1)

images = paddle.to\_tensor(im).reshape((1,3,image\_size,image\_size))

predicts = model(images)

result.append(predicts.tolist()[0][0])

return result

**数据准备**

任选数据源读取即可

In [55]

# 读取数据源1（小样本数据源）

! unzip -oq /home/aistudio/Calorie.zip -d pic

cal\_df=pd.read\_csv('Calorie.csv')

mylist=[];

for item in os.listdir('pic'):

index=int(item.split('.')[0])

mylist.append(['pic/'+item,cal\_df[cal\_df.Index==index].Calorie.values.tolist()[0]]);

random.shuffle(mylist)

train\_list=mylist[:40]

# val\_list=mylist[80:100]

test\_list=mylist[40:]

In [9]

# 读取数据源2

#! unzip -oq /home/aistudio/data/data121013/Caloriedata.zip -d /home/aistudio/data

#cal\_df=pd.read\_csv('data/Calorie.csv')

#mylist=[];

#for item in os.listdir('data/images'):

# index=int(item.split('.')[0])

# mylist.append(['data/images/'+item,cal\_df[cal\_df.Index==index].Calorie.values.tolist()[0]]);

#random.shuffle(mylist)

#train\_list=mylist[:280]

#val\_list=mylist[280:360]

#test\_list=mylist[360:]

**训练模型**

想要带有验证集进行训练将with\_eval设置为1即可

In [56]

# 构建模型

epoch\_num=100;

with\_eval=0;

model=Network()

# 接续之前的训练结果进行训练

# param\_dict = paddle.load('./Mymodel')

# model.load\_dict(param\_dict)

model.train()

# 构建用于分批读取数据训练的数据读取器

train\_loader = load\_data(train\_list,mode='train',batchsize=40)

if with\_eval:

val\_loader = load\_data(val\_list,mode='train',batchsize=40)

opt = paddle.optimizer.Adam(learning\_rate=0.001, weight\_decay=paddle.regularizer.L2Decay(coeff=1e-5),parameters=model.parameters())

print('train start')

best\_score=99999999;

# 正式训练

for epoch\_id in range(epoch\_num):

for batch\_id, data in enumerate(train\_loader()):

images, labels = data

images = paddle.to\_tensor(images)

labels = paddle.to\_tensor(labels)

labels = labels.reshape(labels.shape+[1])

predicts = model(images)

loss = F.square\_error\_cost(predicts, labels)

loss = loss/labels

avg\_loss = paddle.mean(loss)

#后向传播，更新参数的过程

avg\_loss.backward()

opt.step()

opt.clear\_grad()

if (epoch\_id+1)%10==0 and batch\_id==0:

print("{} epoch {} batch: los is {}".format(epoch\_id,batch\_id,avg\_loss.numpy()))

if with\_eval:

model.eval()

val\_loss=0;

for batch\_id, data in enumerate(val\_loader()):

images, labels = data

images = paddle.to\_tensor(images)

labels = paddle.to\_tensor(labels)

labels = labels.reshape(labels.shape+[1])

predicts = model(images)

loss = F.square\_error\_cost(predicts, labels)

loss = loss/labels

val\_loss+=loss.mean().tolist()[0]\*len(labels)

val\_loss/=len(val\_list)

if val\_loss<best\_score:

paddle.save(model.state\_dict(), 'best\_model')

model.train()

print("{} epoch: val los is {}".format(epoch\_id,val\_loss))

print('train finish')

paddle.save(model.state\_dict(), 'Mymodel')

train start

/opt/conda/envs/python35-paddle120-env/lib/python3.7/site-packages/paddle/nn/layer/norm.py:653: UserWarning: When training, we now always track global mean and variance.

"When training, we now always track global mean and variance.")

9 epoch 0 batch: los is [552.33826]

19 epoch 0 batch: los is [58.95808]

29 epoch 0 batch: los is [9.774215]

39 epoch 0 batch: los is [8.686606]

49 epoch 0 batch: los is [4.776394]

59 epoch 0 batch: los is [4.245353]

69 epoch 0 batch: los is [7.529343]

79 epoch 0 batch: los is [5.3726387]

89 epoch 0 batch: los is [5.009757]

99 epoch 0 batch: los is [5.2784705]

train finish

**查看验证集结果**

In [57]

# 构建模型

model=Network()

param\_dict = paddle.load('./Mymodel')

model.load\_dict(param\_dict)

model.eval()

# 构建用于分批读取数据训练的数据读取器

test\_loader = load\_data(test\_list,mode='test')

print('test start')

predicts\_list=[];

for batch\_id, data in enumerate(test\_loader()):

images, labels = data

images = paddle.to\_tensor(images)

labels = paddle.to\_tensor(labels)

labels = labels.reshape(labels.shape+[1])

predicts = model(images)

predicts\_list+=predicts.tolist()

print('test finish')

predicts\_compare\_list=[]

for i in range(len(test\_list)):

predicts\_compare\_list.append(test\_list[i]+predicts\_list[i])

print('图片名称 | 卡路里标签 | 预测值')

predicts\_compare\_list

test start

test finish

图片名称 | 卡路里标签 | 预测值

[['pic/45.jpg', 800, 974.5698852539062],

['pic/14.jpg', 100, 927.0052490234375],

['pic/35.jpg', 800, 1228.0440673828125],

['pic/24.jpg', 500, 800.8392333984375],

['pic/7.jpg', 1200, 858.04736328125]]

**应用示例**

In [58]

# 预测

# 传入图片地址或者以list传入一组图片

target\_imgdir="mylunch.jpg"

model=Network()

param\_dict = paddle.load('./Mymodel')

model.load\_dict(param\_dict)

result=predict\_imgs(model,target\_imgdir)

print(result)

# print(predict\_imgs(model,'test.jpg'))

plt.imshow(Image.open(target\_imgdir))

plt.title('Calorie:'+str(int(result[0])),color='red')

In [6]

# 预测

target\_imgdir="ShanghaiCodingParty.jpg"

model=Network()

param\_dict = paddle.load('./Mymodel')

model.load\_dict(param\_dict)

result=predict\_imgs(model,target\_imgdir)

print(result)

# print(predict\_imgs(model,'test.jpg'))

plt.imshow(Image.open(target\_imgdir))

plt.title('Calorie:'+str(int(result[0])),color='red')