Python Challenging项目实验报告

姓名：卢麒萱 学号：2010519

项目题目

- 定义：给定一个信息的标题、出处、相关链接以及相关评论，尝试别信息真伪。
- 输入：信息来源、标题、相关超链接、评论
- 输出：真伪标签（0: 消息为真，1: 消息为假）

数据分析

1. 数据获取

- https://github.com/yaqingwang/WeFEND-AAAI20
- 随本文件同URL提供
- 只使用有标签数据

2. 数据读取

- 文件格式为csv格式
- 可以使用Python自带的文件读取方式，手动分列
- 可以使用Pandas库进行csv文件读取
- 文件读取代码可以参考上文提及的git仓库中代码

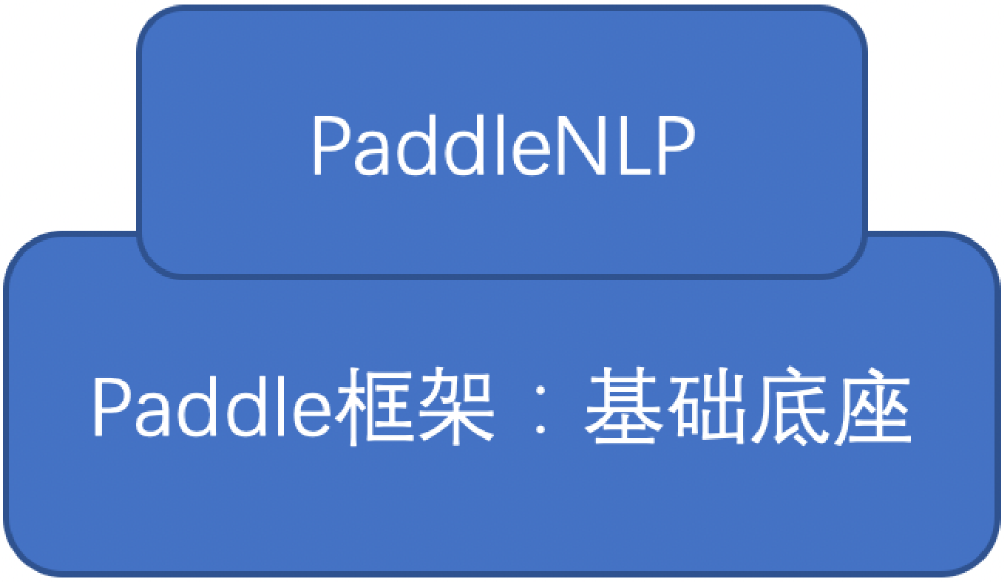
3. 参考读取代码

```
with open(filename, ‘r’, encoding=‘utf’) as f:
Import pandas as pd; dataset = pd.read_csv(filename)
```

深度学习方法

- PaddlePaddle框架+PaddleNLP

  PaddleNLP和PaddlePaddle框架是什么关系?



Paddle框架是基础底座，提供深度学习任务全流程API。PaddleNLP基于Paddle框架开发，适用于NLP任务

- 使用飞桨完成深度学习任务的通用流程
    - 数据集和数据处理
      paddle.io.Dataset
      paddle.io.DataLoader
      paddlenlp.data
    - 组网和网络配置
      paddle.nn.Embedding
      paddlenlp.seq2vec paddle.nn.Linear
      paddle.tanh paddle.nn.CrossEntropyLoss
      paddle.metric.Accuracy
      paddle.optimizer
      model.prepare
    - 网络训练和评估
      model.fit
      model.evaluate
    - 预测
      model.predict
- 引入相关库

```
import os
import random
import csv
import numpy as np
from functools import partial
import paddle
import paddle.nn as nn
import paddle.nn.functional as F
import paddlenlp as ppnlp
from paddlenlp.data import Pad, Stack, Tuple
from utils import load_vocab, convert_example
```

- 训练数据、测试数据的文件路径

```
original_data = 'challenging/data/train.csv'
test_data = 'challenging/data/test.csv'
```

- 解析训练数据

```
with open(original_data, "r", encoding='utf-8') as f:
    text = csv.reader(f)
    for i in text:
        if i[5] != "1" and i[5] != "0":
            continue
        if i[5] == "1":
            all_rumor_list.append(i[0] + i[1] + "\t" + i[5] + "\n")
            rumor_num += 1
        else:
            all_non_rumor_list.append(i[0] + i[1] + "\t" + i[5] + "\n")
            non_rumor_num += 1

with open(test_data, "r", encoding='utf-8') as f:
    text = csv.reader(f)
    for i in text:
        if i[5] != "1" and i[5] != "0":
            continue
        test_list.append(i[0] + i[1] + "\t" + i[5] + "\n")
        test_num += 1

print("谣言数据总量为: " + str(rumor_num))
print("非谣言数据总量为: " + str(non_rumor_num))
```

- 打乱数据，写入文件

```
data_list_path = "data/"
all_data_path = data_list_path + "all_data.txt"
test_data_path = data_list_path + "test_data.txt"
all_data_list = all_rumor_list + all_non_rumor_list

random.shuffle(all_data_list)
random.shuffle(test_list)

#在生成all_data.txt之前，首先将其清空
with open(all_data_path, 'w') as f:
    f.seek(0)
    f.truncate()

with open(test_data_path, 'w') as f:
    f.seek(0)
    f.truncate()

with open(all_data_path, 'a') as f:
    for data in all_data_list:
        f.write(data)

with open(test_data_path, 'a') as f:
    for data in test_list:
        f.write(data)
```

- 划分数据集

  创建序列化表示的数据,并按照一定比例划分训练数据与验证数据

```python
with open(os.path.join(data_list_path, 'eval_list.txt'), 'w',
          encoding='utf-8') as f_eval:
    f_eval.seek(0)
    f_eval.truncate()

with open(os.path.join(data_list_path, 'train_list.txt'),
          'w',
          encoding='utf-8') as f_train:
    f_train.seek(0)
    f_train.truncate()

with open(os.path.join(data_list_path, 'all_data.txt'), 'r',
          encoding='utf-8') as f_data:
    lines = f_data.readlines()

with open(os.path.join(data_list_path, 'test_data.txt'), 'r',
          encoding='utf-8') as f_test_init:
    eval_lines = f_test_init.readlines()

i = 0
with open(os.path.join(data_list_path, 'train_list.txt'),
          'a',
          encoding='utf-8') as f_train:
    for line in lines:
        words = line.split('\t')[-1].replace('\n', '')
        label = line.split('\t')[0]
        labs = ""
        labs = label + '\t' + words + '\n'
        f_train.write(labs)
        i += 1

j = 0
with open(os.path.join(data_list_path, 'eval_list.txt'), 'a',
          encoding='utf-8') as f_test:
    for line in eval_lines:
        words = line.split('\t')[-1].replace('\n', '')
        label = line.split('\t')[0]
        labs = ""
        labs = label + '\t' + words + '\n'
        f_test.write(labs)
        j += 1

print("数据列表生成完成！")

```

- 数据集和数据处理

  自定义数据集

  映射式(map-style)数据集需要继承`paddle.io.Dataset`

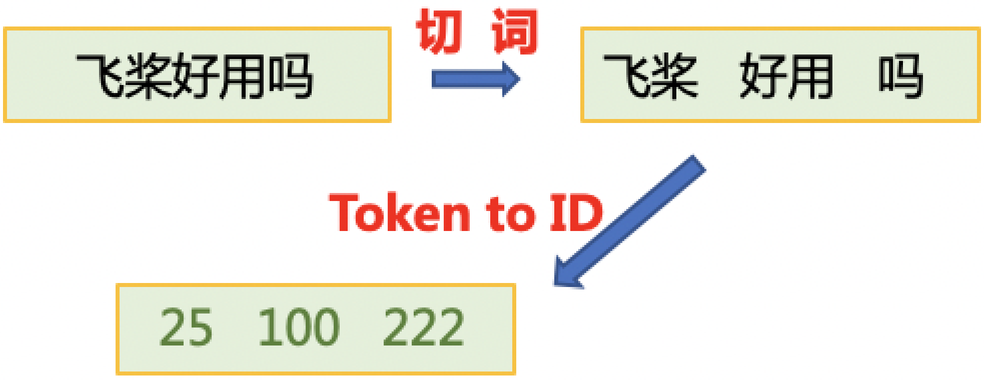
  - `__getitem__`: 根据给定索引获取数据集中指定样本，在 paddle.io.DataLoader 中需要使用此函数通过下标获取样本。
  - `__len__`: 返回数据集样本个数， paddle.io.BatchSampler 中需要样本个数生成下标序列。

```python
class SelfDefinedDataset(paddle.io.Dataset):
    def __init__(self, data):
        super(SelfDefinedDataset, self).__init__()
        self.data = data

    def __getitem__(self, idx):
        return self.data[idx]

    def __len__(self):
        return len(self.data)

    def get_labels(self):
        return ["0", "1"]


def txt_to_list(file_name):
    res_list = []
    for line in open(file_name):
        res_list.append(line.strip().split('\t'))
    return res_list


trainlst = txt_to_list('data/train_list.txt')
devlst = txt_to_list('data/eval_list.txt')
```

看看数据长什么样

```python
label_list = train_ds.get_labels()
print(label_list)

for i in range(10):
    print(train_ds[i])
```

- 数据处理
- 为了将原始数据处理成模型可以读入的格式，本项目将对数据作以下处理：
  - 首先使用jieba切词，之后将jieba切完后的单词映射词表中单词id。



  - 使用`paddle.io.DataLoader`接口多线程异步加载数据。

其中用到了PaddleNLP中关于数据处理的API。PaddleNLP提供了许多关于NLP任务中构建有效的数据pipeline的常用API

API	简介
`paddlenlp.data.Stack`	堆叠N个具有相同shape的输入数据来构建一个batch，它的输入必须具有相同的shape，输出便是这些输入的堆叠组成的batch数据。
`paddlenlp.data.Pad`	堆叠N个输入数据来构建一个batch，每个输入数据将会被padding到N个输入数据中最大的长度
`paddlenlp.data.Tuple`	将多个组batch的函数包装在一起

```python
import jieba

dict_path = 'data/dict.txt'

#创建数据字典，存放位置：dicts.txt。在生成之前先清空dict.txt
#在生成all_data.txt之前，首先将其清空
with open(dict_path, 'w') as f:
    f.seek(0)
    f.truncate()

dict_set = set()
train_data = open('data/train_list.txt')
for data in train_data:
    seg = jieba.lcut(data[:-3])
    for datas in seg:
        if not datas is " ":
            dict_set.add(datas)

dicts = open(dict_path, 'w')
dicts.write('[PAD]\n')
dicts.write('[UNK]\n')
for data in dict_set:
    dicts.write(data + '\n')
dicts.close()
vocab = load_vocab(dict_path)

for k, v in vocab.items():
    print(k, v)
    break
```

- 构造dataloder

  下面的`create_data_loader`函数用于创建运行和预测时所需要的`DataLoader`对象。

  - `paddle.io.DataLoader`返回一个迭代器，该迭代器根据`batch_sampler`指定的顺序迭代返回dataset数据。异步加载数据。
  - `batch_sampler`：DataLoader通过 batch_sampler 产生的mini-batch索引列表来 dataset 中索引样本并组成mini-batch
  - `collate_fn`：指定如何将样本列表组合为mini-batch数据。传给它参数需要是一个callable对象，需要实现对组建的batch的处理逻辑，并返回每个batch的数据。在这里传入的是`prepare_input`函数，对产生的数据进行pad操作，并返回实际长度等。

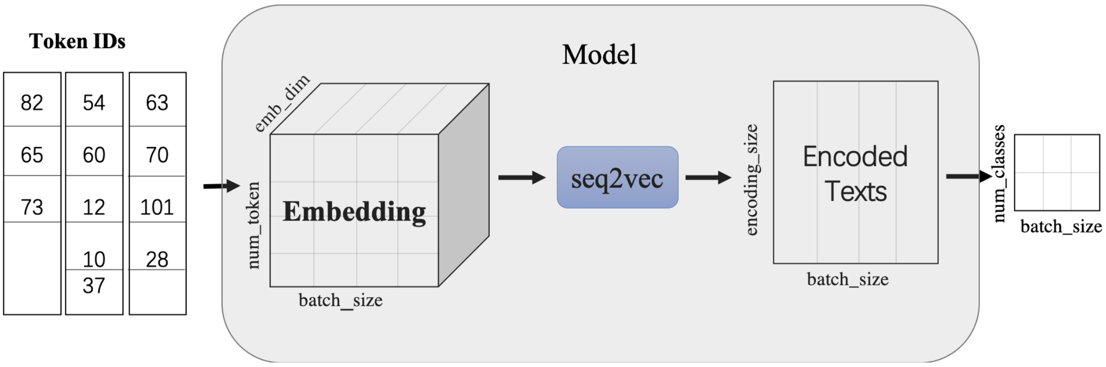
```python

# Reads data and generates mini-batches.
def create_dataloader(dataset,
                      trans_function=None,
                      mode='train',
                      batch_size=1,
                      pad_token_id=0,
                      batchify_fn=None):
    if trans_function:
        dataset = dataset.apply(trans_function, lazy=True)

    # return_list 数据是否以list形式返回
    # collate_fn  指定如何将样本列表组合为mini-batch数据。传给它参数需要是一个callable对象，需要实现对组建的batch的处理逻辑，并返回每个batch的数据。在这里传入的是`prepare_input`函数，对产生的数据进行pad操作，并返回实际长度等。
    dataloader = paddle.io.DataLoader(dataset,
                                      return_list=True,
                                      batch_size=batch_size,
                                      collate_fn=batchify_fn)

    return dataloader


# python中的偏函数partial，把一个函数的某些参数固定住（也就是设置默认值），返回一个新的函数，调用这个新函数会更简单。
trans_function = partial(convert_example,
                         vocab=vocab,
                         unk_token_id=vocab.get('[UNK]', 1),
                         is_test=False)

# 将读入的数据batch化处理，便于模型batch化运算。
# batch中的每个句子将会padding到这个batch中的文本最大长度batch_max_seq_len。
# 当文本长度大于batch_max_seq时，将会截断到batch_max_seq_len；当文本长度小于batch_max_seq时，将会padding补齐到batch_max_seq_len.
batchify_fn = lambda samples, fn=Tuple(
    Pad(axis=0, pad_val=vocab['[PAD]']),  # input_ids
    Stack(dtype="int64"),  # seq len
    Stack(dtype="int64")  # label
): [data for data in fn(samples)]

train_loader = create_dataloader(train_ds,
                                 trans_function=trans_function,
                                 batch_size=32,
                                 mode='train',
                                 batchify_fn=batchify_fn)
dev_loader = create_dataloader(dev_ds,
                               trans_function=trans_function,
                               batch_size=32,
                               mode='validation',
                               batchify_fn=batchify_fn)
```

- 模型搭建

  使用`LSTMencoder`搭建一个BiLSTM模型用于进行句子建模，得到句子的向量表示。

  然后接一个线性变换层，完成二分类任务。

  - `paddle.nn.Embedding`组建word-embedding层
  - `ppnlp.seq2vec.LSTMEncoder`组建句子建模层
  - `paddle.nn.Linear`构造二分类器



```python

class LSTMModel(nn.Layer):
    def __init__(self,
                 vocab_size,
                 num_classes,
                 emb_dim=128,
                 padding_idx=0,
                 lstm_hidden_size=198,
                 direction='forward',
                 lstm_layers=1,
                 dropout_rate=0,
                 pooling_type=None,
                 fc_hidden_size=96):
        super().__init__()

        # 首先将输入word id 查表后映射成 word embedding
        self.embedder = nn.Embedding(num_embeddings=vocab_size,
                                     embedding_dim=emb_dim,
                                     padding_idx=padding_idx)

        # 将word embedding经过LSTMEncoder变换到文本语义表征空间中
        self.lstm_encoder = ppnlp.seq2vec.LSTMEncoder(
            emb_dim,
            lstm_hidden_size,
            num_layers=lstm_layers,
            direction=direction,
            dropout=dropout_rate,
            pooling_type=pooling_type)

        # LSTMEncoder.get_output_dim()方法可以获取经过encoder之后的文本表示hidden_size
        self.fc = nn.Linear(self.lstm_encoder.get_output_dim(), fc_hidden_size)

        # 最后的分类器
        self.output_layer = nn.Linear(fc_hidden_size, num_classes)

    def forward(self, text, seq_len):
        # text shape: (batch_size, num_tokens)
        # print('input :', text.shape)

        # Shape: (batch_size, num_tokens, embedding_dim)
        embedded_text = self.embedder(text)
        # print('after word-embeding:', embedded_text.shape)

        # Shape: (batch_size, num_tokens, num_directions*lstm_hidden_size)
        # num_directions = 2 if direction is 'bidirectional' else 1
        text_repr = self.lstm_encoder(embedded_text, sequence_length=seq_len)
        # print('after lstm:', text_repr.shape)

        # Shape: (batch_size, fc_hidden_size)
        fc_out = paddle.tanh(self.fc(text_repr))
        # print('after Linear classifier:', fc_out.shape)

        # Shape: (batch_size, num_classes)
        logits = self.output_layer(fc_out)
        # print('output:', logits.shape)

        # probs 分类概率值
        probs = F.softmax(logits, axis=-1)
        # print('output probability:', probs.shape)
        return probs


model = LSTMModel(len(vocab),
                  len(label_list),
                  direction='bidirectional',
                  padding_idx=vocab['[PAD]'])
model = paddle.Model(model)
```

- 模型配置

```
optimizer = paddle.optimizer.Adam(parameters=model.parameters(),
                                  learning_rate=5e-5)

loss = paddle.nn.CrossEntropyLoss()
metric = paddle.metric.Accuracy()

model.prepare(optimizer, loss, metric)
# 设置visualdl路径
log_dir = './visualdl'
callback = paddle.callbacks.VisualDL(log_dir=log_dir)
```

- 模型训练

  训练过程中会输出loss、acc等信息。这里设置了10个epoch。

```
model.fit(train_loader,
          dev_loader,
          epochs=10,
          save_dir='./checkpoints',
          save_freq=5,
          callbacks=callback)
```

- 查看训练结果

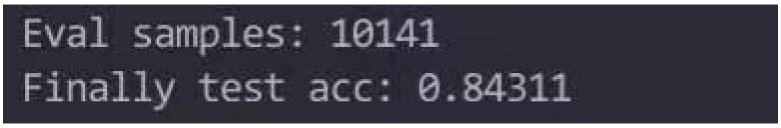
```
results = model.evaluate(dev_loader)
print("Finally test acc: %.5f" % results['acc'])
```

- 预测

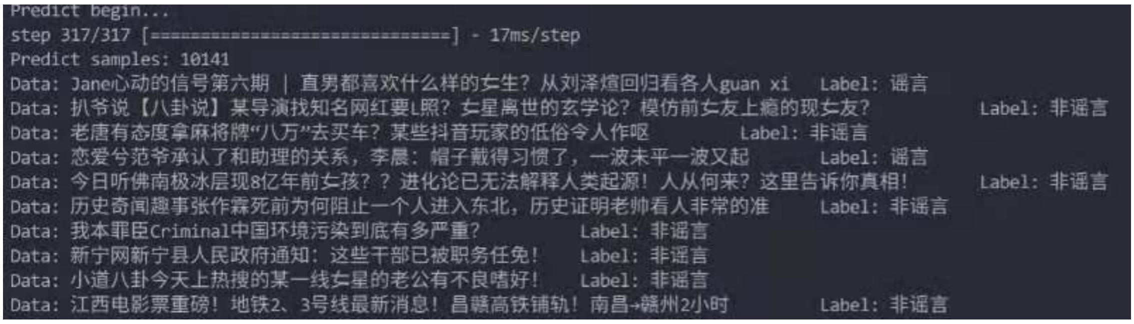
```python
label_map = {1: '谣言', 0: '非谣言'}
results = model.predict(dev_loader, batch_size=128)[0]
predictions = []

for batch_probs in results:
    # 映射分类label
    idx = np.argmax(batch_probs, axis=-1)
    idx = idx.tolist()
    labels = [label_map[i] for i in idx]
    predictions.extend(labels)

# 看看预测数据前5个样例分类结果
for idx, data in enumerate(dev_ds.data[:10]):
    print('Data: {} \t Label: {}'.format(data[0], predictions[idx]))
```

评价指标

- Accuracy

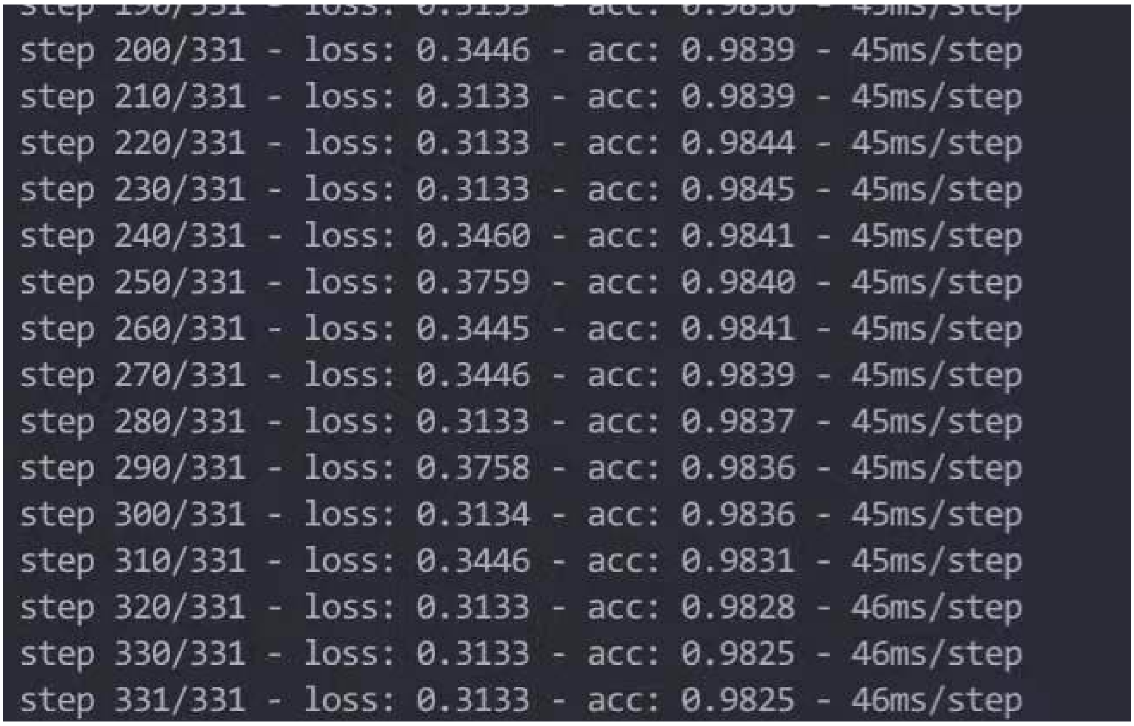


```
Eval samples: 10141
Finally test acc: 0.84311
```



```
Predict begin...
step 317/317 [==============================] - 17ms/step
Predict samples: 10141
Data: Jane心动的信号第六期 | 直男都喜欢什么样的女生? 从刘泽煊回归看各人guan xi    Label: 谣言
Data: 扒爷说【八卦说】某导演找知名网红要L照? 女星离世的玄学论? 模仿前女友上瘾的现女友?        Label: 非谣言
Data: 老唐有态度拿麻将牌“八万”去买车? 某些抖音玩家的低俗令人作呕         Label: 非谣言
Data: 恋爱兮范爷承认了和助理的关系, 李晨: 帽子戴得习惯了, 一波未平一波又起        Label: 谣言
Data: 今日听佛南极冰层现8亿年前女孩? ? 进化论已无法解释人类起源! 人从何来? 这里告诉你真相!       Label: 非谣言
Data: 历史奇闻趣事张作霖死前为何阻止一个人进入东北, 历史证明老帅看人非常的准      Label: 非谣言
Data: 我本罪臣Criminal中国环境污染到底有多严重?         Label: 非谣言
Data: 新宁网新宁县人民政府通知: 这些干部已被职务任免!     Label: 非谣言
Data: 小道八卦今天上热搜的某一线女星的老公有不良嗜好!     Label: 非谣言
Data: 江西电影票重磅! 地铁2、3号线最新消息! 昌赣高铁铺轨! 南昌→赣州2小时          Label: 非谣言
```

- Loss



```
step 190/331 - loss: 0.3133 - acc: 0.9836 - 45ms/step
step 200/331 - loss: 0.3446 - acc: 0.9839 - 45ms/step
step 210/331 - loss: 0.3133 - acc: 0.9839 - 45ms/step
step 220/331 - loss: 0.3133 - acc: 0.9844 - 45ms/step
step 230/331 - loss: 0.3133 - acc: 0.9845 - 45ms/step
step 240/331 - loss: 0.3460 - acc: 0.9841 - 45ms/step
step 250/331 - loss: 0.3759 - acc: 0.9840 - 45ms/step
step 260/331 - loss: 0.3445 - acc: 0.9841 - 45ms/step
step 270/331 - loss: 0.3446 - acc: 0.9839 - 45ms/step
step 280/331 - loss: 0.3133 - acc: 0.9837 - 45ms/step
step 290/331 - loss: 0.3758 - acc: 0.9836 - 45ms/step
step 300/331 - loss: 0.3134 - acc: 0.9836 - 45ms/step
step 310/331 - loss: 0.3446 - acc: 0.9831 - 45ms/step
step 320/331 - loss: 0.3133 - acc: 0.9828 - 46ms/step
step 330/331 - loss: 0.3133 - acc: 0.9825 - 46ms/step
step 331/331 - loss: 0.3133 - acc: 0.9825 - 46ms/step
```