

光学文字识别代码实现

1. 数据预处理

```
1  # 预处理数据，将其转化为标准格式。同时将数据拆分成两份，以便训练和计算预估准确率
2  import codecs
3  import os
4  import random
5  import shutil
6  from PIL import Image
7
8  train_ratio = 9 / 10 # 训练集大小
9  all_file_dir = "data/data6927/word-recognition" # 数据文件路径
10 image_path_pre = os.path.join(all_file_dir, "imageSet") # 路径
11
12 # 创建训练集路径
13 train_image_dir = os.path.join(all_file_dir, "trainImageSet")
14 if not os.path.exists(train_image_dir):
15     os.makedirs(train_image_dir)
16
17 eval_image_dir = os.path.join(all_file_dir, "evalImageSet")
18 if not os.path.exists(eval_image_dir):
19     os.makedirs(eval_image_dir)
20
21 train_file = codecs.open(os.path.join(all_file_dir, "train.txt"), 'w')
22 eval_file = codecs.open(os.path.join(all_file_dir, "eval.txt"), 'w')
23 label_list = os.path.join(all_file_dir, "image_label.txt") # 标签文件
24
25 train_count = 0
26 eval_count = 0
27 class_set = set()
28 with open(label_list) as f:
29     for line in f:
30         parts = line.strip().split()
31         file, label = parts[0], parts[1]
32         if '/' in label or '\\' in label or '.' in label or '!' in
label or '-' in label or '$' in label or '&' in label or '@' in label
or '?' in label or '%' in label or '(' in label or ')' in label or '~'
in label:
33             continue
34         for e in label:
35             class_set.add(e)
36         if random.uniform(0, 1) <= train_ratio:
```

```

37         shutil.copyfile(os.path.join(image_path_pre, file),
os.path.join(train_image_dir, file))
38         train_file.write("
{0}\t{1}\n".format(os.path.join(train_image_dir, file), label))
39         train_count += 1
40     else:
41         shutil.copyfile(os.path.join(image_path_pre, file),
os.path.join(eval_image_dir, file))
42         eval_file.write("
{0}\t{1}\n".format(os.path.join(eval_image_dir, file), label))
43         eval_count += 1
44
45 print("train image count: {0} eval image count:
{1}".format(train_count, eval_count))
46 class_list = list(class_set)
47 class_list.sort()
48 print("class num: {0}".format(len(class_list)))
49 print(class_list)
50 with codecs.open(os.path.join(all_file_dir, "label_list.txt"), "w") as
label_list:
51     label_id = 0
52     for c in class_list:
53         label_list.write("{0}\t{1}\n".format(c, label_id))
54         label_id += 1

```

2. 模型训练

```

1  # -*- coding: UTF-8 -*-
2  """
3  训练常基于crnn-ctc的网络，文字行识别
4  """
5  from __future__ import absolute_import
6  from __future__ import division
7  from __future__ import print_function
8  import os
9  import uuid
10 import numpy as np
11 import time
12 import six
13 import math
14 import random
15 import paddle
16 import paddle.fluid as fluid
17 import logging
18 import xml.etree.ElementTree
19 import codecs

```

```

20 import json
21
22 from paddle.fluid.initializer import MSRA
23 from paddle.fluid.param_attr import ParamAttr
24 from paddle.fluid.regularizer import L2Decay
25 from PIL import Image, ImageEnhance, ImageDraw
26
27 logger = None
28 train_params = {
29     "input_size": [1, 48, 512], # 输入数据维度
30     "data_dir": "data/data6927/word-recognition", # 数据集路径
31     "train_dir": "trainImageSet", # 训练数据目录
32     "eval_dir": "evalImageSet", # 评估数据目录
33     "train_list": "train.txt", # 训练集文件
34     "eval_list": "eval.txt", # 评估集文件
35     "label_list": "label_list.txt", # 标签文件
36     "class_dim": -1,
37     "label_dict": {}, # 标签字典
38     "image_count": -1,
39     "continue_train": False,
40     "pretrained": True, # 预训练
41     "pretrained_model_dir": "./pretrained-model", # 预训练模型目录
42     "save_model_dir": "./crnn-model", # 模型保存目录
43     "num_epochs": 400, # 训练轮次
44     "train_batch_size": 256, # 训练批次大小
45     "use_gpu": True, # 是否使用gpu
46     "ignore_thresh": 0.7, # 阈值
47     "mean_color": 127.5, #
48     "mode": "train", # 模式
49     "multi_data_reader_count": 4, # reader数量
50     "apply_distort": True, # 是否进行扭曲
51     "image_distort_strategy": { # 扭曲策略
52         "expand_prob": 0.5, # 放大比率
53         "expand_max_ratio": 2, # 最大放大比率
54         "hue_prob": 0.5, # 色调
55         "hue_delta": 18,
56         "contrast_prob": 0.5, # 对比度
57         "contrast_delta": 0.5,
58         "saturation_prob": 0.5, # 饱和度
59         "saturation_delta": 0.5,
60         "brightness_prob": 0.5, # 亮度
61         "brightness_delta": 0.125
62     },
63     "rsm_strategy": { # 梯度下降配置
64         "learning_rate": 0.0005,
65         "lr_epochs": [70, 120, 170, 220, 270, 320], # 学习率衰减分段（6

```

个数字分为7段）

```

66         "lr_decay": [1, 0.5, 0.1, 0.05, 0.01, 0.005, 0.001], # 每段采
用的学习率，对应lr_epochs参数7段
67     },
68     "early_stop": { # 控制训练停止条件
69         "sample_frequency": 50,
70         "successive_limit": 5,
71         "min_instance_error": 0.1
72     }
73 }
74
75
76 # CRNN网络模型
77 class CRNN(object):
78     def __init__(self, num_classes, label_dict):
79         self.outputs = None # 输出
80         self.label_dict = label_dict # 标签字典
81         self.num_classes = num_classes # 类别数量
82
83     def name(self):
84         return 'crnn'
85
86     def conv_bn_pool(self, input, group, out_ch, act="relu",
param=None,
87                     bias=None, param_0=None, is_test=False,
pooling=True, use_cudnn=False):
88         tmp = input
89
90         # six提供了简单的实用程序包来封装Python 2和Python 3之间的差异
91         # 它旨在支持无需修改即可在Python 2和Python 3上工作的代码库
92         # six只包含一个Python文件，因此无需复制到一个项目中
93         for i in six.moves.xrange(group):
94             # 卷积层
95             tmp = fluid.layers.conv2d(input=tmp, # 输入数据
num_filters=out_ch[i], # 卷积核
数量
96
97             filter_size=3, # 卷积核大小
98             padding=1, # 填充
99             param_attr=param if param_0 is
None else param_0,
100             act=None, # LinearActivation
101             use_cudnn=use_cudnn)
102             # batch normal
103             tmp = fluid.layers.batch_norm(input=tmp,
104             act=act,
105             param_attr=param,
106             bias_attr=bias,
107             is_test=is_test)

```

```

108         # 池化
109         if pooling:
110             tmp = fluid.layers.pool2d(input=tmp, # 输入数据
111                                       pool_size=2, # 池化区域大小
112                                       pool_type='max', # 池化类型
113                                       pool_stride=2, # 池化步长
114                                       use_cudnn=use_cudnn, # 是否使用
115                                       cudnn
116                                       ceil_mode=True)
117         return tmp
118
119     # OCR convs
120     def ocr_convs(self, input, regularizer=None, gradient_clip=None,
121                  is_test=False, use_cudnn=False):
122         b = fluid.ParamAttr(regularizer=regularizer,
123                             gradient_clip=gradient_clip,
124                             initializer=fluid.initializer.Normal(0.0,
125                             0.0)) # 初始化为正态分布
126         w0 = fluid.ParamAttr(regularizer=regularizer,
127                              gradient_clip=gradient_clip,
128                              initializer=fluid.initializer.Normal(0.0,
129                              0.0005)) # 初始化为正态分布
130         w1 = fluid.ParamAttr(regularizer=regularizer,
131                              gradient_clip=gradient_clip,
132                              initializer=fluid.initializer.Normal(0.0,
133                              0.01)) # 初始化为正态分布
134
135         tmp = input
136         # 第一组卷积池化
137         tmp = self.conv_bn_pool(tmp,
138                                 2, # 组数量
139                                 [16, 16], # 输出
140                                 param=w1,
141                                 bias=b,
142                                 param_0=w0,
143                                 is_test=is_test,
144                                 use_cudnn=use_cudnn)
145
146         # 第二组卷积池化
147         tmp = self.conv_bn_pool(tmp,
148                                 2, [32, 32],
149                                 param=w1,
150                                 bias=b,
151                                 is_test=is_test,
152                                 use_cudnn=use_cudnn)
153
154         # 第三组卷积池化
155         tmp = self.conv_bn_pool(tmp,

```

```

150         2, [64, 64],
151         param=w1,
152         bias=b,
153         is_test=is_test,
154         use_cudnn=use_cudnn)
155     # 第四组卷积池化
156     tmp = self.conv_bn_pool(tmp,
157         2, [128, 128],
158         param=w1,
159         bias=b,
160         is_test=is_test,
161         pooling=False,
162         use_cudnn=use_cudnn)
163     return tmp
164
165     # 组网函数
166     def net(self, images, rnn_hidden_size=200, regularizer=None,
167         gradient_clip=None, is_test=False, use_cudnn=True):
168         # 卷积池化
169         conv_features = self.ocr_convs(images,
170             regularizer=regularizer,
171             gradient_clip=gradient_clip,
172             is_test=is_test,
173             use_cudnn=use_cudnn)
174         # 转序列
175         sliced_feature = fluid.layers.im2sequence(input=conv_features,
176             stride=[1, 1],
177             filter_size=
[conv_features.shape[2], 1])
178
179         para_attr = fluid.ParamAttr(regularizer=regularizer,
180             gradient_clip=gradient_clip,
181
initializer=fluid.initializer.Normal(0.0, 0.02))
182         bias_attr = fluid.ParamAttr(regularizer=regularizer,
183             gradient_clip=gradient_clip,
184
initializer=fluid.initializer.Normal(0.0, 0.02))
185         bias_attr_nobias = fluid.ParamAttr(regularizer=regularizer,
186
gradient_clip=gradient_clip,
187
initializer=fluid.initializer.Normal(0.0, 0.02))
188         # 全连接层
189         fc_1 = fluid.layers.fc(input=sliced_feature,
190             size=rnn_hidden_size * 3,
191             param_attr=para_attr,

```

```

192         bias_attr=bias_attr_nobias)
193     fc_2 = fluid.layers.fc(input=sliced_feature,
194                             size=rnn_hidden_size * 3,
195                             param_attr=para_attr,
196                             bias_attr=bias_attr_nobias)
197     # gru(门控循环单元), LSTM变种
198     # 对检测到的字符连接成字符串序列
199     gru_forward = fluid.layers.dynamic_gru(input=fc_1,
200                                             size=rnn_hidden_size,
201                                             param_attr=para_attr,
202                                             bias_attr=bias_attr,
203
204     candidate_activation='relu')
205     gru_backward = fluid.layers.dynamic_gru(input=fc_2,
206                                             size=rnn_hidden_size,
207                                             is_reverse=True,
208                                             param_attr=para_attr,
209                                             bias_attr=bias_attr,
210
211     candidate_activation='relu')
212
213     w_attr = fluid.ParamAttr(regularizer=regularizer,
214                               gradient_clip=gradient_clip,
215
216     initializer=fluid.initializer.Normal(0.0, 0.02))
217     b_attr = fluid.ParamAttr(regularizer=regularizer,
218                               gradient_clip=gradient_clip,
219
220     initializer=fluid.initializer.Normal(0.0, 0.0))
221
222     fc_out = fluid.layers.fc(input=[gru_forward, gru_backward],
223                               size=self.num_classes + 1,
224                               param_attr=w_attr,
225                               bias_attr=b_attr)
226
227     self.outputs = fc_out
228     return fc_out
229
230     def get_infer(self):
231         return fluid.layers.ctc_greedy_decoder(input=self.outputs,
232         blank=self.num_classes)
233
234     def init_train_params():
235         """
236         初始化训练参数, 主要是初始化图片数量, 类别数
237         :return:
238         """

```

```
234     train_list = os.path.join(train_params['data_dir'],
train_params['train_list'])
235     label_list = os.path.join(train_params['data_dir'],
train_params['label_list'])
236
237     index = 0
238
239     with codecs.open(label_list, encoding='utf-8') as flist:
240         lines = [line.strip() for line in flist]
241         for line in lines:
242             parts = line.split()
243             train_params['label_dict'][parts[0]] = int(parts[1])
244             index += 1
245         train_params['class_dim'] = index
246
247     with codecs.open(train_list, encoding='utf-8') as flist:
248         lines = [line.strip() for line in flist]
249         train_params['image_count'] = len(lines)
250
251
252 # 初始化日志相关配置
253 def init_log_config():
254     global logger
255     logger = logging.getLogger()
256     logger.setLevel(logging.INFO)
257     log_path = os.path.join(os.getcwd(), 'logs')
258     if not os.path.exists(log_path):
259         os.makedirs(log_path)
260     log_name = os.path.join(log_path, 'train.log')
261     sh = logging.StreamHandler()
262     fh = logging.FileHandler(log_name, mode='w')
263     fh.setLevel(logging.DEBUG)
264     formatter = logging.Formatter("%(asctime)s - %(filename)s[line:%
(lineno)d] - %(levelname)s: %(message)s")
265     fh.setFormatter(formatter)
266     sh.setFormatter(formatter)
267     logger.addHandler(sh)
268     logger.addHandler(fh)
269
270
271 # 重设图像大小
272 def resize_img(img, input_size):
273     target_size = input_size
274     percent_h = float(target_size[1]) / img.size[1]
275     percent_w = float(target_size[2]) / img.size[0]
276     percent = min(percent_h, percent_w)
277     resized_width = int(round(img.size[0] * percent))
```



```

278     resized_height = int(round(img.size[1] * percent))
279     w_off = (target_size[2] - resized_width) / 2
280     h_off = (target_size[1] - resized_height) / 2
281     img = img.resize((resized_width, resized_height), Image.ANTIALIAS)
282     array = np.ndarray((target_size[1], target_size[2], 3), np.uint8)
283     array[:, :, 0] = 127
284     array[:, :, 1] = 127
285     array[:, :, 2] = 127
286     ret = Image.fromarray(array)
287     ret.paste(img, (np.random.randint(0, w_off + 1), int(h_off)))
288     return ret
289
290
291 # 调节亮度
292 def random_brightness(img):
293     prob = np.random.uniform(0, 1)
294     if prob < train_params['image_distort_strategy']
['brightness_prob']:
295         brightness_delta = train_params['image_distort_strategy']
['brightness_delta']
296         delta = np.random.uniform(-brightness_delta, brightness_delta)
+ 1
297         img = ImageEnhance.Brightness(img).enhance(delta)
298     return img
299
300
301 # 对比度
302 def random_contrast(img):
303     prob = np.random.uniform(0, 1)
304     if prob < train_params['image_distort_strategy']['contrast_prob']:
305         contrast_delta = train_params['image_distort_strategy']
['contrast_delta']
306         delta = np.random.uniform(-contrast_delta, contrast_delta) + 1
307         img = ImageEnhance.Contrast(img).enhance(delta)
308     return img
309
310
311 # 饱和度
312 def random_saturation(img):
313     prob = np.random.uniform(0, 1)
314     if prob < train_params['image_distort_strategy']
['saturation_prob']:
315         saturation_delta = train_params['image_distort_strategy']
['saturation_delta']
316         delta = np.random.uniform(-saturation_delta, saturation_delta)
+ 1
317         img = ImageEnhance.Color(img).enhance(delta)

```

```
318     return img
319
320
321 def random_hue(img):
322     prob = np.random.uniform(0, 1)
323     if prob < train_params['image_distort_strategy']['hue_prob']:
324         hue_delta = train_params['image_distort_strategy']
325         ['hue_delta']
326         delta = np.random.uniform(-hue_delta, hue_delta)
327         img_hsv = np.array(img.convert('HSV'))
328         img_hsv[:, :, 0] = img_hsv[:, :, 0] + delta
329         img = Image.fromarray(img_hsv, mode='HSV').convert('RGB')
330     return img
331
332 def distort_image(img):
333     prob = np.random.uniform(0, 1)
334     # Apply different distort order
335     if prob > 0.5:
336         img = random_brightness(img)
337         img = random_contrast(img)
338         img = random_saturation(img)
339         img = random_hue(img)
340     else:
341         img = random_brightness(img)
342         img = random_saturation(img)
343         img = random_hue(img)
344         img = random_contrast(img)
345     return img
346
347
348 def rotate_image(img):
349     """
350     图像增强，增加随机旋转角度
351     """
352     prob = np.random.uniform(0, 1)
353     if prob > 0.5:
354         angle = np.random.randint(-8, 8)
355         img = img.rotate(angle)
356     return img
357
358
359 def random_expand(img, keep_ratio=True):
360     if np.random.uniform(0, 1) <
361     train_params['image_distort_strategy']['expand_prob']:
362         return img
```

```

363     max_ratio = train_params['image_distort_strategy']
364     ['expand_max_ratio']
365     w, h = img.size
366     c = 3
367     ratio_x = random.uniform(1, max_ratio)
368     if keep_ratio:
369         ratio_y = ratio_x
370     else:
371         ratio_y = random.uniform(1, max_ratio)
372     oh = int(h * ratio_y)
373     ow = int(w * ratio_x)
374     off_x = random.randint(0, ow - w)
375     off_y = random.randint(0, oh - h)
376
377     out_img = np.zeros((oh, ow, c), np.uint8)
378     for i in range(c):
379         out_img[:, :, i] = train_params['mean_color']
380
381     out_img[off_y: off_y + h, off_x: off_x + w, :] = img
382
383     return Image.fromarray(out_img)
384
385 def preprocess(img, input_size):
386     img_width, img_height = img.size
387     if train_params['apply_distort']:
388         img = distort_image(img)
389         img = random_expand(img)
390         img = rotate_image(img)
391         # img = resize_img(img, input_size)
392         # img = img.convert('L')
393         # img = np.array(img).astype('float32') -
394         train_params['mean_color']
395         # img *= 0.007843
396     return img
397
398 # reader
399 def custom_reader(file_list, data_dir, input_size, mode):
400     def reader():
401         np.random.shuffle(file_list)
402         for line in file_list:
403             # img_name, label
404             parts = line.split()
405             image_path = parts[0]
406             img = Image.open(image_path)
407             # img = Image.open(os.path.join(data_dir, image_path))

```

```

408         if img.mode != 'RGB':
409             img = img.convert('RGB')
410         label = [int(train_params['label_dict'][c]) for c in
parts[-1]]
411         if len(label) == 0:
412             continue
413         if mode == 'train':
414             img = preprocess(img, input_size)
415             img = resize_img(img, input_size)
416             img = img.convert('L')
417             # img.save(image_path)
418             img = np.array(img).astype('float32') -
train_params['mean_color']
419             # img *= 0.007843
420             img = img[np.newaxis, ...]
421             # print("{0} {1}".format(image_path, label))
422             yield img, label
423
424     return reader
425
426
427 def multi_process_custom_reader(file_path, data_dir, num_workers,
input_size, mode):
428     """
429     创建多进程reader
430     :param file_path:
431     :param data_dir:
432     :param num_workers:
433     :param input_size:
434     :param mode:
435     :return:
436     """
437     file_path = os.path.join(data_dir, file_path)
438     readers = []
439     images = [line.strip() for line in open(file_path)]
440     n = int(math.ceil(len(images) // num_workers))
441     image_lists = [images[i: i + n] for i in range(0, len(images), n)]
442     train_path = os.path.join(train_params['data_dir'],
train_params['train_dir'])
443     for l in image_lists:
444         reader = paddle.batch(custom_reader(1, train_path, input_size,
mode),
445
446         batch_size=train_params['train_batch_size'])
447         readers.append(paddle.reader.shuffle(reader,
train_params['train_batch_size']))

```

```

448     return paddle.reader.multiprocess_reader(readers, False) # 返回多
    进程读取器
449
450
451 # 评估reader
452 def create_eval_reader(file_path, data_dir, input_size, mode):
453     file_path = os.path.join(data_dir, file_path)
454     images = [line.strip() for line in open(file_path)]
455     eval_path = os.path.join(train_params['data_dir'],
    train_params['eval_dir'])
456     return paddle.batch(custom_reader(images, eval_path, input_size,
    mode),
457                          batch_size=train_params['train_batch_size'])
458
459
460 def optimizer_rms_setting():
461     batch_size = train_params["train_batch_size"]
462     iters = train_params["image_count"] // batch_size # 计算总批次
463     learning_strategy = train_params['rsm_strategy']
464     lr = learning_strategy['learning_rate']
465
466     boundaries = [i * iters for i in learning_strategy["lr_epochs"]]
467     values = [i * lr for i in learning_strategy["lr_decay"]]
468
469     # 均方根传播（RMSProp）法
470     optimizer =
    fluid.optimizer.RMSProp(learning_rate=fluid.layers.piecewise_decay(bou
    ndaries, values),
471
    regularization=fluid.regularizer.L2Decay(0.00005))
472
473     return optimizer
474
475
476 def build_train_program_with_async_reader(main_prog, startup_prog):
477     """
478     定义异步读取器、预测、构建损失函数及优化器
479     :param main_prog:
480     :param startup_prog:
481     :return:
482     """
483     # 将main_prog, startup_prog设置为默认主program, startup_program
484     with fluid.program_guard(main_prog, startup_prog):
485         img = fluid.layers.data(name='img',
    shape=train_params['input_size'], dtype='float32')
486         gt_label = fluid.layers.data(name='gt_label', shape=[1],
    dtype='int32', lod_level=1)

```

```

487         # 创建reader
488         data_reader =
fluid.layers.create_py_reader_by_data(capacity=train_params['train_batch_size'],
489                                     feed_list=
[img, gt_label],
490
         name='train')
491         # 创建多进程reader
492         multi_reader =
multi_process_custom_reader(train_params['train_list'],
493
         train_params['data_dir'],
494
         train_params['multi_data_reader_count'],
495
         train_params['input_size'],
496                                     'train')
497         data_reader.decorate_paddle_reader(multi_reader)
498
499         with fluid.unique_name.guard(): # 更换namespace
500             img, gt_label = fluid.layers.read_file(data_reader)
501
502             model = CRNN(train_params['class_dim'],
train_params['label_dict']) # 实例化
503             fc_out = model.net(img) # 预测
504
505             cost = fluid.layers.warpctc(input=fc_out, label=gt_label,
blank=train_params['class_dim'],
506                                     norm_by_times=True) # 计算CTC
损失函数
507             loss = fluid.layers.reduce_sum(cost) # 损失函数求和
508             optimizer = optimizer_rms_setting()
509             optimizer.minimize(loss)
510             # 执行CTC去重
511             decoded_out =
fluid.layers.ctc_greedy_decoder(input=fc_out,
512
         blank=train_params['class_dim'])
513             casted_label = fluid.layers.cast(x=gt_label,
dtype='int64')
514             # 计算字符串的编辑距离
515             # 编辑距离又称Levenshtein距离，由俄罗斯的数学家Vladimir
Levenshtein在1965年提出
516             # 是指利用字符操作，把字符串A转换成字符串B所需要的最少操作数
517             # 例如: "kitten" -> "sitten" -> "sittin" -> "sitting"

```

```

518         distances, seq_num =
fluid.layers.edit_distance(decoded_out, casted_label)
519
520         return data_reader, loss, distances, seq_num, decoded_out
521
522
523 def build_eval_program_with_feeder(main_prog, startup_prog, place):
524     """
525     执行评估
526     :param main_prog:
527     :param startup_prog:
528     :param place:
529     :return:
530     """
531     with fluid.program_guard(main_prog, startup_prog):
532         img = fluid.layers.data(name='img',
shape=train_params['input_size'], dtype='float32')
533         gt_label = fluid.layers.data(name='gt_label', shape=[1],
dtype='int32', lod_level=1)
534         feeder = fluid.DataFeeder(feed_list=[img, gt_label],
place=place, program=main_prog)
535         reader = create_eval_reader(train_params['eval_list'],
train_params['data_dir'],
536                                     train_params['input_size'],
537                                     'eval')
538
539         with fluid.unique_name.guard():
540             model = CRNN(train_params['class_dim'],
train_params['label_dict'])
541             outputs = model.net(img)
542             return feeder, reader, outputs, gt_label
543
544
545 def load_pretrained_params(exe, program):
546     # 如果设置了增量训练, 则加载之前训练的模型
547     if train_params['continue_train'] and
os.path.exists(train_params['save_model_dir']):
548         logger.info('load param from retrain model')
549         fluid.io.load_persistables(executor=exe,
550
dirname=train_params['save_model_dir'],
551                                     main_program=program)
552     # 如果设置了预训练, 则加载预训练模型
553     elif train_params['pretrained'] and
os.path.exists(train_params['pretrained_model_dir']):
554         logger.info('load param from pretrained model')
555
556     def if_exist(var):

```

```

557         return
558     os.path.exists(os.path.join(train_params['pretrained_model_dir'],
559                                var.name))
560
561     fluid.io.load_vars(exe, train_params['pretrained_model_dir'],
562                       main_program=program,
563                       predicate=if_exist)
564
565 def train():
566     """
567     训练
568     :return:
569     """
570     init_log_config()
571     init_train_params()
572     logger.info("start train crnn, train params:%s",
573               str(train_params))
574
575     logger.info("create place, use gpu:" +
576               str(train_params['use_gpu']))
577     place = fluid.CUDAPlace(0) if train_params['use_gpu'] else
578     fluid.CPUPlace()
579
580     logger.info("build network and program")
581
582     train_program = fluid.Program()
583     start_program = fluid.Program()
584     eval_program = fluid.Program()
585     # start_program = fluid.Program() # wdb del 20200322
586
587     # 定义异步读取器、预测、构建损失函数及优化器
588     train_reader, loss, distances, seq_num, decoded_out = \
589         build_train_program_with_async_reader(train_program,
590         start_program)
591
592     # 评估
593     eval_feeder, eval_reader, output, gt_label = \
594         build_eval_program_with_feeder(eval_program, start_program,
595         place)
596
597     eval_program = eval_program.clone(for_test=True)
598
599     logger.info("build executor and init params")
600
601     exe = fluid.Executor(place)
602     exe.run(start_program)

```



```

596     train_fetch_list = [loss.name, distances.name, seq_num.name,
597                          decoded_out.name]
598     eval_fetch_list = [output.name]
599     load_pretrained_params(exe, train_program)
600
601     stop_strategy = train_params['early_stop']
602     successive_limit = stop_strategy['successive_limit']
603     sample_freq = stop_strategy['sample_frequency']
604     min_instance_error = stop_strategy['min_instance_error']
605     stop_train = False
606     successive_count = 0
607     total_batch_count = 0
608     distance_evaluator = fluid.metrics.EditDistance("edit-distance")
609
610     # 执行训练
611     for pass_id in range(train_params["num_epochs"]):
612         logger.info("current pass: %d, start read image", pass_id)
613         batch_id = 0
614         train_reader.start() # 启动reader线程
615         distance_evaluator.reset()
616
617         try:
618             while True:
619                 t1 = time.time()
620                 loss, distances, seq_num, decoded_out =
621                 exe.run(train_program,
622
623                        fetch_list=train_fetch_list,
624
625                        return_numpy=False)
626
627                 distances = np.array(distances)
628                 seq_num = np.array(seq_num)
629                 distance_evaluator.update(distances, seq_num)
630                 period = time.time() - t1
631                 loss = np.mean(np.array(loss))
632                 batch_id += 1
633                 total_batch_count += 1
634
635                 if batch_id % 10 == 0:
636                     distance, instance_error =
637                     distance_evaluator.eval()
638                     # logger.info(np.array(decoded_out))
639                     logger.info("Pass {0}, trainbatch {1}, loss {2}
640                                distance {3} instance error {4} time {5}"
641                                .format(pass_id, batch_id, loss,
642                                distance, instance_error, "%2.2f sec" % period))
635

```

```

636         # 采用简单的定时采样停止办法，可以调整为更精细的保存策略
637         if total_batch_count % 100 == 0:
638             logger.info("temp save {0} batch train
result".format(total_batch_count))
639
640         fluid.io.save_persistables(dirname=train_params['save_model_dir'],
main_program=train_program,
641                                     executor=exe)
642
643         if total_batch_count % sample_freq == 0:
644             if instance_error <= min_instance_error:
645                 successive_count += 1
646                 logger.info("instance error {0} successive
count {1}".format(instance_error, successive_count))
647                 if successive_count >= successive_limit:
648                     stop_train = True
649                     break
650             else:
651                 successive_count = 0
652
653         except fluid.core.EOFException:
654             train_reader.reset()
655
656         distance, instance_error = distance_evaluator.eval()
657         logger.info("Pass {0} distance {1} instance error
{2}".format(pass_id, distance, instance_error))
658
659         if stop_train:
660             logger.info("early stop")
661             break
662
663         logger.info("training till last, end training")
664         fluid.io.save_persistables(dirname=train_params['save_model_dir'],
main_program=train_program, executor=exe)
665
666
667 if __name__ == '__main__':
668     train()

```

3. 模型持久化

```

1
2 from __future__ import absolute_import
3 from __future__ import division
4 from __future__ import print_function

```

```
5
6 import os
7 import six
8 import numpy as np
9 import random
10 import time
11 import codecs
12 import sys
13 import functools
14 import math
15 import paddle
16 import paddle.fluid as fluid
17 from paddle.fluid import core
18 from paddle.fluid.param_attr import ParamAttr
19 from PIL import Image, ImageEnhance
20
21 # 读取 label_list.txt 文件获取类别数量
22 class_dim = -1
23 all_file_dir = "data/data6927/word-recognition"
24 with codecs.open(os.path.join(all_file_dir, "label_list.txt")) as
    label_list:
25     class_dim = len(label_list.readlines())
26 target_size = [1, 48, 512]
27 mean_rgb = 127.5
28 save_freeze_dir = "./crnn-model"
29
30
31 class CRNN(object):
32     def __init__(self, num_classes, label_dict):
33         self.outputs = None
34         self.label_dict = label_dict
35         self.num_classes = num_classes
36
37     def name(self):
38         return 'crnn'
39
40     def conv_bn_pool(self, input, group, out_ch, act="relu",
    param=None, bias=None, param_0=None, is_test=False,
41                     pooling=True, use_cudnn=False):
42         tmp = input
43         for i in six.moves.xrange(group):
44             tmp = fluid.layers.conv2d(
45                 input=tmp,
46                 num_filters=out_ch[i],
47                 filter_size=3,
48                 padding=1,
49                 param_attr=param if param_0 is None else param_0,
```

```

50         act=None, # LinearActivation
51         use_cudnn=use_cudnn)
52     tmp = fluid.layers.batch_norm(
53         input=tmp,
54         act=act,
55         param_attr=param,
56         bias_attr=bias,
57         is_test=is_test)
58     if pooling:
59         tmp = fluid.layers.pool2d(
60             input=tmp,
61             pool_size=2,
62             pool_type='max',
63             pool_stride=2,
64             use_cudnn=use_cudnn,
65             ceil_mode=True)
66
67     return tmp
68
69     def ocr_convs(self, input, regularizer=None, gradient_clip=None,
is_test=False, use_cudnn=False):
70         b = fluid.ParamAttr(
71             regularizer=regularizer,
72             gradient_clip=gradient_clip,
73             initializer=fluid.initializer.Normal(0.0, 0.0))
74         w0 = fluid.ParamAttr(
75             regularizer=regularizer,
76             gradient_clip=gradient_clip,
77             initializer=fluid.initializer.Normal(0.0, 0.0005))
78         w1 = fluid.ParamAttr(
79             regularizer=regularizer,
80             gradient_clip=gradient_clip,
81             initializer=fluid.initializer.Normal(0.0, 0.01))
82         tmp = input
83         tmp = self.conv_bn_pool(
84             tmp,
85             2, [16, 16],
86             param=w1,
87             bias=b,
88             param_0=w0,
89             is_test=is_test,
90             use_cudnn=use_cudnn)
91
92         tmp = self.conv_bn_pool(
93             tmp,
94             2, [32, 32],
95             param=w1,

```

```

96         bias=b,
97         is_test=is_test,
98         use_cudnn=use_cudnn)
99     tmp = self.conv_bn_pool(
100         tmp,
101         2, [64, 64],
102         param=w1,
103         bias=b,
104         is_test=is_test,
105         use_cudnn=use_cudnn)
106     tmp = self.conv_bn_pool(
107         tmp,
108         2, [128, 128],
109         param=w1,
110         bias=b,
111         is_test=is_test,
112         pooling=False,
113         use_cudnn=use_cudnn)
114     return tmp
115
116 def net(self, images, rnn_hidden_size=200, regularizer=None,
117         gradient_clip=None, is_test=False, use_cudnn=True):
118     conv_features = self.ocr_convs(
119         images,
120         regularizer=regularizer,
121         gradient_clip=gradient_clip,
122         is_test=is_test,
123         use_cudnn=use_cudnn)
124     sliced_feature = fluid.layers.im2sequence(
125         input=conv_features,
126         stride=[1, 1],
127         filter_size=[conv_features.shape[2], 1])
128
129     para_attr = fluid.ParamAttr(
130         regularizer=regularizer,
131         gradient_clip=gradient_clip,
132         initializer=fluid.initializer.Normal(0.0, 0.02))
133     bias_attr = fluid.ParamAttr(
134         regularizer=regularizer,
135         gradient_clip=gradient_clip,
136         initializer=fluid.initializer.Normal(0.0, 0.02))
137     bias_attr_nobias = fluid.ParamAttr(
138         regularizer=regularizer,
139         gradient_clip=gradient_clip,
140         initializer=fluid.initializer.Normal(0.0, 0.02))
141
142     fc_1 = fluid.layers.fc(input=sliced_feature,

```

```

143         size=rnn_hidden_size * 3,
144         param_attr=para_attr,
145         bias_attr=bias_attr_nobias)
146     fc_2 = fluid.layers.fc(input=sliced_feature,
147         size=rnn_hidden_size * 3,
148         param_attr=para_attr,
149         bias_attr=bias_attr_nobias)
150
151     gru_forward = fluid.layers.dynamic_gru(
152         input=fc_1,
153         size=rnn_hidden_size,
154         param_attr=para_attr,
155         bias_attr=bias_attr,
156         candidate_activation='relu')
157     gru_backward = fluid.layers.dynamic_gru(
158         input=fc_2,
159         size=rnn_hidden_size,
160         is_reverse=True,
161         param_attr=para_attr,
162         bias_attr=bias_attr,
163         candidate_activation='relu')
164
165     w_attr = fluid.ParamAttr(
166         regularizer=regularizer,
167         gradient_clip=gradient_clip,
168         initializer=fluid.initializer.Normal(0.0, 0.02))
169     b_attr = fluid.ParamAttr(
170         regularizer=regularizer,
171         gradient_clip=gradient_clip,
172         initializer=fluid.initializer.Normal(0.0, 0.0))
173
174     fc_out = fluid.layers.fc(input=[gru_forward, gru_backward],
175         size=self.num_classes + 1,
176         param_attr=w_attr,
177         bias_attr=b_attr)
178     self.outputs = fc_out
179     return fc_out
180
181     def get_loss(self, label):
182         cost = fluid.layers.warpctc(input=self.outputs, label=label,
183         blank=self.num_classes, norm_by_times=True)
184         sum_cost = fluid.layers.reduce_sum(cost)
185         return sum_cost
186
187     def get_infer(self):
188         return fluid.layers.ctc_greedy_decoder(input=self.outputs,
189         blank=self.num_classes)

```

```

188
189
190 def freeze_model():
191     """
192     保存模型
193     :return:
194     """
195     exe = fluid.Executor(fluid.CPUPlace())
196     image = fluid.layers.data(name='image', shape=target_size,
dtype='float32')
197
198     model = CRNN(class_dim, {}) # 创建CRNN模型
199     pred = model.net(image) # 组网
200     out = model.get_infer()
201
202     freeze_program = fluid.default_main_program()
203     fluid.io.load_persistables(exe, save_freeze_dir, freeze_program)
204     # 加载模型
205     freeze_program = freeze_program.clone(for_test=True)
206     fluid.io.save_inference_model("./freeze-model", ['image'], out,
exe, freeze_program) # 保存模型
207
208 if __name__ == '__main__':
209     freeze_model()

```

4. 测试

```

1 from __future__ import absolute_import
2 from __future__ import division
3 from __future__ import print_function
4
5 import os
6 import numpy as np
7 import random
8 import time
9 import codecs
10 import sys
11 import functools
12 import math
13 import paddle
14 import paddle.fluid as fluid
15 from paddle.fluid import core
16 from paddle.fluid.param_attr import ParamAttr
17 from PIL import Image, ImageEnhance
18 import matplotlib.pyplot as plt

```

```

19
20 target_size = [1, 48, 512]
21 mean_rgb = 127.5
22 data_dir = 'data/data6927/word-recognition'
23 eval_file = "eval.txt"
24 label_list = "label_list.txt"
25 use_gpu = True
26 label_dict = {}
27 save_freeze_dir = "./freeze-model"
28
29 place = fluid.CUDAPlace(0) if use_gpu else fluid.CPUPlace()
30 exe = fluid.Executor(place)
31
32 # 加载模型
33 [inference_program, feed_target_names, fetch_targets] = \
34     fluid.io.load_inference_model(dirname=save_freeze_dir,
35                                   executor=exe)
36
37 # print(fetch_targets)
38
39
40 def init_eval_parameters():
41     """
42     初始化训练参数，主要是初始化图片数量，类别数
43     :return:
44     """
45     label_list_path = os.path.join(data_dir, label_list)
46     index = 0
47
48     # 读取样本文件内容，并存入字典
49     with codecs.open(label_list_path, encoding='utf-8') as flist:
50         lines = [line.strip() for line in flist]
51         for line in lines:
52             parts = line.split()
53             label_dict[int(parts[1])] = parts[0]
54
55
56 def resize_img(img):
57     """
58     重设图像大小
59     :param img:
60     :return:
61     """
62     percent_h = float(target_size[1]) / img.size[1]
63     percent_w = float(target_size[2]) / img.size[0]
64     percent = min(percent_h, percent_w)

```



```

65
66     resized_width = int(round(img.size[0] * percent))
67     resized_height = int(round(img.size[1] * percent))
68
69     w_off = (target_size[2] - resized_width) / 2
70     h_off = (target_size[1] - resized_height) / 2
71
72     img = img.resize((resized_width, resized_height), Image.ANTIALIAS)
73
74     array = np.ndarray((target_size[1], target_size[2], 3), np.uint8)
75     array[:, :, 0] = 127
76     array[:, :, 1] = 127
77     array[:, :, 2] = 127
78     ret = Image.fromarray(array)
79     ret.paste(img, (np.random.randint(0, w_off + 1), int(h_off)))
80     return ret
81
82
83 def read_image(img_path):
84     """
85     读取图像
86     :param img_path:
87     :return:
88     """
89     img = Image.open(img_path)
90     if img.mode != 'RGB':
91         img = img.convert('RGB')
92     img = resize_img(img)
93     img = img.convert('L') # 返回一个转换后的副本，L模式进行转换
94     img = np.array(img).astype('float32') - mean_rgb
95     img = img[..., np.newaxis]
96     img = img.transpose((2, 0, 1))
97     img = img[np.newaxis, :]
98     return img
99
100
101 def infer(image_path):
102     """
103     执行预测
104     :param image_path:
105     :return:
106     """
107     tensor_img = read_image(image_path)
108
109     label = exe.run(inference_program,
110                     feed={feed_target_names[0]: tensor_img},
111                     fetch_list=fetch_targets,

```

```

112         return_numpy=False)
113     label = np.array(label[0])
114     ret = ""
115     if label[0] != -1:
116         ret = ret.join([label_dict[int(c[0])] for c in label])
117     return ret
118
119
120 def eval_all():
121     """
122     评估所有
123     :return:
124     """
125     eval_file_path = os.path.join(data_dir, eval_file) # 评估文件路径
126     total_count = 0
127     right_count = 0
128
129     with codecs.open(eval_file_path, encoding='utf-8') as flist:
130         lines = [line.strip() for line in flist]
131         t1 = time.time()
132         random.shuffle(lines) # 打乱样本
133
134         i = 0
135         for line in lines:
136             i += 1
137             if i > 3:
138                 break
139
140             total_count += 1
141             parts = line.strip().split()
142
143             result = infer(parts[0]) # 执行推测
144
145             img = Image.open(parts[0])
146             plt.imshow(img)
147             plt.show()
148             plt.savefig("logs/%s" % parts[0])
149
150             print("infer result:{0} answer:{1}".format(result,
151 parts[1]))
152
153             if str(result) == parts[1]:
154                 right_count += 1
155
156             period = time.time() - t1
157             print("total eval count:{0} cost time:{1} predict accuracy:
158 {2}".format(total_count, "%2.2f sec" % period,

```

```
157         right_count / total_count))
158
159
160 if __name__ == '__main__':
161     init_eval_parameters()
162     eval_all()
```