光学文字识别代码实现

1. 数据预处理

```
# 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
import codecs
import os
import random
import shutil
from PIL import Image

train_ratio = 9 / 10  # 训练集大小
all_file_dir = "data/data6927/word-recognition"  # 数据文件路径
image_path_pre = os.path.join(all_file_dir, "imageSet")  # 路径

# 创建训练集路径
train_image_dir = os.path.join(all_file_dir, "trainImageSet")
if not os.path.exists(train_image_dir):
    os.makedirs(train_image_dir)

eval_image_dir = os.path.join(all_file_dir, "evalImageSet")
if not os.path.exists(eval_image_dir):
    os.makedirs(eval_image_dir)

train_file = codecs.open(os.path.join(all_file_dir, "train.txt"), 'w')
eval_file = codecs.open(os.path.join(all_file_dir, "eval.txt"), 'w')
label_list = os.path.join(all_file_dir, "image_label.txt")  # 标签文件

train_count = 0
eval_count = 0
class_set = set()
with open(label_list) as f:
    for line in f:
        parts = line.strip().split()
        file, label = parts[0], parts[1]
        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:
            continue
        for e in label:
            class_set.add(e)
        if random.uniform(0, 1) <= train_ratio:
```

```
            shutil.copyfile(os.path.join(image_path_pre, file), os.path.join(train_image_dir, file))
            train_file.write("{0}\t{1}\n".format(os.path.join(train_image_dir, file), label))
            train_count += 1
        else:
            shutil.copyfile(os.path.join(image_path_pre, file), os.path.join(eval_image_dir, file))
            eval_file.write("{0}\t{1}\n".format(os.path.join(eval_image_dir, file), label))
            eval_count += 1

print("train image count: {0} eval image count: {1}".format(train_count, eval_count))
class_list = list(class_set)
class_list.sort()
print("class num: {0}".format(len(class_list)))
print(class_list)
with codecs.open(os.path.join(all_file_dir, "label_list.txt"), "w") as label_list:
    label_id = 0
    for c in class_list:
        label_list.write("{0}\t{1}\n".format(c, label_id))
        label_id += 1
```

2. 模型训练

```
# -*- coding: UTF-8 -*-
"""
训练常基于crnn-ctc的网络，文字行识别
"""
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function
import os
import uuid
import numpy as np
import time
import six
import math
import random
import paddle
import paddle.fluid as fluid
import logging
import xml.etree.ElementTree
import codecs
```

```
import json

from paddle.fluid.initializer import MSRA
from paddle.fluid.param_attr import ParamAttr
from paddle.fluid.regularizer import L2Decay
from PIL import Image, ImageEnhance, ImageDraw

logger = None
train_params = {
    "input_size": [1, 48, 512],  # 输入数据维度
    "data_dir": "data/data6927/word-recognition",  # 数据集路径
    "train_dir": "trainImageSet",  # 训练数据目录
    "eval_dir": "evalImageSet",  # 评估数据目录
    "train_list": "train.txt",  # 训练集文件
    "eval_list": "eval.txt",  # 评估集文件
    "label_list": "label_list.txt",  # 标签文件
    "class_dim": -1,
    "label_dict": {},  # 标签字典
    "image_count": -1,
    "continue_train": False,
    "pretrained": True,  # 预训练
    "pretrained_model_dir": "./pretrained-model",  # 预训练模型目录
    "save_model_dir": "./crnn-model",  # 模型保存目录
    "num_epochs": 400,  # 训练轮次
    "train_batch_size": 256,  # 训练批次大小
    "use_gpu": True,  # 是否使用gpu
    "ignore_thresh": 0.7,  # 阈值
    "mean_color": 127.5,  #
    "mode": "train",  # 模式
    "multi_data_reader_count": 4,  # reader数量
    "apply_distort": True,  # 是否进行扭曲
    "image_distort_strategy": {  # 扭曲策略
        "expand_prob": 0.5,  # 放大比率
        "expand_max_ratio": 2,  # 最大放大比率
        "hue_prob": 0.5,  # 色调
        "hue_delta": 18,
        "contrast_prob": 0.5,  # 对比度
        "contrast_delta": 0.5,
        "saturation_prob": 0.5,  # 饱和度
        "saturation_delta": 0.5,
        "brightness_prob": 0.5,  # 亮度
        "brightness_delta": 0.125
    },
    "rsm_strategy": {  # 梯度下降配置
        "learning_rate": 0.0005,
        "lr_epochs": [70, 120, 170, 220, 270, 320],  # 学习率衰减分段（6
个数字分为7段）
```

```
        "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
    },
    "early_stop": { # 控制训练停止条件
        "sample_frequency": 50,
        "successive_limit": 5,
        "min_instance_error": 0.1
    }
}


# CRNN网络模型
class CRNN(object):
    def __init__(self, num_classes, label_dict):
        self.outputs = None  # 输出
        self.label_dict = label_dict  # 标签字典
        self.num_classes = num_classes  # 类别数量

    def name(self):
        return 'crnn'

    def conv_bn_pool(self, input, group, out_ch, act="relu", param=None,
                     bias=None, param_0=None, is_test=False, pooling=True, use_cudnn=False):
        tmp = input

        # 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
        # 它旨在支持无需修改即可在Python 2和Python 3上工作的代码库
        # six只包含一个Python文件，因此无需复制到一个项目中
        for i in six.moves.xrange(group):
            # 卷积层
            tmp = fluid.layers.conv2d(input=tmp,  # 输入数据
                                      num_filters=out_ch[i],  # 卷积核数量
                                      filter_size=3,  # 卷积核大小
                                      padding=1,  # 填充
                                      param_attr=param if param_0 is None else param_0,
                                      act=None,  # LinearActivation
                                      use_cudnn=use_cudnn)
            # batch normal
            tmp = fluid.layers.batch_norm(input=tmp,
                                          act=act,
                                          param_attr=param,
                                          bias_attr=bias,
                                          is_test=is_test)
```

```
        # 池化
        if pooling:
            tmp = fluid.layers.pool2d(input=tmp,  # 输入数据
                                      pool_size=2,  # 池化区域大小
                                      pool_type='max',  # 池化类型
                                      pool_stride=2,  # 池化步长
                                      use_cudnn=use_cudnn,  # 是否使用cudnn
                                      ceil_mode=True)

        return tmp

    # OCR convs
    def ocr_convs(self, input, regularizer=None, gradient_clip=None, is_test=False, use_cudnn=False):
        b = fluid.ParamAttr(regularizer=regularizer,
                            gradient_clip=gradient_clip,
                            initializer=fluid.initializer.Normal(0.0, 0.0)) # 初始化为正态分布
        w0 = fluid.ParamAttr(regularizer=regularizer,
                             gradient_clip=gradient_clip,
                             initializer=fluid.initializer.Normal(0.0, 0.0005))# 初始化为正态分布
        w1 = fluid.ParamAttr(regularizer=regularizer,
                             gradient_clip=gradient_clip,
                             initializer=fluid.initializer.Normal(0.0, 0.01))# 初始化为正态分布

        tmp = input
        # 第一组卷积池化
        tmp = self.conv_bn_pool(tmp,
                                2,  # 组数量
                                [16, 16],  # 输出
                                param=w1,
                                bias=b,
                                param_0=w0,
                                is_test=is_test,
                                use_cudnn=use_cudnn)
        # 第二组卷积池化
        tmp = self.conv_bn_pool(tmp,
                                2, [32, 32],
                                param=w1,
                                bias=b,
                                is_test=is_test,
                                use_cudnn=use_cudnn)
        # 第三组卷积池化
        tmp = self.conv_bn_pool(tmp,
```

```
                                2, [64, 64],
                                param=w1,
                                bias=b,
                                is_test=is_test,
                                use_cudnn=use_cudnn)
        # 第四组卷积池化
        tmp = self.conv_bn_pool(tmp,
                                2, [128, 128],
                                param=w1,
                                bias=b,
                                is_test=is_test,
                                pooling=False,
                                use_cudnn=use_cudnn)
        return tmp

    # 组网函数
    def net(self, images, rnn_hidden_size=200, regularizer=None,
            gradient_clip=None, is_test=False, use_cudnn=True):
        # 卷积池化
        conv_features = self.ocr_convs(images,
                                       regularizer=regularizer,
                                       gradient_clip=gradient_clip,
                                       is_test=is_test,
                                       use_cudnn=use_cudnn)
        # 转序列
        sliced_feature = fluid.layers.im2sequence(input=conv_features,
                                                  stride=[1, 1],
                                                  filter_size=
[conv_features.shape[2], 1])

        para_attr = fluid.ParamAttr(regularizer=regularizer,
                                    gradient_clip=gradient_clip,

 initializer=fluid.initializer.Normal(0.0, 0.02))
        bias_attr = fluid.ParamAttr(regularizer=regularizer,
                                    gradient_clip=gradient_clip,

 initializer=fluid.initializer.Normal(0.0, 0.02))
        bias_attr_nobias = fluid.ParamAttr(regularizer=regularizer,

gradient_clip=gradient_clip,

initializer=fluid.initializer.Normal(0.0, 0.02))
        # 全连接层
        fc_1 = fluid.layers.fc(input=sliced_feature,
                               size=rnn_hidden_size * 3,
                               param_attr=para_attr,
```

```
                                bias_attr=bias_attr_nobias)
        fc_2 = fluid.layers.fc(input=sliced_feature,
                               size=rnn_hidden_size * 3,
                               param_attr=para_attr,
                               bias_attr=bias_attr_nobias)
        # gru(门控循环单元), LSTM变种
        # 对检测到的字符连接成字符串序列
        gru_forward = fluid.layers.dynamic_gru(input=fc_1,
                                               size=rnn_hidden_size,
                                               param_attr=para_attr,
                                               bias_attr=bias_attr,
candidate_activation='relu')
        gru_backward = fluid.layers.dynamic_gru(input=fc_2,
                                                size=rnn_hidden_size,
                                                is_reverse=True,
                                                param_attr=para_attr,
                                                bias_attr=bias_attr,
 candidate_activation='relu')

        w_attr = fluid.ParamAttr(regularizer=regularizer,
                                 gradient_clip=gradient_clip,
initializer=fluid.initializer.Normal(0.0, 0.02))
        b_attr = fluid.ParamAttr(regularizer=regularizer,
                                 gradient_clip=gradient_clip,
initializer=fluid.initializer.Normal(0.0, 0.0))

        fc_out = fluid.layers.fc(input=[gru_forward, gru_backward],
                                 size=self.num_classes + 1,
                                 param_attr=w_attr,
                                 bias_attr=b_attr)
        self.outputs = fc_out
        return fc_out

    def get_infer(self):
        return fluid.layers.ctc_greedy_decoder(input=self.outputs,
blank=self.num_classes)


def init_train_params():
    """
    初始化训练参数，主要是初始化图片数量，类别数
    :return:
    """
```

```
    train_list = os.path.join(train_params['data_dir'], train_params['train_list'])
    label_list = os.path.join(train_params['data_dir'], train_params['label_list'])

    index = 0

    with codecs.open(label_list, encoding='utf-8') as flist:
        lines = [line.strip() for line in flist]
        for line in lines:
            parts = line.split()
            train_params['label_dict'][parts[0]] = int(parts[1])
            index += 1
        train_params['class_dim'] = index

    with codecs.open(train_list, encoding='utf-8') as flist:
        lines = [line.strip() for line in flist]
        train_params['image_count'] = len(lines)


# 初始化日志相关配置
def init_log_config():
    global logger
    logger = logging.getLogger()
    logger.setLevel(logging.INFO)
    log_path = os.path.join(os.getcwd(), 'logs')
    if not os.path.exists(log_path):
        os.makedirs(log_path)
    log_name = os.path.join(log_path, 'train.log')
    sh = logging.StreamHandler()
    fh = logging.FileHandler(log_name, mode='w')
    fh.setLevel(logging.DEBUG)
    formatter = logging.Formatter("%(asctime)s - %(filename)s[line:%(lineno)d] - %(levelname)s: %(message)s")
    fh.setFormatter(formatter)
    sh.setFormatter(formatter)
    logger.addHandler(sh)
    logger.addHandler(fh)


# 重设图像大小
def resize_img(img, input_size):
    target_size = input_size
    percent_h = float(target_size[1]) / img.size[1]
    percent_w = float(target_size[2]) / img.size[0]
    percent = min(percent_h, percent_w)
    resized_width = int(round(img.size[0] * percent))
```

```
    resized_height = int(round(img.size[1] * percent))
    w_off = (target_size[2] - resized_width) / 2
    h_off = (target_size[1] - resized_height) / 2
    img = img.resize((resized_width, resized_height), Image.ANTIALIAS)
    array = np.ndarray((target_size[1], target_size[2], 3), np.uint8)
    array[:, :, 0] = 127
    array[:, :, 1] = 127
    array[:, :, 2] = 127
    ret = Image.fromarray(array)
    ret.paste(img, (np.random.randint(0, w_off + 1), int(h_off)))
    return ret


# 调节亮度
def random_brightness(img):
    prob = np.random.uniform(0, 1)
    if prob < train_params['image_distort_strategy']
['brightness_prob']:
        brightness_delta = train_params['image_distort_strategy']
['brightness_delta']
        delta = np.random.uniform(-brightness_delta, brightness_delta)
+ 1
        img = ImageEnhance.Brightness(img).enhance(delta)
    return img


# 对比度
def random_contrast(img):
    prob = np.random.uniform(0, 1)
    if prob < train_params['image_distort_strategy']['contrast_prob']:
        contrast_delta = train_params['image_distort_strategy']
['contrast_delta']
        delta = np.random.uniform(-contrast_delta, contrast_delta) + 1
        img = ImageEnhance.Contrast(img).enhance(delta)
    return img


# 饱和度
def random_saturation(img):
    prob = np.random.uniform(0, 1)
    if prob < train_params['image_distort_strategy']
['saturation_prob']:
        saturation_delta = train_params['image_distort_strategy']
['saturation_delta']
        delta = np.random.uniform(-saturation_delta, saturation_delta)
+ 1
        img = ImageEnhance.Color(img).enhance(delta)
```

```
    return img


def random_hue(img):
    prob = np.random.uniform(0, 1)
    if prob < train_params['image_distort_strategy']['hue_prob']:
        hue_delta = train_params['image_distort_strategy']
['hue_delta']
        delta = np.random.uniform(-hue_delta, hue_delta)
        img_hsv = np.array(img.convert('HSV'))
        img_hsv[:, :, 0] = img_hsv[:, :, 0] + delta
        img = Image.fromarray(img_hsv, mode='HSV').convert('RGB')
    return img


def distort_image(img):
    prob = np.random.uniform(0, 1)
    # Apply different distort order
    if prob > 0.5:
        img = random_brightness(img)
        img = random_contrast(img)
        img = random_saturation(img)
        img = random_hue(img)
    else:
        img = random_brightness(img)
        img = random_saturation(img)
        img = random_hue(img)
        img = random_contrast(img)
    return img


def rotate_image(img):
    """
    图像增强，增加随机旋转角度
    """
    prob = np.random.uniform(0, 1)
    if prob > 0.5:
        angle = np.random.randint(-8, 8)
        img = img.rotate(angle)
    return img


def random_expand(img, keep_ratio=True):
    if np.random.uniform(0, 1) <
train_params['image_distort_strategy']['expand_prob']:
        return img

```

```
    max_ratio = train_params['image_distort_strategy']
['expand_max_ratio']
    w, h = img.size
    c = 3
    ratio_x = random.uniform(1, max_ratio)
    if keep_ratio:
        ratio_y = ratio_x
    else:
        ratio_y = random.uniform(1, max_ratio)
    oh = int(h * ratio_y)
    ow = int(w * ratio_x)
    off_x = random.randint(0, ow - w)
    off_y = random.randint(0, oh - h)

    out_img = np.zeros((oh, ow, c), np.uint8)
    for i in range(c):
        out_img[:, :, i] = train_params['mean_color']

    out_img[off_y: off_y + h, off_x: off_x + w, :] = img

    return Image.fromarray(out_img)


def preprocess(img, input_size):
    img_width, img_height = img.size
    if train_params['apply_distort']:
        img = distort_image(img)
    img = random_expand(img)
    img = rotate_image(img)
    # img = resize_img(img, input_size)
    # img = img.convert('L')
    # img = np.array(img).astype('float32') -
train_params['mean_color']
    # img *= 0.007843
    return img


# reader
def custom_reader(file_list, data_dir, input_size, mode):
    def reader():
        np.random.shuffle(file_list)
        for line in file_list:
            # img_name, label
            parts = line.split()
            image_path = parts[0]
            img = Image.open(image_path)
            # img = Image.open(os.path.join(data_dir, image_path))
```

```
            if img.mode != 'RGB':
                img = img.convert('RGB')
            label = [int(train_params['label_dict'][c]) for c in
parts[-1]]
            if len(label) == 0:
                continue
            if mode == 'train':
                img = preprocess(img, input_size)
            img = resize_img(img, input_size)
            img = img.convert('L')
            # img.save(image_path)
            img = np.array(img).astype('float32') -
train_params['mean_color']
            # img *= 0.007843
            img = img[np.newaxis, ...]
            # print("{0} {1}".format(image_path, label))
            yield img, label

    return reader


def multi_process_custom_reader(file_path, data_dir, num_workers,
input_size, mode):
    """
    创建多进程reader
    :param file_path:
    :param data_dir:
    :param num_workers:
    :param input_size:
    :param mode:
    :return:
    """
    file_path = os.path.join(data_dir, file_path)
    readers = []
    images = [line.strip() for line in open(file_path)]
    n = int(math.ceil(len(images) // num_workers))
    image_lists = [images[i: i + n] for i in range(0, len(images), n)]
    train_path = os.path.join(train_params['data_dir'],
train_params['train_dir'])
    for l in image_lists:
        reader = paddle.batch(custom_reader(l, train_path, input_size,
mode),

 batch_size=train_params['train_batch_size'])
        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(boundaries, 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)
```

```
        # 创建reader
        data_reader = fluid.layers.create_py_reader_by_data(capacity=train_params['train_batch_size'],
                                                            feed_list=[img, gt_label],
                                                            name='train')
        # 创建多进程reader
        multi_reader = multi_process_custom_reader(train_params['train_list'],
                                                   train_params['data_dir'],
                                                   train_params['multi_data_reader_count'],
                                                   train_params['input_size'],
                                                   'train')
        data_reader.decorate_paddle_reader(multi_reader)

        with fluid.unique_name.guard():  # 更换namespace
            img, gt_label = fluid.layers.read_file(data_reader)

            model = CRNN(train_params['class_dim'], train_params['label_dict'])  # 实例化
            fc_out = model.net(img)  # 预测

            cost = fluid.layers.warpctc(input=fc_out, label=gt_label, blank=train_params['class_dim'],
                                        norm_by_times=True)  # 计算CTC损失函数
            loss = fluid.layers.reduce_sum(cost)  # 损失函数求和
            optimizer = optimizer_rms_setting()
            optimizer.minimize(loss)
            # 执行CTC去重
            decoded_out = fluid.layers.ctc_greedy_decoder(input=fc_out,
                                                          blank=train_params['class_dim'])
            casted_label = fluid.layers.cast(x=gt_label, dtype='int64')
            # 计算字符串的编辑距离
            # 编辑距离又称Levenshtein距离，由俄罗斯的数学家Vladimir Levenshtein在1965年提出
            # 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
            # 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'],
536                                     train_params['data_dir'],
537                                     train_params['input_size'],
538                                     'eval')
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):
```

```
            return
os.path.exists(os.path.join(train_params['pretrained_model_dir'],
var.name))

        fluid.io.load_vars(exe, train_params['pretrained_model_dir'],
main_program=program,
                           predicate=if_exist)


def train():
    """
    训练
    :return:
    """
    init_log_config()
    init_train_params()
    logger.info("start train crnn, train params:%s",
str(train_params))

    logger.info("create place, use gpu:" +
str(train_params['use_gpu']))
    place = fluid.CUDAPlace(0) if train_params['use_gpu'] else
fluid.CPUPlace()

    logger.info("build network and program")

    train_program = fluid.Program()
    start_program = fluid.Program()
    eval_program = fluid.Program()
    # start_program = fluid.Program()  # wdb del 20200322

    # 定义异步读取器、预测、构建损失函数及优化器
    train_reader, loss, distances, seq_num, decoded_out = \
        build_train_program_with_async_reader(train_program,
start_program)

    # 评估
    eval_feeder, eval_reader, output, gt_label = \
        build_eval_program_with_feeder(eval_program, start_program,
place)

    eval_program = eval_program.clone(for_test=True)

    logger.info("build executor and init params")

    exe = fluid.Executor(place)
    exe.run(start_program)
```

```
    train_fetch_list = [loss.name, distances.name, seq_num.name,
decoded_out.name]
    eval_fetch_list = [output.name]
    load_pretrained_params(exe, train_program)

    stop_strategy = train_params['early_stop']
    successive_limit = stop_strategy['successive_limit']
    sample_freq = stop_strategy['sample_frequency']
    min_instance_error = stop_strategy['min_instance_error']
    stop_train = False
    successive_count = 0
    total_batch_count = 0
    distance_evaluator = fluid.metrics.EditDistance("edit-distance")

    # 执行训练
    for pass_id in range(train_params["num_epochs"]):
        logger.info("current pass: %d, start read image", pass_id)
        batch_id = 0
        train_reader.start()  # 启动reader线程
        distance_evaluator.reset()

        try:
            while True:
                t1 = time.time()
                loss, distances, seq_num, decoded_out =
exe.run(train_program,

 fetch_list=train_fetch_list,

 return_numpy=False)
                distances = np.array(distances)
                seq_num = np.array(seq_num)
                distance_evaluator.update(distances, seq_num)
                period = time.time() - t1
                loss = np.mean(np.array(loss))
                batch_id += 1
                total_batch_count += 1

                if batch_id % 10 == 0:
                    distance, instance_error =
distance_evaluator.eval()
                    # logger.info(np.array(decoded_out))
                    logger.info("Pass {0}, trainbatch {1}, loss {2}
distance {3} instance error {4} time {5}"
                                .format(pass_id, batch_id, loss,
distance, instance_error, "%2.2f sec" % period))

```

```
                # 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
                if total_batch_count % 100 == 0:
                    logger.info("temp save {0} batch train result".format(total_batch_count))

 fluid.io.save_persistables(dirname=train_params['save_model_dir'], main_program=train_program,
                                                executor=exe)

                if total_batch_count % sample_freq == 0:
                    if instance_error <= min_instance_error:
                        successive_count += 1
                        logger.info("instance error {0} successive count {1}".format(instance_error, successive_count))
                        if successive_count >= successive_limit:
                            stop_train = True
                            break
                    else:
                        successive_count = 0

        except fluid.core.EOFException:
            train_reader.reset()

        distance, instance_error = distance_evaluator.eval()
        logger.info("Pass {0} distance {1} instance error {2}".format(pass_id, distance, instance_error))

        if stop_train:
            logger.info("early stop")
            break

    logger.info("training till last, end training")
    fluid.io.save_persistables(dirname=train_params['save_model_dir'], main_program=train_program, executor=exe)


if __name__ == '__main__':
    train()
```

3. 模型持久化

```

from __future__ import absolute_import
from __future__ import division
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,
```

```
                act=None,  # LinearActivation
                use_cudnn=use_cudnn)
            tmp = fluid.layers.batch_norm(
                input=tmp,
                act=act,
                param_attr=param,
                bias_attr=bias,
                is_test=is_test)
        if pooling:
            tmp = fluid.layers.pool2d(
                input=tmp,
                pool_size=2,
                pool_type='max',
                pool_stride=2,
                use_cudnn=use_cudnn,
                ceil_mode=True)

        return tmp

    def ocr_convs(self, input, regularizer=None, gradient_clip=None, is_test=False, use_cudnn=False):
        b = fluid.ParamAttr(
            regularizer=regularizer,
            gradient_clip=gradient_clip,
            initializer=fluid.initializer.Normal(0.0, 0.0))
        w0 = fluid.ParamAttr(
            regularizer=regularizer,
            gradient_clip=gradient_clip,
            initializer=fluid.initializer.Normal(0.0, 0.0005))
        w1 = fluid.ParamAttr(
            regularizer=regularizer,
            gradient_clip=gradient_clip,
            initializer=fluid.initializer.Normal(0.0, 0.01))
        tmp = input
        tmp = self.conv_bn_pool(
            tmp,
            2, [16, 16],
            param=w1,
            bias=b,
            param_0=w0,
            is_test=is_test,
            use_cudnn=use_cudnn)

        tmp = self.conv_bn_pool(
            tmp,
            2, [32, 32],
            param=w1,
```

```
            bias=b,
            is_test=is_test,
            use_cudnn=use_cudnn)
        tmp = self.conv_bn_pool(
            tmp,
            2, [64, 64],
            param=w1,
            bias=b,
            is_test=is_test,
            use_cudnn=use_cudnn)
        tmp = self.conv_bn_pool(
            tmp,
            2, [128, 128],
            param=w1,
            bias=b,
            is_test=is_test,
            pooling=False,
            use_cudnn=use_cudnn)
        return tmp

    def net(self, images, rnn_hidden_size=200, regularizer=None,
            gradient_clip=None, is_test=False, use_cudnn=True):
        conv_features = self.ocr_convs(
            images,
            regularizer=regularizer,
            gradient_clip=gradient_clip,
            is_test=is_test,
            use_cudnn=use_cudnn)
        sliced_feature = fluid.layers.im2sequence(
            input=conv_features,
            stride=[1, 1],
            filter_size=[conv_features.shape[2], 1])

        para_attr = fluid.ParamAttr(
            regularizer=regularizer,
            gradient_clip=gradient_clip,
            initializer=fluid.initializer.Normal(0.0, 0.02))
        bias_attr = fluid.ParamAttr(
            regularizer=regularizer,
            gradient_clip=gradient_clip,
            initializer=fluid.initializer.Normal(0.0, 0.02))
        bias_attr_nobias = fluid.ParamAttr(
            regularizer=regularizer,
            gradient_clip=gradient_clip,
            initializer=fluid.initializer.Normal(0.0, 0.02))

        fc_1 = fluid.layers.fc(input=sliced_feature,
```

```
                                size=rnn_hidden_size * 3,
                                param_attr=para_attr,
                                bias_attr=bias_attr_nobias)
        fc_2 = fluid.layers.fc(input=sliced_feature,
                                size=rnn_hidden_size * 3,
                                param_attr=para_attr,
                                bias_attr=bias_attr_nobias)

        gru_forward = fluid.layers.dynamic_gru(
            input=fc_1,
            size=rnn_hidden_size,
            param_attr=para_attr,
            bias_attr=bias_attr,
            candidate_activation='relu')
        gru_backward = fluid.layers.dynamic_gru(
            input=fc_2,
            size=rnn_hidden_size,
            is_reverse=True,
            param_attr=para_attr,
            bias_attr=bias_attr,
            candidate_activation='relu')

        w_attr = fluid.ParamAttr(
            regularizer=regularizer,
            gradient_clip=gradient_clip,
            initializer=fluid.initializer.Normal(0.0, 0.02))
        b_attr = fluid.ParamAttr(
            regularizer=regularizer,
            gradient_clip=gradient_clip,
            initializer=fluid.initializer.Normal(0.0, 0.0))

        fc_out = fluid.layers.fc(input=[gru_forward, gru_backward],
                                 size=self.num_classes + 1,
                                 param_attr=w_attr,
                                 bias_attr=b_attr)
        self.outputs = fc_out
        return fc_out

    def get_loss(self, label):
        cost = fluid.layers.warpctc(input=self.outputs, label=label,
blank=self.num_classes, norm_by_times=True)
        sum_cost = fluid.layers.reduce_sum(cost)
        return sum_cost

    def get_infer(self):
        return fluid.layers.ctc_greedy_decoder(input=self.outputs,
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)
     # 加载模型
204     freeze_program = freeze_program.clone(for_test=True)
205     fluid.io.save_inference_model("./freeze-model", ['image'], out,
    exe, freeze_program)  # 保存模型
206
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,
executor=exe)
35
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,
```

```
                    return_numpy=False)
    label = np.array(label[0])
    ret = ""
    if label[0] != -1:
        ret = ret.join([label_dict[int(c[0])] for c in label])
    return ret


def eval_all():
    """
    评估所有
    :return:
    """
    eval_file_path = os.path.join(data_dir, eval_file)  # 评估文件路径
    total_count = 0
    right_count = 0

    with codecs.open(eval_file_path, encoding='utf-8') as flist:
        lines = [line.strip() for line in flist]
        t1 = time.time()
        random.shuffle(lines) # 打乱样本

        i = 0
        for line in lines:
            i += 1
            if i > 3:
                break

            total_count += 1
            parts = line.strip().split()

            result = infer(parts[0])  # 执行推测

            img = Image.open(parts[0])
            plt.imshow(img)
            plt.show()
            plt.savefig("logs/%s" % parts[0])

            print("infer result:{0} answer:{1}".format(result,
parts[1]))

            if str(result) == parts[1]:
                right_count += 1

        period = time.time() - t1
        print("total eval count:{0} cost time:{1} predict accuracy:
{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()
```