

安徽大学人工智能学院

《机器学习程序设计》

实验案例设计报告

课程名称: 机器学习程序设计

专 业: 人工智能

班 级: 人工智能二班

学 号: WA2214014

姓 名: 杨跃浙

任课老师: 谭春雨

实验名称	基于深度学习的图片识别系统	实验次序	03
实验地点	笃行南楼 A104	实验日期	03.23
实验内容:			
<p>1. 业务背景分析</p> <p>图片识别技术称为 OCR (Optical Character Recognition) 即光学字符识别，它是计算机视觉领域中重要的研究分支，其较常见的应用是印刷体和手写体识别文本的识别。</p> <p>打印字体相对比较规整，但是在印刷过程中设备和纸张的原因导致的印刷质量问题，会对 OCR 模型的输入产生噪声影响。印刷样式、底纹背景和拍摄光线等也会对识别结果产生干扰。而手写体识别由于写字风格不一，没有明显的规范，其识别准确率相较于印刷体更低。</p> <p>1. 基于深度学习的文字识别是针对手写的数字、日期、字母、汉字等进行文本识别。 2. 通过计算机图形学对表格进行识别，并将单元格进行切分，对每个单元格的图片，通过采用深度学习算法实现文本识别。</p> <p>减少大量地特征提取和处理的工作，并能提高模型训练的准确率。</p> <p>某制造业企业需要在生产现场要填写较多的产品工艺流程数据，大部分数据以表格格式记录，现在需要对表格中记录的日期、数值、型号、备注等信息进行识别，并将识别结果存储于业务系统中。</p> <ul style="list-style-type: none"> ➢ 表格识别：识别是哪一类表格，然后采用 OpenCV 对不同类型的表格进行识别和裁剪，将表格单元格作为识别单位交给文本识别模型。 ➢ 训练数据生成：基于写手数字、字母、汉字通过变形、扭曲、合并等操作自动生成训练样本，并且使生成的训练样本尽可能贴合实际的写法和模式。此外，还要将生产过程中识别错误的样本放入训练集，进行模型调优。 ➢ 文本识别模型训练：采用密集连接卷积网络 (DenseNet) 对样本进行训练，其激活函数使用 CTC 作为评价指标。 ➢ 识别结果修正：通过语言模型和业务规则对识别结果进行修正，例如某一列为日期，则其为数值列，即不存在汉字或字母的可能性，且年、月、日等均存在一定的取值范围，通过施加这种限制提高最终的识别正确率。 <p>图片识别技术方案</p> <ul style="list-style-type: none"> ➢ 图片识别的基本思路是先做图片预处理，对图片进行校正和去噪，然后对图片进行切割，最后进行识别和修正。 			

- ✓ 表格模板检测与识别：不同表格的判断，通过对表格特征的学习来判断是哪一型号的表格，然后对其进行确认。
- ✓ 图片预处理：将图片进行二值化和降噪处理，并对其进行腐蚀和膨胀以突出显示横向和纵向的表格线，计算不同横线的平均倾斜角度，将表格进行旋转，转化为水平表格。
- ✓ 分割单元格：将图片中的表格全部定位出来，然后按单元格裁剪成一个个小图片，以便后续识别。
- ✓ 利用深度学习模型进行文本识别：采用 DenseNet 网络对训练样本进行学习，建立文字、数字、日期、字母及其混合的文本识别模型，并结合业务规则对表格结果进行纠正。
- ✓ 识别结果输出：模型采用 Python 中的 Flask 组件进行 WEB 接口封装，将图片文件上传、表格识别、文本识别等功能进行开放，通过 JSON 的形式输出给业务终端，供用户确认和修正。

2. 图片预处理

(1) 表格旋转

调用 OpenCV 的处理接口，可先通过 `pip3 install opencv-python` 安装相应的组件库。将表格图片进行旋转，使其成为水平方向显示的图片。

为了使横向的线条更加明显，使用 `cv2.dilate` 方法对其再次进行膨胀，其中 `iterations` 表示执行膨胀的次数，其值越大，膨胀的效果越强。

通过 `math.degrees` 方法计算霍夫曼直线检测到的每一条线段的倾斜角度，并累加求其均值 `median_angle`，然后使用 `ndimage.rotate` 将图像进行旋转，并将结果保存。整个纸张上的表格和标题整体进行了旋转，使其在水平方向上成为标准表格。这样在后续应用中，可以减少误差。

(2) 表格提取

为了对表格中的内容进行识别，需要先将表格提取出来，然后再进行单元格的提取，这里采用 OpenCV 中的轮廓检测(`findContours`)方法提取表格的轮廓信息，并选择最大的四边形轮廓作为表格，将其裁剪出来。

引入 OpenCV(`cv2`)、`imutils`、`matplotlib` 等 Python 库，使用 `imutils.resize` 对图像进行大小转换，使其高

度为 500 像素,宽度按宽高比自动适 应,并记录宽高比,用于后续对原图进行裁剪。并且将图像转化为单通道灰度图,并进行自适应阈值变换(cv2.adaptiveThreshold)对图 像进行简单化处理。

对于找到的表格,需要将其变换并进行裁剪,其中 four_point_transform 方法通过对 4 个点的位置坐标进行变换,找到其变换矩阵,从而产现裁剪。

order_points 方法是将 4 个坐标点按照左上、右上、右下、左下的顺序进行排列,即 tl、tr、br、bl 这个值的值,然后计算 4 个点包围的区 域最大宽度(maxWidth)和最大高度(maxHeight),并使 cv2.getPerspectiveTransform 实现从原区域坐标到目标区域坐标之间转换矩 阵 M 的构建,通过 cv2.warpPerspective 方法进行表格裁剪。

(3) 线条去除

对表格中的线条进行去除。

将原始的表格图片 resize 为高度是 1080 宽度自适应,将其转化为单通道的灰度图,并使用 cv2.threshold 简单化反转二值化处 理。

分别定义横向和纵向的核大小(kernel_length_horizontal,1)和(1,kernel_length_vertical),通过 cv2.getStructuringElement 方法提取核元素, 用于提取横行和纵向的线条,分别对图像进行腐蚀(cv2.erode)和膨胀(cv2.dilate),迭代次数过程中,纵向线迭代 4 次,横线 3 次。

将纵向表格线(vertical_lines_img)和横向表格线(horizontal_lines_img)进行加操作,构建蒙版层(mask_img),并使用 numpy 中的 np.bitwise_xor 方法与原始图进行与或操作,对表格线进行去除。

部分表格线未处理干净,存在较多细小的噪声线条,对其再次进行腐蚀和膨胀,核大小采用(2,2),以防对正常的数字和 文字产生影响,迭代次数均为 1 次。

代码:

```
import cv2
import imutils
import numpy as np
import math
from math import *
from scipy import ndimage
import matplotlib.pyplot as plt

def rotate_image(img_for_box_extraction_path):
```

```

image_height = 1080
image = cv2.imread(img_for_box_extraction_path)
img = imutils.resize(image, height=image_height)

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
(thresh, blur_gray) = cv2.threshold(gray, 128, 255, cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)

kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (30, 1))
morph_img = cv2.morphologyEx(blur_gray, cv2.MORPH_OPEN, kernel, (-1, -1))
cv2.imwrite('./Figure/linesDetected.jpg', morph_img)

kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3), (-1, -1))
lines_img = cv2.dilate(morph_img, kernel, iterations=1)
cv2.imwrite("./Figure/lines_dilated.jpg", lines_img)

low_threshold = 50
high_threshold = 150
edges = cv2.Canny(lines_img, low_threshold, high_threshold)

lines = cv2.HoughLinesP(edges, rho=1, theta=np.pi / 180, threshold=15, lines=np.array([]), minLineLength=50, maxLineGap=20)

angles = []
for line in lines:
    for x1, y1, x2, y2 in line:
        angle = math.degrees(math.atan2(y2 - y1, x2 - x1))
        angles.append(angle)

median_angle = np.median(angles)
img_rotated = ndimage.rotate(img, median_angle)
print("Angle is {}".format(median_angle))
cv2.imwrite('./Figure/rotated.jpg', img_rotated)
return img_rotated

def warp_image(image_height, image):
    orig = image.copy()
    ratio = image.shape[0] / float(image_height)
    image = imutils.resize(image, height=image_height)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gray = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY_INV, 11, 0)

    # 保存对比图: Original 和 gray
    plt.figure(figsize=(12, 6))
    plt.subplot(121)
    plt.imshow(cv2.cvtColor(orig, cv2.COLOR_BGR2RGB)) # matplotlib 显示 BGR 转 RGB
    plt.title('Original')
    plt.xticks([]), plt.yticks([])
    plt.subplot(122)
    plt.imshow(gray, cmap='gray')
    plt.title('gray')
    plt.xticks([]), plt.yticks([])
    plt.tight_layout()
    plt.savefig("./Figure/orig_gray_comparison.jpg")
    plt.close()

    major = cv2.__version__.split('.')[0]
    if major == '3':
        _, contours, hierarchy = cv2.findContours(gray, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
    else:
        contours, hierarchy = cv2.findContours(gray, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)

    contours = sorted(contours, key=cv2.contourArea, reverse=True)
    screen_cnt = None
    for c in contours:

```

```

epsilon = cv2.arcLength(c, True)
approx = cv2.approxPolyDP(c, 0.02 * epsilon, True)
area = cv2.contourArea(c)

if area < 2500:
    continue

if len(approx) == 4:
    screen_cnt = approx
    break

if screen_cnt is None:
    return -1, orig

warped = four_point_transform(orig, screen_cnt.reshape(4, 2) * ratio)

# 保存对比图: 处理后的 Original 和 warped
plt.figure(figsize=(12, 6))
vis_image = image.copy()
cv2.drawContours(vis_image, [screen_cnt], -1, (0, 255, 0), 2)
for point in screen_cnt.reshape(4, 2):
    cv2.circle(vis_image, (point[0], point[1]), 5, (0, 0, 255), 4)
plt.subplot(121)
plt.imshow(cv2.cvtColor(vis_image, cv2.COLOR_BGR2RGB))
plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122)
plt.imshow(cv2.cvtColor(warped, cv2.COLOR_BGR2RGB))
plt.title('Warped')
plt.xticks([]), plt.yticks([])
plt.tight_layout()
plt.savefig("./Figure/vis_warped_comparison.jpg")
plt.close()

cv2.imwrite("./Figure/warped.jpg", warped)
return warped

def four_point_transform(image, pts):
    rect = order_points(pts)
    (tl, tr, br, bl) = rect
    widthA = np.sqrt(((br[0] - bl[0]) ** 2) + ((br[1] - bl[1]) ** 2))
    widthB = np.sqrt(((tr[0] - tl[0]) ** 2) + ((tr[1] - tl[1]) ** 2))
    maxWidth = max(int(widthA), int(widthB))
    heightA = np.sqrt(((tr[0] - br[0]) ** 2) + ((tr[1] - br[1]) ** 2))
    heightB = np.sqrt(((tl[0] - bl[0]) ** 2) + ((tl[1] - bl[1]) ** 2))
    maxHeight = max(int(heightA), int(heightB))
    dst = np.array([
        [0, 0],
        [maxWidth - 1, 0],
        [maxWidth - 1, maxHeight - 1],
        [0, maxHeight - 1]], dtype="float32")
    M = cv2.getPerspectiveTransform(rect, dst)
    return cv2.warpPerspective(image, M, (maxWidth, maxHeight))

def order_points(pts):
    rect = np.zeros((4, 2), dtype="float32")
    s = pts.sum(axis=1)
    rect[0] = pts[np.argmin(s)]
    rect[2] = pts[np.argmax(s)]
    diff = np.diff(pts, axis=1)
    rect[1] = pts[np.argmin(diff)]
    rect[3] = pts[np.argmax(diff)]
    return rect

def remove_line(warped_image):
    img_bin = imutils.resize(warped_image, height=1080)

```

```

img_bin = cv2.cvtColor(img_bin, cv2.COLOR_BGR2GRAY)
(thresh, binary_src) = cv2.threshold(img_bin, 128, 255, cv2.THRESH_BINARY_INV
| cv2.THRESH_OTSU)
cv2.imwrite("./Figure/Image_bin_warp_invert.jpg", binary_src)
kernel_length_horizontal = np.array(binary_src).shape[1] // 100
kernel_length_vertical = np.array(binary_src).shape[0] // 30
verticle_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (1,
kernel_length_vertical))
hori_kernel = cv2.getStructuringElement(cv2.MORPH_RECT,
(kernel_length_horizontal, 1))
img_temp1 = cv2.erode(binary_src, verticle_kernel, iterations=4)
verticle_lines_img = cv2.dilate(img_temp1, verticle_kernel, iterations=4)
cv2.imwrite("./Figure/verticle_lines.jpg", verticle_lines_img)
img_temp2 = cv2.erode(binary_src, hori_kernel, iterations=3)
horizontal_lines_img = cv2.dilate(img_temp2, hori_kernel, iterations=3)
cv2.imwrite("./Figure/horizontal_lines.jpg", horizontal_lines_img)
mask_img = verticle_lines_img + horizontal_lines_img
binary_src = np.bitwise_xor(binary_src, mask_img)
cv2.imwrite("./Figure/no_border_image.jpg", binary_src)
clean_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (2, 2))
img_erode = cv2.erode(binary_src, clean_kernel, iterations=1)
binary_src = cv2.dilate(img_erode, clean_kernel, iterations=1)
cv2.imwrite("./Figure/no_border_image_clean.jpg", binary_src)

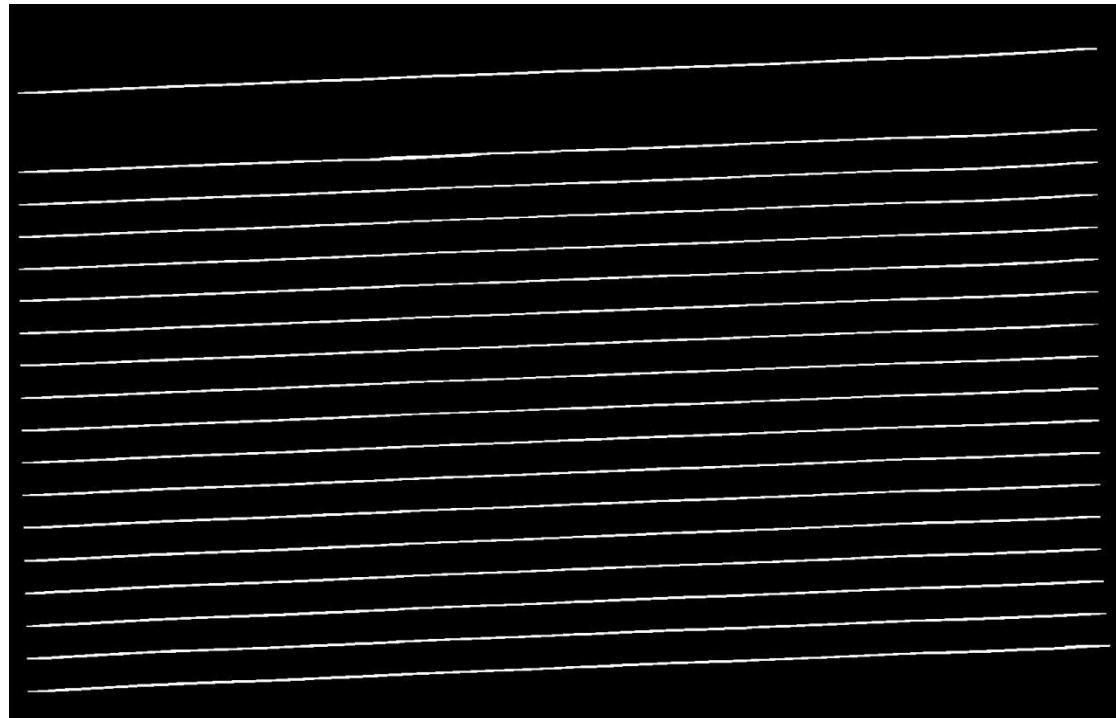
if __name__ == '__main__':
rotate_image(img_for_box_extraction_path='./Data/test.jpg')
warped_image = warp_image(image_height=1080,
image=cv2.imread('./Figure/rotated.jpg'))
remove_line(warped_image)

```

运行结果：

(1) 表格旋转

linesDetected.jpg





Angle:

The screenshot shows a Jupyter Notebook interface with the following details:

- File Bar:** Contains tabs for "test.jpg U", "preprocess.py U X", and "no_border_image.jpg U".
- Code Cell:** Displays Python code for image processing, specifically for extracting lines from an image. The code includes imports for cv2, imutils, numpy, math, scipy, and matplotlib.pyplot. It defines functions for rotating images, applying morphological operations like dilation and erosion, and detecting edges using the Canny algorithm. It then uses HoughLinesP to find lines in the image and stores the results in a variable named "anales".
- Output Cell:** Shows the output of the code execution, which includes a warning message about Angle is -2.1001376832314342.
- Bottom Navigation:** Includes buttons for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and COMMENTS. A Python icon indicates the current kernel.

rotated.jpg

综合生产过程指标																		
表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏
2022.07.10	2	55	9	0	6	17	48	196	141	119	36	18	21	7	30	30	6	77
2018.05.21	41	55	70	24	17	62	3	87	43	59	117	0	1	67	93	93	5	128
2021.03.17	12	35	110	93	22	106	136	65	59	39	53	93	61	84	15	15	17	152
2025.11.29	50	164	136	70	51	185	45	37	118	2	80	58	119	13	73	73	54	32
2021.12.01	28	50	12	186	104	58	121	0	7	5	2	92	0	79	81	81	28	4
2024.08.14	7	21	2	118	8	61	83	20	15	4	57	3	73	90	10	10	40	2
2022.02.17	37	3	38	100	38	9	13	0	68	91	48	0	9	13	70	70	18	7
2025.12.10	109	104	84	30	37	5	26	143	0	35	85	70	18	65	168	168	15	22
2026.02.07	59	12	78	43	58	50	81	17	47	112	70	4	5	13	120	120	44	44
2022.08.14	13	58	50	0	166	93	50	73	18	69	105	100	18	80	10	10	60	13
2025.06.14	41	17	185	44	64	92	62	63	56	138	48	52	17	53	64	64	8	43
2018.11.26	56	6	103	25	98	2	8	29	10	62	13	24	7	121	15	11	39	14
2026.12.31	18	22	52	49	46	5	20	74	0	2	50	6	53	2	6	60	98	113
2026.03.29	28	3	87	39	102	35	75	4	54	17	7	79	54	6	7	84	36	82
2026.02.13	155	5	1	54	39	29	60	78	70	31	89	19	110	22	86	4	40	8
2026.08.17	3	110	186	15	159	47	5	10	0	17	39	68	118	14	21	18	56	9

(2) 表格提取

orig_gray_comparison.jpg

综合生产过程指标																		
表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏
2022.07.10	2	55	9	0	6	17	48	196	141	119	36	18	21	7	30	30	6	77
2018.05.21	41	55	70	24	17	62	3	87	43	59	117	0	1	67	93	93	5	128
2021.03.17	12	35	110	93	22	106	136	65	59	39	53	93	61	84	15	15	17	152
2025.11.29	50	164	136	70	51	185	45	37	118	2	80	58	119	13	73	73	54	32
2021.12.01	28	59	12	186	106	58	21	0	7	5	2	94	0	79	81	81	28	4
2024.08.14	7	21	2	116	8	61	83	20	15	4	51	3	73	90	10	10	40	2
2022.02.17	37	3	38	100	38	9	13	0	68	91	48	0	9	13	70	70	18	7
2026.02.07	59	12	78	43	58	50	81	17	47	112	70	4	5	13	120	120	44	44
2022.08.14	13	58	50	0	166	93	50	73	18	69	105	100	18	80	10	10	60	13
2025.06.14	41	17	185	44	64	92	62	63	56	138	48	52	17	53	64	64	8	43
2018.11.26	56	6	103	25	98	2	8	29	10	62	13	24	7	121	15	11	39	14
2026.12.31	18	22	52	49	46	5	20	74	0	2	50	6	53	2	6	60	98	113
2026.03.29	28	3	87	39	102	35	75	4	54	17	7	79	54	6	7	84	36	82
2026.02.13	155	5	1	54	39	29	60	78	70	31	89	19	110	22	86	4	40	8
2026.08.17	3	110	186	15	159	47	5	10	0	17	39	68	118	14	21	18	56	9

综合生产过程指标																		
表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏
2022.07.10	2	55	55	17	48	196	141	119	36	18	21	7	30	30	6	77	77	77
2018.05.21	41	55	55	70	24	17	62	3	87	43	59	117	0	1	67	93	93	5
2021.03.17	12	35	110	93	22	106	136	65	59	39	53	93	61	84	15	15	17	152
2025.11.29	50	164	136	70	51	185	45	37	118	2	80	58	119	13	73	73	54	32
2021.12.01	28	59	12	186	106	58	21	0	7	5	2	92	0	79	81	81	28	4
2024.08.14	7	21	2	116	8	61	83	20	15	4	51	3	73	90	10	10	40	2
2022.02.17	37	3	38	100	38	9	13	0	68	91	48	0	9	13	70	70	18	7
2026.02.07	59	12	78	43	58	50	81	17	47	112	70	4	5	13	120	120	44	44
2022.08.14	13	58	50	0	166	93	50	73	18	69	105	100	18	80	10	10	60	13
2025.06.14	41	17	185	44	64	92	62	63	56	138	48	52	17	53	64	64	8	43
2018.11.26	56	6	103	25	98	2	8	29	10	62	13	24	7	121	15	11	39	14
2026.12.31	18	22	52	49	46	5	20	74	0	2	50	6	53	2	6	60	98	113
2026.03.29	28	3	87	39	102	35	75	4	54	17	7	79	54	6	7	84	36	82
2026.02.13	155	5	1	54	39	29	60	78	70	31	89	19	110	22	86	4	40	8
2026.08.17	3	110	186	15	159	47	5	10	0	17	39	68	118	14	21	18	56	9

vis_warped_comparison.jpg

Original																Warped															
表头已隐藏				表头已隐藏				表头已隐藏				表头已隐藏				表头已隐藏				表头已隐藏				表头已隐藏							
综合生产过程指标																															
2022.07.10	2	55	9	0	6	17	48	196	141	119	36	18	21	7	30	30	6	77	脸如马	展电											
2018.05.21	41	55	70	24	17	62	3	81	43	59	117	0	1	67	93	93	5	128	紧许	装关											
2021.03.17	12	35	110	93	22	106	136	65	59	39	53	48	34	15	15	17	152	忘菜	将影												
2025.11.29	50	164	136	70	51	185	45	37	18	2	80	58	19	13	73	73	54	32	书于夏	挥种架											
2021.01.01	24	59	12	106	58	121	0	7	5	2	49	0	79	81	28	6	游起黄	塞怀换													
2024.05.14	7	21	2	116	8	61	83	20	15	4	51	3	73	98	16	10	48	2	束小	弹支											
2021.01.17	3	35	100	36	9	13	0	68	91	48	0	9	13	70	16	7	处奈思	队特收													
2025.12.10	109	104	84	30	37	5	26	45	0	35	185	78	18	65	168	15	22	欢盛元	速育园												
2024.05.21	57	12	75	43	58	50	81	17	47	12	70	45	13	120	120	44	64	板拉便	买木原												
2022.08.14	13	58	58	0	166	93	50	73	78	18	105	100	15	80	10	10	60	13	卖饭	词准角											
2025.08.14	17	145	446	14	92	62	65	56	198	146	55	117	53	66	14	8	43	油馆	拉石												
2025.08.14	56	6	103	25	96	2	8	29	19	62	13	26	7	72	15	11	39	14	塞石	乐重要											
2024.11.21	18	22	52	45	46	5	20	14	0	2	59	6	53	2	6	68	13	13	发充发	职柔优											
2024.01.29	78	3	91	59	162	35	75	4	54	17	74	54	6	7	84	36	12	亚立拿	莱斯突												
2024.01.13	155	5	1	54	39	28	16	78	31	89	19	116	22	86	4	40	8	以毛	专社刑												
2020.06.17	3	110	186	15	169	47	5	10	0	17	39	68	118	14	21	18	56	9	社会进	做碧运											
2020.06.17	3	110	186	15	169	47	5	10	0	17	39	68	118	14	21	18	56	9	社会进	做碧运											

warped.jpg

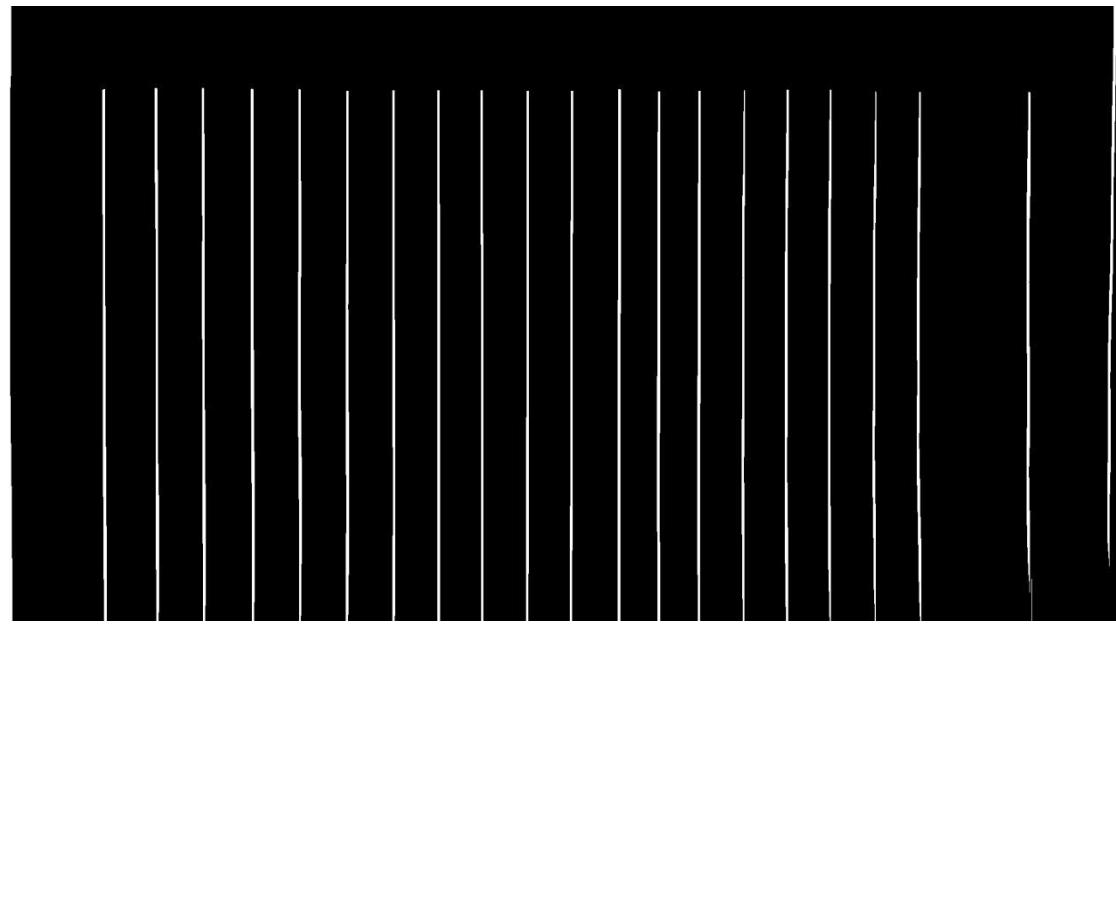
表头已隐藏				表头已隐藏				表头已隐藏				表头已隐藏				表头已隐藏				表头已隐藏			
2022.07.10	2	55	9	0	6	17	48	196	141	119	36	18	21	7	30	30	6	77	脸如马	展电			
2018.05.21	41	55	70	24	17	62	3	81	43	59	117	0	1	67	93	93	5	128	紧许	装关			
2021.03.17	12	35	110	93	22	106	136	65	59	39	53	48	34	15	15	17	152	忘菜	将影				
2025.11.29	50	164	136	70	51	185	45	37	18	2	80	58	119	13	73	73	54	32	书于夏	挥种架			
2021.12.01	28	50	12	186	104	58	121	0	7	5	2	92	0	79	81	28	4	游起黄	塞怀换				
2024.08.14	7	21	2	116	8	61	83	20	15	4	57	3	73	90	10	10	40	2	束小	弹支			
2022.02.17	37	3	38	100	38	9	18	0	68	91	48	0	9	13	70	70	18	7	处奈思	队特收			
2025.12.10	109	104	84	30	37	5	26	143	0	35	185	70	118	65	168	15	22	欢盛元	速育园				
2026.02.07	59	12	76	43	58	50	81	17	47	112	70	4	5	13	120	120	44	64	板拉便	买木原			
2022.08.14	13	58	50	0	166	93	50	73	18	69	105	100	18	80	10	10	60	13	卖饭	词准角			
2025.08.14	41	17	185	44	64	92	62	63	56	178	46	55	17	53	64	64	8	43	油馆	拉石			
2026.11.26	56	6	103	25	98	2	8	29	19	62	13	26	7	72	15	11	39	14	塞石	乐重要			
2026.12.31	18	22	52	45	46	5	20	14	0	2	59	6	53	2	6	60	98	13	发充发	职柔优			
2026.01.29	78	3	91	59	162	35	75	4	54	17	74	54	6	7	84	36	12	亚立拿	莱斯突				
2026.01.13	155	5	1	54	39	28	16	78	31	89	19	116	22	86	4	40	8	以毛	专社刑				
2020.06.17	3	110	186	15	159	47	5	10	0	17	39	68	118	14	21	18	56	9	社会进	做碧运			

(3) 线条去除

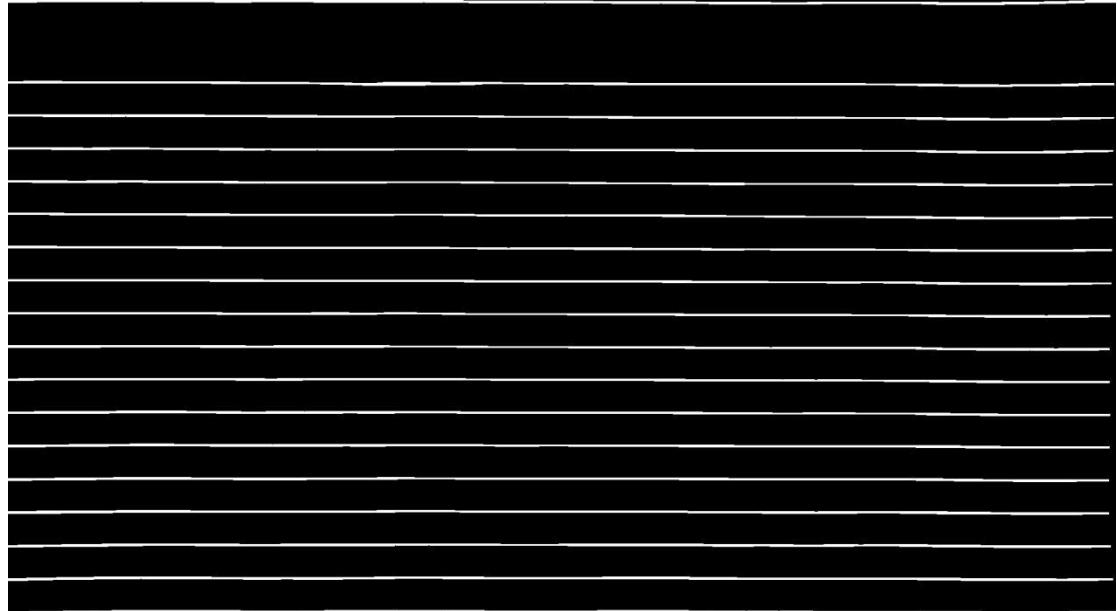
Image_bin_wrap_invert.jpg

表头已隐藏		表头已隐藏		表头已隐藏		表头已隐藏		表头已隐藏		表头已隐藏	
2022.07.10	2	55	9	0	6	17	43	196	141	119	36
2023.05.21	41	55	70	24	17	62	3	87	43	59	117
2022.03.17	12	35	110	93	22	106	136	65	59	39	53
2025.11.29	50	164	136	70	51	185	45	37	118	2	80
2027.12.01	28	50	12	186	104	58	121	0	7	5	2
2024.08.14	7	21	2	118	8	61	83	20	15	4	57
2022.02.17	37	3	38	100	38	9	13	0	68	91	48
2025.12.10	109	104	84	30	37	5	26	143	0	35	185
2026.02.07	59	12	13	43	58	50	81	17	47	112	70
2022.08.14	13	58	50	0	166	93	50	73	18	69	105
2025.08.14	41	17	185	44	64	92	62	63	56	138	48
2028.11.26	56	6	103	25	98	2	8	29	10	62	13
2026.12.31	18	22	52	49	46	5	20	74	0	2	50
2026.03.24	28	3	87	39	102	35	75	4	54	17	7
2026.02.19	155	5	1	54	39	29	60	78	70	31	89
2020.08.17	3	110	186	15	159	47	5	10	0	17	39

verticle_lines.jpg



horizontal_lines.jpg



no_border_image.jpg

表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏
2022.07.10 2 55 9 0 6 17 48 196 141 119 36 18 24 7 30 30 6 77 脸灼马 展电				
2013.05.21 41 55 70 24 17 62 3 87 43 59 117 0 1 67 93 93 5 128 紧件 装夹				
2021.03.17 12 35 110 93 22 106 136 65 59 39 53 93 61 84 15 15 17 152 忘来 搪刷				
2015.11.29 50 164 136 70 51 185 45 37 118 2 80 58 119 13 73 73 54 32 书千夏 挥种架				
2021.12.01 28 50 12 186 104 58 121 0 7 5 2 92 0 79 81 81 28 4 游起童 塞坏模				
2024.08.14 7 21 2 118 8 61 83 29 15 4 5 3 73 90 10 10 40 2 束小 弹吉				
2022.02.17 37 3 38 100 38 9 13 0 68 91 48 0 9 13 70 70 18 7 处京思 风精吸				
2015.12.10 109 104 84 30 37 5 26 143 0 35 185 70 118 65 163 168 15 22 欢盛元 演角丽				
2026.02.07 59 12 73 43 58 50 81 17 47 112 70 4 5 13 120 120 44 64 木板拉便 买木板				
2022.08.14 13 58 50 0 166 93 50 73 18 69 105 100 18 80 10 10 60 13 家银 词佳角				
2015.08.14.41 17 185 44 64 92 62 63 56 138 48 52 117 53 64 64 8 43 13 拉馆 拉石				
2018.11.26 56 6 103 25 98 2 8 29 10 62 13 24 7 121 15 11 39 14 塞石 众乘察				
2026.12.31 18 22 52 49 46 5 20 74 0 2 50 6 53 2 6 60 98 113 岁充发 聚景优				
2026.03.24 28 3 87 39 102 35 75 4 54 17 7 79 54 6 7 84 36 82 亚江拿 莱斯宾				
2026.02.13 155 5 1 54 39 29 60 78 70 31 89 19 110 22 86 4 40 80 以无 专社刊				
2020.08.17 3 110 186 15 159 47 5 10 0 17 39 68 118 14 21 18 156 91 社茶洪 做思云				

no_border_image_clean.jpg

表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	表头已隐藏	展电 装关 特制 挥种架 墨杯换 弹去 风特坚 病痛症 买木原 词佳角 拉石 众案察 聚录优 莱斯突 专社刊 做墨云
2022.07.10	2 55 9 0 6 17 48	196 141 119 36 18 21 7 30 30 6 177	脸如马		
2023.05.21	41 55 70 24 17 62 3 81 43 59 117 0 1 67 93 93 5 128	累计			
2023.03.17	12 35 110 93 22 106 136 65 59 39 53 93 61 84 15 15 17 152	忘矣			
2025.11.29	50 164 136 70 51 185 45 37 118 2 80 58 119 13 73 73 54 32	书千首			
2027.12.01	28 50 12 186 104 58 121 0 7 5 2 92 0 79 81 81 28 4	激起黄			
2024.08.14	7 21 2 118 8 61 83 20 15 4 5 7 3 73 90 10 10 40 2	素小			
2022.02.17	37 3 38 100 38 9 13 0 68 91 48 0 9 13 70 70 18 7	处寒思			
2025.12.10	109 104 84 30 37 5 26 143 0 35 185 70 118 65 163 168 15 22	欢度元			
2026.02.07	59 12 78 43 58 50 81 17 47 112 70 4 5 13 120 120 44 64	板起便			
2022.08.14	13 58 50 0 166 93 50 73 18 69 105 100 18 80 10 10 60 13	卖银			
2025.08.14	41 17 185 44 64 92 62 63 56 138 48 52 117 53 64 64 8 43	泡馆			
2018.11.26	56 6 103 25 98 2 8 29 10 62 13 24 7 121 15 11 39 14	墨在			
2026.11.31	18 22 52 49 46 5 20 74 0 2 50 6 53 2 6 60 98 113	岁充发			
2026.03.24	28 13 87 39 102 35 75 4 54 17 7 79 54 6 7 84 36 82	亚之库			
2026.02.13	155 5 1 54 39 29 60 78 70 31 89 19 110 22 86 4 40 80 以无				
2020.08.17	3 110 186 15 1159 47 5 10 0 17 39 68 118 114 21 18 1156 9 11	社会洪			
					做墨云

3. 基于密度卷积网络的文本识别

文本识别模型采用密集卷积网络(DenseNet)模型，并结合 CTC 损失函数进行训练，其中 DenseNet 是深度残差网络 (ResNet) 的特例。

CTC(Connectionist Temporal Classification)是计算一种损失值，用来解决输入序列和输出序列难以一一对应的问题。基于开源的 DenseNet 实现 OCR 识别。

每个块中的卷积层数都是 8 层，卷积核的初始数量是 8。

在多个卷积块之后叠加批标准化和 ReLu 激活函数，并使用 Permute 进行维度重排，使用 TimeDistributed 进行层封装，最后使用 softmax 作为全连接层的类别计算方法，将预测结果返回。通过使用 keras 中的 model.summary()方法将模型结果和参数信息输出。

代码：

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import (
    Dense, Dropout, Activation, Reshape, Permute, Conv2D, Conv2DTranspose,
    ZeroPadding2D, AveragePooling2D, GlobalAveragePooling2D, Input, Flatten,
    concatenate, BatchNormalization, TimeDistributed
)
from tensorflow.keras.regularizers import l2

def dense_cnn(input, nclass):
    _dropout_rate = 0.4
    _weight_decay = 1e-4
    _nb_filter = 64
    # conv 64 5*5 s=2
    x = Conv2D(_nb_filter, (5, 5), strides=(2, 2), kernel_initializer='he_normal',
               padding='same', use_bias=False, kernel_regularizer=l2(_weight_decay))(input)
    # 64 + 8 * 8 = 128
    x, _nb_filter = dense_block(x, 8, _nb_filter, 8, None, _weight_decay)
    # 128
    x, _nb_filter = transition_block(x, 128, _dropout_rate, 2, _weight_decay)
```

```

# 128 + 8 * 8 = 192
x, _nb_filter = dense_block(x, 8, _nb_filter, 8, None, _weight_decay)
# 192 -> 128
x, _nb_filter = transition_block(x, 128, _dropout_rate, 2, _weight_decay)

# 128 + 8 * 8 = 192
x, _nb_filter = dense_block(x, 8, _nb_filter, 8, None, _weight_decay)

x = BatchNormalization(axis=-1, epsilon=1.1e-5)(x)
x = Activation('relu')(x)

x = Permute((2, 1, 3), name='permute')(x)
x = TimeDistributed(Flatten(), name='flatten')(x)
y_pred = Dense(nclass, name='out', activation='softmax')(x)

return y_pred

def conv_block(input, growth_rate, dropout_rate=None, weight_decay=1e-4):
    x = BatchNormalization(axis=-1, epsilon=1.1e-5)(input)
    x = Activation('relu')(x)
    x = Conv2D(growth_rate, (3, 3), kernel_initializer='he_normal',
               padding='same')(x)
    if (dropout_rate):
        x = Dropout(dropout_rate)(x)
    return x

def dense_block(x, nb_layers, nb_filter, growth_rate, dropout_rate=0.4,
               weight_decay=1e-4):
    for i in range(nb_layers):
        cb = conv_block(x, growth_rate, dropout_rate, weight_decay)
        x = concatenate([x, cb], axis=-1)
        nb_filter += growth_rate
    return x, nb_filter

def transition_block(input, nb_filter, dropout_rate=None, pooltype=1,
                     weight_decay=1e-4):
    x = BatchNormalization(axis=-1, epsilon=1.1e-5)(input)
    x = Activation('relu')(x)
    x = Conv2D(nb_filter, (1, 1), kernel_initializer='he_normal', padding='same',
               use_bias=False, kernel_regularizer=l2(weight_decay))(x)
    if (dropout_rate):
        x = Dropout(dropout_rate)(x)
    if (pooltype == 2):
        x = AveragePooling2D((2, 2), strides=(2, 2))(x)
    elif (pooltype == 1):
        x = ZeroPadding2D(padding=(0, 1))(x)
        x = AveragePooling2D((2, 2), strides=(2, 1))(x)
    elif (pooltype == 3):
        x = AveragePooling2D((2, 2), strides=(2, 1))(x)
    return x, nb_filter

input_layer = Input(shape=(32, 200, 1)) # 适配 Dense-CNN 结构的输入尺寸
n_classes = 10

output_layer = dense_cnn(input_layer, n_classes)
model = Model(inputs=input_layer, outputs=output_layer)
model.summary()

```

运行结果：

The screenshot shows the PyCharm IDE interface. The code editor displays a Python script named `net_ctc.py` with code for defining a transition block and creating a DenseNet model. The terminal tab shows the execution results, including layer details and memory usage:

```
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS Python + □

batch_normalization_26 (BatchNormalization) (None, 4, 25, 192) 768 ['concatenate_23[0][0]']
activation_26 (Activation) (None, 4, 25, 192) 0 ['batch_normalization_26[0][0]']
permute (Permute) (None, 25, 4, 192) 0 ['activation_26[0][0]']
flatten (TimeDistributed) (None, 25, 768) 0 ['permute[0][0]']
out (Dense) (None, 25, 10) 7690 ['flatten[0][0]']

=====
Total params: 298122 (1.14 MB)
Trainable params: 290634 (1.11 MB)
Non-trainable params: 7488 (29.25 KB)
```

4. 训练 DenseNet 模型

训练样本准备好之后，利用如下代码进行 DenseNet 模型训练。构建模型的输入是图片的高度值，和训练样本的类别数量，即字典(lables)数字的数量。

`densenet.dense_cnn` 是利用前面 8.5 定义的 `densenet` 生成网络结构，使用 `ctc_lambda_func` 构建 CTC 损失函数。

模型构建完成之后，进入训练阶段。

代码：

```
import numpy as np
from PIL import Image
import tensorflow as tf
import os
from keras import backend as K
from keras.layers import Input, Dense, Flatten
from tensorflow.keras.layers import Reshape, Masking, Lambda, Permute
from keras.models import Model
from keras.callbacks import EarlyStopping, ModelCheckpoint,
LearningRateScheduler, TensorBoard
from imp import reload
import net_ctc as net

def get_model(img_h, nclass):
    input = Input(shape=(img_h, None, 1), name='the_input')
    y_pred = net.dense_cnn(input, nclass)

    labels = Input(name='the_labels', shape=[None], dtype='float32')
    input_length = Input(name='input_length', shape=[1], dtype='int64')
    label_length = Input(name='label_length', shape=[1], dtype='int64')
```

```

loss_out = Lambda(ctc_lambda_func, output_shape=(1,), name='ctc')([y_pred,
labels, input_length, label_length])

model = Model(inputs=[input, labels, input_length, label_length],
outputs=loss_out)
model.compile(loss={'ctc': lambda y_true, y_pred: y_pred}, optimizer='adam',
metrics=['accuracy'])
return model

def ctc_lambda_func(args):
y_pred, labels, input_length, label_length = args
return K.ctc_batch_cost(labels, y_pred, input_length, label_length)

def readfile(filename):
res = []
with open(filename, 'r') as f:
lines = f.readlines()
for i in lines:
res.append(i.strip())
dic = {}
for i in res:
p = i.split(' ')
dic[p[0]] = p[1]
return dic

class random_uniform_num():
"""均匀随机，确保每轮每个只出现一次"""
def __init__(self, total):
self.total = total
self.range = [i for i in range(total)]
np.random.shuffle(self.range)
self.index = 0

def get(self, batchsize):
r_n = []
if self.index + batchsize > self.total:
r_n_1 = self.range[self.index:self.total]
np.random.shuffle(self.range)
self.index = (self.index + batchsize) - self.total
r_n_2 = self.range[0:self.index]
r_n.extend(r_n_1)
r_n.extend(r_n_2)
else:
r_n = self.range[self.index : self.index + batchsize]
self.index = self.index + batchsize
return r_n

def gen(data_file, image_path, batchsize=128, maxlabellength=10, imagesize=(32,
280)):
image_label = readfile(data_file)
_imagefile = [i for i, j in image_label.items()]
x = np.zeros((batchsize, imagesize[0], imagesize[1], 1), dtype=np.float64)
labels = np.ones((batchsize, maxlabellength)) * 10000
input_length = np.zeros((batchsize, 1))
label_length = np.zeros((batchsize, 1))

r_n = random_uniform_num(len(_imagefile))
_imagefile = np.array(_imagefile)
while 1:
shufimagefile = _imagefile[r_n.get(batchsize)]
for i, j in enumerate(shufimagefile):
img1 = Image.open(os.path.join(image_path, j)).convert('L')
img = np.array(img1, 'f') / 255.0 - 0.5
x[i] = np.expand_dims(img, axis=2)
str_ = image_label[j]
label_length[i] = len(str_)

```

```

if len(str_) < 0:
    print("len < 0", j)
    input_length[i] = imagesize[1] // 8
    labels[i, :len(str_)] = [int(k) for k in str_]

inputs = {
    'the_input': x,
    'the_labels': labels,
    'input_length': input_length,
    'label_length': label_length,
}
outputs = {'ctc': np.zeros((batchsize,))}

yield (inputs, outputs)

if __name__ == '__main__':
    img_height = 32
    img_width = 200
    batch_size = 128
    char_set = open('./Data/labels.txt', 'r', encoding='utf-8').readlines()
    char_set = ''.join([ch.strip('\n') for ch in char_set[1:]])
    num_class = len(char_set)

    #K.set_session(get_session()) only in 1.X
    reload(net)
    model = get_model(img_height, num_class)

    train_set_file = './Data/data_3_train.txt'
    test_set_file = './Data/data_3_test.txt'
    train_loader = gen(train_set_file, './Data/train_imgs/', batchsize=batch_size,
    maxlabellength=num_class, imagesize=(img_height, img_width))
    test_loader = gen(test_set_file, './Data/test_imgs/', batchsize=batch_size,
    maxlabellength=num_class, imagesize=(img_height, img_width))

    checkpoint =
    ModelCheckpoint(filepath='./models/weights_densenet-{epoch:02d}-{val_loss:.2f}.h5', monitor='val_loss', save_best_only=True, save_weights_only=True)
    lr_schedule = lambda epoch: 0.0005 * 0.1**epoch
    learning_rate = np.array([lr_schedule(i) for i in range(9)])
    changlr = LearningRateScheduler(lambda epoch: float(learning_rate[epoch]))
    earlystop = EarlyStopping(monitor='val_loss', patience=10, verbose=1)

    train_num_lines = sum(1 for line in open(train_set_file))
    test_num_lines = sum(1 for line in open(test_set_file))

    model.fit_generator(train_loader,
    steps_per_epoch=train_num_lines // batch_size,
    epochs=20,
    initial_epoch=0,
    validation_data=test_loader,
    validation_steps=test_num_lines // batch_size,
    callbacks=[checkpoint, earlystop, changlr])

```

运行结果：

```

OPEN EDITORS
  test.jpg U  net.py 3, U  train.py 1, U  train_ctc.py 1, U
  Code > train_ctc.py > get_model
  1 import numpy as np
  2 from PIL import Image
  3 import tensorflow as tf
  4 import os
  5 from keras import backend as K
  6 from keras.layers import Input, Dense, Flatten
  7 from tensorflow.keras.layers import Reshape, Masking, Lambda, Permute
  8 from keras.callbacks import EarlyStopping, ModelCheckpoint, LearningRateScheduler, TensorBoard
  9 from img import reload
 10 import net_ctc as net
 11
 12 def get_model(img_h, nclass):
 13     input = Input(shape=(img_h, None, 1), name='the_input')
 14     y_pred = net.dense_cnn(input, nclass)
 15
 16     labels = Input(name='the_labels', shape=[None], dtype='float32')
 17     input_length = Input(name='input_length', shape=[1], dtype='int64')
 18     label_length = Input(name='label_length', shape=[1], dtype='int64')
 19
 20     loss_out = Lambda(ctc_lambda_func, output_shape=(1,), name='ctc')([y_pred, labels, input_length, label_length])
 21
 22     model = Model(inputs=[input, labels, input_length, label_length], outputs=loss_out)
 23     model.compile(loss="ctc", lambda y_true, y_pred: y_pred, optimizer='adam', metrics=['accuracy'])
 24
 25     return model
 26
 27 def ctc_lambda_func(args):
 28
 29 Epoch 1/20
 30 22/22 [=====] - 87s 4s/step - loss: 19.7313 - accuracy: 0.0000e+00 - val_loss: 9.9002 - val_accuracy: 0.0000e+00 - lr: 5.0000e-04
 31 Epoch 2/20
 32 22/22 [=====] - 95s 4s/step - loss: 5.5989 - accuracy: 0.0000e+00 - val_loss: 9.0387 - val_accuracy: 0.0000e+00 - lr: 5.0000e-05
 33 Epoch 3/20
 34 22/22 [=====] - 102s 5s/step - loss: 5.2687 - accuracy: 0.0000e+00 - val_loss: 8.8533 - val_accuracy: 0.0000e+00 - lr: 5.0000e-06
 35 Epoch 4/20
 36 22/22 [=====] - ETA: 1:32 - loss: 5.0886 - accuracy: 0.0000e+00

```

发现损失确实有所下降，但是 Acc 始终为 0，模型无法学习到有效特征。

在后面我又尝试了交叉熵损失，虽然这个损失不适用 OCR 识别，但是对于模型来说会更容易收敛

代码：

```

from tensorflow.keras.models import Model
from tensorflow.keras.layers import (
    Dense, Dropout, Activation, Reshape, Permute, Conv2D, Conv2DTranspose,
    ZeroPadding2D, AveragePooling2D, GlobalAveragePooling2D, Input, Flatten,
    concatenate, BatchNormalization, TimeDistributed
)
from tensorflow.keras.regularizers import l2

def dense_cnn(input, nclass):
    _dropout_rate = 0.4
    _weight_decay = 1e-4
    _nb_filter = 64

    x = Conv2D(_nb_filter, (5, 5), strides=(2, 2), kernel_initializer='he_normal',
               padding='same', use_bias=False, kernel_regularizer=l2(_weight_decay))(input)

    x, _nb_filter = dense_block(x, 8, _nb_filter, 8, None, _weight_decay)
    x, _nb_filter = transition_block(x, 128, _dropout_rate, 2, _weight_decay)

    x, _nb_filter = dense_block(x, 8, _nb_filter, 8, None, _weight_decay)
    x, _nb_filter = transition_block(x, 128, _dropout_rate, 2, _weight_decay)

    x, _nb_filter = dense_block(x, 8, _nb_filter, 8, None, _weight_decay)

    x = BatchNormalization(axis=-1, epsilon=1.1e-5)(x)
    x = Activation('relu')(x)

    x = GlobalAveragePooling2D()(x) # 变成 [batch_size, channels]
    y_pred = Dense(nclass, activation='softmax', name='out')(x) # 分类层

    return y_pred

def conv_block(input, growth_rate, dropout_rate=None, weight_decay=1e-4):
    x = BatchNormalization(axis=-1, epsilon=1.1e-5)(input)
    x = Activation('relu')(x)
    x = Conv2D(growth_rate, (3, 3), kernel_initializer='he_normal',
               padding='same')(x)
    if (dropout_rate):
        x = Dropout(dropout_rate)(x)

```

```

return x

def dense_block(x, nb_layers, nb_filter, growth_rate, dropout_rate=0.4,
weight_decay=1e-4):
for i in range(nb_layers):
cb = conv_block(x, growth_rate, dropout_rate, weight_decay)
x = concatenate([x, cb], axis=-1)
nb_filter += growth_rate
return x, nb_filter

def transition_block(input, nb_filter, dropout_rate=None, pooltype=1,
weight_decay=1e-4):
x = BatchNormalization(axis=-1, epsilon=1.1e-5)(input)
x = Activation('relu')(x)
x = Conv2D(nb_filter, (1, 1), kernel_initializer='he_normal', padding='same',
use_bias=False, kernel_regularizer=l2(weight_decay))(x)
if (dropout_rate):
x = Dropout(dropout_rate)(x)
if (pooltype == 2):
x = AveragePooling2D((2, 2), strides=(2, 2))(x)
elif (pooltype == 1):
x = ZeroPadding2D(padding=(0, 1))(x)
x = AveragePooling2D((2, 2), strides=(2, 1))(x)
elif (pooltype == 3):
x = AveragePooling2D((2, 2), strides=(2, 1))(x)
return x, nb_filter

input_layer = Input(shape=(32, 200, 1)) # 适配 Dense-CNN 结构的输入尺寸
n_classes = 10

output_layer = dense_cnn(input_layer, n_classes)
model = Model(inputs=input_layer, outputs=output_layer)
model.summary()

import numpy as np
from PIL import Image
import tensorflow as tf
import os
from keras import backend as K
from keras.layers import Input, Dense, Flatten
from tensorflow.keras.layers import Reshape, Masking, Lambda, Permute # type: ignore
from keras.models import Model
from keras.callbacks import EarlyStopping, ModelCheckpoint,
LearningRateScheduler, TensorBoard
from imp import reload
import net

from tensorflow.keras.losses import SparseCategoricalCrossentropy

def get_model(img_h, img_w, nclass):
input = Input(shape=(img_h, img_w, 1), name='the_input')
# CNN 提取特征
y_pred = net.dense_cnn(input, nclass)

# 交叉熵损失需要 softmax 输出
y_pred = tf.keras.layers.Dense(nclass, activation="softmax")(y_pred)
model = Model(inputs=input, outputs=y_pred)
model.compile(
loss=tf.keras.losses.SparseCategoricalCrossentropy(),

```

```

optimizer='adam',
metrics=['accuracy']
)
return model

def ctc_lambda_func(args):
y_pred, labels, input_length, label_length = args
return K.ctc_batch_cost(labels, y_pred, input_length, label_length)

def readfile(filename):
res = []
with open(filename, 'r') as f:
lines = f.readlines()
for i in lines:
res.append(i.strip())
dic = {}
for i in res:
p = i.split(' ')
dic[p[0]] = p[1]
return dic

class random_uniform_num():
"""均匀随机，确保每轮每个只出现一次"""
def __init__(self, total):
self.total = total
self.range = [i for i in range(total)]
np.random.shuffle(self.range)
self.index = 0

def get(self, batchsize):
r_n = []
if self.index + batchsize > self.total:
r_n_1 = self.range[self.index:self.total]
np.random.shuffle(self.range)
self.index = (self.index + batchsize) - self.total
r_n_2 = self.range[0:self.index]
r_n.extend(r_n_1)
r_n.extend(r_n_2)
else:
r_n = self.range[self.index : self.index + batchsize]
self.index = self.index + batchsize
return r_n

def gen(data_file, image_path, batchsize=128, imagesize=(32, 280)):
image_label = readfile(data_file)
_imagefile = list(image_label.keys())
x = np.zeros((batchsize, imagesize[0], imagesize[1], 1), dtype=np.float64)
labels = np.zeros((batchsize,), dtype=np.int32) # 只存单个类别索引

r_n = random_uniform_num(len(_imagefile))
_imagefile = np.array(_imagefile)
while True:
shufimagefile = _imagefile[r_n.get(batchsize)]
for i, j in enumerate(shufimagefile):
img1 = Image.open(os.path.join(image_path, j)).convert('L')
img = np.array(img1, 'f') / 255.0 - 0.5
x[i] = np.expand_dims(img, axis=2)
str_ = image_label[j]
labels[i] = int(str_) # 只取单个整数类别索引

yield x, labels

if __name__ == '__main__':
img_height = 32
img_width = 200
batch_size = 128
char_set = open('./Data/labels.txt', 'r', encoding='utf-8').readlines()

```

```

char_set = ''.join([ch.strip('\n') for ch in char_set[1:]])
num_class = len(char_set) + 1

#K.set_session(get_session()) only in 1.X
reload(net)
model = get_model(img_height, img_width, num_class)

train_set_file = './Data/data_3_train.txt'
test_set_file = './Data/data_3_test.txt'
train_loader = gen(train_set_file, './Data/train_imgs/', batchsize=batch_size,
imagesize=(img_height, img_width))
test_loader = gen(test_set_file, './Data/test_imgs/', batchsize=batch_size,
imagesize=(img_height, img_width))

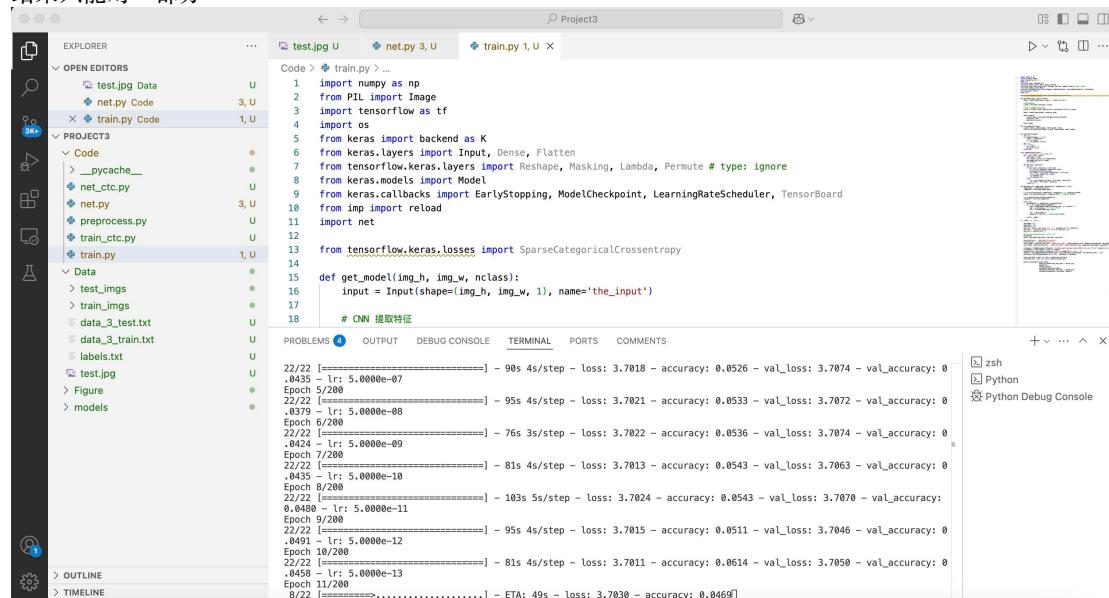
checkpoint =
ModelCheckpoint(filepath='./models/weights_densenet-{epoch:02d}-{val_loss:.2f}.h5', monitor='val_loss', save_best_only=True, save_weights_only=True)
lr_schedule = lambda epoch: 0.0005 * 0.1**epoch
learning_rate = np.array([lr_schedule(i) for i in range(200)]) # 适用于 20 轮
changlr = LearningRateScheduler(lambda epoch: float(learning_rate[min(epoch,
len(learning_rate) - 1)]))
earlystop = EarlyStopping(monitor='val_loss', patience=10, verbose=1)

train_num_lines = sum(1 for line in open(train_set_file))
test_num_lines = sum(1 for line in open(test_set_file))

model.fit_generator(train_loader,
steps_per_epoch=train_num_lines // batch_size,
epochs=200,
initial_epoch=0,
validation_data=test_loader,
validation_steps=test_num_lines // batch_size,
callbacks=[checkpoint, earlystop, changlr])

```

结果只能对一部分：



我认为可能是由于任务本身不容易，且训练样本数量较少（仅 3000 个）。

在使用 DenseNet 模型进行 OCR 识别任务时，实验中分别尝试了 CTC 损失和交叉熵损失两种训练方式，训练结果反映出不同的问题和挑战。首先，当采用 CTC 损失函数时，虽然训练过程中损失值呈现下降趋势，但准确率始终为 0，这表明模型未能有效学习到字符特征。这一现象可能与 CTC 损失的特性有关，该损失函数通过计算预测序列与标签序列的对齐概率来优化模型，对输入序列的长度和对齐方式较为敏感。在小规模数据集（仅 3000 个样本）的情况下，模型难以捕捉到复杂的序列模式，导致无法正确分类字符。此外，CTC 损失在处理多字符组合时需要更精确的标签对齐，而训练数据中可能存在的噪声或标注误差进一步加剧了模型的学习难度。

转而使用交叉熵损失函数后，模型能够识别部分字符，但整体准确率仍然较低。这一结果可能源于交叉熵损失的局限性，该损失函数将序列识别任务简化为单字符分类问题，忽略了字符之间的顺序关系，导致模型无法有效处理连续文本。例如，在表格单元格中常见的多位数字或混合字符场景下，交叉熵损失无法捕捉到字符间的依赖关系，从而降低了识别准确率。此外，实验中采用的 DenseNet 模型虽然具备较强的特征提取能力，但其结构设计更适合图像分类任务，而非序列识别。网络中全局平均池化层的使用可能导致空间信息丢失，进一步影响了对连续字符的定位和识别。

数据层面的问题同样显著。训练样本数量不足（3000 个）限制了模型的泛化能力，尤其在处理字体、大小、倾斜角度等多样化变化时，模型无法充分学习到各种情况下的字符特征。此外，数据增强策略的缺失可能导致模型过拟合，无法适应实际应用中复杂的图像背景和噪声干扰。例如，表格图像中常见的线条残留、光照不均等问题，可能在训练数据中未得到充分模拟，导致模型在真实场景下表现不佳。

综合来看，实验中遇到的问题是多因素共同作用的结果。模型结构与任务类型的不匹配、损失函数的选择与数据特征的不适应，以及数据量和多样性的不足，共同导致了识别效果的不理想。未来的改进方向应聚焦于优化模型架构（如引入循环神经网络处理序列信息）、采用更适合的损失函数（如结合 CTC 与注意力机制），以及通过数据增强和迁移学习等方法扩大有效训练数据量。此外，针对制造业表格的特定场景，可进一步优化预处理流程，提高图像质量，并结合业务规则进行后处理，从而提升整体识别准确率。

5. 总结

我在本次基于深度学习的图片识别系统开发中，深入探索了表格预处理与文本识别的关键技术环节。实验目标是针对制造业表格数据设计 OCR 系统，通过预处理流程解决表格倾斜、背景干扰等问题，再利用 DenseNet 模型实现文本识别。

在图片预处理阶段，我通过 OpenCV 的霍夫直线检测和透视变换成功实现了表格的旋转校正与精准裁剪。具体来说，通过计算倾斜角度的中位数并进行图像旋转，将歪斜的表格调整为水平状态，同时利用轮廓检测和四边形拟合提取出完整的表格区域。在去除表格线时，采用形态学操作结合二值化处理，有效分离了文本与线条，生成了干净的单元格图像。这些步骤为后续识别提供了高质量的输入数据，实验结果显示预处理后的图像清晰保留了文本信息，验证了流程的有效性。

然而在文本识别环节，我遇到了显著挑战。使用 DenseNet 结合 CTC 损失函数进行训练时，虽然损失值从初始的 19.73 逐步下降至 5.27，但准确率始终为 0。这表明模型未能捕捉到字符特征，可能是因为 CTC 损失对序列对齐的严格要求与小规模数据集（仅 3000 样本）之间的矛盾。随后尝试的交叉熵损失虽然使模型收敛，准确率达到 5%-6%，但依然无法满足实际需求。分析认为，DenseNet 的全局平均池化设计弱化了空间信息，而交叉熵损失将序列问题简化为单字符分类，导致模型无法处理连续文本的上下文关系。

数据层面的局限性同样显著。训练样本仅覆盖有限字体和书写风格，且缺乏有效的数据增强策略，使得模型泛化能力不足。例如，表格中的数字与字母混合场景、不同光照条件下的图像变化等复杂情况未被充分覆盖，导致模型在真实场景下表现不佳。

总结而言，本次实验成功构建了完整的表格预处理流水线，但文本识别效果受限于模型结构、损

失函数选择及数据规模。未来需从以下方向改进：引入 CRNN 等序列模型强化上下文建模，结合注意力机制优化 CTC 损失；通过生成对抗网络（GAN）合成多样化训练样本，增强数据多样性；探索迁移学习策略，利用预训练模型初始化网络参数。此外，针对制造业场景的业务规则进行后处理，可进一步提升识别准确率。

思考题：

1. 在图像分类中，常用的图像样本数据增强方法有哪些？

在图像分类任务中，常用的图像样本数据增强方法旨在通过对原始数据进行变换生成多样化样本，从而提升模型泛化能力。常见的增强技术包括几何变换（如旋转、翻转、缩放、平移、裁剪），颜色空间调整（亮度、对比度、饱和度变化），噪声添加（高斯噪声、椒盐噪声），以及更复杂的操作如弹性变形、随机擦除、生成对抗网络（GAN）合成样本等。例如，旋转和翻转可以模拟不同视角的拍摄场景，而裁剪和缩放有助于处理物体在图像中的位置和大小变化。颜色增强则能让模型适应不同光照条件下的色彩偏差。这些方法通过扩大训练数据的分布范围，迫使模型学习更鲁棒的特征表示，避免过拟合。

2. 总结 OpenCV 在计算机视觉方面的常用功能。

OpenCV 作为计算机视觉领域的重要工具库，提供了丰富的功能支持。其核心功能涵盖图像读取与保存、基本处理（如灰度转换、二值化、滤波）、几何变换（旋转、缩放、透视变换）、特征检测与匹配（SIFT、SURF、ORB）、目标检测（基于 Haar 级联或 HOG 特征的分类器）、图像分割（阈值处理、轮廓检测）、视频分析（光流法、背景建模）以及与深度学习框架的集成。例如，在本次实验中，OpenCV 被用于表格的旋转校正、轮廓检测和形态学操作去除表格线。其强大的矩阵运算能力和优化后的算法实现，使其成为快速原型开发和实际部署的首选工具。

3. DenseNet 比一般的残差网络有什么优势？

DenseNet 相比一般的残差网络（如 ResNet）具有显著优势，其核心在于密集连接机制。传统残差网络通过跳跃连接缓解梯度消失问题，而 DenseNet 进一步将每一层与后续所有层直接连接，每一层的输入包含之前所有层的特征图。这种结构促进了特征的重用，减少了参数冗余，同时增强了梯度流的传播效率。例如，DenseNet 的密集块内部通过特征拼接方式传递信息，使得浅层特征能够直接参与深层决策，避免了特征丢失。此外，由于特征复用，DenseNet 的参数数量通常比同深度的 ResNet 更少，且在训练中表现出更快的收敛速度和更好的泛化性能。这种设计特别适合需要高效利用计算资源的场景，同时保持了对复杂模式的建模能力。