



课程报告

年 级 专 业： 2020 级智能科学与技术

学 生 姓 名： 孙成

学 号： 20203101694

课 程 名 称： 机器学习

任 课 老 师： 胡祝华教授

平时成绩	
报告成绩	
总评成绩	
任课教师签名	

海南大学·信息与通信工程学院

School of Information and Communication Engineering, Hainan University

车牌识别与定位

一、摘要

在城市交通管理、视频监控、车辆识别和停车场管理中车牌识别是一项富有挑战而重要的任务，图像的清晰度、光照条件、天气因素、图像形变和车牌字符的多变性使车牌识别问题变的比较复杂。一个鲁棒的车牌识别系统应当能够应对各种环境的变换而不失准确性。LPRNet 没有对字符进行预分割，可以说是一个端到端的车牌识别算法，为什么说是“可以说”呢？。因为文章没有考虑车牌检测问题，车牌检测通过 LBP-cascade 获得。卷积神经网络在图像分类、目标检测和语义分割等计算机视觉任务中凸显了有效性和优越性，很多 LPR 车牌卷识别算法基于积神经网络设计，LPRNet 也是如此。不同于其他车牌识别算法，LPRNet 瞄准嵌入式设备。

二、针对研究内容采取的具体的技术方案

2.1 具体技术路线

基于 pytorch 深度学习框架，实用开源模型 yolov4 实现模板检测与 yolov5 实现车牌检测与 LPRNet 实现车牌检测

2.2.算法原理

LPRNet 是一种非常高效的神经网络，它只需要 0.34 GFLOps 就可以进行一次前向传播。此外，模型在保证实时性的同时，准确率也有保证。面对中文车牌，LPRNet 可进行端到端的训练。LPRNet 过程就是一个分类过程，输入裁剪后的车牌图片就得出车牌信息对应的数值。

2.3.代码及注释

```
import argparse
from yolo import YOLO
import torch.backends.cudnn as cudnn
from models.experimental import *
from package_car.datasets import *
from package_car.utils import *
from models.LPRNet import *

def detect(save_img=False):
    yolo = YOLO()
    out, source, weights, view_img, save_txt, imgsz = \
        opt.output, opt.source, opt.weights, opt.view_img,
    opt.save_txt, opt.img_size
    webcam = source == '0' or source.startswith('rtsp') or
    source.startswith('http') or source.endswith('.txt')
```

```

# 初始化
device = torch_utils.select_device(opt.device)
if os.path.exists(out):
    shutil.rmtree(out) # 删除输出文件夹
os.makedirs(out) # 创建新输出文件夹
half = device.type != 'cpu'

# 载入模型
model = attempt_load(weights, map_location=device) # 载入模型 FP32
imgsz = check_img_size(imgsz, s=model.stride.max()) # 检查
img_size

if half:
    model.half() # to FP16

# 第二级分级机
classify = True
if classify:
    modelc = LPRNet(lpr_max_len=8, phase=False,
class_num=len(CHARS), dropout_rate=0).to(device)

modelc.load_state_dict(torch.load('./weights/Final_LPRNet_model.pth',
map_location=torch.device('cpu'))))
print("load pretrained model successful!")
modelc.to(device).eval()

# 数据加载器
vid_path, vid_writer = None, None
if webcam:
    view_img = True
    cudnn.benchmark = True # 设置为 True 以加快恒定图像大小推断
    dataset = LoadStreams(source, img_size=imgsz)
else:
    save_img = True
    dataset = LoadImages(source, img_size=imgsz)

# 获取名字和颜色
names = model.module.names if hasattr(model, 'module') else
model.names
colors = [[random.randint(0, 255) for _ in range(3)] for _ in
range(len(names))]

# 运行预测
t0 = time.time()
img = torch.zeros((1, 3, imgsz, imgsz), device=device) # init img

```

```

    _ = model(img.half() if half else img) if device.type != 'cpu'
else None # run once
for path, img, im0s, vid_cap in dataset:
    img = torch.from_numpy(img).to(device)
    img = img.half() if half else img.float() # uint8 to fp16/32
    img /= 255.0 # 0 - 255 to 0.0 - 1.0
    if img.ndimension() == 3:
        img = img.unsqueeze(0)

    # 预测
    t1 = torch_utils.time_synchronized()
    pred = model(img, augment=opt.augment)[0]
    print(pred.shape)
    # 应用NMS
    pred = non_max_suppression(pred, opt.conf_thres,
opt.iou_thres, classes=opt.classes, agnostic=opt.agnostic_nms)
    t2 = torch_utils.time_synchronized()

    # 应用程序分类器
    if classify:
        pred, plat_num = apply_classifier(pred, modelc, img, im0s)

    # 过程检测
    for i, det in enumerate(pred): # 每个图像的检测数
        if webcam: # batch_size >= 1
            p, s, im0 = path[i], '%g: ' % i, im0s[i].copy()
        else:
            p, s, im0 = path, '', im0s

        save_path = str(Path(out) / Path(p).name)
        txt_path = str(Path(out) / Path(p).stem) + ('_%g' %
dataset.frame if dataset.mode == 'video' else '')
        s += '%gx%g ' % img.shape[2:] # 打印字符串
        gn = torch.tensor(im0.shape)[[1, 0, 1, 0]] # whwh 标准化增益
        if det is not None and len(det):
            # 将框从 img_size 重缩放为 im0 大小
            det[:, :4] = scale_coords(img.shape[2:], det[:, :4],
im0.shape).round()

            # 打印结果
            for c in det[:, 5].unique():
                n = (det[:, 5] == c).sum() # 每类检测数
                s += '%g %ss, ' % (n, names[int(c)]) # 添加到字符串

```

```

# 写入结果
for de, lic_plat in zip(det, plat_num):
    # xyxy, conf, cls, lic_plat=de[:4], de[4], de[5], de[6:]
    *xyxy, conf, cls=de

    if save_txt: # 写入文件
        xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4))
/ gn).view(-1).tolist() # normalized xywh
        with open(txt_path + '.txt', 'a') as f:
            f.write(('g ' * 5 + '\n') % (cls, xywh)) #
label format

    if save_img or view_img: # Add bbox to image
        # label = '%s %.2f' % (names[int(cls)], conf)
        lb = ""
        for a, i in enumerate(lic_plat):
            # if a == 0:
            #     continue
            lb += CHARS[int(i)]
            label = '%s %.2f' % (lb, conf)
            im0=plot_one_box(xyxy, im0, label=label,
color=colors[int(cls)], line_thickness=3)
            # print(type(im0))
            img_pil = Image.fromarray(im0) # ndarray 转化为图片
            im0 = yolo.detect_image(img_pil) # 图片才能检测
# Print time (inference + NMS)
# print('%sDone. (%.3fs)' % (s, t2 - t1))#不打印东西

# Stream results
if view_img:
    cv2.imshow(p, im0)
    if cv2.waitKey(1) == ord('q'): # q to quit
        raise StopIteration

# Save results (image with detections)
if save_img:
    if dataset.mode == 'images':
        im0 = np.array(im0) # 图片转化为 ndarray
        cv2.imwrite(save_path, im0) # 这个地方的 im0 必须为
ndarray

    else:
        if vid_path != save_path: # new video

```

```

        vid_path = save_path
        if isinstance(vid_writer, cv2.VideoWriter):
            vid_writer.release() # release previous video
writer

        fourcc = 'mp4v' # output video codec
        fps = vid_cap.get(cv2.CAP_PROP_FPS)
        w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
        h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
        vid_writer = cv2.VideoWriter(save_path,
cv2.VideoWriter_fourcc(*fourcc), fps, (w, h))
        im0 = np.array(im0) # 图片转化为 ndarray#JMW 添加
        vid_writer.write(im0)

    if save_txt or save_img:
        # print('Results saved to %s' % os.getcwd() + os.sep + out)
        if platform == 'darwin': # MacOS
            os.system('open ' + save_path)

    # print('Done. (%.3fs)' % (time.time() - t0))#不打印一堆东西

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+', type=str,
default='./weights/last.pt', help='model.pt path(s)')
    parser.add_argument('--source', type=str,
default='./inference/images/', help='source') # file/folder, 0 for
webcam
    parser.add_argument('--output', type=str,
default='inference/output', help='output folder') # output folder
    parser.add_argument('--img-size', type=int, default=640,
help='inference size (pixels)')
    parser.add_argument('--conf-thres', type=float, default=0.4,
help='object confidence threshold')
    parser.add_argument('--iou-thres', type=float, default=0.5,
help='IOU threshold for NMS')
    parser.add_argument('--device', default='', help='cuda device,
i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument('--view-img', action='store_true',
help='display results')
    parser.add_argument('--save-txt', action='store_true', help='save

```

```

results to *.txt')
    parser.add_argument('--classes', nargs='+', type=int,
help='filter by class')
    parser.add_argument('--agnostic-nms', action='store_true',
help='class-agnostic NMS')
    parser.add_argument('--augment', action='store_true',
help='augmented inference')
    parser.add_argument('--update', action='store_true', help='update
all models')
    opt = parser.parse_args()
    # print(opt)

    with torch.no_grad():
        if opt.update: # 更新所有模型 (to fix SourceChangeWarning)
            for opt.weights in ['yolov5s.pt', 'yolov5m.pt',
'yolov5l.pt', 'yolov5x.pt', 'yolov3-spp.pt']:
                detect()
                create_pretrained(opt.weights, opt.weights)
        else:
            detect()

```

三、实验及结果分析

3.1 实验数据

我们的训练数据，以及 yolo4 的权重数据存放在 ‘model_data’ 下面：

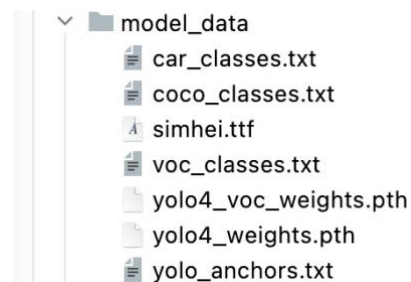


图 1 实验数据

3.2 实验环境

MacBook Pro (macOS Monterey12.6)
PyCharm 2021.3.2 (Professional Edition)
Build #PY-213.6777.50, built on January 27, 2022
Licensed to 成 孙
订阅有效期至 2023 年 10 月 11 日。
For educational use only.
Runtime version: 11.0.13+7-b1751.25 x86_64
VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o.
macOS 12.6
GC: G1 Young Generation, G1 Old Generation

Memory: 2048M

Cores: 12

3.3 实验结果

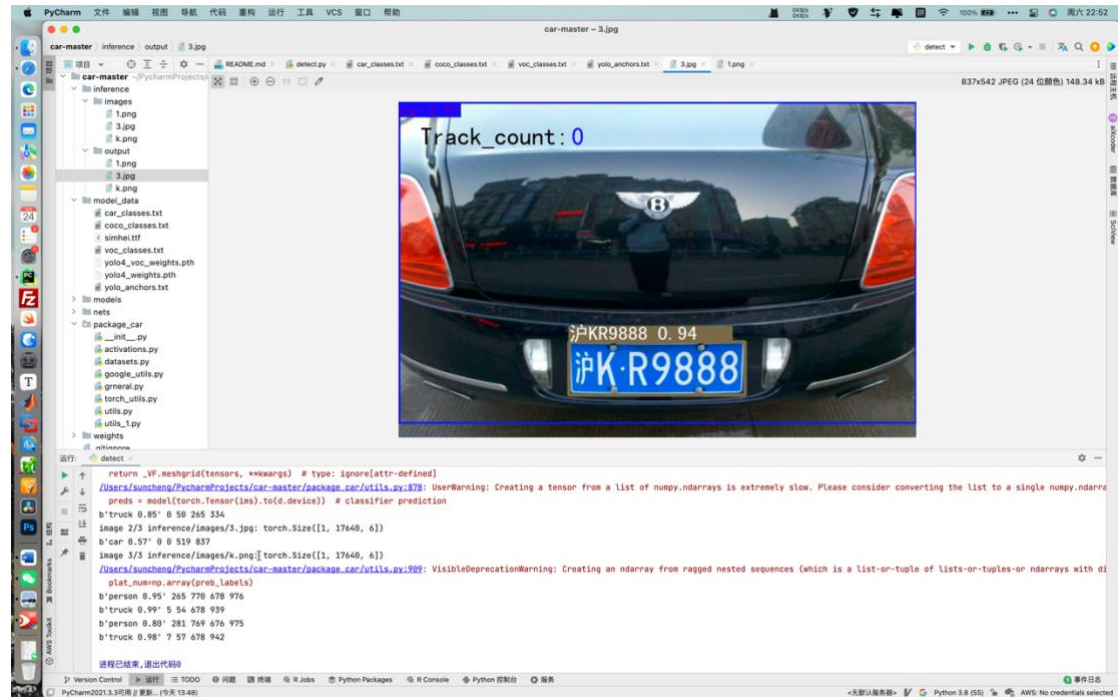


图 2 结果图一

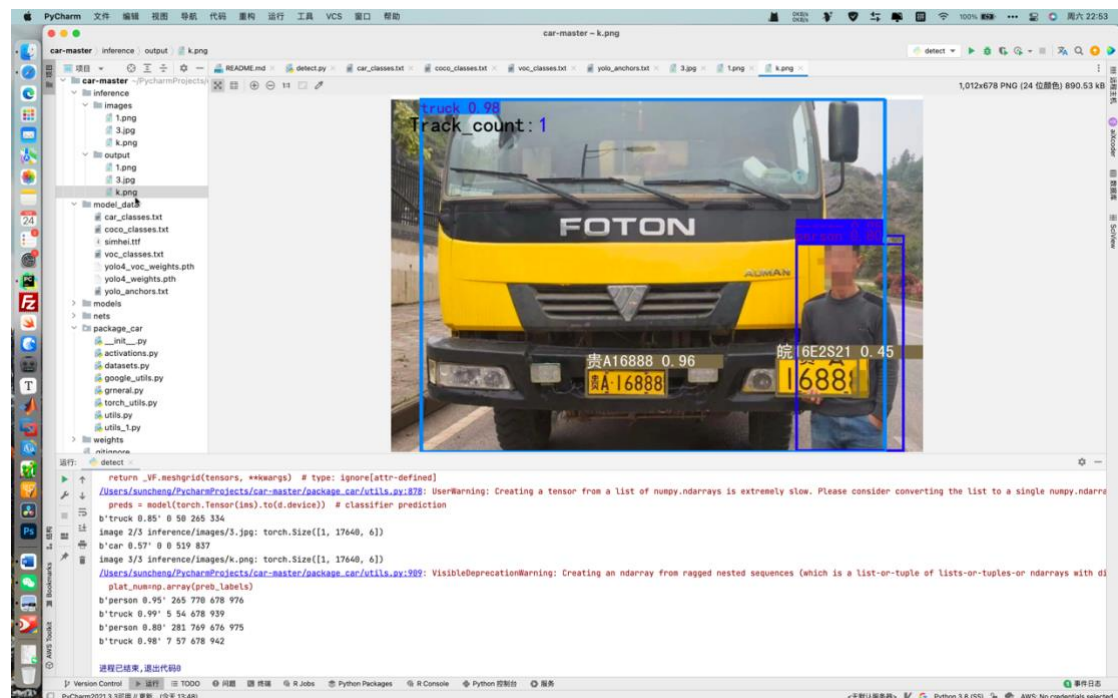


图 3 结果图二

3.4 实验分析

由实验结果分析可知，对于车牌的定位能够达到很高的精度，当时对于有遮挡的车牌识别正确率还有待提升。

四、总结和下一步的展望

车牌识别是指使用计算机视觉技术，通过对车牌图像的分析 and 处理，识别出车牌上的字符信息。这是一个广泛应用于交通管理、智能停车场、公路收费站等领域的重要技术。

在车牌识别项目中，首先需要进行图像预处理，即对车牌图像进行裁剪、灰度化、去噪等操作，以便为后续识别做好准备。然后，可以使用形态学处理、阈值处理等方法来分离出车牌中的字符。最后，可以使用训练好的字符识别模型对分离出的字符进行识别，得到完整的车牌号码。

展望未来，车牌识别技术可能会继续发展，提高准确率和实时性，同时也可能会拓展到更多的应用场景。例如，可能会开发出能够识别多种不同车牌格式的系统，或者在汽车行业中使用车牌识别技术来帮助管理车辆信息。同时，随着计算机视觉技术的不断提升，车牌识别系统可能会变得更加智能，能够自动学习和改进。

五、自我评价

车牌识别项目是一个比较复杂的计算机视觉任务，需要综合运用许多不同的技术才能实现。在进行车牌识别项目时，需要考虑到车牌图像的各种变化，例如角度、遮挡、光照、噪声等，并综合使用各种图像处理技术来应对这些变化。

在进行车牌识别项目时，需要认真规划并制定合理的步骤，并结合实际情况选择合适的算法和技术。在项目进行过程中，还需要不断测试和调试，以提高系统的准确率和可靠性。

总的来说，车牌识别项目是一个具有挑战性的任务，但是通过学习，已然将困难克服，完成了既定任务

房价预测

一、摘要

人们希望能够对房价进行较为准确的预测。房价变动不仅受时间、区域等因素影响，而且与房屋年限、附近地理条件、人文、交通等因素存在一定联系，其中涉及很多随机影响因素，故无法使用 Logistic 回归模型等简单模型进行有效预测，通常使用多元回归模型、神经网络模型、随机森林模型等具有较强拟合能力的机器学习模型作为预测分析工具。

二、针对研究内容采取的具体技术方案

2.1 具体技术路线

先将数据缺失值处理（空值即为没有特征的数据类型），然后增加数据特征，进行偏态数据处理，剔除离散数据和没有相关属性占多数的特征。使用 box-cos 进行偏态数据处理，然后进行建模预测。

2.2.算法原理

Box-Cox 变换是 Box 和 Cox 在 1964 年提出的一种广义幂变换方法，是统计建模中常用的一种数据变换，用于连续的响应变量不满足正态分布的情况。Box-Cox 变换之后，可以一定程度上减小不可观测的误差和预测变量的相关性。Box-Cox 变换的主要特点是引入一个参数，通过数据本身估计该参数进而确定应采取的数据变换形式，Box-Cox 变换可以明显地改善数据的正态性、对称性和方差相等性，对许多实际数据都是行之有效的

2.3.代码及注释

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

def get_data():
    train_df = pd.read_csv("train.csv", index_col='Id')
    test_df = pd.read_csv("test.csv", index_col='Id')
    return train_df, test_df

train_df, test_df = get_data()
data = pd.concat([train_df, test_df], axis=0)

def null_count(data):
    """
    缺失值统计
    """
    null_data = data.isnull().sum()
    # print(type(null_data))
    null_data = null_data[null_data>0].sort_values(ascending=False)
    return null_data

null_count(data)
```

进一步检查空值即为没有特征的数据类型

```
not_exists = [
    "PoolQC",
    "MiscFeature",
    "Alley" ,
    "Fence" ,
    "FireplaceQu",

    "GarageType",
    "GarageYrBlt",
    "GarageFinish",
    "GarageQual",
    "GarageCond",

    "BsmtExposure",
    "BsmtFinType2",
    "BsmtFinType1",
    "BsmtCond",
    "BsmtQual",
    "MasVnrArea",
    "MasVnrType"
]

for col in not_exists:
    print(col, train_df[col].dtype)
# 除了 GarageYrBlt, MasVnrArea 是 float, 其余都是 object, 将其填充为 "None",
def null_fill(df, columns, fill_value):
    """
    填充空值
    """
    result = df.copy(deep=True)
    for col in columns:
        result[col].fillna(fill_value, inplace=True)

    return result

fill_none_cols = [i for i in not_exists if i not in ["GarageYrBlt",
"MasVnrArea"]]
data = null_fill(data, fill_none_cols, 'None')

null_count(data)
# 没有车库的建造年份填充为 0
# 没有墙砖的墙砖面积填充为 0
data["GarageYrBlt"].fillna(0, inplace=True)
data["MasVnrArea"].fillna(0, inplace=True)
```

```
# MSZoning          4  出售的房屋空间的分区分级
# Utilities          2  表示房子里提供的水电设施，空值可能表示没有任何水电气设施
```

```
# 表示地下室全部作为浴室还是一半作为浴室，缺失值可能是因为无地下室
```

```
# BsmtFullBath      2
```

```
# BsmtHalfBath      2
```

```
# 表示地下室某种面积
```

```
# BsmtFinSF1        1
```

```
# BsmtFinSF2        1
```

```
# BsmtUnfSF         1
```

```
# TotalBsmtSF       1
```

```
# Functional        2  表示房子的实用性 空值表示 typical
```

```
# Exterior1st       1  房子外观
```

```
# Exterior2nd       1
```

```
# GarageCars        1
```

```
# GarageArea        1
```

```
# KitchenQual       1
```

```
# KitchenQual: Kitchen quality
```

```
#      Ex  Excellent
```

```
#      Gd   Good
```

```
#      TA  Typical/Average
```

```
#      Fa   Fair
```

```
#      Po   Poor
```

```
# SaleType          1
```

```
# SaleType: Type of sale
```

```
#      WD  Warranty Deed - Conventional
```

```
#      CWD Warranty Deed - Cash
```

```
#      VWD Warranty Deed - VA Loan
```

```
#      New  Home just constructed and sold
```

```
#      COD  Court Officer Deed/Estate
```

```
#      Con  Contract 15% Down payment regular terms
```

```
#      ConLw Contract Low Down payment and low interest
```

```
#      ConL  Contract Low Interest
```

```
#      ConLD Contract Low Down
```

```
#      Oth  Other
```

```
# SaleType, KitchenQual ,MSZoning 用众数填充
```

```
def fill_na_mode(df, cols):
```

```

"""
用众数填充空值
"""
for col in cols:
    df[col].fillna(df[col].mode()[0], inplace=True)
fill_na_mode(data, ["SaleType", "KitchenQual", "MSZoning"])
cols = [
    "Utilities",
    "BsmtFullBath" ,
    "BsmtHalfBath",
    "Exterior1st",
    "Exterior2nd",
    "BsmtFinSF1",
    "BsmtFinSF2",
    "BsmtUnfSF",
    "TotalBsmtSF",
    "GarageCars",
    "GarageArea"
]
data[cols].info()
# ["Utilities","Exterior2nd","Exterior1st"] 用'None'填充空值
data = null_fill(data, ["Utilities","Exterior2nd","Exterior1st"],
'None')
cols = [
    "BsmtFullBath" ,
    "BsmtHalfBath",
    "BsmtFinSF1",
    "BsmtFinSF2",
    "BsmtUnfSF",
    "TotalBsmtSF",
    "GarageCars",
    "GarageArea"
]
data[cols].describe()
# [
#     "BsmtFullBath" ,
#     "BsmtHalfBath",
#     "BsmtFinSF1",
#     "BsmtFinSF2",
#     "BsmtUnfSF",
#     "TotalBsmtSF",
#     "GarageCars",
#     "GarageArea"
# ]

```

```

# 用 0 填充空值
data = null_fill(data, cols, 0)

# Functional 用'Typ'填充
data["Functional"].fillna('Typ', inplace=True)

# Electrical 用众数填充
data["Electrical"].fillna(data["Electrical"].mode()[0], inplace=True)

# 填充 LotFrontage
def fill_lotfront(data):
    df = data.copy(deep=True)
    df['LotFrontage'] =
data.groupby("Neighborhood")["LotFrontage"].transform(
    lambda x: x.fillna(x.median()))
    df.drop("LotFrontage", axis=1, inplace=True)
    return df
df = fill_lotfront(data)

# 增加特征 TotalSF 地下室、一层、二层面积之和
df['TotalSF'] = df['TotalBsmntSF'] + df['1stFlrSF'] + df['2ndFlrSF']

def get_feats(df, type_="object"):
    """
    获取某一类型的全部特征
    """
    ans = []
    for col in df.columns:
        if df[col].dtype==type_:
            ans.append(col)
    return ans
int_feats = get_feats(df, 'int64')

for feat in int_feats:
    sns.distplot(df[feat], kde=False)
    plt.show()

# MSSubClass: 建筑类
# OverallQual: 评估房屋的整体材料和装饰
# OverallCond: 评估房屋的整体状况
# YearBuilt 建造年份
# YearRemodAdd 重塑年份
# FullBath
# HalfBath 浴室等级
# Fireplaces 壁炉

# MoSold 月销量
# YrSold 年销量
cols = ["MSSubClass", "OverallQual", "OverallCond", "YearBuilt",

```

```

"YearRemodAdd", "FullBath", "HalfBath",
    "MoSold", "YrSold"]
for c in cols:
    if c not in int_feats:
        print(c)
    else:
        int_feats.remove(c)
int_feats
for c in float_feats:
    sns.distplot(df[c], kde=False)
    plt.show()
# BsmtHalfBath, BsmtFullBath
# 表示地下室全部作为浴室还是一半作为浴室
# GarageYrBlt 车库建造年份
# garageCars 车库可停车辆
float_feats = get_feats(df, 'float64')
col_f = ["BsmtHalfBath", "BsmtFullBath", "GarageYrBlt", "GarageCars"]
for c in col_f:
    float_feats.remove(c)
float_feats.remove("SalePrice")
# 对偏态系数大的进行处理
def get_skew_cols(df, cols, eps=0.75):
    """
    获取偏态系数大于 eps 的列
    """
    skew = []
    for c in cols:
        if abs(df[c].skew()) > eps:
            skew.append(c)
    return skew
skw = get_skew_cols(df, int_feats+float_feats)
for c in skw:
    sns.distplot(df[c], kde=False)
    plt.title(c)
    plt.show()
# 剔除离散数据和没有相关属性占多数的特征
# ['KitchenAbvGr', 'TotRmsAbvGrd']
skw = get_skew_cols(df, int_feats+float_feats)
skw.remove("KitchenAbvGr")
skw.remove("TotRmsAbvGrd")
skw.remove("2ndFlrSF")
skw.remove("LowQualFinSF")
skw.remove("WoodDeckSF")
skw.remove("EnclosedPorch")

```

```

skw.remove("3SsnPorch")
skw.remove("ScreenPorch")
skw.remove("PoolArea")
skw.remove("MiscVal")
skw.remove("MasVnrArea")
skw.remove("BsmtFinSF1")
skw.remove("BsmtFinSF2")

for c in skw:
    sns.distplot(df[c], kde=False)
    plt.title(c)
    plt.show()
# 使用 box-cos 进行偏态数据处理
from scipy.special import boxcoxlp
lam = 0.1
for feat in skw:
    df[feat] = boxcoxlp(df[feat], lam)
for feat in skw:
    sns.distplot(df[feat],kde=False)
    plt.show()
# YearBuilt 建造年份
# YearRemodAdd 重塑年份
# GarageYrBlt 车库建造年份
import seaborn as sns
cols= ['GarageYrBlt','YearRemodAdd', 'GarageYrBlt']
for c in cols:
    sns.distplot(df[[c]])
    plt.title(c)
    plt.show()
def num_to_str(df, cols):
    for c in cols:
        df[c] = df[c].astype(str)
num_to_str(df, cols)
house_df = pd.get_dummies(df)
train = house_df[house_df['SalePrice'].notnull()]
test = house_df[house_df['SalePrice'].isnull()]

test.drop("SalePrice", axis=1, inplace=True)
x = train.drop("SalePrice", axis=1)
y = train["SalePrice"]
from sklearn.linear_model import ElasticNet, Lasso
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import RobustScaler

```



```

from sklearn.base import BaseEstimator, TransformerMixin,
RegressorMixin, clone
from sklearn.model_selection import KFold, train_test_split,
cross_val_score
from sklearn.metrics import mean_squared_error
from sklearn.kernel_ridge import KernelRidge
import xgboost as xgb
sns.kdeplot(y, shade=True)
y.skew() # 偏态分布, 右偏
y.describe()
y.describe()
n_folds = 5
def rmsle_cv(model, x=x, y=np.log(y)):
    kf = KFold(n_folds, shuffle=True, random_state=6).get_n_splits(x)
    rmse = np.sqrt(-cross_val_score(
        model, x, y, cv=kf, scoring="neg_mean_squared_error"
    ))
    return rmse

ENet = make_pipeline(RobustScaler(), ElasticNet(alpha=0.0005,
l1_ratio=.9, random_state=3))
lasso = make_pipeline(RobustScaler(), Lasso(alpha =0.0005,
random_state=1))
KRR = KernelRidge(alpha=0.6, kernel='polynomial', degree=2,
coef0=2.5)
GBoost = GradientBoostingRegressor(n_estimators=3000,
learning_rate=0.05,
                                max_depth=4, max_features='sqrt',
                                min_samples_leaf=15,
min_samples_split=10,
                                loss='huber', random_state =5)
model_xgb = xgb.XGBRegressor(colsample_bytree=0.4603, gamma=0.0468,
                                learning_rate=0.05, max_depth=3,
                                min_child_weight=1.7817, n_estimators=2200,
                                reg_alpha=0.4640, reg_lambda=0.8571,
                                subsample=0.5213,
                                nthread = -1)

score = rmsle_cv(lasso)
print("\nLasso score: {:.4f} ({:.4f})\n".format(score.mean(),
score.std()))
score = rmsle_cv(ENet)
print("ElasticNet score: {:.4f} ({:.4f})\n".format(score.mean(),
score.std()))
score = rmsle_cv(KRR)

```

```

print("Kernel Ridge score: {:.4f} ({:.4f})\n".format(score.mean(),
score.std()))
score = rmsle_cv(GBoost)
print("Gradient Boosting score: {:.4f}
({:.4f})\n".format(score.mean(), score.std()))
score = rmsle_cv(model_xgb)
print("Xgboost score: {:.4f} ({:.4f})\n".format(score.mean(),
score.std()))

class AveragingModels(BaseEstimator, RegressorMixin,
TransformerMixin):
    def __init__(self, models):
        self.models = models

    def fit(self, X, y):
        self.models_ = [clone(x) for x in self.models]

        for model in self.models_:
            model.fit(X, y)

        return self

    def predict(self, X):
        predictions = np.column_stack([
            model.predict(X) for model in self.models_
        ])
        return np.mean(predictions, axis=1)

averaged_models = AveragingModels(models = (model_xgb, GBoost, KRR,
ENet,lasso))
score = rmsle_cv(averaged_models)
print("Averaged base models score: {:.4f}
({:.4f})\n".format(score.mean(), score.std()))
averaged_models.fit(x,np.log(y))
pred = np.exp(averaged_models.predict(test))
result=pd.DataFrame({'Id':test[:].index,'SalePrice':pred})
result.to_csv('submission.csv',index=False)
# 提交结果 0.15210

```

三、实验及结果分析

3.1 实验数据

实验数据来自于 kaggle,House Prices - Advanced Regression Techniques 的数据

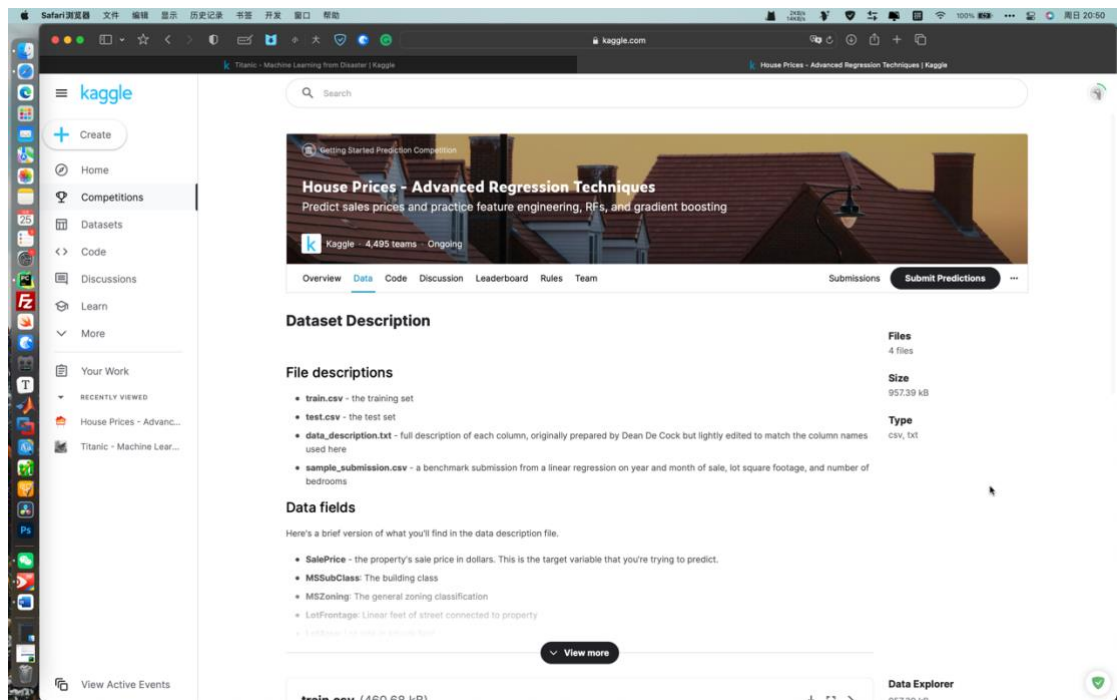


图 4 实验数据来源

The screenshot shows the 'train.csv' file view on Kaggle. The table displays the first 22 rows of data. The columns are: MSZoning, LotFrontage, LotArea, Street, Alley, and LotShape. The first row shows a house with MSZoning 'RL', LotFrontage 1460, LotArea 190, Street 'Pave', Alley 'NA', and LotShape 'Reg'. The second row shows a house with MSZoning 'RM', LotFrontage 60, LotArea 100, Street 'Pave', Alley 'NA', and LotShape 'Reg'. The third row shows a house with MSZoning 'RL', LotFrontage 60, LotArea 11258, Street 'Pave', Alley 'NA', and LotShape 'Reg'. The fourth row shows a house with MSZoning 'RL', LotFrontage 60, LotArea 9558, Street 'Pave', Alley 'NA', and LotShape 'Reg'. The fifth row shows a house with MSZoning 'RL', LotFrontage 64, LotArea 14268, Street 'Pave', Alley 'NA', and LotShape 'Reg'. The sixth row shows a house with MSZoning 'RL', LotFrontage 85, LotArea 14115, Street 'Pave', Alley 'NA', and LotShape 'Reg'. The seventh row shows a house with MSZoning 'RL', LotFrontage 75, LotArea 18884, Street 'Pave', Alley 'NA', and LotShape 'Reg'. The eighth row shows a house with MSZoning 'RL', LotFrontage NA, LotArea 18382, Street 'Pave', Alley 'NA', and LotShape 'Reg'. The ninth row shows a house with MSZoning 'RM', LotFrontage 51, LotArea 6128, Street 'Pave', Alley 'NA', and LotShape 'Reg'. The tenth row shows a house with MSZoning 'RL', LotFrontage 58, LotArea 7428, Street 'Pave', Alley 'NA', and LotShape 'Reg'. The eleventh row shows a house with MSZoning 'RL', LotFrontage 78, LotArea 11288, Street 'Pave', Alley 'NA', and LotShape 'Reg'. The twelfth row shows a house with MSZoning 'RL', LotFrontage 85, LotArea 11924, Street 'Pave', Alley 'NA', and LotShape 'Reg'. The thirteenth row shows a house with MSZoning 'RL', LotFrontage NA, LotArea 12968, Street 'Pave', Alley 'NA', and LotShape 'Reg'. The fourteenth row shows a house with MSZoning 'RL', LotFrontage 91, LotArea 18652, Street 'Pave', Alley 'NA', and LotShape 'Reg'. The fifteenth row shows a house with MSZoning 'RL', LotFrontage NA, LotArea 18928, Street 'Pave', Alley 'NA', and LotShape 'Reg'. The sixteenth row shows a house with MSZoning 'RM', LotFrontage 51, LotArea 6128, Street 'Pave', Alley 'NA', and LotShape 'Reg'. The seventeenth row shows a house with MSZoning 'RL', LotFrontage NA, LotArea 11241, Street 'Pave', Alley 'NA', and LotShape 'Reg'. The eighteenth row shows a house with MSZoning 'RL', LotFrontage 72, LotArea 18791, Street 'Pave', Alley 'NA', and LotShape 'Reg'. The nineteenth row shows a house with MSZoning 'RL', LotFrontage 66, LotArea 13695, Street 'Pave', Alley 'NA', and LotShape 'Reg'. The twentieth row shows a house with MSZoning 'RL', LotFrontage 79, LotArea 7568, Street 'Pave', Alley 'NA', and LotShape 'Reg'. The twenty-first row shows a house with MSZoning 'RL', LotFrontage 181, LotArea 14215, Street 'Pave', Alley 'NA', and LotShape 'Reg'. The twenty-second row shows a house with MSZoning 'RM', LotFrontage 57, LotArea 7449, Street 'Pave', Alley 'Grv1', and LotShape 'Reg'.

图 5 实验数据来源

3.2 实验环境

MacBook Pro (macOS Monterey12.6)

PyCharm 2021.3.2 (Professional Edition)

Build #PY-213.6777.50, built on January 27, 2022

Licensed to 成 孙

订阅有效期至 2023 年 10 月 11 日。

For educational use only.

Runtime version: 11.0.13+7-b1751.25 x86_64
VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o.
macOS 12.6
GC: G1 Young Generation, G1 Old Generation
Memory: 2048M
Cores: 12

3.3 实验结果

```
13 def predict(self, X):
14     predictions = np.column_stack([
15         model.predict(X) for model in self.models_
16     ])
17     return np.mean(predictions, axis=1)
18 averaged_models = AveragingModels(models = (model_xgb, GBoost, KRR, ENet, lasso))
19 score = rmsle_cv(averaged_models)
20 print(" Averaged base models score: {:.4f} ({:.4f})\n".format(score.mean(), score.std()))
21
```

Averaged base models score: 0.1253 (0.0133)

Out 102

	0
0	120287.681051
1	171054.356719
2	190068.147676
3	205691.443056
4	194890.837456
5	171372.342507
6	179082.199710
7	163773.129510
8	188158.804444
9	119509.880037
10	191820.833781

1459 rows x 1 columns [在新选项卡中打开](#)

3.4 实验分析

使用测试集得到的结果还有提升的空间，后期将叠加其他的算法来提高预测的精确度，使其更好的贴合我们的要求。

四、总结和下一步的展望

预测房价是一个有趣且具有挑战性的项目。它可以帮助我们了解市场动向，并做出更明智的购房决策。

在进行房价预测项目时，可以考虑以下几个方面：

1. 数据收集：获取有关房价的数据是项目的第一步。你可以搜索有关房地产的公开数据，或者联系房地产经纪人获取信息。你还可以使用地图数据，如房屋所在地的人口密度和周围基础设施的情况。
2. 数据清洗：在获取大量数据后，你需要对数据进行清洗，去除无用信息和异常值。这一步是很重要的，因为脏数据会对最终的预测结果产生不利影响。
3. 数据分析：在准备好数据之后，你可以使用数据分析工具，如 Excel 或 Python 的 Pandas 库，对数据进行统计分析。你可以查看各个因素（如房屋面积、地理位置、

建造年份等)对房价的影响,并建立相应的模型。

4. 模型训练与评估:接下来,你可以使用机器学习算法,如线性回归、决策树等。

在未来,房价预测项目还可以考虑以下几个方面的发展:

1. 数据更新:随着时间的推移,房地产市场的情况也会发生变化。因此,你可以定期更新数据,以保证预测模型的准确性。
2. 模型改进:机器学习算法是一个不断发展的领域,新的算法和技术不断涌现。因此,可以不断尝试新的算法,并尝试改进模型的准确度。
3. 应用拓展:除了预测房价外,还可以尝试将预测模型应用到其他领域。例如,可以尝试预测某个城市的租金走势,或者预测某个房地产项目的投资回报率。
4. 可视化展示:最后,可以使用数据可视化工具,如 **Matplotlib** 或 **Seaborn**,将预测结果以图表的形式呈现出来。这样就可以更直观地向其他人展示预测结果,并帮助他们做出更明智的决策。

五、自我评价

本次房价预测获得了准确的房价数据,并进行了清洗。考量了多种机器学习算法的优缺点,并选择了合适的模型,但是在训练集上的正确率还有待提升,后期将做出可视化以图表形式呈现,方便他人的理解和使用。

图像目标检测

一、摘要

基于深度学习的目标检测方法横空出世，迅速超过传统目标检测算法的效果。目前，基于深度学习的目标检测方法可大致分为两类：一类是基于候选区域的方法，如 R-CNN、SPP-net、Fast R-CNN、Faster R-CNN 等；另外一类是端到端的目标检测方法，即不需要候选区域，直接产生物体的类别概率和位置坐标值，如 YOLO、SSD 等。实验证明，虽然端到端的方法在速度上优势明显，但是基于候选区域的方法具有较高的准确率。这里主要介绍基于候选区域的相关方法，该类方法的基本思想是，从输入图片中以一定区域提取算法以获得一些可能包含检测目标的候选区域，然后将这些候选区域进行特征提取并分类，最后进行候选框回归以完成目标检测。

二、针对研究内容采取的具体的技术方案

2.1 具体技术路线

从输入图片中以一定区域提取算法以获得一些可能包含检测目标的候选区域，然后将这些候选区域进行特征提取并分类，最后进行候选框回归以完成目标检测。

2.2.算法原理

R-CNN 模型是基于候选区域的目标检测算法系列的开山之作，该模型首先进行区域搜索，然后再对候选区域进行分类，具体包括如下四个基本步骤：首先，使用 Selective Search 算法从输入的图片上提取出约 2000 个候选区域；然后，将候选区域大小归一化为 227x227，并将归一化后的候选区域输入到卷积神经网络中进行特征提取，此处使用的卷积神经网络是经过微调的 Alexnet 和 VGG16 等；接下来，用每类目标对应的 sVM 分别判断每个候选区域是否为该类；最后，通过非极大值抑制和物体边界框回归输出最终检测结果。

2.3.代码及注释

```
import os
import codecs
import Networks
import numpy as np
import process_data
import config as cfg
import tensorflow as tf
import joblib
slim = tf.contrib.slim
flower = {1:'pancy', 2:'Tulip'}
class Solver :
```

'''
此类用于训练或测试自定义的网络结构。自定义的网络和数据都是原材料， Solver 类属于锅，基于网络和数据来实现各种功能

参数: `net` --要用于训练或测试的网络结构（自定义的），属于类属性
 `data` --用来训练网络的数据，属于类属性
 `is_training` --当此类是用于训练网络是为 `True`，用网络进行预测时为 `false`
 `is_fineturn` --当此类用于 `fineturn` 步骤和特征提取步骤时为 `True`，其余时候为 `False`
 `is_Reg` --当此类用于 `bounding_box` 回归时为 `True`，其余时候为 `false`

函数: `save_cfg()` : 将网络中的参数，训练过程中的参数以 `txt` 的文件保存下来
 `train()` : 用于训练网络
 `predict(input_data)` : 将 `input_data` 作为网络的输入，得到网络运行之后的结果

```
'''  
  
def __init__(self, net, data, is_training=False, is_fineturn=False,  
is_Reg=False):  
    self.net = net  
    self.data = data  
    self.is_Reg = is_Reg  
    self.is_fineturn = is_fineturn  
    self.summary_step = cfg.Summary_iter  
    self.save_step = cfg.Save_iter  
    self.max_iter = cfg.Max_iter  
    self.staircase = cfg.Staircase  
  
    if is_fineturn:  
        self.initial_learning_rate = cfg.F_learning_rate  
        self.decay_step = cfg.F_decay_iter  
        self.decay_rate = cfg.F_decay_rate  
        self.weights_file = cfg.T_weights_file  
        self.output_dir = r'./output/fineturn'  
    elif is_Reg:  
        self.initial_learning_rate = cfg.R_learning_rate  
        self.decay_step = cfg.R_decay_iter  
        self.decay_rate = cfg.R_decay_rate  
        if is_training == True:  
            self.weights_file = None  
        else:  
            self.weights_file = cfg.R_weights_file  
        self.output_dir = r'./output/Reg_box'  
    else:  
        self.initial_learning_rate = cfg.T_learning_rate  
        self.decay_step = cfg.T_decay_iter  
        self.decay_rate = cfg.T_decay_rate
```

```

        if is_training == True:
            self.weights_file = None
        else:
            self.weights_file = cfg.F_weights_file
            self.output_dir = r'./output/train_alexnet'
            self.save_cfg()
            #在恢复模型及其参数时, 名字的 R-CNN/fc_11 网络层的参数不进行载入
            exclude = ['R-CNN/fc_11']
            self.variable_to_restore =
slim.get_variables_to_restore(exclude=exclude)
            self.variable_to_save =
slim.get_variables_to_restore(exclude=[])
            self.restorer = tf.train.Saver(self.variable_to_restore,
max_to_keep=1)
            self.saver = tf.train.Saver(self.variable_to_save,
max_to_keep=1)
            self.ckpt_file = os.path.join(self.output_dir, 'save.ckpt')

            self.summary_op = tf.summary.merge_all() # 可以将所有 summary 全部
            保存到磁盘, 以便 tensorboard 显示。如果没有特殊要求, 一般用这一句就可一显示训练时的
            各种信息了。
            self.writer = tf.summary.FileWriter(self.output_dir) # 指定一个
            文件用来保存图。

            self.global_step = tf.get_variable('global_step', [],
initializer=tf.constant_initializer(0), trainable = False)
            # exponential_decay() 是应用于学习率的指数衰减函数
            self.learning_rate = tf.train.exponential_decay(

self.initial_learning_rate,

self.global_step,
self.decay_step,
self.decay_rate,
self.staircase,
name='learning_rate'
)

        if is_training :
            self.optimizer =
tf.train.AdamOptimizer(learning_rate=self.learning_rate).minimize(

self.net.total_loss ,global_step=self.global_step
)
            self.ema = tf.train.ExponentialMovingAverage(0.99)

```



```

        self.average_op = self.ema.apply(tf.trainable_variables())
        with tf.control_dependencies([self.optimizer]):
            self.train_op = tf.group(self.average_op)

    self.sess = tf.Session()
    self.sess.run(tf.global_variables_initializer())

    if self.weights_file is not None:
        self.restorer.restore(self.sess, self.weights_file)
    self.writer.add_graph(self.sess.graph)

    def save_cfg(self):
        with open(os.path.join(self.output_dir, 'config.txt'), 'w') as
f:
            cfg_dict = cfg.__dict__
            for key in sorted(cfg_dict.keys()):
                if key[0].isupper():
                    cfg_str = '{}: {}\n'.format(key, cfg_dict[key])
                    f.write(cfg_str)

    def train(self):
        for step in range(1, self.max_iter+1):
            if self.is_Reg:
                input, labels = self.data.get_Reg_batch()
            elif self.is_fineturn:
                input, labels = self.data.get_fineturn_batch()
            else:
                input, labels = self.data.get_batch()

            feed_dict = {self.net.input_data:input,
self.net.label:labels}
            if step % self.summary_step == 0 :
                summary, loss,
_=self.sess.run([self.summary_op, self.net.total_loss, self.train_op],
feed_dict=feed_dict)
                self.writer.add_summary(summary, step)
                print("Data_epoch:"+str(self.data.epoch)+"
"*5+"training_step:"+str(step)+" "*5+ "batch_loss:"+str(loss))
            else:
                self.sess.run([self.train_op], feed_dict=feed_dict)
            if step % self.save_step == 0 :
                print("saving the model into " + self.ckpt_file)
                self.saver.save(self.sess, self.ckpt_file,
global_step=self.global_step)

```

```

def predict(self, input_data):
    feed_dict = {self.net.input_data :input_data}
    predict_result = self.sess.run(self.net.logits, feed_dict =
feed_dict)
    return predict_result

def get_Solvers():
    '''
    此函数用于得到三个 Solver: 特征提取的 Solver, SVM 预测分类的 solver, Reg_Box
    预测框回归的 Solver

    :return:
    '''
    weight_outputs = ['train_alexnet', 'fineturn', 'SVM_model',
'Reg_box']
    for weight_output in weight_outputs:
        output_path = os.path.join(cfg.Out_put, weight_output)
        if not os.path.exists(output_path):
            os.makedirs(output_path)

    if len(os.listdir(r'./output/train_alexnet')) == 0:
        # 训练网络
        Train_alexnet = tf.Graph()
        with Train_alexnet.as_default():
            Train_alexnet_data = process_data.Train_Alexnet_Data()
            Train_alexnet_net = Networks.Alexnet_Net(is_training=True,
is_fineturn=False, is_SVM=False)
            Train_alexnet_solver = Solver(Train_alexnet_net,
Train_alexnet_data, is_training=True, is_fineturn=False, is_Reg=False)
            Train_alexnet_solver.train()
        # 微调
    if len(os.listdir(r'./output/fineturn')) == 0:
        Fineturn = tf.Graph()
        with Fineturn.as_default():
            Fineturn_data = process_data.FineTun_And_Predict_Data()
            Fineturn_net = Networks.Alexnet_Net(is_training=True,
is_fineturn=True, is_SVM=False)
            Fineturn_solver = Solver(Fineturn_net, Fineturn_data,
is_training=True, is_fineturn=True, is_Reg=False)
            Fineturn_solver.train()

    # 提取特征向量
    Features = tf.Graph()

```

```

    with Features.as_default():
        Features_net = Networks.Alexnet_Net(is_training=False,
is_fineturn=True, is_SVM=True)
        Features_solver = Solver(Features_net, None, is_training=False,
is_fineturn=True, is_Reg=False)
        Features_data = process_data.FineTun_And_Predict_Data(Features_solver, is_svm=True,
is_save=True)
# SVM 对特征向量分类
        svms = []
        if len(os.listdir(r'./output/SVM_model')) == 0:
            SVM_net = Networks.SVM(Features_data)
            SVM_net.train()
        for file in os.listdir(r'./output/SVM_model'):
            svms.append(joblib.load(os.path.join('./output/SVM_model',
file)))

# 利用边框回归修正候选框
        Reg_box = tf.Graph()
        with Reg_box.as_default():
            Reg_box_data = Features_data
            Reg_box_net = Networks.Reg_Net(is_training=True)
            if len(os.listdir(r'./output/Reg_box')) == 0:
                Reg_box_solver = Solver(Reg_box_net, Reg_box_data,
is_training=True, is_fineturn=False, is_Reg=True)
                Reg_box_solver.train()
            else:
                Reg_box_solver = Solver(Reg_box_net, Reg_box_data,
is_training=False, is_fineturn=False, is_Reg=True)
            return Features_solver, svms, Reg_box_solver

if __name__ == '__main__':

    Features_solver, svms, Reg_box_solver = get_Solvers()

    img_path = './17flowers/jpg/7/image_0561.jpg' # or
    './17flowers/jpg/16/****.jpg'
    imgs, verts = process_data.image_proposal(img_path) # 对图片进行候选框
    提取
    process_data.show_rect(img_path, verts, ' ')
    features = Features_solver.predict(imgs)
    print(np.shape(features))

    results = []

```

```

results_old = []
results_label = []
count = 0
for f in features:
    for svm in svms:
        pred = svm.predict([f.tolist()])
        # not background
        if pred[0] != 0:
            results_old.append(verts[count])
            #print(Reg_box_solver.predict([f.tolist()]))
            if Reg_box_solver.predict([f.tolist()])[0][0] > 0.5:
                px, py, pw, ph = verts[count][0], verts[count][1],
verts[count][2], verts[count][3]
                old_center_x, old_center_y = px + pw / 2.0, py + ph /
2.0
                x_ping, y_ping, w_suo, h_suo =
Reg_box_solver.predict([f.tolist()])[0][1], \

Reg_box_solver.predict([f.tolist()])[0][2], \

Reg_box_solver.predict([f.tolist()])[0][3], \

Reg_box_solver.predict([f.tolist()])[0][4]
                new_center_x = x_ping * pw + old_center_x
                new_center_y = y_ping * ph + old_center_y
                new_w = pw * np.exp(w_suo)
                new_h = ph * np.exp(h_suo)
                new_verts = [new_center_x, new_center_y, new_w,
new_h]

                results.append(new_verts)
                results_label.append(pred[0])
            count += 1

average_center_x, average_center_y, average_w, average_h = 0, 0, 0,
0
#给预测出的所有的预测框区一个平均值，代表其预测出的最终位置
for vert in results:
    average_center_x += vert[0]
    average_center_y += vert[1]
    average_w += vert[2]
    average_h += vert[3]
average_center_x = average_center_x / len(results)
average_center_y = average_center_y / len(results)
average_w = average_w / len(results)

```

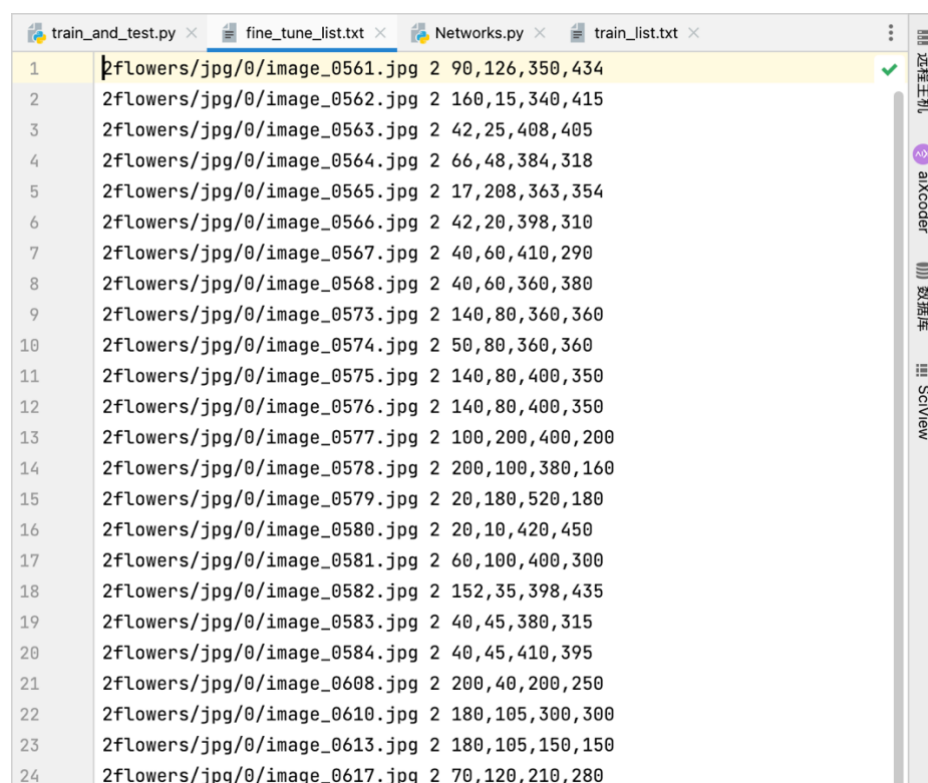
```

average_h = average_h / len(results)
average_result = [[average_center_x, average_center_y, average_w,
average_h]]
result_label = max(results_label, key=results_label.count)
process_data.show_rect(img_path, results_old, ' ')
process_data.show_rect(img_path,
average_result, flower[result_label])

```

三、实验及结果分析

3.1 实验数据



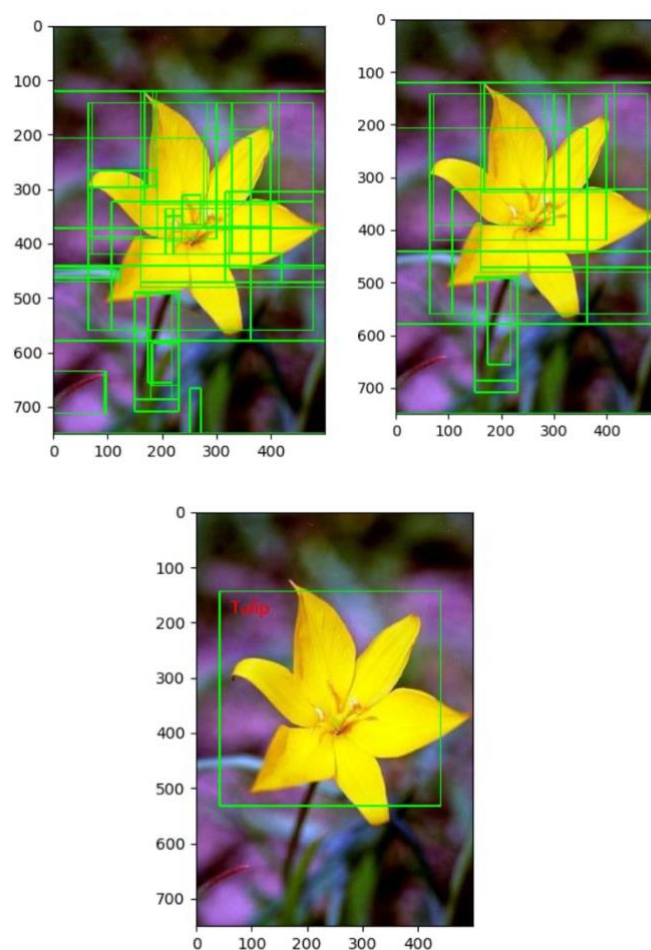
1	2flowers/jpg/0/image_0561.jpg	2	90,126,350,434	✓
2	2flowers/jpg/0/image_0562.jpg	2	160,15,340,415	
3	2flowers/jpg/0/image_0563.jpg	2	42,25,408,405	
4	2flowers/jpg/0/image_0564.jpg	2	66,48,384,318	
5	2flowers/jpg/0/image_0565.jpg	2	17,208,363,354	
6	2flowers/jpg/0/image_0566.jpg	2	42,20,398,310	
7	2flowers/jpg/0/image_0567.jpg	2	40,60,410,290	
8	2flowers/jpg/0/image_0568.jpg	2	40,60,360,380	
9	2flowers/jpg/0/image_0573.jpg	2	140,80,360,360	
10	2flowers/jpg/0/image_0574.jpg	2	50,80,360,360	
11	2flowers/jpg/0/image_0575.jpg	2	140,80,400,350	
12	2flowers/jpg/0/image_0576.jpg	2	140,80,400,350	
13	2flowers/jpg/0/image_0577.jpg	2	100,200,400,200	
14	2flowers/jpg/0/image_0578.jpg	2	200,100,380,160	
15	2flowers/jpg/0/image_0579.jpg	2	20,180,520,180	
16	2flowers/jpg/0/image_0580.jpg	2	20,10,420,450	
17	2flowers/jpg/0/image_0581.jpg	2	60,100,400,300	
18	2flowers/jpg/0/image_0582.jpg	2	152,35,398,435	
19	2flowers/jpg/0/image_0583.jpg	2	40,45,380,315	
20	2flowers/jpg/0/image_0584.jpg	2	40,45,410,395	
21	2flowers/jpg/0/image_0608.jpg	2	200,40,200,250	
22	2flowers/jpg/0/image_0610.jpg	2	180,105,300,300	
23	2flowers/jpg/0/image_0613.jpg	2	180,105,150,150	
24	2flowers/jpg/0/image_0617.jpg	2	70,120,210,280	

图 6 实验数据

3.2 实验环境

MacBook Pro (macOS Monterey12.6)
 PyCharm 2021.3.2 (Professional Edition)
 Build #PY-213.6777.50, built on January 27, 2022
 Licensed to 成 孙
 订阅有效期至 2023 年 10 月 11 日。
 For educational use only.
 Runtime version: 11.0.13+7-b1751.25 x86_64
 VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o.
 macOS 12.6
 GC: G1 Young Generation, G1 Old Generation
 Memory: 2048M
 Cores: 12

3.3 实验结果



3.4 实验分析

由实验结果我们可以看出，本次试验我们很好的完成实验目标，期待在后期能够继续完善识别的速度和效果。

四、总结和下一步的展望

图像目标检测是指在图像中识别出特定目标并标记其位置的过程。这是一个广泛的研究领域，并且可以应用于许多不同的应用程序，如自动驾驶、机器人、安防、医学影像等。

近年来，图像目标检测领域取得了显著进展，其中大量使用深度学习方法，如卷积神经网络 (CNN)。目前最先进的方法之一是基于单次推理的检测器，其中包括 YOLO (You Only Look Once) 和 SSD (Single Shot Detector)。这些方法在速度和准确性方面都十分优秀。

此外，还有许多基于阶段的方法，例如 R-CNN (Regions with CNN features) 和 Fast R-CNN。这些方法通常比单次推理方法慢，但在准确性方面可能会更好。

在未来，预计图像目标检测领域将继续取得进展，并可能出现新的方法和技术。例

如，可能会出现更快速、更准确的检测器，并且可能会有更多的应用程序。此外，随着计算能力的提高，可能会出现更复杂的模型，这些模型可能在准确性和性能方面都有所提高。

五、自我评价

通过这次试验我使用 R-CNN 模型，完成了图像目标检测，其准确率达到我们预期的标准。在实验过程中我学到了编程的技巧和方法，以及阅读代码的能力。本次机器学习大作业是一次很好的机会，让我的能力得到了很大的提升。