摘　要

人机交互是计算机领域的热门话题，计算机的发展不能只专注于硬件提升，在人机交互方面尤也为重要，如今的交互技术也是日新月异。手势识别技术作为一种比较新的交互技术，它可以帮助人们更便捷的操控软件，加快文档处理效率。手势识别技术按输入设备的不同可分为基于数据手套及其他硬件的手势识别和基于计算机视觉的手势识别两种。手势的种类繁多，因而使计算机在进行手势识别分类操作的时候显得比较困难，随着计算机领域的硬件性能的提升，加上更加适合的软件算法，许多之前不可能解决的问题变成了可能。

业界手势识别大致分为静态手势识别和动态手势识别两种。第一种是计算机只通过识别手部的静态图像来确认用户的指令。第二种是计算机通过识别手部形态，并且识别手部运动轨迹来精确用户指令。针对静态手势，本项目提出了一种基于 CNN 的有效提取方法，经过 opencv 库调取摄像头得到手的轮廓，最后，通过轮廓匹配得到识别结果。在各种灯光下进行重复测试，并调整系统参数。实验表明，该方案来提取手势并实现游戏控制是可行的，取得了良好的游戏体验效果。

关键词：　手势识别　CNN　Opencv　游戏控制

Abstract

Human-computer interaction is a hot topic in the field of computers. The development of computers cannot only focus on hardware improvement, but also is particularly important in terms of human-computer interaction. Today's interaction technology is also changing with each passing day. Gesture recognition technology as a relatively new interactive technology, it can help people more easily control the software, speed up the efficiency of document processing. Gesture recognition technology can be divided into gesture recognition based on data gloves and other hardware and gesture recognition based on computer vision according to different input devices. There are many types of gestures, which makes it difficult for computers to perform gesture recognition and classification operations. With the improvement of hardware performance in the computer field and more suitable software algorithms, many problems that were impossible to solve before became possible.

Gesture recognition in the industry is roughly divided into static gesture recognition and dynamic gesture recognition. The first is that the computer only confirms the user's instructions by recognizing the static image of the hand. The second is that the computer can accurately determine user instructions by recognizing the shape of the hand and recognizing the movement trajectory of the hand. For static gestures, this project proposes an effective CNN-based extraction method. The camera is obtained through the opencv library to obtain the contour of the hand, and finally, the recognition result is obtained through contour matching. Repeat the test under various lights and adjust the system parameters. Experiments show that this scheme is feasible to extract gestures and achieve game control, and has achieved good game experience effects.

Keywords:Gesture recognition　CNN　Opencv　Game control

目录

第一章 绪论

1.1 研究背景与意义

时间来到了 2020 年，计算机的性能有了飞跃的提升，计算机在各个领域都得到了普及，使得人机交互技术越来越成为计算机领域的研究热点，手势识别就是其中之一。将手势识别作为一种交互技术能应用到很多场景中，因此，研究手势识别技术拥有很好的前景。平时通过键盘和触控实现的人机交互技术在手势识别技术面前，显得不那么高端，手势识别技术应用计算机视觉收集手部信息，通过程序转化输出识别结果，达到交互效果。但是，手势变化多样，使得手势识别成为一项极具挑战性的研究项目。

目前的手势识别技术日趋成熟，有的是基于手环感知肌肉抖动信息来完成手势识别的，有的是通过指套的形式，结合距离传感器来实现手势识别，这两种比较依赖硬件，还有一种是通过计算机视觉来实现，对硬件要求不高，但可以达到相近的水平，为了达到预期的操作效果和开发速度，本文中引入了微软的 Opencv 函数库和 CNN(卷积神经网络)。手势识别的意义在于通过手势隔空操作，不用去触碰操作实体，为人机交互增添新的方式。

图象处理和图象识别是实现手势识别技术的前提，手势识别技术的发展，使计算机可以完成很多看似不可能的任务。在很多领域中，手势识别都起着很大的作用，下面列出了一些在手势识别中有应用前景的领域。

在视频直播领域引入手势识别。直播已成为大众娱乐项目之一，不光是直播的内容更加丰富多彩了，各种直播特效让屏幕内容眼花缭乱，手势识别在其中的作用也非同小可，主播可以在不操作键盘，直接通过手势来使用各种直播特效来吸引观众的眼球。

在物联网领域引入手势识别技术。手势控制开关灯，手势控制电视，手势拍照，手势唤醒智能终端等等，增加了人机互动的趣味性。

在虚拟现实领域引入手势识别。虚拟现实技术的不仅给我们带来丰富的视觉体验，手势识别技术的加入使得它能操控虚拟世界的物品，所以手势识别技术是虚拟现实技术交互中的一个重要环节。现如今的虚拟现实设备的手势识别大多都是在硬件层面实现的，成本较高，要想在普通相机上挑战高质量的手指追踪效果，难度较大。

在手机领域引入手势识别。随着手机性能的提升，图像处理能力也得到了相应的提高，越来越多的手机厂商也开始注重手机的 AI 性能，像谷歌的 pixe 系列，华为的高端系列，小米等，特别是华为的 EMUI10 智慧全连接系统，加入了系统级别的手势操作，为手机操作增添了不少乐趣。

图 1-1 手势识别应用场景

1.2 国内外研究现状及分析

1.2.1 国外研究现状

21 世纪初，手势识别技术就得到了技术人员的广泛研究，特别是基于计算机视觉的手势识别。2004 年韩国 Inda 大学和 Korea Polytechnic 大学的 JongShill Lee、YoungJoo Lee 等人用熵分析法从背景复杂的视频流中分割出手势区域并进行手势识别[1]。利用 freeman 码技术进行手部边缘的检测，计算出手势中心到手部边缘的距离。它在六种手势的分类识别上可以达到百分之九十五的识别率。2010 年 chendong Yu 等人[2] 在手势识别中采用了基于视觉的组合特征，并结合了手部面积、周长、重心、面积比和长宽比等特征点，提高了手势识别率。2014 年，脸书收购了一家 VR 的初创公司，并吸收了其在手势识别技术深耕多年的技术人员。2019 年，Oculus 发布了可通过四个较为廉价的摄像头实现精度较高的手势识别的 V12 测试版本，并开放 API。

1.2.2 国内研究现状

我国在手势识别方面的研究与国外相比并没有落后太多，但随着国内学者的努力，我们也在一步步的追赶。

2006 年来自中国科学技术大学和哈尔滨工业大学刘岩[3]等人，研究了基于“大小手”的实时徒手手势识别，将双手分为大手和小手，并根据单手的重叠情况进行处理。他们的论文用了他们首创的大小手特征提取算法，实现了基于动态的手势识别技术，对 17 种常用的手势进行了识别率达百分之九十四。

2012 年根据 AdaBoost 算法和光流匹配的原理，王凯等人[4]提出了一种实时手势识别方案：只要与计算机相连，通过摄像头读取 2 D 手势视频片段，就可以对手势进行更精确的识别。运用迭代算法，对一个训练集训练多个分类器，将多个分类器再重组成一个强分类器，实现手势的分类识别。

最近几年上线的百度 AI 开放平台提供了多种手势识别产品，通过 API 连接可以识别多种常见手势，如抱拳，停止，比心，数字，剪刀手等。

1.3 文本主要工作和内容

本文的研究重点在于：

研究并开发基于 Pygame 框架的小游戏开发。

研究 CNN 的网络结构，采用一种合适的模型结构进行训练。

研究用合适的分类算法，用于提高手势识别的准确率。

研究使用何种方式进行手势分类识别。

1.4 文本章节安排

第一章：绪论。介绍了本论文研究背景与研究意义，分析国内外关于手势识别的研究历史与各大厂商研究现状，并撰写了本文研究的主要内容和章节安排。

第二章：基于 pygame 小游戏开发。本章简单介绍了 pygame 模块，并开发一款基于 pygame 的小游戏。

第三章：CNN 理论介绍。本章简单介绍了 CNN 的结构，介绍了时下主流的几种深度学习框架。

第四章：神经网络搭建。本章编写从搭建到设计 CNN 结构再到训练模型的全过程，使用训练好的模型进行手势识别并操控游戏。

第五章：总结与展望。首先为整个实验做实验总结，简述了整个项目开发

过程并思考了实验的不足之处，以及改进方向以供后续开发。

第二章 基于 Pygame 的小游戏开发

2 .1 Pygame 简介

Pygame 是一个用于游戏开发的跨平台的 Python 模块，由于基于 SDL,所以他有很多多媒体相关函数。对于 Python 程序，使用 Pygame 模块可以创建功能丰富的游戏和各式各样的多媒体程序，并且因为 Pygame 模块的部分是由 C 语言开发而成，所以它执行起来并不算慢，它还拥有 Python 语言的面向对象、脚本编程和高度可移植等特性。

当你再开发游戏时使用 pygame 模块，你会发现十分简单，每一个控件都是一个相对独立的 surface 对象。Surface 对象的作用是在 Pygame 中显示图像，该对象可以将一个图像绘制到另一个图像的上方，并设置图像的透明度。并且 Pygame 中的 Rect 类能储和处理矩形对象，比如文本框控件，Button 控件等。React 类能定义组件的参数，这样可以让开发者更加容易地使用图像的属性来完成容器中的定位布局。

2.2 《奔跑吧恐龙》游戏开发

2.2.1 搭建 python3.6.8 实验环境

如图 2-1，本小游戏用 python3.6.8 作为基础实验环境。Python 是一种高级编程语言，它设计得非常容易阅读，而且是可移植的，不同的操作系统之间只需要重新编译即可完美运行,它还有丰富的科学计算库,并且是免费开源的。Python 是一种交互式语言，可以使用众多开发工具来编写和调试程序。Python 支持面向对象编程，对于初级程序员来说， Python 是一门很棒的语言，它支持简单的文字处理，用 flash 框架搭建 web 后端框架,用 Pygame 编写高性能游戏。由于语法简单，对初级程序员友好，python 成为当下热门编程语言之一。由于各大厂都有用 Python 的项目，所以 Python 的生态也一直很好，有丰富的学习资源，完善的中文文档，开发者社区等。

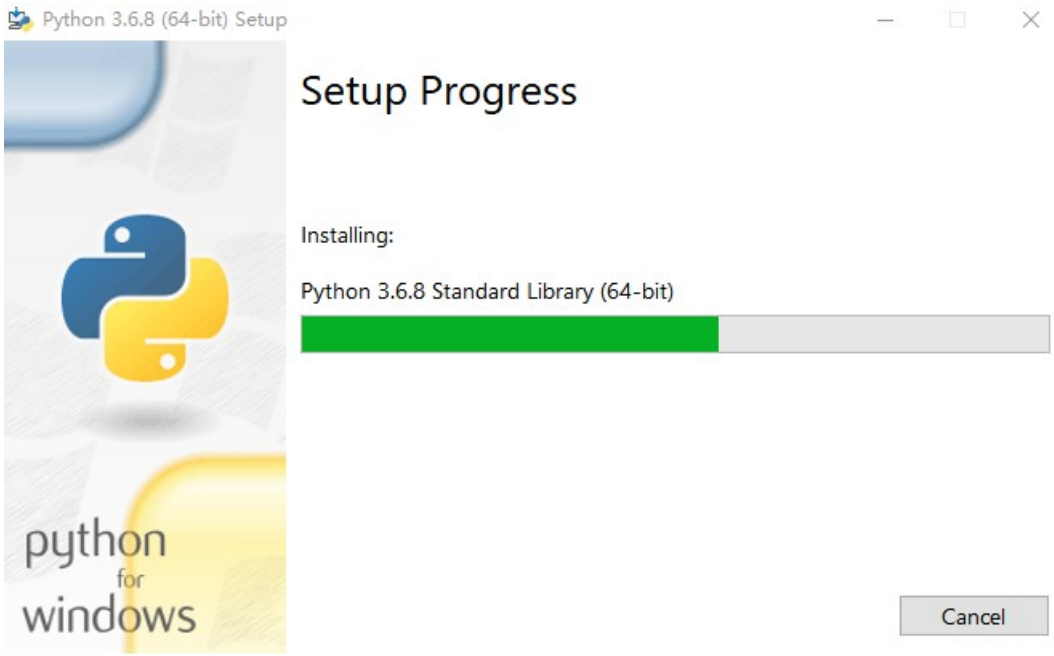

图 2-1 python3.6.8 安装

2.2.2 加载 pygame 模块

Pip 是一款现代化的公共 Python 包管理工具，它可以用来查找、下载、安装和卸载 Python 包。Pip 默认源国内访问比较慢，所以用国内清华源加载，如图 2-2 所示。

```
C:\Users\Administrator>pip install pygame -i https://pypi.tuna.tsinghua.edu
.cn/simple
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting pygame
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/79/b0/2fec9a5d9151
c8f8e20721349950c1bde2d02b331c19e178c3d76876a218/pygame-1.9.6-cp36-cp36m-wi
n_amd64.whl (4.3MB)
    19% |                                | 849kB 8.0MB/s eta 0:00:01
    20% |                                | 860kB 7.4MB/s eta 0:00:01
    20% |                                | 870kB 6.8MB/s eta 0:00:01
    20% |                                | 880kB 8.0MB/s eta 0:00:01
    20% |                                | 890kB 8.0MB/s eta 0:00:01
    21% |                                | 901kB 6.9MB/s eta 0:00:01
    21% |                                | 911kB 7.7MB/s eta 0:00:01
    21% |                                | 921kB 8.4MB/s eta 0:00:01
    21% |                                | 931kB 7.7MB/s eta 0:00:01
    22% |                                | 942kB 7.7MB/s eta 0:00:01
    22% |                                | 952kB 6.7MB/s eta 0:00:01
    22% |                                | 962kB 7.7MB/s eta 0:00:01
    22% |                                | 972kB 7.8MB/s eta 0:00:01
    23% |                                | 983kB 7.2MB/s eta 0:00:0
    23% |                                | 993kB 7.8MB/s eta 0:00:0
    23% |                                | 1.0MB 8.4MB/s eta 0:00:0
    23% |                                | 1.0MB 8.4MB/s eta 0:00:0
    24% |                                | 1.0MB 9.2MB/s eta 0:00:0
```

图 2-2 pip 加载 pygame 模块

2.2.3 实现小游戏主窗体

```
import pygame
from pygame.locals import *
#将游戏窗口设置在显示中心
os.environ['SDL_VIDEO_CENTERED'] = '1'
#定义一些全局变量
SCREEN_WIDTH = 1280
SCREEN_HEIGHT = 720
FPS = 30
GROUND_HEIGHT = SCREEN_HEIGHT - 70
PLAY_GAME = True
#初始化 pygame 并创建窗口
pygame.init()
window = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
pygame.display.set_caption("奔跑吧恐龙")
```

实现效果如图 2-3：

图 2-3 《奔跑吧恐龙》主窗口

2.2.4 游戏背景的滚动

其实《奔跑吧恐龙》这个游戏只有背景在滚动，恐龙并没有前移。背景由 4 张图片组成，如图 2-4，并不断向后移动和刷新。

```
# 创建和移动单个背景的类
class Background:
    def __init__(self, image_path, speed=10):
        self.image0, self.rect0 = load_image(image_path, 1280, 720)
        self.image1, self.rect1 = load_image(image_path, 1280, 720)
        self.rect0.bottom = SCREEN_HEIGHT
        self.rect1.bottom = SCREEN_HEIGHT
        self.rect1.left = self.rect0.right
        self.speed = speed
    def draw(self):
        window.blit(self.image0, self.rect0)
        window.blit(self.image1, self.rect1)
    def update(self):
        self.rect0.left -= self.speed
        self.rect1.left -= self.speed
        if self.rect0.right < 0:
            self.rect0.left = self.rect1.right
        if self.rect1.right < 0:
            self.rect1.left = self.rect0.right
# 使用 Background 类创建和移动单个或多个背景
class AllBackgrounds:
    def __init__(self, game_speed):
```

```
        self.background_0 = Background("image/background/bg_0.png",
game_speed)
        self.background_1 = Background("image/background/bg_1.png",
game_speed - 7)
        self.background_2 = Background("image/background/bg_2.png",
game_speed - 8)
        self.background_3 = Background("image/background/bg_3.png",
game_speed - 9)
    def update_speed(self, speed):
        self.background_0.speed = speed
        self.background_1.speed = speed - 7
        self.background_2.speed = speed - 8
        self.background_3.speed = speed - 9
    def draw(self):
        self.background_3.draw()
        self.background_2.draw()
        self.background_1.draw()
        self.background_0.draw()
    def update(self):
        self.background_3.update()
        self.background_2.update()
        self.background_1.update()
        self.background_0.update()
```

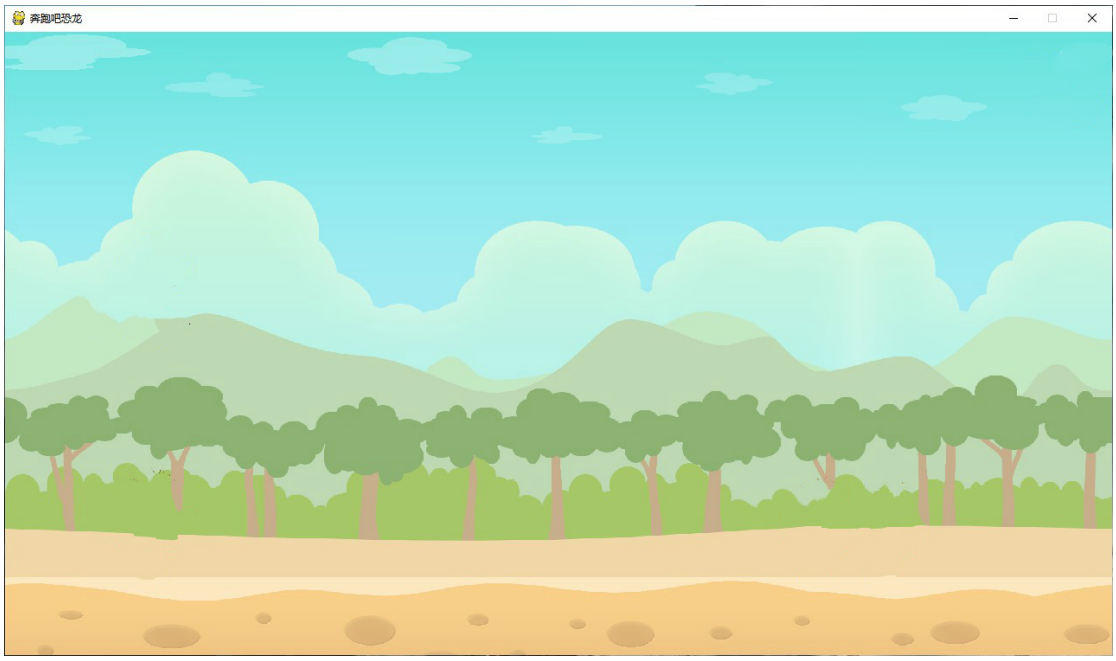

图 2-4

2.2.5 加入障碍物仙人掌

游戏的规则是恐龙越过仙人掌就加分，如果在跳跃或落地时碰到仙人掌则死亡，仙人掌是随机出现的，如图 2-5。

```
# 创建和移动障碍仙人掌的类
class Cactus:
    def __init__(self, speed=10):
        self.cactus_images=load_sprites("image/cactus/","cactus_",5, 160, 160)
        self.cactus_image_0, self.rect_0 = self.cactus_images[0], self.cactus_images[0].get_rect()
        self.cactus_image_1, self.rect_1 = self.cactus_images[1], self.cactus_images[1].get_rect()
        self.rect_0.bottom = GROUND_HEIGHT - 11
        self.rect_0.left = SCREEN_WIDTH
        self.rect_1.bottom = GROUND_HEIGHT - 11
```

```
        self.rect_1.left = self.rect_0.right + SCREEN_WIDTH/2
        self.speed = speed
        self.range_0 = 240
        self.range_1 = 720
    def get_cactus(self):
        current_cactus = [self.cactus_image_0, self.cactus_image_1]
        cactus_rect = [self.rect_0, self.rect_1]
        return current_cactus, cactus_rect
    def update_speed(self, speed):
        self.speed = speed
        self.range_0 += 1
        self.range_1 += 1
    def draw(self):
        window.blit(self.cactus_image_0, self.rect_0)
        window.blit(self.cactus_image_1, self.rect_1)
    def update(self):
        self.rect_0.left -= self.speed
        self.rect_1.left -= self.speed
        if self.rect_0.right < 0:
            temp_position = self.rect_1.right +
random.randrange(self.range_0, self.range_1)
            if temp_position > SCREEN_WIDTH:
                self.rect_0.left = temp_position
            else:
                self.rect_0.left = SCREEN_WIDTH
```

```
            temp_index = random.randrange(0, 5)
            self.cactus_image_0 = self.cactus_images[temp_index]
        if self.rect_1.right < 0:
            temp_position = self.rect_0.right +
random.randrange(self.range_0, self.range_1)
            if temp_position > SCREEN_WIDTH:
                self.rect_1.left = temp_position
            else:
                self.rect_1.left = SCREEN_WIDTH
            temp_index = random.randrange(0, 5)
            self.cactus_image_1 = self.cactus_images[temp_index]
```

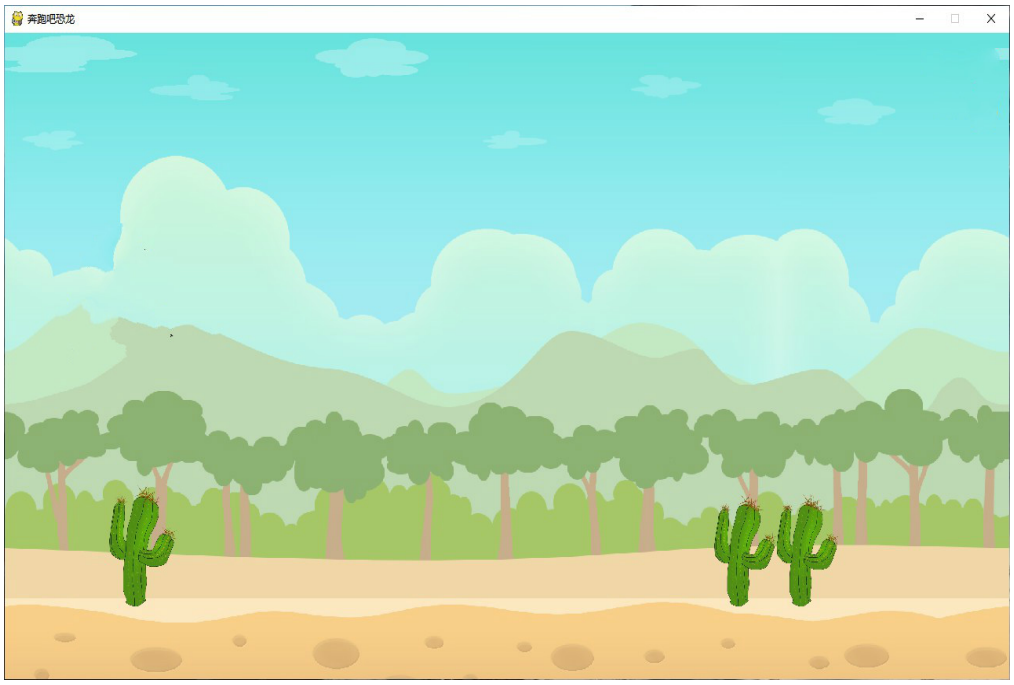

图 2-5

2.2.6 加入恐龙跳跃

这个游戏中，恐龙只有跳跃障碍物这个动作。如图

```
# 创建和恐龙跳跃的类
class Dino:
    def __init__(self):
        self.idle_images = load_sprites("image/dino/","idle_",10,220,
153)
        self.running_images = load_sprites("image/dino/", "run_", 8,
220, 153)
        self.jumping_images = load_sprites("image/dino/", "jump_", 16,
220, 153)
        self.rect = self.idle_images[0].get_rect()
        self.rect.bottom = GROUND_HEIGHT
        self.rect.left = 70
        self.jump_limit = GROUND_HEIGHT - 290
        self.jump_speed = 50  # 开始跳跃的速度
        self.gravity_up = 4  # 跳跃时的变化率
        self.gravity_down = 2  # 跌落时的变化率
        # 这些索引在子画面的图像之间循环，使恐龙看起来动起来
        self.idle_index = 0
        self.running_index = 0
        self.jumping_index = 0
        # 这些布尔值确定应显示哪些图像
        self.idle = True
        self.running = False
        self.jumping = False
        self.falling = False
```

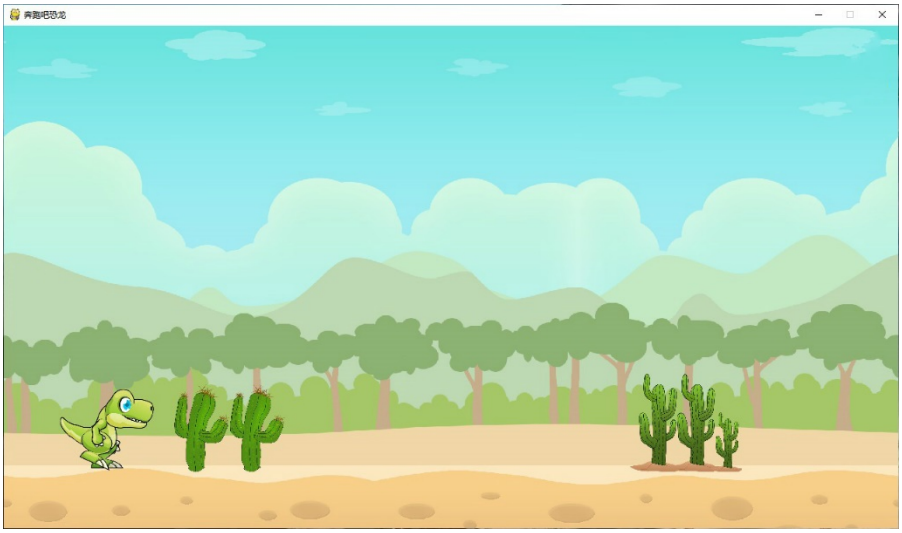

图 2-6

2.2.7 事件监听

整个游戏不止包含对键盘的监听，还包括鼠标点按位置的监听等。

```
for event in pygame.event.get():
            if event.type == QUIT:
                pygame.quit()  # 退出 pygame
                sys.exit()  # 退出程序
            if event.type == pygame.MOUSEBUTTONDOWN:
                mx, my = pygame.mouse.get_pos()  # 获取鼠标单击坐标
                if game_over:
                    # 检查是否单击了“play agin”按钮
                    if game_over_screen.rect.left < mx < game_over_screen.rect.right and \
                            game_over_screen.rect.top < my < game_over_screen.rect.bottom:
                        play_again = True
                        run = False
```

```
            key = pygame.key.get_pressed()  # 按下空格键
            if key[K_SPACE] or key[K_UP]:
                if game_over:
                    play_again = True
                    run = False
                elif not dino.jumping:
                    jump_sound.play()
                    dino.jumping = True
                    dino.running = False
                    if dino.idle:
                        dino.idle = False
```

2.2.8 检测是否碰撞

当恐龙碰撞到障碍物仙人掌的时候，游戏结束，所以要写一个检测是否碰到障碍物的类。

```
def check_collision(self, all_cactus):
        if self.running:
            dino_mask =
pygame.mask.from_surface(self.running_images[self.running_index])
        elif self.jumping:
            dino_mask =
pygame.mask.from_surface(self.jumping_images[self.jumping_index])
        else:
            dino_mask =
pygame.mask.from_surface(self.idle_images[self.idle_index])
        current_cactus, cactus_rect = all_cactus
        offset_0 = (cactus_rect[0].left - self.rect.left,
```

```
 cactus_rect[0].top - self.rect.top)
        offset_1 = (cactus_rect[1].left - self.rect.left,
cactus_rect[1].top - self.rect.top)
        collide =
dino_mask.overlap(pygame.mask.from_surface(current_cactus[0]),
offset_0) or \
dino_mask.overlap(pygame.mask.from_surface(current_cactus[1]),
offset_1)
        return collide
```

2.2.9 计分功能

加入计分机制，游戏进行的同时，分数不断增加，且分数实时显示在屏幕上直至恐龙碰到仙人掌，停止计分，并将最高分写入 high_score.txt，如图 2-7。

```
# 用于在屏幕上计数和绘制分数并将高分数保存在文件中
class Score:
    def __init__(self):
        self.high_score_image, self.rect_high =
load_image("image/score/high_score.png", 35, 35)
        self.current_score_image, self.rect_current =
load_image("image/score/current_score.png", 35, 35)
        self.rect_high.topright = (SCREEN_WIDTH - 15, 20)
        self.rect_current.topright = (SCREEN_WIDTH - 15, 65)
        self.high_score = 0
        self.score = 0
        self.load()
        self.high_score_achieved = False
        self.call_count = 0
    def count(self):
```

```
        if self.call_count % 2 == 0:
            self.score += 1
            if self.high_score_achieved:
                self.high_score = self.score
            elif self.score > self.high_score:
                self.high_score = self.score
                self.high_score_achieved = True
                score_sound.play()
        self.call_count = self.call_count + 1


    def draw(self):
        window.blit(self.high_score_image, self.rect_high)
        window.blit(self.current_score_image, self.rect_current)
        draw_text(str(self.high_score), "font/monofonto.ttf", 28, (19,
130, 98), SCREEN_WIDTH - 60, 20, "topright")
        draw_text(str(self.score), "font/monofonto.ttf", 28, (19, 130,
98), SCREEN_WIDTH - 60, 65, "topright")
    def load(self):
        # 载入高分
        try:
            with open("high_score.txt", "r") as file:
                self.high_score = int(file.read())
        except (IOError, ValueError):
            self.high_score = 0
    def save(self):
```

```
# 保存高分
if self.high_score_achieved:
    with open("high_score.txt", "w") as file:
        file.write(str(self.high_score))
```

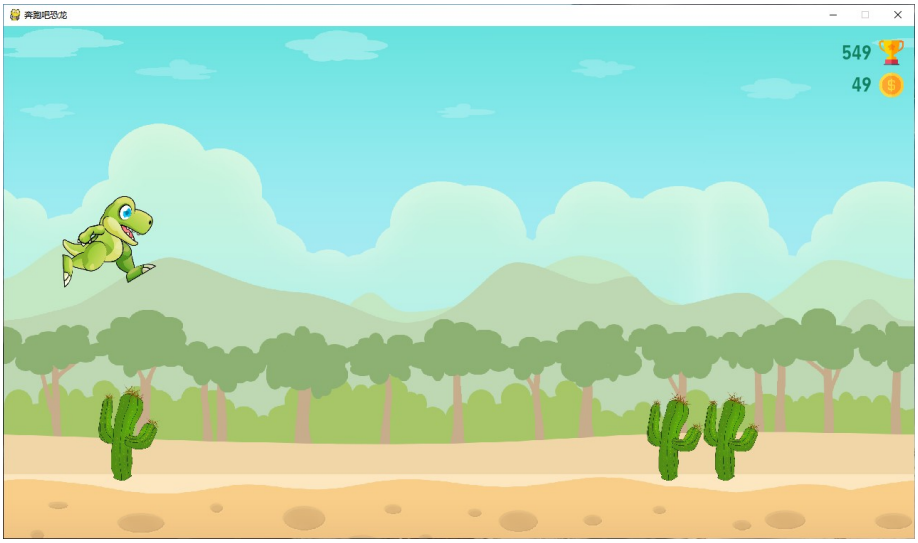

图 2-7

2.2.10 加入游戏结束页面

当检测到恐龙碰撞到障碍物仙人掌后，应该有弹出一个 Game Over 界面，如图 2-8。

```
class GameOver:
    def __init__(self):
        self.replay_image, self.rect =
load_image("image/game_over/replay_0.png", 200, 60)
        self.rect.center = (SCREEN_WIDTH/2, SCREEN_HEIGHT/2)
    def draw(self):
        draw_text("GAME OVER", "font/northcliff_stencil.otf", 80, (255,
0, 0),SCREEN_WIDTH/2, SCREEN_HEIGHT/3, "midtop")
window.blit(self.replay_image, self.rect)
```

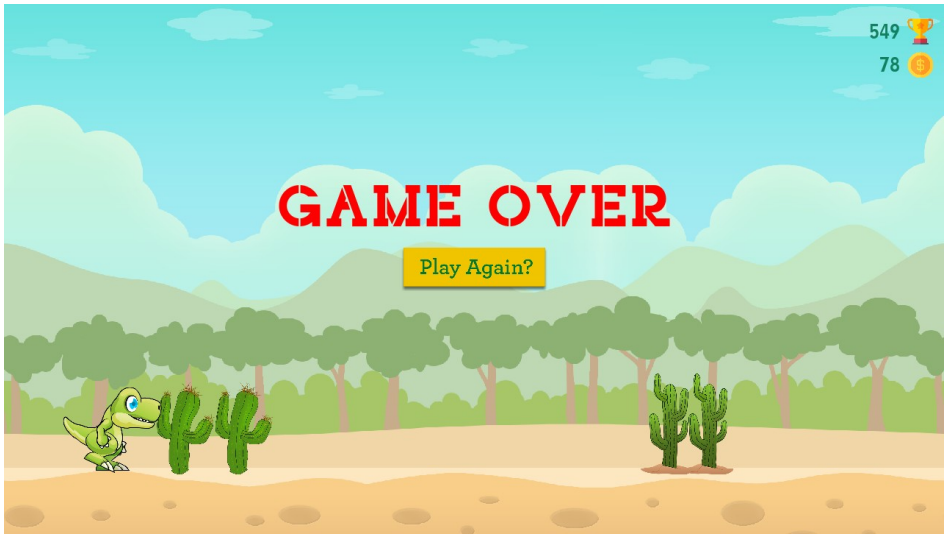

图 2-8 游戏结束画面

第三章 卷积神经网络理论介绍

3.1 人工神经网络概念

人工神经网络（Artificial Neural Network，ANN），人为搭建的神经网络，用输入层模拟人脑的信息输入，经过各个神经元的相互处理运算，传递到下一层神经元，经过一层或多层复杂网络的处理，最后输出处理后的信息。人工神经网络使知识的表现形式更加有特色，智能性强的特点，使它一开始就受到各学科领域的重视。它是一种高度非线性、可进行复杂逻辑运算、可以非线性关系实现的由多个简单元素相互联系构成的复杂网络。

人工神经网络是一个能让计算机处理和优化的数学模型，而生物神经网络是通过刺激，产生新的连接，让信号能够通过新的连接传递而形成反馈。虽然现在的计算机技术越来越高超，不过我们身体里的神经系统经过了数千万年的进化，还是独一无二的，迄今为止，再复杂，再庞大的人工神经网络系统，也不能替换我们的大脑，我们应该感到自豪。

人工神经网络中，各个神经元可以表示不同的处理机制，或一些有意义的抽象模型。根据作用的不同可把神经元分为输入层、隐含层和输出层三层，隐含层的层数是不固定的。输入层接收需要处理的信息；隐含层处于输入层和输出层之间，一般我们在系统外部是无法观察到里面的处理过程和数据的，相当于一个盲盒。输出层输出经过多层神经元计算后的结果；

神经元与神经元之间的连接有一个体现连接强度的数据，称之为权值。ANN对信息的处理是并行的，同时处理多个变量，如图 3-1 是包含两层隐含层的神经网络。

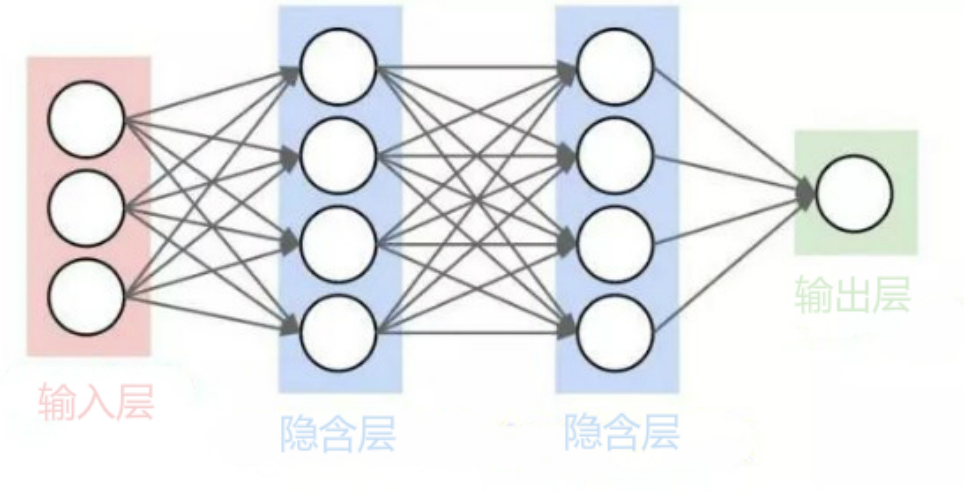

图 3-1 简单神经网络

3.2 卷积神经网络

卷积神经网络(Convolutional Neural Network, CNN)是近些年逐步兴起的一种神经网络结构，它可以提取特征值进行卷积运算，使维度更高的特征被提取出来，减少计算量，加快信息处理能力，同时它也被应用于视频分析、自然语言处理、药物研发，疫情预测，城市大脑等领域.。

3.2.1 卷积层（Convolutional layer）

在卷积神经网络中，卷积层的作用是提取特征，多层卷积层能提取更高维的特征，每层之间都有一个或多个卷积核（滤波器，convolution kernel），第一层卷积层经过卷积运算提取低维特征值送入第二层卷积层中，不断迭代运算，最终输出超高维度的特征值，前提是找到合适的卷积核。

实际应用中，首先要对原始图片数据进行特征提取，然后将所提取的特征输入到全连接网络。卷积能有效的将图片特征提取出来，通常会使用一个正方形的卷积核来遍历图中的每个像素点，有时为了输出和卷积核尺寸相同的图片，需要对图片进行全零填充。图 3-2 模拟卷积运算过程。

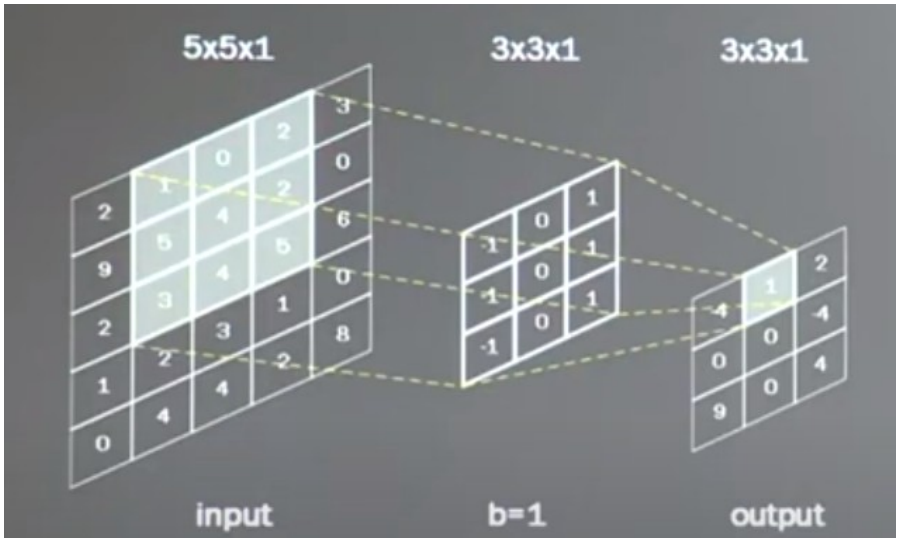

图 3-2 卷积运算

3.2.2 线性整流层（Rectified Linear Units layer）

线性整流层又称修正线性单元，是一种人工神经网络中常用的激活函数（activation function），通常指代以斜坡函数及其变种为代表的非线性函数。以下列出 3 种常见的激活函数图像：

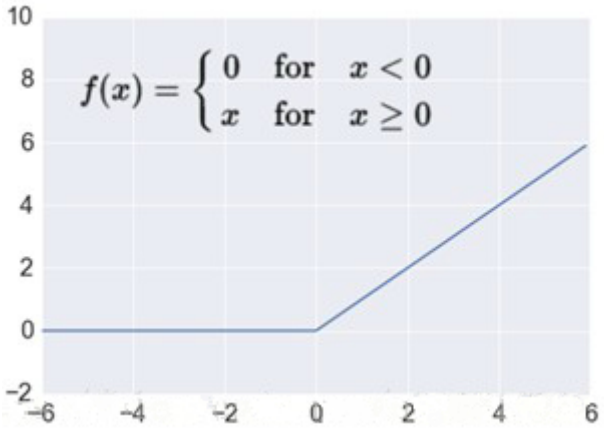

图 3-3 relu 函数

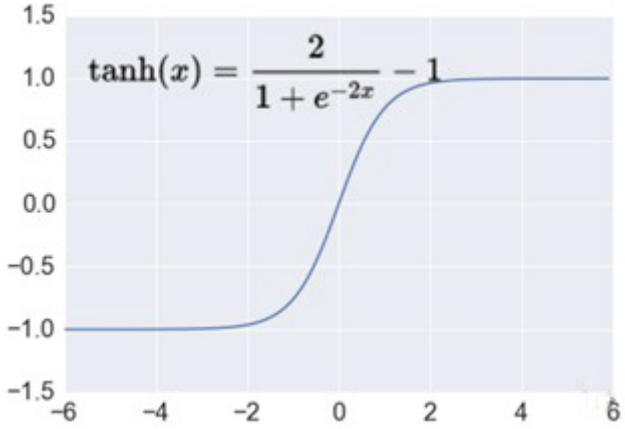

图 3-4 Tanh 函数

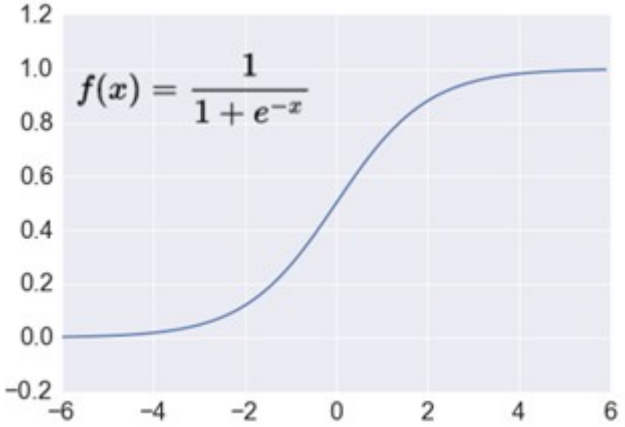

图 3-5 Sigmoid 函数

它们 3 种激活函数的梯度特性各不相同。在饱和区域，Sigmoid 函数和 Tanh 函数的值都无限趋近与一个固定值，也就是说梯度接近于 0，这很容易引起梯度消失的问题，减慢收敛速度。当网络层数逐渐增多，梯度消失的问题就会显现出来。反之，Relu 函数的梯度为常数时，相较于梯度趋于 0 的激活函数，出现深层网络的收敛问题明显减少。Relu 函数有单一的特性，与 Sigmoid 函数和 Tanh 函数相比，更符合生物大脑神经元的特点。sigmoid 函数和 tanh 函数完全可导，特别是在输出层，优势很大。

3.2.3 池化层（Pooling layer）

卷积是为了将大维度的特征提取出来，将特征分割成多个区域，用不同的池化方式减少特征数量。Maxpooling 可提取图片纹理,avergepooling 池化可保留背景特征，还有 stochastic pooling 和 mixed pooling 如图 3-6 列举两种。

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。

如要下载或阅读全文，请访问：

https://d.book118.com/677062056036006060