任务一Python机器学习环境部署实验报告

1. 任务完成摘要

在任务一中，我完成了Python机器学习环境的部署和初步使用。对于安装Python、初步使用Python以及使用Python进行图像处理三个目标，我圆满完成了包括安装Python、验证版本信息、编写函数、绘制正弦曲线、安装Python包，在VS Code和Jupyter Notebook中使用Python、图像处理、拓展图像处理范围和使用ImageTest测试图像处理算法在内的多种任务。在此之上，我还熟悉了flask包，并通过使用flask，解决了我的linux环境中无法使用GUI的缺陷。通过本次实验，我对Python的基本使用、Python绘图和Python在图像处理等方面的应用有了更深入的理解，并且强化了自己解决实际问题的能力。

2. 任务目标

2.1. 安装Python

- 安装 Python 3.9
- 验证 Python 版本信息
- `Hello, world!` in Python
- 编写函数，返回指定范围的斐波那契数列

2.2. 使用Python初步

- 安装 Python 软件包：`numpy`, `matplotlib`
- Python 绘制正弦曲线
- 在 Visual Studio Code 中使用 Python
- 在 Jupyter Notebook 中使用 Python

2.3. Python与图像

- 安装`opencv`，读取并显示图片
- Python 获取摄像头图像，并实时显示
- Python 获取摄像头图像，并实现人脸检测
- 用 Python 对图像进行处理，并显示效果
- 同时支持对MP4视频文件和摄像头捕捉画面进行处理，并显示效果
- 使用 ImageTest 进行图像处理算法的测试

3. 主要内容

环境：Debian 12(Without GUI support), macOS 15

3.1. 安装Python

在Python使用过程中，我们经常会遇到使用不同版本的 Python、不同种类的 Python 包来开发不同项目的情况，为了解决这些 Python 环境的问题，我们可以使用`conda`来进行环境隔离。我使用了 Miniconda 来安装 conda 环境，在Anaconda 的官方文档中，我们可以找到 Miniconda 的安装过程。安装完成后，我们即可新建环境并指定安装 Python 3.9，接着验证版本并打印“Hello, world!”。

```
$ conda activate GC2
$ python3 --version
Python 3.9.19
$ python3
python3
Python 3.9.19 (main, May  6 2024, 14:43:03)
[GCC 11.2.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>print("Hello, world!")
Hello, world!
```

最后我们编写一个返回指定范围的斐波那契数列，这里使用了`yield`来节省开销。

```
def fibonacci(number):
    if number < 0:
        return print("Error!")

    A, B = 0, 1
    for i in range(number + 1):
        yield A
        A, B = B, A + B

if __name__ == "__main__":
    try:
        fibo_length = int(input("Please input the length of fibonacci sequence: "))
        for i in fibonacci(fibo_length):
            print(i)
    except Exception as e:
        print("Error Input:", e)
```

3.2. 使用Python初步

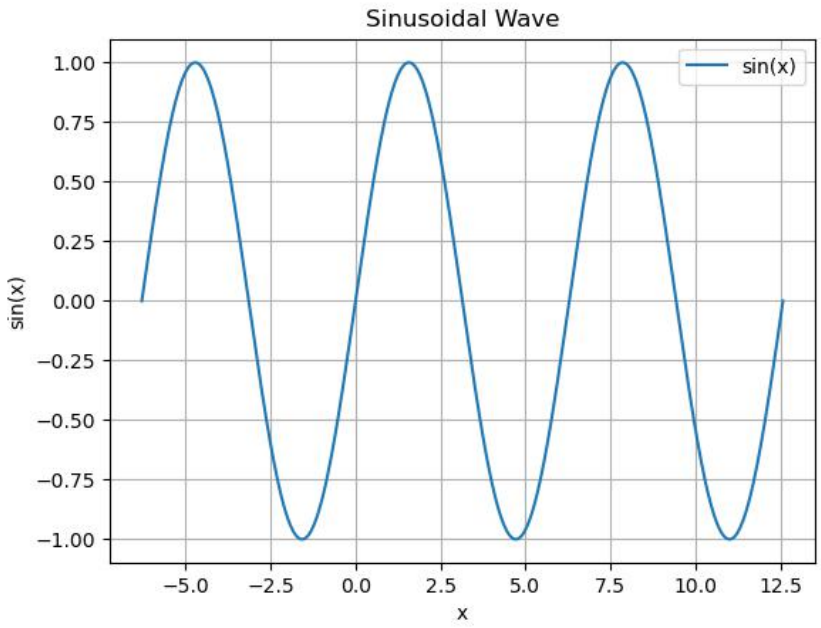
在激活环境后，我们就可以通过`conda`安装 Python 包。

```
$ conda activate GC2
$ conda install numpy
$ conda install matplotlib
$ conda install anaconda  # 可选，一次性安装所有常用包。
```

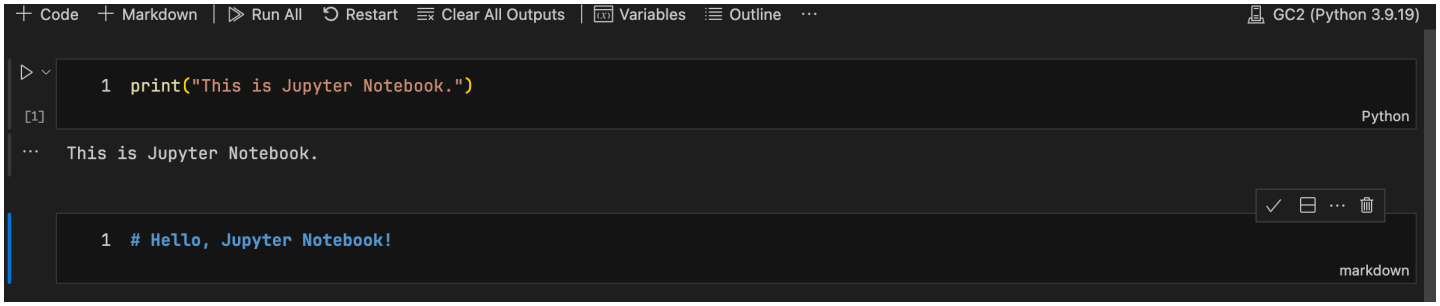
绘画正弦曲线需要刚刚安装的两个包。

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(-2 * np.pi, 4 * np.pi, 500)

y = np.sin(x)

plt.plot(x, y, label="sin(x)")

plt.title("Sinusoidal Wave")
plt.xlabel("x")
plt.ylabel("sin(x)")

plt.grid(True)
plt.legend()
plt.savefig("sin_wave.jpg")
```

运行后我们就可以得到正弦曲线的图片。



在 Visual Studio Code 和 Jupyter Notebook 中使用 Python 也很简单。在VS Code 中，我们需要安装有关 Python 的拓展包，这样即可获得有关 Python 的代码补全、语法高亮、格式修正等等支持，VS Code 也能自动识别系统中的 conda 环境，选择解释器后即可在 VS Code 中直接调用。同样的，安装 Jupyter 相关拓展包，VS Code 即可支持一体式 Jupyter Notebook 编程。我们仅需选择 Python 内核，就可一键运行 Jupyter 服务器，并在 VS Code 中直接查看。



3.3. Python与图像

由于 conda 的内置源并未提供`opencv-python`包，我们需要通过 pip 进行安装，运行`pip install opencv-python`即可安装。

图像识别很基础，我们只需使用`cv2.imread()`和`cv2.imshow()`即可实现。

而获取摄像头画面，则需要我们调用摄像头，并将摄像头的画面逐帧显示。

```
import cv2

def main():
    # 创建 VideoCapture 对象，参数 0 通常表示默认摄像头
    cap = cv2.VideoCapture(0)

    # 检查摄像头是否成功打开
    if not cap.isOpened():
        print("无法打开摄像头")
        return

    while True:
        # 逐帧捕获
        ret, frame = cap.read()

        # 如果正确读取帧，ret 为 True
        if not ret:
            print("无法接收帧，退出 ...")
            break

        cv2.imshow('摄像头画面', frame)

        # 按 'q' 键退出
        if cv2.waitKey(1) & 0xFF == ord('q'):
            print("检测到退出按键，正在退出...")
            break

    # 释放摄像头
    cap.release()
```

```
    # 关闭所有 OpenCV 窗口
    cv2.destroyAllWindows()

if __name__ == "__main__":
    main()
```

对于人脸识别，我们需要对刚刚获取的相机画面进行一些预处理，然后使用 opencv 自带的人脸识别模型进行处理，最后在画面上使用方框对识别出的人脸进行标注，这里只展示人脸识别核心部分的代码。

```
def generate_with_faces():
    for frame in get_frame_from_stream():
        # 对画面进行预处理，将图像转换为灰度图
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        # 应用人脸识别模型
        faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=
(30, 30))

        # 使用方框标注人脸
        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)

        ret, buffer = cv2.imencode('.jpg', frame)
        frame = buffer.tobytes()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
```

对于用 Python 对图像进行处理，同时支持对MP4视频文件和摄像头捕捉画面进行处理，并显示效果，我先创建了有关图像处理的函数。

```
def apply_sepia(image):
    """
    Add image fading effect
    """
    sepia_filter = np.array([[0.272, 0.534, 0.131],
                             [0.349, 0.686, 0.168],
                             [0.393, 0.769, 0.189]])

    sepia_image = cv2.transform(image, sepia_filter)

    sepia_image = np.clip(sepia_image, 0, 255).astype(np.uint8)
    return sepia_image


def add_noise(image):
    """
```

```
    Add noise to picture
    """
    noise_intensity = 25  # Adjust the noise intensity.
    noise = np.random.normal(0, noise_intensity, image.shape).astype(np.uint8)
    noisy_image = cv2.add(image, noise)
    return noisy_image


def apply_blur(image):
    """
    Apply blur effect to image
    """
    blurred_image = cv2.GaussianBlur(image, (5, 5), 0)
    return blurred_image


def adjust_brightness_contrast(image, brightness=-30, contrast=50):
    """
    Adjust the brightness and contrast of image
    """
    adjusted_image = cv2.convertScaleAbs(image, alpha=1 + contrast / 100.0, beta=brightness)
    return adjusted_image


def process_image(image):
    """
    Process image, add the old movie effect.
    """
    sepia_image = apply_sepia(image)
    # noisy_image = add_noise(sepia_image) # 对性能消耗较大
    noisy_image = sepia_image
    blurred_image = apply_blur(noisy_image)
    final_image = adjust_brightness_contrast(blurred_image)

    return final_image
```

然后使用三个函数分别处理图片、视频和摄像头画面。

```
def process_single_image(image_path, output_dir):
    """
    Process for a single image
    """
    image = cv2.imread(image_path)
    # Apply the old movie effect.
    processed_image = process_image(image)

    os.makedirs(output_dir, exist_ok=True)
    output_path = os.path.join(output_dir, os.path.basename(image_path))
```

```
    cv2.imwrite(output_path, processed_image)


def process_video(video_path, output_dir):
    """
    Process every frame in a video, and save the result as a new video.
    """
    cap = cv2.VideoCapture(video_path)

    fps = int(cap.get(cv2.CAP_PROP_FPS))
    width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    os.makedirs(output_dir, exist_ok=True)
    output_path = os.path.join(output_dir, os.path.basename(video_path))
    out = cv2.VideoWriter(output_path, fourcc, fps, (width, height), isColor=True)

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        processed_frame = process_image(frame)
        out.write(processed_frame)

    # Release the memory.
    cap.release()
    out.release()

def generate_processed():
    for frame in get_frame_from_stream():
        frame = process_image(frame)

        ret, buffer = cv2.imencode('.jpg', frame)
        frame = buffer.tobytes()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
```

最后，使用 os 包来对指定目录中的所有图片和视频进行识别，根据其类型调用相关的处理函数。

```
def process_directory(input_dir, output_dir):
    for root, dirs, files in os.walk(input_dir):
        for file in files:
            print(file)
            file_path = os.path.join(root, file)

            if file.endswith(('.jpg', '.jpeg', '.png', '.bmp')):
                process_single_image(file_path, output_dir)
            elif file.endswith(('.mp4', '.avi', '.mov', '.mkv')):
                process_video(file_path, output_dir)
```

最后是使用 ImageTest 进行图像处理算法的测试，在下载 airemote Repo 并安装`inferemote`包后，我们即可使修改`/airemote/python/misc`目录下的`image_test.py`进行测试。

```
#!/usr/bin/env python3
"""
Inferemote: a Remote Inference Toolkit for Ascend 310

"""
''' Prepare a test '''
import cv2 as cv
from image_process import process_image  # 这里我直接调用了刚刚创建的图像处理函数

# 颜色反转/高斯双边滤波
# def bilateral_filter(image):
#     image = cv.bitwise_not(image)
#     image = cv.bilateralFilter(image, 0, 100, 15)
#     return image

from inferemote.testing import ImageTest
class MyTest(ImageTest):
    ''' Define a callback function for inferencing, which will be called for every single
image '''
    def run(self, image):
        ''' an image must be returned in the same shape '''
        # new_img = bilateral_filter(image) # 修改此处更换图像处理方式
        new_img = process_image(image)
        return new_img

if __name__ == '__main__':

    t = MyTest(mode='liveweb', threads=1)
    t.start(input='/Users/haojiash/skiing.mp4', mode='show')
```

接着，即可在 shell 中运行。

```
$ python3 image_test.py -i ~/Downloads/oceans.mp4
```

4. 主要思路及关键步骤

4.1. 斐波那契数列函数初始值的影响

```
def fibonacci(number):
    if number < 0:
        return print("Error!")

    A, B = 0, 1
    for i in range(number + 1):
        yield A
        A, B = B, A + B

if __name__ == "__main__":
    try:
        fibo_length = int(input("Please input the length of fibonacci sequence: "))
        for i in fibonacci(fibo_length):
            print(i)
    except Exception as e:
        print("Error Input:", e)
```

对于这个函数，其实我们可以将 A 和 B 的初始值反过来，然后对 B 进行`yield`操作。

```
def fibonacci(number):
    if number < 0:
        return print("Error!")

    # 交换A，B的初始值
    A, B = 1, 0
    for i in range(number + 1):
        yield B
        A, B = B, A + B

if __name__ == "__main__":
    try:
        fibo_length = int(input("Please input the length of fibonacci sequence: "))
        for i in fibonacci(fibo_length):
            print(i)
    except Exception as e:
        print("Error Input:", e)
```

这样做，函数计算出的数值仍然是正确的，而且还将最后一次无用的`A + B`赋值优化掉了。

4.2. 使用Flask解决无GUI支持的Linux显示图片、视频与摄像头画面问题

在进行第三个目标时，我遇到了很大的问题。我的Linux远程服务器根本不支持GUI，所有能直接显示GUI的函数都没有用，于是，我就尝试了Flask，借助前端的力量解决这个问题，最后也确实解决掉了。

首先，是对图片的显示。

```
# Display an image using OpenCV and Flask.
import cv2
import os
import base64
from flask import Flask, render_template_string

os.chdir("/root/projects/GC2/task/task01")
app = Flask(__name__)

def get_image_base64(image_path):
    image = cv2.imread(image_path)
    _, buffer = cv2.imencode('.jpg', image)
    # 使用base64编码来在HTML中显示图片
    image_base64 = base64.b64encode(buffer).decode('utf-8')

    return image_base64

@app.route('/')
def index():
    image_path = './resource/macintosh.jpg'  # Path to image.

    image_base64 = get_image_base64(image_path)

    html_content = f'''
    <html>
    <head><title>Image Display</title></head>
    <body>
        <h1>Hello, Flask!</h1>
        <img src="{image_base64}" alt="Image">
    </body>
    </html>
    '''

    return render_template_string(html_content)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

运行后，访问`localhost:5000`，结果是无法访问。Flask服务器运行在远程的Linux上，和本地端的端口并不互通。于是，我又需要在使用ssh连接Linux远程服务器时对端口进行转发。用`man ssh`看看手册，查询到`-L`这个 flag 能将远程服务器指定端口转发到本地的指定端口上。

```
$ ssh -L 5000:localhost:5000 XXXX@X.X.X.X
```

这下终于能正常访问了。

视频的显示和摄像头画面显示同理，这里我只说明如何实现摄像头画面的显示。

和图片不同的是，摄像头的显示需要两个过程，一是我需要把本地的摄像头画面传输到远程的Linux服务器上，二才是远程的Linux服务器处理完摄像头画面后显示到HTML页面上。所以，这次我需要同时在本地也使用Flask。

```
import cv2
import threading
from flask import Flask, Response

app = Flask(__name__)

video_capture = None

def capture_frames():
    global video_capture
    video_capture = cv2.VideoCapture(0)

    video_capture.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
    video_capture.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)


def generate():
```

```
    global video_capture
    while True:
        success, frame = video_capture.read()
        if not success:
            break
        else:
            encode_param = [int(cv2.IMWRITE_JPEG_QUALITY), 25]
            ret, buffer = cv2.imencode('.jpg', frame, encode_param)
            frame = buffer.tobytes()

            yield (b'--frame\r\n'
                   b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

@app.route('/video_feed')
def video_feed():
    return Response(generate(),
                    mimetype='multipart/x-mixed-replace; boundary=frame')

if __name__ == '__main__':
    threading.Thread(target=capture_frames).start()
    app.run(host='0.0.0.0', port=4500
```

对应的，远程服务器需要从`localhost:4500`处拿到视频流，经过处理后在HTML页面上显示，并让本地端通过`localhost:5000`访问。

```
import cv2
import numpy as np
import requests
from flask import Flask, Response, render_template_string
from image_process import process_image

app = Flask(__name__)

video_feed_url = 'http://localhost:4500/video_feed'

def get_frame_from_stream():
    stream = requests.get(video_feed_url, stream=True)
    bytes_data = b''
    for chunk in stream.iter_content(chunk_size=1024):
        bytes_data += chunk
        a = bytes_data.find(b'\xff\xd8')
        b = bytes_data.find(b'\xff\xd9')
        if a != -1 and b != -1:
            jpg = bytes_data[a:b + 2]
            bytes_data = bytes_data[b + 2:]
            img = cv2.imdecode(np.frombuffer(jpg, dtype=np.uint8), cv2.IMREAD_COLOR)
            yield img
```

```
def generate_original():
    for frame in get_frame_from_stream():
        ret, buffer = cv2.imencode('.jpg', frame)
        frame = buffer.tobytes()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

def generate_processed():
    for frame in get_frame_from_stream():
        frame = process_image(frame)

        ret, buffer = cv2.imencode('.jpg', frame)
        frame = buffer.tobytes()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

@app.route('/original_feed')
def original_feed():
    return Response(generate_original(),
                    mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/processed_feed')
def processed_feed():
    return Response(generate_processed(),
                    mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/')
def index():
    return render_template_string('''
    <html>
    <head>
        <title>Old Movie Effect</title>
    </head>
    <body>
        <h1>Live Video Stream</h1>
        <div style="display: flex;">
            <div style="flex: 1; margin-right: 10px;">
                <h2>Original Stream</h2>
                <img src="{{ url_for('original_feed') }}" width="100%">
            </div>
            <div style="flex: 1;">
                <h2>Old Movie Stream</h2>
                <img src="{{ url_for('processed_feed') }}" width="100%">
            </div>
        </div>
    </body>
    </html>
    ''')
```

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

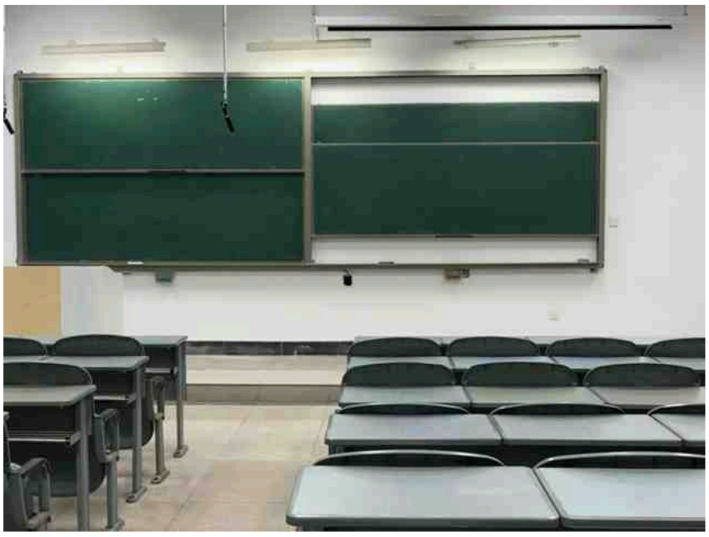
这次，我们还需要将4500端口也进行转发，使用`-R`参数将本地4500端口转发到远程服务器。

```
$ ssh -L 5000:localhost:5000 -R 4500:localhost:4500 XXXX@X.X.X.X
```

两边同时运行，来看看页面，右侧画面成功加上了我设定的老电影效果。

Live Video Stream

Original Stream



Old Movie Stream

5. 完成情况与结果分析

所有任务目标全部完成。

所有任务实现效果良好，对GUI展示无法实现的问题使用了Flask来解决，最终展示效果良好。

6. 总结

通过任务一，我完成了Python机器学习环境的部署和基础功能的实现，还了解并使用了一些在机器学习中常用的包，如经常被用来绘图的`matplotlib`、提供强大数值计算功能的`numpy`、封装有非常多种常用图像处理函数的`opencv`等等，同时，我也使用了 VS Code 用来便捷的开发 Python 程序，并使用了 Jupyter Notebook 的交互式开发功能。对无法使用 GUI 界面的问题，我也搜集资料并使用 Flask 进行解决。任务一为我们后续学习在 Mindspore 框架下的深度学习奠定了良好的基础，也激发了我对计算机科学专业的强烈兴趣。