# 前端代码部分

## BaseHeader.vue

<script setup lang="ts">

import {toggleDark} from '~/composables';

import {useDark} from '@vueuse/core'

const isDark = useDark()

</script>

<template>

<el-menu default-active="content" :router="true" class="el-menu-demo" mode="horizontal">

<el-menu-item index="/content">白内障患者眼底影像增强系统</el-menu-item>

<el-menu-item index="/content">首页</el-menu-item>

<el-menu-item index="/team">团队介绍</el-menu-item>

<el-menu-item>

<div class="toggle" @click.stop="toggleDark()">

<button class="border-none w-full bg-transparent cursor-pointer" style="height: var(--ep-menu-item-height);">

<i inline-flex i="dark:ep-moon ep-sunny" />

</button>

</div>

</el-menu-item>

</el-menu>

</template>

<style>

.logo\_image {

padding-top: 6px;

padding-left: 15px;

padding-right: 15px;

}

</style>

## Content.vue

<template>

<div class="image\_upload">

<el-card style="border-radius: 8px; width: 100%">

<template #header>

<span style="margin-left: -86%; font-weight: bold">眼底图片上传</span>

</template>

<el-upload

v-if="!loading"

v-model:file-list="fileList"

class="upload\_box"

drag

action="https://run.mocky.io/v3/9d059bf9-4660-45f2-925d-ce80ad6c4d15"

multiple

accept=".jpg, .jpeg, .png, .tiff"

:limit="5"

:list-type="'text'"

:auto-upload="false"

:on-error="handleUploadError"

:on-exceed="handleExceed"

:on-success="handleUploadSuccess"

:with-credentials="true"

:on-change="handChange"

>

<el-icon class="el-icon--upload" size="50px">

<upload-filled/>

</el-icon>

<div class="el-upload\_\_text">

拖动图片至此处或点击选择图片

</div>

<template #tip>

<div class="el-upload\_\_tip" style="margin-top: 10px">

请上传最多五张png文件

</div>

</template>

</el-upload>

<div class="loading\_bar" v-if="loading" style="margin-top: 7%; margin-bottom: 9%">

<h3>正在增强眼底图像，请耐心等待</h3>

<el-progress

v-if="loading"

:stroke-width="12"

show-text="show-text"

:percentage="percentage"

:status="percentage === 100 ? 'success' : undefined"/>

</div>

</el-card>

</div>

<div>

</div>

<div style=" padding: 15px" class="record">

<!-- 题目-->

<div style="margin: 15px">

模型选择

</div>

<div style="margin: 15px">

<el-select v-model="model" placeholder="选择模型" class="select">

<el-option

v-for="item in options"

:key="item.value"

:label="item.label"

:value="item.value"

/>

</el-select>

</div>

<el-button type="primary" :disabled="fileList.length < 1 || model === ''" @click="uploadImage">

上传图片

</el-button>

<!-- 题目-->

<!-- 功能区-->

<el-table :data="tableData" style="margin-top: 5%">

<el-table-column fixed prop="date" label="上传时间" align="center" header-align="center"/>

<el-table-column prop="model" label="模型" align="center" header-align="center"/>

<el-table-column prop="name" label="图片名" align="center" header-align="center"/>

<el-table-column prop="raw\_image" label="原始图像" align="center" header-align="center">

<template #default="scope">

<el-image :src="scope.row.raw\_image"

:preview-src-list="[scope.row.raw\_image]" preview-teleported="true">

<div slot="error" class="image-slot">

<i class="el-icon-picture-outline"></i>

</div>

</el-image>

</template>

</el-table-column>

<el-table-column prop="processed\_image" label="结果预览" align="center" header-align="center">

<template #default="scope">

<el-image :src="scope.row.processed\_image"

:preview-src-list="[scope.row.processed\_image]" preview-teleported="true">

<div slot="error" class="image-slot">

<i class="el-icon-picture-outline"></i>

</div>

</el-image>

</template>

</el-table-column>

<el-table-column fixed="right" label="操作" align="center" header-align="center">

<template #default="scope">

<el-button type="primary" @click="download(scope.row.real\_name)">下载</el-button>

</template>

</el-table-column>

</el-table>

<el-dialog v-model="dialogVisible" title="记录">

记录

<el-button link type="primary" size="small" @click="certainInR">

</el-button>

</el-dialog>

</div>

</template>

<script>

import {ref} from "vue";

import {ElMessage, ElMessageBox} from 'element-plus'

import axios from "axios";

axios.defaults.baseURL = 'http://127.0.0.1:5003'

export default {

data() {

return {

indeterminate: true,

show\_text: false,

interval: '',

isUploading: false,

starttime: 0,

currenttime: 0,

averagetime: 2500,

percentage: 0,

model: '',

fileList: [],

nowtime: '',

dialogVisible: false,

loading: false,

resCount: 0,

resNum: 0,

options: [

{

value: 'ArcNet',

label: 'ArcNet',

},

{

value: 'RCDG\_model',

label: 'RCDG',

}

],

tableData: [],

Filename: ''

}

},

methods: {

findPercentage() {

this.currenttime += 500

console.log(this.currenttime)

if (this.resNum === this.resCount) {

this.percentage = 100

}

if ((Math.floor(this.currenttime - this.starttime) / (this.averagetime \* this.resCount)) >= 1) {

this.percentage = 99

} else {

this.percentage = Math.floor((this.currenttime - this.starttime) / (this.averagetime \* this.resCount) \* 100)

}

},

handleExceed() {

ElMessage.warning('一次最多上传五张图片！')

},

handleUploadError() {

ElMessage.error('上传失败！')

},

handleUploadSuccess() {

ElMessage.success('上传成功！')

},

handChange(file, fileList) {

this.fileList = fileList

},

checkRecord() {

this.dialogVisible = true

},

certainInR() {

this.dialogVisible = false

},

load(filename, sha1, raw\_image, model\_name) {

axios.get("show/" + sha1, {responseType: "blob"}).then(res => {

this.resNum += 1

console.log(this.resNum + " " + this.resCount)

if (this.resNum === this.resCount) {

this.loading = false

this.resNum = 0

clearInterval(this.interval)

this.percentage = 0

this.currenttime = 0

}

console.log(this.starttime)

console.log(this.currenttime)

let year = new Date().getFullYear()

let month = new Date().getMonth() + 1

let day = new Date().getDate()

month = month < 10 ? '0' + month : month

day = day < 10 ? '0' + day : day

let hour = new Date().getHours()

let minuite = new Date().getMinutes()

hour = hour < 10 ? '0' + hour : hour

minuite = minuite < 10 ? '0' + minuite : minuite

this.nowtime = year + '/' + month + '/' + day + ' ' + hour + ':' + minuite

if (res.status === 200) {

const im = window.URL.createObjectURL(res.data)

console.log(res);

this.tableData.push(

{

date: this.nowtime,

model: model\_name,

real\_name: sha1,

name: filename.replace(/\.[^/.]+$/, ''),

processed\_image: im,

raw\_image: raw\_image

}

)

}

})

},

async uploadImage() {

this.starttime = 0

this.loading = true

this.resCount = this.fileList.length

this.interval = setInterval(this.findPercentage, 500)

for (let file of this.fileList) {

try {

let fileParam = new FormData();

fileParam.append("file", file["raw"]);

fileParam.append("fileName", file["name"]);

await axios.post('upload/' + this.model, fileParam).then(

(response) => {

let raw\_image = window.URL.createObjectURL(file["raw"])

this.load(file["name"], response.data.filename, raw\_image, this.model)

}

)

} catch (e) {

console.log(e)

}

}

this.fileList = []

},

download(filename) {

try {

axios.get("download/" + filename, {responseType: "blob"}).then(res => {

const url = window.URL.createObjectURL(res.data)

let link = document.createElement("a");

link.style.display = "none";

link.href = url;

link.setAttribute("download", filename);

document.body.appendChild(link);

link.click();

document.body.removeChild(link);

window.URL.revokeObjectURL(url);

if (res.status === 200) {

ElMessage.success("下载成功")

} else {

ElMessage.error("下载失败，请重试")

}

})

} catch (e) {

ElMessage.error("下载失败，请重试")

}

}

}

}

</script>

<style scoped>

.image\_upload {

width: 80%;

display: flex;

justify-content: center;

align-items: center;

margin-left: 8%;

}

.record {

width: 80%;

margin-left: 8%;

}

</style>

## instruction.vue

<template>

<div class="aside">

<el-card class="inst" style="border-radius: 8px" shadow="never">

<template #header>

<span style="margin-left: -50%; font-weight: bold">使用指南</span>

</template>

<div style="height: 180px; margin-left: 15px; margin-top: 15px">

<el-steps direction="vertical" active="3">

<el-step title="上传图像"/>

<el-step title="选择模型"/>

<el-step title="等待结果"/>

</el-steps>

</div>

</el-card>

</div>

</template>

<script lang="ts">

export default {

name: "instruction"

}

</script>

<style scoped>

.aside{

margin-left: 15%;

margin-top: 15%;

}

</style>

## team.vue

<template>

<h1 style="color: #3039a1 ;font-size: 50px ; font-family: 微软雅黑; margin-top: -5px; margin-bottom: -5px">

IMED团队介绍</h1>

<el-divider/>

<div style="display: flex">

<div id="container">

<div id="left" :style="{ 'background-image':`url(${currentImg})` }">

<!-- <img :src="currentImg"/>-->

</div>

<div id="right">

<div v-for="imgSrc in imglist" @mouseenter="setCurrent(imgSrc)" :key="imgSrc" class="item">

<img :src="imgSrc" :style="{opacity: imgSrc === currentImg?0:1}"/>

</div>

</div>

</div>

<div style="width: 40%">

<div

style="width: 100%; padding-left: 50px;margin: 50px 0 ;text-align: left ;font-family: 微软雅黑 ; text-indent: 2em">

iMED 是智能医疗影像处理Intelligent Medical

Imaging的简称，iMED团队是由刘江教授于2007年1月创建于新加坡科研局，是一个以专注眼科人工智能的国际化团队，目前由iMED深圳、iMED宁波、iMED新加坡团队组成。

2007-2016年内iMED新加坡团队从创立1人发展到26余位科学家组成的当时世界上最大的眼科人工智能团队之一，成员包括来自新加坡，中国，马来西亚，德国以及印度的科学家。

包括现在活跃于眼科人工智能领域的DAMON WONG，许言午，付华柱，段立新，刘慧颖，程骏，JIMMY LEE等。

</div>

<div

style=" width: 100%;padding-left: 50px;margin-top: 20px ;text-align: left ;font-family: 微软雅黑 ; text-indent: 2em">

2016年3月，刘江教授及其部分iMED新加坡团队核心成员来到中国发展，刘江教授落户宁波/慈溪，本着iMED做大做强的原则，

仅用3年，在任中国科学院宁波慈溪医工所首任所长的同时，吸引了英国归国的赵一天博士，美国归国的杨建龙博士，

日本归国的胡衍博士, 新加坡归国的程骏、JIMMY LEE博士等，在医工所重新创建了一个约40人的眼科人工智能研究团队iMED宁波。

</div>

</div>

</div>

<div

style="width: 90% ;padding-left: 50px;margin-top: 20px ;text-align: left;font-family: 微软雅黑 ; text-indent: 2em">

2019年2月，刘江教授来到了中国改革前沿城市深圳，加入了南方科技大学计算机系，开始了iMED的第3次创业。

本着做精做深的新理念原则创建了iMED 深圳 （https://faculty.sustech.edu.cn/liuj/），iMED深圳团队已经在短短时间形成包括研究助理教授胡衍领导的手术导航团队，

资深研究学者东田理莎教授领导的眼前节团队，包括缪函霈眼科医生和研究助理教授金日初，博后姜弘羊在内的眼脑联动团队，以及章晓庆博士生领导的跨媒体4个团队。

形成了包括教授，研究教授，博后，博士，硕士，工程师，实验室科研本科生在内的布局优化的100多人团队。

与此同时，2022年开始，赵一天研究员正式接手iMED宁波团队（https://imed.nimte.ac.cn/），

付华柱资深科学家也在新加坡科研局负责继续发展iMED新加坡团队。

在“团结，专注，坚持”的iMED团队文化凝结下， iMED作为一个整体正在有质量地在眼科人工智能方向持续发展。

</div>

<h1 style="color: #3039a1 ;font-size: 50px; font-family: 微软雅黑; margin-top: 50px; margin-bottom: -5px">

模型介绍</h1>

<el-divider/>

<h2 style="color: #3039a1 ;font-size: 30px ; font-family: 微软雅黑 ;">ArcNet</h2>

<div style="left: 50px; display: flex">

<img src="../assets/arcnet.png" style="padding-left: 50px;margin-top: 20px;margin-bottom: 20px">

<div

style="width: 40%; padding-left: 50px ; margin-top: 20px;margin-bottom: 20px; text-indent: 2em; text-align: left">

该模型主要由图像生成和领域自适应两个模块组成，其具体结构如左图所示。模型通过在清晰眼底图像上叠加模拟白内障噪声生成配对的图像，解决了现有手术前后配对图像稀缺的问题。

为了使模型在真实数据集上也能有良好表现，模型加入了领域自适应模块。将真实白内障图片输入共享生成器G，得到增强图像t^,

将真实白内障增强图t^和模拟白内障增强图 sˆ

输入辨别器D\_d，通过训练使原域逐渐向向目标域靠近。 此模型的性能超越了以往的的增强模型，并在真实场景 下也有良好的表现。

</div>

</div>

<h2 style="color: #3039a1 ;font-size: 30px ; font-family: 微软雅黑 ;">RCDG</h2>

<div style="left: 50px; display: flex">

<img src="../assets/rcdg.png" style="padding-left: 50px;margin-top: 20px;margin-bottom: 20px; width: 45%">

<div

style="width: 40%; padding-left: 50px ; margin-top: 20px;margin-bottom: 20px; text-indent: 2em; text-align: left">

该模型通过从清晰的眼底图像模拟白内障患者的眼底 图像来构建训练的源域，并通过随机调整模拟图像的参

数生成不同风格的模拟图像来覆盖潜在的目标域，实现领域泛化。

同时将图像的高频成分作为域不变特征，减小域偏移，对齐源域中的各个域，提高模型泛化性能。

模型的基本结构如图所示。先在清晰图像s上增加模拟白内障噪声生成模拟图像，在提取模拟图像的高频成分。

通过生成器G\_H重建高频成分，最后通过生成器G\_R得到恢复后的图像。

</div>

</div>

<h1 style="color: #3039a1 ;font-size: 50px ; font-family: 微软雅黑 ;margin-bottom: -5px">模型参考文献</h1>

<el-divider/>

<div style="padding-left: 50px ;text-align: left; font-family: 'Times New Roman'; font-style: italic">

[1] Li H , Liu H , Hu Y , et al. Restoration Of Cataract Fundus Images Via Unsupervised Domain Adaptation[C]// 2021

IEEE 18th International Symposium on Biomedical Imaging (ISBI). IEEE, 2021.

</div>

<div style="padding-left: 50px ;text-align: left; font-family: 'Times New Roman'; font-style: italic">

[2] Li H , Liu H , Hu Y , et al. An Annotation-free Restoration Network for Cataractous Fundus Images[J]. 2022.

</div>

</template>

<script setup lang="ts">

import img1 from '../assets/IMED1.png'

import img2 from '../assets/IMED2.png'

import img3 from '../assets/IMED3.png'

import img4 from '../assets/IMED4.png'

import {ref} from "vue";

const imglist = ref([

img1,

img2,

img3,

img4,

])

const currentImg = ref(imglist.value[0])

function setCurrent(imgSrc: string) {

currentImg.value = imgSrc

}

</script>

<style scoped>

#container {

margin-top: 50px;

margin-bottom: 20px;

padding-left: 50px;

padding-right: 50px;

width: 500px;

height: 300px;

display: flex;

}

#left {

flex: 3;

/\*background-image: url("../assets/logo.png");\*/

background-position: center;

background-size: cover;

}

#right {

flex: 1;

}

.item {

display: flex;

flex-direction: column;

margin: 2px;

transition: opacity .2s ease-in-out;

}

.item img {

width: 100%;

height: 100%;

}

.item:hover {

opacity: 0;

}

</style>

## dark.ts

import { useDark, useToggle } from '@vueuse/core'

export const isDark = useDark()

export const toggleDark = useToggle(isDark)

## index.ts

export \* from './dark'

## index.js

import {createRouter, createWebHashHistory, createWebHistory} from 'vue-router'

import head from '../components/layouts/BaseHeader.vue'

import content from "../components/Content.vue";

import team from "../components/team.vue"

const routes = [

{

path:'/',

component: content,

},

{

path: '/content',

component: content,

},{

path:'/team',

component: team,

}

]

const router = createRouter({

history: createWebHashHistory(),

routes

})

export default router

## dark.scss

$--colors: (

"primary": (

"base": #fc6509,

),

);

@forward "element-plus/theme-chalk/src/dark/var.scss" with (

$colors: $--colors

);

## index.scss

// :root {

// --ep-color-primary: red;

// }

body {

font-family: "Helvetica Neue", Helvetica, "PingFang SC", "Hiragino Sans GB",

"Microsoft YaHei", "微软雅黑", Arial, sans-serif;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

margin: 0;

}

a {

color: var(--ep-color-primary);

}

code {

border-radius: 2px;

padding: 2px 4px;

background-color: var(--ep-color-primary-light-9);

color: var(--ep-color-primary);

}

## App.vue

<template>

<el-config-provider namespace="ep">

<el-container>

<el-header>

<BaseHeader/>

</el-header>

<el-container>

<el-aside width="20%">

<instruction class="instruction"/>

</el-aside>

<el-main>

<!-- <Content class="content"/>-->

<router-view class="main"/>

</el-main>

</el-container>

</el-container>

</el-config-provider>

</template>

<style>

#app {

text-align: center;

color: var(--ep-text-color-primary);

}

.main {

display: flex;

justify-content: center;

align-items: center;

}

</style>

<script lang="ts">

export default {

name : "App",

}

</script>

## main.ts

import { createApp } from "vue";

import App from "./App.vue";

import \* as ElIconModules from '@element-plus/icons-vue'

import 'element-plus/theme-chalk/dark/css-vars.css';

// import "~/styles/element/index.scss";

// import ElementPlus from "element-plus";

// import all element css, uncommented next line

// import "element-plus/dist/index.css";

// or use cdn, uncomment cdn link in `index.html`

import "~/styles/index.scss";

import 'uno.css'

// If you want to use ElMessage, import it.

import "element-plus/theme-chalk/src/message.scss"

import router from "../src/router"

const app = createApp(App);

// app.use(ElementPlus);

app.use(router)

for(let iconName in ElIconModules){

// @ts-ignore

app.component(iconName,ElIconModules[iconName])

}

app.mount("#app");

## index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<!-- <link rel="icon" href="src/assets/logo.png" />-->

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>白内障患者眼底影像增强系统</title>

<!-- element css cdn, if you use custom theme, remove it. -->

<!-- <link

rel="stylesheet"

href="https://cdn.jsdelivr.net/npm/element-plus/dist/index.css"

/> -->

</head>

<body>

<div id="app"></div>

<script type="module" src="/src/main.ts"></script>

</body>

</html>

# 后端代码部分

## app.py

import os

from flask import \*

import hashlib

from core.main import process

UPLOAD\_FOLDER = os.path.join('data', 'unprocessed')

ALLOWED\_EXTENSIONS = {'png', 'jpg', 'jpeg', 'tiff', 'JPG'}

app = Flask(\_\_name\_\_)

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

# 添加header解决跨域

@app.after\_request

def after\_request(response):

response.headers['Access-Control-Allow-Origin'] = '\*'

response.headers['Access-Control-Allow-Credentials'] = 'true'

response.headers['Access-Control-Allow-Methods'] = 'POST'

response.headers['Access-Control-Allow-Headers'] = 'Content-Type, X-Requested-With'

return response

@app.route('/')

def hello\_world():

return 'Hello World!'

def allowed\_file(filename):

return '.' in filename and filename.rsplit('.', 1)[1] in ALLOWED\_EXTENSIONS

@app.route('/upload/<model\_name>', methods=['GET', 'POST'])

def upload\_file(model\_name):

file = request.files['file']

msg = ''

if file and allowed\_file(file.filename):

src\_path = os.path.join(app.config['UPLOAD\_FOLDER'], file.filename)

file.save(src\_path)

with open(src\_path, 'rb') as f:

sha1 = hashlib.sha1()

while True:

block = f.read(65536)

if not block:

break

sha1.update(block)

hash\_value = sha1.hexdigest()

file\_name = hash\_value + os.path.splitext(file.filename)[1]

sha1\_path = os.path.join(app.config['UPLOAD\_FOLDER'], file\_name)

os.rename(src\_path, sha1\_path)

process(sha1\_path, model\_name)

# shutil.copy(src\_path, 'data/processed')

# msg = core.main.process(src\_path, model\_name)

# if msg == 'Success':

return jsonify({'filename':file\_name})

app.logger.info("Failed to deal with image!\n", msg)

return jsonify({'filename':file\_name})

# show photo

@app.route('/show/<path:file>', methods=['GET'])

def show\_photo(file):

# print(file)

if file is None:

app.logger.info('file is None')

abort(401)

else:

try:

with open(f'data/processed/{file}', "rb") as f:

image\_data = f.read()

response = make\_response(image\_data)

response.headers['Content-Type'] = 'image/png'

return response

except:

app.logger.info('Exception!')

abort(401)

@app.route("/download/<path:file>", methods=['GET'])

def download\_file(file):

# 需要知道2个参数, 第1个参数是本地目录的path, 第2个参数是文件名(带扩展名)

return send\_from\_directory('data/processed', file, as\_attachment=True)

if \_\_name\_\_ == '\_\_main\_\_':

# with app.app\_context():

# current\_app.model = init\_model()

app.run(host='127.0.0.1', port=5003, debug=True)

## main.py

import os

from core.predict import predict

from core.process import preprocess

from PIL import Image

def process(file\_path, model\_name):

"""

:param file\_path: the path of cataract fundus image

:param model\_name: the model you want to use

"""

# TODO: fill this

image = Image.open(file\_path)

image = image.resize((512, 512))

file\_name = file\_path.split('/')[-1].split('.')[0]

image.save('data/dataset/target/image1.png')

preprocess('data/dataset/target/image1.png', 'data/dataset/target\_mask/image1.png')

if model\_name == 'RCDG\_model':

predict('RCDG\_model')

img = Image.open('results/RCDG\_drive/test\_latest/images/image1\_fake\_TB.png')

img = img.resize((512, 512))

img.save(str(f'data/processed/{file\_name}.png'))

# os.remove('data/dataset/target/image1.png')

# os.remove('data/dataset/target\_mask/image1.png')

elif model\_name == 'ArcNet':

predict('ArcNet')

img = Image.open('results/arcnet/test\_latest/images/image1\_fake\_TB.png')

# img = img.resize((512, 512))

img.save(str(f'data/processed/{file\_name}.png'))

else:

return 'No such model'

return 'Success'

if \_\_name\_\_ == '\_\_main\_\_':

process('data/unprocessed/new\_test.jpg', "AecNet")

# image = Image.open('../data/unprocessed/test.jpg')

## predict.py

import os

import torch

# import model run in server

# from predict\_models.res\_dg.models.RCDG\_model import RCDGModel

from predict\_models.res\_dg.test import test

from predict\_models.res.test import test\_ArcNet

import argparse

# from models.res.test import test

os.environ["CUDA\_VISIBLE\_DEVICES"] = "0"

torch.set\_num\_threads(4)

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

torch.cuda.empty\_cache()

def predict(model):

if model == 'RCDG\_model':

test()

elif model == 'ArcNet':

test\_ArcNet()

## process.py

# preprocess of upload image

from PIL import Image

import cv2

import numpy as np

from scipy import ndimage

def preprocess(image\_path, save\_path):

image = Image.open(image\_path).convert('RGB')

mask = get\_mask(image)

cv2.imwrite(save\_path, mask \* 255)

def get\_mask(img):

gray = np.array(img.convert('L'))

return ndimage.binary\_opening(gray > 10, structure=np.ones((8, 8)))