Retrieval-based-Voice-Conversion-WebUI / **infer-web.py**

github-actions[bot] chore(format): run black on main (#1851) 6061f63 · last year

1619 lines (1539…

```python
import os
import sys
from dotenv import load_dotenv

now_dir = os.getcwd()
sys.path.append(now_dir)
load_dotenv()
from infer.modules.vc.modules import VC
from infer.modules.uvr5.modules import uvr
from infer.lib.train.process_ckpt import (
    change_info,
    extract_small_model,
    merge,
    show_info,
)
from i18n.i18n import I18nAuto
from configs.config import Config
from sklearn.cluster import MiniBatchKMeans
import torch, platform
import numpy as np
import gradio as gr
import faiss
import fairseq
import pathlib
import json
from time import sleep
from subprocess import Popen
from random import shuffle
import warnings
import traceback
import threading
import shutil
import logging


logging.getLogger("numba").setLevel(logging.WARNING)
```

```
    logging.getLogger("httpx").setLevel(logging.WARNING)

    logger = logging.getLogger(__name__)

    tmp = os.path.join(now_dir, "TEMP")
    shutil.rmtree(tmp, ignore_errors=True)
    shutil.rmtree("%s/runtime/Lib/site-packages/infer_pack" % (now_dir), ignore_errors=True)
    shutil.rmtree("%s/runtime/Lib/site-packages/uvr5_pack" % (now_dir), ignore_errors=True)
    os.makedirs(tmp, exist_ok=True)
    os.makedirs(os.path.join(now_dir, "logs"), exist_ok=True)
    os.makedirs(os.path.join(now_dir, "assets/weights"), exist_ok=True)
    os.environ["TEMP"] = tmp
    warnings.filterwarnings("ignore")
    torch.manual_seed(114514)


    config = Config()
    vc = VC(config)


    if config.dml == True:

        def forward_dml(ctx, x, scale):
            ctx.scale = scale
            res = x.clone().detach()
            return res

        fairseq.modules.grad_multiply.GradMultiply.forward = forward_dml
    i18n = I18nAuto()
    logger.info(i18n)
    # 判断是否有能用来训练和加速推理的N卡
    ngpu = torch.cuda.device_count()
    gpu_infos = []
    mem = []
    if_gpu_ok = False

    if torch.cuda.is_available() or ngpu != 0:
        for i in range(ngpu):
            gpu_name = torch.cuda.get_device_name(i)
```

```
                            choices=["pm", "harvest", "dio", "rmvpe", "rmvpe_gpu"],
                            value="rmvpe_gpu",
                            interactive=True,
                        )
                        gpus_rmvpe = gr.Textbox(
                            label=i18n(
                                "rmvpe卡号配置：以-分隔输入使用的不同进程卡号,例如0-0-1使用在卡0上跑2个进程
                            ),
                            value="%s-%s" % (gpus, gpus),
                            interactive=True,
                            visible=F0GPUVisible,
                        )
                    but2 = gr.Button(i18n("特征提取"), variant="primary")
                    info2 = gr.Textbox(label=i18n("输出信息"), value="", max_lines=8)
                    f0method8.change(
                        fn=change_f0_method,
                        inputs=[f0method8],
                        outputs=[gpus_rmvpe],
                    )
                    but2.click(
                        extract_f0_feature,
                        [
                            gpus6,
                            np7,
                            f0method8,
                            if_f0_3,
                            exp_dir1,
                            version19,
                            gpus_rmvpe,
                        ],
                        [info2],
                        api_name="train_extract_f0_feature",
                    )
                with gr.Group():
                    gr.Markdown(value=i18n("step3: 填写训练设置, 开始训练模型和索引"))
                    with gr.Row():
                        save_epoch10 = gr.Slider(
                            minimum=1,
                            maximum=50,
                            step=1,
                            label=i18n("保存频率save_every_epoch"),
                            value=5,
                            interactive=True,
                        )
                        total_epoch11 = gr.Slider(
                            minimum=2,
                            maximum=1000,
                            step=1,
```

```
                    step=1,
                    label=i18n("总训练轮数total_epoch"),
                    value=20,
                    interactive=True,
                )
                batch_size12 = gr.Slider(
                    minimum=1,
                    maximum=40,
                    step=1,
                    label=i18n("每张显卡的batch_size"),
                    value=default_batch_size,
                    interactive=True,
                )
                if_save_latest13 = gr.Radio(
                    label=i18n("是否仅保存最新的ckpt文件以节省硬盘空间"),
                    choices=[i18n("是"), i18n("否")],
                    value=i18n("否"),
                    interactive=True,
                )
                if_cache_gpu17 = gr.Radio(
                    label=i18n(
                        "是否缓存所有训练集至显存. 10min以下小数据可缓存以加速训练, 大数据缓存会炸显存也
                    ),
                    choices=[i18n("是"), i18n("否")],
                    value=i18n("否"),
                    interactive=True,
                )
                if_save_every_weights18 = gr.Radio(
                    label=i18n(
                        "是否在每次保存时间点将最终小模型保存至weights文件夹"
                    ),
                    choices=[i18n("是"), i18n("否")],
                    value=i18n("否"),
                    interactive=True,
                )
            with gr.Row():
                pretrained_G14 = gr.Textbox(
                    label=i18n("加载预训练底模G路径"),
                    value="assets/pretrained_v2/f0G40k.pth",
                    interactive=True,
                )
                pretrained_D15 = gr.Textbox(
                    label=i18n("加载预训练底模D路径"),
                    value="assets/pretrained_v2/f0D40k.pth",
                    interactive=True,
                )
                sr2.change(
                    change_sr2,
                    [sr2, if_f0_3, version19],
                    [pretrained_G14, pretrained_D15],
                )
                version19.change(
                    change_version19,
```

```
1357                    [sr2, if_f0_3, version19],
1362                    [if_f0_3, sr2, version19],
1363                    [f0method8, gpus_rmvpe, pretrained_G14, pretrained_D15],
1364                )
1365                gpus16 = gr.Textbox(
1366                    label=i18n(
1367                        "以-分隔输入使用的卡号, 例如   0-1-2   使用卡0和卡1和卡2"
1368                    ),
1369                    value=gpus,
1370                    interactive=True,
1371                )
1372                but3 = gr.Button(i18n("训练模型"), variant="primary")
1373                but4 = gr.Button(i18n("训练特征索引"), variant="primary")
1374                but5 = gr.Button(i18n("一键训练"), variant="primary")
1375                info3 = gr.Textbox(label=i18n("输出信息"), value="", max_lines=10)
1376                but3.click(
1377                    click_train,
1378                    [
1379                        exp_dir1,
1380                        sr2,
1381                        if_f0_3,
1382                        spk_id5,
1383                        save_epoch10,
1384                        total_epoch11,
1385                        batch_size12,
1386                        if_save_latest13,
1387                        pretrained_G14,
1388                        pretrained_D15,
1389                        gpus16,
1390                        if_cache_gpu17,
1391                        if_save_every_weights18,
1392                        version19,
1393                    ],
1394                    info3,
1395                    api_name="train_start",
1396                )
1397                but4.click(train_index, [exp_dir1, version19], info3)
1398                but5.click(
1399                    train1key,
1400                    [
1401                        exp_dir1,
1402                        sr2,
1403                        if_f0_3,
1404                        trainset_dir4,
1405                        spk_id5,
1406                        np7,
1407                        f0method8,
1408                        save_epoch10,
1409                        total_epoch11,
```

```
                            batch_size12,
                            if_save_latest13,
                            pretrained_G14,
                            pretrained_D15,
                            gpus16,
                            if_cache_gpu17,
                            if_save_every_weights18,
                            version19,
                            gpus_rmvpe,
                        ],
                        info3,
                        api_name="train_start_all",
                    )

        with gr.TabItem(i18n("ckpt处理")):
            with gr.Group():
                gr.Markdown(value=i18n("模型融合, 可用于测试音色融合"))
                with gr.Row():
                    ckpt_a = gr.Textbox(
                        label=i18n("A模型路径"), value="", interactive=True
                    )
                    ckpt_b = gr.Textbox(
                        label=i18n("B模型路径"), value="", interactive=True
                    )
                    alpha_a = gr.Slider(
                        minimum=0,
                        maximum=1,
                        label=i18n("A模型权重"),
                        value=0.5,
                        interactive=True,
                    )
                with gr.Row():
                    sr_ = gr.Radio(
                        label=i18n("目标采样率"),
                        choices=["40k", "48k"],
                        value="40k",
                        interactive=True,
                    )
                    if_f0_ = gr.Radio(
                        label=i18n("模型是否带音高指导"),
                        choices=[i18n("是"), i18n("否")],
                        value=i18n("是"),
                        interactive=True,
                    )
                    info__ = gr.Textbox(
                        label=i18n("要置入的模型信息"),
                        value="",
                        max_lines=8,
                        interactive=True,
                    )
                    name_to_save0 = gr.Textbox(
                        label=i18n("保存的模型名不带后缀"),
                        value="",
```

```
                        max_lines=1,
                        interactive=True,
                    )
                    version_2 = gr.Radio(
                        label=i18n("模型版本型号"),
                        choices=["v1", "v2"],
                        value="v1",
                        interactive=True,
                    )
                with gr.Row():
                    but6 = gr.Button(i18n("融合"), variant="primary")
                    info4 = gr.Textbox(label=i18n("输出信息"), value="", max_lines=8)
                but6.click(
                    merge,
                    [
                        ckpt_a,
                        ckpt_b,
                        alpha_a,
                        sr_,
                        if_f0_,
                        info__,
                        name_to_save0,
                        version_2,
                    ],
                    info4,
                    api_name="ckpt_merge",
                )  # def merge(path1,path2,alpha1,sr,f0,info):
            with gr.Group():
                gr.Markdown(
                    value=i18n("修改模型信息(仅支持weights文件夹下提取的小模型文件)")
                )
                with gr.Row():
                    ckpt_path0 = gr.Textbox(
                        label=i18n("模型路径"), value="", interactive=True
                    )
                    info_ = gr.Textbox(
                        label=i18n("要改的模型信息"),
                        value="",
                        max_lines=8,
                        interactive=True,
                    )
                    name_to_save1 = gr.Textbox(
                        label=i18n("保存的文件名, 默认空为和源文件同名"),
                        value="",
                        max_lines=8,
                        interactive=True,
                    )
```

```
                    label=i18n("目标采样率"),
                    choices=["32k", "40k", "48k"],
                    value="40k",
                    interactive=True,
                )
                if_f0__ = gr.Radio(
                    label=i18n("模型是否带音高指导,1是0否"),
                    choices=["1", "0"],
                    value="1",
                    interactive=True,
                )
                version_1 = gr.Radio(
                    label=i18n("模型版本型号"),
                    choices=["v1", "v2"],
                    value="v2",
                    interactive=True,
                )
                info___ = gr.Textbox(
                    label=i18n("要置入的模型信息"),
                    value="",
                    max_lines=8,
                    interactive=True,
                )
```

```
                    )
                    but9 = gr.Button(i18n("提取"), variant="primary")
                    info7 = gr.Textbox(label=i18n("输出信息"), value="", max_lines=8)
                    ckpt_path2.change(
                        change_info_, [ckpt_path2], [sr__, if_f0__, version_1]
                    )
                but9.click(
                    extract_small_model,
                    [ckpt_path2, save_name, sr__, if_f0__, info___, version_1],
                    info7,
                    api_name="ckpt_extract",
                )

            with gr.TabItem(i18n("Onnx导出")):
                with gr.Row():
                    ckpt_dir = gr.Textbox(
                        label=i18n("RVC模型路径"), value="", interactive=True
                    )
                with gr.Row():
                    onnx_dir = gr.Textbox(
                        label=i18n("Onnx输出路径"), value="", interactive=True
                    )
                with gr.Row():
                    infoOnnx = gr.Label(label="info")
                with gr.Row():
                    butOnnx = gr.Button(i18n("导出Onnx模型"), variant="primary")
                butOnnx.click(
                    export_onnx, [ckpt_dir, onnx_dir], infoOnnx, api_name="export_onnx"
                )

            tab_faq = i18n("常见问题解答")
            with gr.TabItem(tab_faq):
                try:
                    if tab_faq == "常见问题解答":
                        with open("docs/cn/faq.md", "r", encoding="utf8") as f:
                            info = f.read()
                    else:
                        with open("docs/en/faq_en.md", "r", encoding="utf8") as f:
                            info = f.read()
                    gr.Markdown(value=info)
                except:
                    gr.Markdown(traceback.format_exc())

        if config.iscolab:
            app.queue(concurrency_count=511, max_size=1022).launch(share=True)
        else:
            app.queue(concurrency_count=511, max_size=1022).launch(
                server_name="0.0.0.0",
                inbrowser=not config.noautoopen,
                server_port=config.listen_port,
                quiet=True,
            )
```