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

12 people chore(sync): merge dev into main (#1379) e9dd11b · 2 years ago

1529 lines (1450…

```python
import os
import sys

now_dir = os.getcwd()
sys.path.append(now_dir)
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
from dotenv import load_dotenv
import torch

try:
    import intel_extension_for_pytorch as ipex  # pylint: disable=import-error, unused-import

    if torch.xpu.is_available():
        from infer.modules.ipex import ipex_init

        ipex_init()
except Exception:  # pylint: disable=broad-exception-caught
    pass
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)

    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)


    load_dotenv()
    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)
```

```
                    )
                    .fit(big_npy)
                    .cluster_centers_
                )
            except:
                info = traceback.format_exc()
                logger.info(info)
                infos.append(info)
                yield "\n".join(infos)

        np.save("%s/total_fea.npy" % exp_dir, big_npy)
        n_ivf = min(int(16 * np.sqrt(big_npy.shape[0])), big_npy.shape[0] // 39)
        infos.append("%s,%s" % (big_npy.shape, n_ivf))
        yield "\n".join(infos)
        index = faiss.index_factory(256 if version19 == "v1" else 768, "IVF%s,Flat" % n_ivf)
        # index = faiss.index_factory(256if version19=="v1"else 768, "IVF%s,PQ128x4fs,RFlat"%n_ivf)
        infos.append("training")
        yield "\n".join(infos)
        index_ivf = faiss.extract_index_ivf(index)  #
        index_ivf.nprobe = 1
        index.train(big_npy)
        faiss.write_index(
            index,
            "%s/trained_IVF%s_Flat_nprobe_%s_%s_%s.index"
            % (exp_dir, n_ivf, index_ivf.nprobe, exp_dir1, version19),
        )

        infos.append("adding")
        yield "\n".join(infos)
```

```
671         batch_size_add = 8192
672         for i in range(0, big_npy.shape[0], batch_size_add):
673             index.add(big_npy[i : i + batch_size_add])
674         faiss.write_index(
675             index,
676             "%s/added_IVF%s_Flat_nprobe_%s_%s_%s.index"
677             % (exp_dir, n_ivf, index_ivf.nprobe, exp_dir1, version19),
678         )
679         infos.append(
680             "成功构建索引，added_IVF%s_Flat_nprobe_%s_%s_%s.index"
681             % (n_ivf, index_ivf.nprobe, exp_dir1, version19)
682         )
683         # faiss.write_index(index, '%s/added_IVF%s_Flat_FastScan_%s.index'%(exp_dir,n_ivf,version19)
684         # infos.append("成功构建索引，added_IVF%s_Flat_FastScan_%s.index"%(n_ivf,version19))
685         yield "\n".join(infos)
686
687
688     # but5.click(train1key, [exp_dir1, sr2, if_f0_3, trainset_dir4, spk_id5, gpus6, np7, f0method8,
689     def train1key(
690         exp_dir1,
691         sr2,
692         if_f0_3,
693         trainset_dir4,
694         spk_id5,
695         np7,
696         f0method8,
697         save_epoch10,
698         total_epoch11,
699         batch_size12,
700         if_save_latest13,
701         pretrained_G14,
702         pretrained_D15,
703         gpus16,
704         if_cache_gpu17,
705         if_save_every_weights18,
706         version19,
707         gpus_rmvpe,
708     ):
709         infos = []
710
711         def get_info_str(strr):
712             infos.append(strr)
713             return "\n".join(infos)
714
715         # step1:处理数据
716         yield get_info_str(i18n("step1:正在处理数据"))
717         [get_info_str(_) for _ in preprocess_dataset(trainset_dir4, exp_dir1, sr2, np7)]
718
719         # step2a:提取音高
720         yield get_info_str(i18n("step2:正在提取音高&正在提取特征"))
721         [
722             get_info_str(_)
723             for _ in extract_f0_feature(
```

```
            gpus16, np7, f0method8, if_f0_3, exp_dir1, version19, gpus_rmvpe
        )
    ]

    # step3a:训练模型
    yield get_info_str(i18n("step3a:正在训练模型"))
    click_train(
        exp_dir1,
        sr2,
        if_f0_3,
        spk_id5,
        save_epoch10,
        total_epoch11,
        batch_size12,
        if_save_latest13,
        pretrained_G14,
        pretrained_D15,
        gpus16,
        if_cache_gpu17,
```



```
    # step3b:训练索引
    [get_info_str(_) for _ in train_index(exp_dir1, version19)]
    yield get_info_str(i18n("全流程结束！"))


#                    ckpt_path2.change(change_info_,[ckpt_path2],[sr__,if_f0__])
def change_info_(ckpt_path):
    if not os.path.exists(ckpt_path.replace(os.path.basename(ckpt_path), "train.log")):
        return {"__type__": "update"}, {"__type__": "update"}, {"__type__": "update"}
    try:
        with open(
            ckpt_path.replace(os.path.basename(ckpt_path), "train.log"), "r"
        ) as f:
            info = eval(f.read().strip("\n").split("\n")[0].split("\t")[-1])
            sr, f0 = info["sample_rate"], info["if_f0"]
            version = "v2" if ("version" in info and info["version"] == "v2") else "v1"
            return sr, str(f0), version
    except:
        traceback.print_exc()
        return {"__type__": "update"}, {"__type__": "update"}, {"__type__": "update"}


F0GPUVisible = config.dml == False


def change_f0_method(f0method8):
    if f0method8 == "rmvpe_gpu":
        visible = F0GPUVisible
    else:
```

```
            else:
                visible = False
            return {"visible": visible, "__type__": "update"}


    with gr.Blocks(title="RVC WebUI") as app:
        gr.Markdown("## RVC WebUI")
        gr.Markdown(
            value=i18n(
                "本软件以MIT协议开源, 作者不对软件具备任何控制力, 使用软件者、传播软件导出的声音者自负全责. <br>如
            )
        )
        with gr.Tabs():
            with gr.TabItem(i18n("模型推理")):
                with gr.Row():
                    sid0 = gr.Dropdown(label=i18n("推理音色"), choices=sorted(names))
                    with gr.Column():
                        refresh_button = gr.Button(i18n("刷新音色列表和索引路径"), variant="primary")
                        clean_button = gr.Button(i18n("卸载音色省显存"), variant="primary")
                    spk_item = gr.Slider(
                        minimum=0,
                        maximum=2333,
                        step=1,
                        label=i18n("请选择说话人id"),
                        value=0,
                        visible=False,
                        interactive=True,
                    )
                    clean_button.click(
                        fn=clean, inputs=[], outputs=[sid0], api_name="infer_clean"
                    )
                with gr.TabItem(i18n("单次推理")):
                    with gr.Group():
                        with gr.Row():
                            with gr.Column():
                                vc_transform0 = gr.Number(
                                    label=i18n("变调(整数, 半音数量, 升八度12降八度-12)"), value=0
                                )
                                input_audio0 = gr.Textbox(
                                    label=i18n("输入待处理音频文件路径(默认是正确格式示例)"),
                                    placeholder="C:\\Users\\Desktop\\audio_example.wav",
                                )
                                file_index1 = gr.Textbox(
                                    label=i18n("特征检索库文件路径,为空则使用下拉的选择结果"),
                                    placeholder="C:\\Users\\Desktop\\model_example.index",
                                    interactive=True,
                                )
                                file_index2 = gr.Dropdown(
                                    label=i18n("自动检测index路径,下拉式选择(dropdown)"),
                                    choices=sorted(index_paths),
                                    interactive=True,
                                )
                                f0method0 = gr.Radio(
```

```
                                label=i18n(
                                    "选择音高提取算法,输入歌声可用pm提速,harvest低音好但巨慢无比,crepe效
                                ),
                                choices=["pm", "harvest", "crepe", "rmvpe"]
                                if config.dml == False
                                else ["pm", "harvest", "rmvpe"],
                                value="rmvpe",
                                interactive=True,
                            )

                        with gr.Column():
                            resample_sr0 = gr.Slider(
                                minimum=0,
                                maximum=48000,
                                label=i18n("后处理重采样至最终采样率，0为不进行重采样"),
                                value=0,
                                step=1,
                                interactive=True,
                            )
                            rms_mix_rate0 = gr.Slider(
                                minimum=0,
                                maximum=1,
                                label=i18n("输入源音量包络替换输出音量包络融合比例，越靠近1越使用输出包络"
                                value=0.25,
                                interactive=True,
                            )
                            protect0 = gr.Slider(
                                minimum=0,
                                maximum=0.5,
                                label=i18n(
                                    "保护清辅音和呼吸声，防止电音撕裂等artifact，拉满0.5不开启，调低加大保
                                ),
                                value=0.33,
                                step=0.01,
                                interactive=True,
                            )
                            filter_radius0 = gr.Slider(
                                minimum=0,
                                maximum=7,
                                label=i18n(
                                    ">=3则使用对harvest音高识别的结果使用中值滤波，数值为滤波半径，使用可
                                ),
                                value=3,
                                step=1,
                                interactive=True,
                            )
                            index_rate1 = gr.Slider(
                                minimum=0,
                                maximum=1,
                                label=i18n("检索特征占比"),
                                value=0.75,
                                interactive=True,
                            )
```

```
                        f0_file = gr.File(
                            label=i18n("F0曲线文件, 可选, 一行一个音高, 代替默认F0及升降调"),
                            visible=False,
                        )

                        refresh_button.click(
                            fn=change_choices,
                            inputs=[],
                            outputs=[sid0, file_index2],
                            api_name="infer_refresh",
                        )
                        # file_big_npy1 = gr.Textbox(
                        #     label=i18n("特征文件路径"),
```

```
1456                    )
1457                    save_name = gr.Textbox(
1458                        label=i18n("保存名"), value="", interactive=True
1459                    )
1460                    sr__ = gr.Radio(
1461                        label=i18n("目标采样率"),
1462                        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:
```

```
1516                            info = f.read()
1517                    gr.Markdown(value=info)
1518                except:
1519                    gr.Markdown(traceback.format_exc())
1520
1521        if config.iscolab:
1522            app.queue(concurrency_count=511, max_size=1022).launch(share=True)
1523        else:
1524            app.queue(concurrency_count=511, max_size=1022).launch(
1525                server_name="0.0.0.0",
1526                inbrowser=not config.noautoopen,
1527                server_port=config.listen_port,
1528                quiet=True,
1529            )
```