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)
```

```
        global index_paths
        for root, dirs, files in os.walk(index_root, topdown=False):
            for name in files:
                if name.endswith(".index") and "trained" not in name:
                    index_paths.append("%s/%s" % (root, name))


    lookup_indices(index_root)
    lookup_indices(outside_index_root)
    uvr5_names = []
    for name in os.listdir(weight_uvr5_root):
        if name.endswith(".pth") or "onnx" in name:
            uvr5_names.append(name.replace(".pth", ""))


    def change_choices():
        names = []
        for name in os.listdir(weight_root):
            if name.endswith(".pth"):
                names.append(name)
        index_paths = []
        for root, dirs, files in os.walk(index_root, topdown=False):
            for name in files:
                if name.endswith(".index") and "trained" not in name:
                    index_paths.append("%s/%s" % (root, name))
        return {"choices": sorted(names), "__type__": "update"}, {
            "choices": sorted(index_paths),
            "__type__": "update",
        }


    def clean():
        return {"value": "", "__type__": "update"}


    def export_onnx(ModelPath, ExportedPath):
        from infer.modules.onnx.export import export_onnx as eo

        eo(ModelPath, ExportedPath)


    sr_dict = {
        "32k": 32000,
        "40k": 40000,
        "48k": 48000,
    }


    def if_done(done, p):
        while 1:
```

```
            if p.poll() is None:
                sleep(0.5)
            else:
                break
        done[0] = True


    def if_done_multi(done, ps):
        while 1:
            # poll==None代表进程未结束
            # 只要有一个进程未结束都不停
            flag = 1
            for p in ps:
                if p.poll() is None:
                    flag = 0
                    sleep(0.5)
                    break
            if flag == 1:
                break
        done[0] = True


    def preprocess_dataset(trainset_dir, exp_dir, sr, n_p):
        sr = sr_dict[sr]
        os.makedirs("%s/logs/%s" % (now_dir, exp_dir), exist_ok=True)
        f = open("%s/logs/%s/preprocess.log" % (now_dir, exp_dir), "w")
        f.close()
        cmd = '"%s" infer/modules/train/preprocess.py "%s" %s %s "%s/logs/%s" %s %.1f' % (
            config.python_cmd,
            trainset_dir,
            sr,
            n_p,
            now_dir,
            exp_dir,
            config.noparallel,
            config.preprocess_per,
        )
        logger.info("Execute: " + cmd)
        # , stdin=PIPE, stdout=PIPE,stderr=PIPE,cwd=now_dir
        p = Popen(cmd, shell=True)
        # 煞笔gr, popen read都非得全跑完了再一次性读取, 不用gr就正常读一句输出一句;只能额外弄出一个文本流定时读
        done = [False]
        threading.Thread(
            target=if_done,
            args=(
                done,
                p,
            ),
        ).start()
        while 1:
            with open("%s/logs/%s/preprocess.log" % (now_dir, exp_dir), "r") as f:
```

Code | Blame | Raw

```
        with open("%s/logs/%s/preprocess.log" % (now_dir, exp_dir), "r") as f:
            log = f.read()
        logger.info(log)
        yield log


# but2.click(extract_f0,[gpus6,np7,f0method8,if_f0_3,trainset_dir4],[info2])
def extract_f0_feature(gpus, n_p, f0method, if_f0, exp_dir, version19, gpus_rmvpe):
    gpus = gpus.split("-")
    os.makedirs("%s/logs/%s" % (now_dir, exp_dir), exist_ok=True)
    f = open("%s/logs/%s/extract_f0_feature.log" % (now_dir, exp_dir), "w")
    f.close()
    if if_f0:
        if f0method != "rmvpe_gpu":
            cmd = (
                '"%s" infer/modules/train/extract/extract_f0_print.py "%s/logs/%s" %s %s'
                % (
                    config.python_cmd,
                    now_dir,
                    exp_dir,
                    n_p,
                    f0method,
                )
            )
            logger.info("Execute: " + cmd)
            p = Popen(
                cmd, shell=True, cwd=now_dir
            )  # , stdin=PIPE, stdout=PIPE,stderr=PIPE
            # 煞笔gr, popen read都非得全跑完了再一次性读取, 不用gr就正常读一句输出一句;只能额外弄出一个文本流
            done = [False]
            threading.Thread(
                target=if_done,
                args=(
                    done,
                    p,
                ),
            ).start()
        else:
            if gpus_rmvpe != "-":
                gpus_rmvpe = gpus_rmvpe.split("-")
                leng = len(gpus_rmvpe)
                ps = []
                for idx, n_g in enumerate(gpus_rmvpe):
                    cmd = (
                        '"%s" infer/modules/train/extract/extract_f0_rmvpe.py %s %s %s "%s/logs/
                        % (
                            config.python_cmd,
                            leng,
                            idx,
                            n_g,
```

```
                            now_dir,
                            exp_dir,
                            config.is_half,
                        )
                    )
                    logger.info("Execute: " + cmd)
                    p = Popen(
                        cmd, shell=True, cwd=now_dir
                    )  # , shell=True, stdin=PIPE, stdout=PIPE, stderr=PIPE, cwd=now_dir
                    ps.append(p)
                # 煞笔gr, popen read都非得全跑完了再一次性读取, 不用gr就正常读一句输出一句;只能额外弄出一个文
                done = [False]
                threading.Thread(
                    target=if_done_multi,  #
                    args=(
                        done,
                        ps,
                    ),
                ).start()
            else:
                cmd = (
                    config.python_cmd
                    + ' infer/modules/train/extract/extract_f0_rmvpe_dml.py "%s/logs/%s" '
                    % (
                        now_dir,
                        exp_dir,
                    )
                )
                logger.info("Execute: " + cmd)
                p = Popen(
                    cmd, shell=True, cwd=now_dir
                )  # , shell=True, stdin=PIPE, stdout=PIPE, stderr=PIPE, cwd=now_dir
                p.wait()
                done = [True]
            while 1:
                with open(
                    "%s/logs/%s/extract_f0_feature.log" % (now_dir, exp_dir), "r"
                ) as f:
                    yield (f.read())
                sleep(1)
                if done[0]:
                    break
            with open("%s/logs/%s/extract_f0_feature.log" % (now_dir, exp_dir), "r") as f:
                log = f.read()
            logger.info(log)
            yield log
        # 对不同part分别开多进程
        """
        n_part=int(sys.argv[1])
        i_part=int(sys.argv[2])
        i_gpu=sys.argv[3]
        exp_dir=sys.argv[4]
        os.environ["CUDA_VISIBLE_DEVICES"]=str(i_gpu)
```

```
        """
        leng = len(gpus)
        ps = []
        for idx, n_g in enumerate(gpus):
            cmd = (
                '"%s" infer/modules/train/extract_feature_print.py %s %s %s %s "%s/logs/%s" %s %s'
                % (
                    config.python_cmd,
                    config.device,
                    leng,
                    idx,
                    n_g,
                    now_dir,
                    exp_dir,
                    version19,
                    config.is_half,
                )
            )
            logger.info("Execute: " + cmd)
            p = Popen(
                cmd, shell=True, cwd=now_dir
            )  # , shell=True, stdin=PIPE, stdout=PIPE, stderr=PIPE, cwd=now_dir
            ps.append(p)
        # 煞笔gr, popen read都非得全跑完了再一次性读取, 不用gr就正常读一句输出一句;只能额外弄出一个文本流定时读
        done = [False]
        threading.Thread(
            target=if_done_multi,
            args=(
                done,
                ps,
            ),
        ).start()
        while 1:
            with open("%s/logs/%s/extract_f0_feature.log" % (now_dir, exp_dir), "r") as f:
                yield (f.read())
            sleep(1)
            if done[0]:
                break
        with open("%s/logs/%s/extract_f0_feature.log" % (now_dir, exp_dir), "r") as f:
            log = f.read()
        logger.info(log)
        yield log


    def get_pretrained_models(path_str, f0_str, sr2):
```

```
                    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,
            )
```