

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

...

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

1619 lines (1539...

```
1  import os
2  import sys
3  from dotenv import load_dotenv
4
5  now_dir = os.getcwd()
6  sys.path.append(now_dir)
7  load_dotenv()
8  from infer.modules.vc.modules import VC
9  from infer.modules.uvr5.modules import uvr
10 from infer.lib.train.process_ckpt import (
11     change_info,
12     extract_small_model,
13     merge,
14     show_info,
15 )
16 from i18n.i18n import I18nAuto
17 from configs.config import Config
18 from sklearn.cluster import MiniBatchKMeans
19 import torch, platform
20 import numpy as np
21 import gradio as gr
22 import faiss
23 import fairseq
24 import pathlib
25 import json
26 from time import sleep
27 from subprocess import Popen
28 from random import shuffle
29 import warnings
30 import traceback
31 import threading
32 import shutil
33 import logging
34
35
36 logging.getLogger("numba").setLevel(logging.WARNING)
37 logging.getLogger("librosa").setLevel(logging.WARNING)
```

```
37 logging.getLogger("httpx").setLevel(logging.WARNING)
38
39 logger = logging.getLogger(__name__)
40
41 tmp = os.path.join(now_dir, "TEMP")
42 shutil.rmtree(tmp, ignore_errors=True)
43 shutil.rmtree("%s/runtime/Lib/site-packages/infer_pack" % (now_dir), ignore_errors=True)
44 shutil.rmtree("%s/runtime/Lib/site-packages/uvr5_pack" % (now_dir), ignore_errors=True)
45 os.makedirs(tmp, exist_ok=True)
46 os.makedirs(os.path.join(now_dir, "logs"), exist_ok=True)
47 os.makedirs(os.path.join(now_dir, "assets/weights"), exist_ok=True)
48 os.environ["TEMP"] = tmp
49 warnings.filterwarnings("ignore")
50 torch.manual_seed(114514)
51
52
53 config = Config()
54 vc = VC(config)
55
56
57 if config.dml == True:
58
59     def forward_dml(ctx, x, scale):
60         ctx.scale = scale
61         res = x.clone().detach()
62         return res
63
64     fairseq.modules.grad_multiply.GradMultiply.forward = forward_dml
65 i18n = I18nAuto()
66 logger.info(i18n)
67 # 判断是否有能用来训练和加速推理的N卡
68 ngpu = torch.cuda.device_count()
69 gpu_infos = []
70 mem = []
71 if_gpu_ok = False
72
73 if torch.cuda.is_available() or ngpu != 0:
74     for i in range(ngpu):
75         gpu_name = torch.cuda.get_device_name(i)
```



```

146     global index_paths
147     for root, dirs, files in os.walk(index_root, topdown=False):
148         for name in files:
149             if name.endswith(".index") and "trained" not in name:
150                 index_paths.append("%s/%s" % (root, name))
151
152
153     lookup_indices(index_root)
154     lookup_indices(outside_index_root)
155     uvr5_names = []
156     for name in os.listdir(weight_uvr5_root):
157         if name.endswith(".pth") or "onnx" in name:
158             uvr5_names.append(name.replace(".pth", ""))
159
160
161     ✓ def change_choices():
162         names = []
163         for name in os.listdir(weight_root):
164             if name.endswith(".pth"):
165                 names.append(name)
166         index_paths = []
167         for root, dirs, files in os.walk(index_root, topdown=False):
168             for name in files:
169                 if name.endswith(".index") and "trained" not in name:
170                     index_paths.append("%s/%s" % (root, name))
171         return {"choices": sorted(names), "__type__": "update"}, {
172             "choices": sorted(index_paths),
173             "__type__": "update",
174         }
175
176
177     def clean():
178         return {"value": "", "__type__": "update"}
179
180
181     def export_onnx(ModelPath, ExportedPath):
182         from infer.modules.onnx.export import export_onnx as eo
183
184         eo(ModelPath, ExportedPath)
185
186
187     ✓ sr_dict = {
188         "32k": 32000,
189         "40k": 40000,
190         "48k": 48000,
191     }
192
193
194     ✓ def if_done(done, p):
195         while 1:

```

```

196         if p.poll() is None:
197             sleep(0.5)
198         else:
199             break
200     done[0] = True
201
202
203 ✓ def if_done_multi(done, ps):
204     while 1:
205         # poll==None代表进程未结束
206         # 只要有一个进程未结束都不停
207         flag = 1
208         for p in ps:
209             if p.poll() is None:
210                 flag = 0
211                 sleep(0.5)
212                 break
213         if flag == 1:
214             break
215     done[0] = True
216
217
218 ✓ def preprocess_dataset(trainset_dir, exp_dir, sr, n_p):
219     sr = sr_dict[sr]
220     os.makedirs("%s/logs/%s" % (now_dir, exp_dir), exist_ok=True)
221     f = open("%s/logs/%s/preprocess.log" % (now_dir, exp_dir), "w")
222     f.close()
223     cmd = '"%s" infer/modules/train/preprocess.py "%s" %s %s "%s/logs/%s" %s %.1f' % (
224         config.python_cmd,
225         trainset_dir,
226         sr,
227         n_p,
228         now_dir,
229         exp_dir,
230         config.noparallel,
231         config.preprocess_per,
232     )
233     logger.info("Execute: " + cmd)
234     # , stdin=PIPE, stdout=PIPE, stderr=PIPE, cwd=now_dir
235     p = Popen(cmd, shell=True)
236     # 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
237     done = [False]
238     threading.Thread(
239         target=if_done,
240         args=(
241             done,
242             p,
243         ),
244     ).start()
245     while 1:
246         with open("%s/logs/%s/preprocess.log" % (now_dir, exp_dir), "r") as f:

```

Code

Blame

Raw



```

251         with open("%s/logs/%s/preprocess.log" % (now_dir, exp_dir), "r") as f:
252             log = f.read()
253             logger.info(log)
254             yield log
255
256
257     # but2.click(extract_f0, [gpus6, np7, f0method8, if_f0_3, trainset_dir4], [info2])
258     def extract_f0_feature(gpus, n_p, f0method, if_f0, exp_dir, version19, gpus_rmvpe):
259         gpus = gpus.split("-")
260         os.makedirs("%s/logs/%s" % (now_dir, exp_dir), exist_ok=True)
261         f = open("%s/logs/%s/extract_f0_feature.log" % (now_dir, exp_dir), "w")
262         f.close()
263         if if_f0:
264             if f0method != "rmvpe_gpu":
265                 cmd = (
266                     '"%s" infer/modules/train/extract/extract_f0_print.py "%s/logs/%s" %s %s'
267                     % (
268                         config.python_cmd,
269                         now_dir,
270                         exp_dir,
271                         n_p,
272                         f0method,
273                     )
274                 )
275                 logger.info("Execute: " + cmd)
276                 p = Popen(
277                     cmd, shell=True, cwd=now_dir
278                 ) # , stdin=PIPE, stdout=PIPE, stderr=PIPE
279                 # 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
280                 done = [False]
281                 threading.Thread(
282                     target=if_done,
283                     args=(
284                         done,
285                         p,
286                     ),
287                 ).start()
288             else:
289                 if gpus_rmvpe != "-":
290                     gpus_rmvpe = gpus_rmvpe.split("-")
291                     leng = len(gpus_rmvpe)
292                     ps = []
293                     for idx, n_g in enumerate(gpus_rmvpe):
294                         cmd = (
295                             '"%s" infer/modules/train/extract/extract_f0_rmvpe.py %s %s %s "%s/logs/'
296                             % (
297                                 config.python_cmd,
298                                 leng,
299                                 idx,
300                                 n_g,

```

```

301         now_dir,
302         exp_dir,
303         config.is_half,
304     )
305 )
306 logger.info("Execute: " + cmd)
307 p = Popen(
308     cmd, shell=True, cwd=now_dir
309 ) # , shell=True, stdin=PIPE, stdout=PIPE, stderr=PIPE, cwd=now_dir
310 ps.append(p)
311 # 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
312 done = [False]
313 threading.Thread(
314     target=if_done_multi, #
315     args=(
316         done,
317         ps,
318     ),
319 ).start()
320 else:
321     cmd = (
322         config.python_cmd
323         + ' infer/modules/train/extract/extract_f0_rmvp_dml.py "%s/logs/%s" '
324         % (
325             now_dir,
326             exp_dir,
327         )
328     )
329 logger.info("Execute: " + cmd)
330 p = Popen(
331     cmd, shell=True, cwd=now_dir
332 ) # , shell=True, stdin=PIPE, stdout=PIPE, stderr=PIPE, cwd=now_dir
333 p.wait()
334 done = [True]
335 while 1:
336     with open(
337         "%s/logs/%s/extract_f0_feature.log" % (now_dir, exp_dir), "r"
338     ) as f:
339         yield (f.read())
340     sleep(1)
341     if done[0]:
342         break
343     with open("%s/logs/%s/extract_f0_feature.log" % (now_dir, exp_dir), "r") as f:
344         log = f.read()
345         logger.info(log)
346         yield log
347 # 对不同part分别开多进程
348 """
349 n_part=int(sys.argv[1])
350 i_part=int(sys.argv[2])
351 i_gpu=sys.argv[3]
352 exp_dir=sys.argv[4]
353 os.environ["CUDA_VISIBLE_DEVICES"]=str(i_gpu)

```

```

354     """
355     leng = len(gpus)
356     ps = []
357     for idx, n_g in enumerate(gpus):
358         cmd = (
359             "%s" infer/modules/train/extract_feature_print.py %s %s %s %s "%s/logs/%s" %s %s'
360             % (
361                 config.python_cmd,
362                 config.device,
363                 leng,
364                 idx,
365                 n_g,
366                 now_dir,
367                 exp_dir,
368                 version19,
369                 config.is_half,
370             )
371         )
372         logger.info("Execute: " + cmd)
373         p = Popen(
374             cmd, shell=True, cwd=now_dir
375         ) # , shell=True, stdin=PIPE, stdout=PIPE, stderr=PIPE, cwd=now_dir
376         ps.append(p)
377     # 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
378     done = [False]
379     threading.Thread(
380         target=if_done_multi,
381         args=(
382             done,
383             ps,
384         ),
385     ).start()
386     while 1:
387         with open("%s/logs/%s/extract_f0_feature.log" % (now_dir, exp_dir), "r") as f:
388             yield (f.read())
389             sleep(1)
390             if done[0]:
391                 break
392     with open("%s/logs/%s/extract_f0_feature.log" % (now_dir, exp_dir), "r") as f:
393         log = f.read()
394         logger.info(log)
395         yield log
396
397
398  ✓ def get_pretrained_models(path_str, f0_str, sr2):

```



```
1546         label=i18n("目标采样率"),
1547         choices=["32k", "40k", "48k"],
1548         value="40k",
1549         interactive=True,
1550     )
1551     if_f0__ = gr.Radio(
1552         label=i18n("模型是否带音高指导,1是0否"),
1553         choices=["1", "0"],
1554         value="1",
1555         interactive=True,
1556     )
1557     version_1 = gr.Radio(
1558         label=i18n("模型版本型号"),
1559         choices=["v1", "v2"],
1560         value="v2",
1561         interactive=True,
1562     )
1563     info___ = gr.Textbox(
1564         label=i18n("要置入的模型信息"),
1565         value="",
1566         max_lines=8,
1567         interactive=True,
1568     )
```

```

1568         but9 = gr.Button(i18n("提取"), variant="primary")
1569         info7 = gr.Textbox(label=i18n("输出信息"), value="", max_lines=8)
1570         ckpt_path2.change(
1571             change_info_, [ckpt_path2], [sr__, if_f0__, version_1]
1572         )
1573     but9.click(
1574         extract_small_model,
1575         [ckpt_path2, save_name, sr__, if_f0__, info____, version_1],
1576         info7,
1577         api_name="ckpt_extract",
1578     )
1579
1580 with gr.TabItem(i18n("Onnx导出")):
1581     with gr.Row():
1582         ckpt_dir = gr.Textbox(
1583             label=i18n("RVC模型路径"), value="", interactive=True
1584         )
1585     with gr.Row():
1586         onnx_dir = gr.Textbox(
1587             label=i18n("Onnx输出路径"), value="", interactive=True
1588         )
1589     with gr.Row():
1590         infoOnnx = gr.Label(label="info")
1591     with gr.Row():
1592         butOnnx = gr.Button(i18n("导出Onnx模型"), variant="primary")
1593         butOnnx.click(
1594             export_onnx, [ckpt_dir, onnx_dir], infoOnnx, api_name="export_onnx"
1595         )
1596
1597 tab_faq = i18n("常见问题解答")
1598 with gr.TabItem(tab_faq):
1599     try:
1600         if tab_faq == "常见问题解答":
1601             with open("docs/cn/faq.md", "r", encoding="utf8") as f:
1602                 info = f.read()
1603         else:
1604             with open("docs/en/faq_en.md", "r", encoding="utf8") as f:
1605                 info = f.read()
1606         gr.Markdown(value=info)
1607     except:
1608         gr.Markdown(traceback.format_exc())
1609
1610 if config.iscolab:
1611     app.queue(concurrency_count=511, max_size=1022).launch(share=True)
1612 else:
1613     app.queue(concurrency_count=511, max_size=1022).launch(
1614         server_name="0.0.0.0",
1615         inbrowser=not config.noautoopen,
1616         server_port=config.listen_port,
1617         quiet=True,
1618     )
1619

```

