sola project demo

今回の変更点

- 外部環境の提供開始
- `interactive.py` について、オプションの追加
- `interactive.py` のリスト機能について、出力を整形
- `main.py` にリクエストなし生成待機を追加

エンドポイントが変更となりました。

外部環境について

外部環境の提供を開始しました。内部のRTX A6000より2倍程度高速です。
内部環境とは`run_id`が独立に付与されます。ご注意ください。

環境構築

```
git clone https://github.com/YutoGCN/deploy-demo

cd deploy-demo

(optional)python3 -m venv venv

(optional)./venv/bin/activate

pip3 install -r requirements.txt
```

設定すべき項目

- `endpoints.py`: エンドポイントの指定
- `params_comfy.json`: 生成フローの指定、パラメータの設定

パラメータファイルについて、`params_comfy.json`, `params_comfy_by_character.json`, `params_comfy_PartialCN-Chara.json` がサンプルです。

`interactive.py` オプションの追加

```
python interactive.py param_file.json [--mode MODE] [--run_id RUN_ID]
```

```
オプション
    --mode MODE       動作モードを指定します。モードは以下の通りです:
                          1/request, 2/list, 3/status, 4/download, 5/terminate
    --run_id RUN_ID   必要な操作（status, download, terminate）のrun_idを指定します。
    --help            このヘルプメッセージを表示して終了します。

--mode, --run_idが必要なモードで指定されていない場合、スクリプトはユーザーに対話的に入力を求めます。
```

`interactive.py` 出力整形について

listを行った際整形を行います
また、全ユーザのjobを出力するよう変更しました。

```
run_id | user_name   | job_name                          | status         | run_time(s)
------ | ----------- | --------------------------------- | -------------- | -----------
1      | lovemachine | test20241009                      | FINISHED       | 457
2      | lovemachine | test20241009_by_chara             | FINISHED       | 572
3      | lovemachine | test20241009_partialcn_chara      | FINISHED       | 1371
4      | lovemachine | test20241014_partialcn_chara      | FAILED         | 0
5      | lovemachine | test20241015_partialcn_chara      | INSTANCE_INIT  | 0
```

`main.py` リクエストなし生成待機

ステータスチェックとダウンロードのみを行うオプションを追加しました。

```
main.py parameter_file [-h] [--no_request] [--run_id RUN_ID]

  parameter_file   パラメータJSONファイル

オプション:
  -h, --help         ヘルプメッセージを表示
  --no_request       新しいジョブをリクエストせず、ステータスチェックと
                     ダウンロードのみを実行します
  --run_id RUN_ID    既存のジョブのrun_idを指定します
                     指定がない場合対話的に入力を求められます
```

`main.py` リクエストなし生成待機に伴う変更

`Ctrl+C` による中断ができなくなっております。中断したい場合は `interactive.py` を使ってください。

以下各種設定方法

変更はございません

エンドポイントの設定

```
BASE_URL = ""
```

`endopoint.py` の `""` の間にお渡ししたエンドポイントのurlを入れてください。

入力・出力等の設定

```
{
    "material_folder": "test",
    "user_name": "lovemachine",
    "job_name": "dafult_20240929",
    "output_path": "output/test",
```

出力が`.png`になったことにより、出力の指定がディレクトリになりました。

ワークフローの指定

```
    "workflow_name": "dafault",
```

`dafult`, `by_character`, `PartialCN-Chara`のいずれかを使用してください。それぞれ通常ワークフロー、キャラ別生成フロー、部分ControlNetフローです。

Loraの指定

通常ワークフローでは3種、キャラ別生成フロー・部分コントロールネットフローでは1種のLoraを指定できます。

`PARAM_LORA_x_NAME` にて `"angel\\\\angel_man_test00.safetensors"` か `"angel\\\\angel_woman_test01-000120.safetensors"` のどちらかを指定してください。

`PARAM_LORA_x_STRENGTH` はLoraの強さで、0.0から1.0の値です。

Loraを適用しない場合は `PARAM_LORA_x_STRENGTH` を0.0としてください。

各種パラメータの説明(2)

`"PARAM_PROMPT"` にはプロンプトの文字列を入れてください。

`"PARAM_UPSCALE"` は0.7から1.0の値にしてください。大きい値にするほど生成時の解像度が向上します。（生成時間も長くなります）

`"PARAM_DENOISE"` は0.0から1.0の値にしてください。

ベースモデルの指定

すべてのワークフローにおいて、ベースモデルの指定が可能です。

```
    "parameters":{
        "PARAM_BASEMODEL": "animelike25D_animelike25DV11Pruned.safetensors",
```

`"animelike25D_animelike25DV11Pruned.safetensors"` または
`"cetusMix_Whalefall2.safetensors"` をご使用ください。

ControlNetの指定（通常ワークフロー、キャラ別生成フロー）

通常ワークフローでは3種、キャラ別生成フローにおけるControlNetの説明です。

`"PARAM_CONTROLNET_x_STRENGTH"` はcontrolnetの強さです。それぞれの適用先は以下です。

`"PARAM_CONTROLNET_1_STRENGTH"`:
control_v11p_sd15s2_lineart_anime_fp16.safetensors

`"PARAM_CONTROLNET_2_STRENGTH"`:
lightingBasedPicture_v10.safetensors

`"PARAM_CONTROLNET_3_STRENGTH"`:

ControlNetの指定（部分ControlNet）

部分ControlNetフローについて、LINEARTとTILEについて、マスク内・マスク外それぞれに対し、強度を指定できます。
LIGHTBASEDPICTUREは全体への適用となります。

```
"PARAM_CONTROLNET_LINEART_WITH_MASK_STRENGTH": 0.8,
"PARAM_CONTROLNET_LINEART_OUT_MASK_STRENGTH": 0.4,
"PARAM_CONTROLNET_LIGHTBASEDPICTURE_STRENGTH": 0.4,
"PARAM_CONTROLNET_TILE_WITH_MASK_STRENGTH": 0.6,
"PARAM_CONTROLNET_TILE_OUT_MASK_STRENGTH": 0.4
```

マスクの設定ついて（部分ControlNet）

部分ControlNetフローについて、マスクの指定が必要です。

```
        "PARAM_MASK_PROMPT": "",
        "PARAM_MASK_OBJECT_PROMPT": "face",
```

`"PARAM_MASK_PROMPT"` にはマスク部分に適用するプロンプトをいれてください。 `"PARAM_MASK_OBJECT_PROMPT"` にはマスクを適用したい部位に対するプロンプトをいれてください。

`main.py` の実行

```
python main.py
```

以下が順に実行されます。

- ジョブの投入
- 待ちがある場合待機
- 実行（進捗が表示されます）
- ダウンロード