AlphaFold 3.0.0

AlphaFold3 のモデルパラメータについては各自で申請、ダウンロードしてください。

ウェブページ

https://github.com/google-deepmind/alphafold3

バージョン

3.0.0 (2024/11/26 時点の最新コード; commit id: cdbcf41b71235)

導入環境

- NVIDIA ドライバ 560.35.03
- Miniforge3-Linux-x86_64.sh (miniforge installer)

導入手順(抜粋)

導入時のメモ書きから抜粋した導入手順です。実際の手順とは異なる場合があります。基本的にはリリース直後の情報(11/12 実施)で設定していますが、一部コードの更新に対応するために後から追加実行している部分もあります。

hmmer-3.4

```
$ wget http://eddylab.org/software/hmmer/hmmer-3.4.tar.gz
$ tar zxf hmmer-3.4.tar.gz
$ ./configure --prefix /apl/alphafold/hmmer-3.4/
$ make -j8
$ make install
$ cd easel/
$ make install
```

python 環境(conda+pip)+alphafold3

```
$ sh Miniforge3-Linux-x86_64.sh
...
[.....] >>> /apl/alphafold/miniforge1-3.0.0
...
$ cd /apl/alphafold
$ /apl/alphafold/miniforge1-3.0.0/bin/conda shell.bash hook > af300_init.sh
$ /apl/alphafold/miniforge1-3.0.0/bin/conda shell.csh hook > af300_init.csh
$ vi af300_init.sh
(末尾に以下の行を追加)
export PATH="/apl/alphafold/hmmer-3.4/bin:$PATH"
$ vi af300_init.csh
(末尾に以下の行を追加)
setenv PATH "/apl/alphafold/hmmer-3.4/bin:$PATH"
$ . af300_init.sh
$ conda create -n af3 python=3.11
$ conda activate af3
$ sed -i -e "s/base/af3/" af300_init.sh af300_init.csh
$ conda install -c nvidia absl-py=2.1.0 \
                         chex=0.1.87 \
                         dm-tree=0.1.8 \
                         filelock=3.16.1 \
                         jaxtyping=0.2.34 \
                         jmp=0.0.4 \
                         ml_dtypes=0.5.0 \
                         numpy=2.1.3 \
                         opt_einsum=3.4.0 \
                         pillow=11.0.0 \
                         rdkit=2024.03.5 \
                         scipy=1.14.1 \
                         tabulate=0.9.0 \
```

```
                    toolz=1.0.0 \
                    tqdm=4.67.0 \
                    typeguard=2.13.3 \
                    typing-extensions=4.12.2 \
                    zstandard=0.23.0 \
                    cuda
$ pip install jax[cuda12]==0.4.34 \
        jaxlib==0.4.34 \
        jmp==0.0.4 \
        chex==0.1.87 \
        opt-einsum==3.4.0 \
        dm-haiku==0.0.13 \
        triton==3.1.0 \
        jax-triton==0.2.0
$ cd /some/where
$ git clone https://github.com/google-deepmind/alphafold3.git
$ cd alphafold
$ git pull
$ cd ../
$ cp -r alphafold /apl/alphafold/3.0.0-20241126
$ cd /apl/alphafold
$ ln -s ./3.0.0-20241126 3.0.0
$ cd 3.0.0
$ pip install --no-deps .
$ build_data
```

- メモ: パッケージリストについては厳密な検証を行っていません。不要な操作も入っている可能性があります。
    - hmmer 関連のバイナリは PATH に入れておけば run_alphafold.py で指定する必要無いようです。
    - jax や jax-triton のパッケージは pip のものを使わないと動作しないようでした。

データベース

```
$ cd /apl/alphafold/3.0.0
$ python3 fetch_databases.py --download_destination=/apl/alphafold/databases3/20241112
$ cd /apl/alphafold/databases3/20241112
$ tar xf pdb_2022_09_28_mmcif_files.tar
```

- メモ：11/12 時点でダウンロードしたデータベースでは pdb_2022_09_28_mmcif_files.tar は tar のままで問題ありませんでしたが、コードの更新によって展開されたデータが必要になっていました
    - 上記手順で fetch_databases.py は 2024/11/12 に実行。pdb_2022_09_28_mmcif_files.tar の展開は 2024/11/26 に実行。
    - リリース後に更新された fetch_databases.py ならば pdb_2022_09_28_mmcif_files.tar の展開は自動で行われるため手動対応不要。
    - (ファイルの所有者やパーミッションの変更は上記手順では省略しています)

Lustre システム用、データベースファイルの migration 処理

```
$ lfs migrate -c 2 bfd-first_non_consensus_sequences.fasta
$ lfs migrate -c 10 mgy_clusters_2022_05.fa
$ lfs migrate -c 6 nt_rna_2023_02_23_clust_seq_id_90_cov_80_rep_seq.fasta
$ lfs migrate -c 18 pdb_2022_09_28_mmcif_files.tar
$ lfs migrate -c 2 rnacentral_active_seq_id_90_cov_80_linclust.fasta
$ lfs migrate -c 8 uniprot_all_2021_04.fa
$ lfs migrate -c 5 uniref90_2022_05.fa
```

wrapper script (run-af-300.sh)

```
#!/bin/bash
#
# wrapper for alphafold3 non-container version
#
# NOTE: This script is designed for AlphaFold 3.0.0.

# set default params
AFROOT=/apl/alphafold
AF3ROOT=${AFROOT}/3.0.0 # where run_alphafold.py resides
```

```
# default database path
DB_DIR=${AFROOT}/databases3/20241112

MYOPTS=""
ECHO_COMMAND=false
NO_DB_SPEC=true
DRY_RUN=false

HAS_INPUT=false
HAS_OUTPUT=false

usage() {
 echo ""
 echo "Usage: $0 <OPTIONS>"
 echo ""
 echo "Options:"
 echo "  -j <input json>, --json_path=<input json>"
 echo "    Path to input json file (single input file)."
 echo "  -i <input directory>, --input_dir=<input directory>"
 echo "    Path to directory which contains input json files (multiple input files)."
 echo "  -o <output directory>, --output_dir=<output directory>"
 echo "    Path to output directory."
 echo "  -m <model directory>, --model_dir=<model directory>"
 echo "    Path to model for inference (THIS IS NOT PROVIDED BY RCCS)."
 echo "  -M, --msa, --msaonly"
 echo "    Do only data pipeline (MSA)."
 echo "  -I, --inference, --inferenceonly"
 echo "    Skip MSA and do only inference."
 echo ""
 echo "  Other run_alphafold.py options may also be accepted."

  exit 1
}

while getopts "a:d:ehi:j:o:m:IM-:" c; do
 optarg="${OPTARG}"
 if [[ "$c" = - ]]; then
   c="-${OPTARG%%=*}"
   optarg="${OPTARG/${OPTARG%%=*}/}"
   optarg="${optarg#=}"
   if [[ -z "${optarg}" ]] && [[ ! "${!OPTIND}" = -* ]]; then
     optarg="${!OPTIND}"
     shift
   fi
 fi

  case "-$c" in
   -a|--af3root)
     AF3ROOT=${optarg}
   ;;
   -d|--db_dir)
     MYOPTS="$MYOPTS --db_dir=${optarg}"
     NO_DB_SPEC=false
   ;;
   --dryrun)
     ECHO_COMMAND=true
     DRY_RUN=true
   ;;
   -e|--echo)
     ECHO_COMMAND=true
   ;;
   -i|--input_dir)
     MYOPTS="$MYOPTS --input_dir=${optarg}"
     HAS_INPUT=true
   ;;
   -j|--json_path)
```

```
  MYOPTS="$MYOPTS --json_path=${optarg}"
   HAS_INPUT=true
;;
-h|--help)
  usage
;;
-m|--model_dir)
  MYOPTS="$MYOPTS --model_dir=${optarg}"
;;
-o|--output_dir)
  MYOPTS="$MYOPTS --output_dir=${optarg}"
  HAS_OUTPUT=true
;;
-I|--inference|--inferenceonly|--norun_data_pipeline)
  MYOPTS="$MYOPTS --norun_data_pipeline"
;;
-M|--msa|--msaonly|--norun_inference)
  MYOPTS="$MYOPTS --norun_inference"
;;
--flash_attention_implementation)
  MYOPTS="$MYOPTS --flash_attention_implementation=${optarg}"
 ;;
--jackhmmer_binary_path)
  MYOPTS="$MYOPTS --jackhmmer_binary_path=${optarg}"
;;
--nhmmer_binary_path)
  MYOPTS="$MYOPTS --nhmmer_binary_path=${optarg}"
;;
--hmmalign_binary_path)
  MYOPTS="$MYOPTS --hmmalign_binary_path=${optarg}"
;;
--hmmsearch_binary_path)
  MYOPTS="$MYOPTS --hmmsearch_binary_path=${optarg}"
;;
--hmmbuild_binary_path)
  MYOPTS="$MYOPTS --hmmbuild_binary_path=${optarg}"
;;
--small_bfd_database_path)
  MYOPTS="$MYOPTS --small_bfd_database_path=${optarg}"
;;
--mgnify_database_path)
  MYOPTS="$MYOPTS --mgnify_database_path=${optarg}"
 ;;
--uniprot_cluster_annot_database_path)
  MYOPTS="$MYOPTS --uniprot_cluster_annot_database_path=${optarg}"
;;
--uniref90_database_path)
  MYOPTS="$MYOPTS --uniref90_database_path=${optarg}"
;;
--ntrna_database_path)
  MYOPTS="$MYOPTS --ntrna_database_path=${optarg}"
;;
--rfam_database_path)
  MYOPTS="$MYOPTS --rfam_database_path=${optarg}"
;;
--rna_central_database_path)
  MYOPTS="$MYOPTS --rna_central_database_path=${optarg}"
;;
--pdb_database_path)
  MYOPTS="$MYOPTS --pdb_database_path=${optarg}"
;;
--seqres_database_path)
  MYOPTS="$MYOPTS --seqres_database_path=${optarg}"
 ;;
--jackhmmer_n_cpu)
```

```
    MYOPTS="$MYOPTS --jackhmmer_n_cpu=${optarg}"
  ;;
  --nhmmer_n_cpu)
    MYOPTS="$MYOPTS --nhmmer_n_cpu=${optarg}"
  ;;
  --jax_compilation_cache_dir)
    MYOPTS="$MYOPTS --jax_compilation_cache_dir=${optarg}"
  ;;
 esac
done

if "${NO_DB_SPEC}"; then
 MYOPTS="$MYOPTS --db_dir=${DB_DIR}"
fi

if "${ECHO_COMMAND}"; then
 echo "python ${AF3ROOT}/run_alphafold.py $MYOPTS"
 exit 0
fi

if ! "${HAS_INPUT}" || ! "${HAS_OUTPUT}"; then
 echo "ERROR: Input data (-j or -i) and output directory (-o) must be specified."
 usage
fi

if ! "${DRY_RUN}"; then
 python ${AF3ROOT}/run_alphafold.py $MYOPTS
fi
```

サンプル実行手順

システムの都合上、data pipeline と推論部分を別々に実行した方が安全です。

data pipeline (MSA)

```
#!/bin/sh
#PBS -l select=1:ncpus=64:mpiprocs=1:ompthreads=64
#PBS -l walltime=72:00:00

if [ ! -z "${PBS_O_WORKDIR}" ]; then
 cd "${PBS_O_WORKDIR}"
fi

module -s purge
. /apl/alphafold/af300_init.sh
RUNAF3=/apl/alphafold/run-af-300.sh

$RUNAF3 \
  -j af3-input.json \
  -o af_output \
  -M
```

- インプットの json ファイルについてはこちらのページの内容をご確認ください。

推論

```
#!/bin/sh
#PBS -l select=1:ncpus=16:mpiprocs=1:ompthreads=16:ngpus=1
#PBS -l walltime=72:00:00

if [ ! -z "${PBS_O_WORKDIR}" ]; then
 cd "${PBS_O_WORKDIR}"
fi

module -s purge
. /apl/alphafold/af300_init.sh
```

```
RUNAF3=/apl/alphafold/run-af-300.sh
$RUNAF3 \
  -j af_output/2pv7/2pv7_data.json \
  -o af_output \
  -m ~/AlphaFold3/models \
  -I
```

- 推論時のインプット json ファイルは上記 data pipeline の結果として出力されたものになります
- モデルパラメータファイルは各自で用意してください