GTNet

Release 0.0.1

unknown

Mar 05, 2024

GETTING STARTED

The Genome Taxonomy Network, or *GTNet*, is a taxonomic classifier that uses a deep neural network to label DNA sequences with the Genome Taxonomy Database taxonomy.

CHAPTER

ONE

INSTALLATION

GTNet is available on the Python Package Index.

```
pip install gtnet
```

1.1 GPU acceleration

GTNet uses PyTorch, so it is capable of GPU acceleration with CUDA. As long as CUDA is available on your system, GTNet will detect if CUDA is available and make GPU acceleration available.

If your system is equipped with NVIDIA GPUs, but are unsure if CUDA is installed, we recommend installing PyTorch and the CUDA Toolkit using Conda.

For example, if you would like to run PyTorch with CUDA Toolkit 11.8, you can run the following commands:

```
conda create -n gtnet-env
conda activate gtnet-env
conda install pytorch pytorch-cuda=11.8 -c pytorch -c nvidia
pip install gtnet
```

CHAPTER

TWO

RUNNING GTNET

GTNet comes with multiple commands. The simplest way of running GTNet is to use the `classify` command.

```
gtnet classify genome.fna > genome.tax.csv
```

This command generates one classification for the entire file, and should be used to get classification for metagenome bin. Use the `-s/--seqs` flag to get classifications for the individual sequences in `genome.fna`

Attention: The first time you run `classify` and `predict` (see below), the model file will be downloaded and stored in the same directory that the *gtnet* package is installed in. Therefore, for the this to be successful, you must have write privileges on the directory that *gtnet* is installed in.

```
gtnet classify --seqs genome.fna > genome.seqs.tax.csv
```

The `classify` command can take multiple fasta files, and will produce line per file in the output. For example, the following command will contain two lines:

```
gtnet classify bin1.fna bin2.fna > bins.tax.csv
```

2.1 GTNet steps

GTNet consists of two main steps: 1) get scored predictions of taxonoimc assignments and 2) filter scored predictions. The previous command combines these two commands into a single command with a default false-positive rate. The two steps have been separated into two commands for those who want to experiment with different false-positive rates.

2.1.1 Getting predictions

To get predictinos for all sequences in a Fasta file, use the `predict` subcommand. This command also accepts multiple fasta files and the `-s/--seqs` argument for getting predictions for individual sequences.

```
gtnet predict genome.fna > genome.tax.raw.csv
```

2.1.2 Filtering predictions

After getting predicted and scored taxonomic classifications, you can filter the raw classifications to a desired false-positive rate.

```
gtnet filter --fpr 0.05 genome.tax.raw.csv > genome.tax.csv
```

The `filter` command supports predictions for whole files and individual sequences.

2.2 GPU acceleration

If CUDA is available on your system, the `classify` and `predict` commands will have the option `-g/--gpu` to enable using the available GPU to accelerate neural network calculations.

CHAPTER

THREE

API DOCUMENTATION

3.1 gtnet.classify module

gtnet.classify.**classify**(*argv=None*)

Get taxonomic classification for each sequence in a Fasta file.

Parameters

argv (*Namespace, default=sys.argv*) – The command-line arguments to use for running this command

3.2 gtnet.predict module

gtnet.predict.**predict**(*argv=None*)

Get network predictions for each sequence in Fasta file

Parameters

argv (*Namespace, default=sys.argv*) – The command-line arguments to use for running this command

gtnet.predict.**run_torchscript_inference**(*fastas*, *model*, *conf_models*, *window*, *step*, *vocab*, *seqs=False*, *n_chunks=10000*, *device=device(type='cpu')*, *logger=None*)

Run Torchscript inference

Parameters

- **fastas** (*str*) – The path to the Fasta file with sequences to do inference on
- **model** (*RecursiveScriptModule*) – The Torchscript model to run inference with
- **conf_models** (*dict*) – A dictionary with the confidence model for each taxonomic level. Each model should be a RecursiveScriptModule. The expected keys in this dict are 'domain', 'phylum', 'class', 'order', 'family', 'genus' and 'species'.
- **window** (*int*) – The length of the sliding window to use for doing inference
- **step** (*int*) – The length of the step of the sliding window to use for doing inference
- **vocab** (*str*) – The vocabulary used for training *model*
- **n_chunks** (*int, default=10000*) – The length of the step of the sliding window to use for doing inference
- **device** (*device, default=torch.device('cpu')*) – The Pytorch device to run inference on

- **logger** (*Logger*) – The Python logger to use when running inference

3.3 gtnet.filter module

gtnet.filter.**get_cutoffs**(*rocs*, *fpr*)

Get score cutoffs to achieve desired false-positive rate

Parameters

- **rocs** (*dict*) – The ROC curves for each taxonomic level
- **fpr** (*float*) – The false-positive rate to get the score for

gtnet.filter.**filter**(*argv=None*)

Filter raw taxonomic classifications

gtnet.filter.**filter_predictions**(*pred_df*, *cutoffs*)

Filter taxonomic classification predictions

Parameters

- **pred_df** (*DataFrame*) – The DataFrame containing predictions and confidence scores for each taxonomic level
- **cutoffs** (*dict*) – A dictionary containing the confidence score cutoff for each taxonomic level

3.4 gtnet.utils module

gtnet.utils.**parse_logger**(*string*)

gtnet.utils.**get_logger**()

class gtnet.utils.**DeployPkg**

Bases: object

A class to handle loading and manipulating the deployment package

classmethod check_pkg()

path(*path*)

Map paths to be relative to current working directory

property manifest

__getitem__(*key*)

gtnet.utils.**load_deploy_pkg**(*for_predict=False*, *for_filter=False*, *contigs=False*)

class gtnet.utils.**GPUModel**(*model*, *device*)

Bases: Module

Initialize internal Module state, shared by both nn.Module and ScriptModule.

forward(*x*)

Define the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

training: bool

gtnet.utils.**check_cuda**(*parser*)

gtnet.utils.**check_device**(*args*)

gtnet.utils.**write_csv**(*output*, *args*)

3.5 gtnet package

3.5.1 Submodules

gtnet.main module

class gtnet.main.**Command**(*module*, *doc*)

Bases: object

get_func()

gtnet.main.**print_help**()

gtnet.main.**run**()

gtnet.sequence module

class gtnet.sequence.**FastaSequenceEncoder**(*window*, *step*, *vocab=None*, *padval=None*, *min_seq_len=100*, *device=device(type='cpu')*)

Bases: object

encode(*seq*)

classmethod get_dna_map(*vocab=None*)

Create data structures for mapping DNA sequence to

Returns

vocab: the DNA vocabulary used for building the data structures basemap: a 128 element array for mapping ASCII character values to encoded values rcmap: an array for mapping between complementary characters of encoded values

classmethod get_revcomp_map(*vocab*)

class gtnet.sequence.**FastaReader**(*encoder*, **fastas*, *parallel=False*)

Bases: Process

3.5.2 Module contents

modindex

CHAPTER

FOUR

UPDATING GTNET

As the GTDB taxonomy is updated, GTNet will also need to be updated. This amounts to retraining the network with the new taxonomy and updating the gtnet software to use the new model and taxonomy.

4.1 Training a new model

Software for training GTNet is available in the deep-taxon repository.

4.1.1 Uploading to OSF

Once a model is trained, calibrated, and packaged, the deployment package needs to be made publicly available. GTNet is currently carried hosted on OSF.

4.2 Updating the gtnet software

After training a new model and packaging the model, the `DeployPkg` class will need to be updated with the new URL and checksum of the new deployment package. This can be done starting around here in the code.

CHAPTER

FIVE

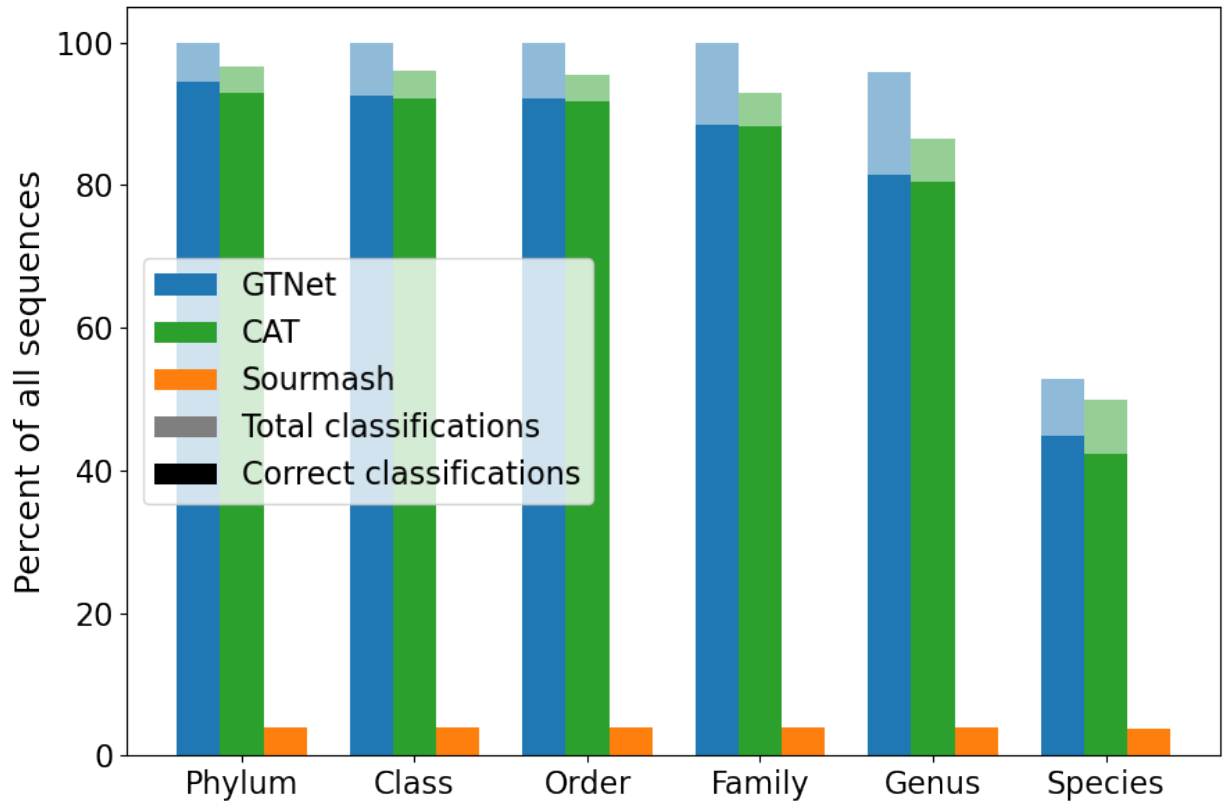
GTNET PERFORMANCE

Attention: This page is currently under construction. The results presented here may not accurately reflect what is said in text.

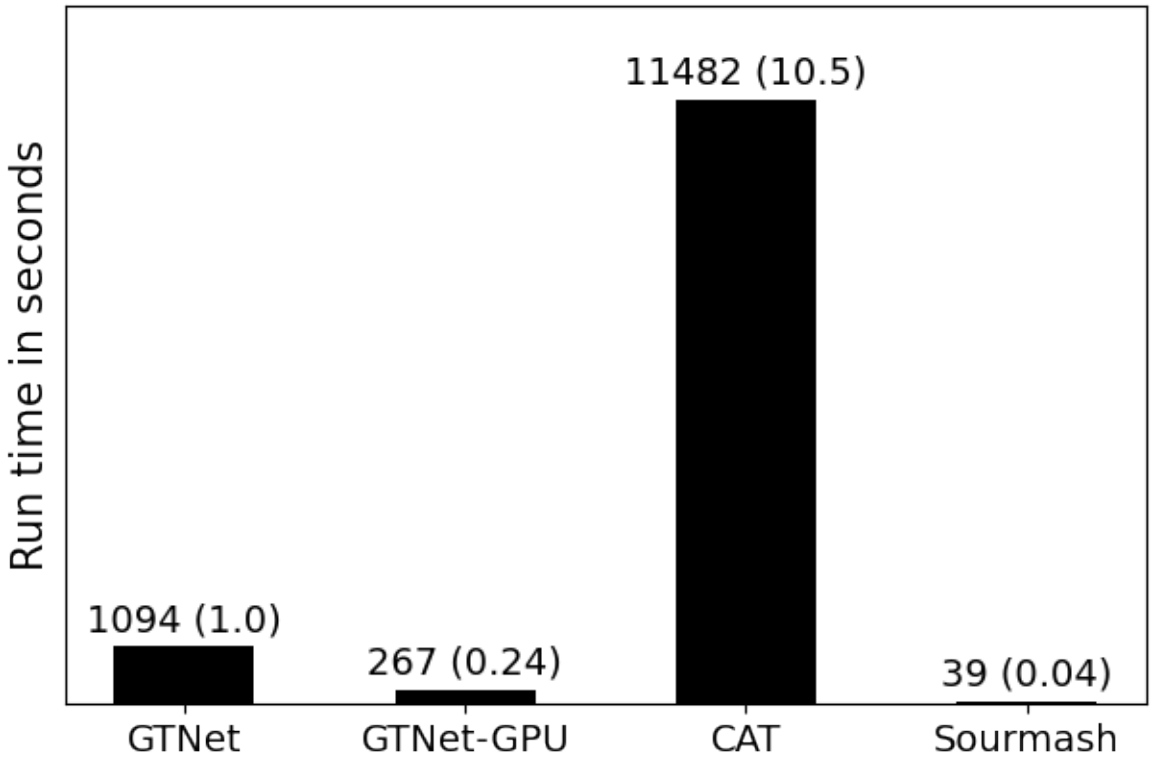
Taxonomic classifiers fall into two main categories: fast-and-incomplete or slow-and-complete. GTNet strives to be both fast and complete. In this page, we demonstrate GTNet capabilities by comparing to state-of-the-art methods from each of these categories. We compare to Sourmash, a fast-and-incomplete method, and CAT, a slow-and-complete method.

Our choice of tools for comparison should not be perceived as a criticism or an endorsement for either tool. These tools were chosen based on their ease of use for labelling contigs with the GTDB taxonomy and the algorithmic approaches underlying these tools.

Here are accuracy comparisons for a subset of non-representative GTDB taxa.



Here are speed comparisons for a subset of 40 non-representative genomes.



CHAPTER

SIX

COPYRIGHT

CHAPTER

SEVEN

LICENSE

PYTHON MODULE INDEX

g

INDEX

Symbols

C

D

E

F

G

L

M

P

R

T

W