

GENOME UNDERSTANDING USING LLMS

Dr. Muhammad Aammar Tufail

PostDoc (Bioinformatician)

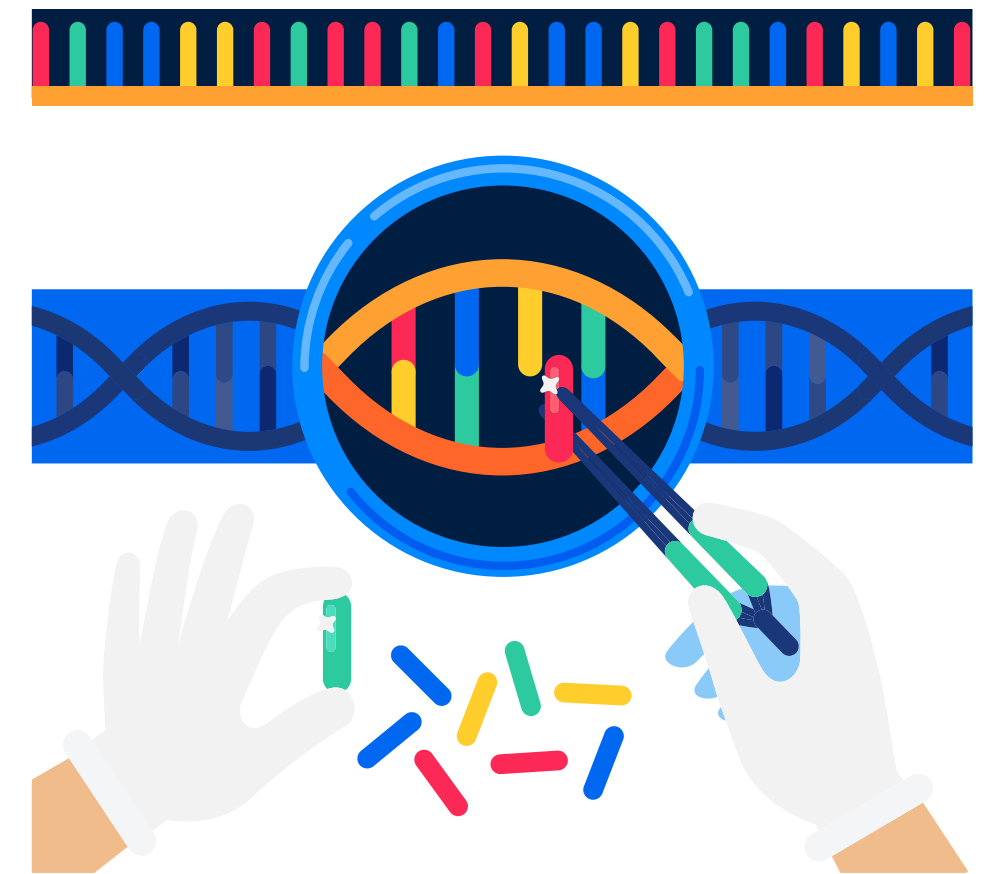
Institute for General Microbiology,

CAU Kiel.

Why?

Students spend many hours just to find the specific information in a genome.

- Chat with Genomes
- Find Answers to your questions
- Find Gene Functions
- Find the Gene Clusters
- Find the Small or Large Protein sequences
- Sort or filter specific genes
- much more...



file format

fasta

```
>VIT_201s0011g03530.1
AATTAAGCATAAATACTCACTCTTACCCCTTATTTTCTTATCTCTCATCACTTTTGGTGCGAAG
GACCATGAGAACAAGCTGCAATGGGTGTAGGGTTCTTCGCAAGGCATGCAGCCAAGACTGCATCA

>VIT_201s0011g03540.1
CAGGTAGCGTGAAGTTAAACCCTAGCGCTTTAGACAAACAGCTGTAGTCACCGCCCACAAACACC
AGCCTCTGAGACACCACCTCAAACCTTTCCACTTAAATACACATCCCTCACACCCTTTTCAATTC

>VIT_201s0011g03550.1
CATGCAAAGCTGAACGCGATGCTGTGATTGGTGGTAAGTGGTAGTTGAGTAAATTTGACAGTGAA
GCCGAAATGGTAAAAGACTAAGGCTAGAAGTAGAATACCACTGTTCTTCTCATCACGTGGGCCCA
```

Header —●>VIT_201s0011g03530.1

Sequence —●AATTAAGCATAAATACTCACTCTTACCCCTTATTTTCTTATCTCTCATCACTTTTGGTGCGAAG

 —●GACCATGAGAACAAGCTGCAATGGGTGTAGGGTTCTTCGCAAGGCATGCAGCCAAGACTGCATCA

Header —●>VIT_201s0011g03540.1

Sequence —●CAGGTAGCGTGAAGTTAAACCCTAGCGCTTTAGACAAACAGCTGTAGTCACCGCCCACAAACACC

 —●AGCCTCTGAGACACCACCTCAAACCTTTCCACTTAAATACACATCCCTCACACCCTTTTCAATTC

Header —●>VIT_201s0011g03550.1

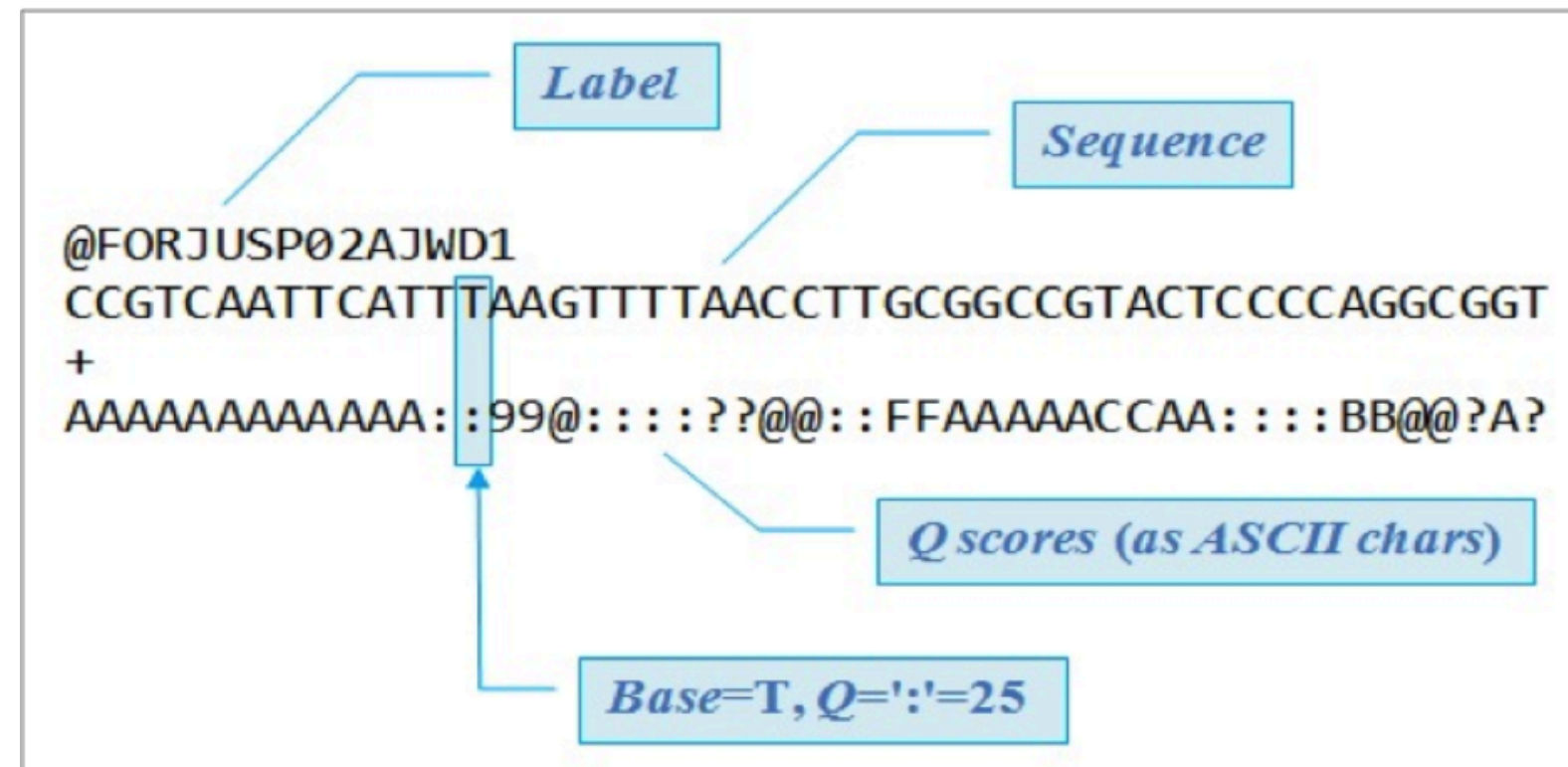
Sequence —●CATGCAAAGCTGAACGCGATGCTGTGATTGGTGGTAAGTGGTAGTTGAGTAAATTTGACAGTGAA

 —●GCCGAAATGGTAAAAGACTAAGGCTAGAAGTAGAATACCACTGTTCTTCTCATCACGTGGGCCCA

Don't be confused in these files.
These are just text files!



fastq



fasta

Header —●>VIT_201s0011g03530.1

Sequence —●AATTAAGCATAAATACTCACTCTTACCCCTTATTTTCTTATCTCTCATCACTTTTGGTGCGAAG
●GACCATGAGAACAAAGCTGCAATGGGTGTAGGGTTCTTCGCAAGGCATGCAGCCAAGACTGCATCA

Header —●>VIT_201s0011g03540.1

Sequence —●CAGGTAGCGTGAAGTTAAACCCTAGCGCTTTAGACAAACAGCTGTAGTCACCGCCCACAAACACC
●AGCCTCTGAGACACCACCTCAAACCTTTCCACTTAAATACACATCCCTCACACCCTTTTCAATTC

Header —●>VIT_201s0011g03550.1

Sequence —●CATGCAAAGCTGAACGCGATGCTGTGATTGGTGGTAAGTGGTAGTTGAGTAAATTTGACAGTGAA
●GCCGAAATGGTAAAAGACTAAGGCTAGAAGTAGAATACCACTGTTCTTCTCATCACGTGGGCCCA

File Formats:

1. **FASTA:** Used to store sequences in a plain text format, with a header line starting with ">" and the sequence data following on the next line(s).
2. **GenBank:** A text-based format for storing DNA and protein sequences along with their annotation information.
3. **SAM/BAM:** Binary file formats for storing alignments of next-generation sequencing (NGS) data to a reference genome.
4. **VCF:** A text-based format for storing variant information from NGS data.
5. **GFF/GTF:** Formats for storing gene annotation information in a tab-delimited text format.
6. **BED:** A format for storing genomic interval information in a simple tab-delimited text format.
7. **FASTQ:** A format for storing NGS data as sequences and quality scores in a plain text format.
8. **WIG (Wiggle) files** are a type of bioinformatics file format used to store continuous-valued data such as GC content or read coverage along a genome.

Natural Language vs. DNA sequence

Aspect	DNA Data	Natural Language
Data Nature	Sequences of nucleotides (A, T, C, G)	Sequences of words or characters
Sequence Length	Often longer sequences	Typically shorter sequences
Complexity	Less complex alphabet but higher order structure	More complex alphabet and semantic structure
Task Example	Gene prediction, mutation analysis	Text generation, sentiment analysis
Error Tolerance	Low tolerance for errors (high precision needed)	Higher tolerance for variability
Output	Predictive models output biological insights	Outputs are often human-readable text
Training Challenges	Requires alignment and handling of large genomes	Requires understanding context and semantics
Applications	Bioinformatics, medical research	Content generation, dialogue systems, analysis

Current Models for DNA Data

DNA foundation models made significant progress in decoding the linguistic intricacies of the genome.

- DNABERT
- DNABERT2
- Nucleotide Transformer (NT)



DNABERT

This repository includes the implementation of 'DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome'. Please cite our paper if you use the models or codes. The repo is still actively under development, so please kindly report if there is any issue encountered.

Quantitative Biology > Genomics

[Submitted on 26 Jun 2023 (v1), last revised 18 Mar 2024 (this version, v2)]

DNABERT-2: Efficient Foundation Model and Benchmark For Multi-Species Genome

[Zhihan Zhou](#), [Yanrong Ji](#), [Weijian Li](#), [Pratik Dutta](#), [Ramana Davuluri](#), [Han Liu](#)

The Nucleotide

Transformer: Building and Evaluating Robust Foundation Models for Human Genomics

Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza Revilla, Nicolas Lopez Carranza, Adam Henryk Grzywaczewski, Francesco Oteri, Christian Dallago, Evan Trop, Hassan Sirelkhatim, Guillaume Richard, Marcin Skwark, Karim Beguir, Marie Lopez, Thomas Pierrot

doi: <https://doi.org/10.1101/2023.01.11.523679>

Fine Tuning DNABERT2 Model


Load the model



```
1 import torch
2 from transformers import AutoTokenizer, AutoModel
3 tokenizer = AutoTokenizer.from_pretrained("zhihan1996/DNABERT-2-117M", trust_remote_code=True)
4 model = AutoModel.from_pretrained("zhihan1996/DNABERT-2-117M", trust_remote_code=True)
```

Fine Tuning DNABERT2 Model

Calculate the embeddings of a DNA Sequence



```
1 dna = "ACGTAGCATCGGATCTATCTATCGACACTTGGTTATCGATCTACGAGCATCTCGTTAGC"
2 inputs = tokenizer(dna, return_tensors = 'pt')['input_ids']
3 hidden_states = model(inputs)[0] # [1, sequence_length, 768]
4
5 # embedding with mean pooling
6 embedding_mean = torch.mean(hidden_states[0], dim=0)
7 print(embedding_mean.shape) # expect to be 768
8
9 # embedding with max pooling
10 embedding_max = torch.max(hidden_states[0], dim=0)[0]
11 print(embedding_max.shape) # expect to be 768
```

Fine Tuning DNABERT2 Model

Fine Tuning following the instructions:

6.2.1 Format your dataset

First, please generate 3 `csv` files from your dataset: `train.csv`, `dev.csv`, and `test.csv`. In the training process, the model is trained on `train.csv` and is evaluated on the `dev.csv` file. After the training is finished, the checkpoint with the smallest loss on the `dev.csv` file is loaded and evaluated on `test.csv`. If you do not have a validation set, please just make the `dev.csv` and `test.csv` the same.

Please see the `sample_data` folder for an example of data format. Each file should be in the same format, with the first row as document head named `sequence, label`. Each following row should contain a DNA sequence and a numerical label concatenated by a `,` (e.g., `ACGTCAGTCAGCGTACGT, 1`).

Other ways to fine tune

To use these DNA foundation models for our purposes to understand DNA we have to:

Pre-train + Fine tune

i-e: pre-training on unlabeled genomic sequences, and then adaptation to a particular genome understanding task.



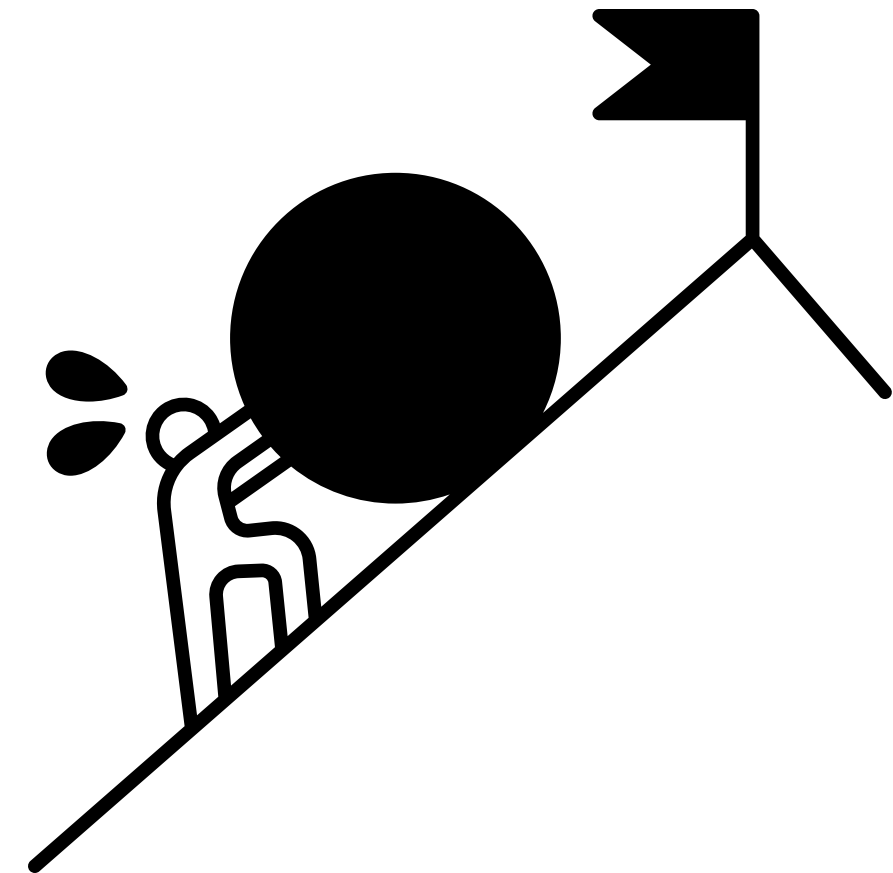
Challenges

Genome annotation and downstream tasks depends on the number and diversity of Genomes.



For example, state-of-the-art deep learning models in epigenetics alone can encompass nearly 22,000 individual tasks

A Significant Challenge!



Challenges

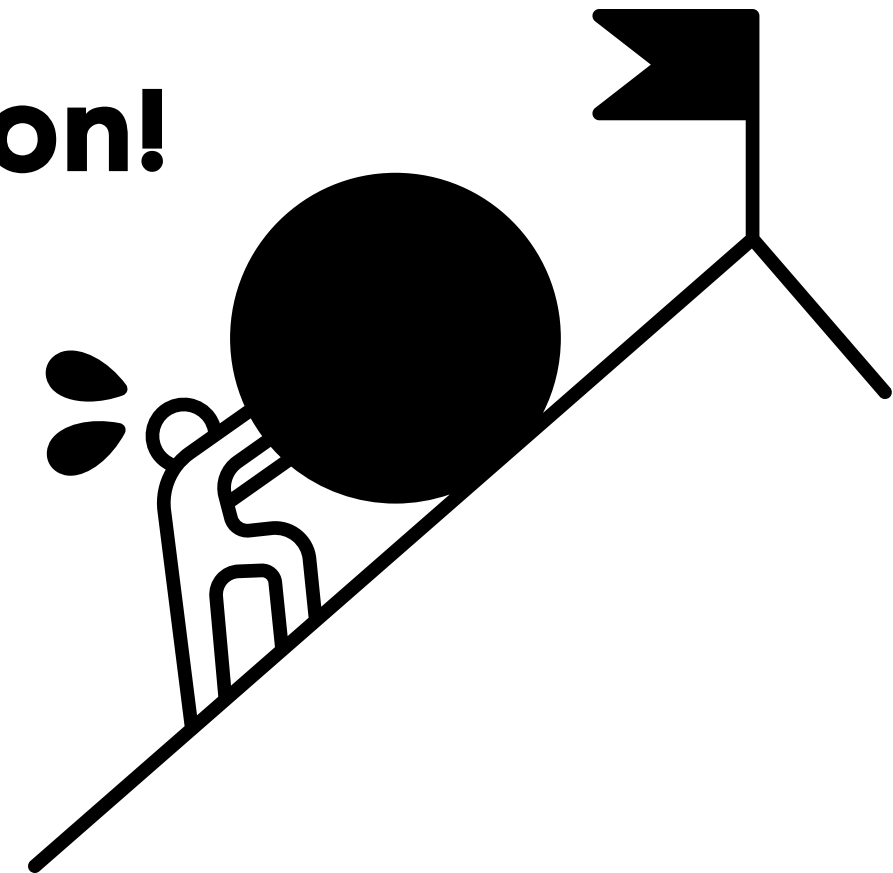
Genome annotation and downstream tasks depends on the number and diversity of Genomes.



For example, state-of-the-art deep learning models in epigenetics alone can encompass nearly 22,000 individual tasks

A Significant Challenge is extensive computation!

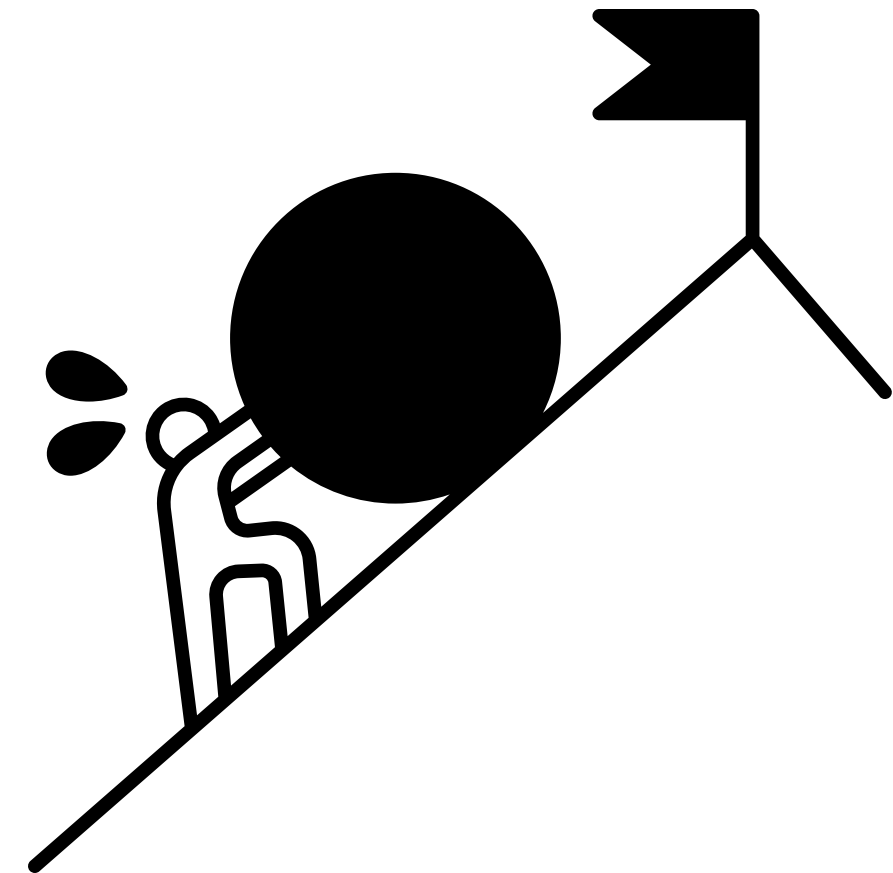
As models grow in size and complexity, the practice of full-model fine-tuning (FMFT) – entailing the retraining of every model parameter for each task – becomes increasingly impractical for genomic studies.



Challenges



Using one of the largest **NT** models with 2.5B parameters as an example - deploying independent instances of fine-tuned models, each with 2.5B parameters, is prohibitively resource intensive.



Possible Solutions

To fine tune DNA foundational models there two solutions:

- 1.model compression** to reduce model size
- 2.parameter-efficient fine-tuning (PEFT)** to add task-specific adapters in the small parameter regime

Possible Solutions

To fine tune DNA foundational models there two solutions:

- 1.model compression** to reduce model size
- 2.parameter-efficient fine-tuning (PEFT)** to add task-specific adapters in the small parameter regime

Model compression approaches have been well-established in recent years, implementing them on large language models can be very expensive, as these techniques typically necessitate full model fine tuning.

Possible Solutions

To fine tune DNA foundational models there are two solutions:

1. **model compression** to reduce model size
2. **parameter-efficient fine-tuning (PEFT)** to add task-specific adapters in the small parameter regime

PEFTs focus on fine-tuning the model on only a small number of additional parameters, significantly decreasing the computational costs.

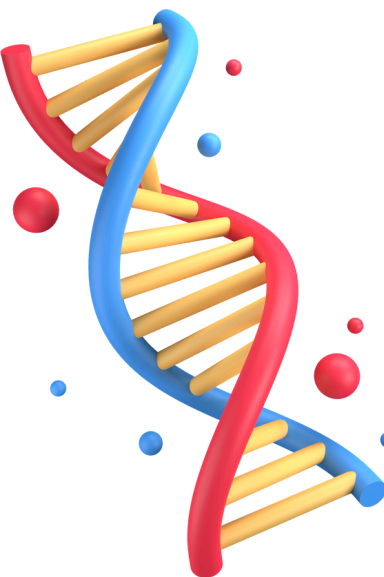
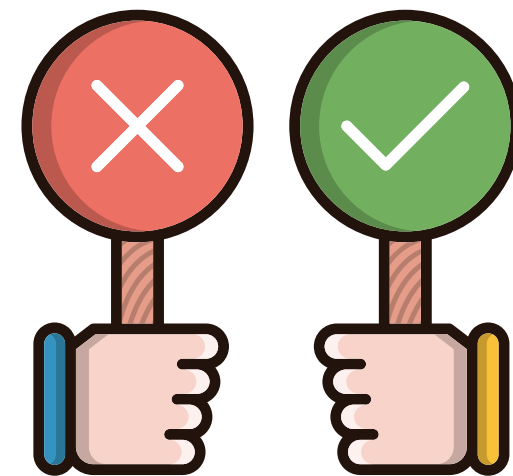
In PEFTs, these are increasingly prominent

- low-rank adaptation, e.g., low-rank adapters (**LoRA**)
- adaptive low-rank adaptation (**AdaLoRA**)

LLMs for DNA

The idea of large language models as universal compute machines has been demonstrated effective for NLPs and images, but not yet for DNAs.

- Pre-trained Natural Language Foundational models are initially trained on vast amounts of **general language data** and thus develop a **strong understanding of natural language**.
- However, genomic sequences, despite being sequential and preserving complex patterns, **do not conform to the rules of human language**.
- The **effective tokenization and context length** in genomics are still debatable



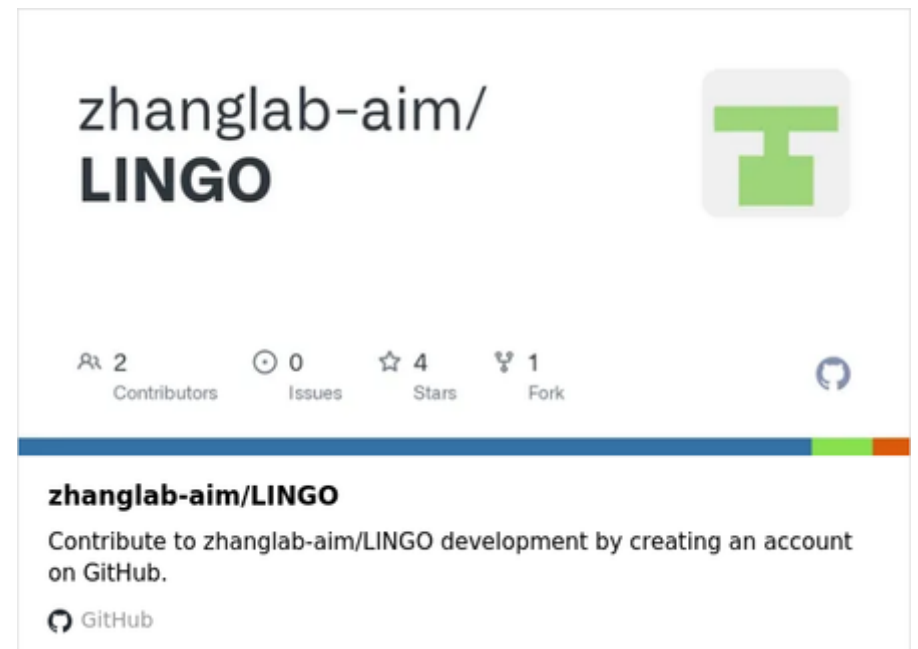
LLMs for DNA

We can use Genome-Specific Adapters to improve LLMs use for Genome understanding.

Efficient and Scalable Fine-Tune of Language Models for Genome Understanding

Parameter-Efficient Fine-Tuning (PEFT) has become the de facto approach to fine-tune PFM while decreasing the computational costs. The current status of PEFT includes:

1. Prefix Tuning methods, e.g., [Prefix-Tuning: Optimizing Continuous Prompts for Generation](#), [P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks](#)
2. Prompt Tuning methods, e.g., [The Power of Scale for Parameter-Efficient Prompt Tuning](#)
3. Low-rank adaptation method, e.g., [LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS](#) and AdaLoRA: [Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning](#)



LLMs for DNA

AdaLoRa for fine tuning LLMs

arXiv > cs > arXiv:2303.10512

Search

Help

Computer Science > Computation and Language

[Submitted on 18 Mar 2023 (v1), last revised 20 Dec 2023 (this version, v2)]

AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, Tuo Zhao

Fine-tuning large pre-trained language models on downstream tasks has become an important paradigm in NLP. However, common practice fine-tunes all of the parameters in a pre-trained model, which becomes prohibitive when a large number of downstream tasks are present. Therefore, many fine-tuning methods are proposed to learn incremental updates of pre-trained weights in a parameter efficient way, e.g., low-rank increments. These methods often evenly distribute the budget of incremental updates across all pre-trained weight matrices, and overlook the varying importance of different weight parameters. As a consequence, the fine-tuning performance is suboptimal. To bridge this gap, we propose AdaLoRA, which adaptively allocates the parameter budget among weight matrices according to their importance score. In particular, AdaLoRA parameterizes the incremental updates in the form of singular value decomposition. Such a novel approach allows us to effectively prune the singular values of unimportant updates, which is essentially to reduce their parameter budget but circumvent intensive exact SVD computations. We conduct extensive experiments with several pre-trained models on natural language processing, question answering, and natural language generation to validate the effectiveness of AdaLoRA. Results demonstrate that AdaLoRA manifests notable improvement over baselines, especially in the low budget settings. Our code is publicly available at [this https URL](#).

Next plan is to fine tune LINGO

Language prefix fine-tuning for GenOmes

Efficient and Scalable Fine-Tune of Language Models for Genome Understanding

Huixin Zhan¹, Ying Nian Wu², Zijun Zhang^{1,3*}

¹Division of Artificial Intelligence in Medicine, Cedars-Sinai Medical Center, Los Angeles, 90048, CA, USA.

²Department of Statistics, University of California, Los Angeles, Los Angeles, 90095, CA, USA.

³Department of Computational Biomedicine, Cedars-Sinai Medical Center, Los Angeles, 90048, CA, USA.

*Corresponding author(s). E-mail(s): zijun.zhang@cshs.org;

Contributing authors: huixin.zhan@cshs.org; ywu@stat.ucla.edu;

Models support matrix

Find models that are supported out of the box below.

Model	LoRA	AdaLoRA	Adaptive rank sampling	LINGO + one-hot	LINGO + BBPE
1000G-500M	✓	✓	✓		
DNABERT-2	✓	✓	✓		
OPT	✓	✓	✓	✓	✓
LLaMA	✓	✓	✓		

Computationally Extensive!

Currently available only for NVIDIA not for M1 Max Mac.



