Genomic Language Models: From Basics to Evo2

Junhao Liu

April 16, 2025

Outline

- Language Models Pre-training
  - Transformer Revisit
  - Masked Language Modeling (MLM)
  - Causal Language Modeling (CLM)
  - Language models vs. vision models
- Genomic Language Models (gLMs)
  - Transformer-based gLMs
    - DNA-BERT, DNA-BERT v2, Nucleotide Transformer series, GENErator, etc.
  - Non-Transformer-based gLMs
    - Hyena-DNA, Caduceus, Evo series, etc.
- Fine-tuning Techniques
  - Parameter-efficient fine-tuning methods (PEFTs)
  - Distributed Training

Language Models Pre-training

Transformer Revisit

- It was originally designed for sequence-to-sequence (seq2seq) tasks
  - E.g., machine translation
    - Input: a sentence in English
    - Output: a translation in Chinese
  - It consists of an Encoder and a Decoder

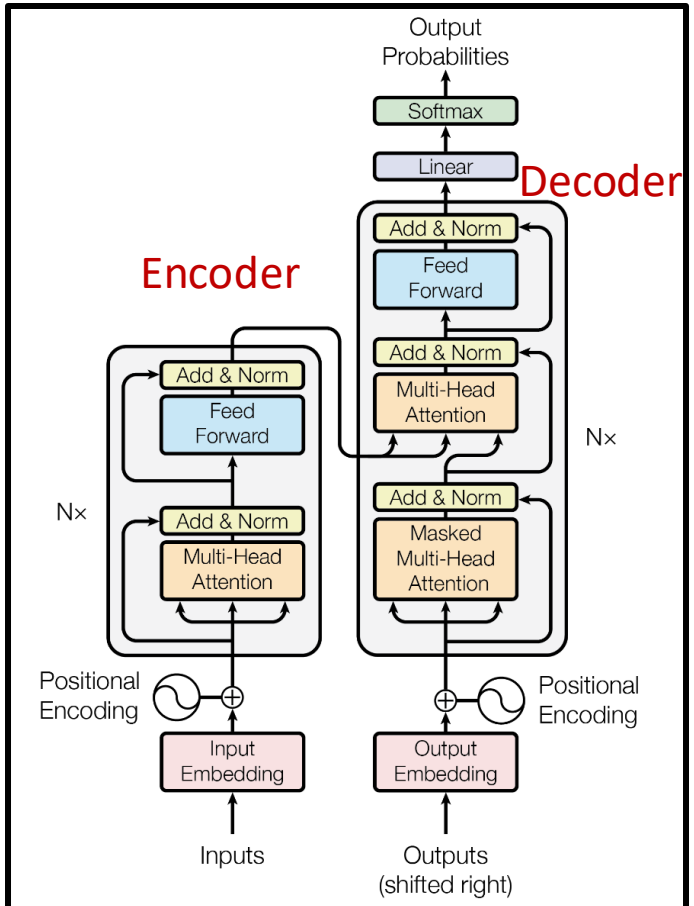
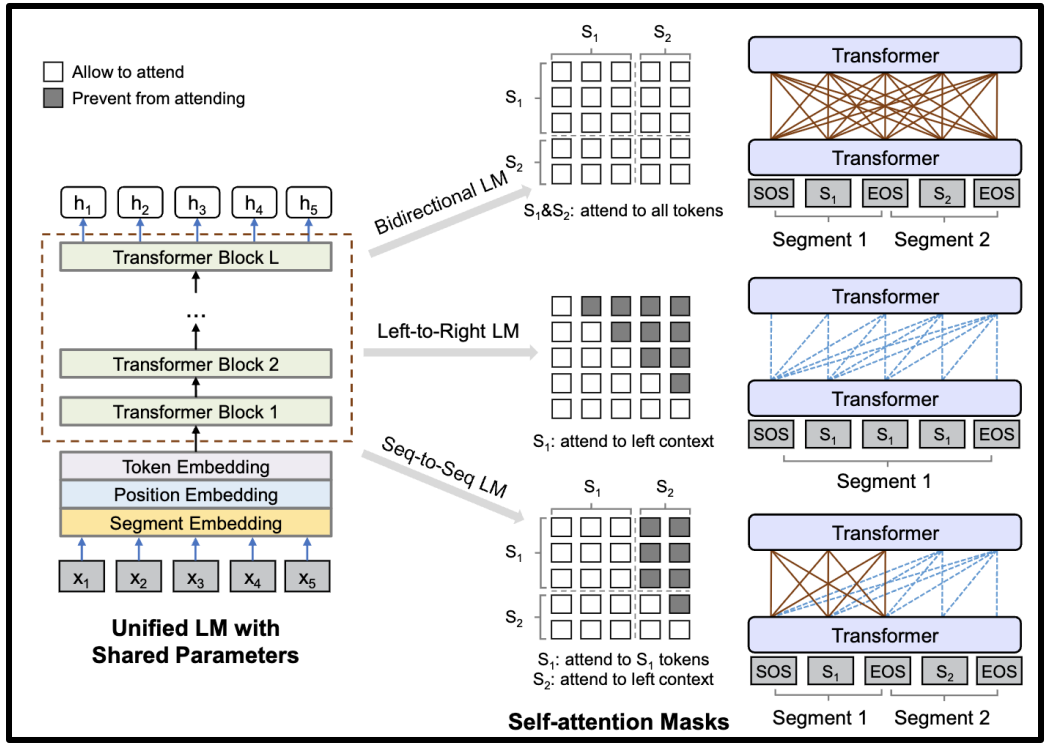


Figure 1: The Transformer - model architecture.

Attention Mask

- Attention mask enables flexible probabilistic modeling on what conditions are considered on.



Ref: Unified Language Model Pre-training for Natural Language Understanding and Generation (NeurIPS 2019)

Attention Mask – Flex Attention

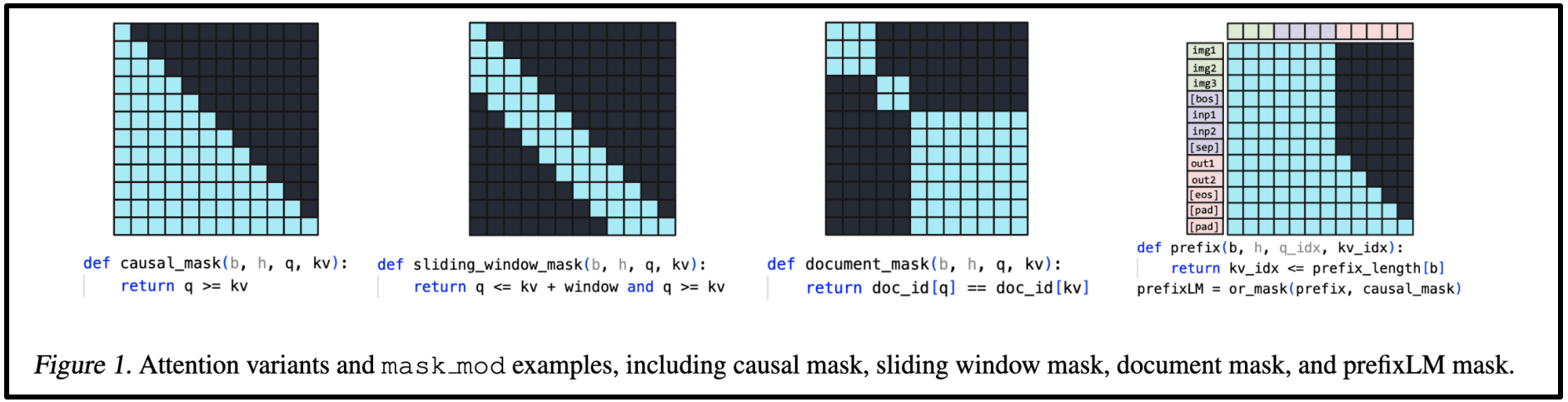


Figure 1. Attention variants and `mask_mod` examples, including causal mask, sliding window mask, document mask, and prefixLM mask.

- Different usages depending on the problem you are working on
  - document mask trains multiple documents simultaneously and requires a token to only attend to other tokens in the same document

Ref: FLEX ATTENTION: A PROGRAMMING MODEL FOR GENERATING OPTIMIZED ATTENTION KERNELS (Arxiv 2024)

Language Modeling – Generation (Open AI GPT, June 2018)

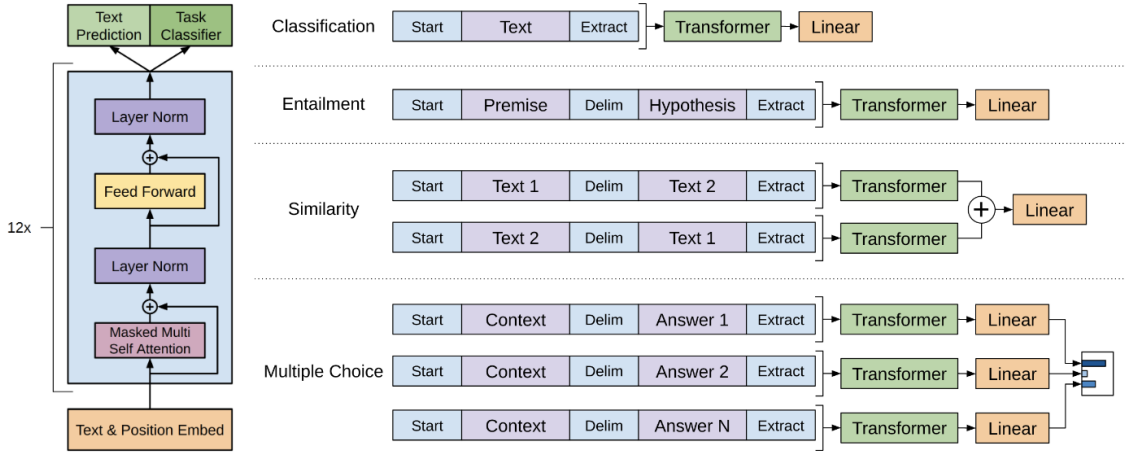
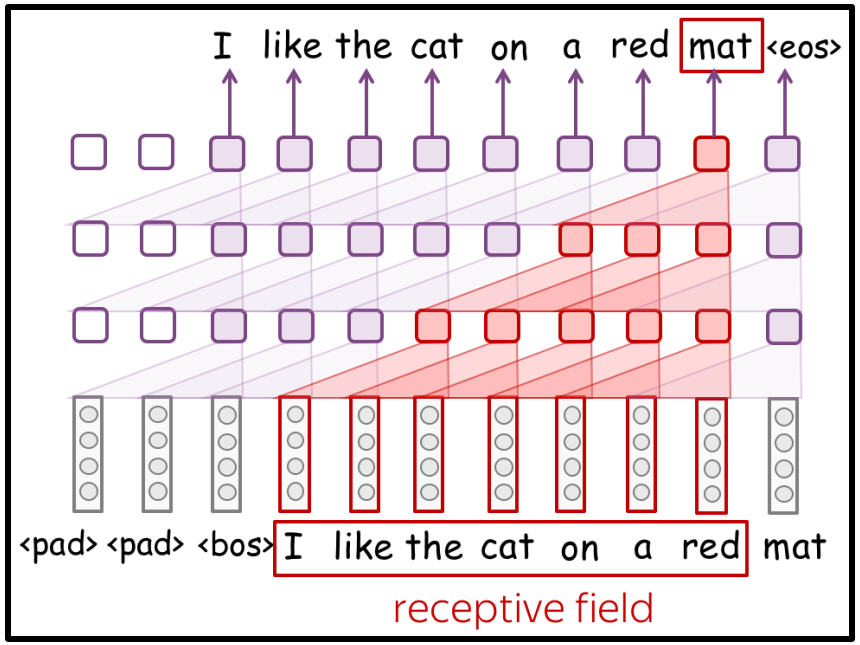


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.



- Pretraining corpus: BooksCorpus dataset including 7,000 unique unpublished books
- Model size: 12-layer decoder blocks, 12 attention heads, 768 dimensional states -> 117 million parameters.

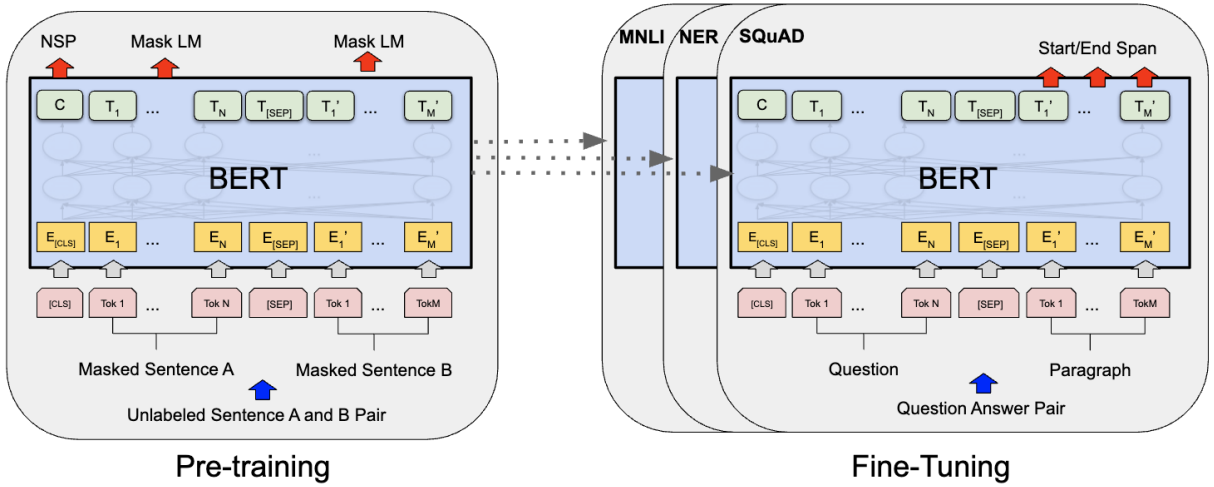
Given an unsupervised corpus of tokens $\mathcal{U} = \{u_1, \ldots, u_n\}$, we use a standard language modeling objective to maximize the following likelihood:

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta) \quad (1)$$

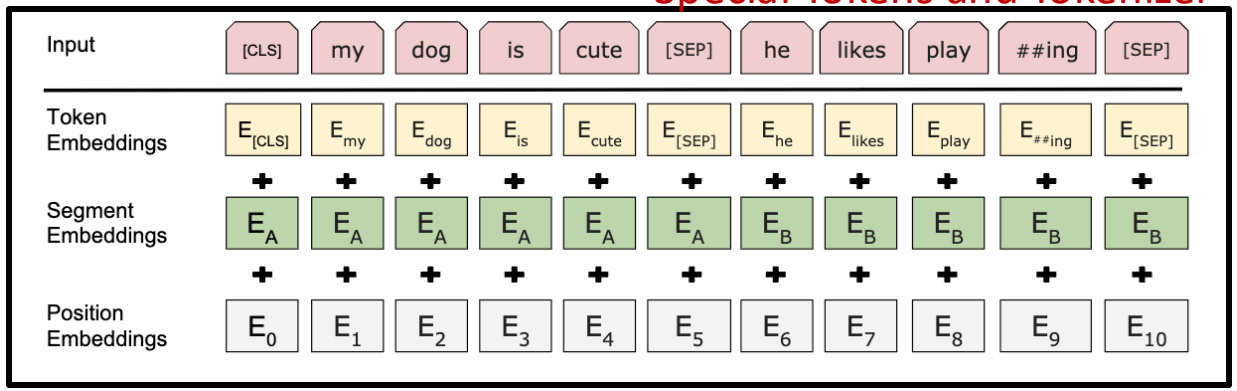
where k is the size of the context window, and the conditional probability P is modeled using a neural network with parameters Θ. These parameters are trained using stochastic gradient descent [51].

Ref: Improving Language Understanding by Generative Pre-Training

Language Modeling – Masked Prediction (BERT, Oct 2018)



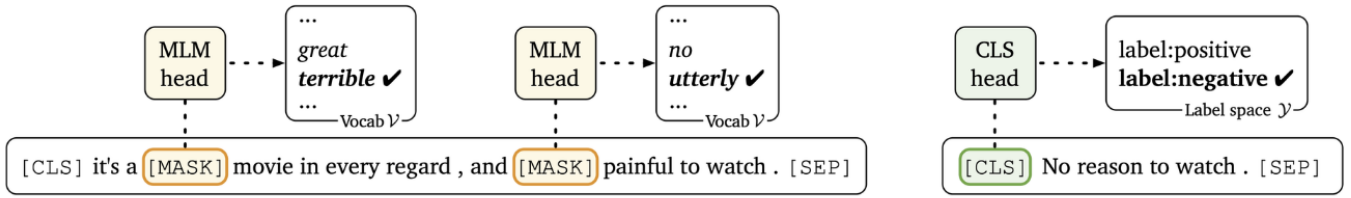
Special Tokens and Tokenizer



- Pretraining corpus: BooksCorpus dataset including 7,000 unique unpublished books
- Training strategy: 15% mask rate
  - If the i-th token is chosen, we replace the i-th token with (1) the [MASK] token 80% of the time (2) a random token 10% of the time (3) the unchanged i-th token 10% of the time.
- Model size:
  - BERT BASE (L=12, H=768, A=12, Total Parameters=110M)
  - BERT LARGE (L=24, H=1024, A=16, Total Parameters=340M)
  - Context length: 512 tokens

Ref: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (NAACL 2019), RoBERTa: A Robustly Optimized BERT Pretraining Approach (Submitted to ICLR 2020)

Language Modeling – Masked Prediction (BERT, Oct 2018)



(a) MLM pre-training (b) Fine-tuning

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Ref: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (NAACL 2019)

Transformer Architecture Evolution

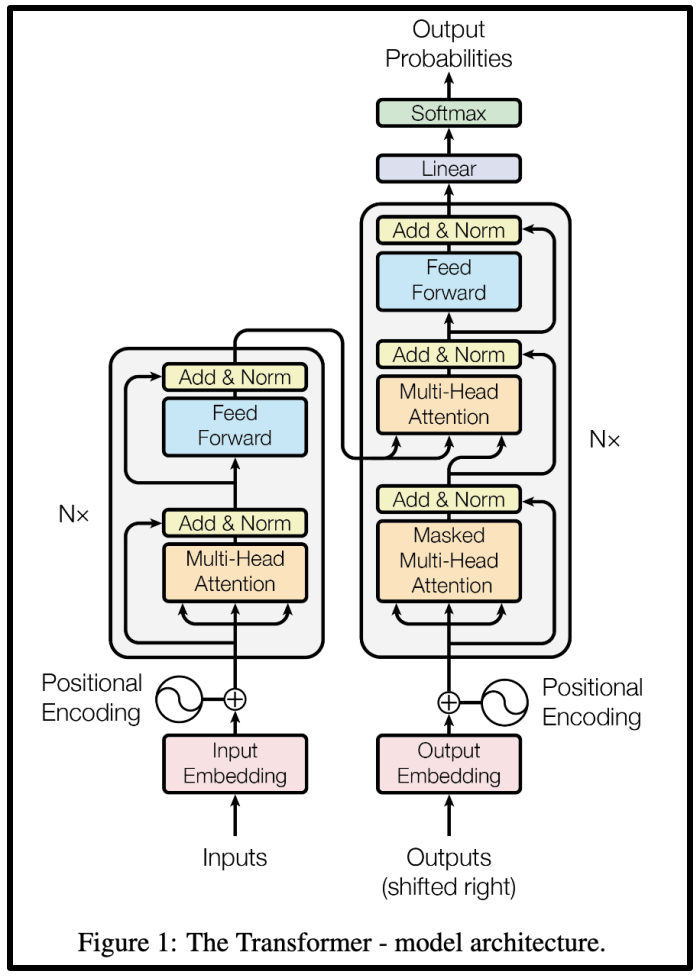
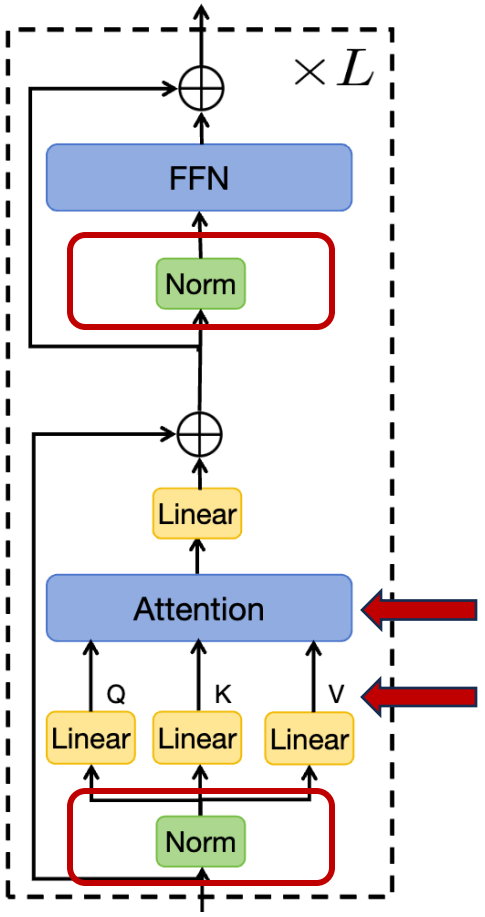


Figure 1: The Transformer - model architecture.

Transformer++

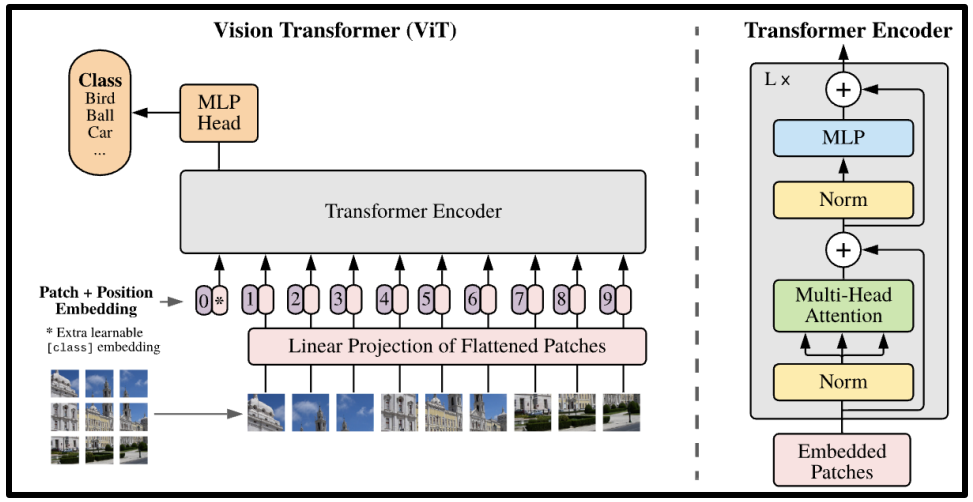


Flash Attention

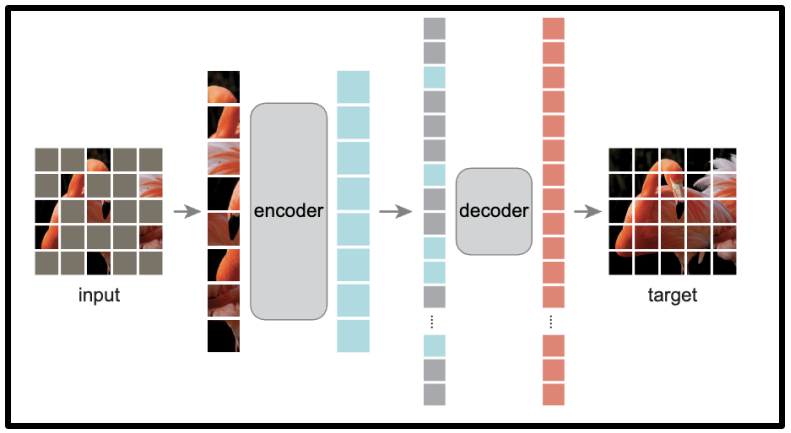
Rotary Position Encoding

Language models vs. vision models

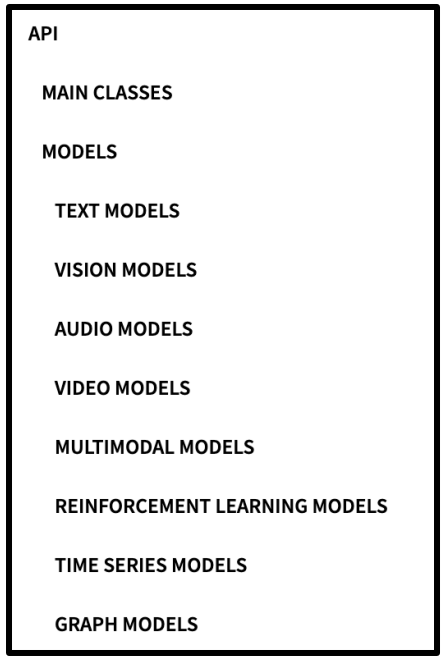
ViT



MAE



DINO…



- Different objectives + Encoding strategies…
- Hugging Face `transformers` is all you need

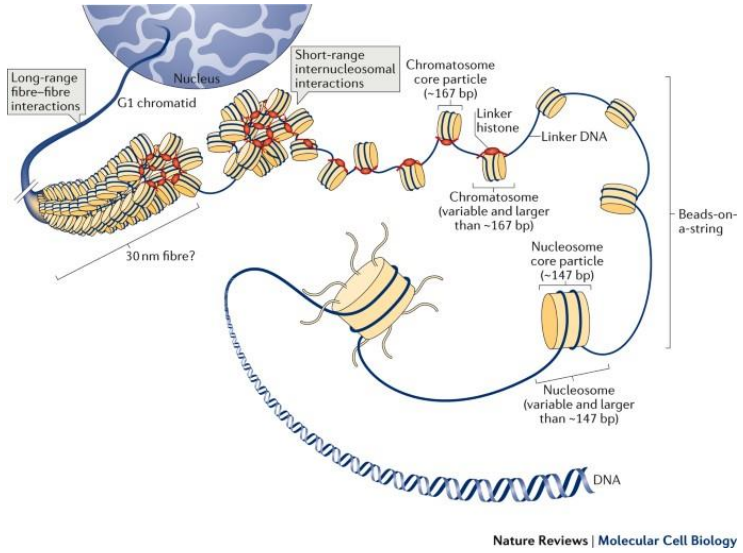
```python
from transformers import AutoModelForSequenceClassification, AutoTokenizer
from datasets import load_dataset

model = AutoModelForSequenceClassification.from_pretrained("distilbert/distilbert-base-uncased")
tokenizer = AutoTokenizer.from_pretrained("distilbert/distilbert-base-uncased")
dataset = load_dataset("rotten_tomatoes")
```

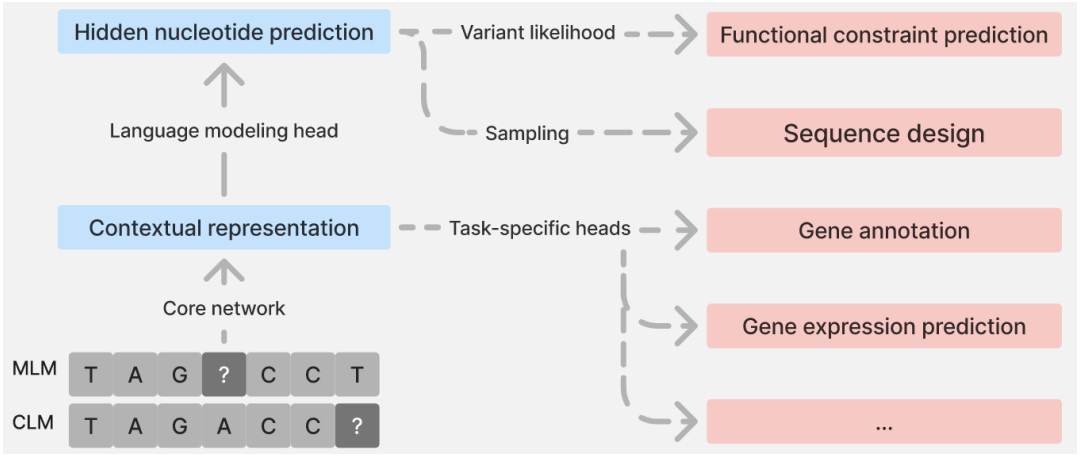

Ref: AN IMAGE IS WORTH 16X16 WORDS:
TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE (ICLR 2021), Masked Autoencoders Are Scalable Vision Learners (CVPR 2022)

Genomic Language Models

Genomic Language Models (gLMs)

- Leverage NLP techniques (LLMs) to understand biological sequences, e.g., DNA, RNA
  - Different challenges from LLMs
    - Long context and hierarchical interactions: short, long
    - Non-functional regions: complex non-functional regions that overshadow the amount of functional elements
    - Limited training corpus: whole-genome sequences
  - Resulting in different model architectures other than Transformer
    - Reducing quadratic complexity of multi-head attentions, E.g., HyenaDNA, Caduceus

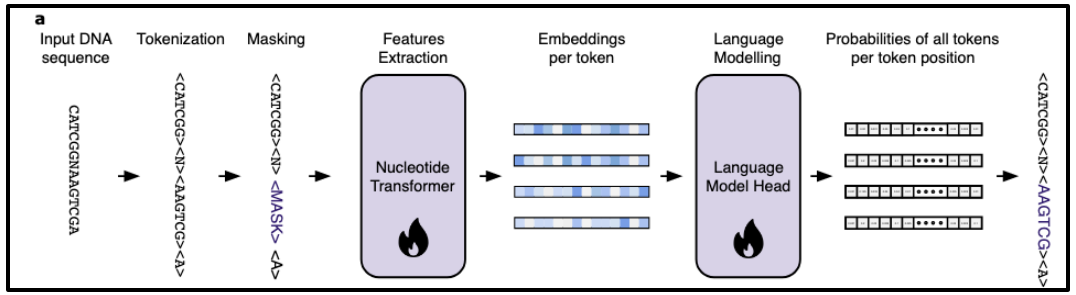




Genomic Language Models (gLMs)

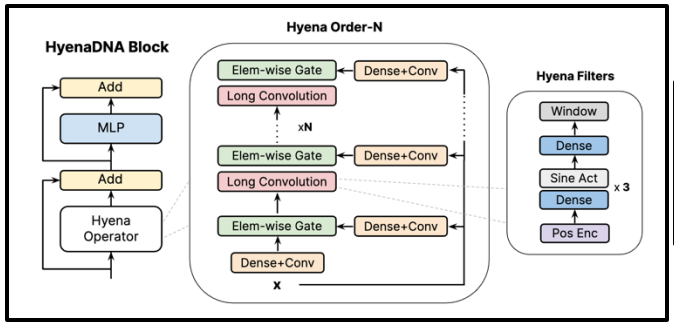
- Transformer-based gLMs vs. Non-transformer-based gLMs

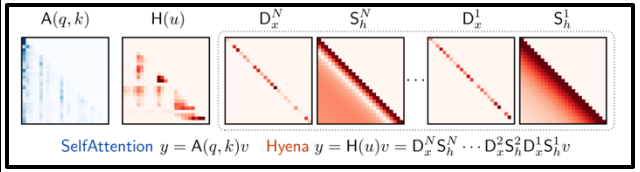
Transformer-based gLMs e.g., Nucleotide Transformer



- Sliding window (k-mers) tokenizer
- Masked token prediction
- Non-single-nucleotide resolution
  - Up to 2500 tokens (15,000 bps)
  - 500M and 2.5B

Non-transformer-based gLMs e.g., HyenaDNA



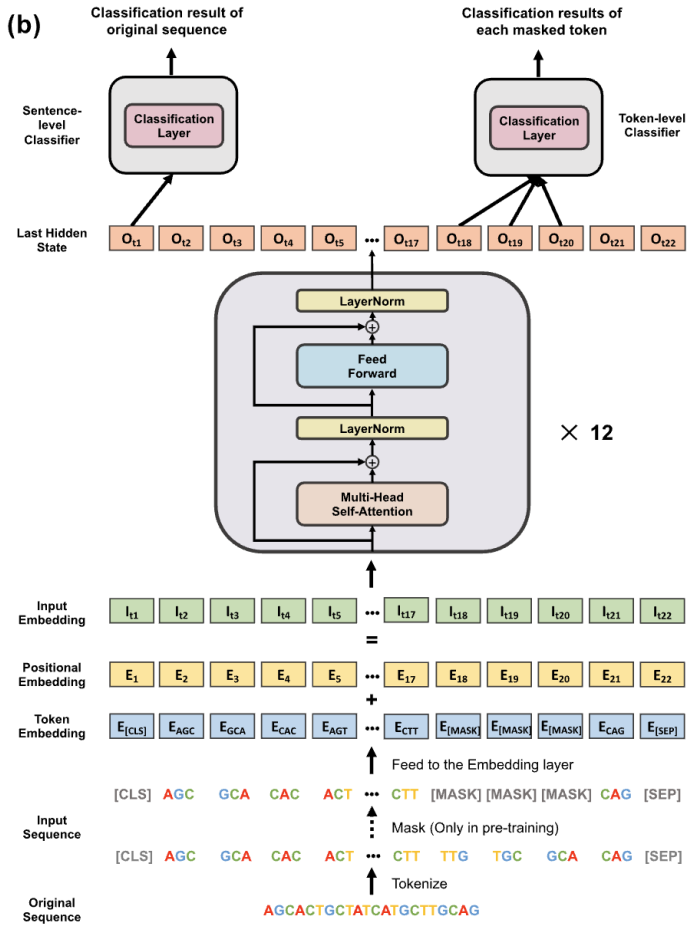


- Single-nucleotide resolution
- Next token prediction
- Up to 1M context length
  - 14M
  - Recently, Evo2 has a 40B size

Dalla-Torre, Hugo, et al. "Nucleotide Transformer: building and evaluating robust foundation models for human genomics." NatureMethods (2024): 1-11.
Nguyen, Eric, et al. "Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution." Advances in neural information processing systems 36 (2023): 43177-43201.

DNABERT V1

- Tokenization: k-mers with overlapping
- Pre-trained dataset: human reference genome, length of the sequence between 5 and 510
- Downstream tasks:
  - identifying proximal promoter regions
  - identifying transcription factor binding sites
  - recognizing canonical and non-canonical splice sites



DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome (Bioinformatics 2021)

DNABERT V1 - Identifying proximal promoter regions

- Binary classification on whether a sequence is a promoter
  - TATA Promoters
  - Non-TATA Promoters
  - Eukaryotic Promoter Database (EPDnew)
- Baselines:
  - DeePromoter
  - CNN
  - CNN + LSTM, CNN + GRU

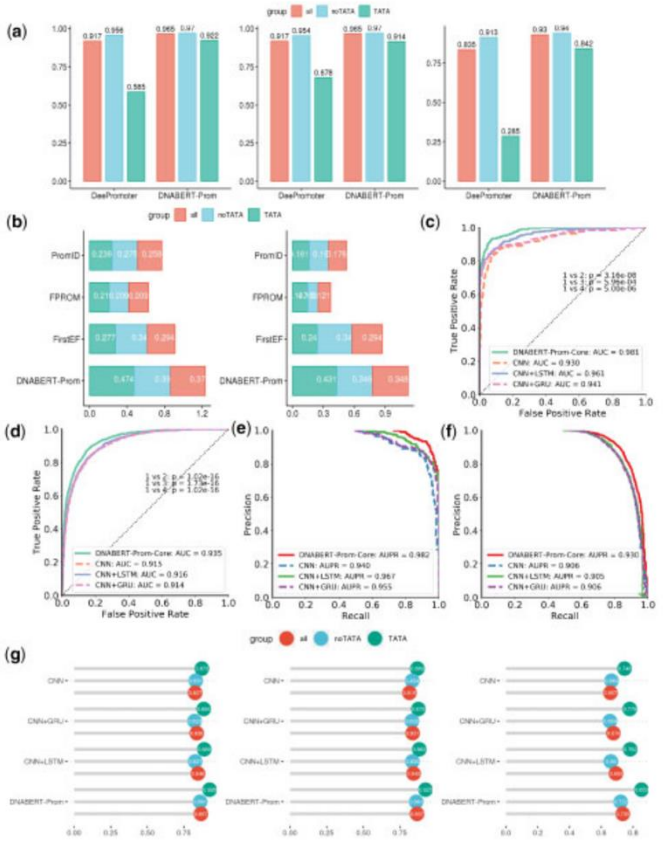
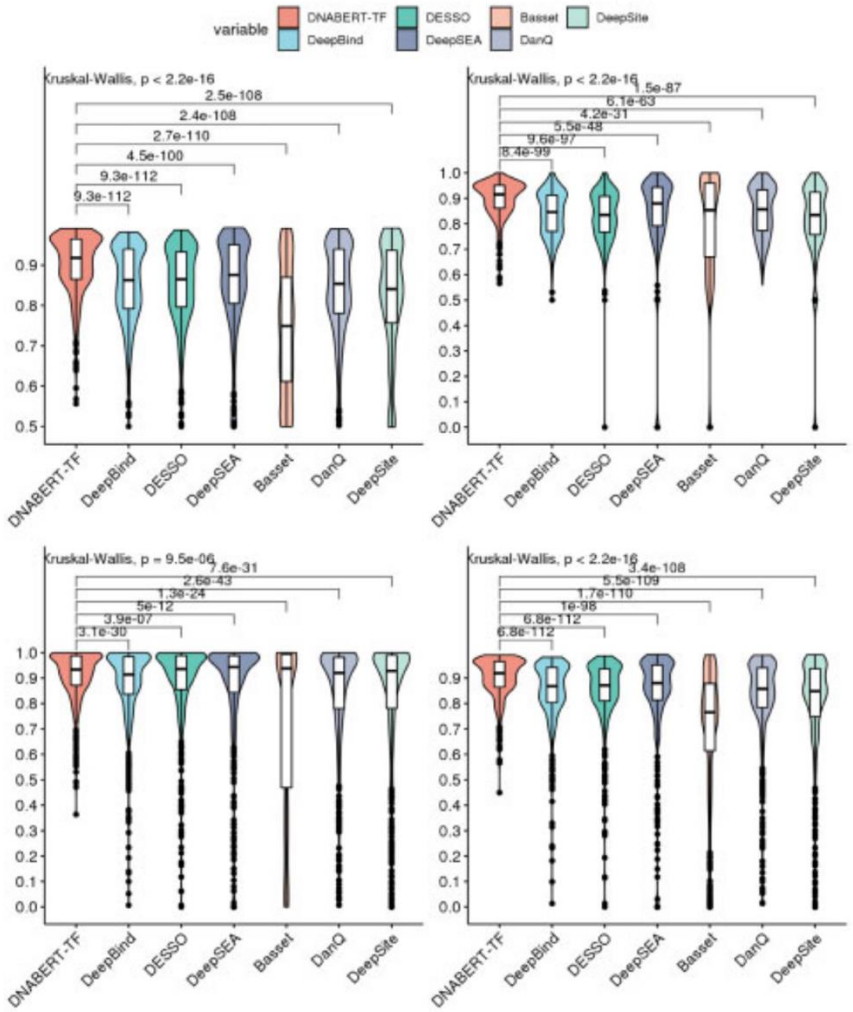


Fig. 2. DNABERT significantly outperforms other models in identifying promoter regions. (**a**) (Left to right) accuracy, F1 and MCC of prom-300 prediction in TATA, no-TATA and combined datasets. (**b**) Stacked barplot showing F1 (left) and MCC (right) of Prom-scan predictions in different settings. (**c–f**) ROC (c, TATA; d, noTATA) and Preci- sion-recall (PR) curves (e, TATA; f, noTATA) with adjusted *P*-values from Delong test. (g) (Left to right) accuracy, F1 and MCC of core promoters prediction in TATA, no-TATA and combined datasets

DNABERT V1 - Identifying transcription factor binding sites

- Binary classification on whether a sequence is a binding site
  - 690 TF ChIP-seq uniform peak profiles from ENCODE database
- Baselines: expert models



DNABERT V1 - Recognizing canonical and non-canonical splice sites

- Binary classification on whether a sequence is a binding site
  - 690 TF ChIP-seq uniform peak profiles from ENCODE database
- Baselines: expert models

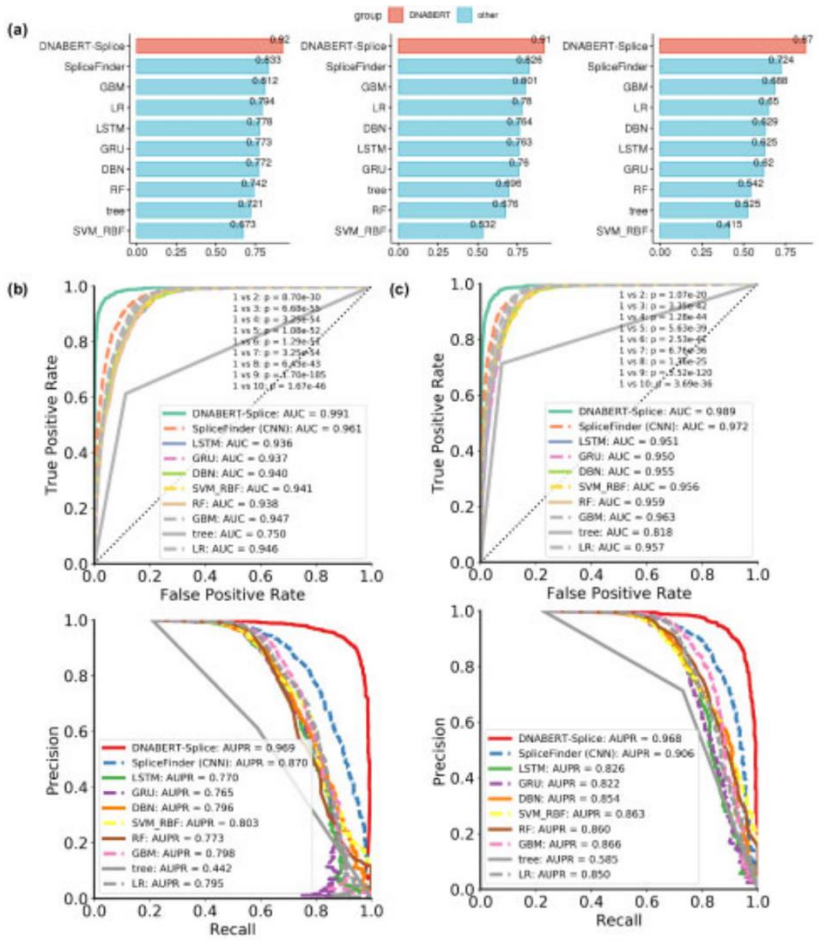


Fig. 5. DNABERT significantly outperforms other models in finding splice sites.(**a**) (Left to right) multiclass accuracy, F1 and MCC of splice donor and acceptor prediction. GBM: gradient boosting; LR: logistic regression; DBN: deep belief network; RF: random forest; tree: decision tree; SVM_RBF: support vector machine with radial basis function kernel. (**b, c**) ROC (top) and PR curves (bottom) on splice donor (b) and acceptor (c) datasets with adjusted *P*-values from Delong test

DNABERT V2 - Recognizing canonical and non-canonical splice sites

- BPE tokenization
- Pre-trained dataset:
  - The human genome dataset is the one used in DNABERT (Ji et al., 2021), which comprises 2.75B nucleotide bases.
  - The multi-species genome dataset encompasses genomes from 135 species, spread across 6 categories. In total, this dataset includes 32.49B nucleotide bases
  - Nearly 12 times the volume of the human genome dataset

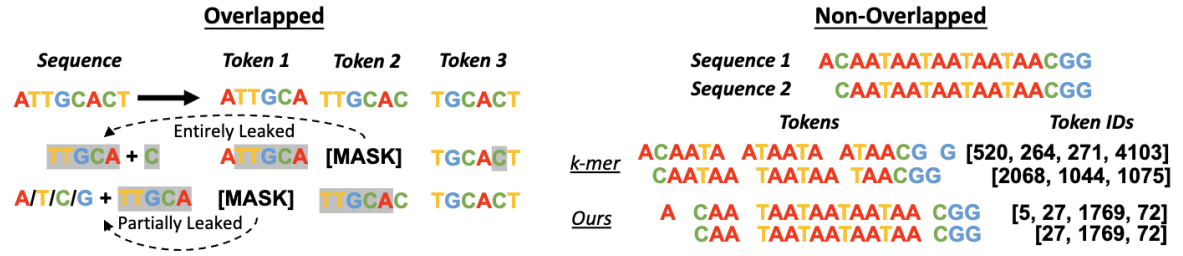


Figure 1: Illustration of the drawbacks of k-mer tokenization. In the overlapping setting, information about a masked token is leaked by its adjacent tokens, while in the non-overlapping setting, adding/deleting one nucleotide base leads to a dramatic change in the tokenized sequence.

DNABERT V2 - Recognizing canonical and non-canonical splice sites

- Evaluation Benchmarks: GENOME UNDERSTANDING EVALUATION (GUE and GUE+)

Species	Task	Num. Datasets	Num. Classes	Sequence Length
Human	Core Promoter Detection	3	2	70
	Transcription Factor Prediction	5	2	100
	Promoter Detection	3	2	300
	Splice Site Detection	1	3	400
Mouse	Transcription Factor Prediction	5	2	100
Yeast	Epigenetic Marks Prediction	10	2	500
Virus	Covid Variant Classification	1	9	1000

Table 1: Summarization of the Genome Understanding Evaluation (GUE) benchmark.

Species	Task	Num. Datasets	Num. Classes	Sequence Length
Human	Enhancer Promoter Interaction	6	2	5000
Fungi	Species Classification	1	25	5000
Virus	Species Classification	1	20	10000

Table 2: Summarization of the Genome Understanding Evaluation Plus (GUE^{+}) benchmark.

DNABERT V2 - Recognizing canonical and non-canonical splice sites

- Model comparisons

Model	Params. ↓	FLOPs ↓	Trn. Tokens	Num. Top-2 ↑	Ave. Scores ↑
DNABERT (3-mer)	86M	3.27	122B	2 || 0	61.62
DNABERT (4-mer)	86M	3.26	122B	0 || 1	61.14
DNABERT (5-mer)	87M	3.26	122B	0 || 1	60.05
DNABERT (6-mer)	89M	3.25	122B	0 || 1	60.51
NT-500M-human	480M	3.19	50B	0 || 0	55.43
NT-500M-1000g	480M	3.19	50B	0 || 1	58.23
NT-2500M-1000g	2537M	19.44	300B	0 || 1	61.41
NT-2500M-multi	2537M	19.44	300B	7 || 9	66.93
DNABERT-2	117M	1.00	262B	8 || 4	66.80
DNABERT-2◆	117M	1.00	263B	**11** || **10**	**67.77**

Table 3: The statistics and performance of each model. The five columns represent the number of model parameters, relative FLOPs compared to DNABERT-2, the number of tokens used in pre-training, and the number of being top-2 among all the models (`1st || 2nd`) and the average evaluation scores on the 28 datasets of the GUE benchmark. ◆: perform further masked language modeling pre-training on the training sets of the GUE benchmark.

DNABERT V2 - Recognizing canonical and non-canonical splice sites

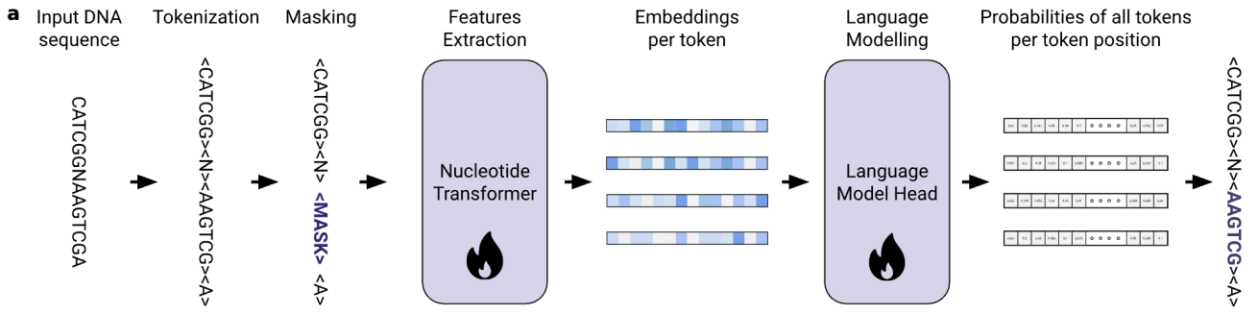
- Model comparisons

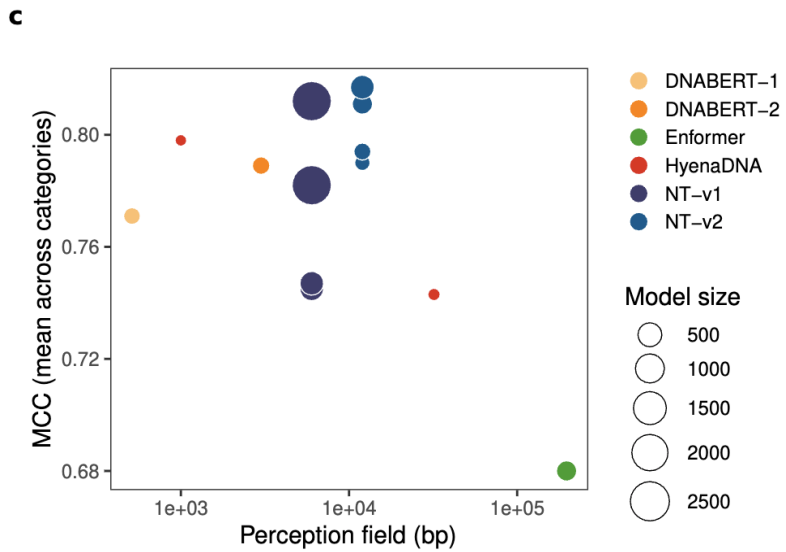
Species	Yeast	Mouse	Virus	Human			
Task	**EMP**	**TF-M**	**CVC**	**TF-H**	**PD**	**CPD**	**SSP**
DNABERT (3-mer)	49.54	57.73	62.23	64.43	84.63	**72.96**	84.14
DNABERT (4-mer)	48.59	59.58	59.87	64.41	82.99	71.10	84.05
DNABERT (5-mer)	48.62	54.85	63.64	50.46	84.04	72.03	84.02
DNABERT (6-mer)	49.10	56.43	55.50	64.17	81.70	71.81	84.07
NT-500M-human	45.35	45.24	57.13	50.82	85.51	66.54	79.71
NT-500M-1000g	47.68	49.31	52.06	58.92	86.58	69.13	80.97
NT-2500M-1000g	50.86	56.82	66.73	61.99	86.61	68.17	85.78
NT-2500M-multi	58.06	67.01	**73.04**	63.32	**88.14**	71.62	**89.36**
DNABERT-2	55.98	67.99	71.02	**70.10**	84.21	70.52	84.99
DNABERT-2◆	**58.83**	**71.21**	68.49	66.84	83.81	71.07	85.93

Table 4: The models' averaged performance on the 7 tasks in the GUE benchmark, including Epigenetic Marks Prediction (EMP), Transcription Factor Prediction on the Human genome and the Mouse genome (TF-H and TF-M), Covid Variants Classification (CVC), Promoter Detection (PD), Core Promoter Detection (CPD), and Splice Site Prediction (SSP).

Nucleotide Transformer

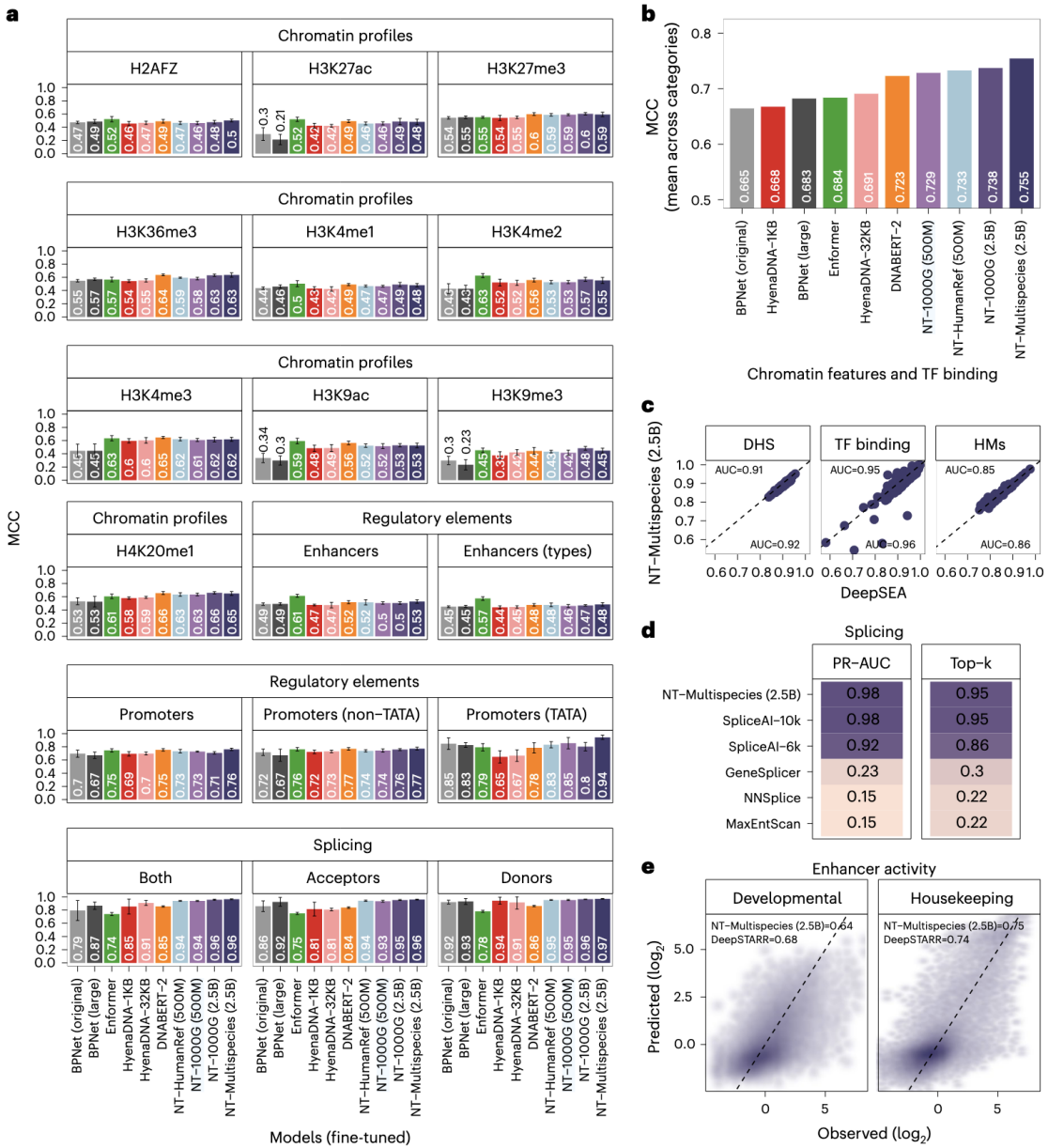
- 6-mers without overlapping
- Pre-trained dataset:
  - The Human reference genome dataset hg38
  - The 1000G dataset: genetic variants arising from different human populations
  - The Multispecies dataset
- All models were trained on the Cambridge-1 Nvidia supercomputer system, using 16 nodes, each equipped with eight A100 GPUs, leading to a total of 128 A100 GPUs used





Nucleotide Transformer

- BPNet:
  - convolutional architecture from scratch on each of the 18 tasks
  - Original, 121,000 parameters
  - Large, 28 million parameters
- There are some tasks pre-trained models can achieve better performance.



The Nucleotide Transformer: Building and Evaluating Robust Foundation Models for Human Genomics (Nature Methods 2024)

GENERator

- 6-mers without overlapping
- Next k-mers prediction
- Scale up to 1.2B

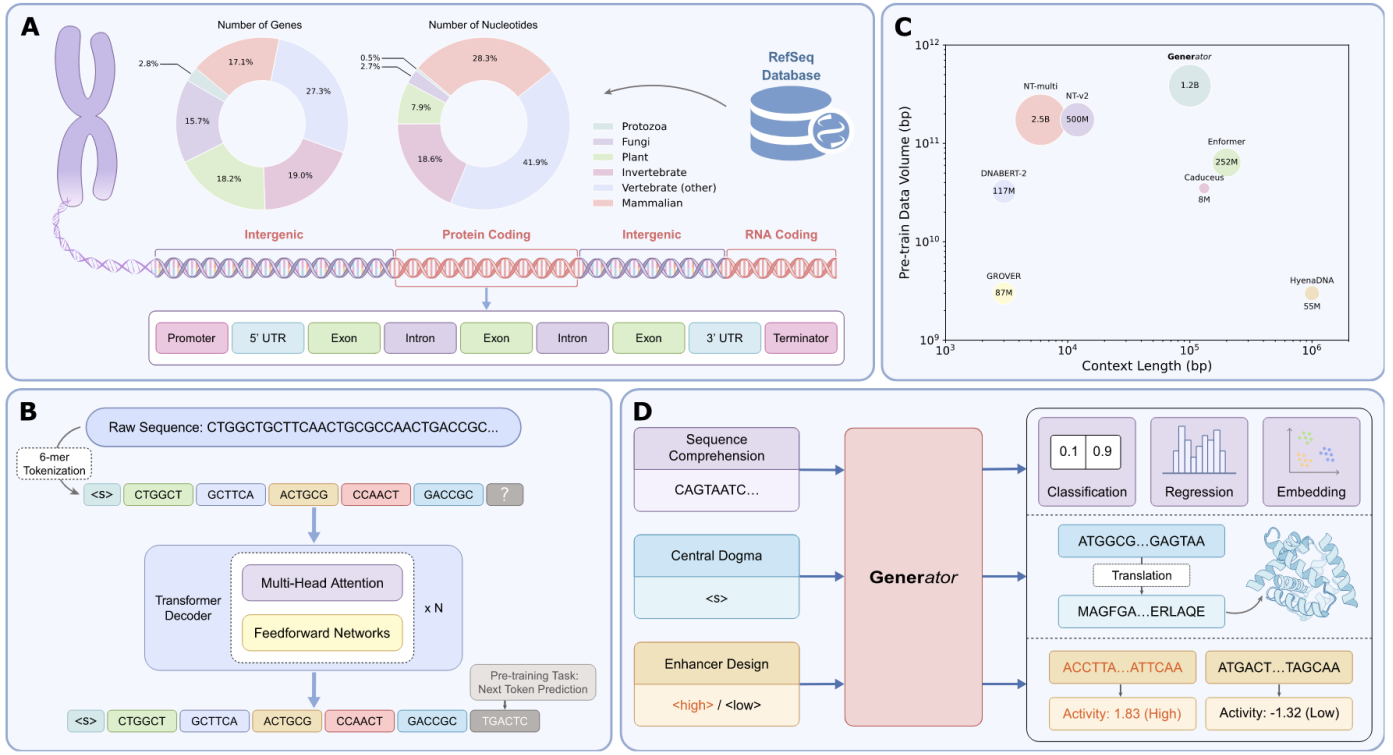


Figure 1: Overview of the **Gener***ator*. (A) The pre-training dataset of the **Gener***ator* encompasses a diverse range of eukaryotic organisms and gene types, totaling 386B nucleotides. (B) The pre-training employs the next token prediction (NTP) task, utilizing a 6-mer tokenizer. (C) Model comparison regarding the context length, pre-train data volume, and model parameters. (D) Downstream applications include sequence comprehension, interpretation of the central dogma by generating DNA sequences that encode functional proteins, and prompt-responsive enhancer design with targeted activity profiles.

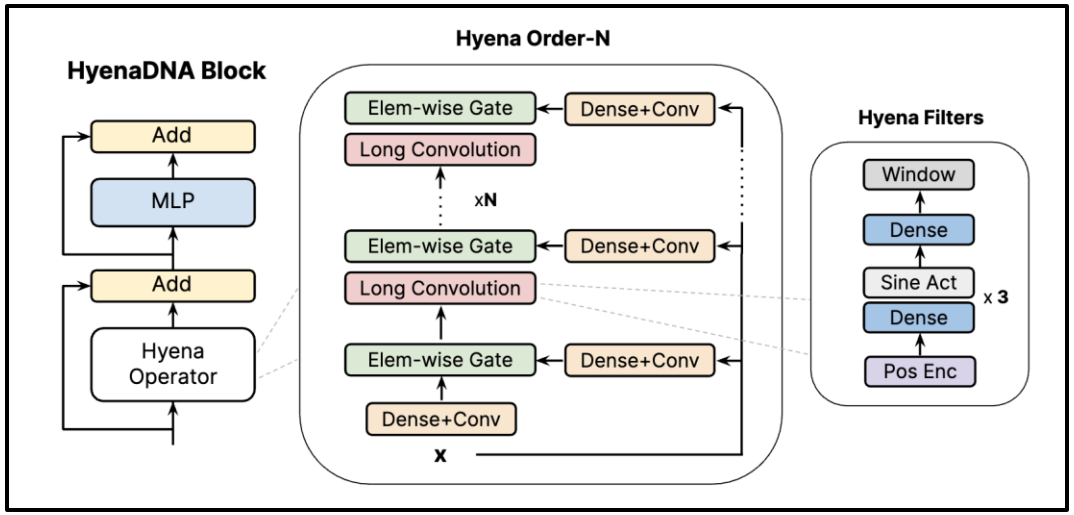
GENERator

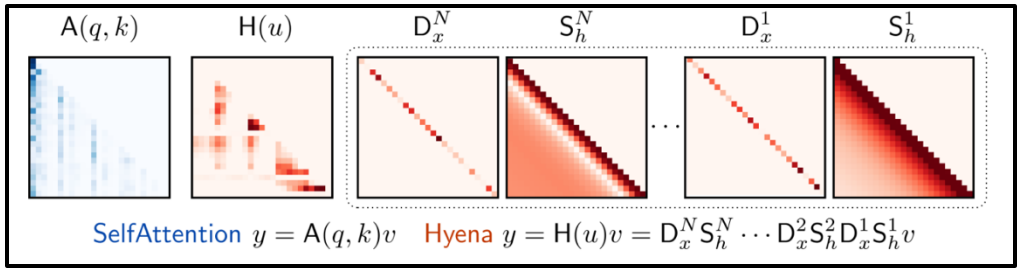
- Performance is very close to each other…

Table 2: Evaluation of the revised Nucleotide Transformer tasks. The reported values represent the Matthews correlation coefficient (MCC) averaged over 10-fold cross-validation, with the standard error in parentheses.

	Enformer (252M)	DNABERT-2 (117M)	HyenaDNA (55M)	NT-multi (2.5B)	NT-v2 (500M)	Caduceus-Ph (8M)	Caduceus-PS (8M)	GROVER (87M)	**Gener*ator*** (1.2B)	**Gener*ator*-All** (1.2B)
H2AFZ	0.522 (0.019)	0.490 (0.013)	0.455 (0.015)	0.503 (0.010)	0.524 (0.008)	0.417 (0.016)	0.501 (0.013)	0.509 (0.013)	**0.529 (0.009)**	0.506 (0.019)
H3K27ac	0.520 (0.015)	0.491 (0.010)	0.423 (0.017)	0.481 (0.020)	0.488 (0.013)	0.464 (0.018)	0.464 (0.022)	0.489 (0.023)	**0.546 (0.015)**	0.496 (0.014)
H3K27me3	0.552 (0.007)	0.599 (0.010)	0.541 (0.018)	0.593 (0.016)	0.610 (0.006)	0.547 (0.010)	0.561 (0.036)	0.600 (0.008)	**0.619 (0.008)**	0.590 (0.014)
H3K36me3	0.567 (0.017)	0.637 (0.007)	0.543 (0.010)	0.635 (0.016)	0.633 (0.015)	0.543 (0.009)	0.602 (0.008)	0.585 (0.008)	**0.650 (0.006)**	0.621 (0.013)
H3K4me1	**0.504 (0.021)**	0.490 (0.008)	0.430 (0.014)	0.481 (0.012)	0.490 (0.017)	0.411 (0.012)	0.434 (0.030)	0.468 (0.011)	**0.504 (0.010)**	0.490 (0.016)
H3K4me2	**0.626 (0.015)**	0.558 (0.013)	0.521 (0.024)	0.552 (0.022)	0.552 (0.013)	0.480 (0.013)	0.526 (0.035)	0.558 (0.012)	0.607 (0.010)	0.569 (0.012)
H3K4me3	0.635 (0.019)	0.646 (0.008)	0.596 (0.015)	0.618 (0.015)	0.627 (0.020)	0.588 (0.020)	0.611 (0.015)	0.634 (0.011)	**0.653 (0.008)**	0.628 (0.018)
H3K9ac	**0.593 (0.020)**	0.564 (0.013)	0.484 (0.022)	0.527 (0.017)	0.551 (0.016)	0.514 (0.014)	0.518 (0.018)	0.531 (0.014)	0.570 (0.017)	0.556 (0.018)
H3K9me3	0.453 (0.016)	0.443 (0.025)	0.375 (0.026)	0.447 (0.018)	0.467 (0.044)	0.435 (0.019)	0.455 (0.019)	0.441 (0.017)	**0.509 (0.013)**	0.480 (0.037)
H4K20me1	0.606 (0.016)	0.655 (0.011)	0.580 (0.009)	0.650 (0.014)	0.654 (0.011)	0.572 (0.012)	0.590 (0.020)	0.634 (0.006)	**0.670 (0.006)**	0.652 (0.010)
Enhancer	**0.614 (0.010)**	0.517 (0.011)	0.475 (0.006)	0.527 (0.012)	0.575 (0.023)	0.480 (0.008)	0.490 (0.009)	0.519 (0.009)	0.594 (0.013)	0.553 (0.020)
Enhancer type	**0.573 (0.013)**	0.476 (0.009)	0.441 (0.010)	0.484 (0.012)	0.541 (0.013)	0.461 (0.009)	0.459 (0.011)	0.481 (0.009)	0.547 (0.017)	0.510 (0.022)
Promoter all	0.745 (0.012)	0.754 (0.009)	0.693 (0.016)	0.761 (0.009)	0.780 (0.012)	0.707 (0.017)	0.722 (0.014)	0.721 (0.011)	**0.795 (0.005)**	0.765 (0.009)
Promoter non-TATA	0.763 (0.012)	0.769 (0.009)	0.723 (0.013)	0.773 (0.010)	0.785 (0.009)	0.740 (0.012)	0.746 (0.009)	0.739 (0.018)	**0.801 (0.005)**	0.786 (0.007)
Promoter TATA	0.793 (0.026)	0.784 (0.036)	0.648 (0.044)	0.944 (0.016)	0.919 (0.028)	0.868 (0.023)	0.853 (0.034)	0.891 (0.041)	**0.950 (0.009)**	0.862 (0.024)
Splice acceptor	0.749 (0.007)	0.837 (0.006)	0.815 (0.049)	0.958 (0.003)	**0.965 (0.004)**	0.906 (0.015)	0.939 (0.012)	0.812 (0.012)	0.964 (0.003)	0.951 (0.006)
Splice site all	0.739 (0.011)	0.855 (0.005)	0.854 (0.053)	0.964 (0.003)	**0.968 (0.003)**	0.941 (0.006)	0.942 (0.012)	0.849 (0.015)	0.966 (0.003)	0.959 (0.003)
Splice donor	0.780 (0.007)	0.861 (0.004)	0.943 (0.024)	0.970 (0.002)	0.976 (0.003)	0.944 (0.026)	0.964 (0.010)	0.842 (0.009)	**0.977 (0.002)**	0.971 (0.002)

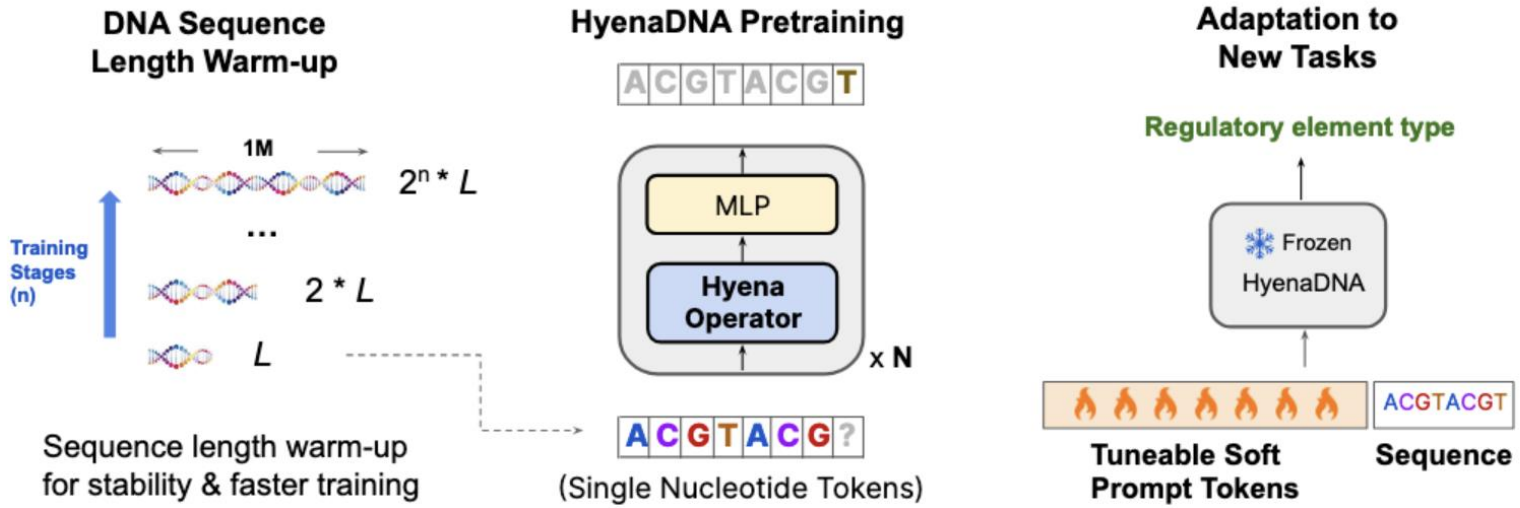
Hyena Operator





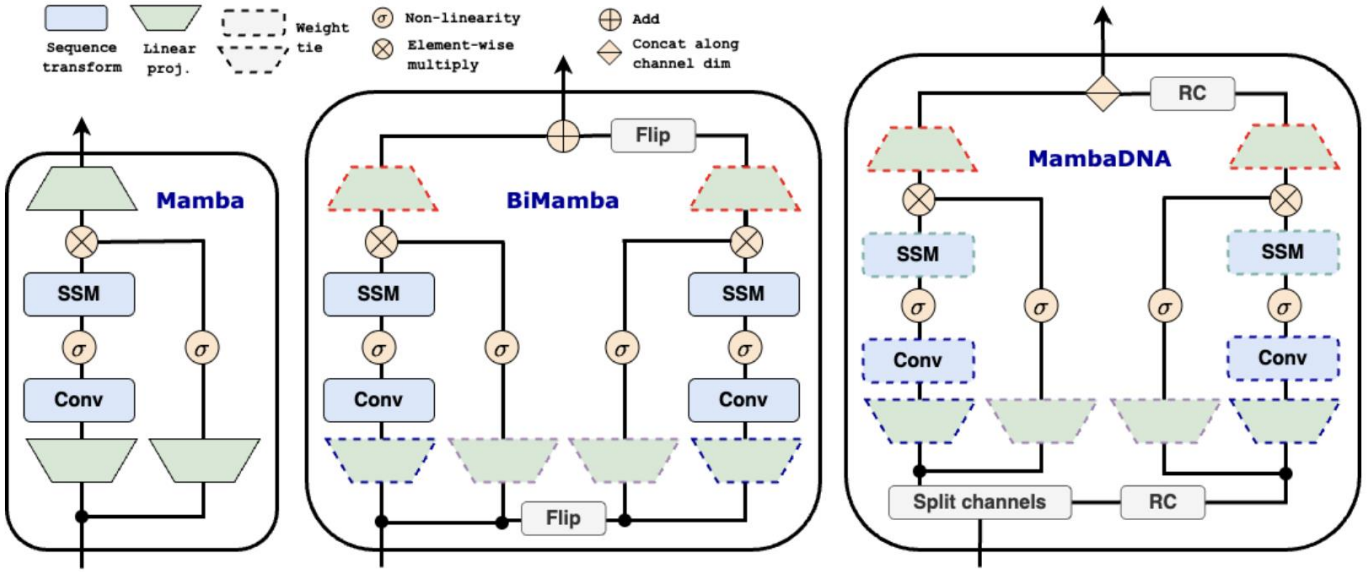
- Neural network for kernel materializations
- Convolutional neural network is implemented in matrix multiplication under hood.
- Still convolutional neural network but implemented using FFT

HyenaDNA



- Single-nucleotide resolution
- Next token prediction
  - human reference genome
- Up to 1M context length
  - 14M
  - Recently, Evo2 has a 40B size

Caduceus



- Single-nucleotide resolution
- masked language modeling
  - human reference genome
- Up to 131k context length

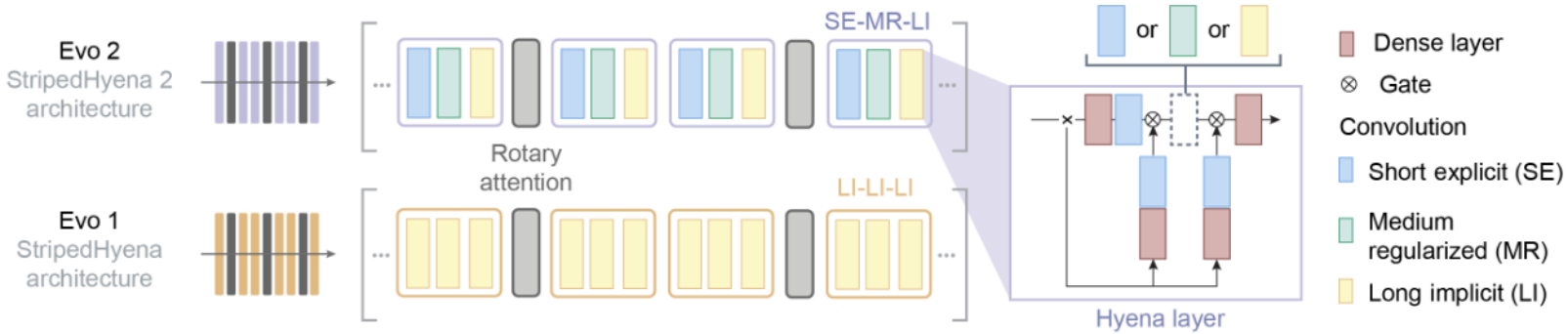
Caduceus

Table 1. Genomic Benchmarks. Top-1 accuracy (↑) across 5-fold cross-validation (CV) for pretrained HyenaDNA, Mamba NTP, Caduceus models, and a supervised CNN baseline (trained from scratch). Best values per task are **bolded**, second best are *italicized*. Error bars indicate the difference between the maximum and minimum values across 5 random seeds used for CV.

	CNN (264K)	HyenaDNA (436K)	Mamba (468K)	Caduceus w/o Equiv. (470K)	Caduceus-Ph (470K)	Caduceus-PS (470K)
Mouse Enhancers	0.715 ±0.087	*0.780* ±0.025	0.743 ±0.054	0.770 ±0.058	0.754 ±0.074	**0.793** ±0.058
Coding vs. Intergenomic	0.892 ±0.008	0.904 ±0.005	0.904 ±0.004	0.908 ±0.003	**0.915** ±0.003	*0.910* ±0.003
Human vs. Worm	0.942 ±0.002	0.964 ±0.002	0.967 ±0.002	*0.970* ±0.003	**0.973** ±0.001	0.968 ±0.002
Human Enhancers Cohn	0.702 ±0.021	0.729 ±0.014	0.732 ±0.029	0.741 ±0.008	**0.747** ±0.004	*0.745* ±0.007
Human Enhancer Ensembl	0.744 ±0.122	0.849 ±0.006	0.862 ±0.008	0.883 ±0.002	*0.893* ±0.008	**0.900** ±0.006
Human Regulatory	0.872 ±0.005	0.869 ±0.012	0.814 ±0.211	0.871 ±0.007	*0.872* ±0.011	**0.873** ±0.007
Human OCR Ensembl	0.698 ±0.013	0.783 ±0.007	0.815 ±0.002	0.818 ±0.003	**0.828** ±0.006	*0.818* ±0.006
Human NonTATA Promoters	0.861 ±0.009	0.944 ±0.002	0.933 ±0.007	0.933 ±0.006	**0.946** ±0.007	*0.945* ±0.010

- Single-nucleotide resolution
- masked language modeling
  - human reference genome
- Up to 131k context length

Evo series

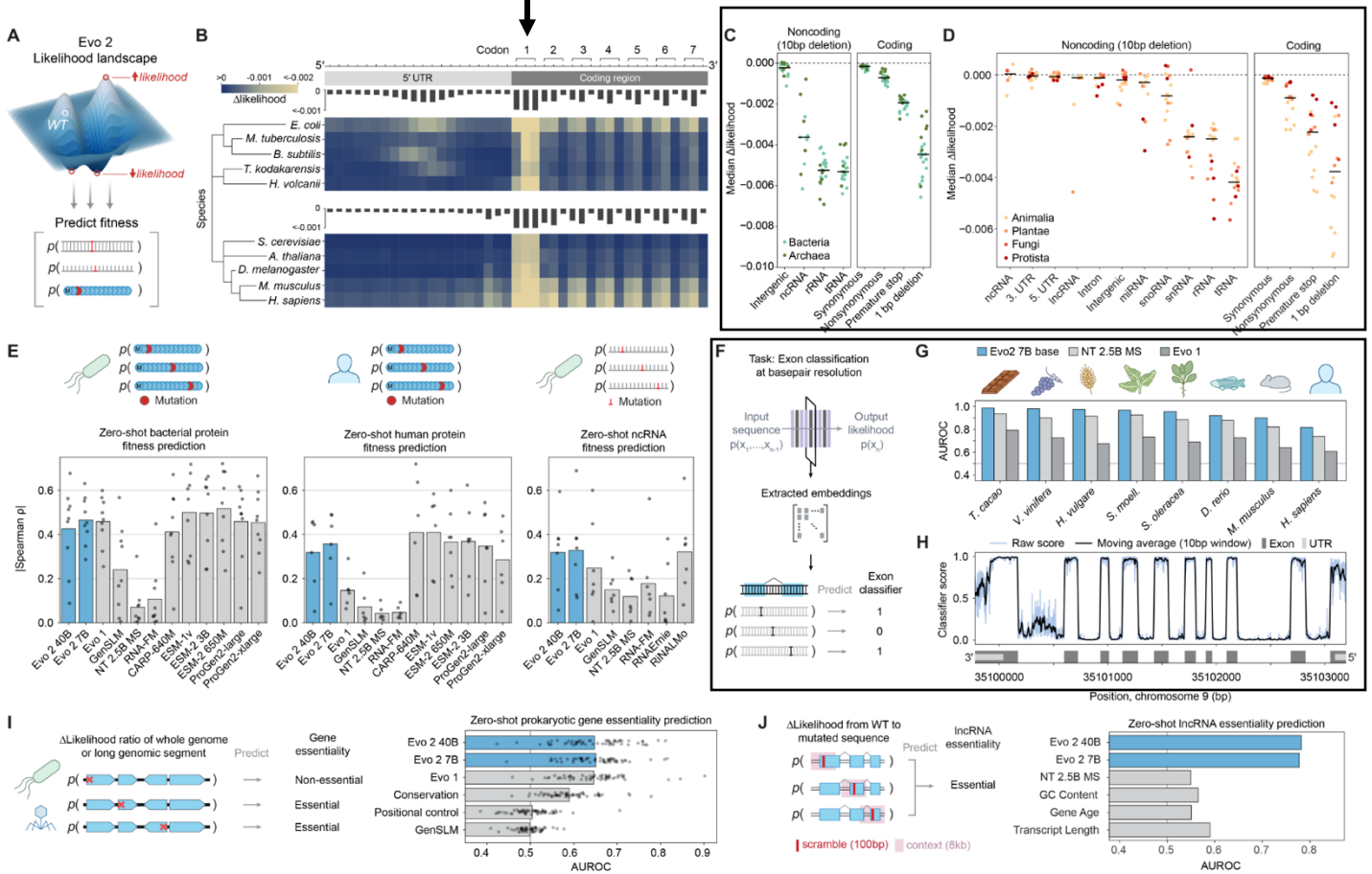


- Single-nucleotide resolution
- Next token prediction
  - A pretraining phase at 8192 token context focused more on functional elements and midtraining phase during which we extend up to 1M token context length with more entire genomes in the data mix.
- Up to 1M context length
  - 7B and 40B
- Pre-trained corpus (8.84 trillion tokens):
  - Opengenome2
  - prokaryotic genomes, Eukaryotic reference genomes, Metagenomes and RNA

Evo series

Very unlikely

regions under stronger evolutionary constraint are also more sensitive to mutational likelihoods

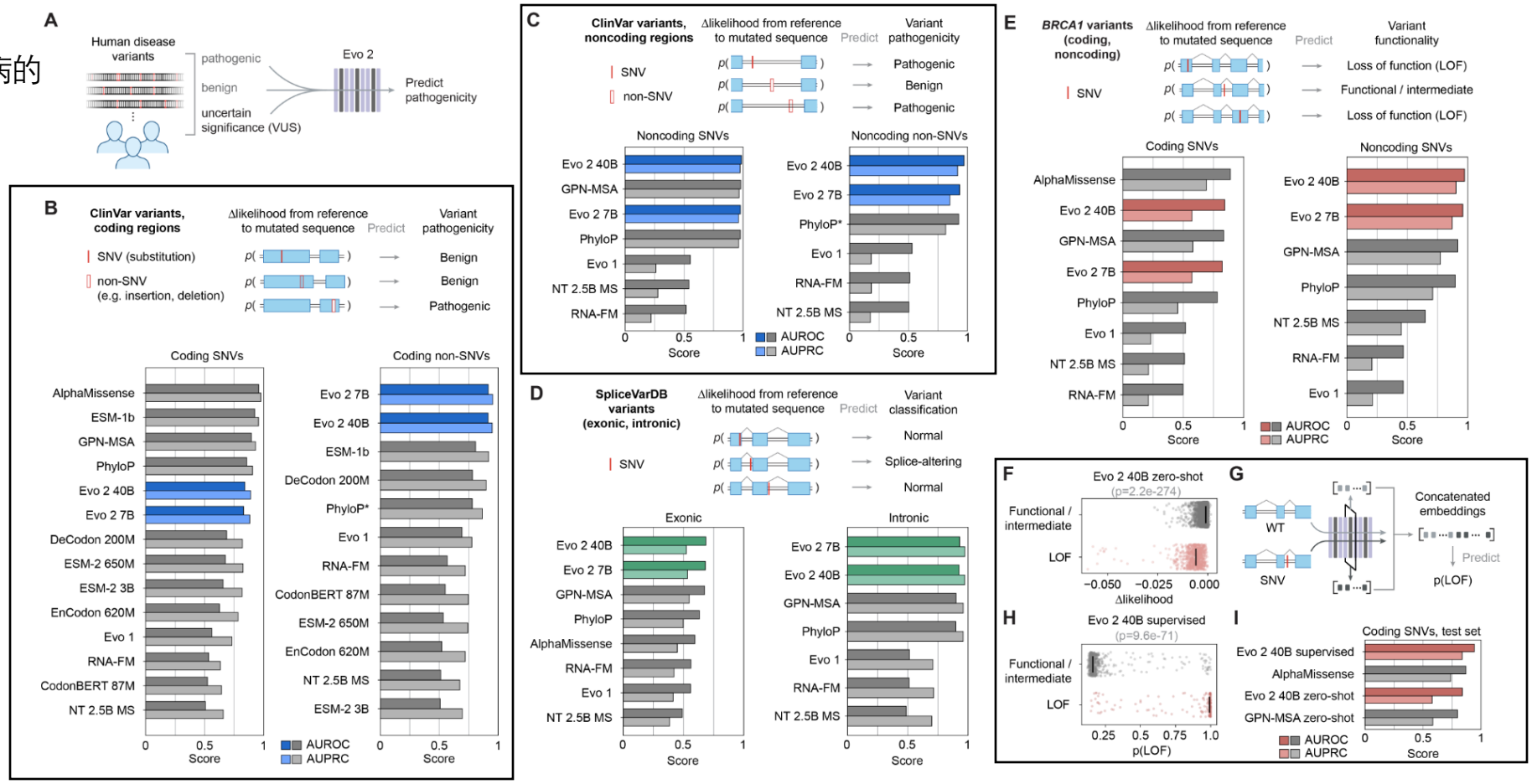


- Align well with deep mutational scanning (DMS) results in experiments →
- Essentiality

Exon classification

Evo series

- Pathogenic: 致病的
- Benign: 良性



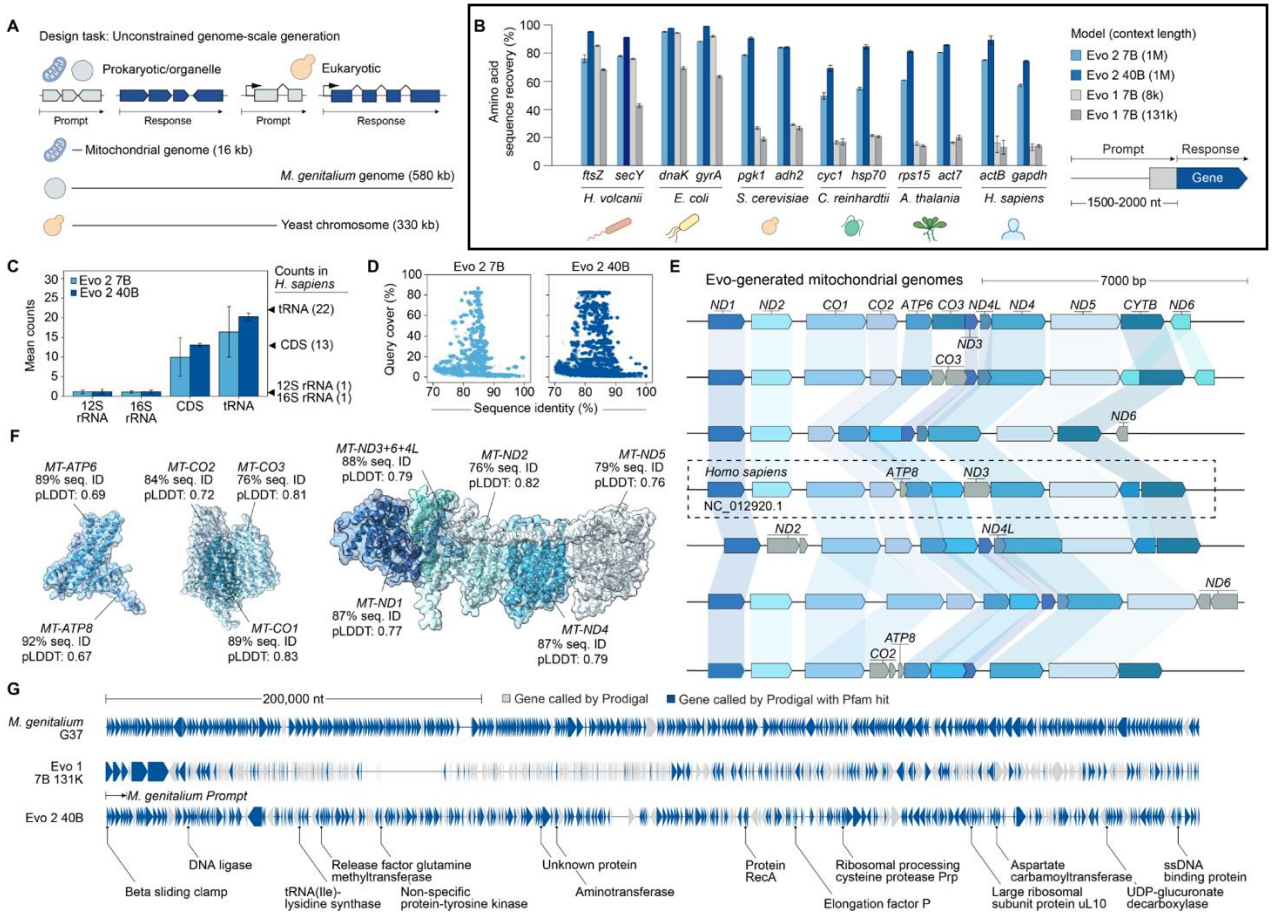
loss-of-function

Evo series

generate chromosome- and genome-scale sequences using unconstrained autoregressive generation (generate DNA sequences with similar lengths as those of the native sequence)

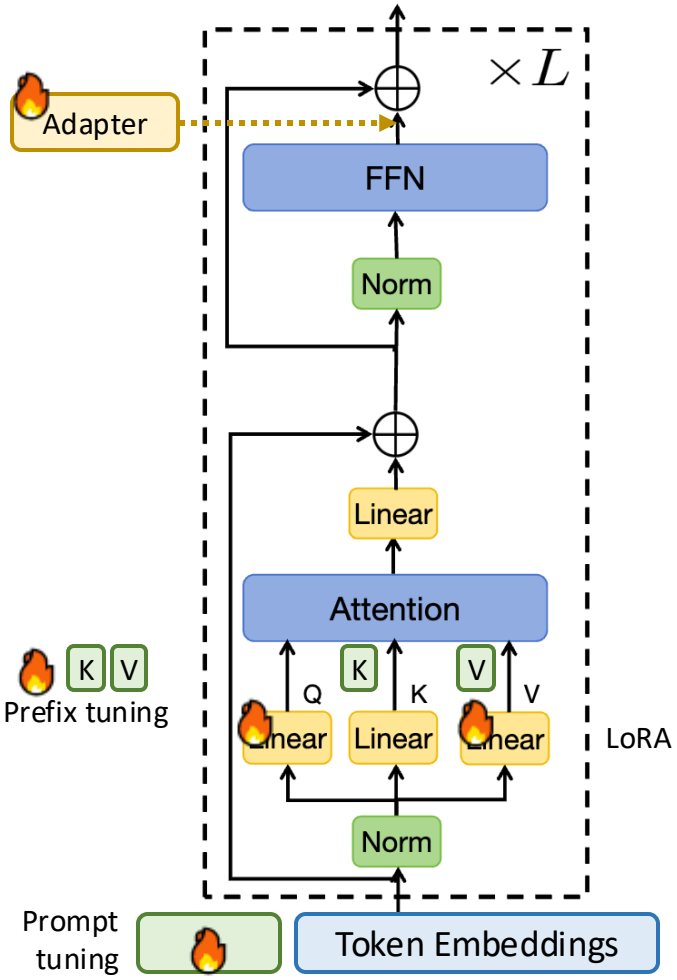
Predicted rRNA, CDS, and tRNA counts in Evo 2

the sequence recovery of the Evo 2 generated gene against the natural gene



Fine-tuning Techniques: Parameter Efficient Fine-Tuning (PEFT)

Taxonomy of PEFT for Large Models



Additive Fine-tuning

- Features
  - injecting new trainable modules or parameters
- Representative methods
  - Adapter-based, Soft Prompt-based e.g., Prefix-tuning, p-tuning, prompt-tuning

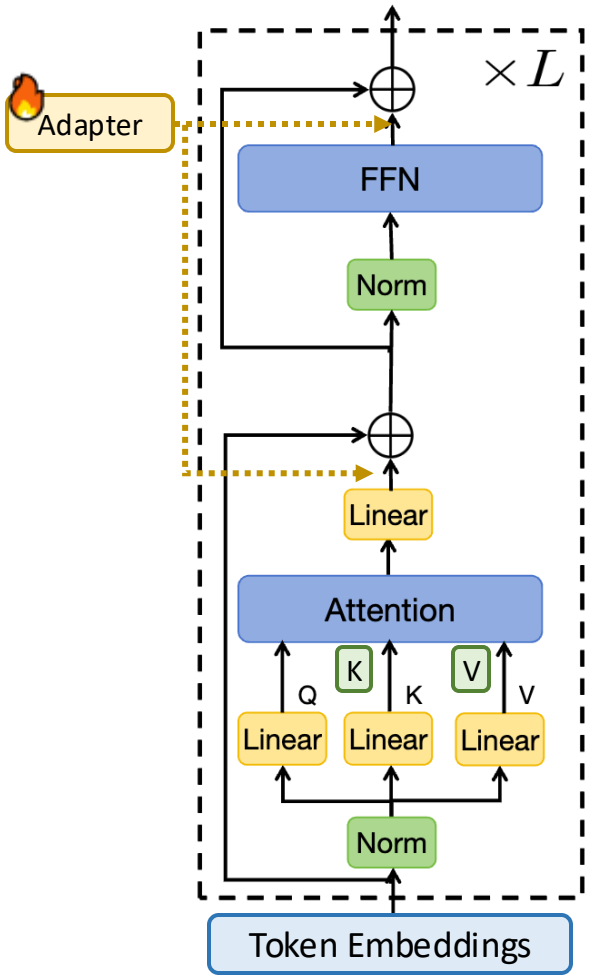
Selective Fine-tuning

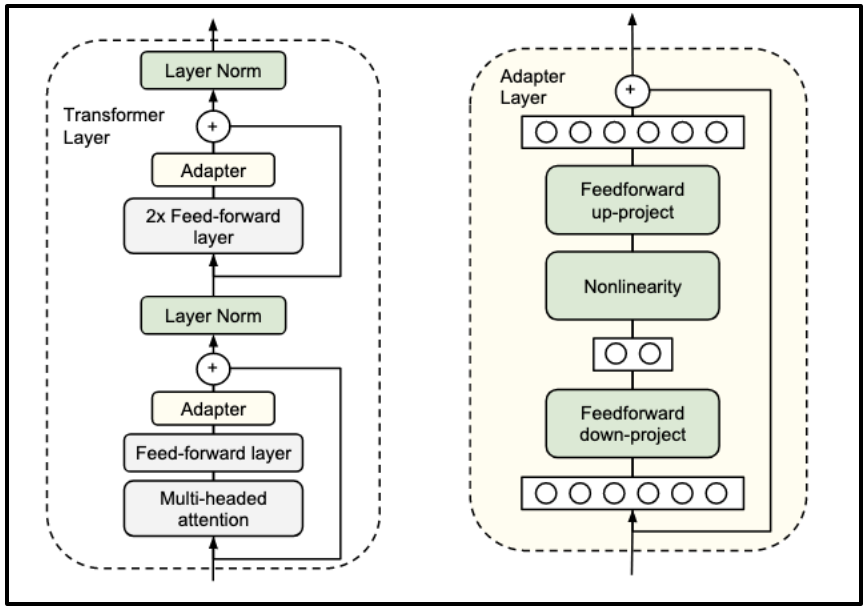
- Features
  - make a subset of model parameters trainable during fine-tuning
- Representative method
  - Diff pruning

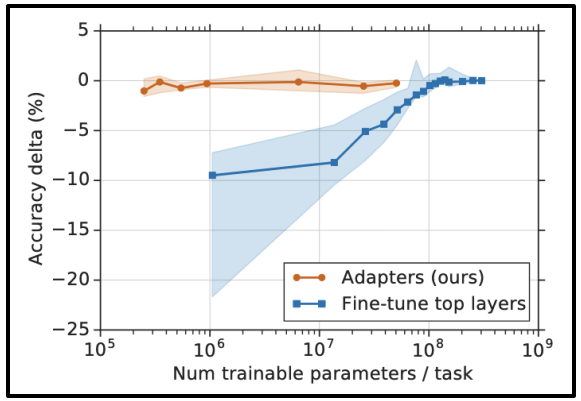
Reparameterized Fine-tuning

- Features
  - constructs a (low-dimensional) reparameterization of the original model parameters for training, then equivalently transforms it back for inference;
- Representative methods
  - LoRA, SoRA, QLoRA

Adapter

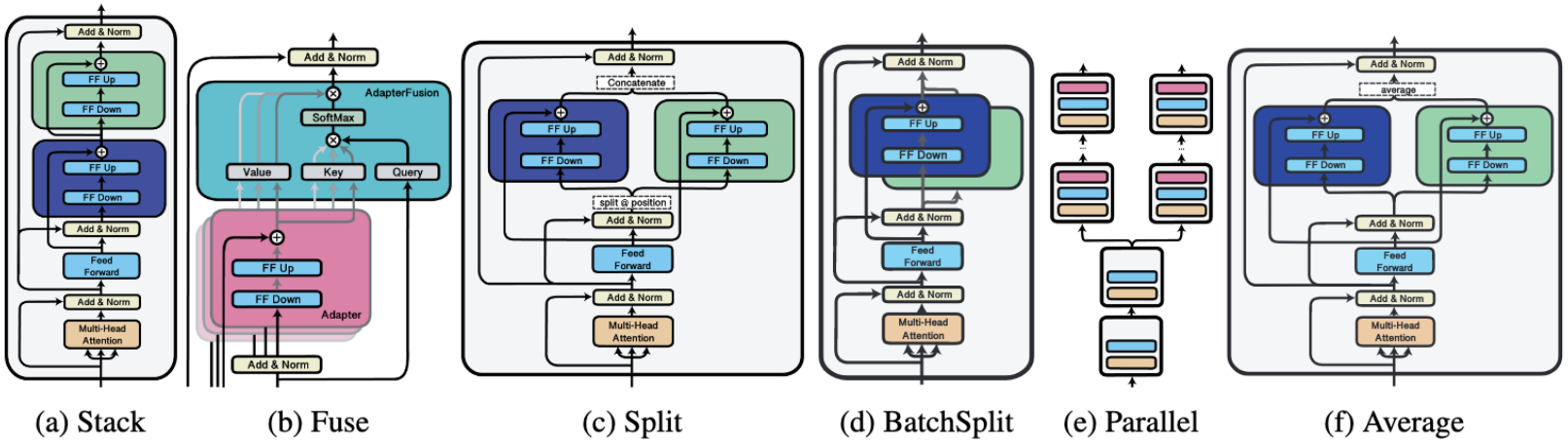






- Freeze model parameters
- Insert and fine-tune low-rank adapter layers after each FFN layer
- The performance is much better than fine-tuning on selected top layers.

Adapter Variations

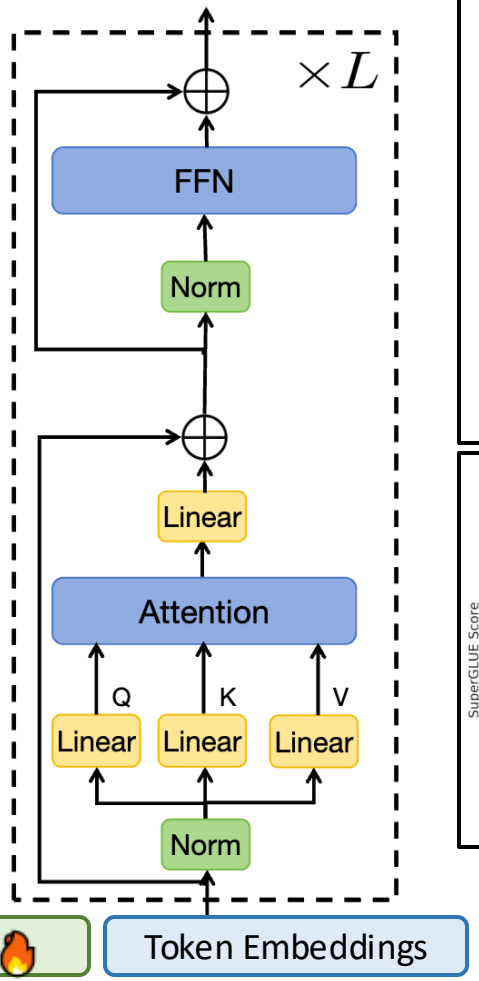


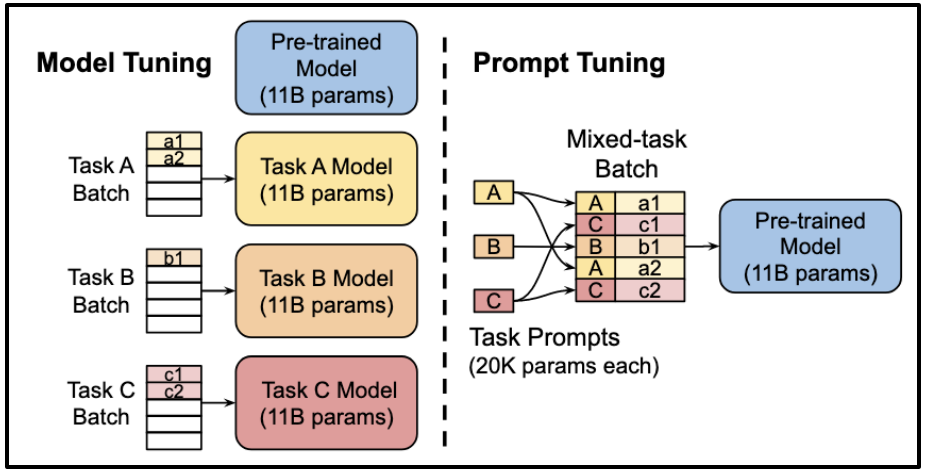
- Stack: sequentially insert adapter layers
- Fuse: combine multiple pre-trained adapters
- Split: split an input sequence along the sequence dimension
- Batch split: split an input sequence along the batch dimension
- Parallel: parallel training and inference on different adapters
- Average: adapter ensembling

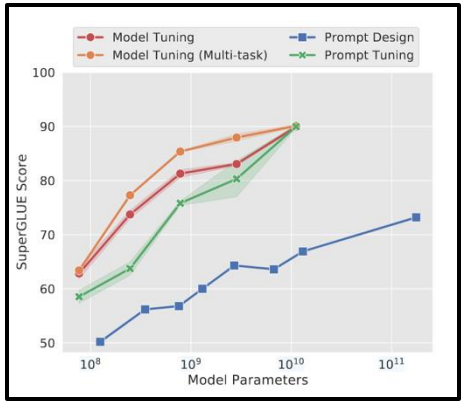
In summary:

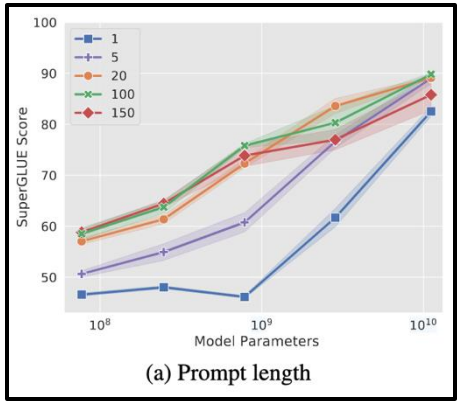
- Pros: flexible design space and can be applied to models other than transformers
- Cons: Induce extra inference cost due to extra layers

Prompt Tuning



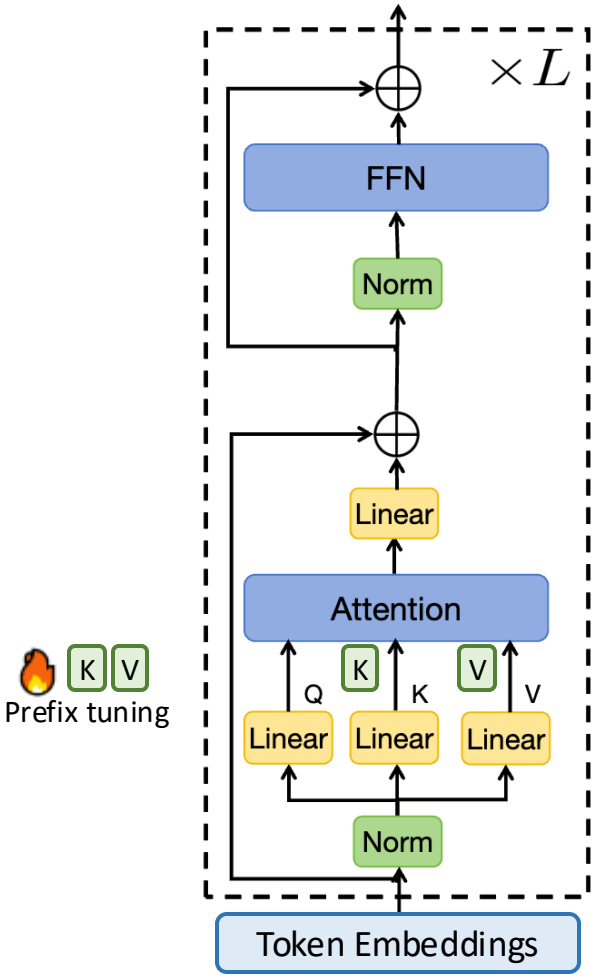


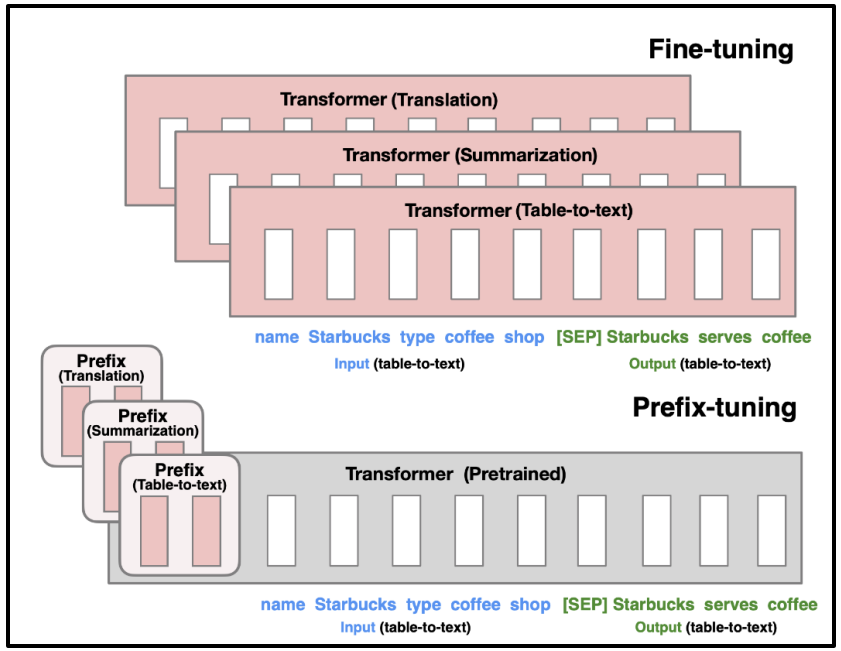




- storing a small task-specific prompt for each task, and enables mixed-task inference using the original pretrained model
- T5 models
  - One token is good for large-scale model (>10^9)
  - increasing beyond 20 tokens only yields marginal gains
  - Much better than prompts designed by GPT-3

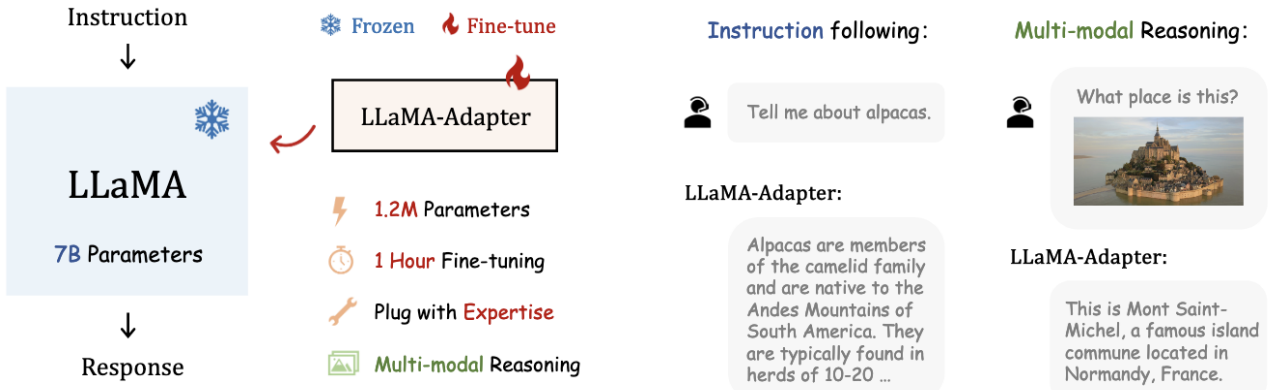
Prefix Tuning





- prepends trainable prefix vectors P^K and P^V to the keys and values of the multi-head attention block inputs

LLaMA-Adapter



$$Q_l = \text{Linear}_q(t_l); \quad (2)$$
$$K_l = \text{Linear}_k([P_l; T_l; t_l]); \quad (3)$$
$$V_l = \text{Linear}_v([P_l; T_l; t_l]). \quad (4)$$

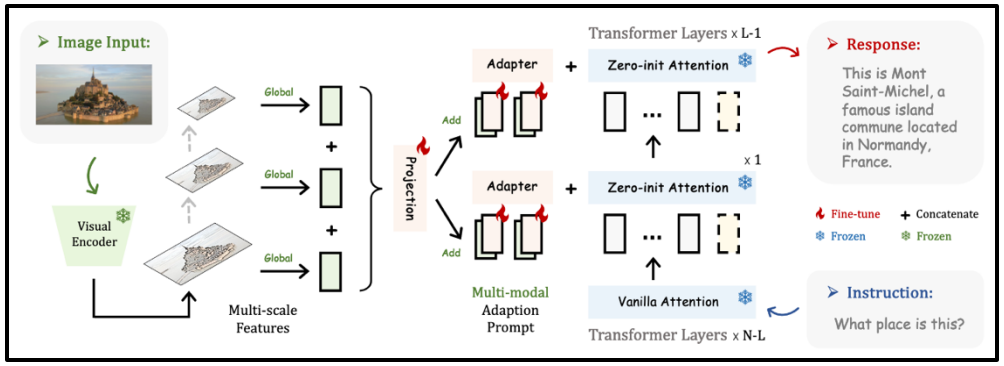
Then, the attention scores of Q_l and K_l before the softmax function are calculated as

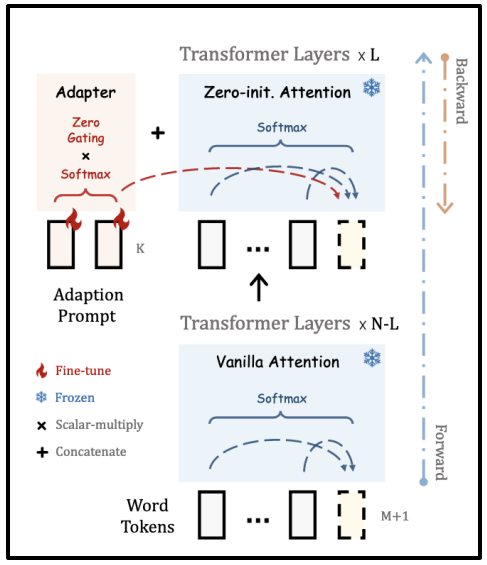
$$S_l = Q_l K_l^T / \sqrt{C} \in \mathbb{R}^{1\times(K+M+1)}, \quad (5)$$

which records the feature similarities between the new word t_l and all $K + M + 1$ tokens. Meanwhile, S_l can be reformulated by two components as

$$S_l = [S_l^K; S_l^{M+1}]^T, \quad (6)$$

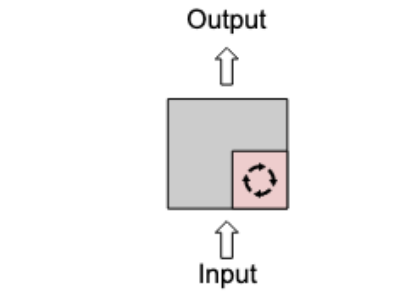
$$S_l^g = [\text{softmax}(S_l^K) \cdot \tanh(g_l); \ \text{softmax}(S_l^{M+1})]^T,$$





- Turning LLaMA 7B into a multimodal instruction following model
- Add prefix tuning into the top L layers
  - Introduce a zero-initialized attention to avoid bringing disturbance to the original word embeddings

Diff Pruning



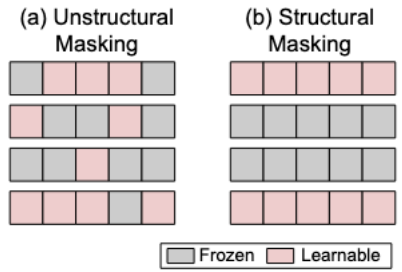


Fig. 7: Illustration of two parameter masking methods.

$$\theta_i' = \theta_i - \eta \cdot m_i \cdot \frac{\partial \mathcal{L}}{\partial \theta_i}$$

- Update parameters iff mask is 1

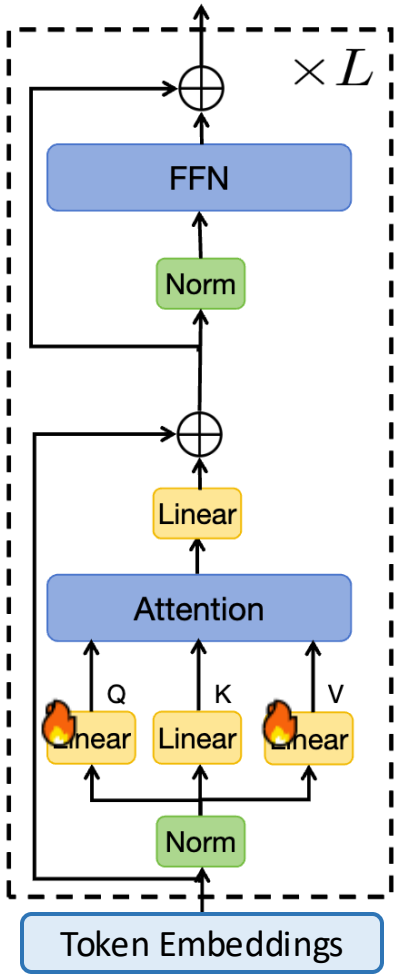
$$\min_{\boldsymbol{\alpha}_\tau, \mathbf{w}_\tau} \mathbb{E}_{\mathbf{z}_\tau \sim p(\mathbf{z}_\tau; \boldsymbol{\alpha}_\tau)} \left[L(\mathcal{D}_\tau, f_\tau, \boldsymbol{\theta} + \boldsymbol{\delta}_\tau) + \lambda \|\boldsymbol{\delta}_\tau\|_0 \right].$$

- Discrete binary mask
  - L0 norm to achieve sparsity
  - However, L0 norm is not differentiable
  - The author proposed a method to estimate the approximated gradient.

$$\min_{\boldsymbol{\alpha}_\tau, \mathbf{w}_\tau} \mathbb{E}_{\mathbf{u} \sim U[\mathbf{0},\mathbf{1}]} \left[L(\mathcal{D}_\tau, f_\tau, \boldsymbol{\theta} + \mathbf{z}_\tau \odot \mathbf{w}_\tau) \right]$$

$$+\lambda \sum_{i=1}^{d} \sigma \left(\boldsymbol{\alpha}_{\tau,i} - \log \frac{-l}{r} \right), \qquad p(\mathbf{z}_\tau; \boldsymbol{\alpha}_\tau)$$

LoRA



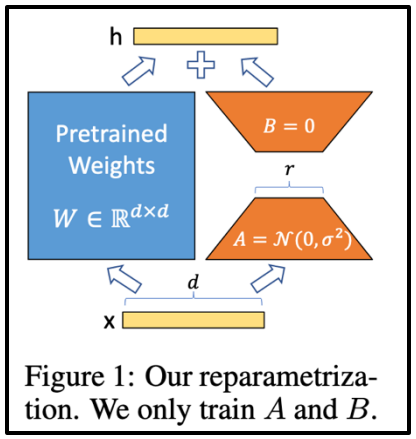


Figure 1: Our reparametrization. We only train A and B.

$$h = (W + \Delta W)x$$
$$\Delta W = BA$$

- Use a low rank approximation (Down projection times up projection) to parameterize the model update
- B matrix is initialized as 0, which can avoid adding disturbance to the original feature space.
- "r" is a hyper-parameter
- After training, one can fuse the update into the original weight matrix.
- Pros:
  - Without requiring extra inference cost

SoRA

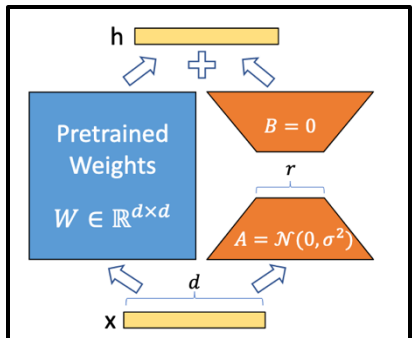
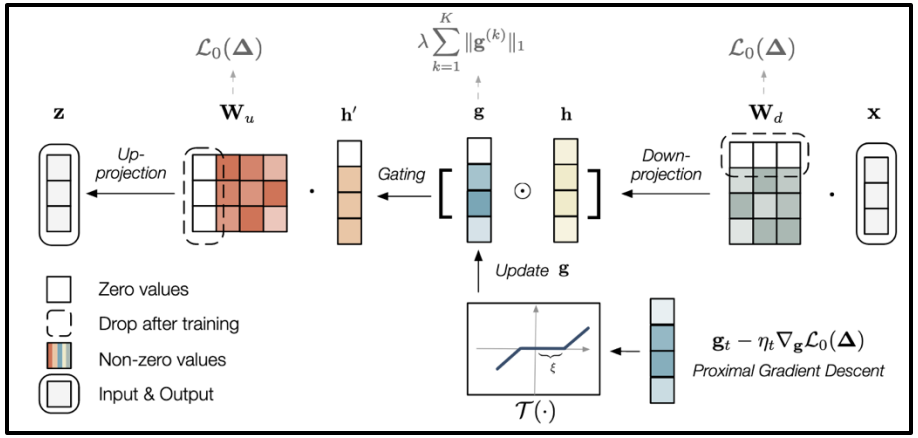


Figure 1: Our reparametrization. We only train A and B.



$$\mathbf{h} = (\mathbf{W} + \Delta\mathbf{W})\mathbf{x}$$

AdaLoRA $$\Delta\mathbf{W} = \mathbf{U}_{d\times d}\mathbf{\Sigma}_{d\times d}\mathbf{V}^T_{d\times d}$$

SoRA $$\Delta\mathbf{W} = \mathbf{W}_y(\mathbf{g} \odot (\mathbf{W}_d\mathbf{x}))$$

$$\mathbf{g}_{t+1} \leftarrow \mathcal{T}_{\eta_t \cdot \lambda}(\mathbf{g}_t - \eta_t \nabla_{\mathbf{g}} \mathcal{L}_0(\mathbf{\Delta}_t)),$$

$$\mathcal{T}_\xi(x) := \begin{cases} x - \xi, & x > \xi \\ 0, & -\xi < x \leq \xi \\ x + \xi, & x \leq -\xi \end{cases} \qquad (11)$$

with $\xi = \eta_t \cdot \lambda$ being the threshold. In practice, the

- In LoRA, “r” is a hyper-parameter.
  - It would be good if there is a method can automatically learn a best “r” in the fine-tuning stage.
- AdaLoRA
  - keep top singular values until the number of nonzero entries
  - Has orthogonal regularization
- SoRA
  - adaptive rank selection in the learning process
  - Post-pruning by selecting non-zero entry in the gate vector
  - Smaller ξ , lower rank

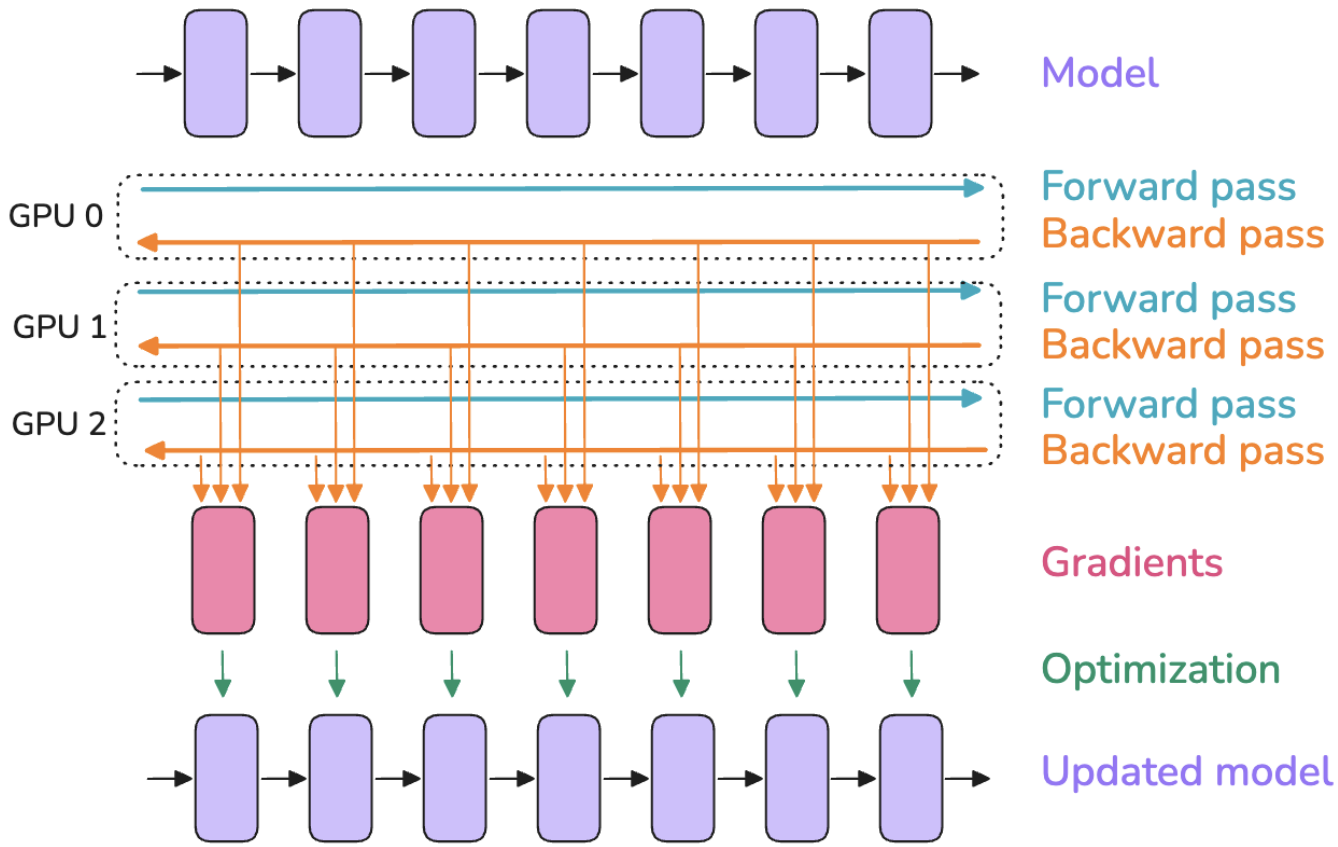
Summary

- Three types of PEFT methods
- GLUE benchmark from the SoRA paper

Method	#Params	CoLA	SST-2	MRPC	QQP	STS-B	MNLI	QNLI	RTE	Avg.
Fine-Tune	184M	69.21	95.64	89.22	92.05/89.31	91.59	89.98/89.95	93.78	82.49	87.82
Adapter	1.41M	69.00	95.16	89.90	91.45/88.88	92.21	90.11/90.11	93.79	82.44	87.85
Bitfit	0.1M	68.70	94.38	87.16	87.86/84.20	89.71	87.45/87.45	91.90	76.12	85.18
LoRA (r=8)	1.33M	69.73	95.57	89.71	91.95/89.26	91.86	**90.47/90.46**	93.76	85.32	88.38
AdaLoRA	1.27M	70.86	**95.95**	90.22	92.13/88.41	91.39	90.27/90.30	**94.28**	87.36	88.83
SoRA	0.91M	**71.48**	95.64	**91.98**	**92.39/89.87**	**92.22**	90.35/90.38	**94.28**	**87.77**	**89.36**

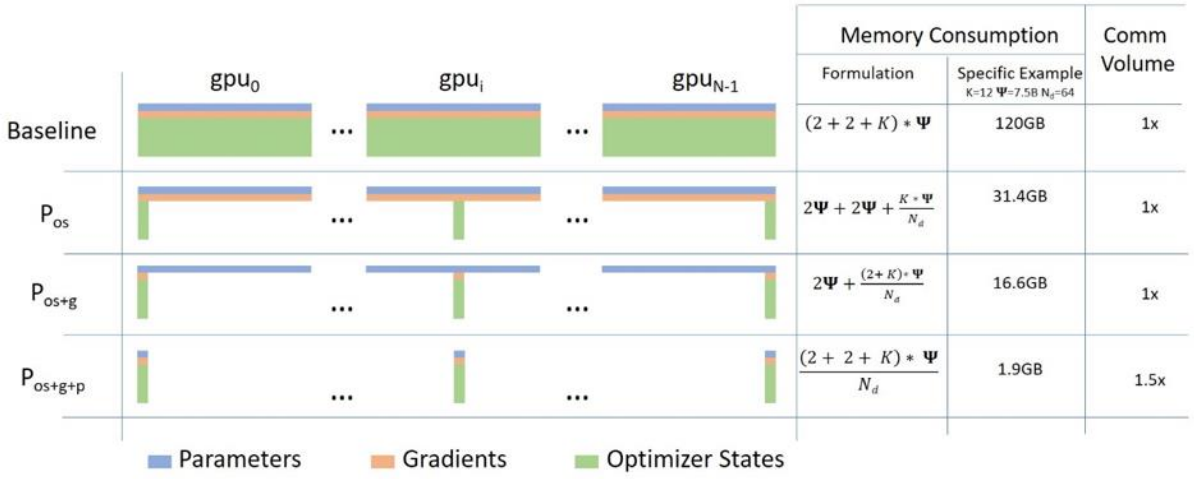
Fine-tuning Techniques: Distributed Training

Data Parallel



Distributed Training

Sharded Training - DeepSpeed



	Memory Consumption		Comm Volume
	Formulation	Specific Example K=12 Ψ=7.5B N_d=64	
Baseline	$(2+2+K)*\Psi$	120GB	1x
P_{os}	$2\Psi+2\Psi+\frac{K*\Psi}{N_d}$	31.4GB	1x
P_{os+g}	$2\Psi+\frac{(2+K)*\Psi}{N_d}$	16.6GB	1x
P_{os+g+p}	$\frac{(2+2+K)*\Psi}{N_d}$	1.9GB	1.5x

input : γ (lr), β_1, β_2 (betas), θ_0 (params), $f(\theta)$ (objective)
λ (weight decay), *amsgrad*, *maximize*, ϵ (epsilon)
initialize : $m_0 \leftarrow 0$ (first moment), $v_0 \leftarrow 0$ (second moment), $\widehat{v_0}^{max} \leftarrow 0$

for $t = 1$ **to** … **do**
  if *maximize* :
    $g_t \leftarrow -\nabla_\theta f_t(\theta_{t-1})$
  else
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
  if $\lambda \neq 0$
    $g_t \leftarrow g_t + \lambda\theta_{t-1}$
  $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$
  $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$
  $\widehat{m_t} \leftarrow m_t / (1-\beta_1^t)$
  $\widehat{v_t} \leftarrow v_t / (1-\beta_2^t)$
  if *amsgrad*
    $\widehat{v_t}^{max} \leftarrow \max(\widehat{v_{t-1}}^{max}, \widehat{v_t})$
    $\theta_t \leftarrow \theta_{t-1} - \gamma \widehat{m_t} / (\sqrt{\widehat{v_t}^{max}} + \epsilon)$
  else
    $\theta_t \leftarrow \theta_{t-1} - \gamma \widehat{m_t} / (\sqrt{\widehat{v_t}} + \epsilon)$

return θ_t

For every parameter θ, Adam maintains:

1. **The parameter value itself** – θ
2. **Gradient (g)** – the current gradient
3. **First moment estimate (m)** – running average of past gradients
4. **Second moment estimate (v)** – running average of squared gradients

So compared to vanilla SGD which needs only:

- θ and g (2 tensors)

Adam needs:

- θ, g, m, and v (4 tensors)

That means:
📈 **~2x more memory** per parameter.

Sharded Training - FSDP

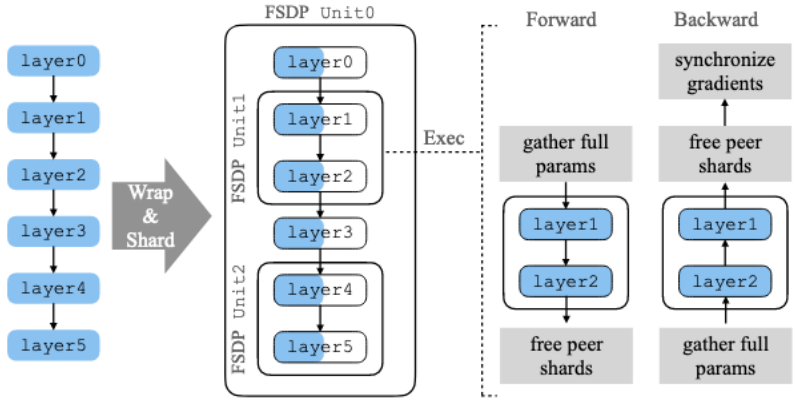
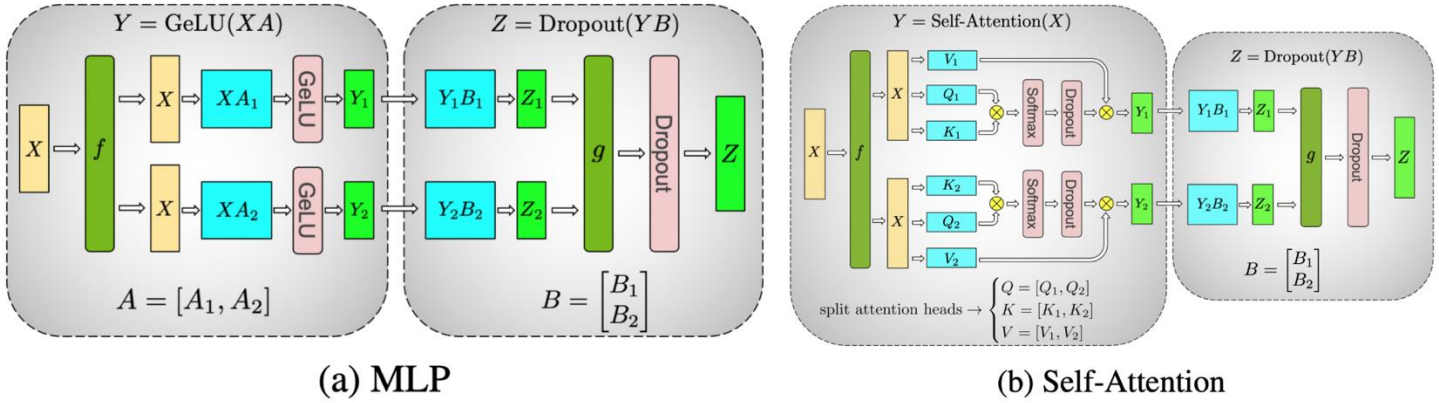


Figure 1: FSDP Algorithm Overview

Tensor Parallel - Megatron



(a) MLP (b) Self-Attention

```
class f(torch.autograd.Function):
    def forward(ctx, x):
        return x
    def backward(ctx, gradient):
        all_reduce(gradient)
        return gradient
```

Code 1. Implementation of f operator. g is similar to f with identity in the backward and all-reduce in the forward functions.

- Column-wise splitting and Row-wise splitting