UNC CHARLOTTE

The WILLIAM STATES LEE COLLEGE *of* ENGINEERING

Introduction to ML
Lecture 14: Transformers-at-scale

Hamed Tabkhi
Department of Electrical and Computer Engineering,
University of North Carolina Charlotte (UNCC)
htabkhiv@uncc.edu

UNC CHARLOTTE

Large-Scale Pretraining with Transformers

- So far in our image classification and machine translation experiments, models were trained on datasets with input-output examples *from scratch* to perform specific tasks. For example, a Transformer was trained with English-French pairs so that this model can translate input English text into French.
- As a result, each model becomes a *specific expert* that is sensitive to even slight shift in data distribution.
- For better generalized models, or even more competent *generalists* that can perform multiple tasks with or without adaptation, *pretraining* models on large data has been increasingly common.

Large-Scale Pretraining with Transformers

- Given larger data for pretraining, the Transformer architecture performs better with an increased model size and training compute, demonstrating superior *scaling* behavior.
- Specifically, performance of Transformer-based language models scales as a power-law with the amount of model parameters, training tokens, and training compute (Kaplan *et al.*, 2020).
- The scalability of Transformers is also evidenced by the significantly boosted performance from larger vision Transformers trained on larger data.
- More recent success stories include Gato, a *generalist* model that can play Atari, caption images, chat, and act as a robot (Reed *et al.*, 2022).
- Gato is a single Transformer that scales well when pretrained on diverse modalities, including text, images, joint torques, and button presses.
- Notably, all such multi-modal data is serialized into a flat sequence of tokens, which can be processed akin to text tokens or image patches (Section 11.8) by Transformers.
- Before compelling success of pretraining Transformers for multi-modal data, Transformers were extensively pretrained with a wealth of text.

Large-Scale Pretraining with Transformers

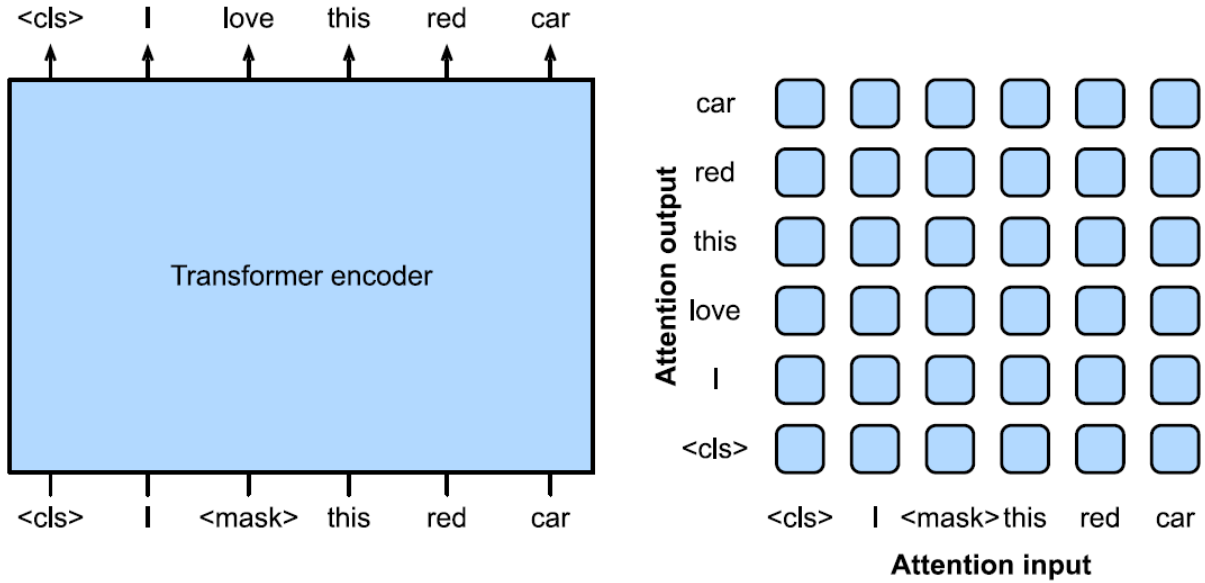
- Originally proposed for machine translation, the Transformer architecture consists of an encoder for representing input sequences and a decoder for generating target sequences.
- Primarily, Transformers can be used in three different modes: *encoder-only*, *encoder-decoder*, and *decoder-only*.
- To conclude this chapter, we will review these three modes and explain the scalability in pretraining Transformers.

Encoder-Only

- When only the Transformer encoder is used, a sequence of input tokens is converted into the same number of representations that can be further projected into output (e.g., classification).
- A Transformer encoder consists of self-attention layers, where all input tokens attend to each other.
- For example, vision Transformers depicted are encoder-only, converting a sequence of input image patches into the representation of a special “<cls>” token.
- Since this representation depends on all input tokens, it is further projected into classification labels.
- This design was inspired by an earlier encoder-only Transformer pretrained on text.

Pretraining BERT

- BERT (Bidirectional Encoder Representations from Transformers) (Devlin *et al.*, 2018).



- Left: Pretraining BERT with masked language modeling. Prediction of the masked love token depends on all input tokens before and after love.
- Right: Attention pattern in the Transformer encoder. Each token along the vertical axis attends to all input tokens along the horizontal axis.

Pretraining BERT

- BERT is pretrained on text sequences using *masked language modeling*: input text with randomly masked tokens is fed into a Transformer encoder to predict the masked tokens.
- As illustrated in the figure, an original text sequence “I”, “love”, “this”, “red”, “car” is prepended with the “<cls>” token, and the “<mask>” token randomly replaces “love”.
- The crossentropy loss between the masked token “love” and its prediction is to be minimized during pretraining.
- **Note that there is no constraint in the attention pattern of Transformer encoders (right of Fig) so all tokens can attend to each other.**
- Thus, prediction of “love” depends on input tokens before and after it in the sequence. This is why BERT is a “bidirectional encoder”.
- **Without need for manual labeling, large-scale text data from books and Wikipedia can be used for pretraining BERT.**

Fine-Tuning BERT

- The pretrained BERT can be *fine-tuned* to downstream encoding tasks involving single text or text pairs.
- During fine-tuning, additional layers can be added to BERT with randomized parameters: these parameters and those pretrained BERT parameters will be *updated* to fit training data of downstream tasks.

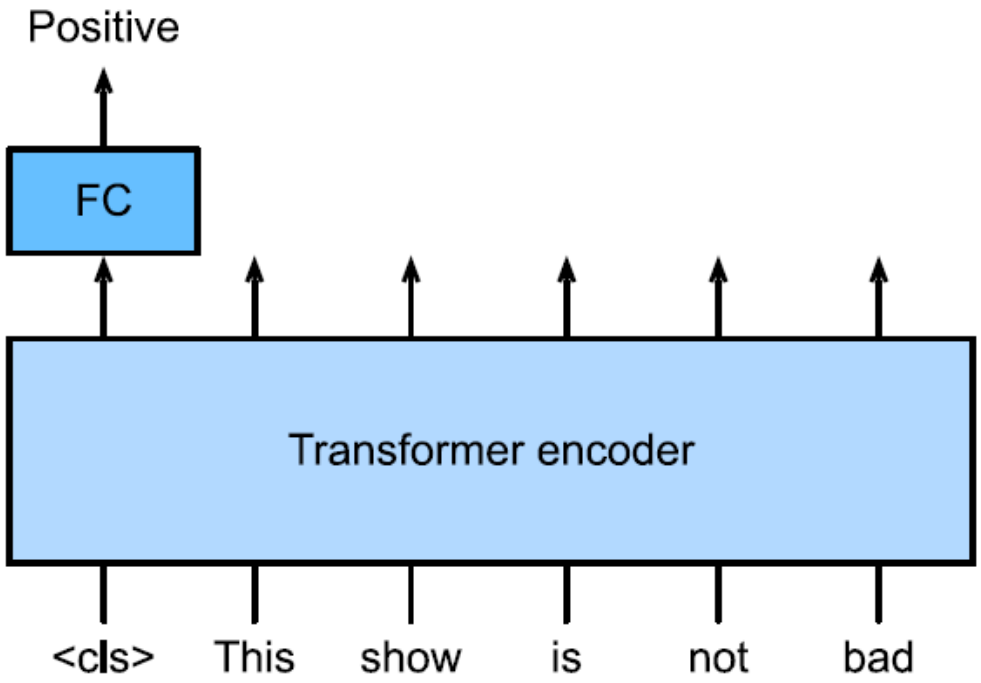


Figure illustrates fine-tuning of BERT for sentiment analysis.

Fine-Tuning BERT

- The Transformer encoder is a pretrained BERT, which takes a text sequence as input and feeds the “<cls>” representation (global representation of the input) into an additional fully connected layer to predict the sentiment.
- During fine-tuning, the cross-entropy loss between the prediction and the label on sentiment analysis data is minimized via gradient-based algorithms, where the additional layer is trained from scratch while pretrained parameters of BERT are updated.
- These downstream tasks include text pair understanding.
- The general language representations learned by the 350-million-parameter BERT from 250 billion training tokens advanced the state of the art for natural language tasks such as single text classification, text pair classification or regression, text tagging, and question answering.

Sentiment analysis is the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information.

Fine-Tuning BERT

- BERT pretraining has another loss for predicting whether one sentence immediately follows the other.
- However, this loss was later found not useful when pretraining RoBERTa, a BERT variant of the same size, on 2000 billion tokens (Liu *et al.*, 2019).

- Other derivatives of BERT improved model architectures or pretraining objectives, such as:
  - ALBERT (enforcing parameter sharing) (Lan *et al.*, 2019),
  - SpanBERT (representing and predicting spans of text) (Joshi *et al.*, 2020),
  - Distil- BERT (lightweight via knowledge distillation) (Sanh *et al.*, 2019),
  - ELECTRA (replaced token detection) (Clark *et al.*, 2020).

- Moreover, BERT inspired Transformer pretraining in computer vision, such as:
  - vision Transformers (Dosovitskiy *et al.*, 2021)
  - Swin Transformers (Liu *et al.*, 2021)
  - MAE (masked autoencoders) (He *et al.*, 2022)

Encoder-Decoder

- Since a Transformer encoder converts a sequence of input tokens into the same number of output representations**, the encoder-only mode cannot generate a sequence of arbitrary length like in machine translation.**

- As originally proposed for machine translation, the Transformer architecture can be outfitted with a decoder that autoregressively predicts the target sequence of arbitrary length, token by token, conditional on both encoder output and decoder output:

(i) Conditioning on encoder output is achieved through encoder-decoder cross-attention (multi-head attention of decoder) allows target tokens to attend to *all* input tokens.
(ii) Conditioning on decoder output is achieved by a so-called *causal* attention (this name is common in the literature but is misleading as it has little connection to the proper study of causality) with (masked multi-head attention of decoder), where any target token can only attend to *past* and *present* tokens in the target sequence.

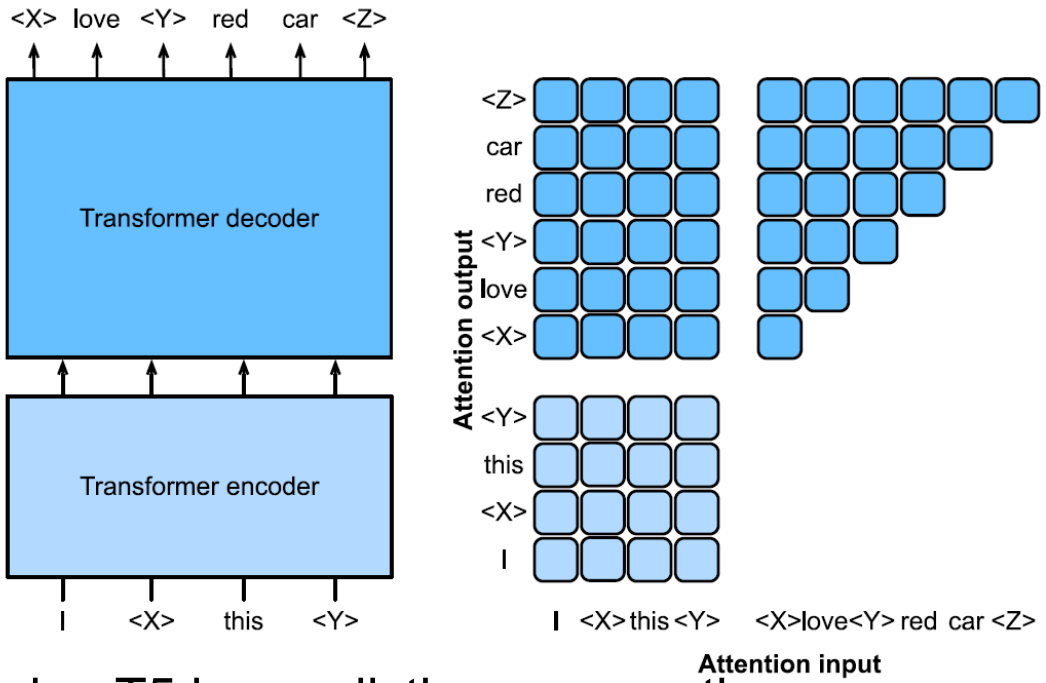
Encoder-Decoder

- To pretrain encoder-decoder Transformers BART (Lewis *et al.*, 2019) and T5 (Raffel *et al.*, 2020) are two concurrently proposed encoder-decoder Transformers pretrained on large-scale text corpora.

Pretraining T5

- As an example of the pretrained Transformer encoder-decoder, T5 (Text-to-Text Transfer Transformer) unifies many tasks as the same text-to-text problem.
- For any task, the input of the encoder is a task description (e.g., “Summarize”, “:”) followed by task input (e.g., a sequence of tokens from an article), and the decoder predicts the task output (e.g., a sequence of tokens summarizing the input article).
- To perform as text-to-text, T5 is trained to generate some target text conditional on input text.

Pretraining T5



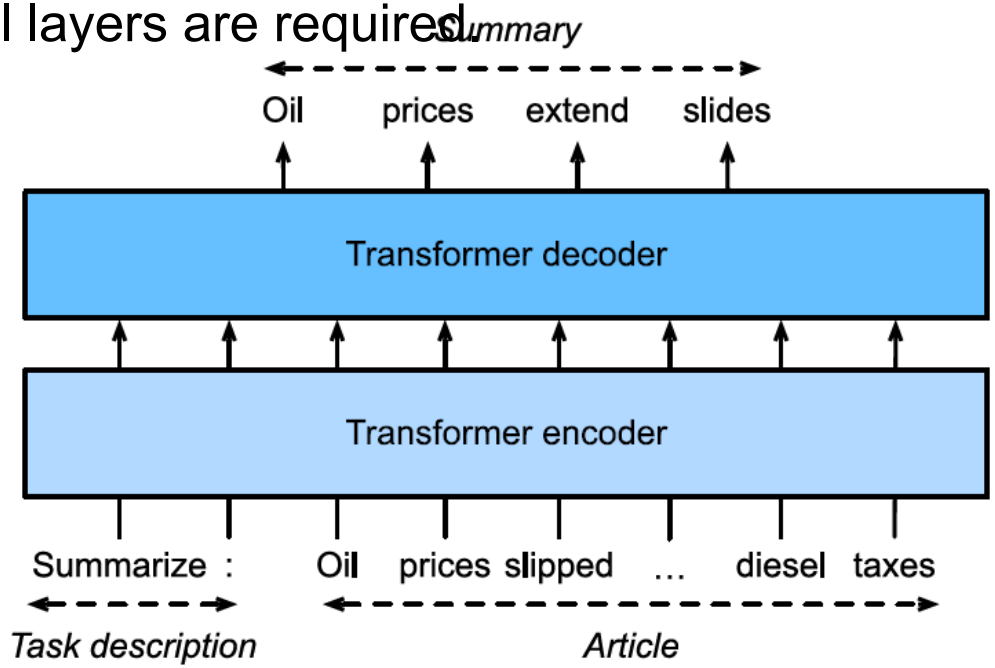
- Left: Pretraining T5 by predicting consecutive spans.
  - The original sentence is I, love, this, red, car, where love is replaced by a special <X> token, and consecutive red, car are replaced by a special <Y> token.
  - The target sequence ends with a special <Z> token.
- Right: Attention pattern in the Transformer encoder-decoder.
  - In the encoder self-attention (lower square), all input tokens attend to each other
  - In the encoder-decoder cross-attention (upper rectangle), each target token attends to all input tokens
  - In the decoder self-attention (upper triangle), each target token attends to

Pretraining T5

- To obtain input and output from any original text, T5 is pretrained to predict consecutive spans.
- **Specifically, tokens from text are randomly replaced by special tokens where each consecutive span is replaced by the same special token.**
- For example, the original text is “I”, “love”, “this”, “red”, “car”. Tokens “love”, “red”, “car” are randomly replaced by special tokens.
- Since “red” and “car” are a consecutive span, they are replaced by the same special token.
- The decoder has a causal attention pattern to prevent itself from attending to future tokens during sequence prediction.
- In T5, predicting consecutive span is also referred to as reconstructing corrupted text.
- With this objective, T5 is pretrained with 1000 billion tokens from the C4 (Colossal Clean Crawled Corpus) data, which consists of clean English text from the Web (Raffel *et al.*, 2020).

Fine-Tuning T5

- Similar to BERT, T5 needs to be fine-tuned (updating T5 parameters) on task-specific training data to perform this task.
- Major differences from BERT fine-tuning include:
  (i) T5 input includes task descriptions
  (ii) T5 can generate sequences with arbitrary length with its Transformer Decoder
  (iii) No additional layers are required.



Fine-tuning T5 for text summarization. Both the task description and article tokens are fed into the Transformer encoder for predicting the summary.

Fine-Tuning T5

- The figure explains fine-tuning T5 using text summarization as an example.
- In its downstream task, the task description tokens “Summarize”, “:” followed by the article tokens are input to the encoder.
- After fine-tuning, the 11-billion-parameter T5 (T5-11B) achieved state-of-the-art results on multiple encoding (e.g., classification) and generation (e.g., summarization) benchmarks.
- Since released, T5 has been extensively used in later research.
- For example, switch Transformers are designed based off T5 to activate a subset of the parameters for better computational efficiency (Fedus *et al.*, 2022).
- In a text-to-image model called Imagen, text is input to a frozen T5 encoder (T5-XXL) with 4.6 billion parameters (Saharia *et al.*, 2022).

Fine-Tuning T5

- The photorealistic text-to-image examples suggest that the T5 encoder alone may effectively represent text even without fine-tuning.

Teddy bears swimming at the Olympics 400m Butter-fly event.

A cute corgi lives in a house made out of sushi.

A cute sloth holding a small treasure chest. A bright golden glow is coming from the chest.

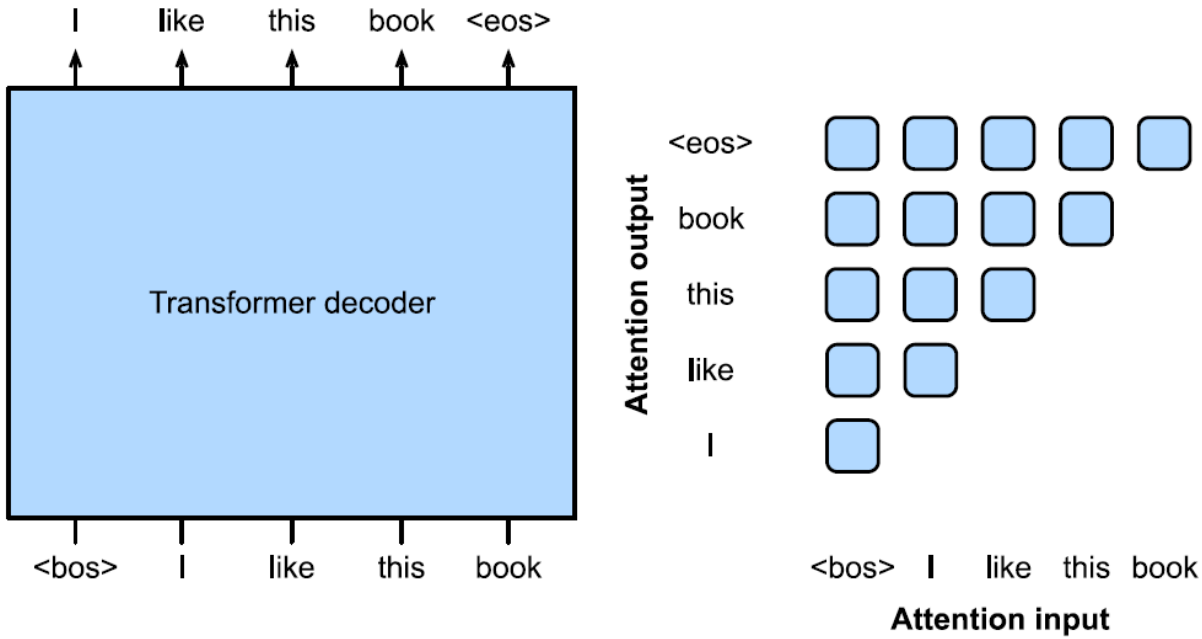
Text-to-image examples by the Imagen model, whose text encoder is from T5

Decoder-Only

- We have reviewed encoder-only and encoder-decoder Transformers
- **Decoder-only Transformers remove the entire encoder and the decoder sublayer with the encoder2decoder cross-attention from the original encoder-decoder architecture.**
- Nowadays, decoder-only Transformers have been the de facto architecture in large-scale language modeling, which leverages the world's abundant unlabeled text corpora via self-supervised learning.

GPT and GPT-2

- Using language modeling as the training objective, the GPT (generative pre-training) model chooses a Transformer decoder as its backbone (Radford *et al.*, 2018).



- Left: Pretraining GPT with language modeling. The target sequence is the input sequence shifted by one token. Both <bos> and <eos> are special tokens marking the beginning and end of sequences, respectively.
- Right: Attention pattern in the Transformer decoder. Each token along the vertical axis attends to only its past tokens along the horizontal axis (causal).

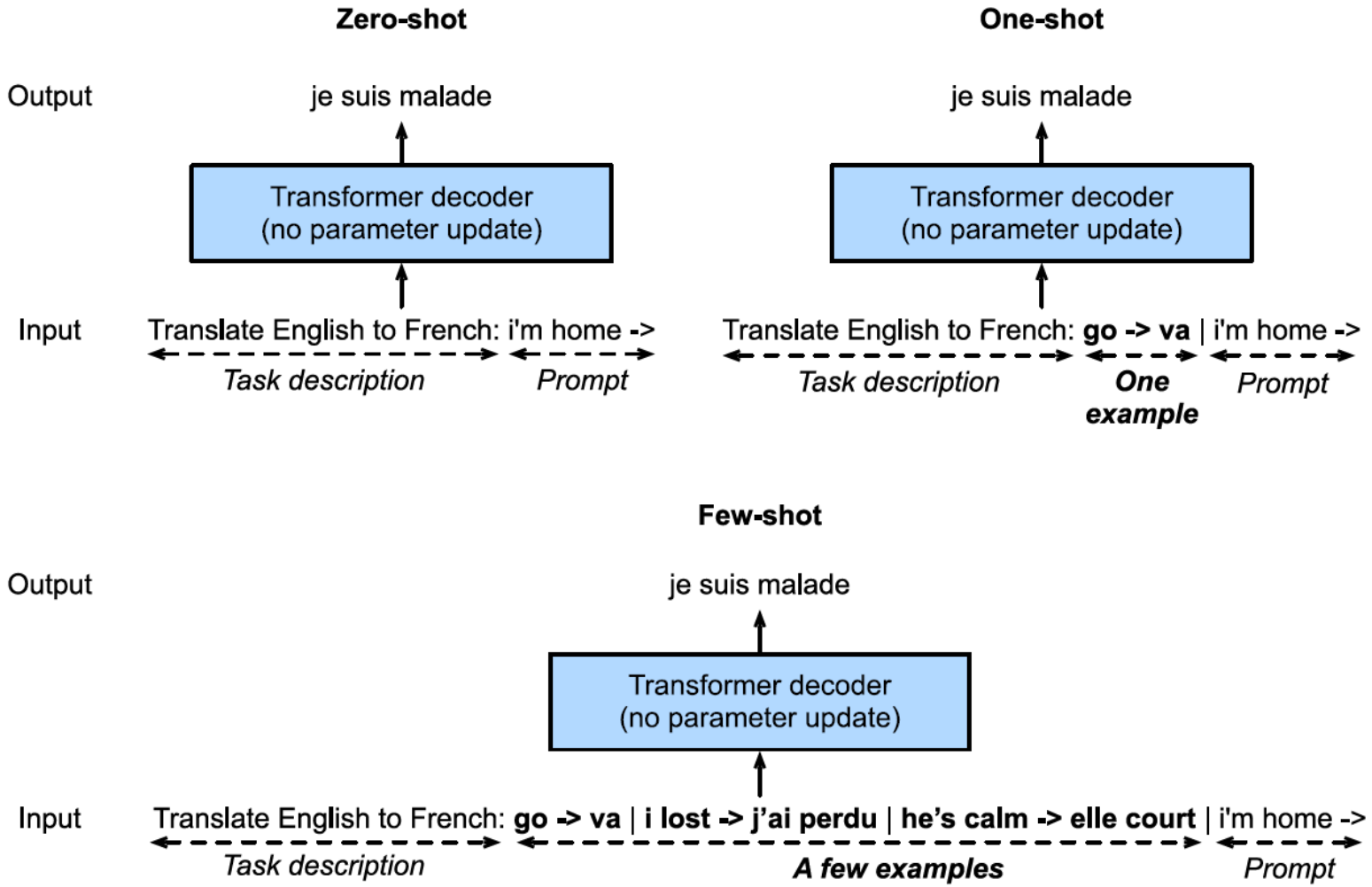
GPT and GPT-2

- **GPT uses pretraining with a Transformer decoder, where the target sequence is the input sequence shifted by one token.**
- Note that the attention pattern in the Transformer decoder enforces that each token can only attend to its past tokens (future tokens cannot be attended to because they have not yet been chosen).
- GPT has 100 million parameters and needs to be fine-tuned for individual downstream tasks.
- A much larger Transformer-decoder language model, GPT-2, was introduced one year later (Radford *et al.*, 2019).
- Compared with the original Transformer decoder in GPT, pre-normalization and improved initialization and weight-scaling were adopted in GPT-2.
- Pretrained on 40 GB of text, the 1.5-billion-parameter GPT-2 obtained the state-of-the-art results on language modeling benchmarks and promising results on multiple other tasks *without updating the parameters or architecture*.

GPT-3

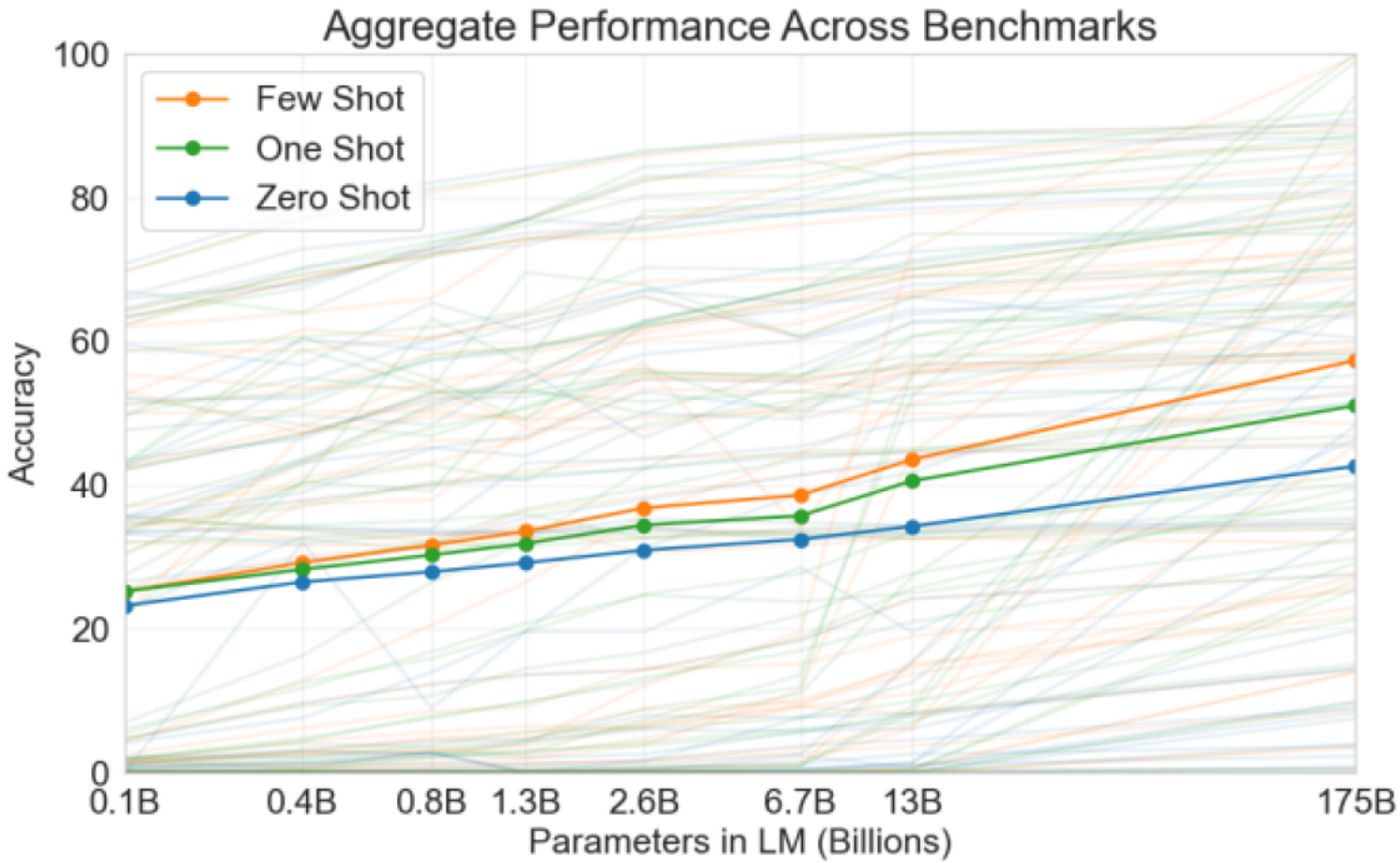
- GPT-2 demonstrated potential of using the same language model for multiple tasks without updating the model.
- This is more computationally efficient than fine-tuning, which requires model updates via gradient computation.
- Recall that language model can be trained to generate a text sequence conditional on some prefix text sequence.
- This learning paradigm is called *in-context learning* (Brown *et al.*, 2020), which can be further categorized into *zero-shot*, *one-shot*, and *few-shot*, when there is no, one, and a few task-specific input-output examples, respectively.

GPT-3



Zero-shot, one-shot, few-shot in-context learning with language models (Transformer
decoders). No parameter update is needed.

GPT-3



Aggregate performance of GPT-3 for all 42 accuracy-denominated benchmarks

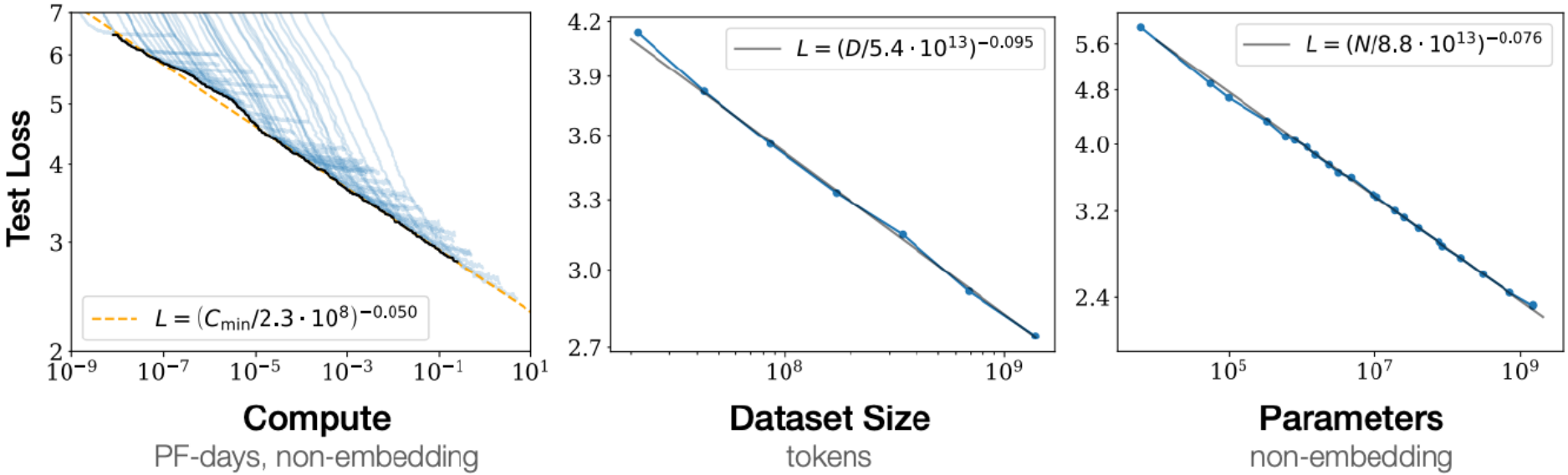
UNC CHARLOTTE

GPT-3

- These three settings were tested in GPT-3 (Brown *et al.*, 2020), whose largest version uses data and model size about two orders of magnitude larger than those in GPT-2.
- GPT-3 uses the same Transformer decoder architecture in its direct predecessor GPT-2 except that attention patterns are sparser at alternating layers.
- Pretrained with 300 billion tokens, GPT-3 performs better with larger model size, where few-shot performance increases most rapidly.
- Although enjoying computational efficiency, GPT-3 fewshot learning underperformed the state-of-the-art fine-tuned models that require model updates.

GPT 3 Scalability

- Fig. empirically demonstrates scalability of Transformers in the GPT -3 language model.
- For language modeling, more comprehensive empirical studies on the scalability of Transformers have led researchers to see promise in training larger Transformers with more data and compute (Kaplan *et al.*, 2020).

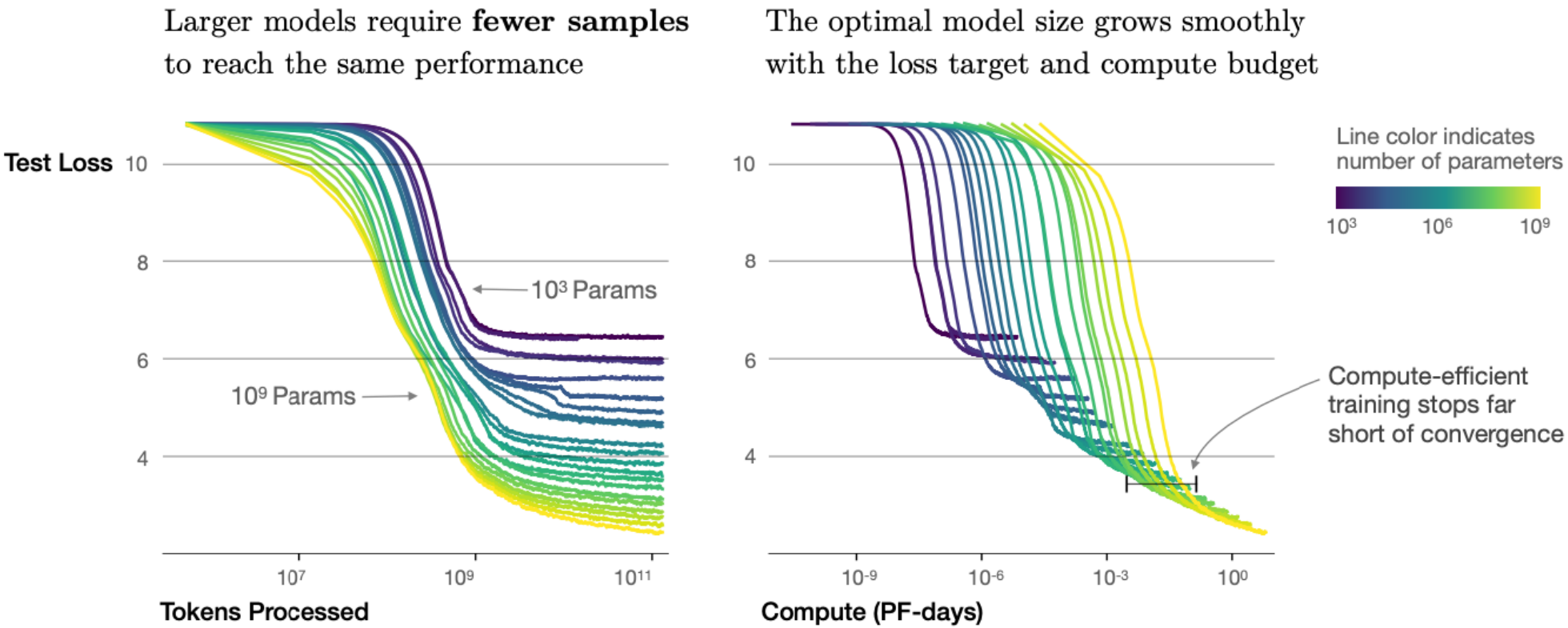


UNC CHARLOTTE

Scalability

- Transformer language model performance improves smoothly as we increase the **model size, dataset size, and amount of compute** used for training.
- For optimal performance all three factors must be scaled up in tandem.
- Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two (caption adapted and figure taken from Kaplan et al. (2020)).
- *Power-law scaling* can be observed in the performance with respect to the model size (number of parameters, excluding embedding layers), dataset size (number of training tokens), and amount of training compute (PetaFLOP/s-days, excluding embedding layers).
- In general, increasing all these three factors in tandem leads to better performance.
- However, *how* to increase them in tandem still remains a matter of debate (Hoffmann *et al.*, 2022).

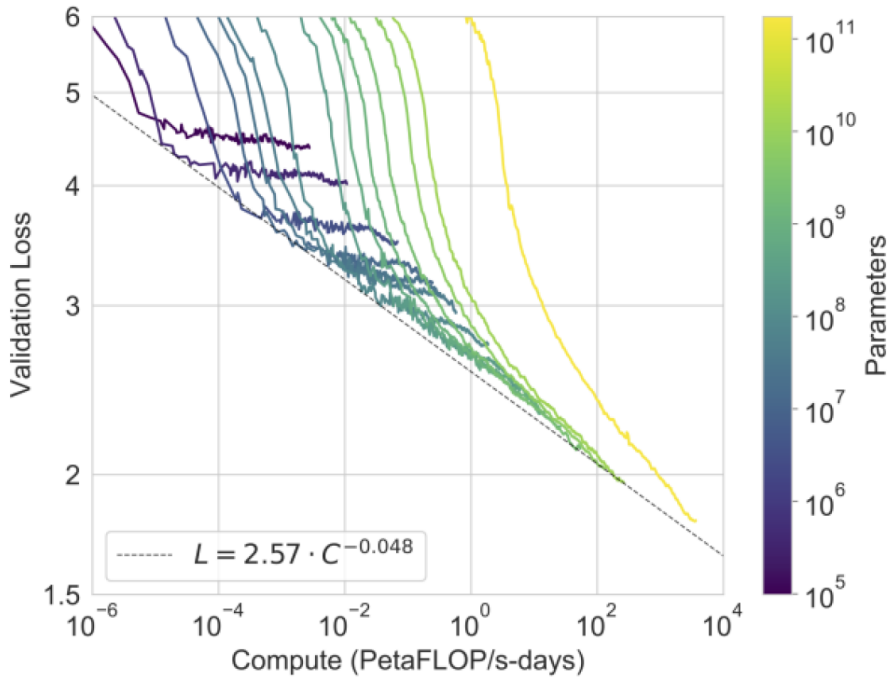
Scalability



Transformer language model training runs (figure taken from Kaplan et al. (2020)).

UNC CHARLOTTE

Scalability



- GPT-3 performance (cross-entropy validation loss) follows a power-law trend with the amount of compute used for training.
- The power-law behavior observed in Kaplan et al. (2020) continues for an additional two orders of magnitude with only small deviations from the predicted curve.
- Embedding parameters are excluded from compute and parameter counts (caption adapted and figure taken from Brown et al. (2020)).

UNC CHARLOTTE

Scalability

- Besides increased performance, large models also enjoy better sample efficiency than small models.
- Large models need fewer training samples (tokens processed) to perform at the same level achieved by small models, and performance is scaled smoothly with compute.
- The empirical scaling behaviors in Kaplan *et al.* (2020) have been tested in subsequent large Transformer models.
- For example, GPT-3 supported this hypothesis with two more orders of magnitude.
- The scalability of Transformers in the GPT series has inspired subsequent Transformer language models.
- While the Transformer decoder in GPT-3 was largely followed in OPT (Open Pretrained Transformers) (Zhang *et al.*, 2022) using only 1/7th the carbon footprint of the former.
- The GPT-2 Transformer decoder was used in training the 530-billion-parameter Megatron-Turing NLG (Smith *et al.*, 2022) with 270 billion training tokens.

Scalability

- Following the GPT-2 design, the 280-billion-parameter Gopher (Rae *et al.*, 2021) pretrained with 300 billion tokens achieved state-of-the-art performance across the majority on about 150 diverse tasks.
- Inheriting the same architecture and using the same compute budget of Gopher, Chinchilla
- (Hoffmann *et al.*, 2022) is a substantially smaller (70 billion parameters) model that trains much longer (1.4 trillion training tokens), outperforming Gopher on many tasks.
- To continue the scaling line of language modeling, PaLM (Pathway Language Model) (Chowdhery *et al.*, 2022), a 540-billion-parameter Transformer decoder with modified designs pretrained on 780 billion tokens, outperformed average human performance on the BIG-Bench benchmark (Srivastava *et al.*, 2022).
- Further training PaLM on 38.5 billion tokens containing scientific and mathematical content results in Minerva (Lewkowycz *et al.*, 2022), a large language model that can answer nearly a third of undergraduate-level problems that require quantitative reasoning, such as in physics, chemistry, biology, and economics.

Summary and Discussion

- Transformers have been pretrained as encoder-only (e.g., BERT), encoder-decoder (e.g., T5), and decoder-only (e.g., GPT series).
- Pretrained models may be adapted to perform different tasks with model update (e.g., fine tuning) or not (e.g., few shot).
- Scalability of Transformers suggests that better performance benefits from larger models, more training data, and more training compute.
- Since Transformers were first designed and pretrained for text data, this section leans slightly towards natural language processing.
- Nonetheless, those models discussed above can be often found in more recent models across multiple modalities.

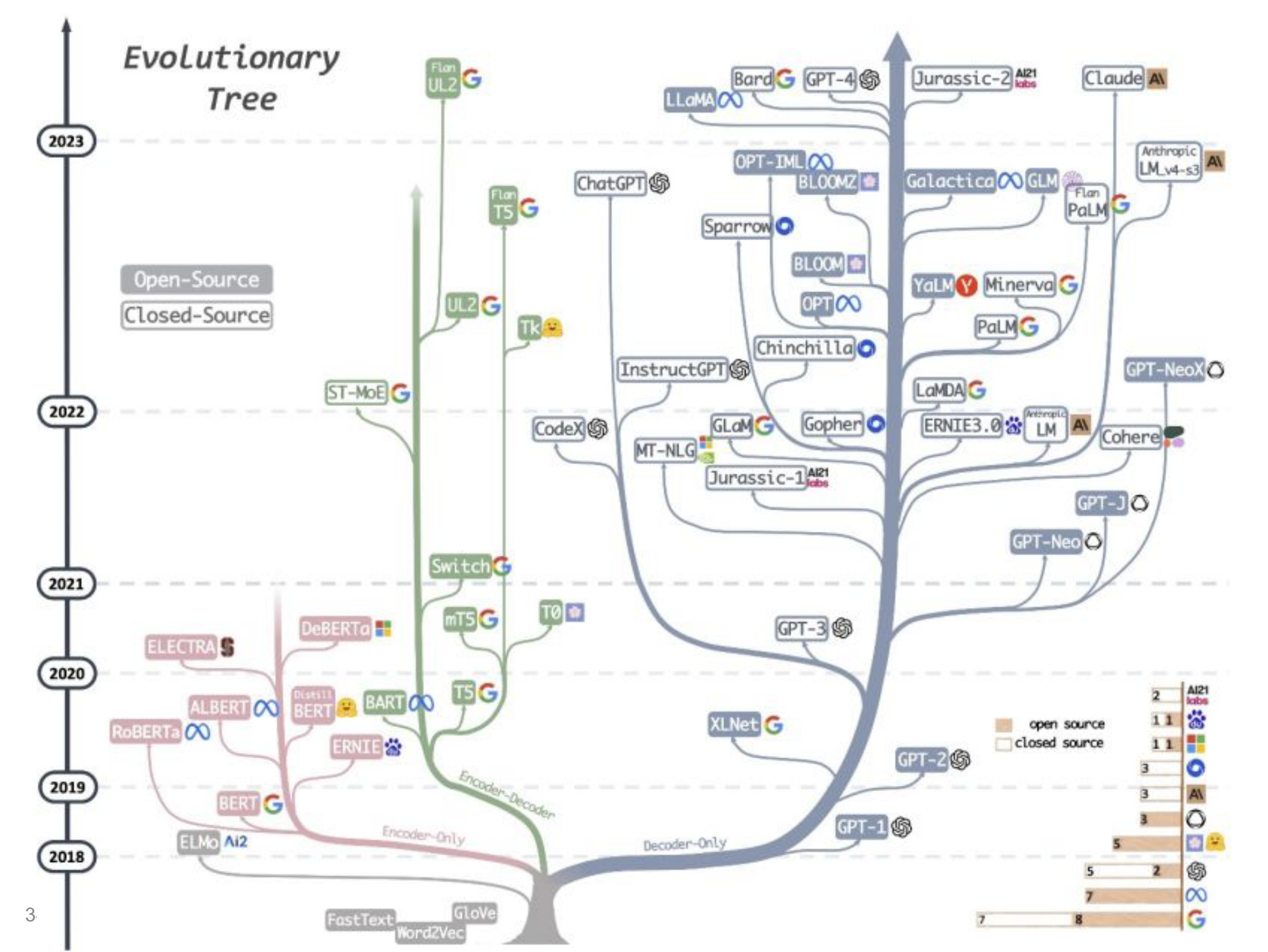
Summary and Discussion

Although there has been no systematic studies on Transformer scalability in multi-modal pretraining yet, a recent all-Transformer text-to-image model, Parti (Yu *et al.*, 2022), shows potential of scalability across modalities.

A portrait photo of a kangaroo wearing an orange hoodie and blue sunglasses standing on the grass in front of the Sydney Opera House holding a sign on the chest that says Welcome Friends!

Image examples generated from the same text by the Parti model of increasing sizes (350M, 750M, 3B, 20B) (examples taken from Yu et al. (2022)).

UNC CHARLOTTE


Evolutionary Tree
2023
2022
2021
2020
2019
2018
Open-Source
Closed-Source
Flan UL2
Bard
GPT-4
Jurassic-2
Claude
LLaMA
OPT-IML
BLOOMZ
ChatGPT
Galactica
GLM
Anthropic LM_v4-s3
Flan T5
Sparrow
Flan PaLM
BLOOM
YaLM
Minerva
UL2
OPT
Tk
PaLM
Chinchilla
InstructGPT
GPT-NeoX
ST-MoE
LaMDA
CodeX
GLaM
Gopher
ERNIE3.0
Anthropic LM
Cohere
MT-NLG
Jurassic-1
GPT-J
GPT-Neo
Switch
mT5
T0
DeBERTa
GPT-3
ELECTRA
ALBERT
Distill BERT
BART
T5
XLNet
RoBERTa
ERNIE
GPT-2
BERT
Encoder-Decoder
Encoder-Only
Decoder-Only
GPT-1
ELMo
AI2
FastText
GloVe
Word2Vec
open source
closed source
2
1 1
1 1
3
3
3
5
5 2
7
7 8