1.5em

Basics of NLP and SSL

Mathematical Foundations of ML 2023

Ido Ben-Shaul [1] [2], Shai Dekel [1]

[1] Tel-Aviv University, Israel

[2] eBay Research

June 10, 2023

Transformers and Typical Tasks

- Based on “Formal Algorithms for Transformers” [PH22]!

Sequence Modelling (DTransformer)

- Given a vocabulary V, let $x_n \in V^*$ for $n \in [N_{\text{data}}]$ be a dataset of sequences "sampled" from dist. P. The goal is to learn an estimate $\hat{P}$ of the distribution $P(\mathbf{x})$.
- In practice, the distribution estimate is often decomposed via the chain rule as $\hat{P}(x) = \hat{P}_\theta(x[1]) \cdot \hat{P}_\theta(x[2] \mid x[1]) \cdots \hat{P}_\theta(x[\ell] \mid x[1:\ell-1])$, where $\boldsymbol{\theta}$ consists of all neural network parameters to be learned.
- The goal is to learn a distribution over a single token $x[t]$ given its preceding tokens $x[1:t-1]$ as context.
- Examples include language modelling, RL policy distillation, or music generation.

Sequence-to-sequence (seq2seq) prediction (EDTransformer)

- Given a vocabulary V and an i.i.d. dataset of sequence pairs $(\boldsymbol{z}_n, \boldsymbol{x}_n) \sim P$, where P is a distribution over $V^* \times V^*$, learn an estimate of the conditional distribution $P(\boldsymbol{x}|\boldsymbol{z})$.
- In practice, the conditional distribution estimate is often decomposed as $\hat{P}(\boldsymbol{x}|\boldsymbol{z}) = \hat{P}_{\boldsymbol{\theta}}(x[1] \,|\, \boldsymbol{z}) \cdot \hat{P}_{\boldsymbol{\theta}}(x[2] \,|\, x[1], \boldsymbol{z}) \cdots \hat{P}_{\boldsymbol{\theta}}(x[\ell] \,|\, \boldsymbol{x}[1 : \ell - 1], \boldsymbol{z})$.
- Examples include translation ($\boldsymbol{z}$ = a sentence in English, $\boldsymbol{x}$ = the same sentence in German), question answering ($\boldsymbol{z}$ = question, $\boldsymbol{x}$ = the corresponding answer), text-to-speech ($\boldsymbol{z}$ = a piece of text, $\boldsymbol{x}$ = a voice recording of someone reading the text).

Classification (ETransformer).

- Given a vocabulary V and a set of classes $[N_{\mathrm{C}}]$, let $(\boldsymbol{x}_n, c_n) \in V^* \times [N_{\mathrm{C}}]$ for $n \in [N_{\mathrm{data}}]$ be an i.i.d. dataset of sequence-class pairs sampled from $P(\boldsymbol{x}, c)$.
- The goal in classification is to learn an estimate of the conditional distribution $P(c|\boldsymbol{x})$.
- Examples include e.g. sentiment classification, spam filtering, toxicity classification.

Tokenization: How Text is Represented

- In NLP, *tokenization* refers to how a piece of text such as "My grandma makes the best apple pie." is represented as a sequence of vocabulary elements (called *tokens*).
- **Character-level tokenization.** One possible choice is to let V be the English alphabet (plus punctuation).

Tokenization: How Text is Represented

- In NLP, *tokenization* refers to how a piece of text such as “My grandma makes the best apple pie.” is represented as a sequence of vocabulary elements (called *tokens*).
- **Character-level tokenization.** One possible choice is to let V be the English alphabet (plus punctuation).
- **Word-level tokenization.** V consists of all English words (plus punctuation).

Tokenization: How Text is Represented

- In NLP, *tokenization* refers to how a piece of text such as “My grandma makes the best apple pie.” is represented as a sequence of vocabulary elements (called *tokens*).
- **Character-level tokenization.** One possible choice is to let V be the English alphabet (plus punctuation).
- **Word-level tokenization.** V consists of all English words (plus punctuation).
- **Subword tokenization.** V is a set of commonly occurring word segments like ‘cious’, ‘ing’, ‘pre’. Common words like ‘is ’ are often a separate token, and single characters are also included in V to ensure all words are expressible. E.g. [SHB15] used in GPT-2 [BMR$^+$20].

Final vocabulary and text representation

- Given a choice of tokenization / vocabulary, each vocabulary element is assigned a unique index in $\{1, 2, \ldots, N_{\mathrm{V}} - 3\}$.
- A number of special tokens are then added to the vocabulary. The number of special tokens varies, and here we will consider three:
  - `mask_token` $:= N_{\mathrm{V}} - 2$, used in masked language modelling;
  - `bos_token` $:= N_{\mathrm{V}} - 1$, used for representing the beginning of sequence;
  - `eos_token` $:= N_{\mathrm{V}}$, used for representing the end of sequence.

  The complete vocabulary has $N_{\mathrm{V}} = |V|$ elements.
- A piece of text is represented as a sequence of indices (called *token IDs*) corresponding to its (sub)words, preceded by `bos_token` and followed by `eos_token`.

Notation

- Let V denote a finite set, called a *vocabulary*, often identified with $[N_V] := \{1, ..., N_V\}$. This could be words or letters, but typically are sub-words, called tokens.
- Let $x \equiv x[1:\ell] \equiv x[1]x[2]\ldots x[\ell] \in V^*$ be a sequence of tokens, e.g. a sentence or a paragraph or a document. Unlike in Python, we use arrays starting from 1, and $x[1:\ell]$ includes $x[\ell]$.
- For a matrix $M \in \mathbb{R}^{d \times d'}$, we write $M[i,:] \in \mathbb{R}^{d'}$ for the ith row and $M[:,j] \in \mathbb{R}^{d}$ for the j-th column.
- The training data may naturally be a collection of (independent) articles, but even then, some may exceed the maximal context length ℓ_{max} transformers can handle. In this case, an article is crudely broken into shorter chunks of length $\leq \ell_{\text{max}}$.

Architectural Components

Token embedding

The token embedding learns to represent each vocabulary element as a vector in $\mathbb{R}^{d_e}$.

Algorithm Token embedding.

Input : $v \in V \cong [N_V]$, a token ID.
Output : $\boldsymbol{e} \in \mathbb{R}^{d_e}$, the vector representation of the token.
Parameter: $\boldsymbol{W_e} \in \mathbb{R}^{d_e \times N_V}$, the token embedding matrix.

return $\boldsymbol{e} = \boldsymbol{W_e}[:, v]$

Positional embedding

The positional embedding(PE) learns to represent a token's position in a sequence as in $\mathbb{R}^{d_e}$. E.g. the position of the 1st token in a sequence is represented by a (learned) vector $\boldsymbol{W_p}[:,1]$, the position of the 2nd token is another (learned) vector $\boldsymbol{W_p}[:,2]$, etc. Learned PE require that the input sequence length is at most some fixed number ℓ_{max}.

Algorithm Positional embedding.

Input : $\ell \in [\ell_{\mathrm{max}}]$, position of a token in the sequence.
Output : $\boldsymbol{e_p} \in \mathbb{R}^{d_e}$, the vector representation of the position.
Parameter: $\boldsymbol{W_p} \in \mathbb{R}^{d_e \times \ell_{\mathrm{max}}}$, the PE matrix.
return $\boldsymbol{e_p} = \boldsymbol{W_p}[:,\ell]$

The PE of a token is added to the token embedding to form a token's initial embedding. For the t-th token of a sequence $\boldsymbol{x}$, the embedding is

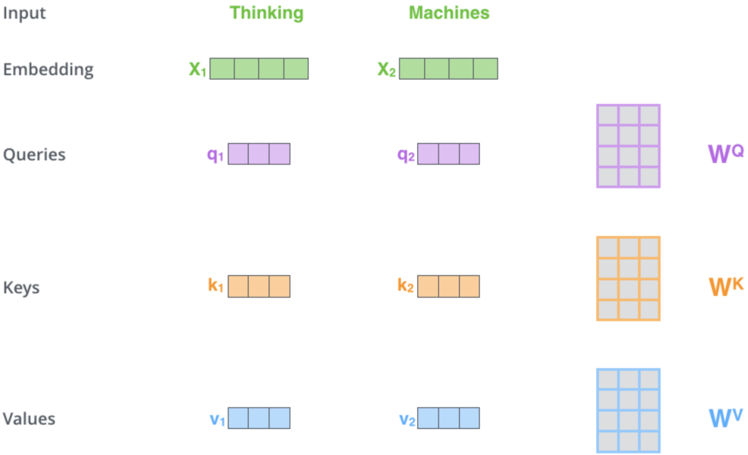
$$\boldsymbol{e} = \boldsymbol{W_e}[:, x[t]] + \boldsymbol{W_p}[:, t]. \tag{1}$$

Attention

Attention is the **main** part of transformers. It enables a neural network to make use of contextual information for predicting the current token.
On a high level:

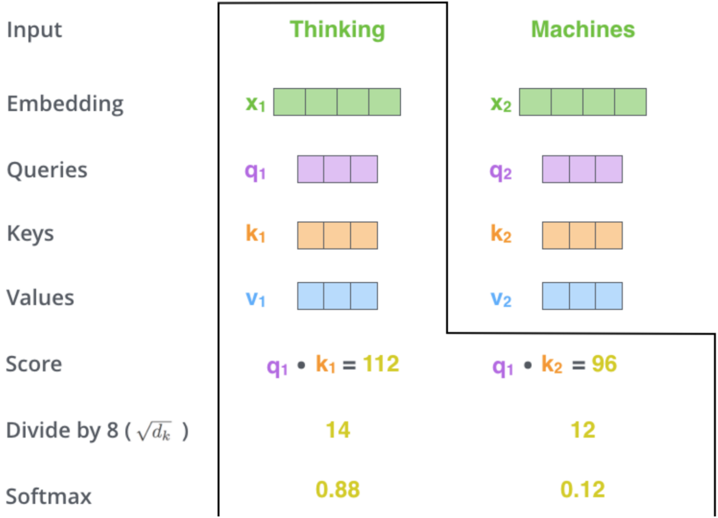
- the token currently being predicted **(destination)** is mapped to a *query* vector $\boldsymbol{q} \in \mathbb{R}^{d_{\mathrm{attn}}}$, and the tokens in the context **(source)** are mapped to *key* vectors $\boldsymbol{k}_t \in \mathbb{R}^{d_{\mathrm{attn}}}$ and *value* vectors $\boldsymbol{v}_t \in \mathbb{R}^{d_{\mathrm{value}}}$.
- The inner products $\boldsymbol{q}^\top \boldsymbol{k}_t$ are interpreted as the degree to which token **(src.)** t is important for predicting the current token **(dst.)** q – they are used to derive a distribution over the context tokens, which is then used to combine the value vectors.

Attention

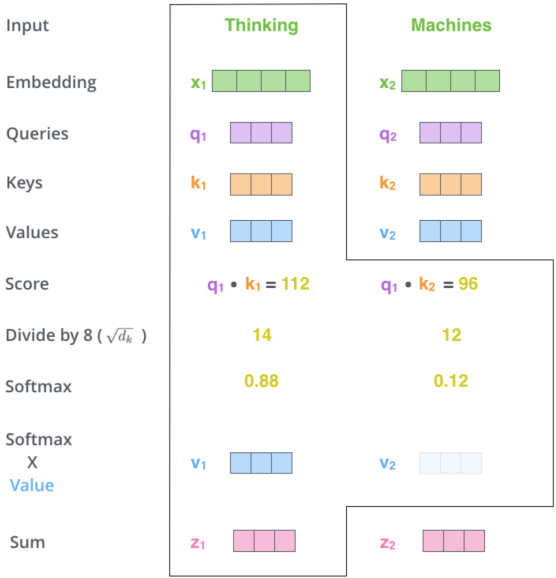


Multiplying x1 by the WQ weight matrix produces q1, the "query" vector associated with that word. We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.

Attention


Input
Thinking
Machines
Embedding
x1
x2
Queries
q1
q2
Keys
k1
k2
Values
v1
v2
Score
q1 • k1 = 112
q1 • k2 = 96
Divide by 8 ($\sqrt{d_k}$)
14
12
Softmax
0.88
0.12

Attention


Input
Thinking
Machines
Embedding
x_1
x_2
Queries
q_1
q_2
Keys
k_1
k_2
Values
v_1
v_2
Score
$q_1 \cdot k_1 = 112$
$q_1 \cdot k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)
14
12
Softmax
0.88
0.12
Softmax X Value
v_1
v_2
Sum
z_1
z_2

Attention

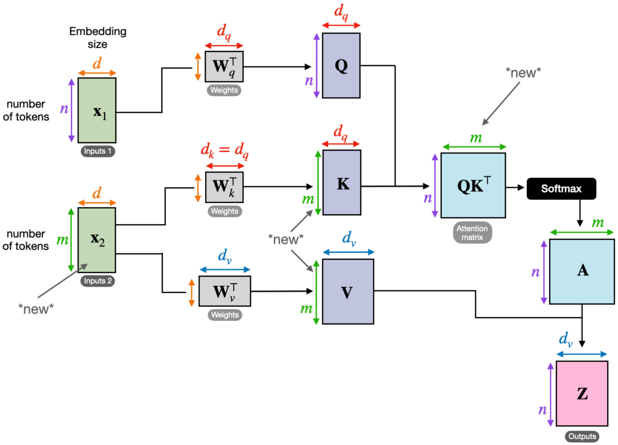


Figure: Source

Basic single-query attention.

Algorithm Basic Single-Query Attention

Input : $\boldsymbol{e} \in \mathbb{R}^{d_{\text{in}}}$, vector representation of the current token.
Input : $\boldsymbol{e}_t \in \mathbb{R}^{d_{\text{in}}}$, vector representations of context tokens $t \in [T]$.
Output : $\tilde{\boldsymbol{v}} \in \mathbb{R}^{d_{\text{out}}}$, vector representation of the token and context combined.
Parameter: $\boldsymbol{W_q}, \boldsymbol{W_k} \in \mathbb{R}^{d_{\text{attn}} \times d_{\text{in}}}$, the query and key linear projections.
Parameter: $\boldsymbol{W_v} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, the value linear projection.

$q \leftarrow W_q e;$

$\forall t: \; k_t \leftarrow W_k e_t;$

$\forall t: \; v_t \leftarrow \boldsymbol{W_v} e_t;$

$$\forall t: \; \alpha_t = \frac{\exp\left(q^\top k_t / \sqrt{d_{\text{attn}}}\right)}{\sum_{u \in [T]} \exp\left(q^\top k_u / \sqrt{d_{\text{attn}}}\right)};$$

return $\tilde{v} = \sum_{t=1}^{T} \alpha_t v_t$

Additional Notation

It will be useful to define the softmax function for matrix arguments, as well as a Mask matrix:

- $$\mathrm{softmax}(\boldsymbol{A})[t_\mathrm{z}, t_\mathrm{x}] := \frac{\exp A[t_\mathrm{z}, t_\mathrm{x}]}{\sum_t \exp A[t, t_\mathrm{x}]}, \tag{2}$$

- $$\mathrm{Mask}[t_\mathrm{z}, t_\mathrm{x}] = \begin{cases} 1 & \text{for bidirectional attention} \\ [[t_\mathrm{z} \le t_\mathrm{x}]] & \text{for unidirectional att.} \end{cases} \tag{3}$$

General Single-Headed Attention

Algorithm $\tilde{V} \leftarrow \texttt{Attention}(\boldsymbol{X}, \boldsymbol{Z} | \boldsymbol{\mathcal{W}}_{\boldsymbol{qkv}}, \text{Mask})$

```
/* Computes a single (masked) self- or cross-attention head.          */
```

Input: $\boldsymbol{X} \in \mathbb{R}^{d_\text{x} \times \ell_\text{x}}, \boldsymbol{Z} \in \mathbb{R}^{d_\text{z} \times \ell_\text{z}}$, vector representations of primary and context sequence.

Output: $\tilde{\boldsymbol{V}} \in \mathbb{R}^{d_\text{out} \times \ell_\text{x}}$, updated representations of tokens in $\boldsymbol{X}$, folding in information from tokens in $\boldsymbol{Z}$.

$\boldsymbol{\mathcal{W}}_{\boldsymbol{qkv}}$ consisting of: $\boldsymbol{W_q} \in \mathbb{R}^{d_\text{attn} \times d_\text{x}}$, $\boldsymbol{W_k} \in \mathbb{R}^{d_\text{attn} \times d_\text{z}}$, $\boldsymbol{W_v} \in \mathbb{R}^{d_\text{out} \times d_\text{z}}$, . Mask$\in \{0,1\}^{\ell_\text{z} \times \ell_\text{x}}$, $\uparrow$(3) ;

$\boldsymbol{Q} \leftarrow \boldsymbol{W_q X}$ $[\![\boldsymbol{Q}\text{uery} \in \mathbb{R}^{d_\text{attn} \times \ell_\text{x}}]\!]$;

$\boldsymbol{K} \leftarrow \boldsymbol{W_k Z}$ $[\![\boldsymbol{K}\text{ey} \in \mathbb{R}^{d_\text{attn} \times \ell_\text{z}}]\!]$;

$\boldsymbol{V} \leftarrow \boldsymbol{W_v Z}$; $[\![\boldsymbol{V}\text{alue} \in \mathbb{R}^{d_\text{out} \times \ell_\text{z}}]\!]$;

$\boldsymbol{S} \leftarrow \boldsymbol{K}^\intercal \boldsymbol{Q}$; $\forall t_\text{z}, t_\text{x}$, if $\neg\text{Mask}[t_\text{z}, t_\text{x}]$ then $S[t_\text{z}, t_\text{x}] \leftarrow -\infty$;

return $\tilde{\boldsymbol{V}} = \boldsymbol{V} \cdot \text{softmax}\left(\boldsymbol{S} / \sqrt{d_\text{attn}}\right)$

Attention Variants

- **Bidirectional / unmasked self-attention.** Given a sequence, attention to each token, treating all tokens in the sequence as the context. Algorithm 5, with $\boldsymbol{Z} = \boldsymbol{X}$ and no masking (Mask $\equiv 1$).
- **Unidirectional / masked self-attention.** Given a sequence, attention to each token, treating all preceding tokens (including itself) as the context. Future tokens are masked out, so this **causal auto-regressive** version can be used for online prediction. $\boldsymbol{Z} = \boldsymbol{X}$ and Mask $[t_z, t_x] := [\![t_z \leq t_x]\!]$. For this Mask, the output $\tilde{\boldsymbol{V}}[:, 1:t]$ only depends on $\boldsymbol{X}[:, 1:t]$, hence can be used to predict $\boldsymbol{X}[:, t+1]$.
- **Cross-attention.** Given two sequences (often in the context of a sequence-to-sequence task), attention to each token of the primary token sequence $\boldsymbol{X}$, treating the second token sequence $\boldsymbol{Z}$ as the context, with Mask $\equiv 1$. While the output $\tilde{\boldsymbol{V}}$ and input sequences $\boldsymbol{X}$ have the same length ℓ_x, the context sequence $\boldsymbol{Z}$ can have different length ℓ_z.

Multi-Headed Attention

Algorithm $\tilde{\boldsymbol{V}} \leftarrow \texttt{MHAttention}(\boldsymbol{X}, \boldsymbol{Z} | \mathcal{W}, \text{Mask})$

```
/* Computes Multi-Head (masked) self- or cross- attention layer.   */
```

Input: $\boldsymbol{X} \in \mathbb{R}^{d_\mathrm{x} \times \ell_\mathrm{x}}$, $\boldsymbol{Z} \in \mathbb{R}^{d_\mathrm{z} \times \ell_\mathrm{z}}$

Output: $\tilde{\boldsymbol{V}} \in \mathbb{R}^{d_\mathrm{out} \times \ell_\mathrm{x}}$, updated representations of tokens in $\boldsymbol{X}$, with information from tokens in $\boldsymbol{Z}$.

for $h \in [H]$, $\mathcal{W}^h_{\boldsymbol{qkv}}$ *consisting of* **do**

$\quad \boldsymbol{W}^h_{\boldsymbol{q}} \in \mathbb{R}^{d_\mathrm{attn} \times d_\mathrm{x}}$, $\boldsymbol{W}^h_{\boldsymbol{k}} \in \mathbb{R}^{d_\mathrm{attn} \times d_\mathrm{z}}$, $\boldsymbol{W}^h_{\boldsymbol{v}} \in \mathbb{R}^{d_\mathrm{mid} \times d_\mathrm{z}}$;

$\boldsymbol{W}_{\boldsymbol{o}} \in \mathbb{R}^{d_\mathrm{out} \times H d_\mathrm{mid}}$.

H, number of attention heads, Mask$\in \{0,1\}^{\ell_\mathrm{z} \times \ell_\mathrm{x}}$

for $h \in [H]$ **do**

$\quad \boldsymbol{Y}^h \leftarrow \texttt{Attention}(\boldsymbol{X}, \boldsymbol{Z} | \mathcal{W}^h_{\boldsymbol{qkv}}, \text{Mask})$

$\boldsymbol{Y} \leftarrow [\boldsymbol{Y}^1; \boldsymbol{Y}^2; \ldots; \boldsymbol{Y}^H]$ **return** $\tilde{\boldsymbol{V}} = \boldsymbol{W}_{\boldsymbol{o}} \boldsymbol{Y}$

return $\tilde{\boldsymbol{V}} = \boldsymbol{V} \cdot \mathrm{softmax}\left(\boldsymbol{S} / \sqrt{d_\mathrm{attn}}\right)$

Layer Norm

Layer normalisation explicitly controls the mean and variance of individual neural network activations; the pseudocode is given in Algorithm 6.

Algorithm $\hat{\boldsymbol{e}} \leftarrow \texttt{layer\_norm}(\boldsymbol{e}|\boldsymbol{\gamma},\boldsymbol{\beta})$

```
/* Normalizes layer activations e.                                        */
```

Input : $\boldsymbol{e} \in \mathbb{R}^{d_e}$, neural network activations.
Output : $\widehat{\boldsymbol{e}} \in \mathbb{R}^{d_e}$, normalized activations.
Parameter: $\boldsymbol{\gamma},\boldsymbol{\beta} \in \mathbb{R}^{d_e}$, element-wise scale and offset.

$\boldsymbol{m} \leftarrow \sum_{i=1}^{d_e} \boldsymbol{e}[i]/d_e$;
$v \leftarrow \sum_{i=1}^{d_e} (\boldsymbol{e}[i] - \boldsymbol{m})^2/d_e$;
return $\widehat{\boldsymbol{e}} = \frac{\boldsymbol{e}-\boldsymbol{m}}{\sqrt{v}} \odot \boldsymbol{\gamma} + \boldsymbol{\beta}$, where $\odot$ denotes element-wise multiplication.

Unembedding

The unembedding learns to convert a vector representation of a token and its context into a distribution over the vocabulary elements

Algorithm Unembedding

Input : $\boldsymbol{e} \in \mathbb{R}^{d_\mathrm{e}}$, a token encoding.
Output : $\boldsymbol{p} \in \Delta(V)$, a probability distribution over the vocabulary.
Parameter: $\boldsymbol{W_u} \in \mathbb{R}^{N_\mathrm{V} \times d_\mathrm{e}}$, the unembedding matrix.
return $\boldsymbol{p} = \mathrm{softmax}(\boldsymbol{W_u e})$

Transformer Architectures

We will go over three example architectures:

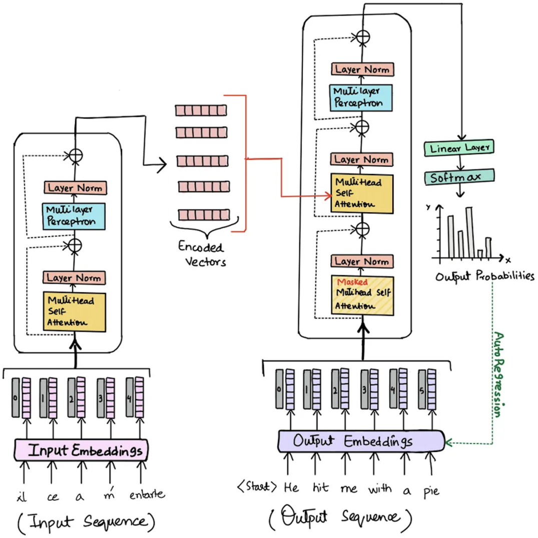
- Encoder-Decoder Transformer (EDT) [VSP+17]
- BERT (Encoder) [DCLT19]
- GPT (Decoder) [BMR+20]

EDT/ Sequence-to-sequence transformer

Intuition:

- First, the context sequence is encoded using bidirectional multi-head attention. The output of this ‘encoder’ part of the network is a vector representation of each context token, taking into account the entire context sequence.
- Second, the primary sequence is encoded. Each token in the primary sequence is allowed to use information from the encoded context sequence, as well as primary sequence tokens that precede it.

EDT - Architecture



EDT - Example: T5, T5-FLAN

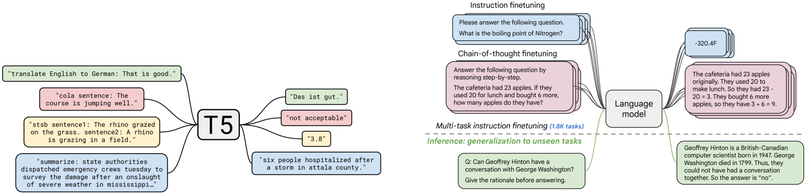


Figure: Left: T5 [RSR+19], Right: FLAN-T5 [CHL+22]

EDT - Architecture

Algorithm 8: $\boldsymbol{P} \leftarrow \texttt{EDTransformer}(\boldsymbol{z}, \boldsymbol{x}|\boldsymbol{\theta})$

`/* Encoder-decoder transformer forward pass */`

Input: $\boldsymbol{z}, \boldsymbol{x} \in V^*$, two sequences of token IDs.

Output: $\boldsymbol{P} \in (0,1)^{N_V \times \mathrm{length}(\boldsymbol{x})}$, where the t-th column of $\boldsymbol{P}$ represents $\hat{P}_{\boldsymbol{\theta}}(x[t+1] | \boldsymbol{x}[1:t], \boldsymbol{z})$.

Hyperparameters: $\ell_{\max}, L_{\mathrm{enc}}, L_{\mathrm{dec}}, H, d_{\mathrm{e}}, d_{\mathrm{mlp}} \in \mathbb{N}$

`/* Encode the context sequence: */`

$\ell_{\mathrm{z}} \leftarrow \mathrm{length}(\boldsymbol{z})$

for $t \in [\ell_{\mathrm{z}}]$: $\boldsymbol{e}_t \leftarrow \boldsymbol{W_e}[:, z[t]] + \boldsymbol{W_p}[:, t]$

$\boldsymbol{Z} \leftarrow [\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots \boldsymbol{e}_{\ell_{\mathrm{z}}}]$

for $l = 1, 2, \ldots, L_{\mathrm{enc}}$ **do**

  $\boldsymbol{Z} \leftarrow \boldsymbol{Z} + \texttt{MHAttention}(\boldsymbol{Z} | \boldsymbol{\mathcal{W}}_l^{\mathrm{enc}}, \mathrm{Mask} \equiv 1)$

  for $t \in [\ell_{\mathrm{z}}]$: $\boldsymbol{Z}[:, t] \leftarrow \texttt{layer\_norm}(\boldsymbol{Z}[:, t] | \boldsymbol{\gamma}_l^1, \boldsymbol{\beta}_l^1)$

  $\boldsymbol{Z} \leftarrow \boldsymbol{Z} + \boldsymbol{W}_{\mathrm{mlp2}}^l \mathtt{ReLU}(\boldsymbol{W}_{\mathrm{mlp1}}^l \boldsymbol{Z} + \boldsymbol{b}_{\mathrm{mlp1}}^l \boldsymbol{1}^\intercal) + \boldsymbol{b}_{\mathrm{mlp2}}^l \boldsymbol{1}^\intercal$

  for $t \in [\ell_{\mathrm{z}}]$: $\boldsymbol{Z}[:, t] \leftarrow \texttt{layer\_norm}(\boldsymbol{Z}[:, t] | \boldsymbol{\gamma}_l^2, \boldsymbol{\beta}_l^2)$

end

EDT - Architecture

/* Decode the primary sequence, conditioning on the context: */

$\ell_{\mathrm{x}} \leftarrow \mathrm{length}(x)$

for $t \in [\ell_{\mathrm{x}}]$: $e_t \leftarrow W_e[:, x[t]] + W_p[:, t]$

$X \leftarrow [e_1, e_2, \ldots e_{\ell_{\mathrm{x}}}]$

for $i = 1, 2, \ldots, L_{\mathrm{dec}}$ **do**

$\quad X \leftarrow X + \mathtt{MHAttention}(X \mid \mathcal{W}_l^{\mathrm{dec}}, \mathrm{Mask}[t, t'] \equiv [[t \leq t']])$

$\quad$ for $t \in [\ell_{\mathrm{x}}]$: $X[:, t] \leftarrow \mathtt{layer\_norm}(X[:, t] \mid \gamma_l^3, \beta_l^3)$

$\quad X \leftarrow X + \mathtt{MHAttention}(X, Z \mid \mathcal{W}_l^{\mathrm{e/d}}, \mathrm{Mask} \equiv 1)$

$\quad$ for $t \in [\ell_{\mathrm{x}}]$: $X[:, t] \leftarrow \mathtt{layer\_norm}(X[:, t] \mid \gamma_l^4, \beta_l^4)$

$\quad X \leftarrow X + W_{\mathrm{mlp4}}^l \mathtt{ReLU}(W_{\mathrm{mlp3}}^l X + b_{\mathrm{mlp3}}^l \mathbf{1}^\intercal) + b_{\mathrm{mlp4}}^l \mathbf{1}^\intercal$

$\quad$ for $t \in [\ell_{\mathrm{x}}]$: $X[:, t] \leftarrow \mathtt{layer\_norm}(X[:, t] \mid \gamma_l^5, \beta_l^5)$

end

/* Derive conditional probabilities and return: */

return $P = \mathrm{softmax}(W_u X)$

EDT - Training

Algorithm 11: $\hat{\theta} \leftarrow \texttt{EDTraining}(z_{1:N_{\text{data}}}, x_{1:N_{\text{data}}}, \theta)$

```
/* Training a seq2seq model */
```

Input: $\{(z_n, x_n)\}_{n=1}^{N_{\text{data}}}$, a dataset of sequence pairs.
Input: θ, initial transformer parameters.
Output: $\hat{\theta}$, the trained parameters.
Hyperparameters: $N_{\text{epochs}} \in \mathbb{N}$, $\eta \in (0, \infty)$

for $i = 1, 2, \ldots, N_{\text{epochs}}$ **do**
  for $n = 1, 2, \ldots N_{\text{data}}$ **do**
    $\ell \leftarrow \text{length}(x_n)$
    $P(\theta) \leftarrow \texttt{EDTransformer}(z_n, x_n | \theta)$
    $\text{loss}(\theta) = -\sum_{t=1}^{\ell-1} \log P(\theta)[x_n[t+1], t]$
    $\theta \leftarrow \theta - \eta \cdot \nabla \text{loss}(\theta)$
  end
end
return $\hat{\theta} = \theta$

EDT: Inference

Algorithm 15: $\hat{x} \leftarrow \mathtt{EDInference}(z, \hat{\theta})$

```
/* Using a trained seq2seq model for prediction. */
```

Input: A seq2seq transformer and trained parameters $\hat{\theta}$ of the transformer.

Input: $z \in V^*$, input sequence, e.g. a sentence in English.

Output: $\hat{x} \in V^*$, output sequence, e.g. the sentence in German.

Hyperparameters: $\tau \in (0, \infty)$

$\hat{x} \leftarrow [\mathtt{bos\_token}]$

$y \leftarrow 0$

while $y \neq \mathit{eos\_token}$ **do**

- $P \leftarrow \mathtt{EDTransformer}(z, \hat{x} \mid \hat{\theta})$
- $p \leftarrow P[:, \text{length}(\hat{x})]$
- sample a token y from $q \propto p^{1/\tau}$
- $\hat{x} \leftarrow [\hat{x}, y]$

end

return $\hat{x}$

Encoder-only transformer: BERT

Intuition: BERT is a bidirectional transformer trained on the task of masked language modelling. Given a piece of text with some tokens masked out, the goal is to correctly recover the masked-out tokens. The original use of BERT was to learn generally useful text representations, which could then be adapted for various downstream NLP tasks. The masking is not performed via the Mask parameter but differently: During training each input token is replaced with probability p_{mask} by a dummy token `mask_token`, and evaluation is based on the reconstruction probability of these knocked-out tokens.

Encoder-only transformer: BERT, architecture

Algorithm 9: $P \leftarrow \mathtt{ETransformer}(x|\theta)$

`/* BERT, an encoder-only transformer, forward pass */`

Input: $x \in V^*$, a sequence of token IDs.

Output: $P \in (0,1)^{N_V \times \text{length}(x)}$, where each column of P is a distribution over the vocabulary.

Hyperparameters: $\ell_{\max}, L, H, d_e, d_{\text{mlp}}, d_f \in \mathbb{N}$

Parameters: θ includes all of the following parameters:

$W_e \in \mathbb{R}^{d_e \times N_V}$, $W_p \in \mathbb{R}^{d_e \times \ell_{\max}}$, the token and positional embedding matrices.

For $l \in [L]$:

- $\mathcal{W}_l$, multi-head attention parameters for layer l, see (4),
- $\gamma_l^1, \beta_l^1, \gamma_l^2, \beta_l^2 \in \mathbb{R}^{d_e}$, two sets of layer-norm parameters,
- $W_{\text{mlp1}}^l \in \mathbb{R}^{d_{\text{mlp}} \times d_e}$, $b_{\text{mlp1}}^l \in \mathbb{R}^{d_{\text{mlp}}}$, $W_{\text{mlp2}}^l \in \mathbb{R}^{d_e \times d_{\text{mlp}}}$, $b_{\text{mlp2}}^l \in \mathbb{R}^{d_e}$, MLP parameters.

$W_f \in \mathbb{R}^{d_f \times d_e}$, $b_f \in \mathbb{R}^{d_f}$, $\gamma, \beta \in \mathbb{R}^{d_f}$, the final linear projection and layer-norm parameters.

$W_u \in \mathbb{R}^{N_V \times d_e}$, the unembedding matrix.

$\ell \leftarrow \text{length}(x)$

for $t \in [\ell]$: $e_t \leftarrow W_e[:, x[t]] + W_p[:, t]$

$X \leftarrow [e_1, e_2, \ldots e_\ell]$

for $l = 1, 2, \ldots, L$ **do**

    $X \leftarrow X + \mathtt{MHAttention}(X \,|\, \mathcal{W}_l, \text{Mask} \equiv 1)$

    for $t \in [\ell]$: $X[:, t] \leftarrow \mathtt{layer\_norm}(X[:, t] \,|\, \gamma_l^1, \beta_l^1)$

    $X \leftarrow X + W_{\text{mlp2}}^l \mathtt{GELU}(W_{\text{mlp1}}^l X + b_{\text{mlp1}}^l \mathbf{1}^\intercal) + b_{\text{mlp2}}^l \mathbf{1}^\intercal$

    for $t \in [\ell]$: $X[:, t] \leftarrow \mathtt{layer\_norm}(X[:, t] \,|\, \gamma_l^2, \beta_l^2)$

end

$X \leftarrow \mathtt{GELU}(W_f X + b_f \mathbf{1}^\intercal)$

for $t \in [\ell]$: $X[:, t] \leftarrow \mathtt{layer\_norm}(X[:, t] \,|\, \gamma, \beta)$

return $P = \text{softmax}(W_u X)$

Encoder-only: Training

Algorithm 12: $\hat{\theta} \leftarrow \texttt{ETraining}(x_{1:N_{\text{data}}}, \theta)$

```
/* Training by masked language
   modelling                                  */
```

Input: $\{x_n\}_{n=1}^{N_{\text{data}}}$, a dataset of sequences.
Input: θ, initial encoder-only transformer parameters.
Output: $\hat{\theta}$, the trained parameters.
Hyperparameters: $N_{\text{epochs}} \in \mathbb{N}$, $\eta \in (0, \infty)$, $p_{\text{mask}} \in (0, 1)$

for $i = 1, 2, \ldots, N_{\text{epochs}}$ **do**
  for $n = 1, 2, \ldots, N_{\text{data}}$ **do**
    $\ell \leftarrow \text{length}(x_n)$
    for $t = 1, 2, \ldots, \ell$ **do**
      $\tilde{x}_n[t] \leftarrow \texttt{mask\_token}$ or $x_n[t]$ randomly with probability p_{mask} or $1 - p_{\text{mask}}$
    end
    $\tilde{T} \leftarrow \{t \in [\ell] : \tilde{x}_n[t] = \texttt{mask\_token}\}$
    $P(\theta) \leftarrow \texttt{ETransformer}(\tilde{x}_n \mid \theta)$
    $\text{loss}(\theta) = -\sum_{t \in \tilde{T}} \log P(\theta)[x_n[t], t]$
    $\theta \leftarrow \theta - \eta \cdot \nabla \text{loss}(\theta)$
  end
end

BERT: Pretraining and Fine Tuning

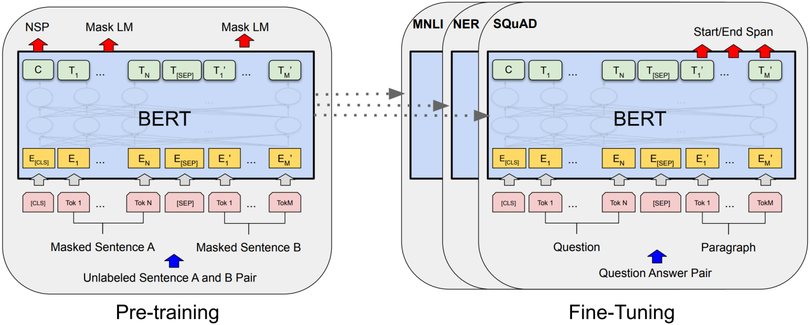


Figure: [DCLT19]

Classification using Encoder Models

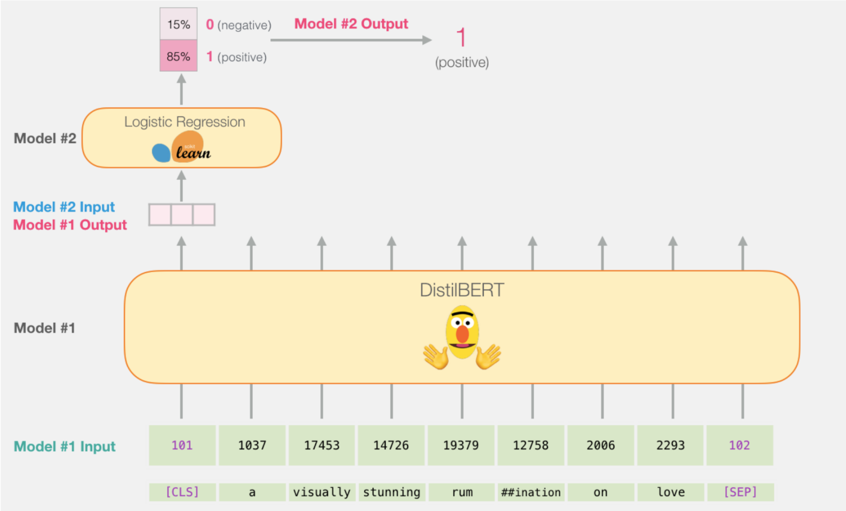


Figure: Source

Encoder-only: Clustering in intermediate layers

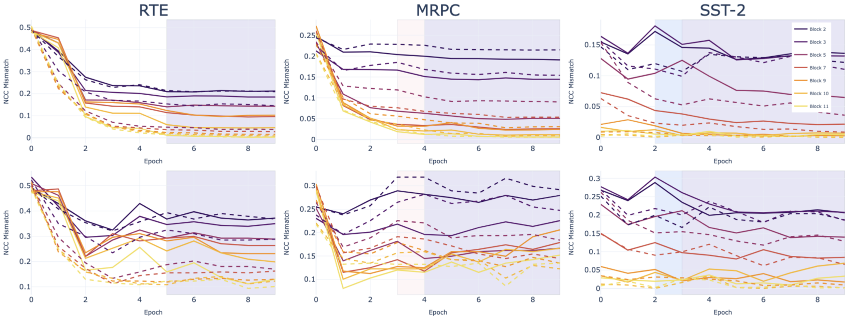
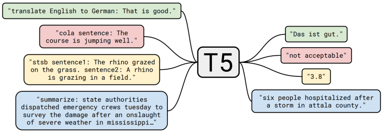


Figure: NCC mismatch for Sequence-Classification datasets: **RTE**, **MRPC** and **CoLA**, using both vanilla (**solid**) and SVSL(**dashed**) losses. **Top:** train NCC mismatch, **Bottom:** test NCC mismatch. We show only a subset of the transformer blocks for clearness. The shaded pink background shows the TPT for the vanilla loss experiment, and the blue for the SVSL. The background is shaded purple at epochs when both experiments are in the TPT.

Encoder-only: Clustering in intermediate layers



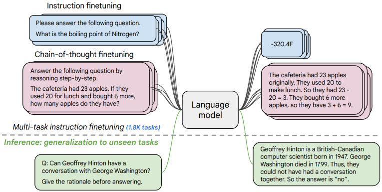


Figure: Left: Pretrained, Right: Finetuned [XQP+22]

Decoder-only transformers: GPT

GPT-2 and GPT-3 are large language models developed by OpenAI.
They all have similar architectures and are trained by autoregressive language modelling: Given an incomplete sentence or paragraph, the goal is to predict the next token.
The main difference from BERT is that GPT use unidirectional attention instead of bidirectional attention.
GPT-3 is identical except larger, and replaces dense attention in Line 6 by sparse attention, i.e. each token only uses a subset of the full context.

Decoder-only transformers: GPT - architecture

Algorithm 10: $\boldsymbol{P} \leftarrow \mathtt{DTransformer}(\boldsymbol{x}|\boldsymbol{\theta})$

`/* GPT, a decoder-only transformer, forward pass */`

Input: $x \in V^*$, a sequence of token IDs.

Output: $\boldsymbol{P} \in (0,1)^{N_V \times \mathrm{length}(x)}$, where the t-th column of $\boldsymbol{P}$ represents $\hat{P}_{\boldsymbol{\theta}}(x[t+1]\,|\,x[1:t])$.

Hyperparameters: $\ell_{\mathrm{max}}, L, H, d_{\mathrm{e}}, d_{\mathrm{mlp}} \in \mathbb{N}$

Parameters: $\boldsymbol{\theta}$ includes all of the following parameters:

$\boldsymbol{W_e} \in \mathbb{R}^{d_{\mathrm{e}} \times N_V}$, $\boldsymbol{W_p} \in \mathbb{R}^{d_{\mathrm{e}} \times \ell_{\mathrm{max}}}$, the token and positional embedding matrices.

For $l \in [L]$:

| $\mathcal{W}_l$, multi-head attention parameters for layer l, see (4),

| $\boldsymbol{\gamma}_l^1, \boldsymbol{\beta}_l^1, \boldsymbol{\gamma}_l^2, \boldsymbol{\beta}_l^2 \in \mathbb{R}^{d_{\mathrm{e}}}$, two sets of layer-norm parameters,

| $\boldsymbol{W}^l_{\mathrm{mlp1}} \in \mathbb{R}^{d_{\mathrm{mlp}} \times d_{\mathrm{e}}}$, $\boldsymbol{b}^l_{\mathrm{mlp1}} \in \mathbb{R}^{d_{\mathrm{mlp}}}$, $\boldsymbol{W}^l_{\mathrm{mlp2}} \in \mathbb{R}^{d_{\mathrm{e}} \times d_{\mathrm{mlp}}}$, $\boldsymbol{b}^l_{\mathrm{mlp2}} \in \mathbb{R}^{d_{\mathrm{e}}}$, MLP parameters.

$\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^{d_{\mathrm{e}}}$, final layer-norm parameters.

$\boldsymbol{W_u} \in \mathbb{R}^{N_V \times d_{\mathrm{e}}}$, the unembedding matrix.

$\ell \leftarrow \mathrm{length}(\boldsymbol{x})$

for $t \in [\ell]$: $\boldsymbol{e}_t \leftarrow \boldsymbol{W_e}[:, x[t]] + \boldsymbol{W_p}[:, t]$

$\boldsymbol{X} \leftarrow [\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots \boldsymbol{e}_\ell]$

for $l = 1, 2, \ldots, L$ **do**

  for $t \in [\ell]$: $\tilde{\boldsymbol{X}}[:, t] \leftarrow \mathtt{layer\_norm}(\boldsymbol{X}[:, t]\,|\,\boldsymbol{\gamma}_l^1, \boldsymbol{\beta}_l^1)$

  $\boldsymbol{X} \leftarrow \boldsymbol{X} + \mathtt{MHAttention}(\tilde{\boldsymbol{X}}\,|\,\mathcal{W}_l, \mathrm{Mask}[t, t'] = [[t \leq t']])$

  for $t \in [\ell]$: $\tilde{\boldsymbol{X}}[:, t] \leftarrow \mathtt{layer\_norm}(\boldsymbol{X}[:, t]\,|\,\boldsymbol{\gamma}_l^2, \boldsymbol{\beta}_l^2)$

  $\boldsymbol{X} \leftarrow \boldsymbol{X} + \boldsymbol{W}^l_{\mathrm{mlp2}}\mathtt{GELU}(\boldsymbol{W}^l_{\mathrm{mlp1}}\tilde{\boldsymbol{X}} + \boldsymbol{b}^l_{\mathrm{mlp1}}\boldsymbol{1}^\intercal) + \boldsymbol{b}^l_{\mathrm{mlp2}}\boldsymbol{1}^\intercal$

end

for $t \in [\ell]$: $\boldsymbol{X}[:, t] \leftarrow \mathtt{layer\_norm}(\boldsymbol{X}[:, t]\,|\,\boldsymbol{\gamma}, \boldsymbol{\beta})$

return $\boldsymbol{P} = \mathrm{softmax}(\boldsymbol{W_u}\boldsymbol{X})$

Decoder-only transformers: GPT - training

Algorithm 13: $\hat{\theta} \leftarrow \texttt{DTraining}(x_{1:N_{\text{data}}}, \theta)$

```
/* Training next token prediction */
```

Input: $\{x_n\}_{n=1}^{N_{\text{data}}}$, a dataset of sequences.

Input: θ, initial decoder-only transformer parameters.

Output: $\hat{\theta}$, the trained parameters.

Hyperparameters: $N_{\text{epochs}} \in \mathbb{N}$, $\eta \in (0, \infty)$

for $i = 1, 2, \ldots, N_{\text{epochs}}$ **do**

  for $n = 1, 2, \ldots N_{\text{data}}$ **do**

    $\ell \leftarrow \text{length}(x_n)$

    $P(\theta) \leftarrow \texttt{DTransformer}(x_n \,|\, \theta)$

    $\text{loss}(\theta) = -\sum_{t=1}^{\ell-1} \log P(\theta)[x_n[t+1], t]$

    $\theta \leftarrow \theta - \eta \cdot \nabla \text{loss}(\theta)$

  end

end

return $\hat{\theta} = \theta$

Decoder-only transformers: Inference

Algorithm 14: $y \leftarrow \texttt{DInference}(x, \hat{\theta})$

/* Prompting a trained model and using it for prediction. */

Input: Trained transformer parameters $\hat{\theta}$.

Input: $x \in V^*$, a prompt.

Output: $y \in V^*$, the transformer's continuation of the prompt.

Hyperparameters: $\ell_{\mathrm{gen}} \in \mathbb{N}$, $\tau \in (0, \infty)$

$\ell \leftarrow \mathrm{length}(x)$

for $i = 1, 2, \ldots \ell_{\mathrm{gen}}$ **do**

  $P \leftarrow \texttt{DTransformer}(x \mid \hat{\theta})$

  $p \leftarrow P[:, \ell + i - 1]$

  sample a token y from $q \propto p^{1/\tau}$

  $x \leftarrow [x, y]$

end

return $y = x[\ell + 1 : \ell + \ell_{\mathrm{gen}}]$

Vision Transformer

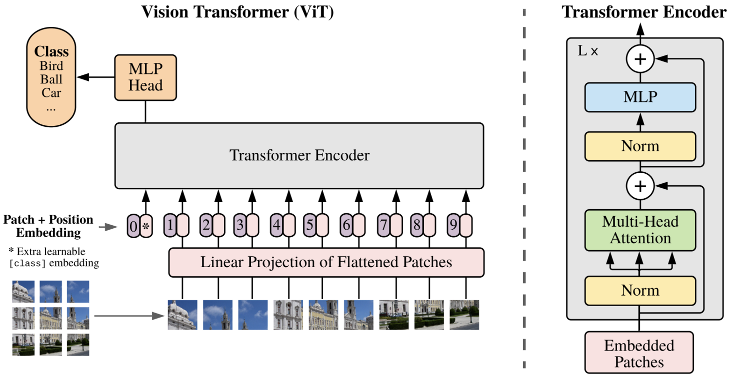


Figure: [DBK+21]

CLIP

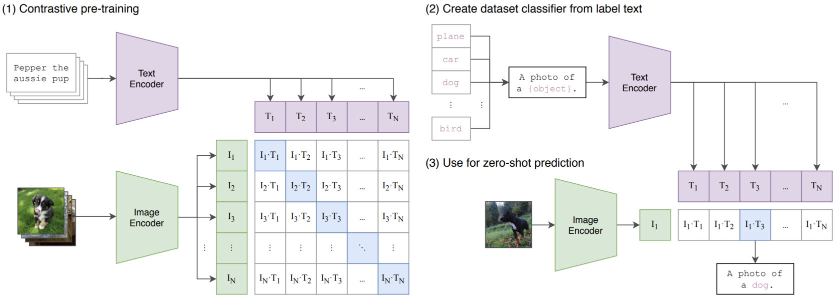


Figure: [RKH+21]

CLIP embeddings

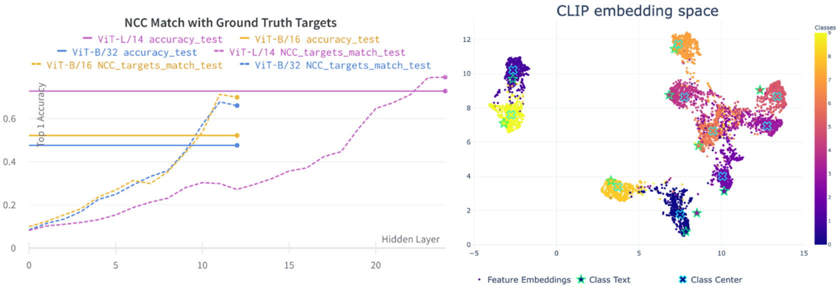


Figure: [RKH+21], Blogpost

Zero/One/Few Shot Learning

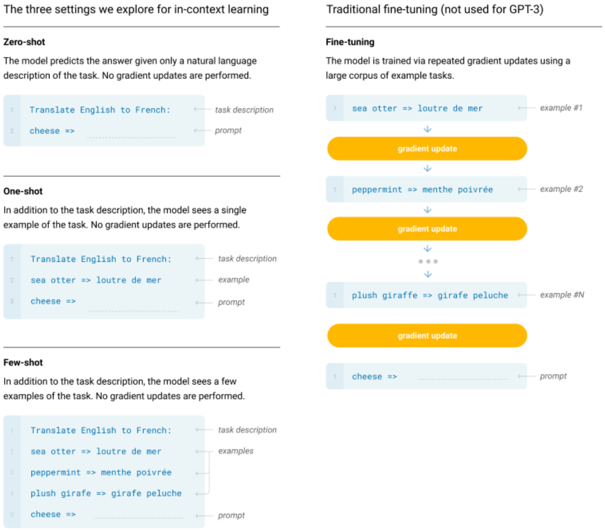


Figure: [BMR+20]

Chain-of-Thought prompting

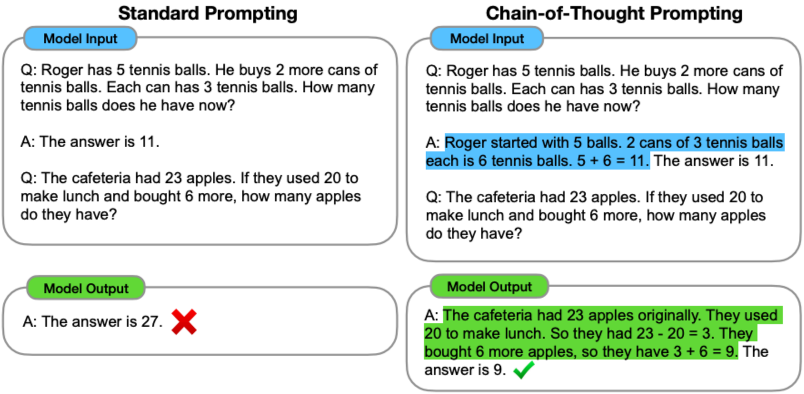


Figure: [WWS+22]

Retrieval Augmented Generation

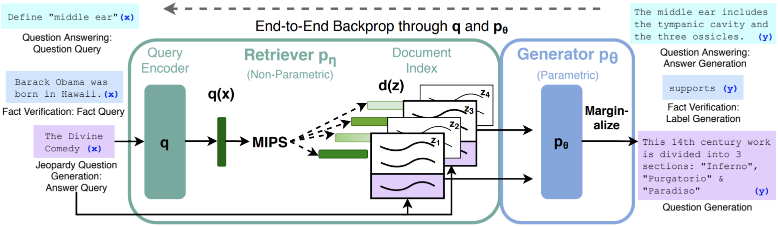
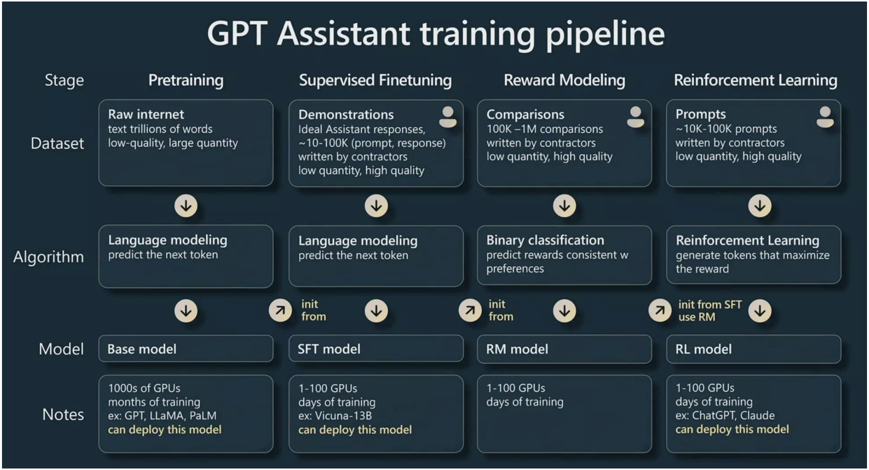


Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder* + *Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x, we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i. For final prediction y, we treat z as a latent variable and marginalize over seq2seq predictions given different documents.

State of GPT



GPT Assistant training pipeline
Stage
Pretraining
Supervised Finetuning
Reward Modeling
Reinforcement Learning
Dataset
Raw internet
text trillions of words
low-quality, large quantity
Demonstrations
Ideal Assistant responses,
~10-100K (prompt, response)
written by contractors
low quantity, high quality
Comparisons
100K –1M comparisons
written by contractors
low quantity, high quality
Prompts
~10K-100K prompts
written by contractors
low quantity, high quality
Algorithm
Language modeling
predict the next token
Language modeling
predict the next token
Binary classification
predict rewards consistent w preferences
Reinforcement Learning
generate tokens that maximize the reward
init from
init from
init from SFT
use RM
Model
Base model
SFT model
RM model
RL model
Notes
1000s of GPUs
months of training
ex: GPT, LLaMA, PaLM
can deploy this model
1-100 GPUs
days of training
ex: Vicuna-13B
can deploy this model
1-100 GPUs
days of training
1-100 GPUs
days of training
ex: ChatGPT, Claude
can deploy this model

LIMA

You are a scientist who just invented a time machine. Where do you travel first?

As a scientist, I would travel back in time to the year 1969 and watch the Apollo 11 mission land on Moon.

The Apollo 11 Mission was the first space flight that landed humans on the Moon. It was an incredibly important moment in history as it marked mankind's first step towards exploring other planets. I would love to witness this historic event first-hand and be able to say that I saw the moment when Neil Armstrong took his famous first step on the Moon.

You are a scientist who just invented a time machine. Where do you travel first?

I would travel to Ancient Egypt.

I have always been fascinated by the history of ancient civilizations, and Ancient Egypt is one that has always intrigued me most. Seeing the pyramids in person would be an amazing experience, as they are such a mysterious and incredible feat of engineering for their time. Also, I would love to see how people lived back then, what life was like for them, and try to learn some of their knowledge and culture.

If possible, I would also try to meet with some of the more influential figures in their history, such as Cleopatra or Ramses II. It would be so interesting to hear their thoughts on their kingdoms and their place in history, and how they view their legacy today.

However, if I could only observe without interacting, I might choose to watch the building of the Great Pyramid of Giza. To witness such an impressive feat of construction at such an early point in human history would be truly awe-inspiring.

Figure: LIMA: Less Is More for Alignment [ZLX+23]

Self-Supervised Learning

Self-Supervised Learning (SSL)

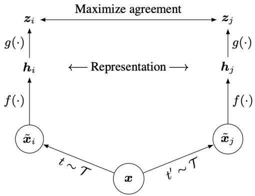
SSL Setting

- How can we learn representations without labeling?

SSL Setting

- How can we learn representations without labeling?
- **Trick:** Define a proxy task to learn representations
- **Objective:** Learn meaningful representations without labels.

SimCLR



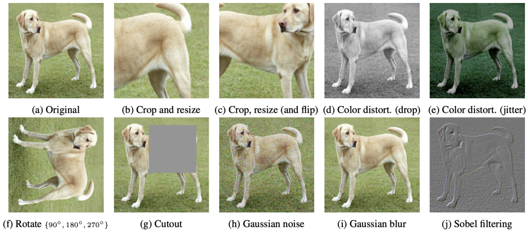


Figure: SimCLR: [CKNH20]

$$L(f) = -\frac{1}{B}\sum_{i=1}^{B} \underbrace{(\mathrm{sim}(Z_i, Z_i')/\tau)}_{\text{Invariance}} \underbrace{- \log \sum_{j\neq i} \exp(\mathrm{sim}(Z_i, Z_j')/\tau)}_{\text{Regularization}},$$

VICReg

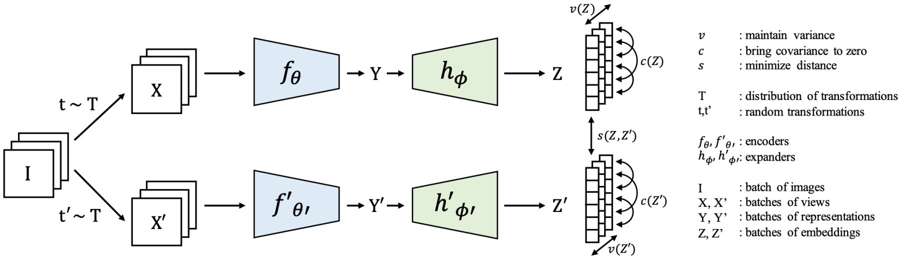


Figure: VICReg: [BPL22]

$$L(f) = \underbrace{\lambda s(Z, Z')}_{\text{Invariance}} + \underbrace{\mu[v(Z) + v(Z')] + \nu[c(Z) + c(Z')]}_{\text{Regularization}}, \tag{4}$$

Understanding SSL

Clustering

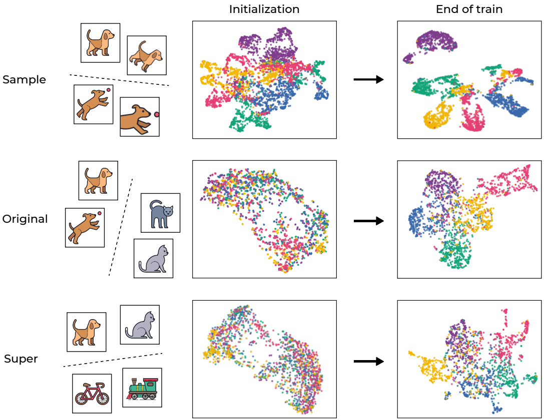


Figure: **SSL training induced semantic clustering.** UMAP of SSL representations in different hierarchies. **(top)** Augmentations of five different samples, each sample colored distinctly. **(middle)** Samples from five different classes within the standard CIFAR-100 dataset. **(bottom)** Samples from five different superclasses within the dataset. [BSSZG$^+$23]

Hierarchy

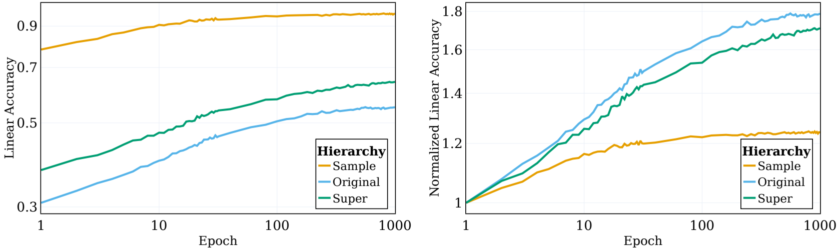


Figure: **SSL algorithms cluster the data with respect to semantic targets.** The linear test accuracy rates, (left) non-normalized, (right) normalized by their values at initialization. All experiments are conducted on CIFAR-100 with VICReg training

NCC

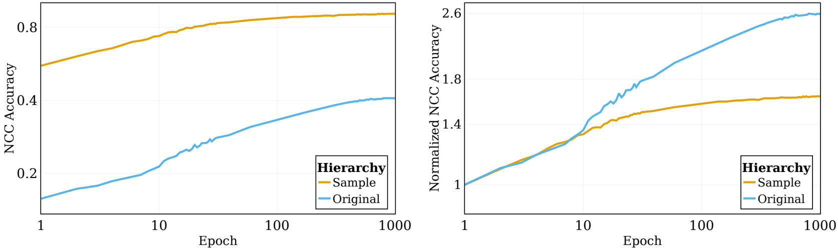


Figure: **SSL algorithms cluster the data with respect to semantic targets.** The normalized NCC train accuracy, computed by dividing the accuracy values by their value at initialization.

Intermediate Layers

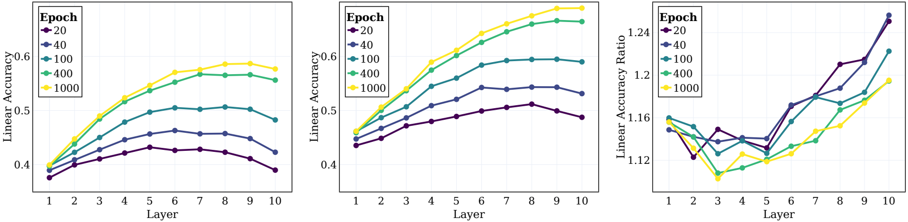
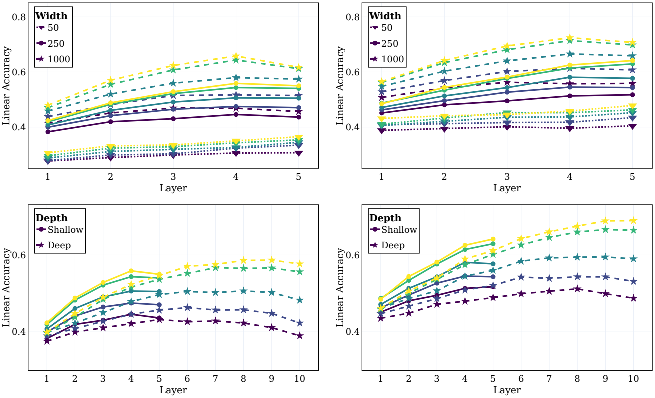


Figure: **SSL efficiently learns semantic classes throughout intermediate layers.** The linear test accuracy of different layers of the model at various epochs **(left)** With respect to the 100 original classes. **(middle)** With respect to the 20 superclasses. **(right)** The ratio between the superclass and the original classes. All experiments are conducted on CIFAR-100 with VICReg training.

Architecture Affects

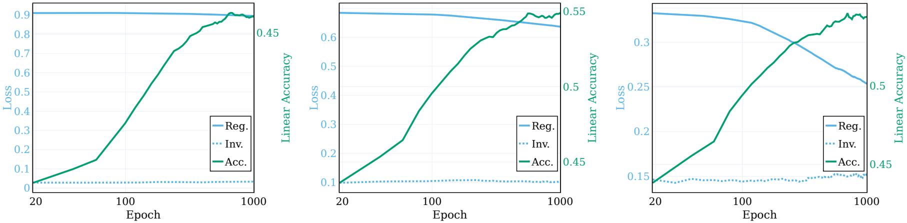


Original classes

Superclasses

Figure: **The influence of width and depth on learning semantic classes at intermediate layers. (top)** Linear test accuracy at different epochs for neural networks of varying widths. **(bottom)** Linear test accuracy of neural networks with different depths. **(left)** The performance is measured in relation to the original classes. **(right)** The performance with respect to the superclasses.

Loss Terms



$\mu = 5$ $\mu = 25$ $\mu = 100$

Figure: **The role of the regularization term in SSL training.** Each plot depicts the regularization and invariance losses, along with the linear test accuracy, throughout the training process of VICReg with $\mu = 5, 25, 100$ respectively.

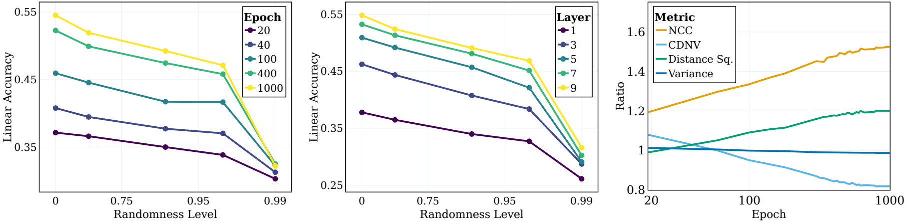


Figure: **SSL continuously learns semantic targets over random ones. (left)** The linear test accuracy for targets with varying levels of randomness from the last layers at different epochs. **(middle)** The linear test accuracy for targets with varying levels of randomness for the trained model. **(right)** The ratios between non-random and random targets for various clustering metrics. All experiments are conducted on CIFAR-100 with VICReg training.

Different Architectures

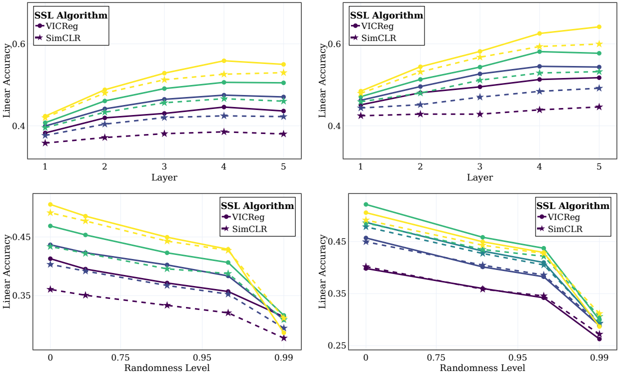


Figure: **SimCLR and VICReg have similar performance. (top)** Linear test accuracy in different training epochs, as a function of the intermediate layer, for original classes and superclasses, from left to right resp. **(bottom) (left)** Linear test accuracy in different training epochs (from dark to light) with respect to different randomness levels. **(right)** Linear test accuracy in different intermediate layers, at the end of training with respect to different randomness levels.

Implicit Bias of Backbone

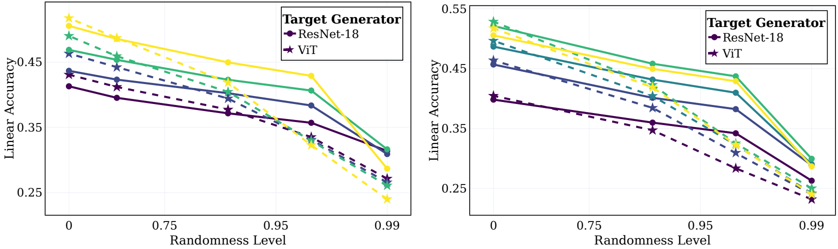


Figure: **The implicit bias of the backbone architecture on the learned representations. (left)** Linear test accuracy of an SSL-trained RES-5-250 network for extracting ResNet-18 and ViT random target functions with varying degrees of randomness (x-axis) at different epochs (color-coded from dark to bright). **(right)** Linear test accuracy of an SSL-trained RES-5-250 network for extracting ResNet-18 and ViT random target functions with varying degrees of randomness (x-axis) at different intermediate layers (color-coded from dark to bright).

Bibliography

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei.
Language models are few-shot learners.
In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

Adrien Bardes, Jean Ponce, and Yann LeCun.
VICReg: Variance-invariance-covariance regularization for self-supervised learning.
In *International Conference on Learning Representations*, 2022.

Bibliography

Ido Ben-Shaul, Ravid Shwartz-Ziv, Tomer Galanti, Shai Dekel, and Yann LeCun.
Reverse engineering self-supervised learning, 2023.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei.
Scaling Instruction-Finetuned Language Models.
arXiv e-prints, page arXiv:2210.11416, October 2022.

Bibliography

- Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton.
  A simple framework for contrastive learning of visual representations.
  In *International Conference on Machine Learning (ICML)*, 2020.

- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby.
  An image is worth 16x16 words: Transformers for image recognition at scale.
  In *International Conference on Learning Representations*, 2021.

Bibliography

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova.
BERT: Pre-training of deep bidirectional transformers for language understanding.
In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Mary Phuong and Marcus Hutter.
Formal Algorithms for Transformers.
arXiv e-prints, page arXiv:2207.09238, July 2022.

Bibliography

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever.
Learning Transferable Visual Models From Natural Language Supervision.
arXiv e-prints, page arXiv:2103.00020, February 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu.
Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.
arXiv e-prints, page arXiv:1910.10683, October 2019.

Rico Sennrich, Barry Haddow, and Alexandra Birch.
Neural Machine Translation of Rare Words with Subword Units.
arXiv e-prints, page arXiv:1508.07909, August 2015.

Bibliography

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin.
Attention is all you need.
In *Proceedings of the 30th International Conference on Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou.
Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.
arXiv e-prints, page arXiv:2201.11903, January 2022.

Bibliography

Shuo Xie, Jiahao Qiu, Ankita Pasad, Li Du, Qing Qu, and Hongyuan Mei.
Hidden State Variability of Pretrained Language Models Can Guide Computation Reduction for Transfer Learning.
arXiv e-prints, page arXiv:2210.10041, October 2022.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy.
LIMA: Less Is More for Alignment.
arXiv e-prints, page arXiv:2305.11206, May 2023.