Large Language Models (Model Adaptation)

Tim Metzler

Department of Computer Science

HBRS / ST.A.

Transformer Decoder

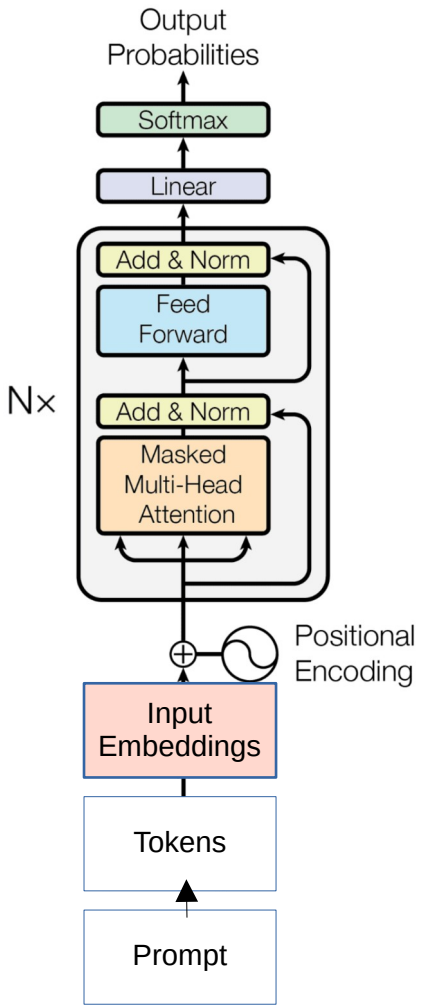
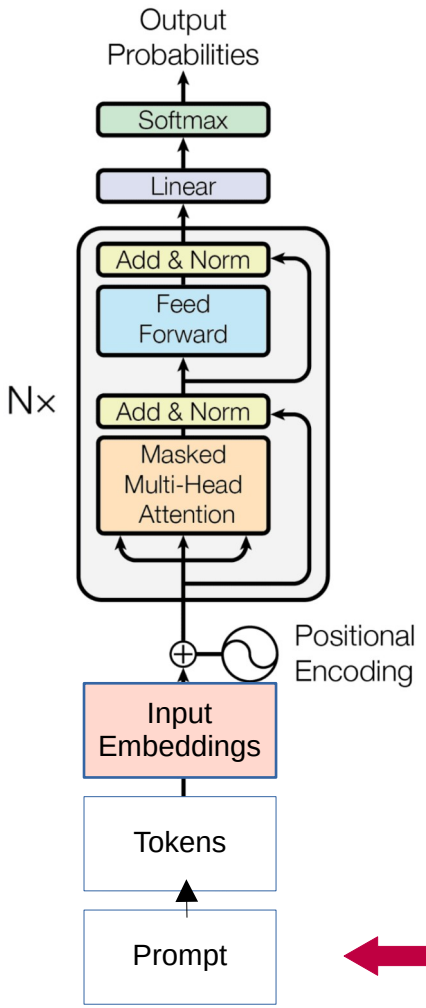


Fig. 1: Transformer Decoder (adapted from: Attention is all you need. Vaswani et al. 2017)

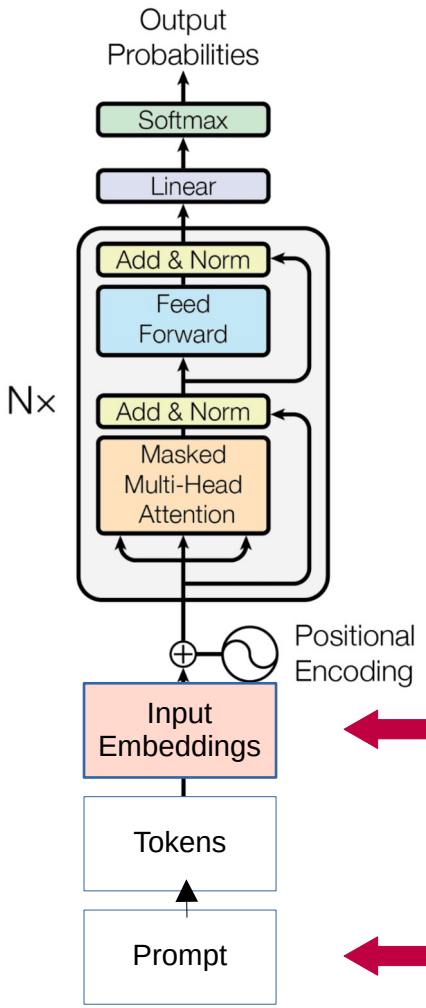
Transformer Decoder



1. Adapt the prompt

Fig. 1: Transformer Decoder (adapted from: Attention is all you need. Vaswani et al. 2017)

Transformer Decoder



2. Adapt the input embeddings (prefix tuning)

1. Adapt the prompt

Fig. 1: Transformer Decoder (adapted from: Attention is all you need. Vaswani et al. 2017)

Transformer Decoder

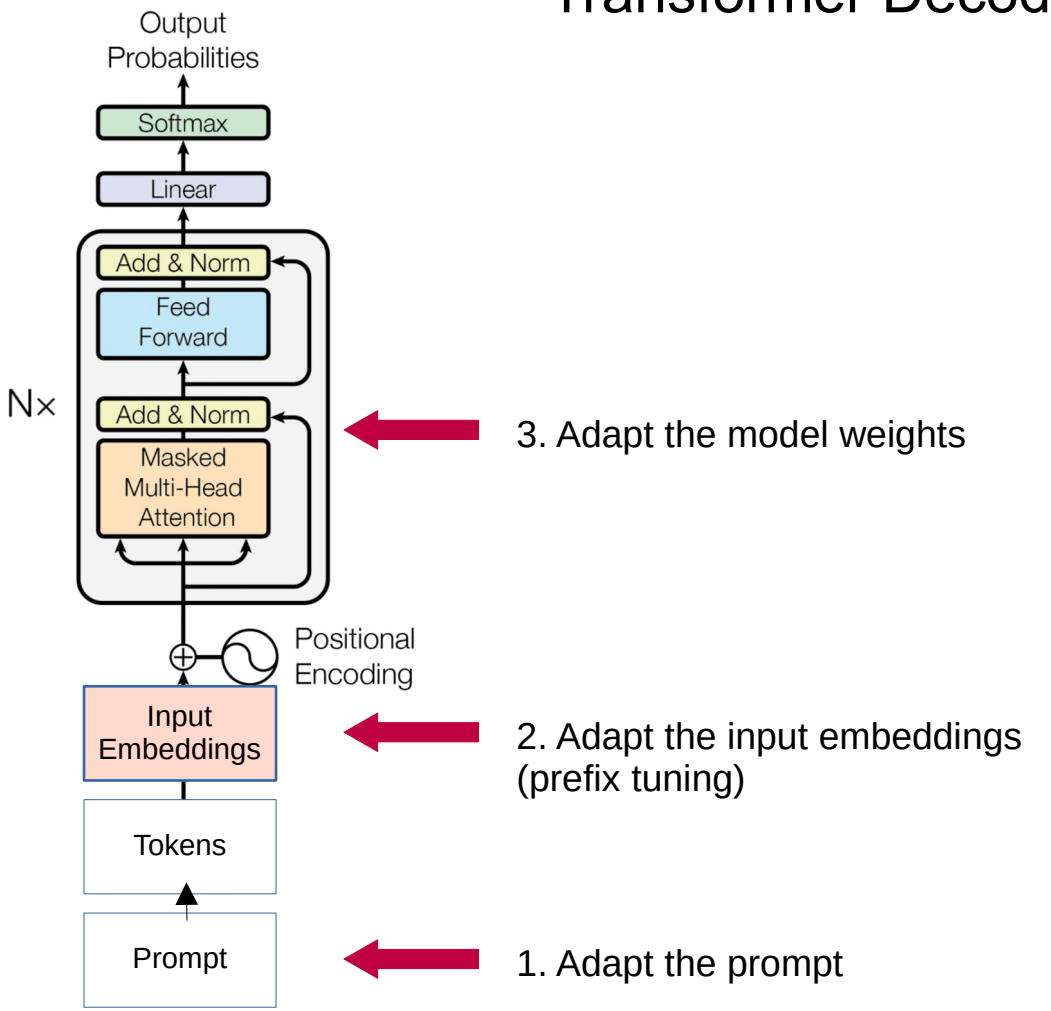
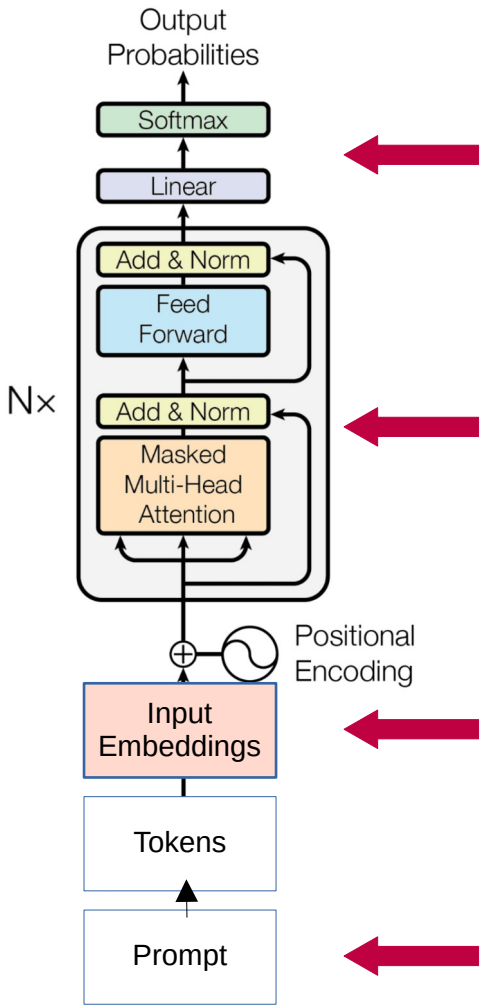


Fig. 1: Transformer Decoder (adapted from: Attention is all you need. Vaswani et al. 2017)

Transformer Decoder



4. Adapt the output layer

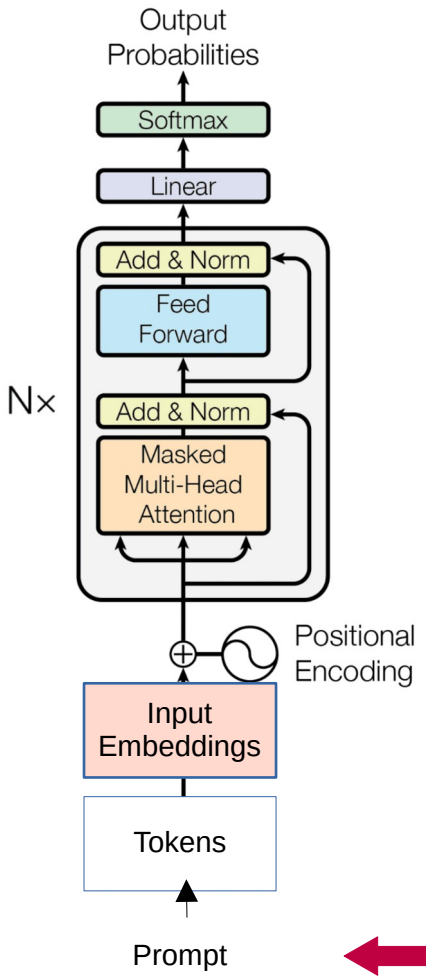
3. Adapt the model weights

2. Adapt the input embeddings (prefix tuning)

1. Adapt the prompt

Fig. 1: Transformer Decoder (adapted from: Attention is all you need. Vaswani et al. 2017)

Adapt the prompts



Types of prompting:

1) Zero Shot Prompting
2) Zero Shot Train of Thought Prompting
3) Few Shot Prompting
4) Few Shot Train of Though Prompting

1. Adapt the prompt

Zero Shot Prompting

Definition:

Model is provided with prompt that is not part of the training data.
The model performs the task without being explicitly trained on it.
The model is given no examples (zero shot).

Problem:
Does not work for more complex tasks.

Zero Shot Prompting

Definition:

Model is provided with prompt that is not part of the training data.
The model performs the task without being explicitly trained on it.
The model is given no examples (zero shot).

Problem:
Does not work for more complex tasks.

Example Prompt:

Prompt:
Classify the following text into the labels [funny, not funny, slightly funny]. Only respond with the label:

Text: Tom went to the pub.

Response (ChatGPT 3.5):
not funny

Zero Shot Prompting

Complex Prompt:

Prompt:

> *A juggler can juggle 16 balls, half of which are golf balls, half of them are green. How many green balls does she have?*

Response (Llama 2, 70B):

> *The juggler has 16 balls in total, and half of them are green. Therefore, the number of blue balls is 16/2 = 8.*
>
> *Since half of the balls are golf balls, the number of golf balls is also 8.*
>
> *So, the juggler has 8 green balls and 8 golf balls.*

Zero Shot Train of Thought Prompting

Tell the LLM to explain the process step by step[1].

Prompt:

> *A juggler can juggle 16 balls, half of which are golf balls, half of them are green. How many green balls does she have?*
> *Lets think step by step and answer how many golf balls there are. Then answer how many of the golf balls are green.*

[1] Wei, J., Wang, X., Schuurmans, D., et al. (2023).
Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.
Google Research, Brain team.

Zero Shot Train of Thought Prompting

Tell the LLM to explain the process step by step[1].

Prompt:

A juggler can juggle 16 balls, half of which are golf balls, half of them are green. How many green balls does she have?
Lets think step by step and answer how many golf balls there are. Then answer how many of the golf balls are green.

Response (Llama 2, 70B):

Sure, let's break this down step by step!

1) *First, we know that the juggler can juggle 16 balls.*
2) *We also know that half of the balls are golf balls. Therefore, the number of golf balls is 16/2 = 8.*
3) *Now, we know that half of the golf balls are green. Therefore, the number of green golf balls is 8/2 = 4.*
4) *Finally, we add the number of green golf balls to the number of green balls that are not golf balls to find the total number of green balls. Since we don't know how many green balls that are not golf balls, we'll just add 4 to the number of green balls that are not golf balls.*

So, the answer to the question "How many green balls does she have?" is 4 + x, where x is the number of green balls that are not golf balls.

[1] Wei, J., Wang, X., Schuurmans, D., et al. (2023).
Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.
Google Research, Brain team.

Few Shot Prompting

Definition:

Model is provided with prompt that is not part of the training data.
The model performs the task without being explicitly trained on it.
The model is given a few examples (few shot).

Few Shot Prompting

Definition:

Model is provided with prompt that is not part of the training data.
The model performs the task without being explicitly trained on it.
The model is given a few examples (few shot).

Advantage:
We can even give completely new tasks and learn by example.

Problem:
Does not work for more complex tasks.

Example Prompt:

Prompt:

Complete this with a single answer (red, green):

These are the examples:
This is great → red
This is good → red
This is awful → green

Complete this:
This is horrible →
This is nice →

Response (ChatGPT 3.5):

This is horrible → green
This is nice → red

Few Shot Prompting

Complex example.

Prompt:

Q: A juggler can juggle 16 balls, half of which are golf balls, half of them are green. How many green balls does she have?
A: 4+x, where x is the number of green balls that are not golf balls.

Q: Susan buys bottles and cans of soda. She has 12 bottles and cans. Half of them are cans. Half of them contain lemonade. How many lemonades does she have?
A:

Response (ChatGPT 3.5):

Let's break down the information given:

1. Susan has 12 bottles and cans of soda.
2. Half of them are cans.

So, the number of cans is 12 / 2 = 6.

3. Half of the bottles and cans contain lemonade.

Now, half of the total number of bottles and cans is 12 / 2 = 6. Since this is also the number of cans, we can conclude that all the cans contain lemonade.

Therefore, Susan has 6 lemonades.

Few Shot Train of Thought Prompting

Complex example with train of thought.

Prompt:

Q: A juggler can juggle 16 balls, half of which are golf balls, half of them are green. How many green balls does she have?
A: Let's think step by step: There are 8 golf balls. 4 golf balls are green. She has 4 green balls plus how many other balls are green

Q: Susan buys bottles and cans of soda. She has 12 bottles and cans. Half of them are cans. Half of them contain lemonade. How many lemonades does she have?
A:

Response (ChatGPT 3.5):

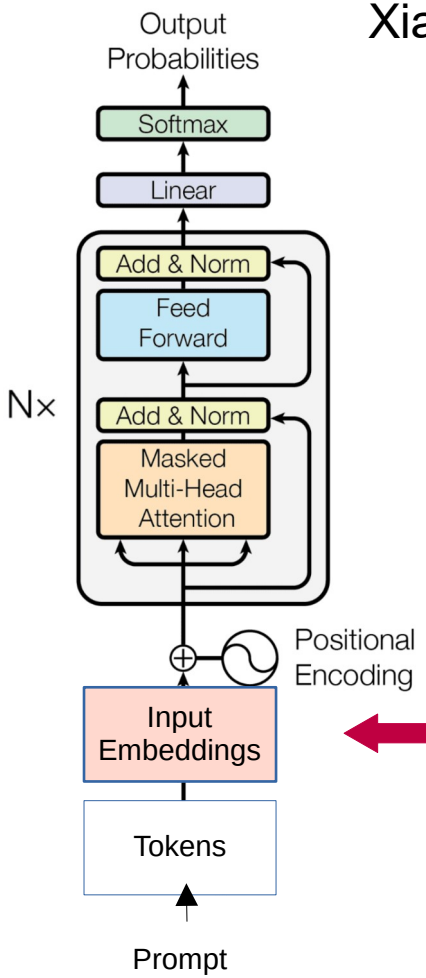
Let's break it down step by step:

1. Half of the bottles and cans are cans.
So, 12 / 2 = 6 of them are cans.

2. Half of the cans contain lemonade.
So, 6 / 2 = 3 cans contain lemonade.

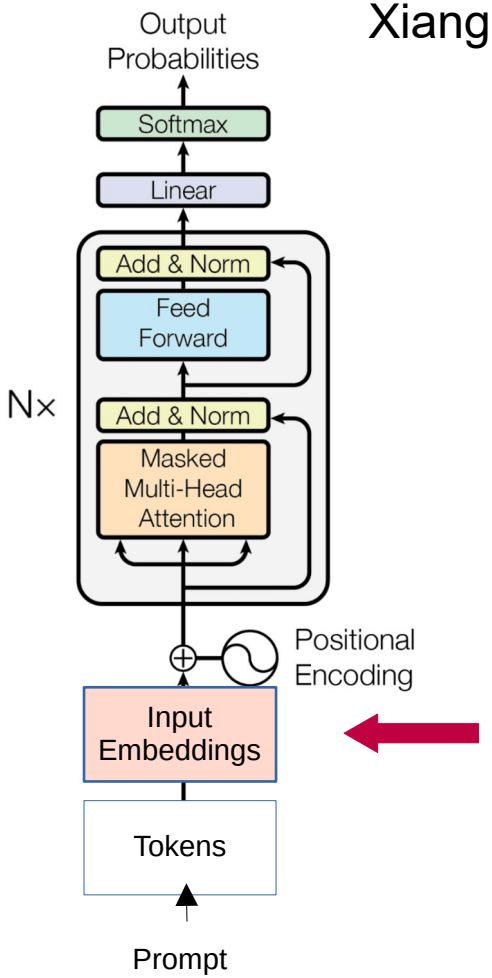
Therefore, Susan has 3 lemonades.

“Prefix-Tuning: Optimizing Continuous Prompts for Generation”
Xiang Lisa Li and Percy Liang, Stanford 2021



2. Adapt the input embeddings (prefix tuning)

“Prefix-Tuning: Optimizing Continuous Prompts for Generation”
Xiang Lisa Li and Percy Liang, Stanford 2021



Recap:
In a transformer model we:

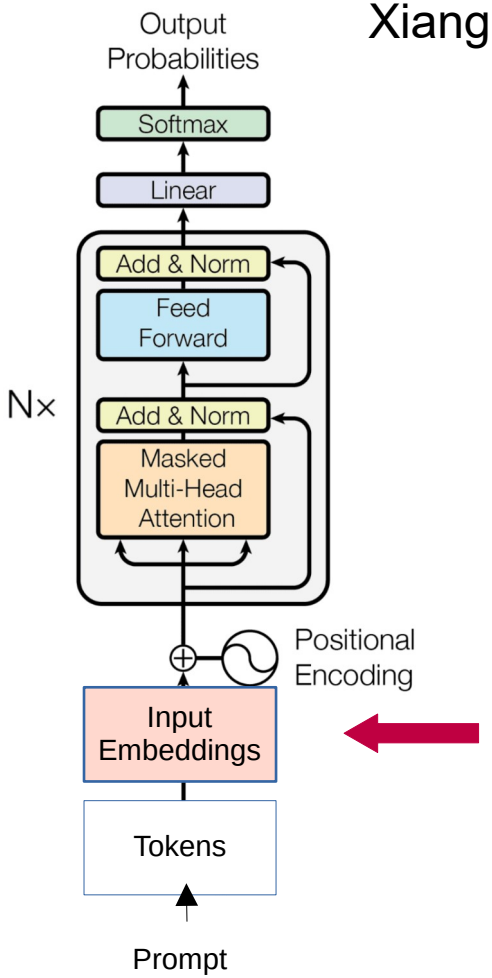
- tokenize the text into token ids,
- look up the initial embeddings for these tokens
- add position information
- feed this trough attention layers

Input text:
”Summarize: NLP is cool. …”

Token ids:
[15, 8, 14, 3, 200, …]

Input Embeddings:
[y_{15}, y_{8}, y_{14}, y_{3}, y_{200}, …]

“Prefix-Tuning: Optimizing Continuous Prompts for Generation”
Xiang Lisa Li and Percy Liang, Stanford 2021



Recap:
In a transformer model we:

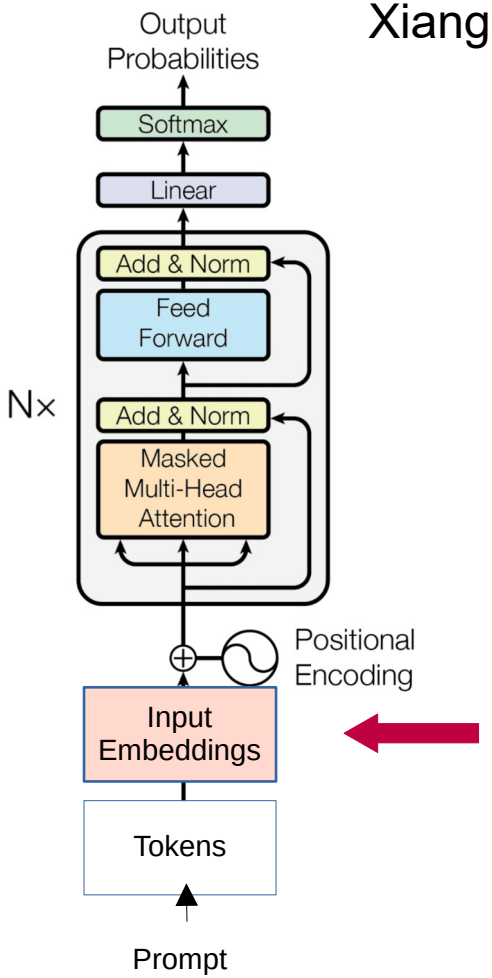
- tokenize the text into token ids,
- look up the initial embeddings for these tokens
- add position information
- feed this trough attention layers

Input text:
”Summarize: NLP is cool. …” ← Prompt engineering optimizes this text

Token ids:
[15, 8, 14, 3, 200, …]

Input Embeddings:
[y_{15}, y_{8}, y_{14}, y_{3}, y_{200}, …]

“Prefix-Tuning: Optimizing Continuous Prompts for Generation”
Xiang Lisa Li and Percy Liang, Stanford 2021



Recap:
In a transformer model we:

- tokenize the text into token ids,
- look up the initial embeddings for these tokens
- add position information
- feed this trough attention layers

Input text:
”Summarize: NLP is cool. …” ← Prompt engineering optimizes this text

Token ids:
[15, 8, 14, 3, 200, …]

Input Embeddings:
[y_{15}, y_{8}, y_{14}, y_{3}, y_{200}, …] ← Can we optimize this instead?

“Prefix-Tuning: Optimizing Continuous Prompts for Generation”
Xiang Lisa Li and Percy Liang, Stanford 2021

Input Embeddings:
[y_{15}, y_8, y_{14}, y_3, y_{200}, …]

Add a prefix to the embeddings:
[p_0, y15, y_8, y_{14}, y_3, y_{200}, …]

“Prefix-Tuning: Optimizing Continuous Prompts for Generation”
Xiang Lisa Li and Percy Liang, Stanford 2021

Input Embeddings:
[y_{15}, y_8, y_{14}, y_3, y_{200}, …]

Add a prefix to the embeddings:
[p_0, y15, y_8, y_{14}, y_3, y_{200}, …]

The prefix p_0 is now a vector of embedding size (e.g. 768).

Fine-tune this vector on a supervised dataset.

In general: Have one or more task dependent prefix vectors

“Prefix-Tuning: Optimizing Continuous Prompts for Generation”
Xiang Lisa Li and Percy Liang, Stanford 2021

Input Embeddings:
[y_{15}, y_8, y_{14}, y_3, y_{200}, …]

Add a prefix to the embeddings:
[p_0, y15, y_8, y_{14}, y_3, y_{200}, …]

The prefix p_0 is now a vector of embedding size (e.g. 768).

Fine-tune this vector on a supervised dataset.

In general: Have one or more task dependent prefix vectors.

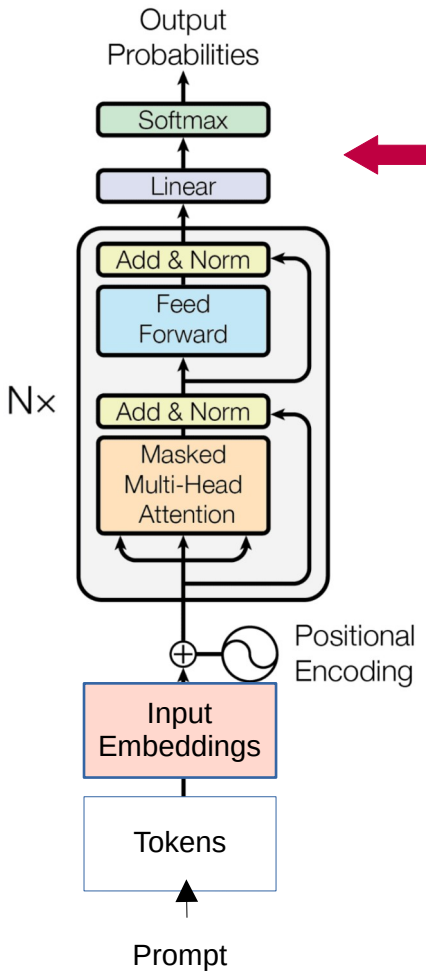
Advantage:
Instead of manually optimizing text prompts we can automatically optimize the parameters of the prefix vector.

Instead of fine-tuning the whole model we only fine-tune the input.

Works with relatively small amount of examples (50-500 examples).

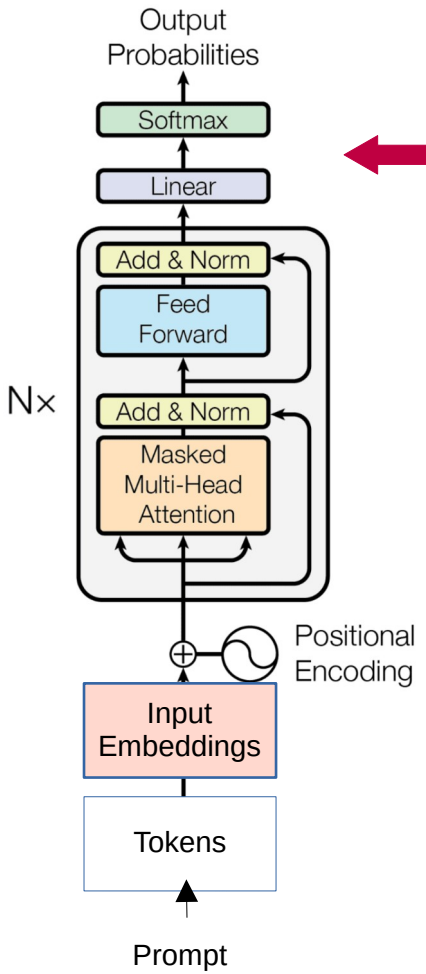
Disadvantage:
We only do instruction tuning. We can not learn completely new tasks.

Fine-Tuning the Output Layer



4. Adapt the output layer

Feature-Based Approaches and Fine-Tuning Approaches



4. Adapt the last layers or train classifier on top of output

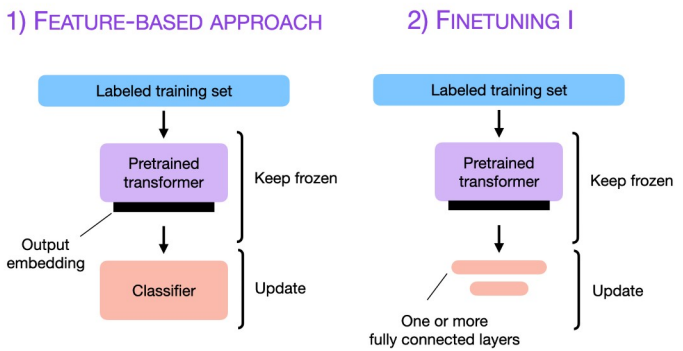


Fig. 2: Fine-Tuning Approaches (adapted from https://magazine.sebastianraschka.com/p/finetuning-large-language-models, accessed 18.01.2024)

Feature-Based Approaches and Fine-Tuning Approaches

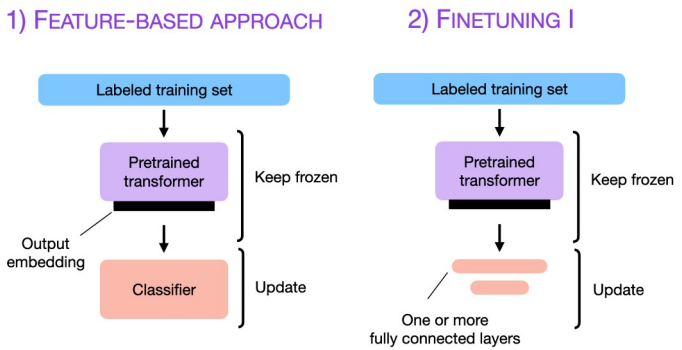


Fig. 2: Fine-Tuning Approaches (adapted from https://magazine.sebastianraschka.com/p/finetuning-large-language-models, accessed 18.01.2024)

Feature-Based Approach:

Take output embeddings of model
Train classifier on top (e.g. sentiment classification)

Advantages:
Only inference is done.
Few parameters in classifier to train. (fast and efficient)
Embeddings can be stored beforehand.

Disadvantages:
Low accuracy

Feature-Based Approaches and Fine-Tuning Approaches

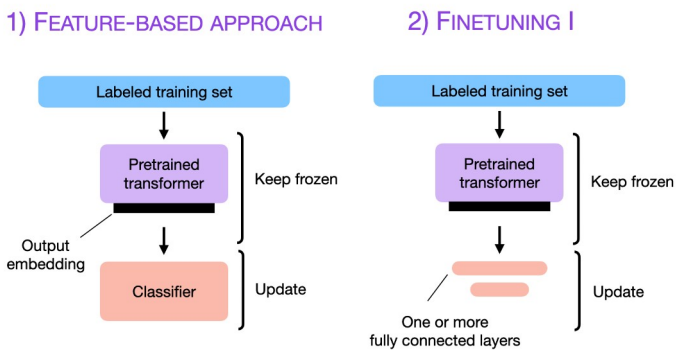


Fig. 2: Fine-Tuning Approaches (adapted from https://magazine.sebastianraschka.com/p/finetuning-large-language-models, accessed 18.01.2024)

Feature-Based Approach:

Take output embeddings of model
Train classifier on top (e.g. sentiment classification)

Advantages:
Only inference is done.
Few parameters in classifier to train. (fast and efficient)
Embeddings can be stored beforehand.

Disadvantages:
Low accuracy

Fine-Tuning I:

Train the output layers of the model on your data.

Advantages:
Slightly higher accuracy.

Disadvantages:
Slightly longer training time.
For inference we need to run the full model.

Fine-Tuning the Full Model

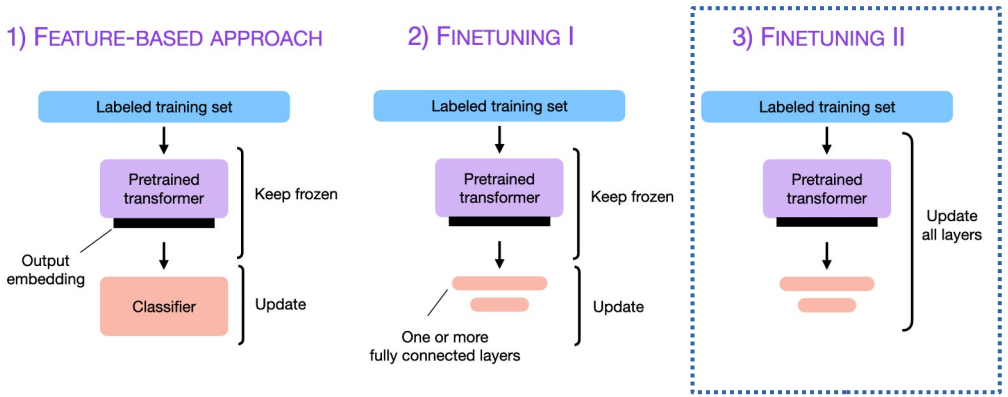


Fig. 2: Fine-Tuning Approaches (adapted from https://magazine.sebastianraschka.com/p/finetuning-large-language-models, accessed 18.01.2024)

Fine-Tuning the Full Model

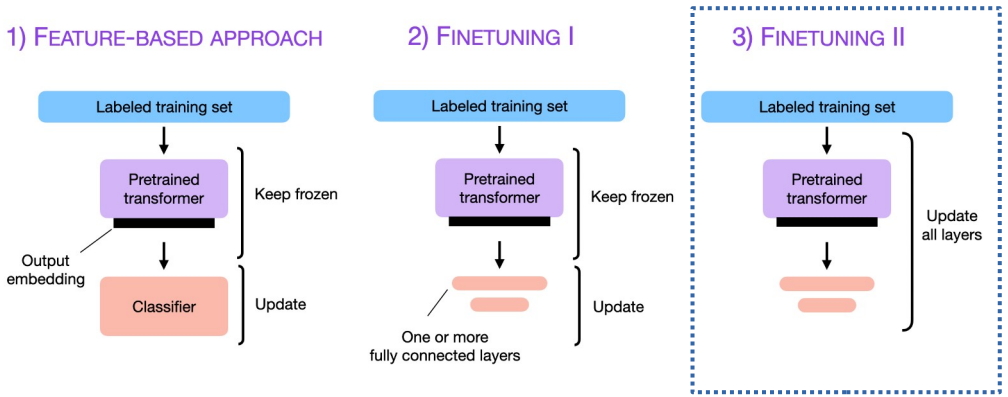


Fig. 2: Fine-Tuning Approaches (adapted from https://magazine.sebastianraschka.com/p/finetuning-large-language-models, accessed 18.01.2024)

Fine-Tuning the Full Model:

Train the full model on new data.

Advantages:
Potentially higher accuracy

Disadvantages:
Overfitting
Needs a lot of examples
Catastrophic forgetting

“LoRa: Low-Rank Adaptation of Large Language Models”
Hu et al, Microsoft 2021

Parameter-efficient Fine-Tuning

Problem:
We only want to adapt the parameters of the model a little.
Typically the model already has knowledge which is beneficial for the task.

How to determine which parameters to update?

“LoRa: Low-Rank Adaptation of Large Language Models”
Hu et al, Microsoft 2021

Parameter-efficient Fine-Tuning

Problem:
We only want to adapt the parameters of the model a little.
Typically the model already has knowledge which is beneficial for the task.

How to determine which parameters to update?

Solution:
Decompose the weight matrices W into an original part W and an update part ΔW.

“LoRa: Low-Rank Adaptation of Large Language Models”
Hu et al, Microsoft 2021

Parameter-efficient Fine-Tuning

Problem:
We only want to adapt the parameters of the model a little. Typically the model already has knowledge which is beneficial for the task.

How to determine which parameters to update?

Solution:
Decompose the weight matrices W into an original part W and an update part ΔW.

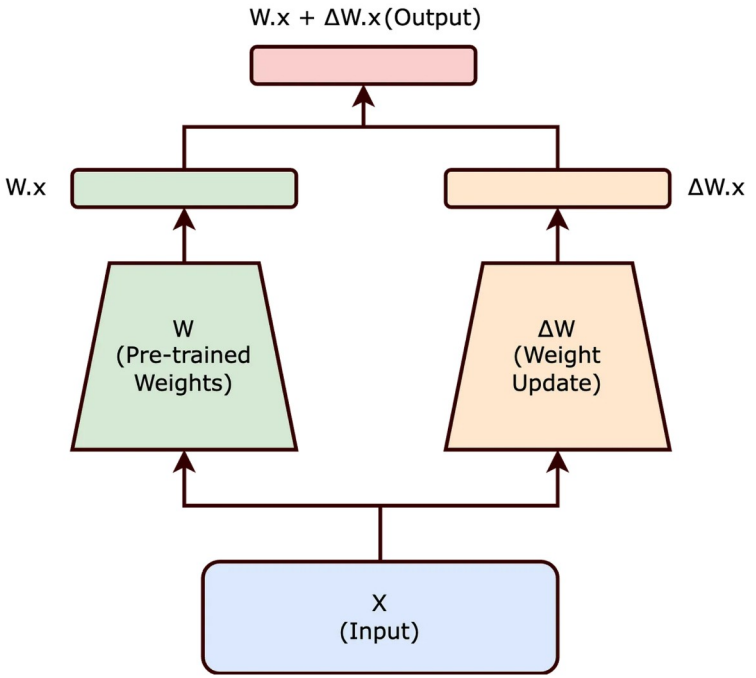


Fig. 3: Frozen original weights and trainable update (from https://towardsdatascience.com/understanding-lora-low-rank-adaptation-for-finetuning-large-models-936bce1a07c6, accessed 18.01.2024)

“LoRa: Low-Rank Adaptation of Large Language Models”
Hu et al, Microsoft 2021

Parameter-efficient Fine-Tuning

Low-Rank Assumption:
The update weight matrix ΔW does not contain a lot of new information.
This means the rank is lower than the original dimension.

→ We can represent the weight update using two smaller matrices:

$\Delta W = AB$

A and B have a lower dimensionality than ΔW

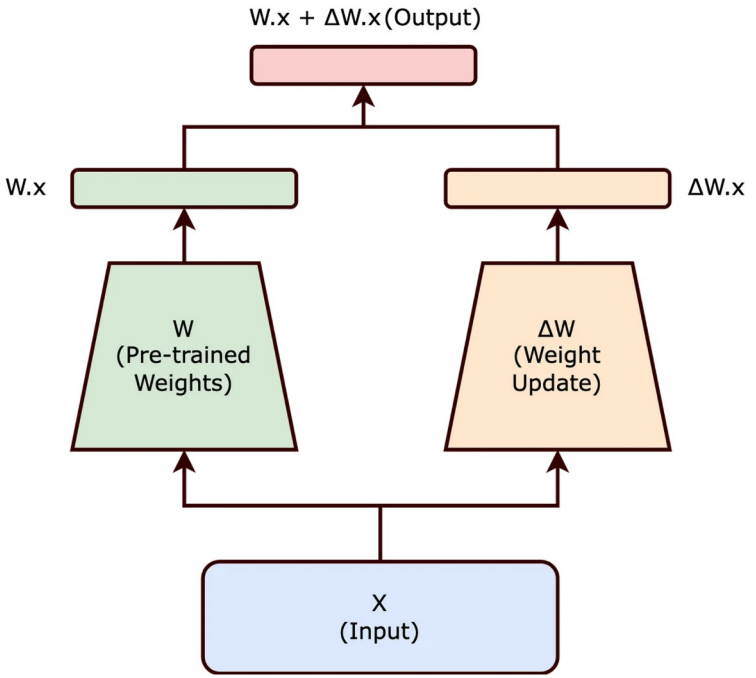


Fig. 3: Frozen original weights and trainable update (from https://towardsdatascience.com/understanding-lora-low-rank-adaptation-for-finetuning-large-models-936bce1a07c6, accessed 18.01.2024)

“LoRa: Low-Rank Adaptation of Large Language Models”
Hu et al, Microsoft 2021

Parameter-efficient Fine-Tuning

Low-Rank Assumption:
The update weight matrix ΔW does not contain a lot of new information.
This means the rank is lower than the original dimension.

→ We can represent the weight update using two smaller matrices:

ΔW = AB

A and B have a lower dimensionality than ΔW

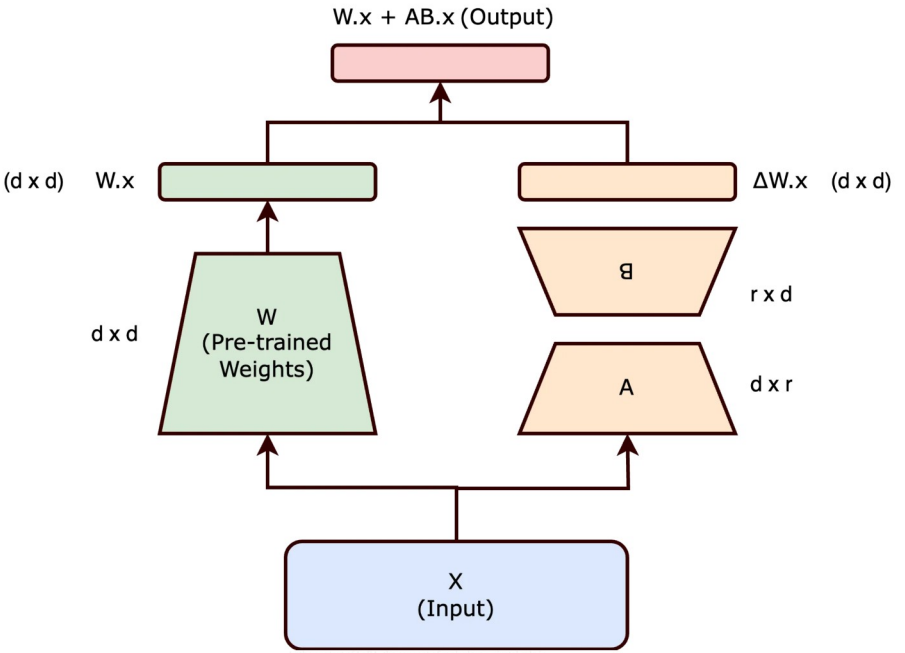


Fig. 4: Frozen original weights and trainable update A and B (from https://towardsdatascience.com/understanding-lora-low-rank-adaptation-for-finetuning-large-models-936bce1a07c6, accessed 18.01.2024)

“LoRa: Low-Rank Adaptation of Large Language Models”
Hu et al, Microsoft 2021

Parameter-efficient Fine-Tuning

Advantages

- Instead of training large weight matrices ΔW we only train smaller matrices A and B
- We can publish the new weights ΔW
  Anyone who wants to use our model downloads the original weights and our update and applies it.
  (Llama → Alpaca → Vicuna etc)
- We avoid licensing issues (we only publish the new part)
- We can have a large base model and switch different update matrices to get a model family (e.g. one for summarization, one for sentiment, etc)

Disadvantages:

- How to choose dimensionality r?
- Which weight matrices to fine-tune?
- Low rank may cause low performance

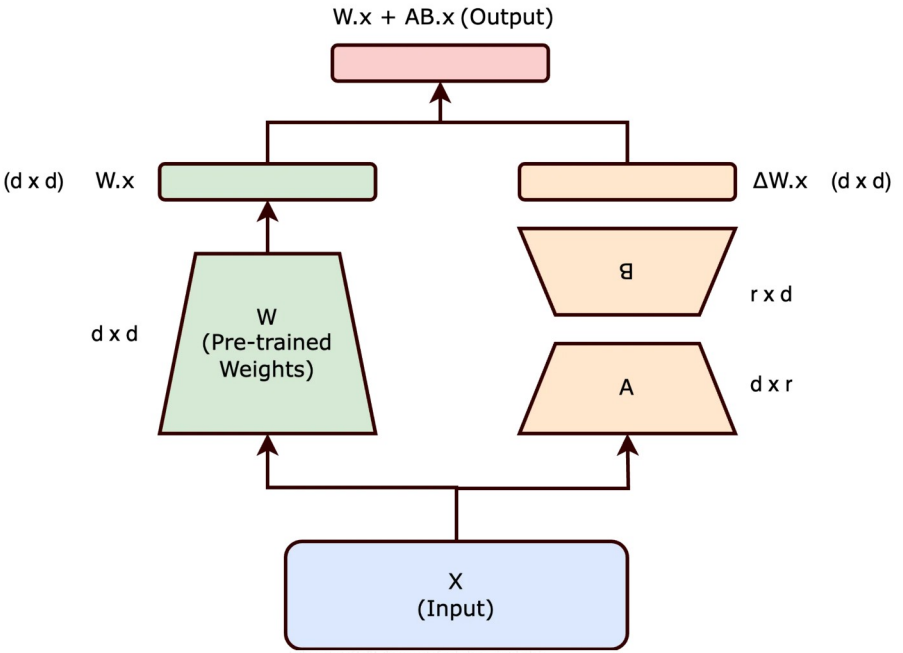


Fig. 4: Frozen original weights and trainable update A and B (from https://towardsdatascience.com/understanding-lora-low-rank-adaptation-for-finetuning-large-models-936bce1a07c6, accessed 18.01.2024)

Prompting and Fine-Tuning Techniques

Types of prompting:

1) Zero Shot Prompting
2) Zero Shot Train of Thought Prompting
3) Few Shot Prompting
4) Few Shot Train of Though Prompting

Types of Fine-Tuning:

1) Prefix Tuning
2) General Fine-Tuning
3) Low Rank Matrix Adaptation