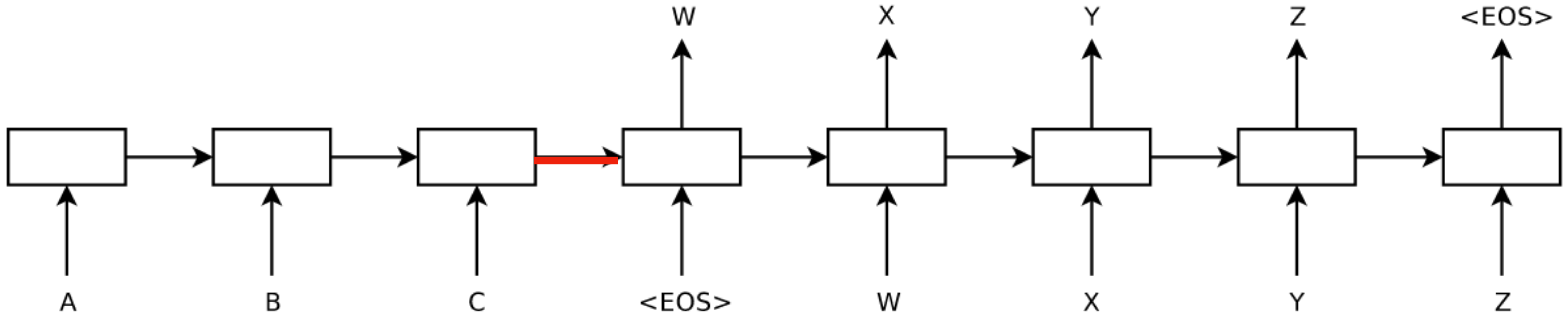
Computational Semantics VT21

Seminar Notes

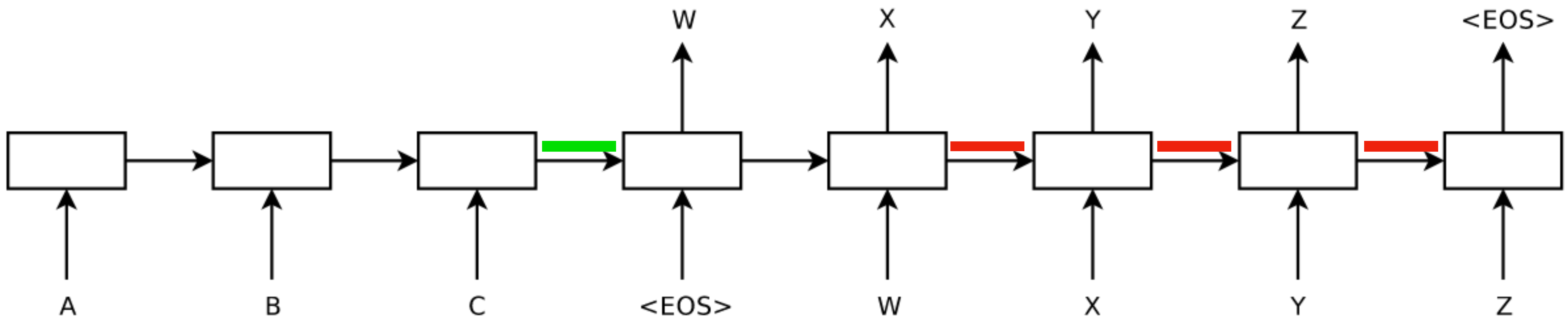
14 May 2021

Encoder-Decoder

- Sutskever 2014: encoder-decoder architecture.
  - First, we have a sequence of source language words (A, B, C). We encode them into a single vector (last hidden state of the encoder LSTM, red line). We use this vector as initial hidden vector for the decoder LSTM. Also, we feed a sequence of target language words to the decoder LSTM (W, X, Y, Z).
  - Through **backpropagation**, weights in the encoder and decoder will be updated. The architecture is called sequence-to-sequence model.



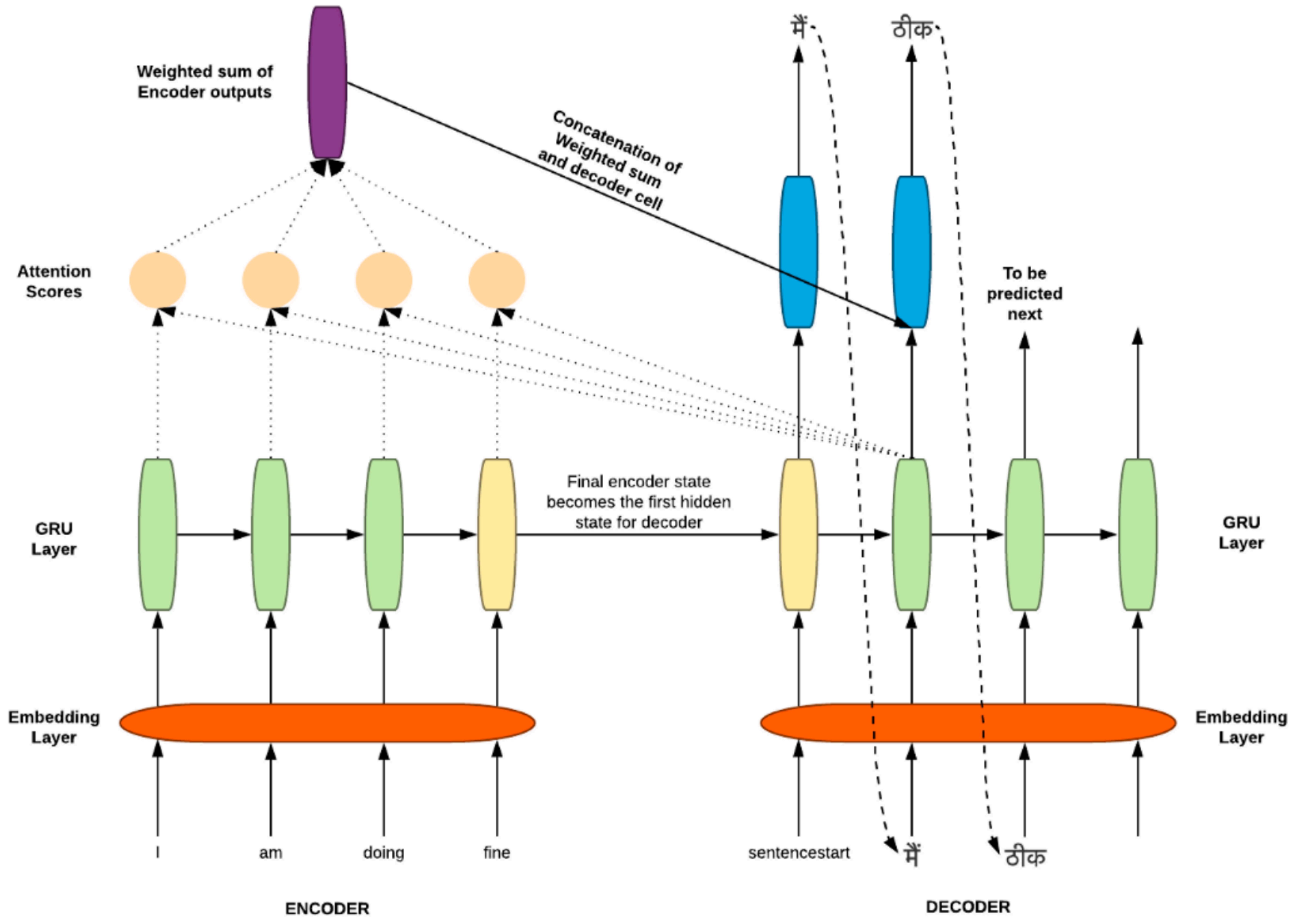
Encoder-Decoder: Problem



- The problem: all hidden states denoted with red lines will receive weaker signal from the green hidden state (encoder's last hidden state). The hidden state from Y to Z, in particular, will forget most of the information about A-B-C.

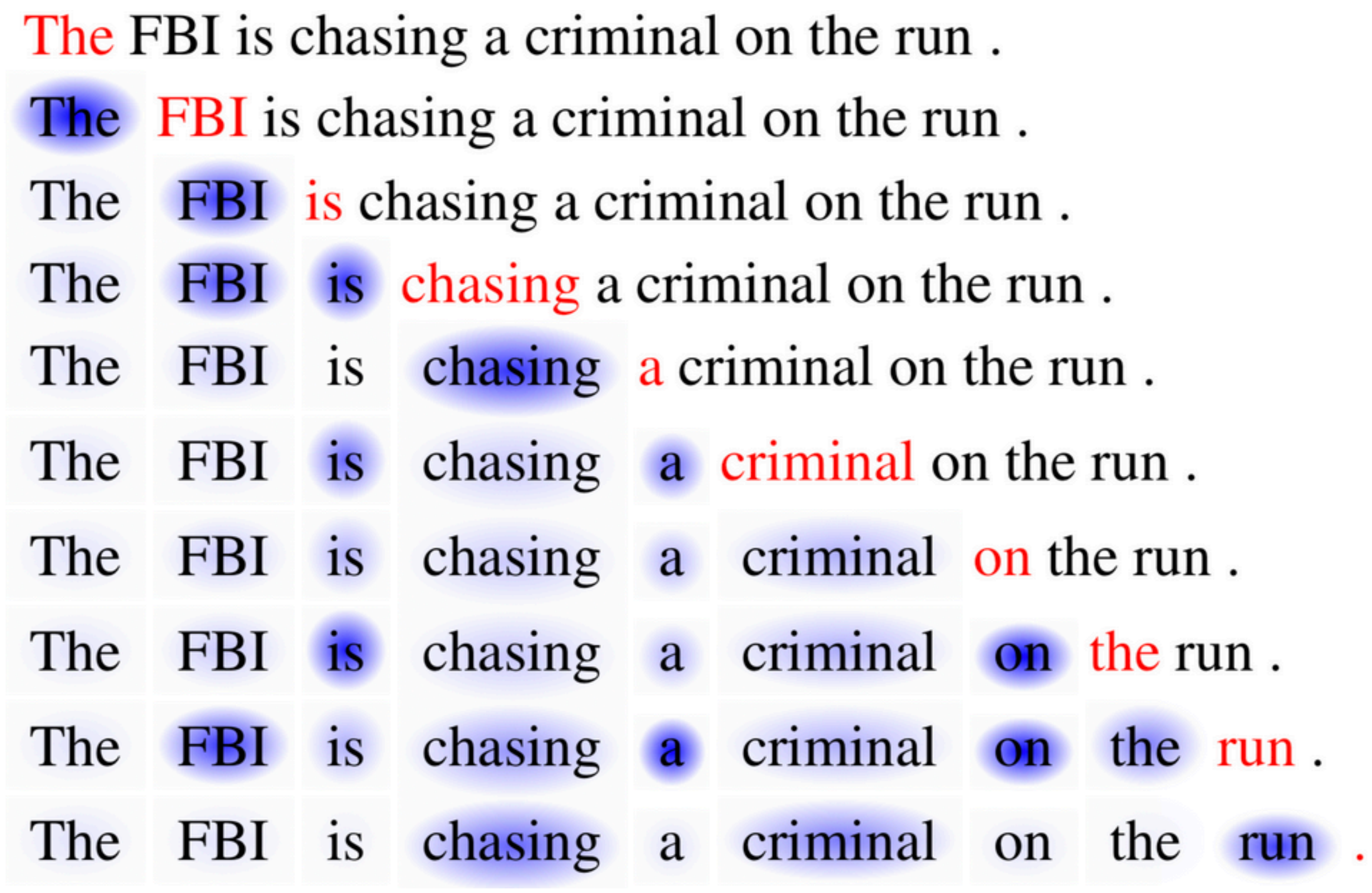
Encoder-Decoder + Attention

- Instead, at each timestep in the decoder (EOS -> W, W-> X, etc.), allow the decoder to look at A, B and C in the encoder. The decoder will **weight** contribution of all source language words for every target language word.
- **Important:** in attention we are attending to elements in the source for each element in the target.

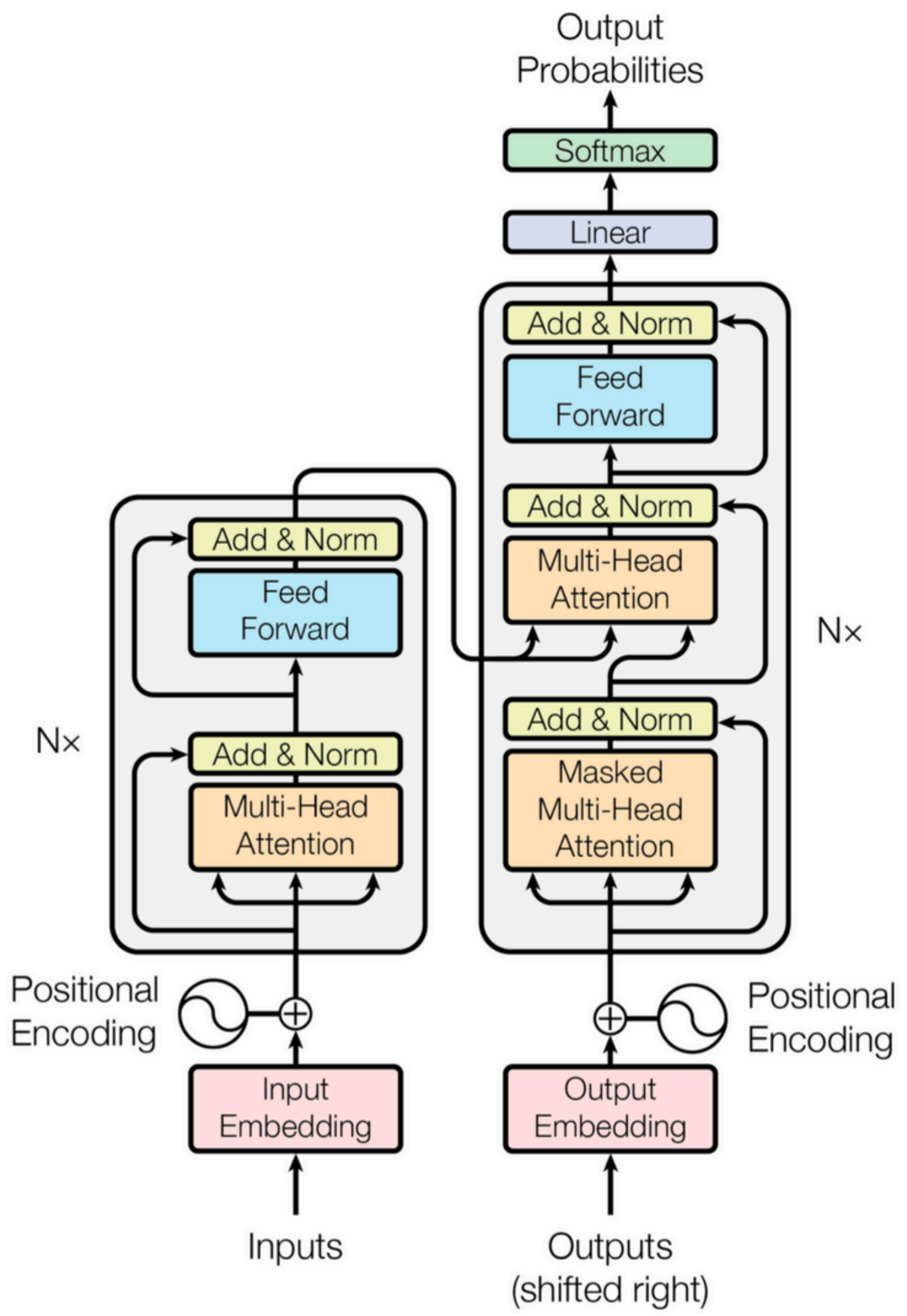

Weighted sum of Encoder outputs
Concatenation of Weighted sum and decoder cell
मैं
ठीक
Attention Scores
To be predicted next
GRU Layer
Final encoder state becomes the first hidden state for decoder
GRU Layer
Embedding Layer
Embedding Layer
I
am
doing
fine
sentencestart
मैं
ठीक
ENCODER
DECODER

Self-Attention

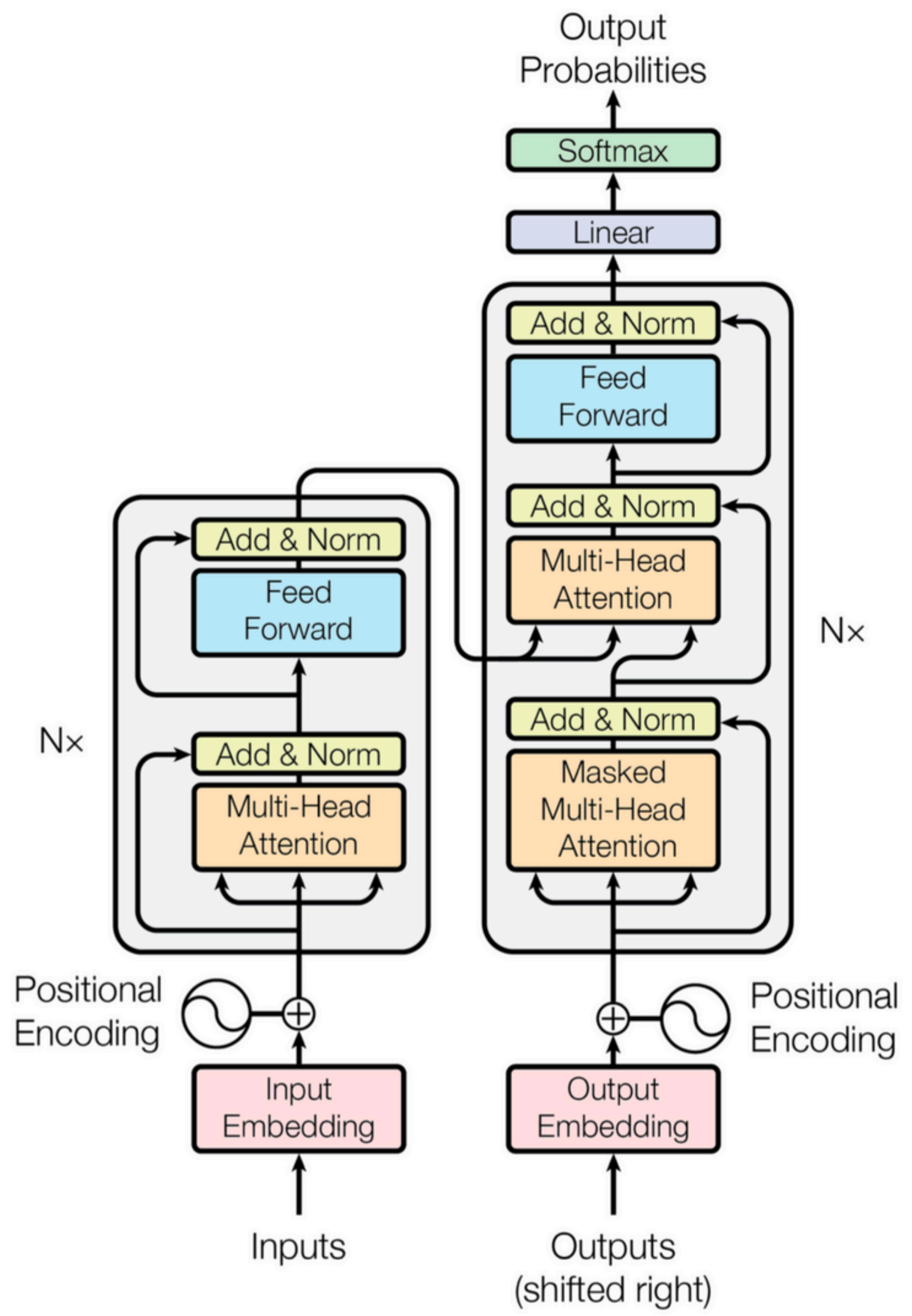
- The attention mechanism allows output to focus attention on input while producing output **while the self-attention model allows inputs to interact with each other (i.e calculate attention of all other inputs wrt one input).**


The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .

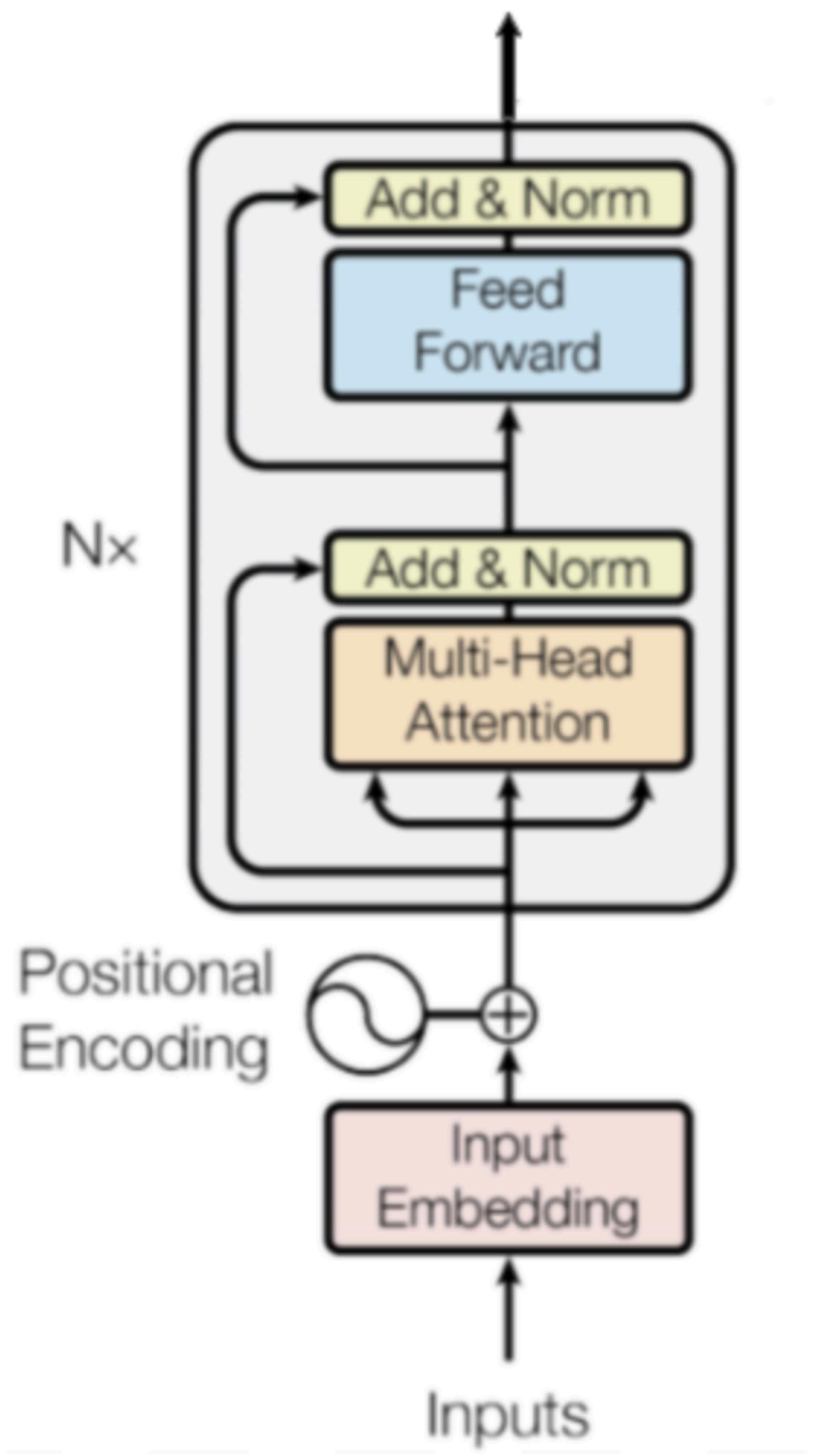
TRANSFORMER



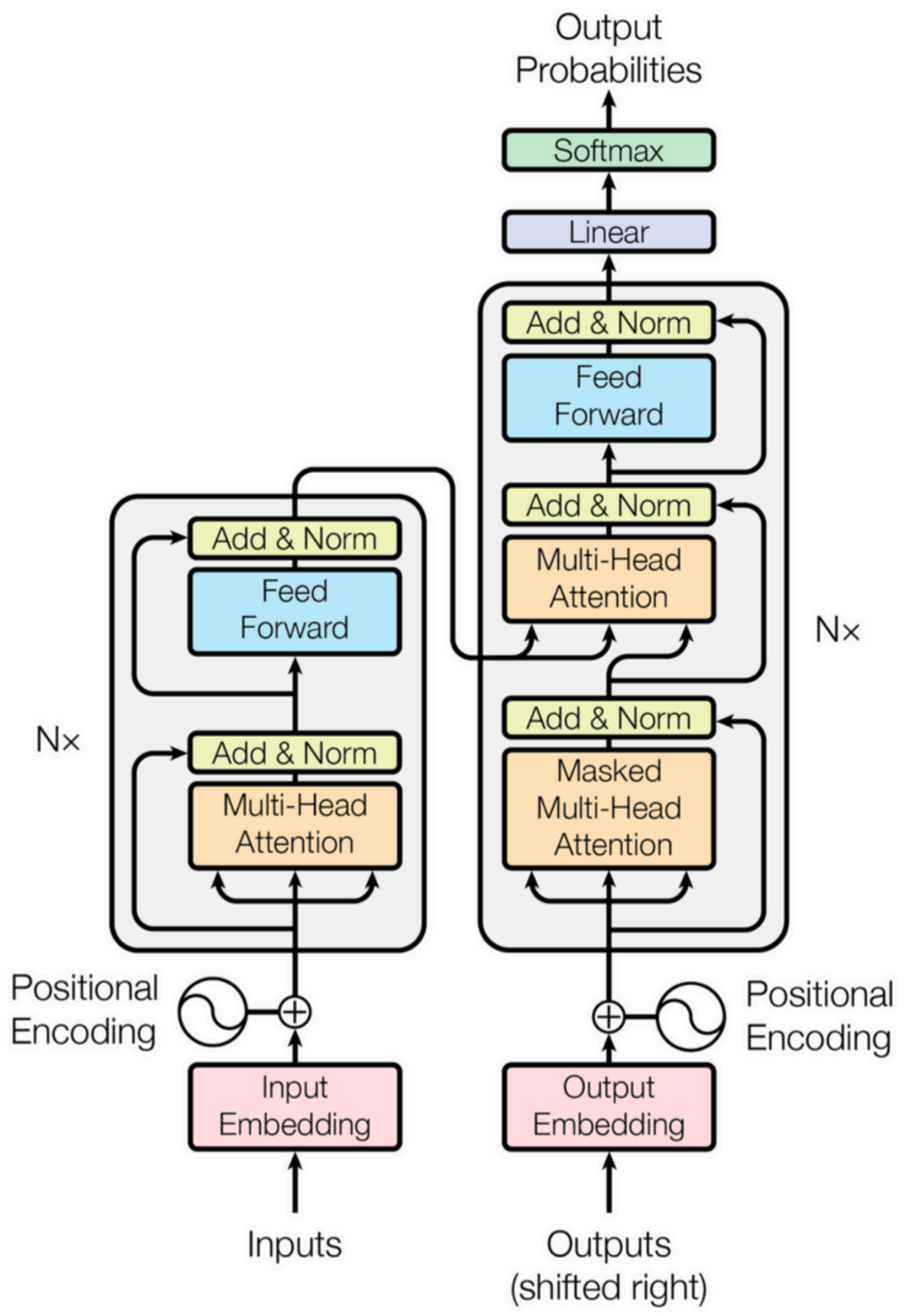
TRANSFORMER



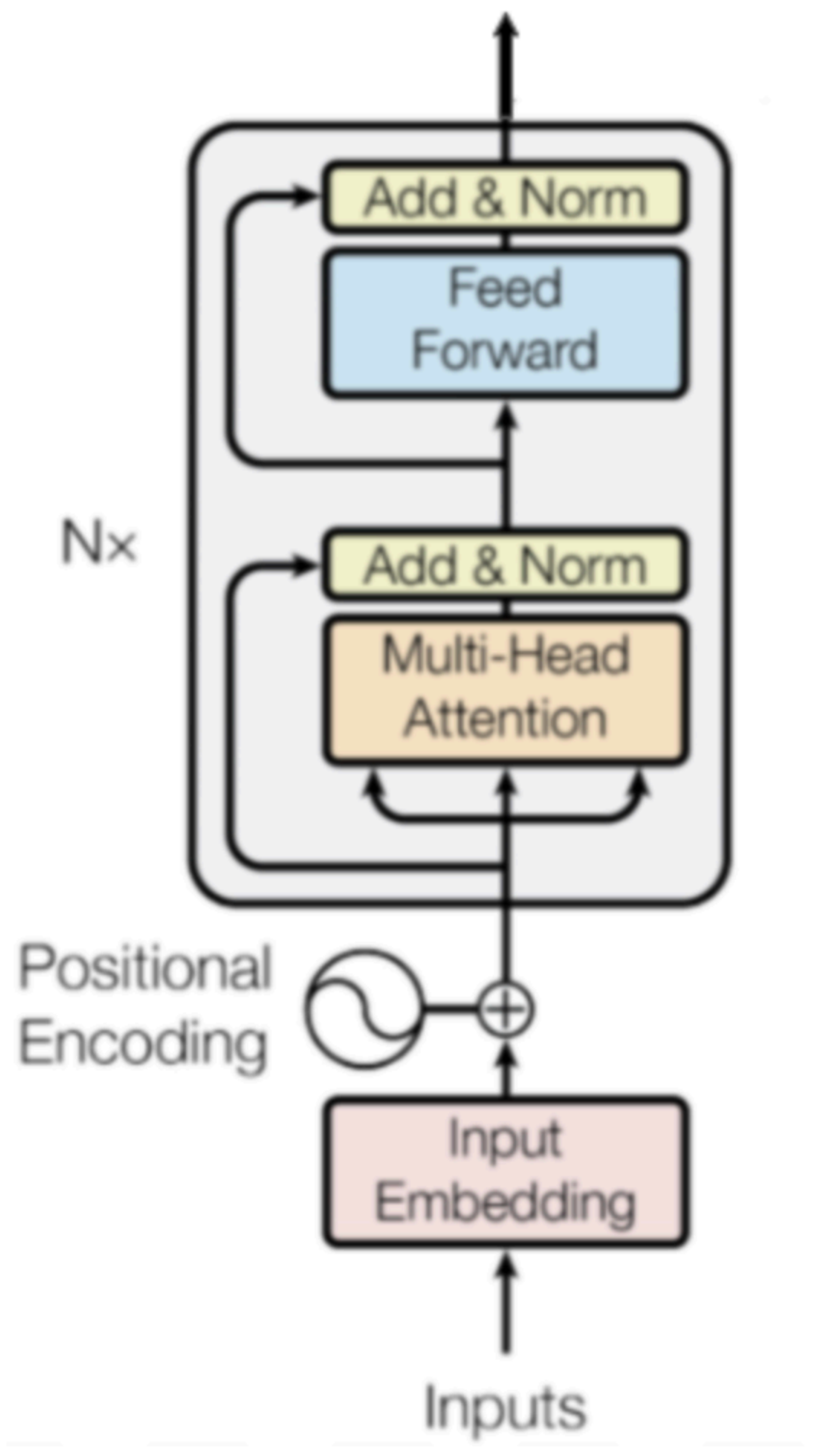
BERT



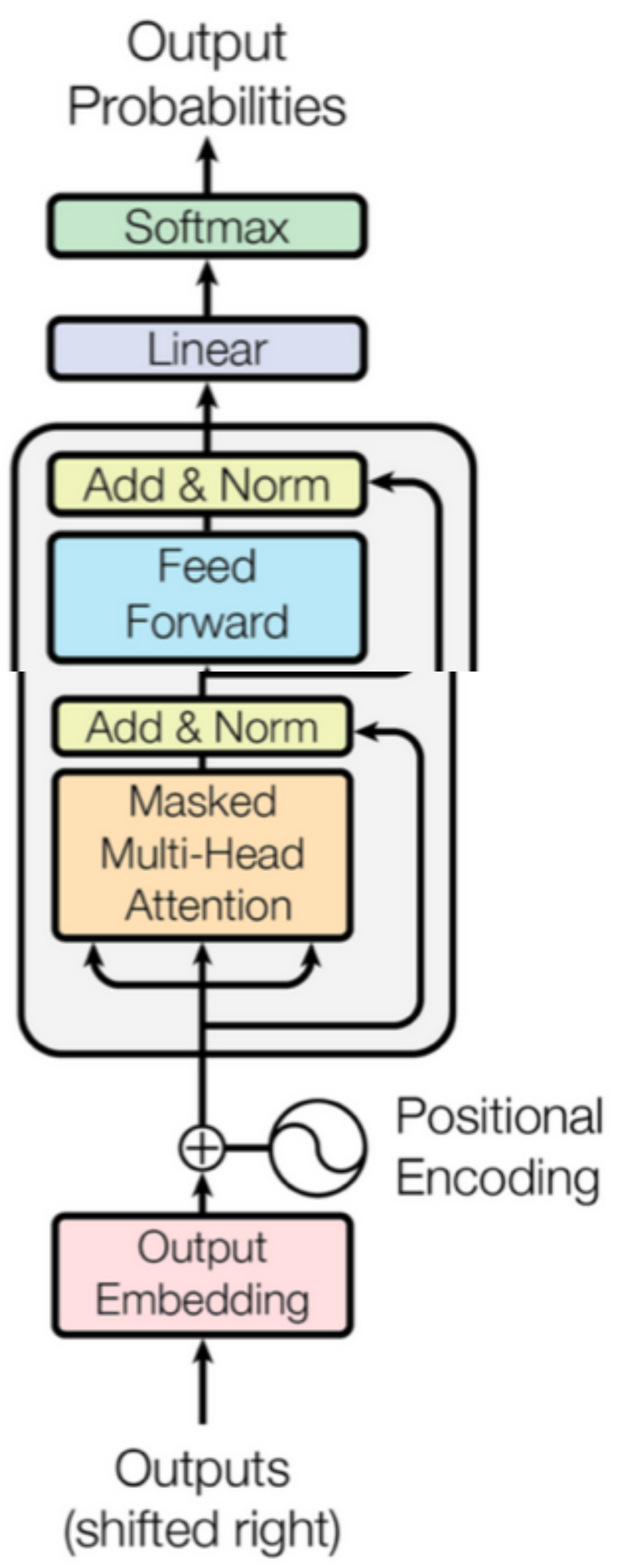
TRANSFORMER



BERT



GPT-2



CLS and SEP tokens

- These tokens are ‘words’, e.g. we learn embeddings for them.
- SEP token should learn where to separate between sentences in the input sequence. Why? We pre-train BERT for Next Sentence Prediction task, and we put SEP between two sentence to explicitly tell BERT that these two sentences are different. Otherwise, both sentences will be treated as a single sequence of words.
- CLS token is the sentence-level representation. It is added because of the pre-training tasks (sentence classification). We also need an input that would represent the entire sentence. We cannot take any other word from the input sequence, because these are unique word representations, so we need a way to get an embedding for the whole sentence.

BERT (base vs large)

- These are two versions of the same model:
- BERT-base is a smaller version with *12 encoder layers, 768 neurons in hidden states and 12 self-attention heads in each encoder*.
- BERT-large is a larger version with more parameters: *24 encoder layers, 1024 neurons in hidden states, 16 self-attention heads in each encoder*.
- General idea: the more, the deeper != ’better' (**it depends!**)
- Problem: hard to explain, hard to interpret (research topic: BERTology, **VG project**)

Training NLP models

- A standard workflow in building NLP models: train, validate, test. We train our model, we validate its performance and choose the best checkpoint, we test our model on the unseen split of data.
    - Train, validation and test sets are coming from the same dataset.
- Often, we do not have enough data to get **good** models. We might have tons of Wikipedia articles for English, but, let's say, five articles for Swedish. Our model will not be able to learn anything good from these five articles.
- In addition, we want our model to learn (i) 'general understanding of language', (ii) more specific understanding required for particular tasks. For example, the model would benefit from first seeing tons of texts in English to learn something about sentence structure, grammar, meaning, everything by a bit. (**Bender, Koller:** https://www.aclweb.org/anthology/2020.acl-main.463/**).** And then we would teach our model how to, for example, predict if the sentence is positive or negative.

Fine-tuning

- We say that we **pre-train our models,** when we train them on a large dataset for some basic task (sentence classification, masked language modelling)
- Now, our model has learned general knowledge about language.
- We are going to **transfer model's knowledge (transfer learning)** to a more specific related task, **which normally has much less data to learn from.**
  - The intuition: if the model is pre-trained on a large dataset, it could serve as a general model for the world of language. We can take advantage of this model and use it as a starting point for training it for our more specific language task - **fine-tune pre-trained model on our small language task.**
- When fine-tuning, we unfreeze **only** some parts of the model and update them for our small task. Why? Because we want our model to keep general knowledge of language from the pre-training step and build a more specific understanding on top of that.

Fine-tuning

- Pre-train BERT on general tasks
- How can we use BERT then? We need to use BERT for our downstream task.
    - Fine-tuning helps, how do we fine-tune?
        - Normally**,** fine-tuning means that you update ALL PRE-TRAINED MODEL'S PARAMETERS. **You're using loss.backward(), opt.step() and opt.zero_grad() (i.e. you're UPDATING the model parameters), if you don't, then you're not fine-tuning :)**
        - However, there are various strategies for different tasks, such as fine-tuning the whole initialised network or 'freezing' some of the pre-trained weights (usually whole layers). Why freezing? Sometimes, you do not want to touch earlier layers in your model, because they have shown to learn general knowledge of syntax and semantics. You would unfreeze later layers and fine-tune them.
            - **VG project:** play around with pre-trained BERT and fine-tune it as a whole, parts of it, or add a layer on top of it and train this layer for your downstream task.
            - Un/freeze parameters: parameter.requires_grad = True/False

Links

- Attention in details: https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html
- BERT in details: https://medium.com/dissecting-bert/dissecting-bert-part-1-d3c3d495cdb3