Outline

- Motivation
- Transformers
  - Goals
  - Components
- Applications

Motivation

- Language translation
- Next word prediction
- Downstream tasks
  - Sentiment analysis
  - Semantic search

Goals

- The goal of any language model is to transform a block of text into a sequence of vectors
  - One vector for each word/token
- We then use that sequence of vectors to do something else
  - Predict the next word
  - Classify a sentence
- We will use a transformer to do this!

Transformers

- A transformer is a type of neural network
- First developed in 2017 (AIAYN)

- Just like word embedding neural networks, we will train on a large corpus of text
  - OpenAI uses ‘web-scale’ data
  - Use SGD to update parameters of the NN
  - Transformers are more powerful!

Transformers – Goals

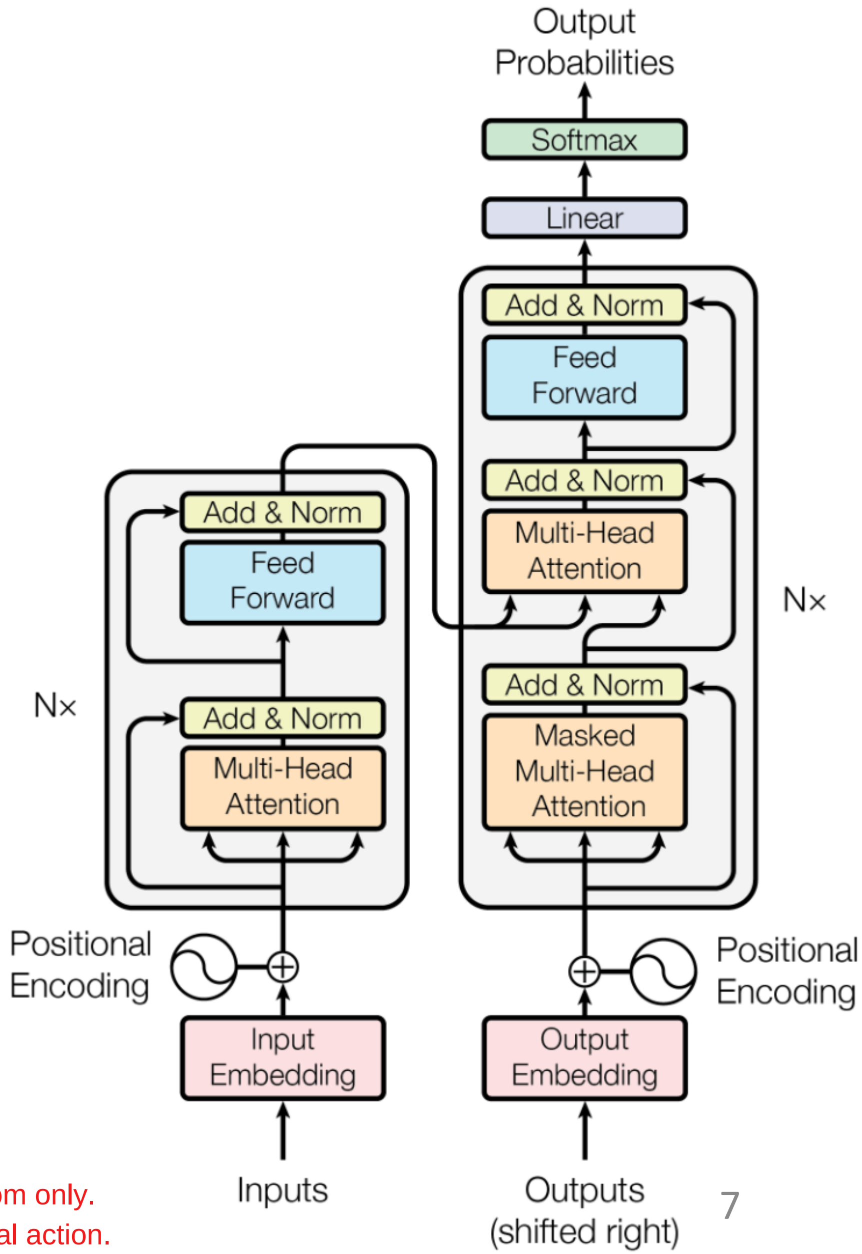
- Contextual awareness
  - I don’t like the sun. When I saw the blue sky, I felt blue.
- Learned relevance
- Speed
  - Every part of a transformer is parallelizable
  - Build bigger models and train them faster
  - OpenAI bought ~30,000 Nvidia gpus in 2022!

I asked chatGPT

- How do transformers work?
  - Transformers are a type of deep learning architecture introduced in the paper "Attention Is All You Need" by Vaswani et al. in 2017. They have since become a fundamental component in various natural language processing (NLP) tasks due to their efficiency in handling long-range dependencies and parallelization. Transformers are particularly renowned for their success in tasks such as machine translation, text generation, and sentiment analysis.
  - The key idea behind transformers is self-attention, a mechanism that allows the model to weigh the importance of different words in a sentence when processing each word. This way, the model can focus on relevant information and capture relationships between words that are crucial for understanding the context.

Transformers – Components

- Embedding
- Positional Encoding
- Multi-Head Self Attention (MHSA)
- Multi-Level Perceptron (MLP)



Embedding

- Convert input (text, images) to sequences of vectors
- Tokenization
  - Word, sub-word, character
  - I am learning about transformers.
  - [BOS], “I”, “am”, “learn”, “##ing”, “about”, “transform”, “##ers”, “.”, [EOS]
- Dictionary
- Convert each token in dictionary to vector
  - Explicit or implicit

Embedding

- We could start with the embeddings from word2vec or GloVe
- Take each token in the block of text's embeddings and feed them into the neural network
- Eventually, we may ask the transformer to modify the initial embedding for each word using SGD

Positional Encoding

- Transformer math is blind to the order that words appear
- Modify each vector in the sequence to represent its position
  - Additive
  - RoPE

Additive Positional Encoding

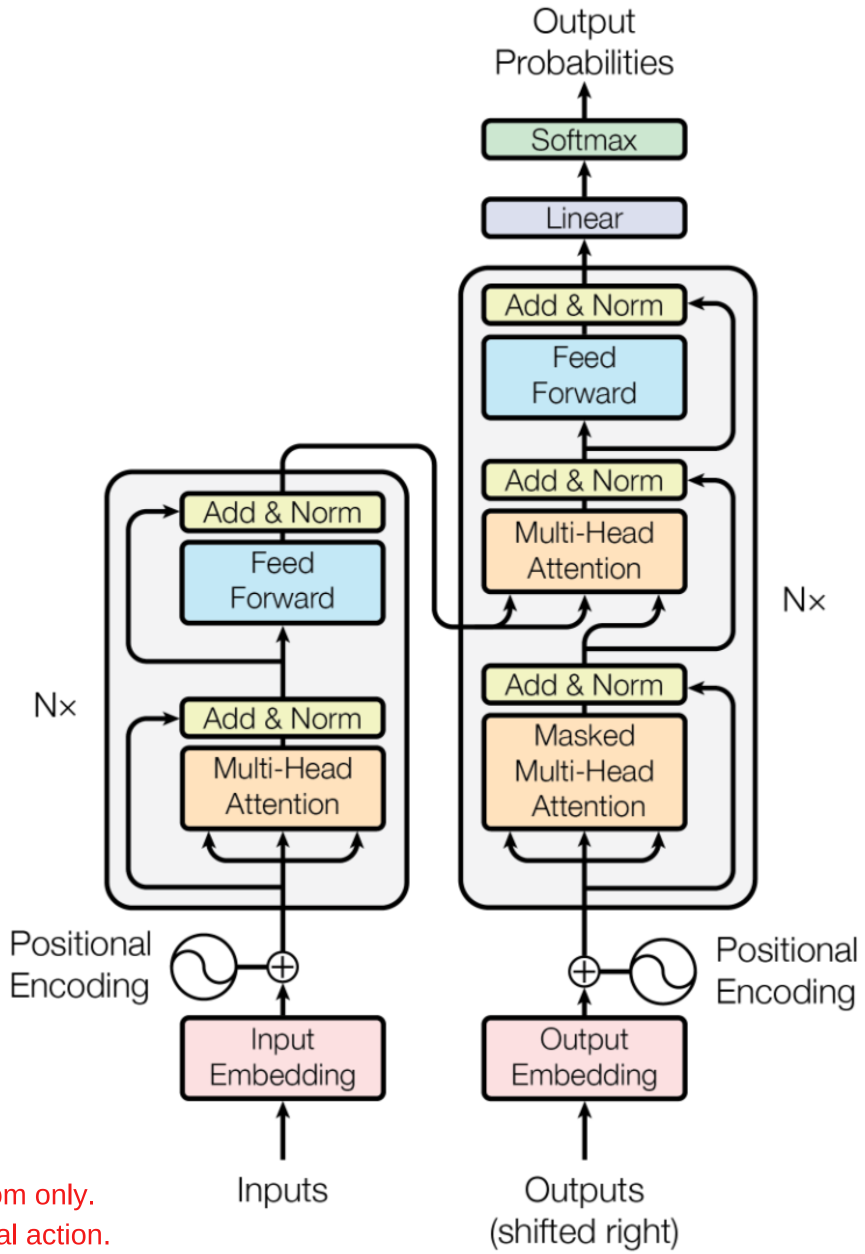
- I like water.

Rotary Positional Encoding

- I like water.

Multi-Headed Self Attention

- We now understand the input to transformer
- Next, these modified input embeddings are fed through a multi-headed self attention layer
- To understand this, we need to build up!



Attention

- Web search
  - Search term – Query
  - Meta Data – Key
  - Content of page – Value
- How similar is the query to each key?
  - Cosine or dot-product similarity between query and keys
- Softmax

Attention

- Query: What types of trees grow in Texas?
- Keys:
  - Fishing, boats, dogs
  - Arborists, Austin, grass
  - Texas, music, motorcycles
- Dot-Product Similarities: -0.1, 0.9, 0.5
- Softmax: 0.18, 0.49, 0.33

Attention

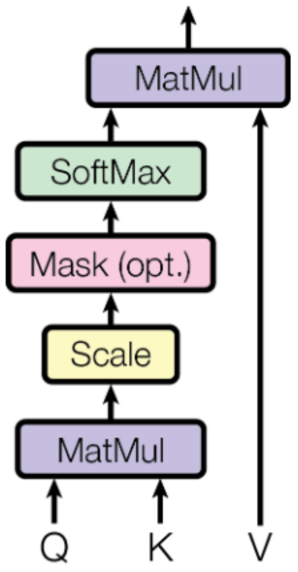
- Weighted average of values

- Pay more ***attention*** to the things that are similar!

Self Attention

- Set each embedded token to be the queries, keys, and values!
  - Get a new vector for each token!
- Before that…
  - Feed each embedding through 3 separate dense layers: for key, query, and value
  - Weights and biases of 3 layers are learned

Scaled Dot-Product Attention

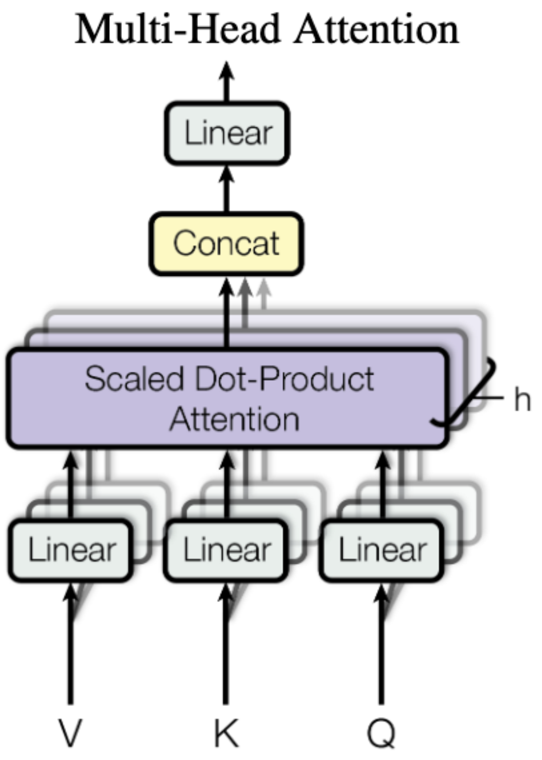


Self Attention – Example

- Attention is cool

Multi-Head Self Attention

- Self attention several times, using several query/key/value matrices
- Concatenate
- Feed all outputs through standard dense layer of MLP with ReLU activation



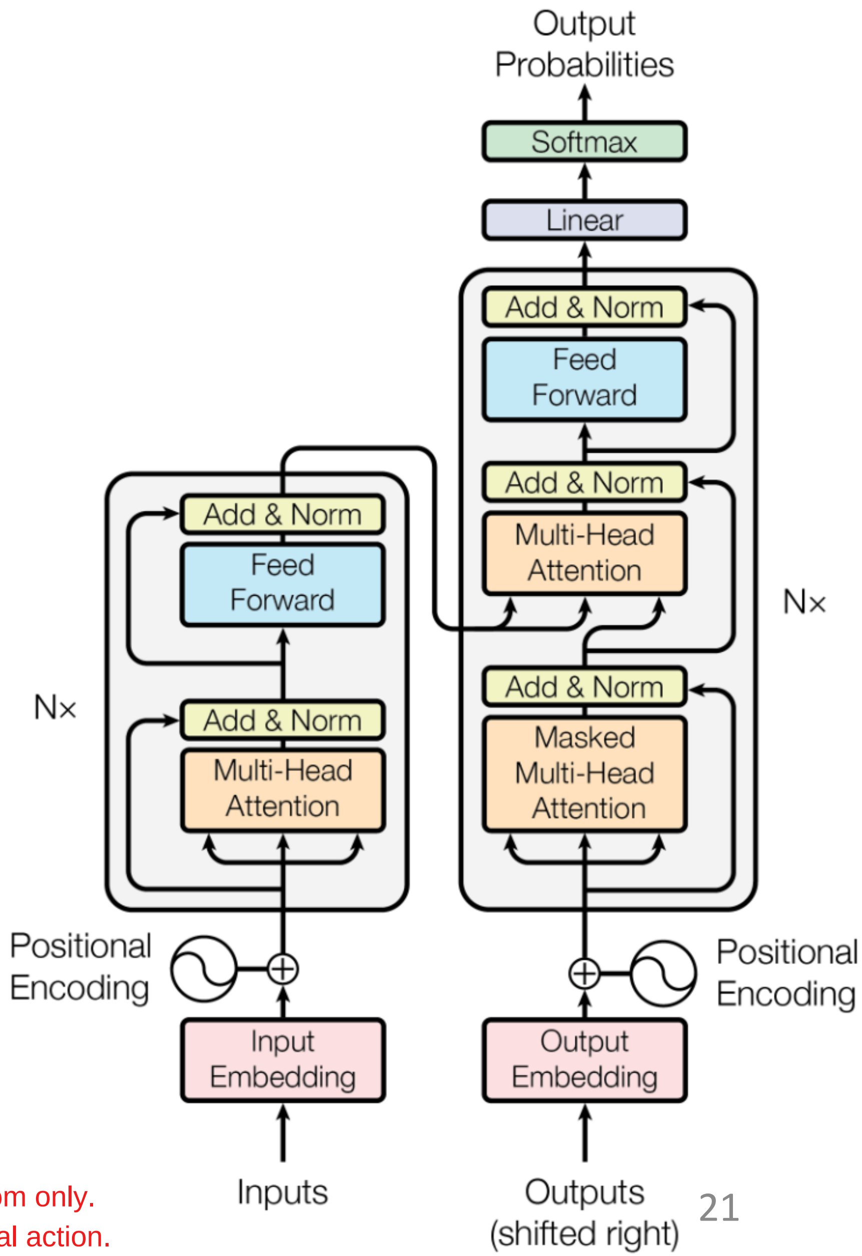
Transformer

- Each token's embedding gets changed to something new from MHSA
- Take outputs, and feed them each through a dense layer
- Then feed that into a new MHSA layer with new query/key/value layers
- Stack several MHSA/Dense layers on top of each other!

Transformer

- Skip Connections

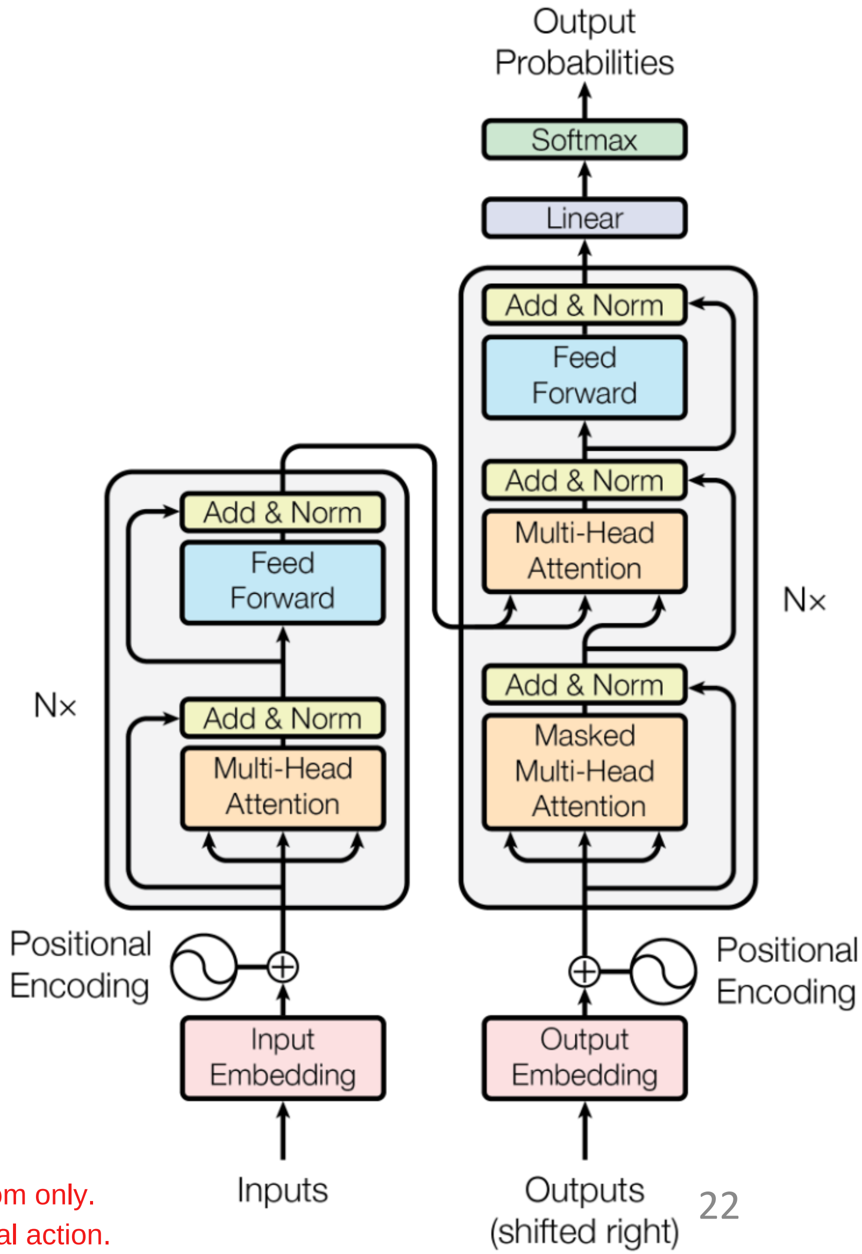
- Masking



Transformer

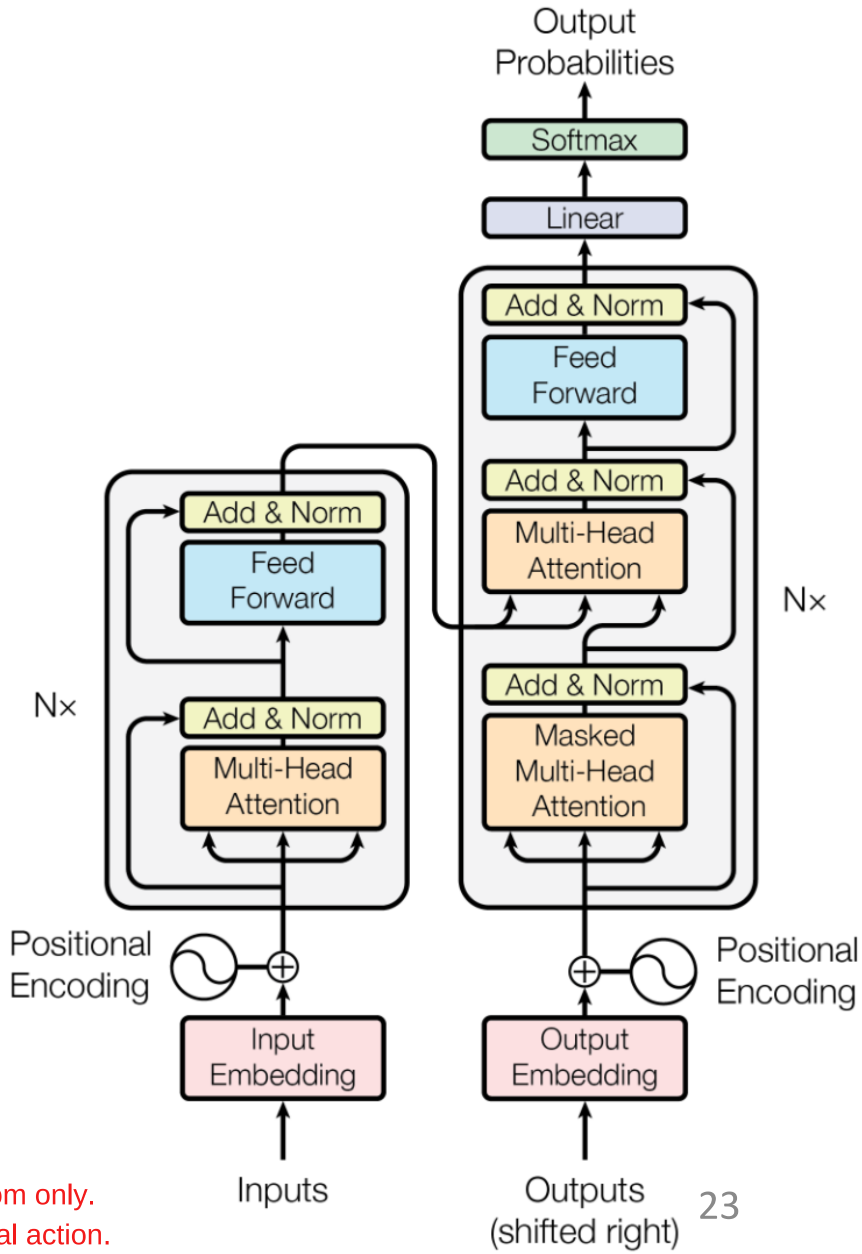
- Encoder

- Decoder



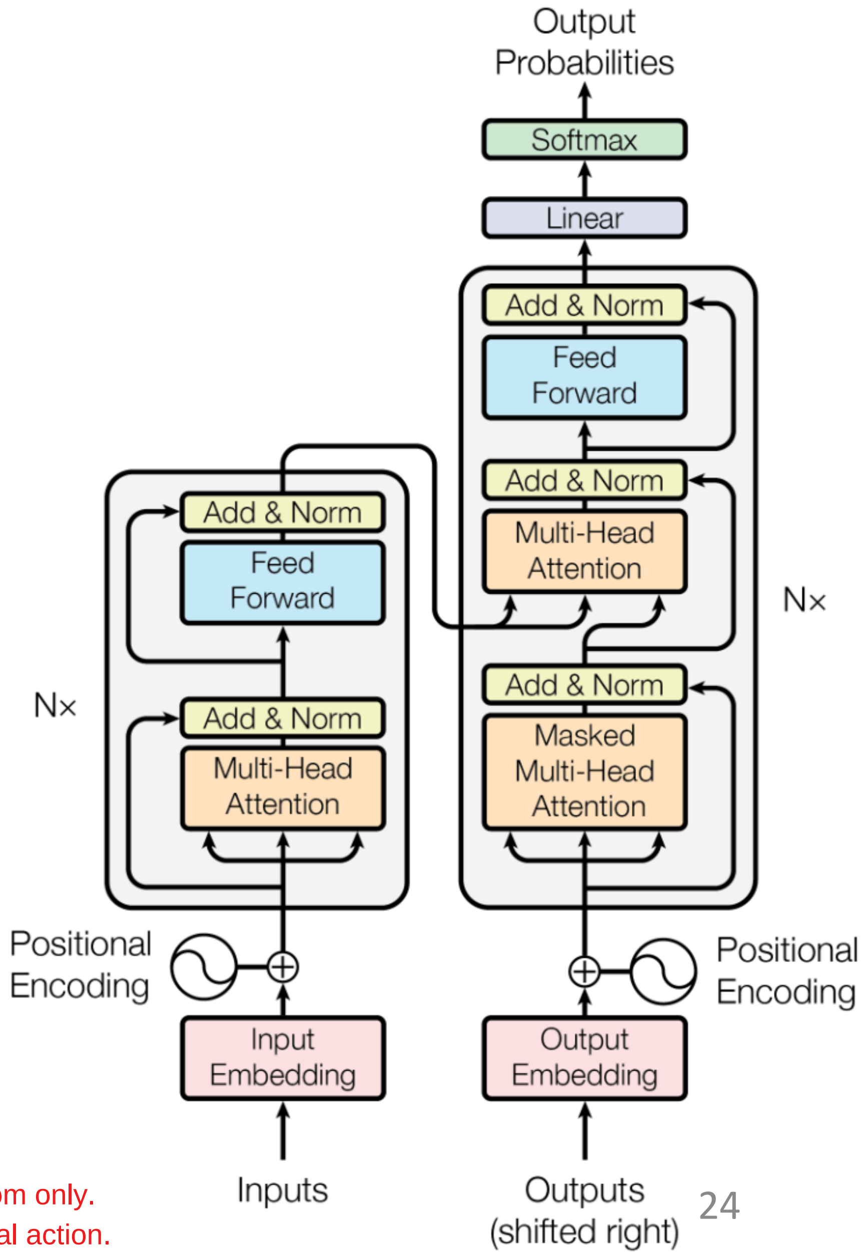
Transformer

- Encoder / Decoder



Transformer

- Encoder only
  - BERT

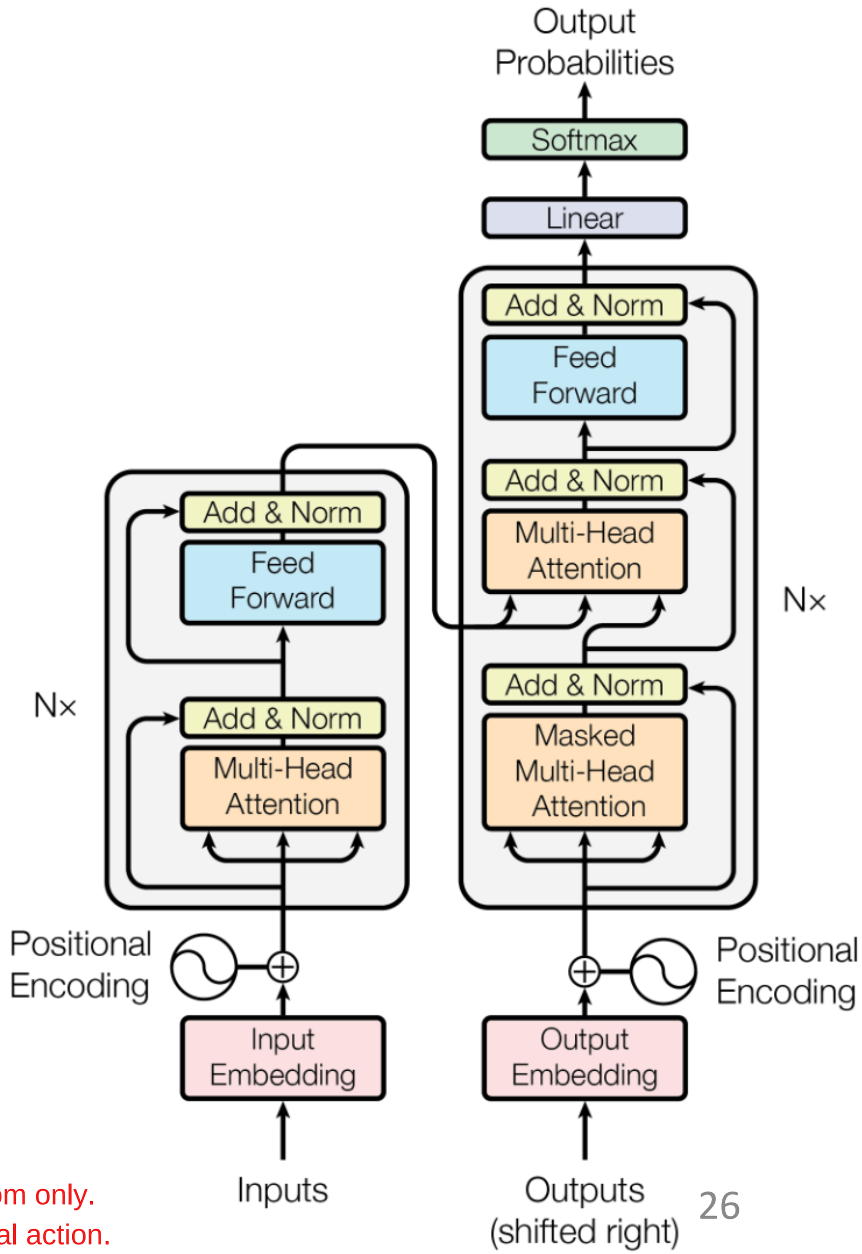


BERT Applications

- Sentiment Analysis
- Text Classification
- Semantic Search

Transformer

- Decoder only
  - GPT



GPT

- Next word prediction with just a decoder
  1. Create a block of text
  2. Input the block of text to decoder
  3. Predict probabilities of next word
  4. Randomly pick one of those words and append it to block of text
  5. Go back to 2, until the chosen word is [EOS]

Training

- Feed block of text to the transformer
- It sequentially predicts the probabilities of every possible next word after each word
- Compare predicted probabilities to true next word
- Stochastic gradient descent

- I like to go to the beach.

Fine Tuning

- Most models are pre-trained over a LONG time
- They are trained using a broad corpus of text
- You may have a task tailored to a specific text
  - Repair manuals for tractors
- You can fine tune a model using your corpus of specific text
  - You'll get better results for your task

Why Transformers?

- Recall goals
  - Contextual Awareness
  - Learned Relevance
- Multi-Head Self Attention achieves both of these
- LSTM models relevance sequentially
- Transformers let this be learned!

Applications

- Language translation (AIAYN)
- Supervised learning (BERT)
- Text Generation (GPT)

Software

- TensorFlow / Torch (2015/2016)

- Hugging Face
  - Transformers package
  - Hub

New Advances

- Multi-modal transformers
  - Personalized text to speech
  - Text to images
  - Speech to text
  - Text to videos
- GANs

Summary

- Transformers are a powerful class of neural networks
- Transformers use a special type of layer called multi-headed self attention
- Transformers are trained using HUGE data sets
- There are many applications of Transformers