School of COMPUTING
Clemson University

Introduction to Transformer Networks

Aaron J. Masino, PhD
Associate Professor, School of Computing

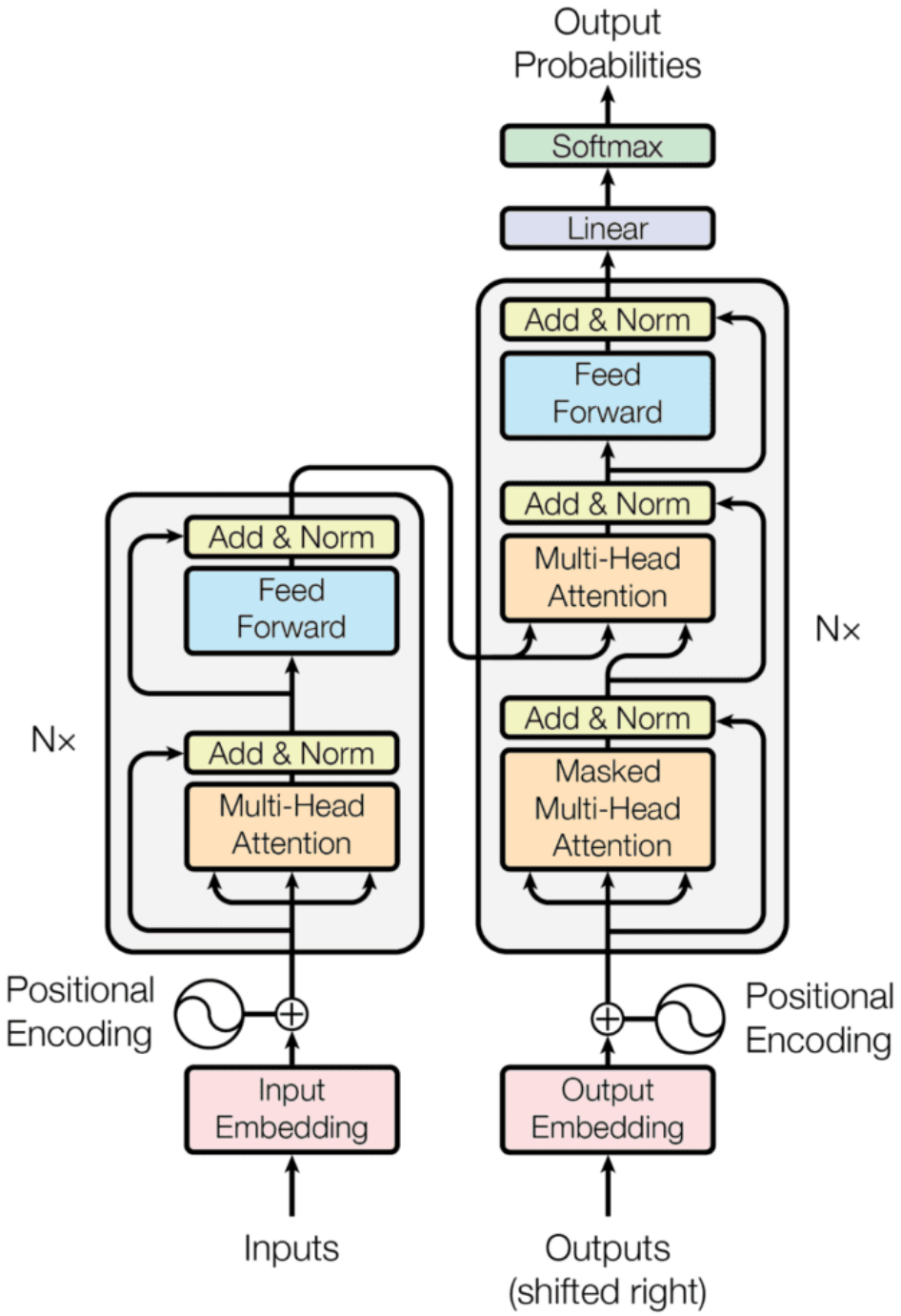
School of COMPUTING Clemson University

Outline

- Transformers

School of COMPUTING
Clemson University

Transformers

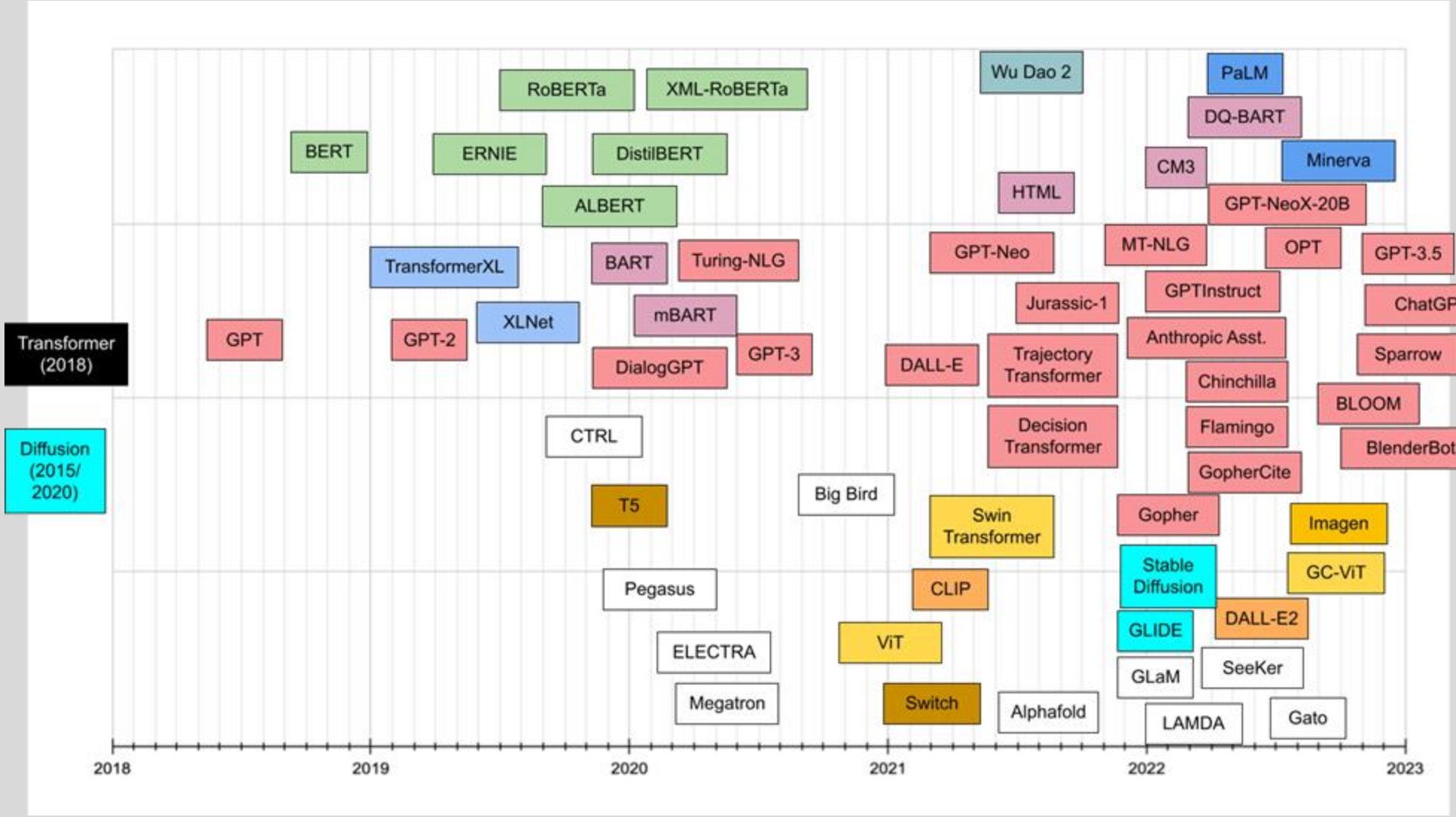


The Transformer

- Introduced by Vaswani et al.* in 2017
- Introduces concept of *self-attention*
- Partially parallelizable for improved speed
- Includes *encoder and decoder* components
  - Encoder forms a representation of the input
  - Decoder processes that to generate output
- Initial applications were NLP focused. Later applications include computer vision and time series.
- Unlike RNN models, entire text sequence is input at once

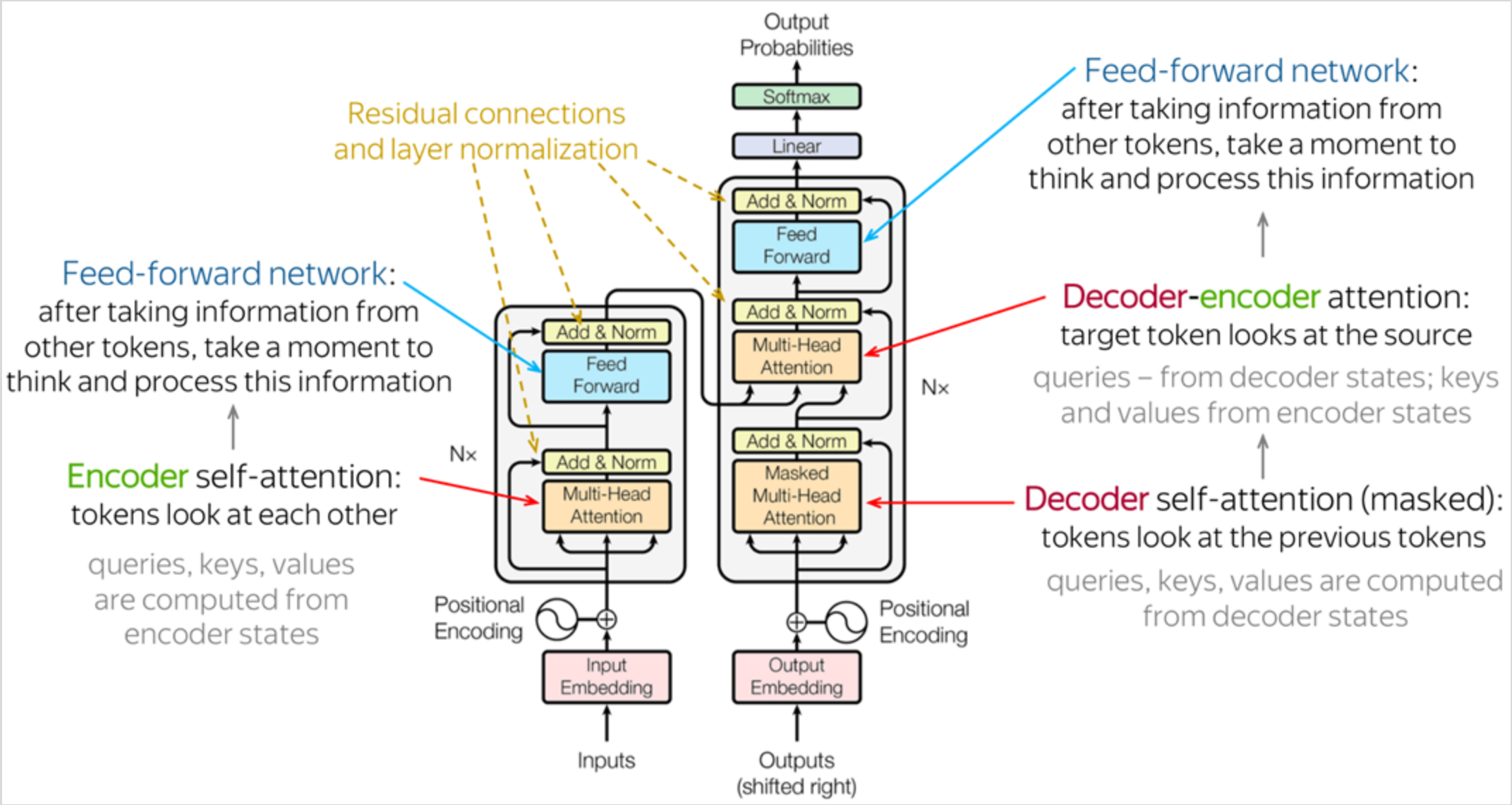
*Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

From “Attention is all you need” to GPT-X



https://amatriain.net/blog/transformer-models-an-introduction-and-catalog-2d1e9039f376/

Transformer architecture



Why encoders and decoders?

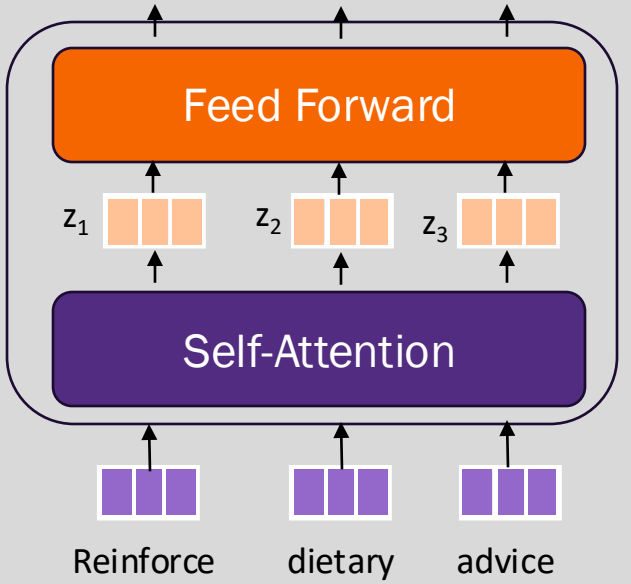
- Transformer was initially introduced for language translation (e.g., English to French)
- The encoder stack first creates a representation for every input word that depends on all other words in the input
- The decoder
  - Processes the encoder output to generate the first word of the translation
  - Processes the encoder output again combined with the current words in the translation to generate the next word in the translation
  - This is repeated until the decoder output word is the <eos> token indicating the translation is complete.

Do I need encoders and decoders for my task?

- Not all tasks require both encoders and decoders
- Encoder-Only Tasks: Sentiment Analysis, Text Classification, Named Entity Recognition (NER), Language Modeling (Contextual, masked text task)
- Decoder-Only Tasks: Text generation, language modeling (unidirectional, next word prediction)
- Encoder-Decoder Tasks: Machine Translation, document summarization, question answering, chatbots, text-to-speech

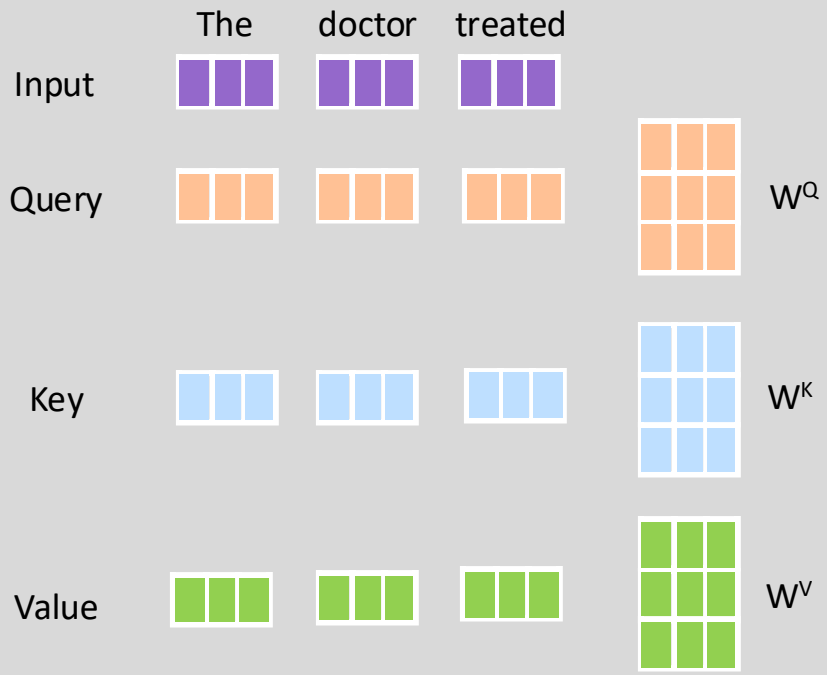
The Transformer Encoder

- An encoder is composed of one or more self-attention layers (referred to as attention heads) and a feed forward layer
- The self-attention layer processes the input:
  - Word embeddings if first encoder block. These are added to a positional embedding (not shown)
  - Output from previous encoder for subsequent blocks
- The self-attention layer outputs are processed by the feed forward layer
  - Each self-attention output is processed independently by by the same feed forward layer
  - Allows self-attention outputs to be processed in parallel



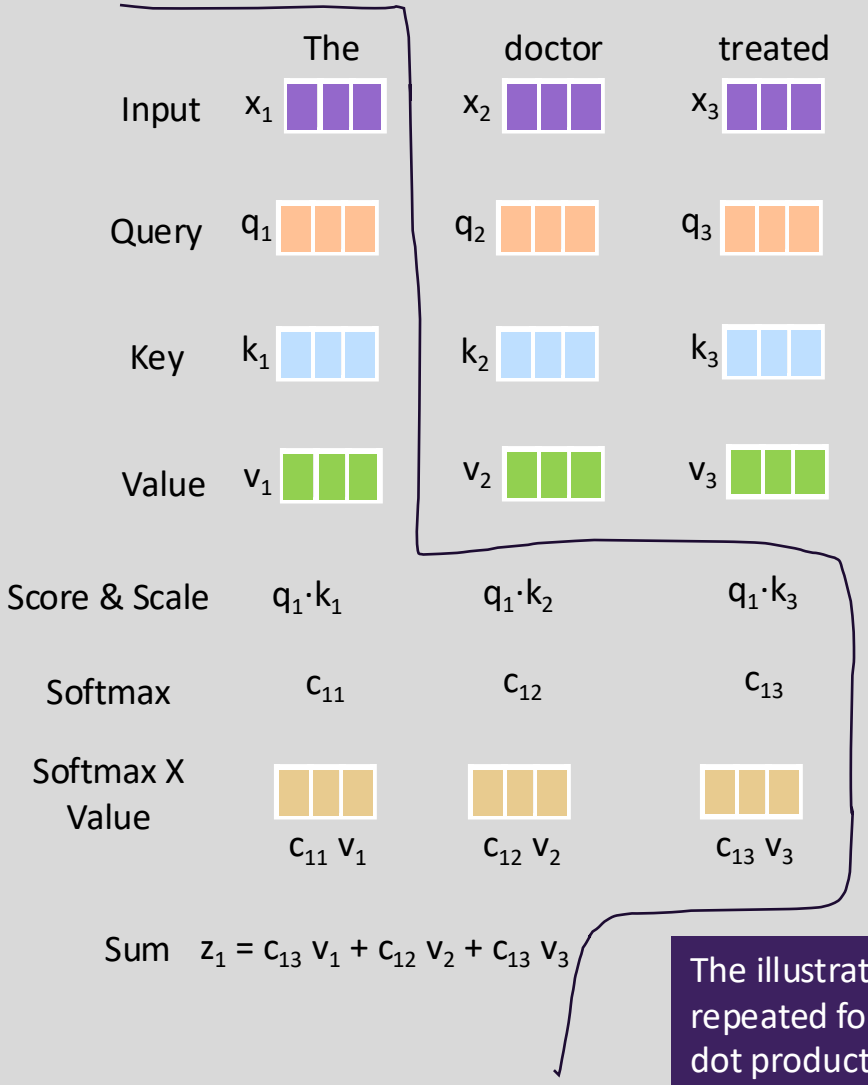
Self-attention : query, key, and values

- Seeks to dynamically modify internal input representation to account for context
- Consider the sentence:
  - *The doctor treated the patient because she was ill.*
  - Should the representation for *she* refer to the *patient* or the *doctor?*
- Utilizes *query, key*, and *value* vectors to facilitate attention
- These will be used to update each input vector relative to the amount of *attention* it should place on each other input vector



A query, key, and value vector is generated for every vector (emedding) in the input using the matrices W^Q, W^K, and W^V respectively. These attention matrices are learned during model training.

Self attention - scoring

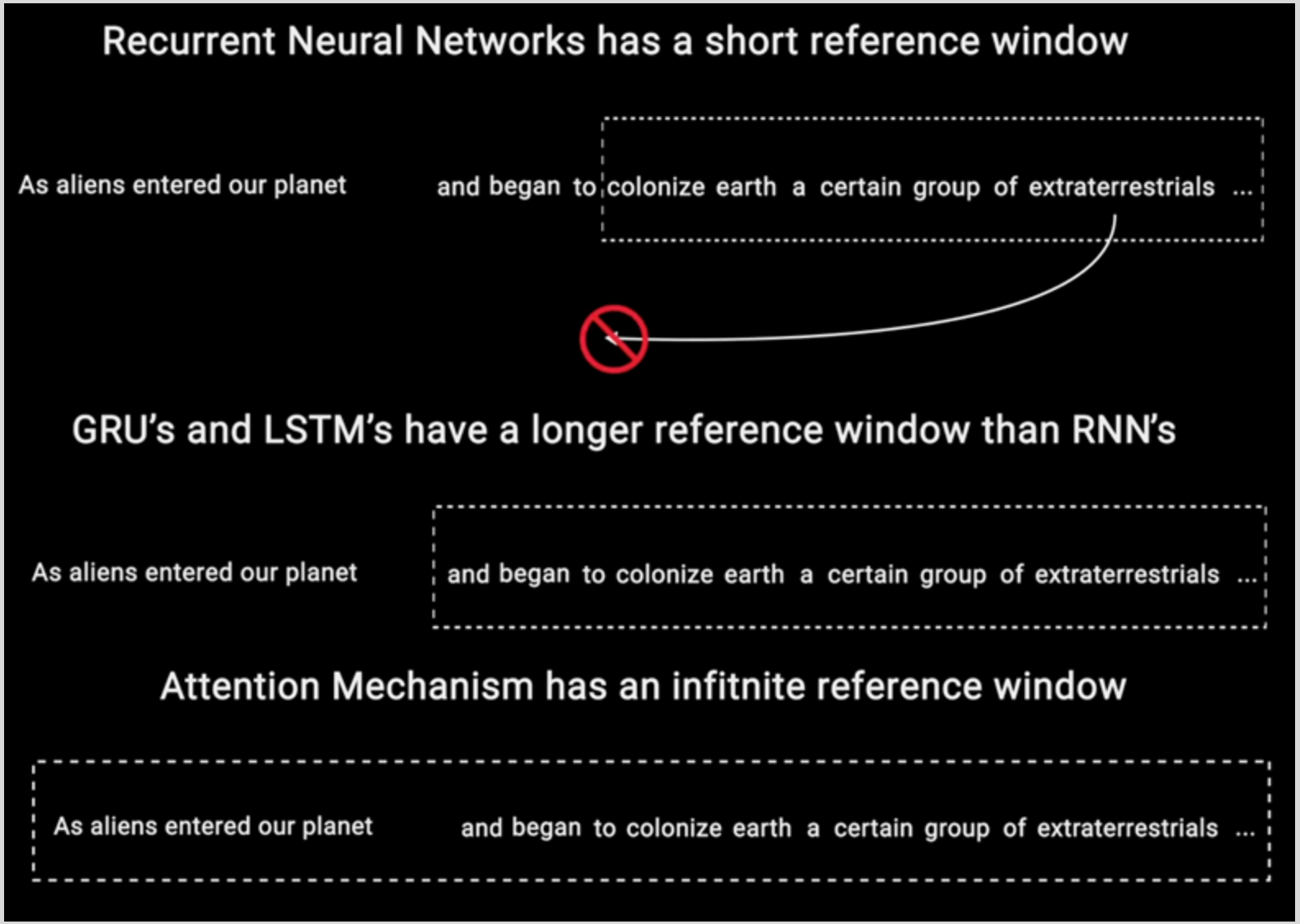


- For a given input vector (embedding), x_i, the dot product between its query vector and its own and every other vector's key vector to form a score
- The dot product scores are scaled (not shown) and then standardized via softmax
- The softmax score for each input vector is used to scale its value vector
- The scaled value vectors are added to form the output representation z_i
- This is repeated for *every* input vector

The illustrated computations are for the computation of z1. The same process is repeated for every input xi. For example, to compute z3, it is necessary to compute dot products q_3k_1, q_3k_2, q_3k_3 to obtain new softmax scores c_{31}, c_{32}, c_{33} and finally new scaled value vectors $c_{31}v_1$, $c_{32}v_2$ and $c_{33}v_3$.

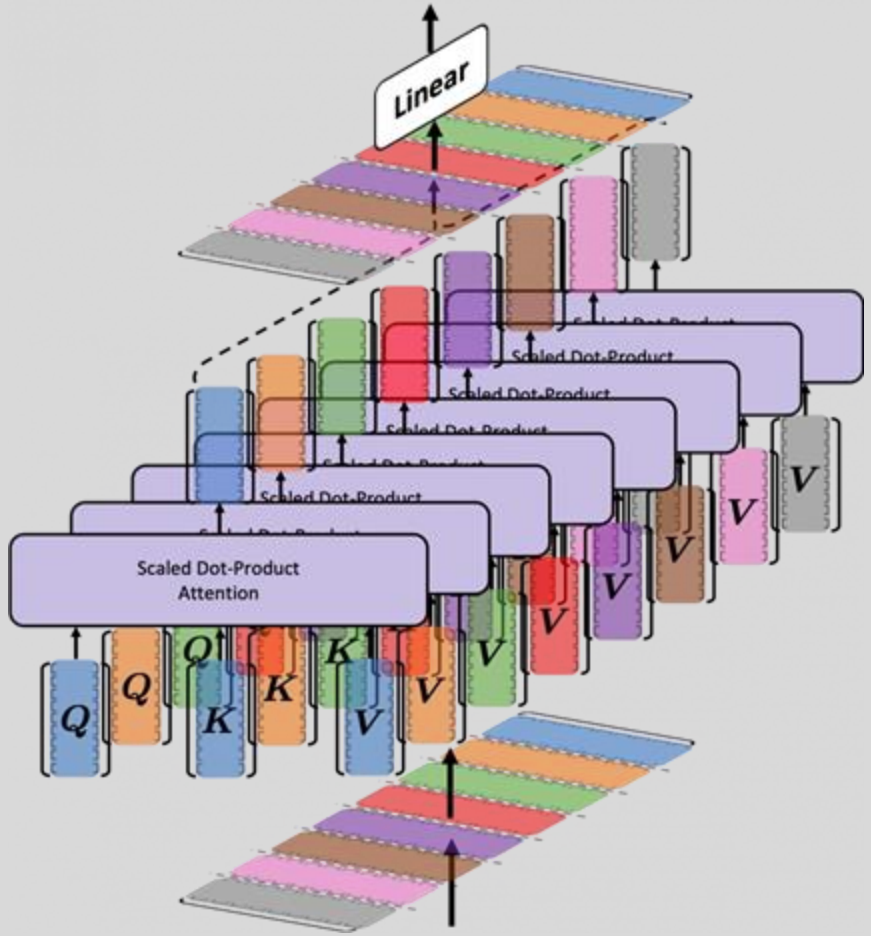
School of COMPUTING Clemson University

Complete reference window



Transformers are highly parallelizable

Every element in compared with every element, no need to do sequential comparisons. It can be processed in parallel.

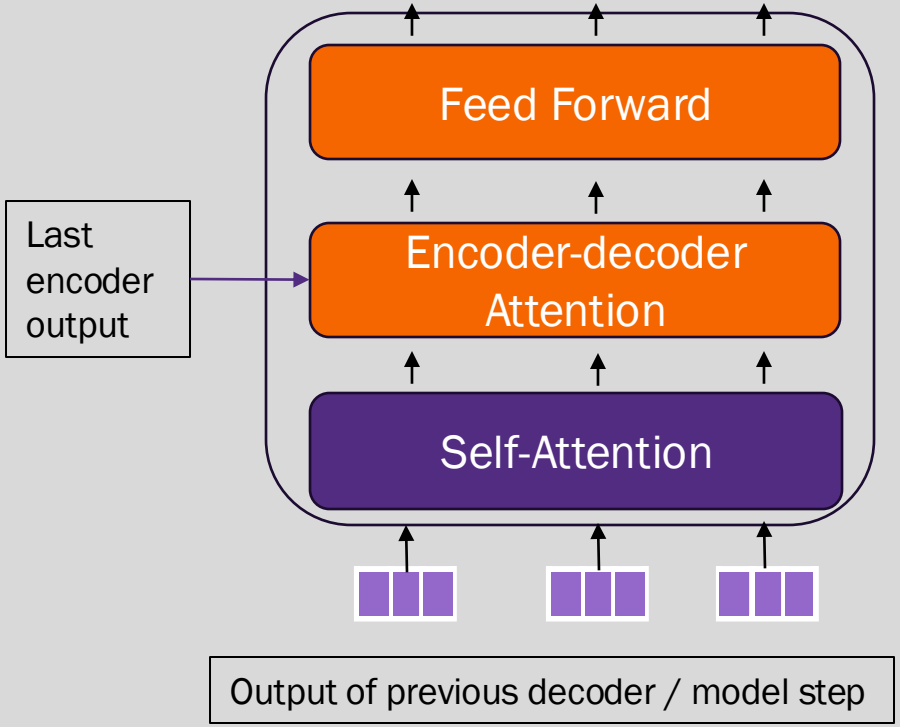


Encoder – other details

- The encoder calculations are position invariant (i.e. input order does not affect output). Positional embeddings are added to initial inputs (word embeddings) to provide position information.
- With an encoder block, the output of the self-attention layer has a residual connection, i.e., the self-attention output is added to its input and normalized before passing to the feed forward layer
- The feed forward layer also has a residual connection and normalization

The Transformer Decoder

- A decoder layer contains a self-attention block, an encoder-decoder attention block, and a feed forward block
- The self-attention blocks process the output of the previous decoder layer except for the *first* decoder for which the self-attention layer processes the previous output of the model.
- The encoder-decoder attention block processes the output of the last encoder layer combined with the output of the decoder's self-attention layer
- The feed forward block processes each of the encoder-decoder outputs
- Residual connections and normalization are used for each block



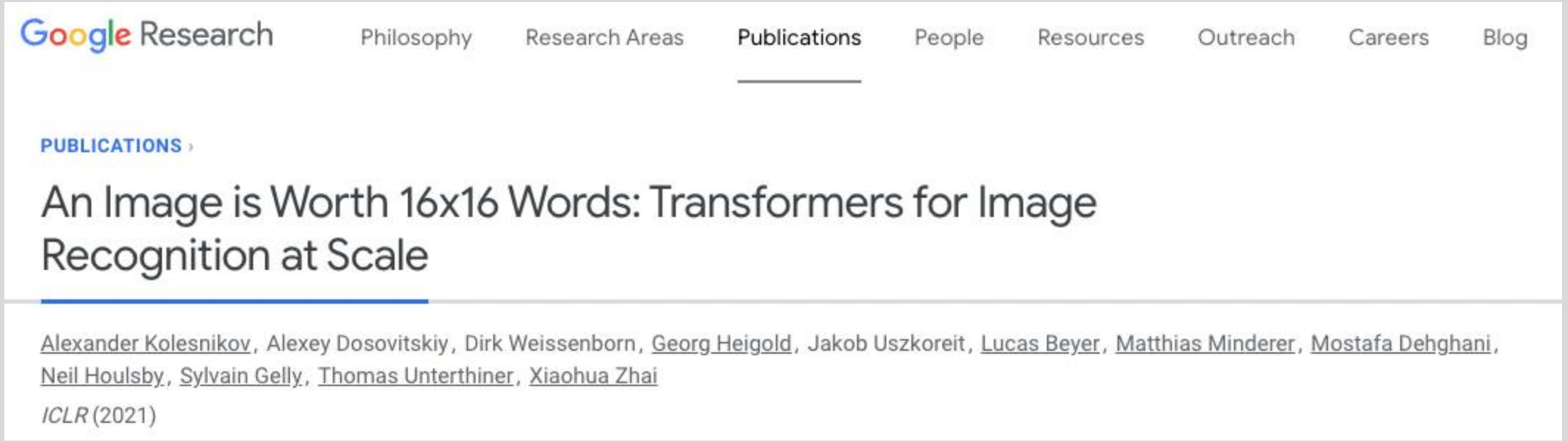
For more details on transformers for text, I highly recommend Jay Alammar's *The Illustrated Transformer*

https://jalammar.github.io/illustrated-transformer/

School of COMPUTING Clemson University

Vision Transformer

In September 2020, Google developed Vision Transformers



Vision Transformers

Segmenting the image in small fragments was enough to use the same architecture with a Multilayer Perceptron at the end.