# ARTIFICIAL NEURAL NETWORKS - WEEK 12

Transformers

Dr. Aamir Akbar

Director of both AWKUM AI Lab and AWKUM Robotics, Final Year Projects (FYPs) coordinator, and lecturer at the department of Computer Science
Abdul Wali Khan University, Mardan (AWKUM)

# CONTENTS

# TRANSFORMERS

## What is a Transformer ANN?

Transformers are a type of artificial neural network architecture that excel at handling sequences of data, making them particularly effective for tasks such as language translation, text generation, and even image processing and audio processing. At the core of the Transformer model is the attention mechanism that allows the model to weigh the importance of different parts of the input data simultaneously. This enables the model to focus on relevant parts of the input for each step of the output generation.

In 2017, the Google Research team published a paper called "Attention Is All You Need", which presented the Transformer architecture and was a paradigm shift in Machine Learning, especially in Deep Learning and the field of natural language processing.

The presented architecture served as the foundation for subsequent models like GPT, DALL-E and Gemini.

# TRANSFORMERS: ENCODER-DECODER ARCHITECTURE

The Transformer uses an encoder-decoder structure:

**Encoder:** Processes the input data and generates a set of features or representations.

**Decoder:** Uses these representations to generate the output data. Both the encoder and the decoder consist of multiple layers, each containing sub-layers for attention and feed-forward neural networks.

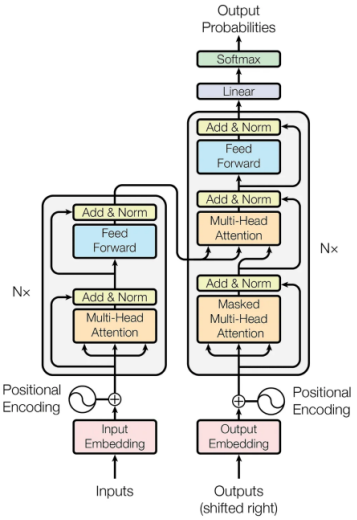
# TRANSFORMERS: SELF-ATTENTION

Self-attention is a mechanism that allows the model to consider other words in the input sequence when encoding a particular word. It calculates a weighted average of all words in the sequence to generate a new representation of each word.

Steps in Self-Attention:

1. Input Representation: Words are converted into vectors.
2. Key, Query, Value: Each word vector is transformed into three vectors: Key, Query, and Value.
3. Dot Product: Compute the dot product of Query and Key to determine the relevance.
4. Softmax: Apply softmax to get attention scores.
5. Weighted Sum: Multiply attention scores by Value vectors to get the final representation.

# TRANSFORMERS ARCHITECTURE



# ENCODER

**Input Embedding:** Convert each word in the input sequence into a dense vector.

**Positional Encoding:** Since the Transformer does not inherently understand the order of the sequence, positional encodings are added to the input embeddings to provide information about the position of each token in the sequence.

**Multi-Head Self-Attention:** This mechanism allows the model to focus on different parts of the input sequence simultaneously. It helps in understanding the context better by considering different positions in the sequence at once.

**Feed-Forward Network:** Apply a feed-forward neural network to each position.

**Output:** Pass the processed sequence to the next encoder layer or to the decoder.

# DECODER

**Input Embedding and Positional Encoding:** Similar to the encoder, convert and positionally encode the target sequence.

**Masked Multi-Head Self-Attention:** Prevents the decoder from attending to future tokens in the sequence, ensuring that the prediction for each position depends only on known outputs up to that position.

**Multiple Heads:** Instead of performing a single attention function, the model uses multiple attention heads. This allows it to capture different aspects of the relationships between words.

**Feed-Forward Network:** Apply a feed-forward neural network.

**Output:** Generate the next token in the sequence.

# RESOURCES

To download the source codes used in the previous slides, follow the link:

▸ Download Source Codes

Import the codes into your preferred development environment, such as Visual Studio Code (VS Code), to practice and explore further.

To learn programming in Python, follow my comprehensive 15-week Programming in Python course at:

▸ Programming in Python