# Module 3

**Transformers in Keras**

**Introduction to Transformers in Keras**

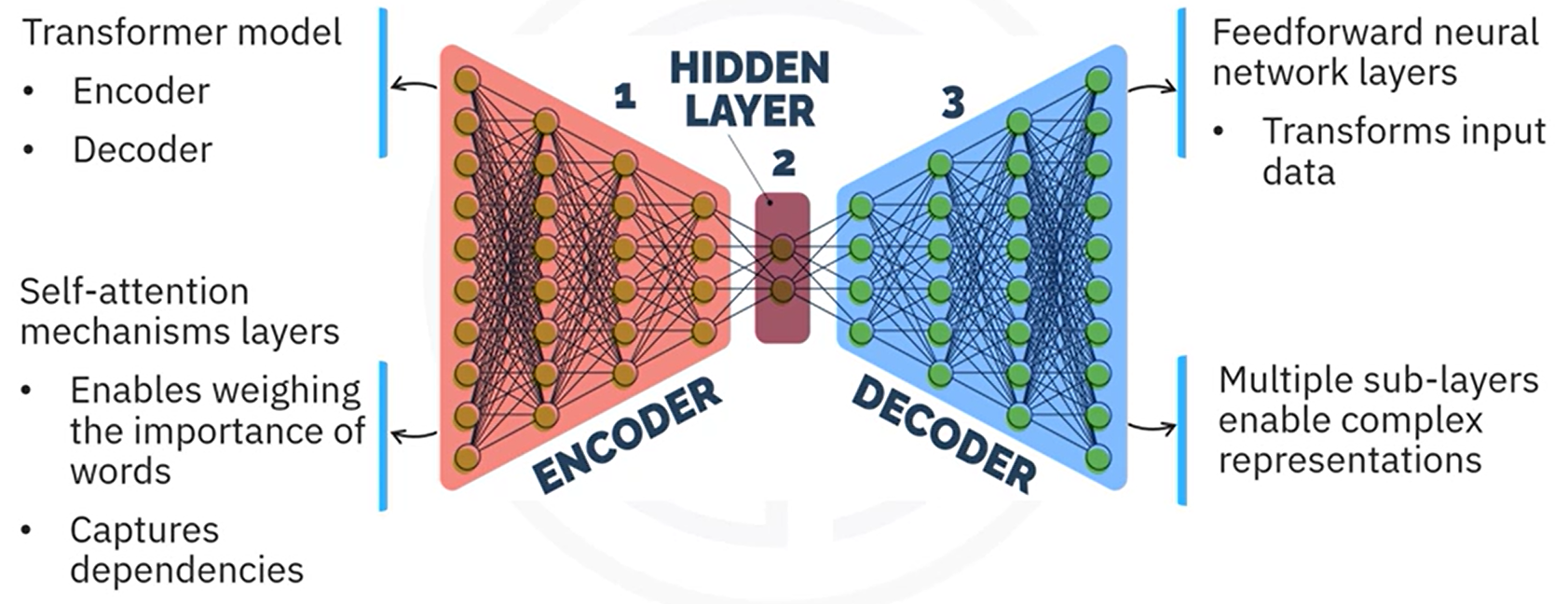
## 📌 Introduction to Transformers in Keras

Transformers have revolutionized deep learning, particularly in **natural language processing (NLP)**, and are now applied to a broader range of domains such as **image processing**, **time series prediction**, and other sequence modeling tasks.

First introduced by **Vaswani et al.** in the seminal paper *"Attention is All You Need"*, transformers utilize **self-attention mechanisms** to process data in parallel, overcoming the sequential limitations of earlier models like RNNs.

Today, transformers form the backbone of state-of-the-art models, including **BERT, GPT**, and many more.

### 🔹 Transformer Architecture Overview



The transformer model consists of **two main components**:

|  |  |
| --- | --- |
| **Encoder** | **Decoder** |
| Responsible for processing the input sequence.  Contains multiple layers, each with:   * Self-attention mechanism * Feed-forward neural network * Residual connections and layer normalization. | Generates de output sequence.  Contains multiple layers, each with:   * Self-attention mechanism * Cross-attention mechanism (attending to encoder output) * Feed-forward neural network * Residual connections and layer normalization |

**Transformers components explanation:**

**🔹 Feed-Forward Neural Network**

The encoder and decoder apply a **feed-forward neural network** in each layer after the attention mechanisms.

* The feed-forward network processes each position independently.
* It consists of two fully connected layers.
* It applies an activation function between the layers to introduce non-linearity.

**🔹 Residual Connections and Layer Normalization**

* The model applies **residual connections** after self-attention and feed-forward sub-layers to help stabilize training.
* It applies **layer normalization** after each residual connection to normalize the layer outputs and maintain stable gradients.

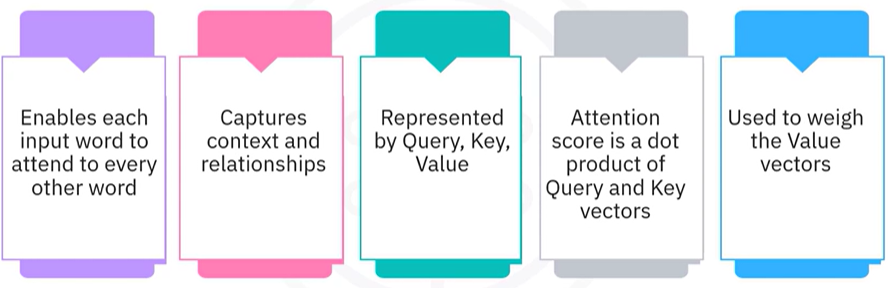
**🔹 Positional Encoding**

The model uses **positional encoding** to inject position information into the input embeddings.

* It applies positional encoding immediately after embedding the input sequence.
* This allows the model to understand the order of the tokens while processing the sequence in parallel

#### 🧠 Key Components Explained

##### **⚡Self-Attention Mechanism**



Self-attention allows the model to focus on **different words within the same sequence** when encoding a particular word, enabling the capture of **long-range dependencies** regardless of their position in the sequence.

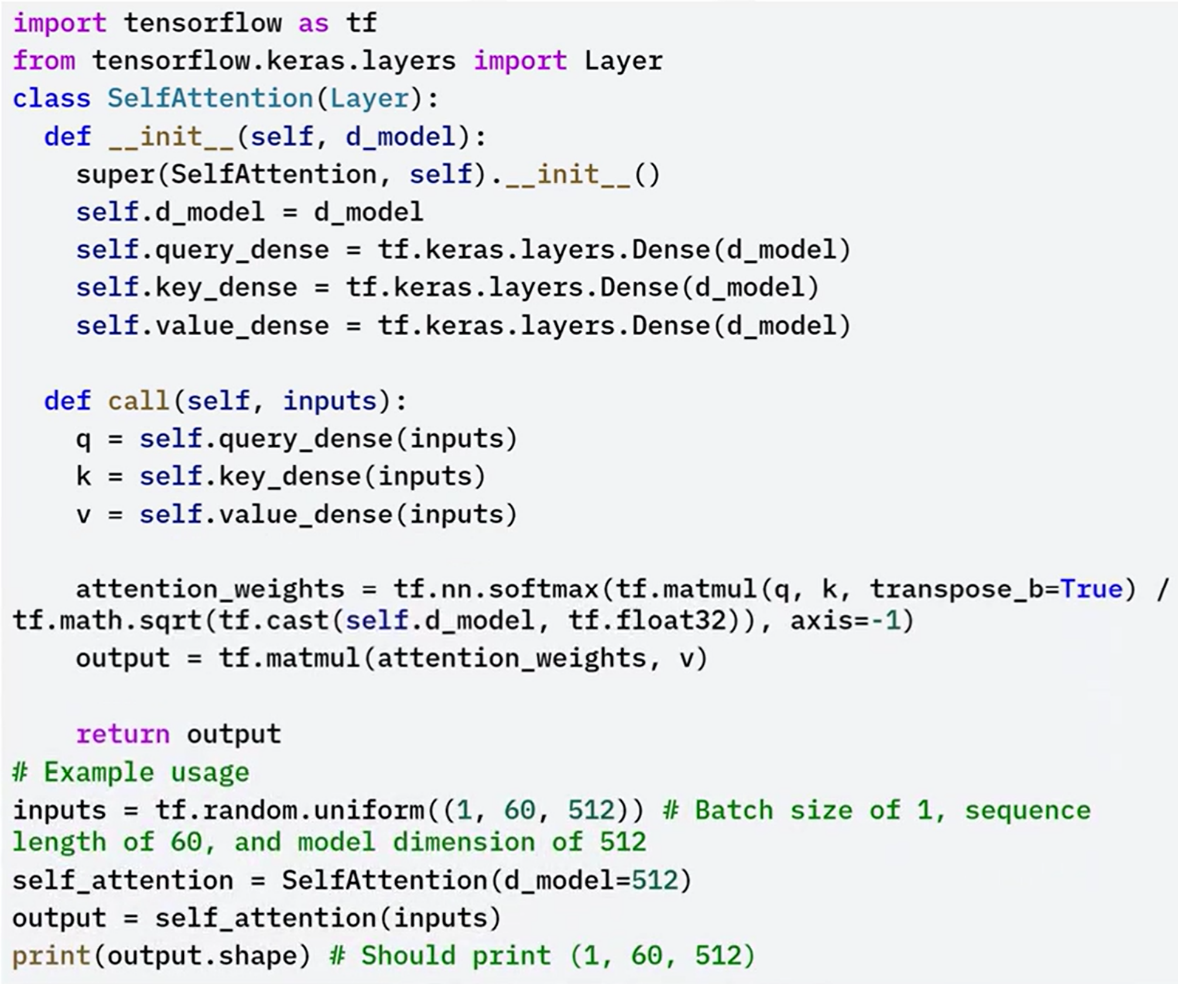
Each token is represented by three vectors:

* **Query (Q)**
* **Key (K)**
* **Value (V)**

The attention score is calculated as the **dot product of the Query and Key vectors**, followed by a **softmax activation** to generate attention weights. These weights are applied to the **Value vectors**, resulting in context-aware outputs.

This mechanism ensures that each word attends to **every other word in the sequence**, enriching contextual understanding.

**Example of Self-Attention Calculation:**



In this code example, the self-attention class defines the self-attention mechanism.

The **init method** initializes the dense layers for query, key and value projections.

 The **call method** calculates the attention scores, applies softmax normalization, and computes the output by applying the attention weights to the value vectors.

The **tf.nn.softmax parameter** applies the Softmax function to the attention scores to get the attention weights.

##### **⚡Transformer Encoder**



The **transformer encoder** is composed of multiple layers, each containing:

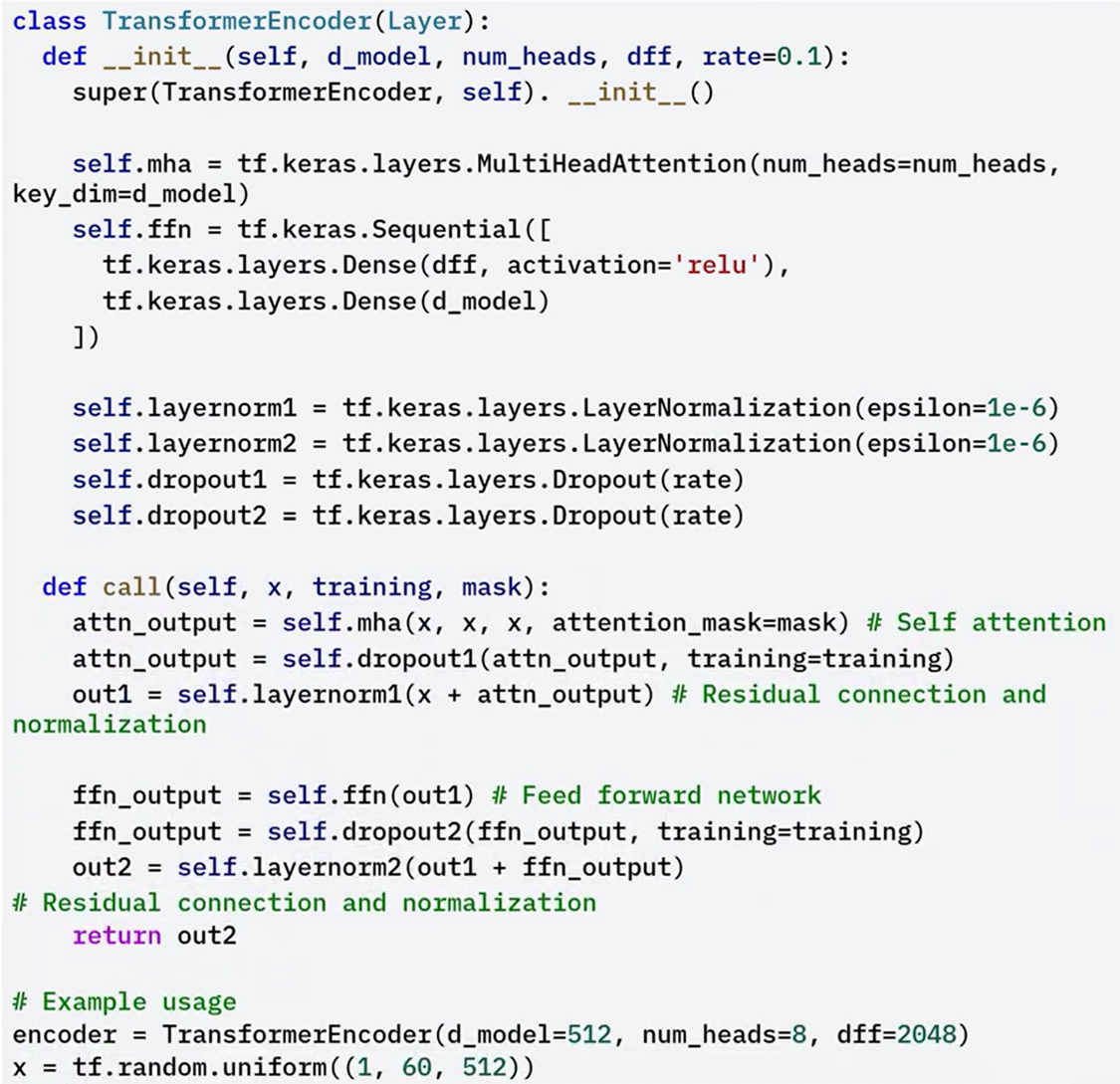
* **Self-attention mechanism**: Enables the encoder to model relationships between all tokens in the input sequence.
* **Feed-forward neural network**: Further transforms the representations after self-attention.
* **Residual connections and layer normalization**: Applied after both self-attention and feed-forward sublayers to stabilize and accelerate training.

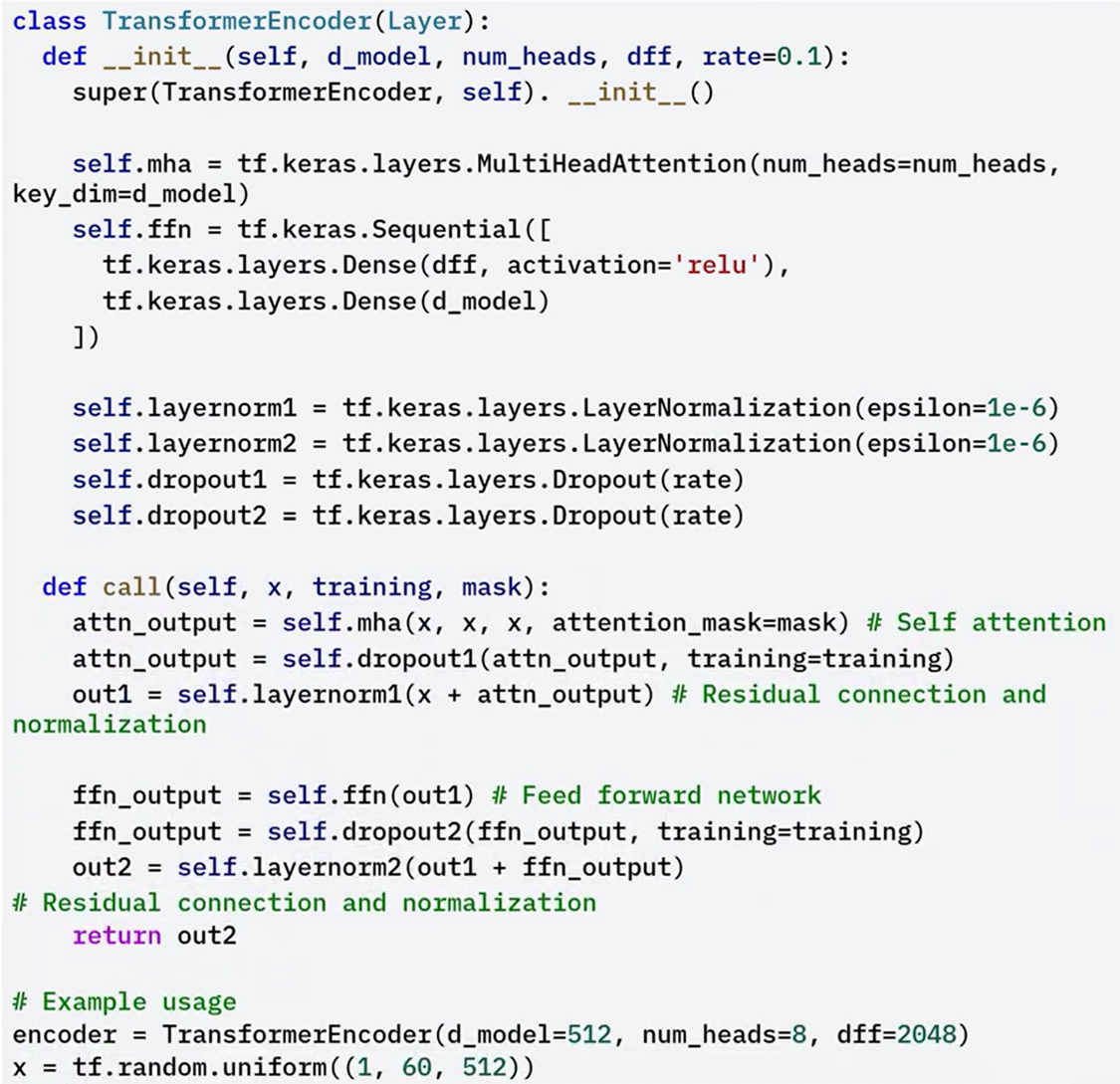
The input to the encoder is first embedded and then passed through

positional encoding to add information about the position of words in the sequence.  This helps the model understand the order of words.

Each encoder layer learns progressively more complex representations of the input sequence, capturing both local and global relationships.

**Example of Transformer Encoder:**





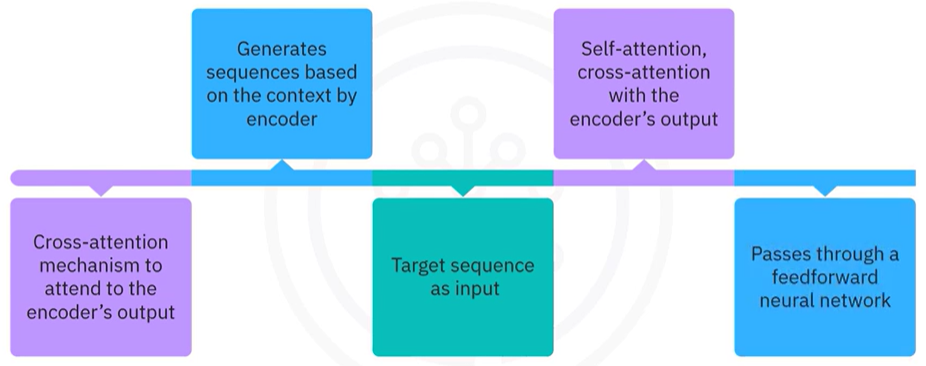
In the example, the transformer encoder class defines the transformer encoder.

The **init** **method** initializes the multi head attention, feed forward network, layer normalization, and dropout layers.

The **multi head attention** method applies multi head attention to the input.

The **call method** applies self-attention, residual connection, normalization, feed forward network, and other residual connection and normalization.

##### **⚡Transformer Decoder**

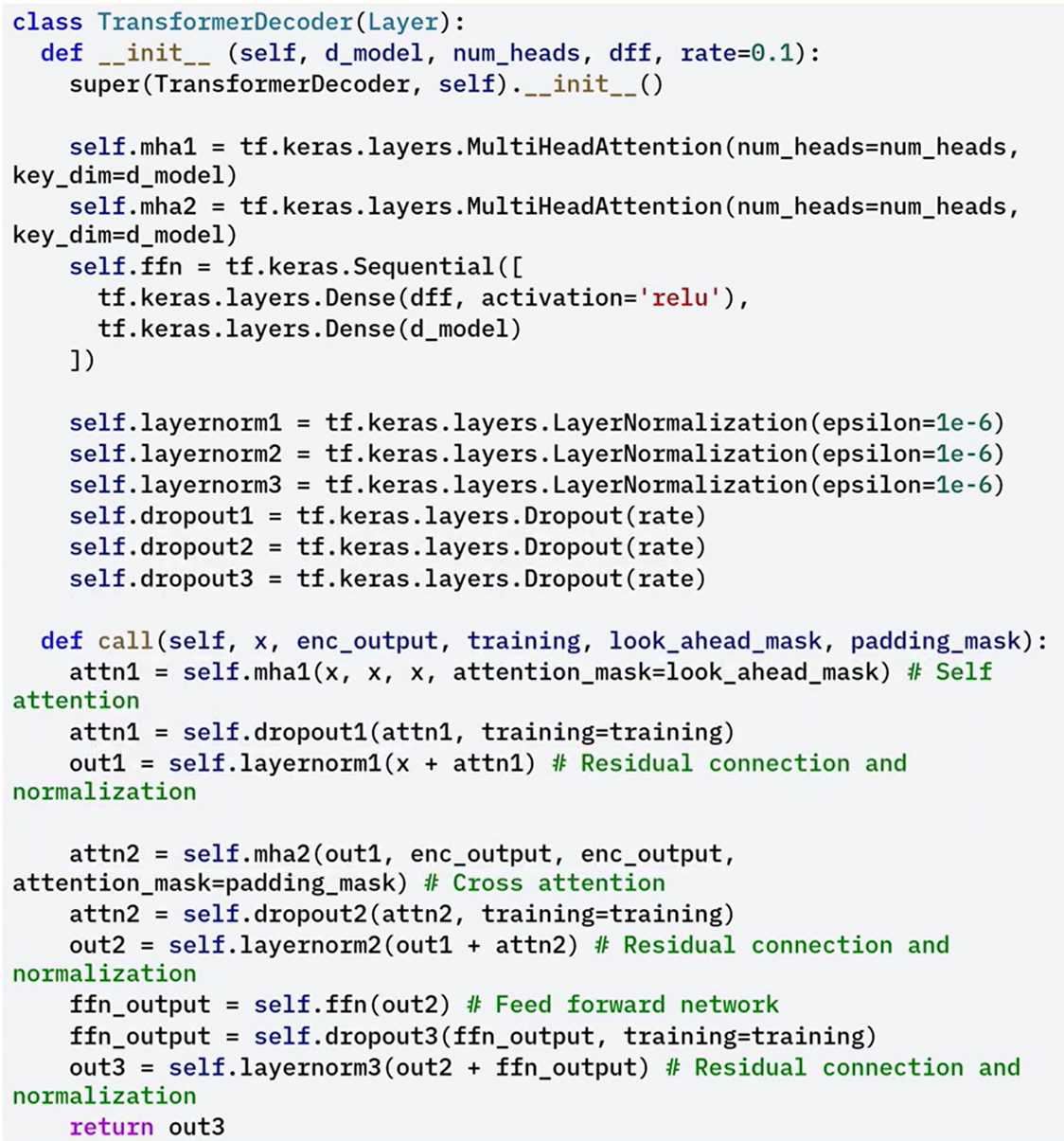


The transformer decoder is similar to the encoder, but with an additional cross attention mechanism to attend to the encoders output.

* The decoder takes the target sequence as input.
* The input passes through an embedding layer and positional encoding.
* The decoder applies **self-attention** to the target sequence, followed by a **residual connection and layer normalization**.
* It then applies **cross-attention** where the decoder attends to the encoder’s output.
* This allows the decoder to use context from the encoder while generating output tokens.
* After cross-attention, the result is passed through another **residual connection and layer normalization**, followed by a **feed-forward neural network**, and another **residual connection and layer normalization**.

This design allows the decoder to generate sequences based on the context provided by the encoder output and the previously generated tokens.

**Example of Transformer Decoder:**



The **init** method initializes the multi head attention, feed forward network, layer normalization, and drop out layers.

The **multi head attention** method applies multi head attention to the input and encoder output.

The **call** method applies self-attention, cross attention, residual connection, normalization, feed forward network, and another residual connection and normalization.

### ✅ Takeaways

✅ Transformers are composed of an encoder and a decoder, both built from layers that include self-attention mechanisms and feed-forward neural networks.

✅ Self-attention allows each word to attend to every other word in the input, enabling the model to capture relationships and dependencies across the entire sequence.

✅ The encoder processes the input by embedding the sequence, applying positional encoding, and passing it through layers that perform self-attention, feed-forward operations, residual connections, and layer normalization.

✅ The decoder processes the target sequence by applying self-attention, cross-attention with the encoder's output, and feed-forward networks, all followed by residual connections and layer normalization.

✅ Understanding and implementing transformers enables the development of powerful models for natural language processing, image processing, and time series prediction.