

Neural Network Architectures

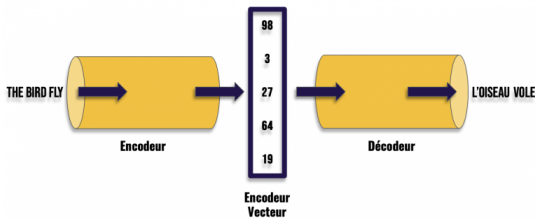
Marie Beurton-Aimar

November 5, 2025

Deep Learning not only for Image Processing

Convolution is not the only to work

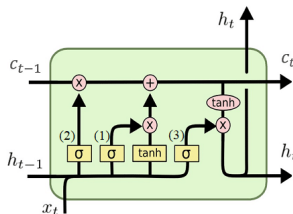
- For example to translate a sentence ?
- Keeping the idea of encoding/decoding the information.
- A sequence of words will be translate in numbers.
- This sequence will be recoded as words.



Text Analysis

LSTM

- Long Short Time Memory - A kind of Recurrent Neural Network.
- Keep in mind several tokens.
- LSTM introduces a cell memory
- 3 gates in each cell memory: input, output and forget.
- Solve the problem of the vanishing gradient problem of RNNs: after a lot of backpropagations gradients become smaller and smaller and lead to negligible modifications of weights.



LSTM in Use

Applications

- Automatic translation. Able to take into account a sequence of words and their global context.
- Can be used:
 - to generate text (generative AI)
 - to speech recognition.
 - to predict temporal sequences and/or classify sequences.

Limitations

- More difficult to train because more parameters.
- Need more computing resources.
- Sensitive to overfitting.

Large Language Models

Concepts

- Transformer-based network dedicated to Natural Language Processing (NLP) and Natural Language Generation (NLG).
- Goal: being able to learn human language complexity.
- Pre-trained with a huge quantity of data: images, video, speech...

Go back from the beginning

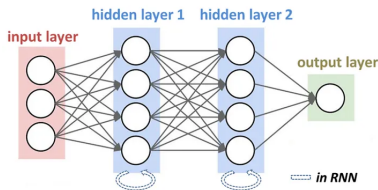
RNNs

- Build to solve problem where sequence is more important than the individual elements.
- Use to analyze sequential data.
- Differ from traditional feedforward neural networks by using a memory allowing to process sequences of data considering relationships between previous and current inputs.

Recurrent Network

Architecture

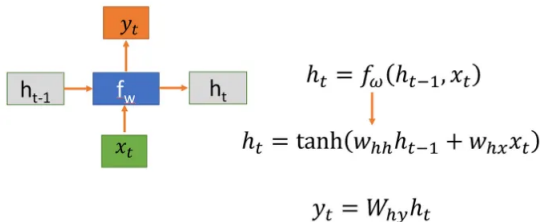
- Input and Hidden Layers:
 - Each RNN layer has a hidden state to capture information from previous time steps.
 - This hidden state is updated continuously with the new sequence.
- Recurrent Connection:
 - The feedback loop is the recurrent structure enables RNNs to process sequences of any length.
- Output Layer:
 - Many to One or Many to Many, for example for machine translation.



Recurrent Network

How they work?

- Each element in the **Input Sequence** is processed one at a time.
- The **hidden state** is updated by combining the previous hidden state and the current input.
- Based on the updated hidden state, an **output** is generated.
- Computing of each hidden state:



LSTM

Why a new version?

- Design to combat the vanishing gradient problem.
- Introduce gates to control the flow of information and to select the important one.

Architecture

- **Input Gate** to determines which new information to store.
- **Forget Gate** to decide which information to remove from the cell state.
- **Output Gate** to decide the information to output.
- Improvement: **Gate Recurrent Unit (GRU)** combines the forget and the input gates into a single one.

LSTM

Applications

- Natural Language Processing: machine translation, sentiment analysis able to capture the relationship between words.
- Speech Recognition: processing speech sequences, converting audio signals into text (using temporal patterns of sounds).
- Time-Series Forecasting: predict time-series data such as stock prices, weather patterns and sales forecasting.
- Video Processing: process video sequences to recognize objects or activities, can be combined with CNNs to analyze images.

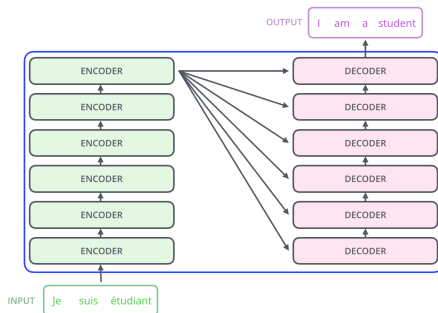
Limitations

- Difficulty with long sequences: like RNNs struggle with very long dependencies for example in long-text generation.
- High computational cost: sequential nature makes parallelization difficult.
- Replaced by **Transformers**

The Transformer

Purpose

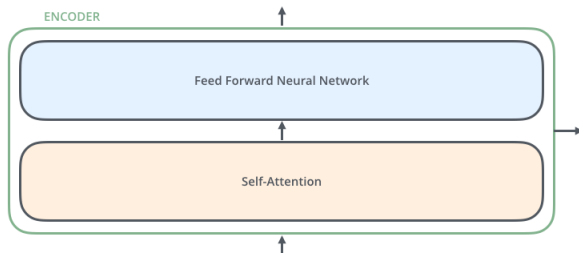
- Strongly linked to the concept of Attention.
- Use the main idea of Encoder/Decoder.
- Both Encoder and Decoder can be a stack of each.



The Transformer

Architecture

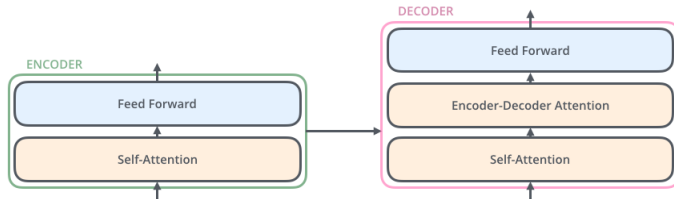
- The Encoders are all identical in structure.
- Each has two sub-layers.
- Inputs first flow through a Self-Attention layer.
- The outputs of Self-Attention layer feed a fFeed-Forward neural network.



The Transformer

Architecture

- Decoder adds an Encoder-Decoder Attention between Self-Attention and Feed Forward layers.

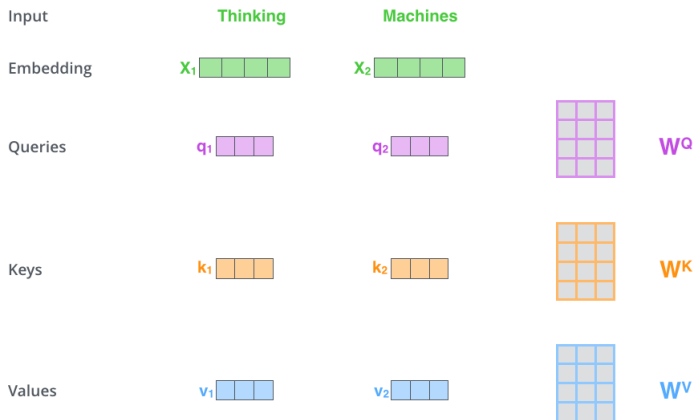


Attention Module

Computing details

- Each word consists on an input.
- The first step is to create the 3 vectors from each encoder's input vector: Query vector, Key vector and Value Vector, using 3 matrices trained during the training process.
- The second step is to calculate a score for each input. Example: score Thinking against Machines. Score is obtained by doing the dot product of the query vector with the key vector.

Attention Module



Attention Module

Input

Embedding


Queries

Keys

Values

Score

Thinking

x_1 

q_1 

k_1 

v_1 

$q_1 \cdot k_1 = 112$

Machines

x_2 

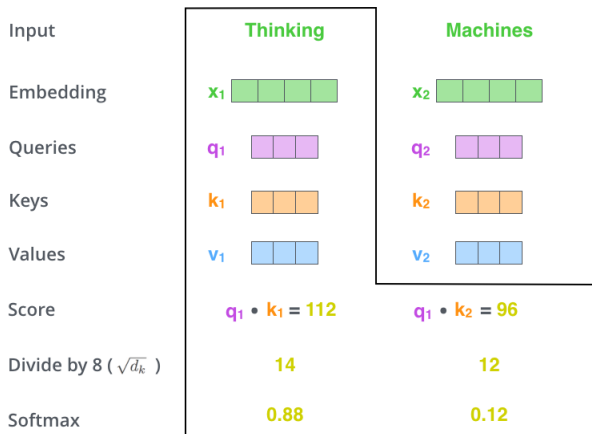
q_2 

k_2 

v_2 

$q_1 \cdot k_2 = 96$

Attention Module

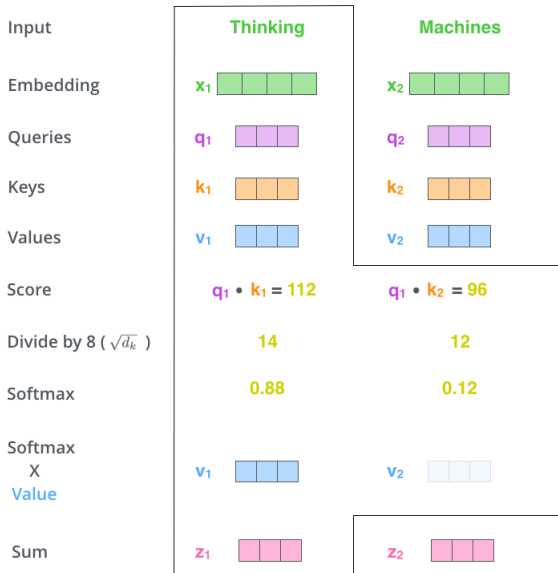


Attention Module

Computing details

- To obtain a more stable gradient, some operations are applied on the resulting score : here division by 8 because the vector size is 64 and 8 is the square root. Secondly a softmax operation allows to determine which word is the best at this position.

Attention Module



Attention Module

Computing details

- The last steps consist of multiplying each value vector by the softmax score and to sum up the weighted value vectors to produce output of the attention module.
- Take a look on: <https://jalammar.github.io/illustrated-transformer/>

Self Attention

- In a self attention module, only one sequence is used and the score of each word depends on the other words in the same sentence.
- For example in the sentence :
The animal did not cross the street because it was too tired.
- In this sentence the goal is to find the link between animal and it.
- In this example:
The animal did not cross the street because it is too full of cars.
- The link to find is between street and it.

Transformers

Models

- BERT (Bidirectional Encoder Representations from Transformers): Text classification, sentiment analysis, question answering, named entity recognition.
Characteristic: Focuses on understanding the context of a word in a sentence by looking at both its left and right context.
- GPT (Generative Pre-trained Transformer): Text generation, summarization, translation.
Characteristic: Uses the Transformer architecture's decoder to generate coherent and contextually relevant text.
- ViT (Vision Transformer): Image classification, object detection.
Characteristic: Adapts the Transformer architecture for processing images, treating image patches as sequence data.

Large Language Models

Description

- LLMs: Can be based on different architectures, but many modern LLMs utilize the Transformer architecture to achieve state-of-the-art performance.
- Used for a wide range of NLP tasks, from text generation and summarization to translation and sentiment analysis.
- Require massive datasets and computational resources for training.
- Generate human-like text, making them suitable for applications that require natural language understanding and generation.

Large Language Models

Tasks

- Natural Language Generation: Generating content for blogs, answering customer queries in natural language.
- Language Understanding: Analyzing social media posts to gauge public opinion.
- Text Completion and Suggestion: Predicting the next word in a sentence or suggesting code snippets in an IDE.
- Question Answering: Creating an AI assistant that answers queries about specific topics like legal advice or tech support.

Large Language Models

Models

- GPT-3 (Generative Pre-trained Transformer 3): Text generation, dialogue systems, content creation, coding assistance.
Characteristic: A massive model with 175 billion parameters, capable of generating highly coherent and contextually relevant text.
- BERT (Bidirectional Encoder Representations from Transformers):
Though primarily a Transformer, BERT is also used in language understanding tasks as part of larger LLM systems.
Characteristic: Focuses on understanding rather than generating text, often used in tasks requiring contextual understanding.

Large Language Models

Models

- LLaMA (Large Language Model Meta AI): Text generation, conversation agents, and research in AI language capabilities.
Characteristic: A family of LLMs developed by Meta, designed for efficiency and versatility in natural language understanding and generation.
- Turing-NLG (Natural Language Generation): Text generation, conversational AI, creative writing.
Characteristic: Developed by Microsoft, it's one of the largest models for natural language generation, with 17 billion parameters.