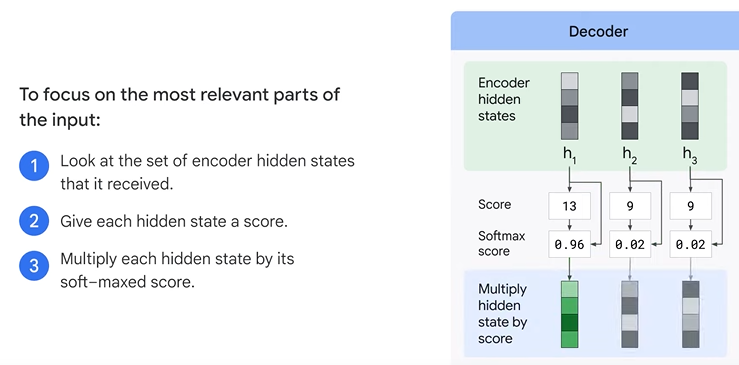
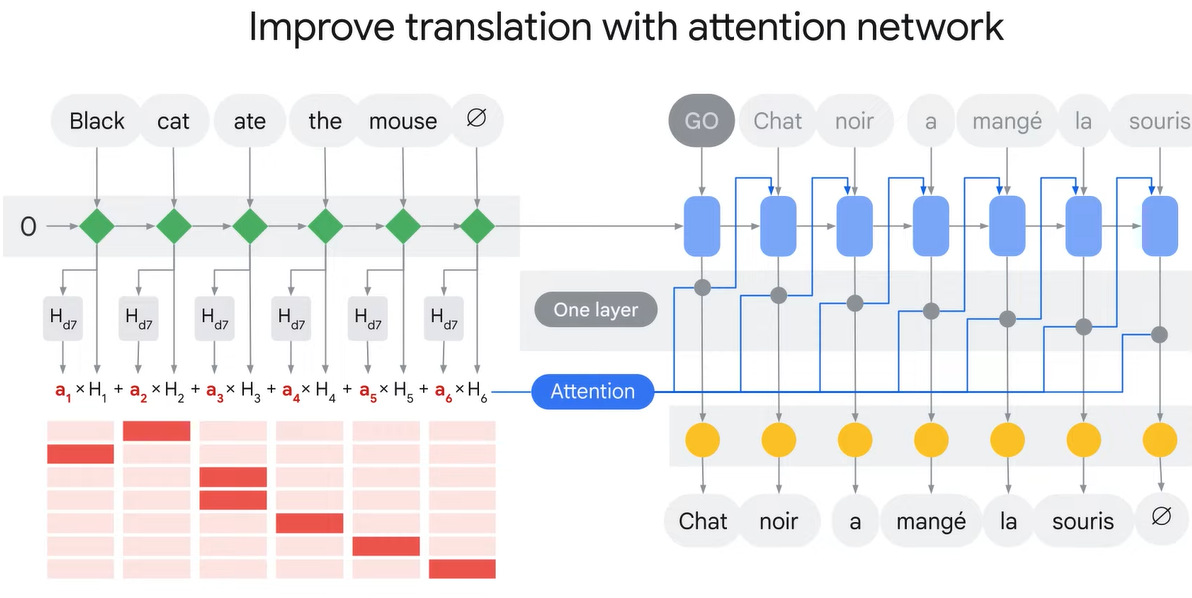
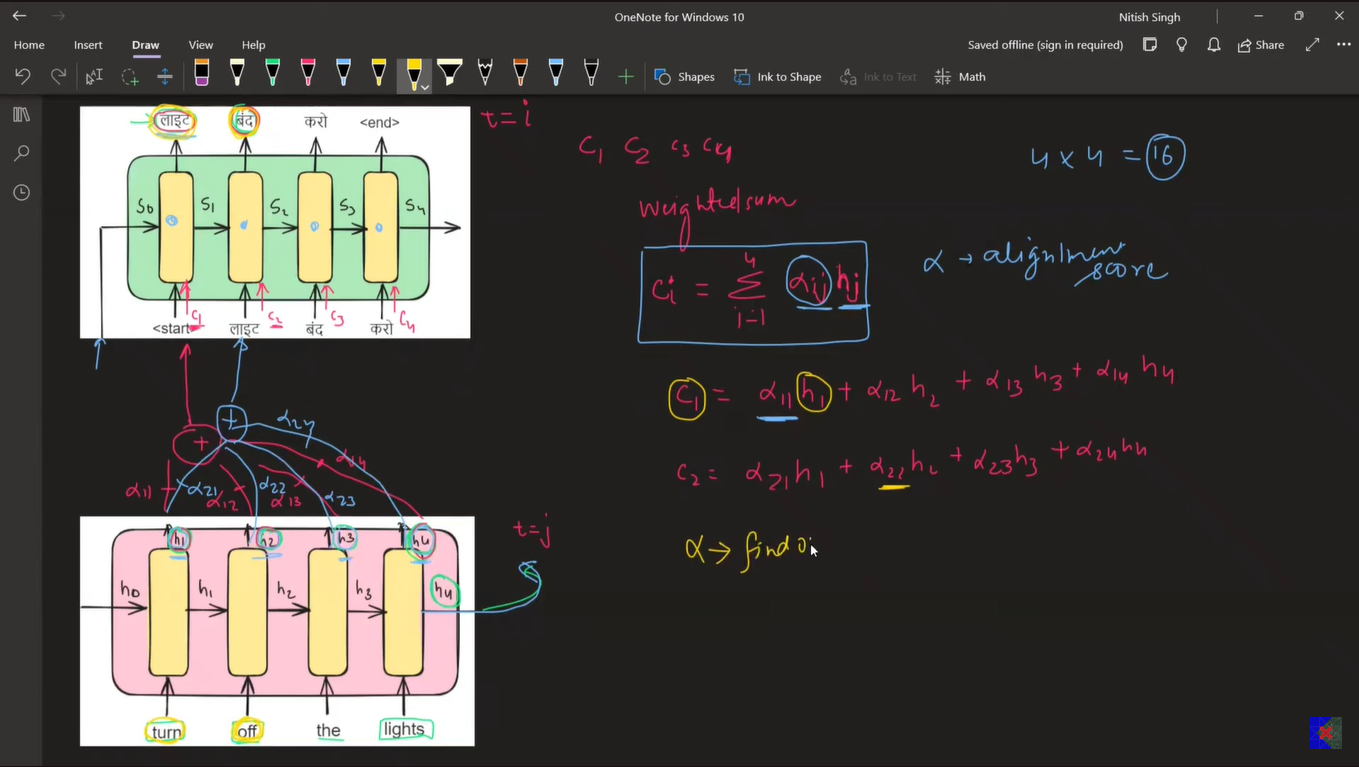
**Attention Mechanism**

======================================================================

* Traditional encoder-decoder architecture works for small sentences (often words < 30 in a sentence).
* Attention mechanism is one of the most powerful ideas in deep learning, especially in sequence modeling tasks like machine translation, text summarization, and image captioning.
* In sequence-to-sequence models, attention allows the decoder to “attend” to different parts of the input sequence, rather than relying on a single context vector (like in the basic Encoder-Decoder architecture).
* Attention mechanism allows models to selectively focus on relevant parts of the input data when making predictions, enhancing performance and efficiency, especially in tasks like natural language processing and computer vision.
* It dynamically weights different encoder outputs based on their relevance to the current decoding step.
* Attention model differs from the traditional model in the below ways:
  + Passing more data to the decoder (instead of passing the final vector, encoder passes all the hidden states to the decoder)
  + Have an extra step before producing an output.







------------------------------------------------------------------------------------------------------------

**Why Was It Introduced?**

In traditional encoder-decoder models:

* The encoder compresses the entire input sequence into one vector.
* That’s too limiting, especially for long sequences.

Attention solves this by allowing the decoder to look at all encoder outputs and decide which parts are more important for generating the next token.

--------------------------------------------------------------------------------------------------------------

**Architecture of Attention Mechanism**

**Step 1: Encoder Outputs**

* The encoder (usually an RNN/LSTM or Transformer encoder) processes the input and outputs a sequence of hidden states:



**Step 2: Decoder State**

* At each time step ‘t’, the decoder has its own hidden state st​ (from previous outputs).

**Step 3: Score Function**

* A **score** is calculated between st ​and each encoder hidden state **h**i​ to determine how relevant each input word is:



**Step 4: Softmax Over Scores**

* Convert scores to probabilities (called attention weights):



**Step 5: Context Vector**

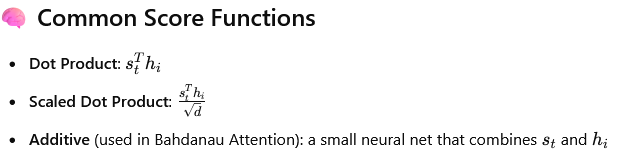
* 



**Step 6: Generate Output**

* The context vector ct​ is combined with the decoder state to generate the output token.

-------------------------------------------------------------------------------------------------------------



--------------------------------------------------------------------------------------------------------------

**Advantages of Attention**

* Handles long sequences better than plain encoder-decoder.
* Helps model focus on relevant parts of the input.
* Improves translation, summarization, captioning, etc.
* Is the core idea behind Transformers (which replaced RNNs in most applications).

**Drawbacks**

* Slightly more computational overhead due to scoring and weighting.
* Original attention models are sequential — less parallelizable than full Transformers.
* Still relies on sequence-based decoders, which can be slow (though Transformers fixed this).

--------------------------------------------------------------------------------------------------------------

**Real-Life Examples Using Attention**

* Google Translate
* Speech Recognition
* Chatbots (e.g., OpenAI models!)
* Image Captioning: Attend to parts of the image while generating a caption.