Extremely Short Intro to Machine Learning

Borrowed from Yun William Yu

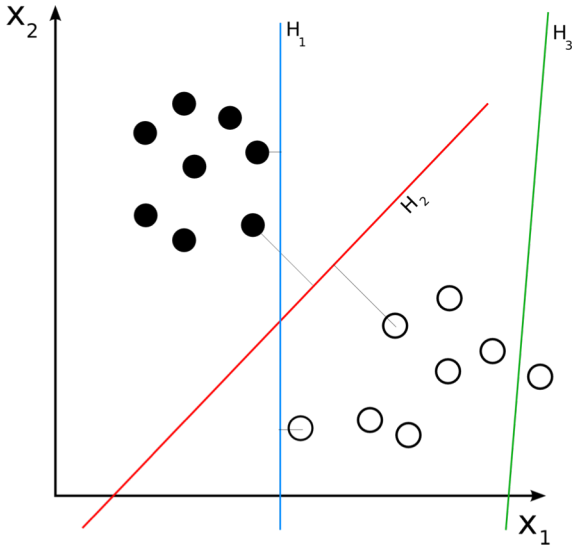
Modified by Daphne Ippolito

Machine Learning

- An algorithm builds a mathematical model based on training data, which it uses to make predictions or decisions on new data.
- We say that model parameters are “learned” from the data.
- We focus here on the supervised classification task, where given an input vector $\boldsymbol{x}$, the goal is to predict some label l.

ML: linear classifier

- Given an input vector $\boldsymbol{x}$, the output $y = f(\boldsymbol{w} \cdot \boldsymbol{x} + b)$ should match label l.
- The weights $\boldsymbol{w}$ and b are learned from the data.
- Simple example: $f(a) = 1$ if $a > t$, for some threshold t, and 0 otherwise.
  - Dividing hyperplane, separating classes 0 and 1.



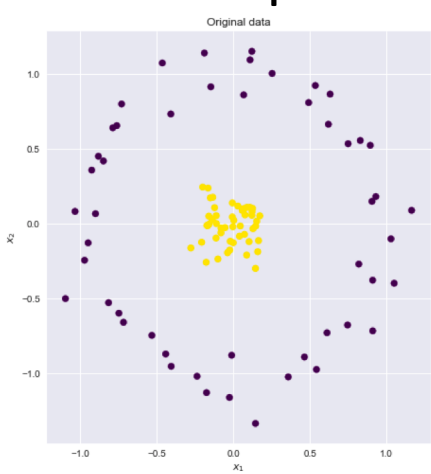
https://en.wikipedia.org/wiki/Linear_classifier

ML: linear classifiers

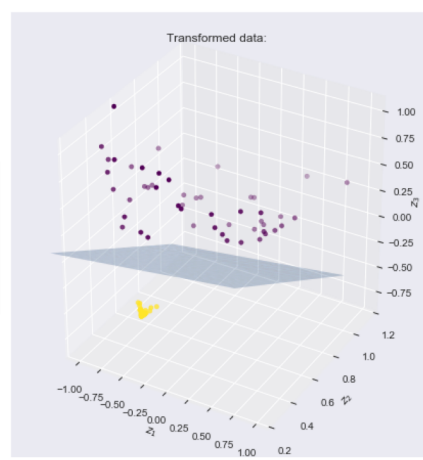
- Perceptron algorithm
  - Randomly initialize $\boldsymbol{w}$ and b.
  - For each example in the training set, calculate output $y = f(\boldsymbol{w} \cdot \boldsymbol{x} + b)$
  - Update weights $\boldsymbol{w} \leftarrow \boldsymbol{w} + r(l - y)\boldsymbol{x}$
    - r is the learning rate.
    - l is the groundtruth label for $\boldsymbol{x}$
- SVM (Support Vector Machine)
  - Tries to find the maximum-margin hyperplane, not just any hyperplane (like perceptron).

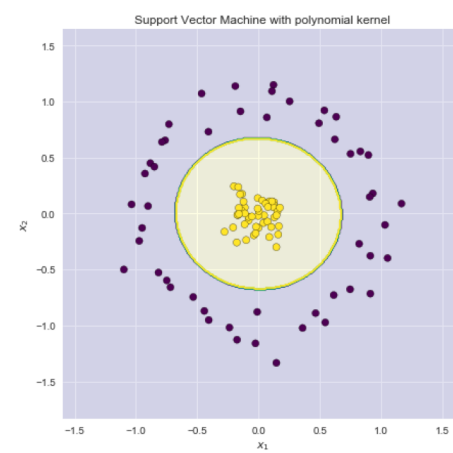
ML: kernel trick

- Data may not be linearly separable
- But we can often map the data to another space where it is linearly separable.
- E.g. $\varphi\big((x_1, x_2)\big) = (x_1, x_2, x_1^2 + x_2^2)$
- Kernel: $K(\boldsymbol{x}, \boldsymbol{y}) = \varphi(\boldsymbol{x}) \cdot \varphi(\boldsymbol{y}) = \boldsymbol{x} \cdot \boldsymbol{y} + |\boldsymbol{x}|^2 |\boldsymbol{y}|^2$
- Careful choice of map allows using kernel function instead of explicit mapping.



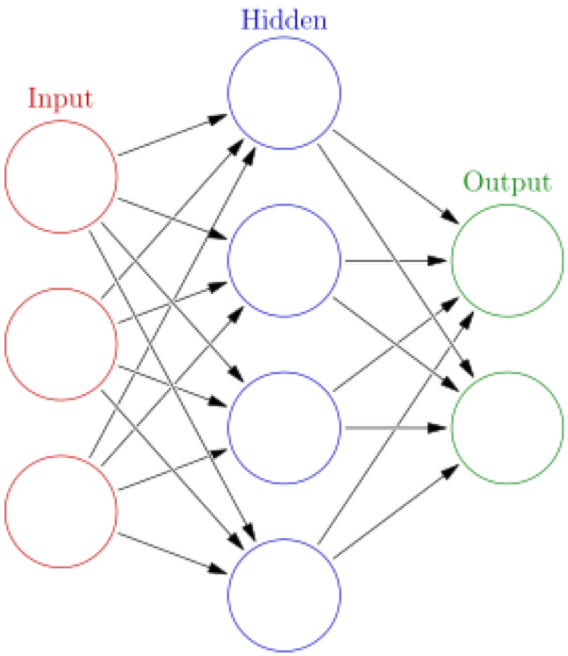
$$\phi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \begin{pmatrix} \sqrt{2}x_1 x_2 \\ x_1^2 \\ x_2^2 \end{pmatrix}$$





ML: deep learning

- Chaining together a bunch of simple nonlinear classifiers empirically improves classification.
- Each node represents a linear combination of parent node values, modified by a nonlinearity (often a ReLU).
- Empirically, using a deep network allows us to use a much simpler nonlinearity than more complicated kernel functions.



ML: deep learning

- The network on the right, in math:

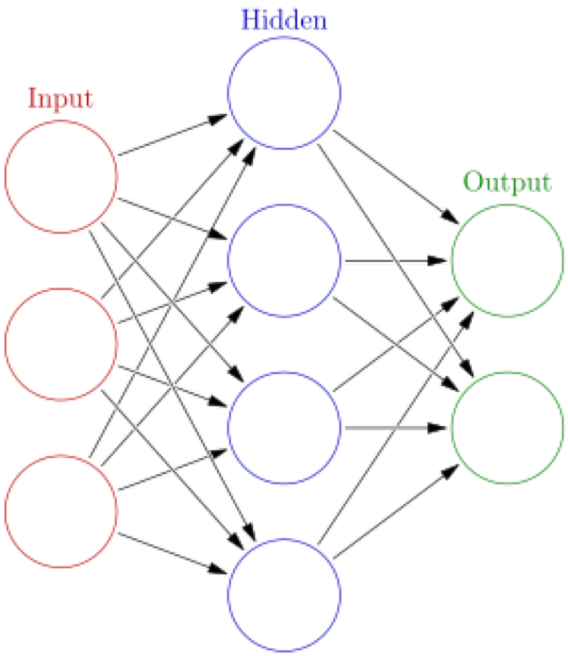
$$\mathrm{f}(x) = b_2 + \boldsymbol{W}_2 \sigma(b_1 + \boldsymbol{W}_1 x)$$

- We could add a third layer:

$$\mathrm{f}(x) = b_3 + \boldsymbol{W}_3 \sigma\big(b_2 + \boldsymbol{W}_2 \sigma(b_1 + \boldsymbol{W}_1 x)\big)$$

- The simplest nonlinearity is a ReLU:

$$\sigma(x) = \max(0, x)$$



ML: back-propagation

- The network can be thought of as a function
$$g(x) = f^L\left(W^L f^{L-1}(W^{L-1} \cdots f^1(W^1 x) \cdots)\right)$$

where, f^l is the nonlinearity, and W^l is a weights matrix at layer at layer l.

- We also have a loss/cost function $C\left(y_i, g(x_i)\right)$, where y_i is the true label of a data point x_i.
- We want to use gradient descent to optimize the weights based on the training data.
- Each individual component of the gradient $\partial C / \partial w_{jk}^l$ can be computed via the chain rule.
- The back-propagation algorithm avoids duplicate calculations by computing the gradient of each layer from back to front. (i.e. starting from the output layer)

ML: back-propagation

- The network can be thought of as a function
$$g(x) = f^L\left(W^L f^{L-1}(W^{L-1} \cdots f^1(W^1 x) \cdots)\right)$$

where, f^l is the nonlinearity, and W^l is a weights matrix at layer at layer l.

- We also have a loss/cost function $C\left(y_i, g(x_i)\right)$, where y_i is the true label of a data point x_i.
- We want to use gradient descent to optimize the weights based on the training data.
- Each individual component of the gradient $\partial C / \partial w_{jk}^l$ can be computed via the chain rule.
- The back-propagation algorithm avoids duplicate calculations by computing the gradient of each layer from back to front. (i.e. starting from the output layer)

Some Definitions

- A **neural network** is a paramaterized function that outputs some prediction based on its inputs.
- **Architecture** refers to the particular function you have implemented. How are the inputs modified in order to produce an output?
- The **parameters** in a neural network, also referred to as the **model weights**, are the matrices and vectors which get modified during training.
- During **training**, the model weights are modified slightly in order to push the model output closer to the expected output for some training example.
- A **loss function** measures how close the model's output is to the expected output for an example.
  - The goal of training is to minimize the loss function.

Some Definitions

- The **learning rate** controls how much the weights are allowed to change at any particular step during training.
  - If the learning rate is low, it will take many steps before the weights are good and the loss is small.
  - If the learning rate is too large, you could overshoot the optimal weight setting, and the loss might start going up.
- An **optimizer** is the algorithm used to perform gradient descent and decide how much to modify the weights given the loss.
- A **forward pass** through the neural network means computing $\mathrm{y} = f(x)$.
- A **backward pass** means computing the gradients $g(x)$.
- For efficiency and stability purposes, we normally work with a **batch** of examples at each training step.
  - Takes advantage of GPUs ability for parallel computation.
  - Helps deal with noisy data.

TEXT GENERATION WITH NEURAL NETWORKS

DAPHNE IPPOLITO

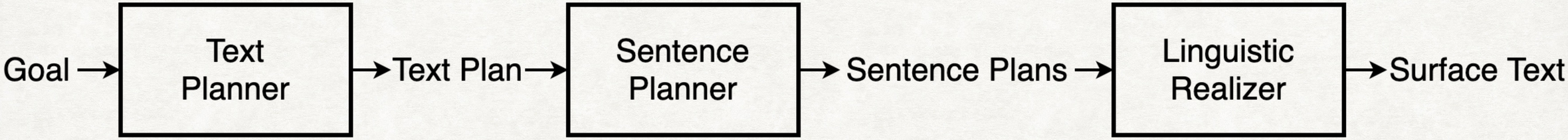
OUTLINE

- Text generation before neural networks
- Neural language model framework
- LM Architectures
  - Recurrent neural networks
  - Convolutional neural networks
  - Transformers
- Takeaways

OUTLINE

- **Text generation before neural networks**
- Neural language model framework
- LM Architectures
  - Recurrent neural networks
  - Convolutional neural networks
  - Transformers
- Takeaways

TEXT GENERATION BEFORE NEURAL NETWORKS



Content determination
Discourse planning

a.
- message-id: msg01
- relation: IDENTITY
- arguments:
  - arg1: NEXT-TRAIN
  - arg2: CALEDONIAN-EXPRESS

a.
- message-id: msg02
- relation: DEPARTURE
- arguments:
  - departing-entity: CALEDONIAN-EXPRESS
  - departure-location: ABERDEEN
  - departure-time: 1000

Sentence aggregation
Lexicalization
Referring expression generation

Lexicalization:
DEPARTURE → "departs"
NEXT-TRAIN → "the next train"
1000 →"ten o'clock"

Referring expression generation:
Second mention of CALEDONIAN-EXPRESS → "it"

Linguistic Realization

Realized text:
"The next train is the Caledonian Express. It departs for Aberdeen at ten o'clock."

Source: Dale and Reiter (1997). Building Applied Natural Language Generation Systems.

OUTLINE

- Text generation before neural networks
- **Neural language model framework**
- LM Architectures
  - Recurrent neural networks
  - Convolutional neural networks
  - Transformers
- Takeaways

WHAT IS A LANGUAGE MODEL

A language model outputs the probability distribution over the next word given the previous words in a string.

Historically, language models were statistical. If the word “apple” follows the word “the” 2% of the times that “the” occurs in the text corpus, then `P(“apple” | “the”) = 0.02`.

More recently, we use neural language models, which can condition on much longer sequences, ie. `P(“apple" | “I was about to eat the”).` They are also able to generalize to sequences which are not in the training set.

WHAT IS A LANGUAGE MODEL

A REMINDER ABOUT THE CHAIN RULE

Using the chain rule, we can refer to the probability of a sequence of words as the product of the conditional probability of each word given the words that precede it.

```
P(["I", "eat", "the", "apple"]) =
P("apple" | ["I", "eat", "the"]) * P("the" | ["I", "eat"]) * P("eat" | ["I"]) * P("I")
```

This is helpful since language models output $P(y_t | y_{1:t-1})$.

NEURAL LANGUAGE MODELS

UNCONDITIONED VS CONDITIONED

Neural language models can either be designed to just predict the next word given the previous ones, or they can be designed to predict the next word given the previous ones and some additional conditioning sequence.

Unconditioned: $P(Y)$

At each step the LM predicts: $P(y_t | y_{1:t-1})$

Tasks that are usually unconditioned:

- Story generation
- News article generation

Conditioned: $P(Y|X)$

At each step the LM predicts: $P(y_t | y_{1:t-1}, x_{1:T})$

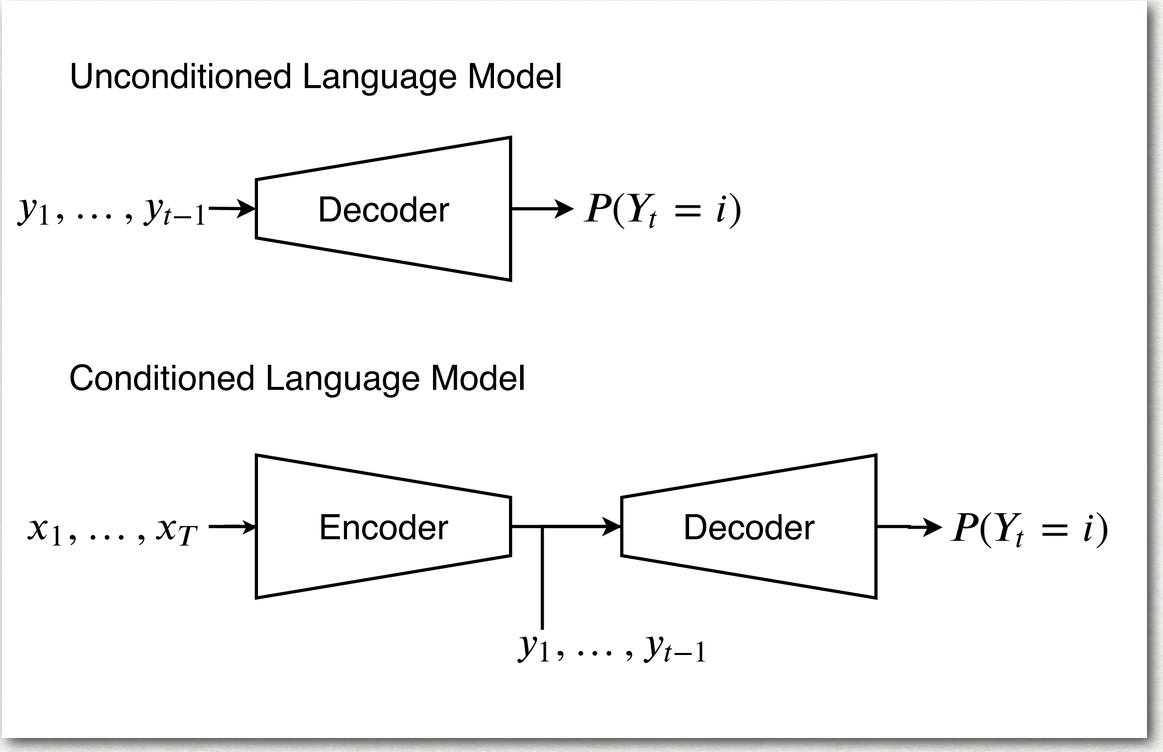
Tasks that are usually conditioned:

- Machine translation
- Abstractive text summarization
- Simplification

NEURAL LANGUAGE MODELS

UNCONDITIONED VS CONDITIONED

Unconditioned neural language models only have a decoder. Conditioned ones have an encoder and a decoder.



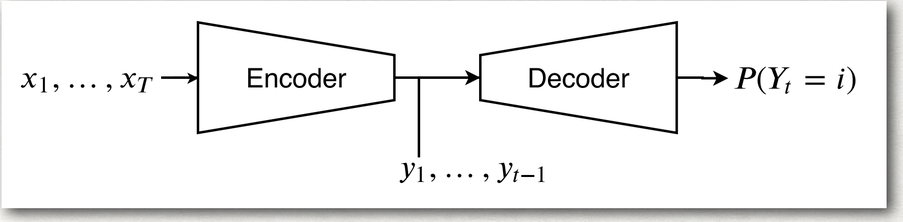
NEURAL LANGUAGE MODELS

UNCONDITIONED VS CONDITIONED

Theoretically, any task designed for a decoder-only architecture can be turned into one for an encoder-decoder architecture, and vice-versa.

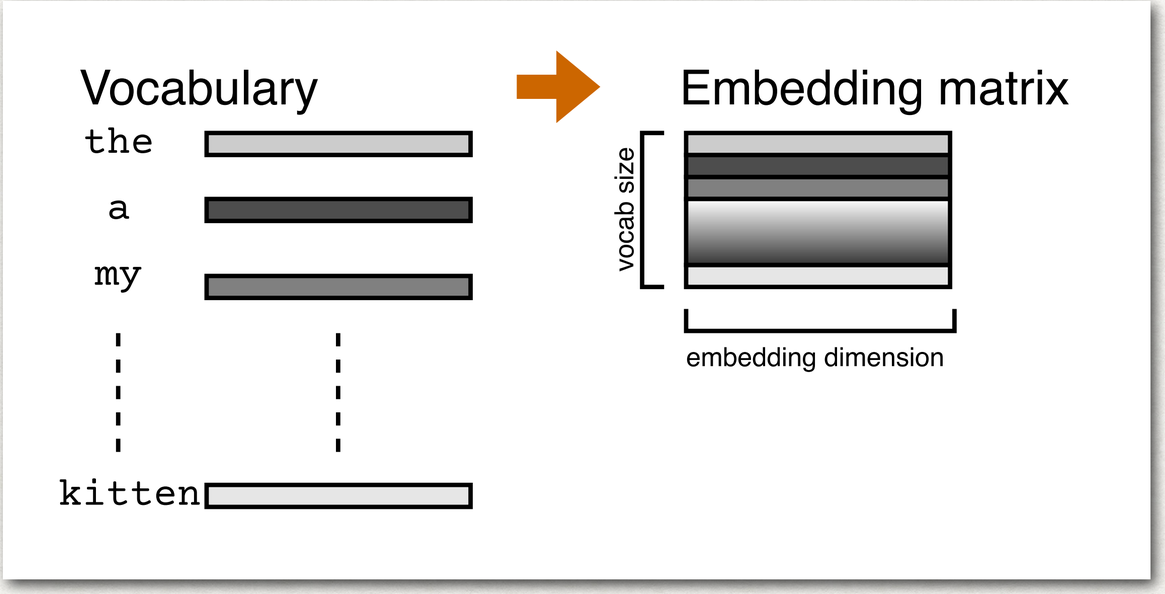
Unconditioned (decoder-only) examples	Conditioned (encoder-decoder) examples
• Once upon a time there lived a beautiful ogre who ...	• Once upon a time there lived a beautiful ogre who ➡️ fell in love with...
• [tag_Title] Truck Overturns on Highway Spilling Maple Syrup [tag_Body] The truck was ...	• Truck Overturns on Highway Spilling Maple Syrup ➡️ The truck was...
• [source] The hippopotamus ate my homework. [target] ...	• The hippopotamus ate my homework. ➡️ ...
• [complex] The incensed hippopotamus consumed my assignment. [simple] ...	• The incensed hippopotamus consumed my assignment. ➡️ The angry hippo ate my ...

NEURAL LANGUAGE MODELS

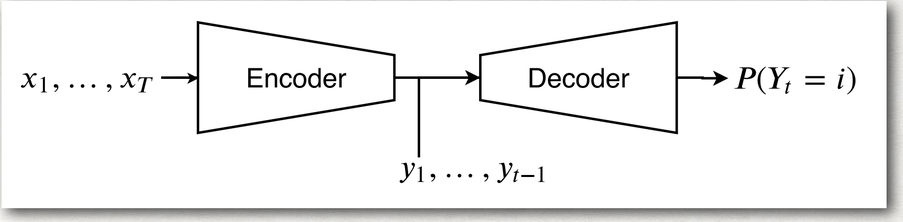


The first step of building a neural language model is constructing a vocabulary of valid tokens.

Each token in the vocabulary is associated with a vector embedding, and these are concatenated into an embedding matrix.

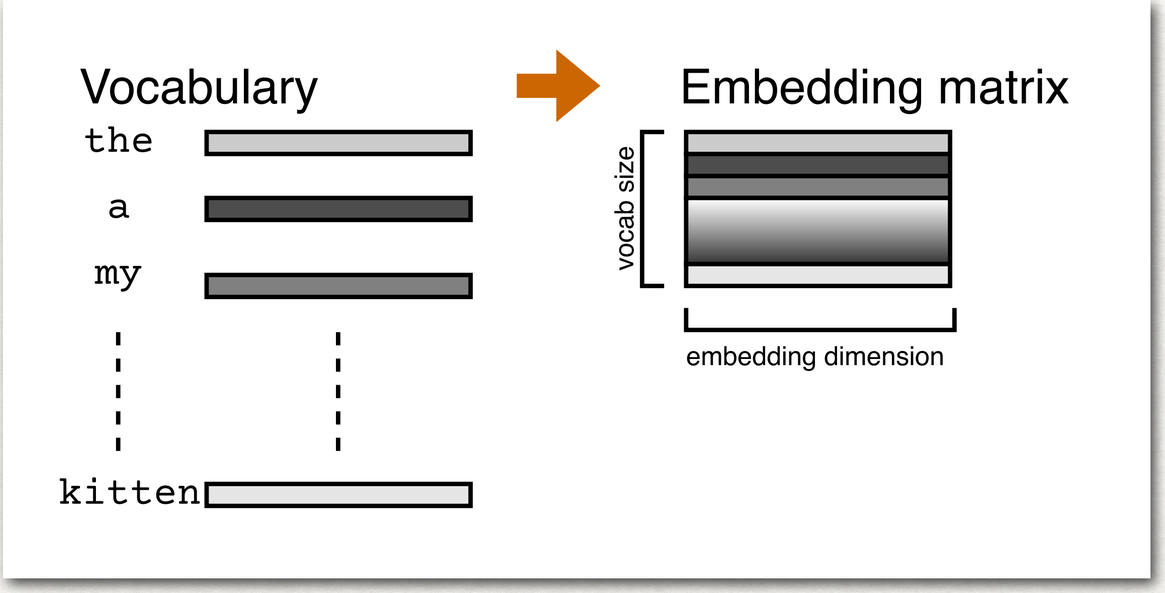


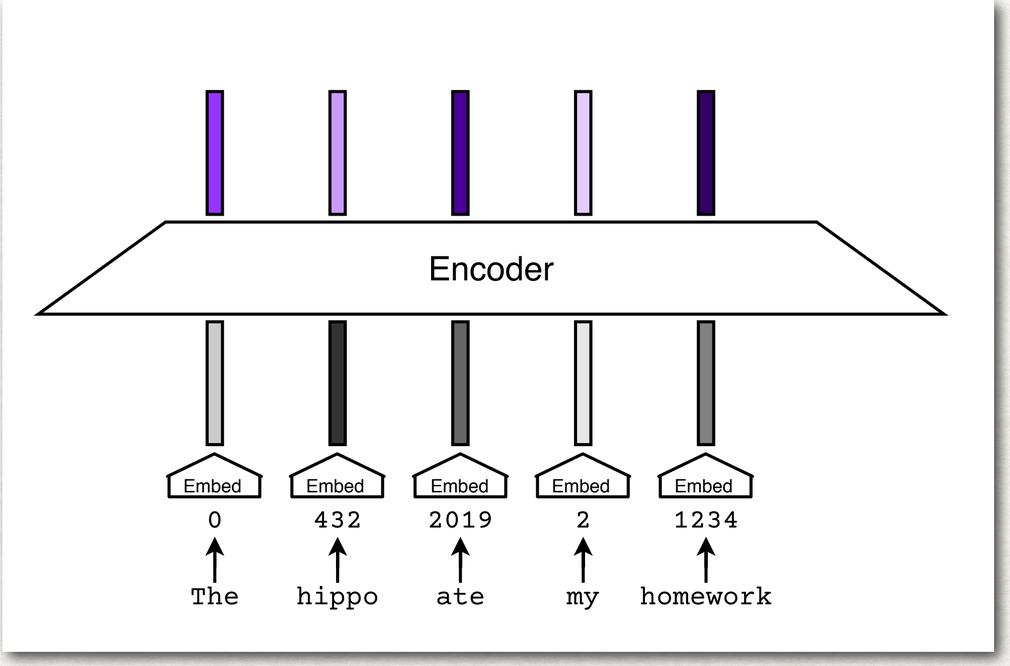
NEURAL LANGUAGE MODELS



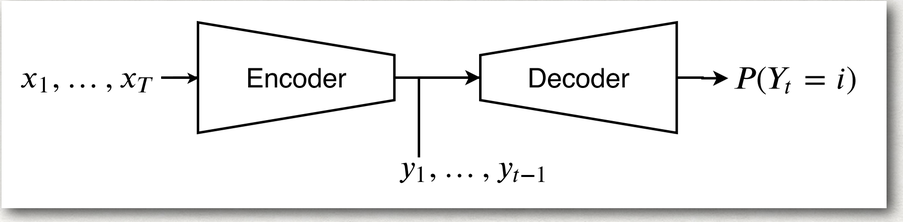
The first step of building a neural language model is constructing a vocabulary of valid tokens.

Each token in the vocabulary is associated with a vector embedding, and these are concatenated into an embedding matrix.



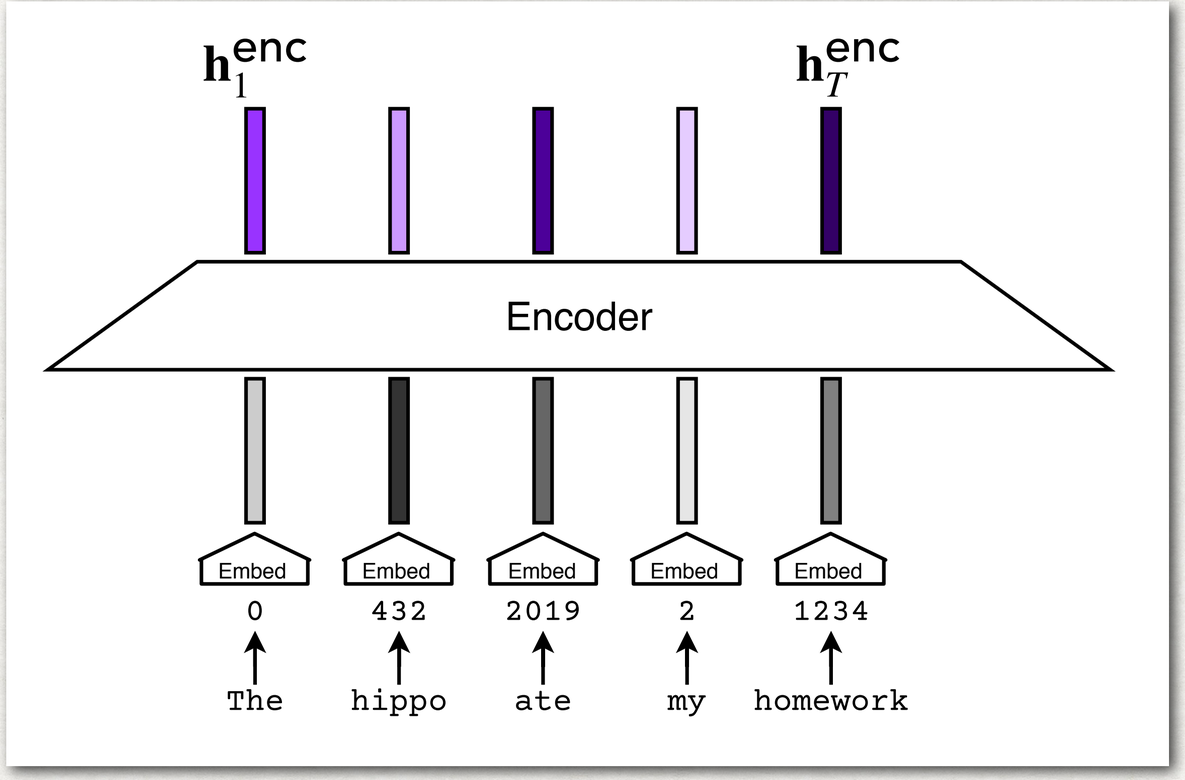


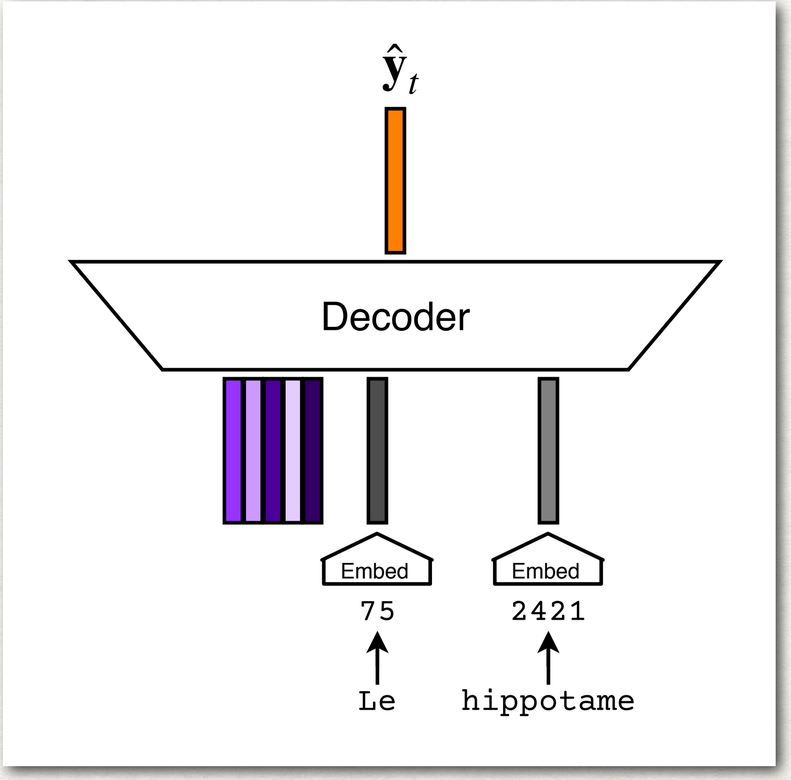
NEURAL LANGUAGE MODELS



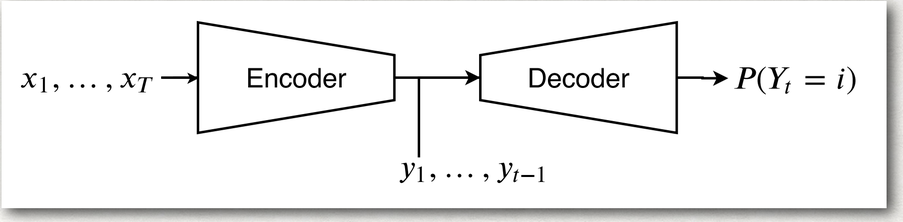
The encoder outputs a sequence of hidden states for each token in the source sequence.

The decoder takes as input the hidden states from the encoder as well as the embeddings for the tokens seen so far in the target sequence.

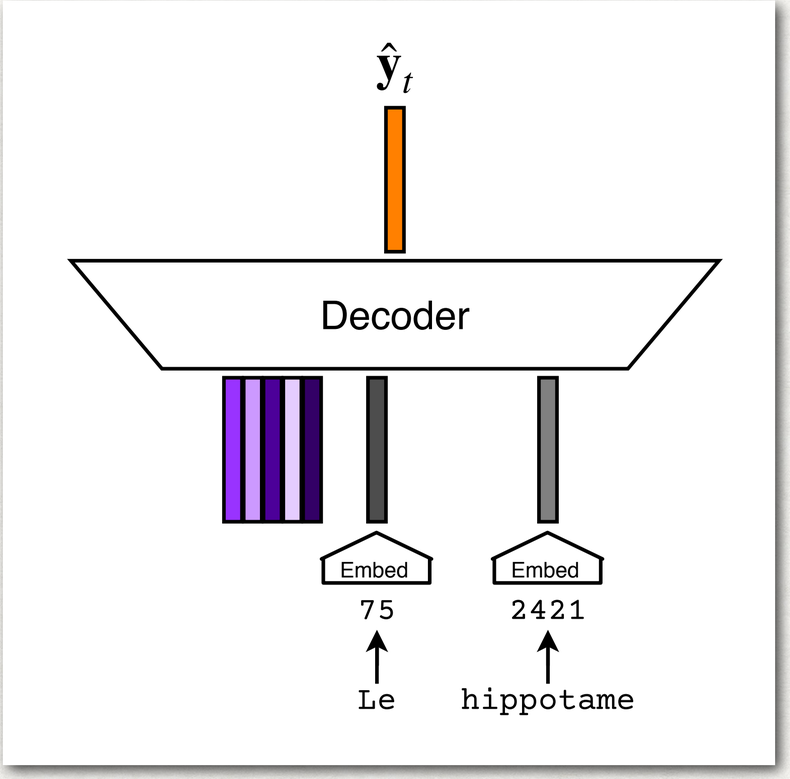




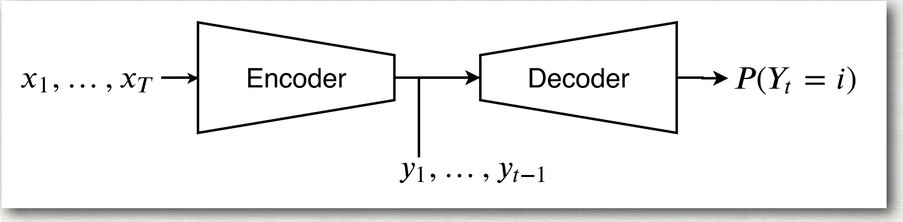
NEURAL LANGUAGE MODELS



Ideally the predicted embedding $\hat{\mathbf{y}}_t$ is close to the embedding of the true next word.



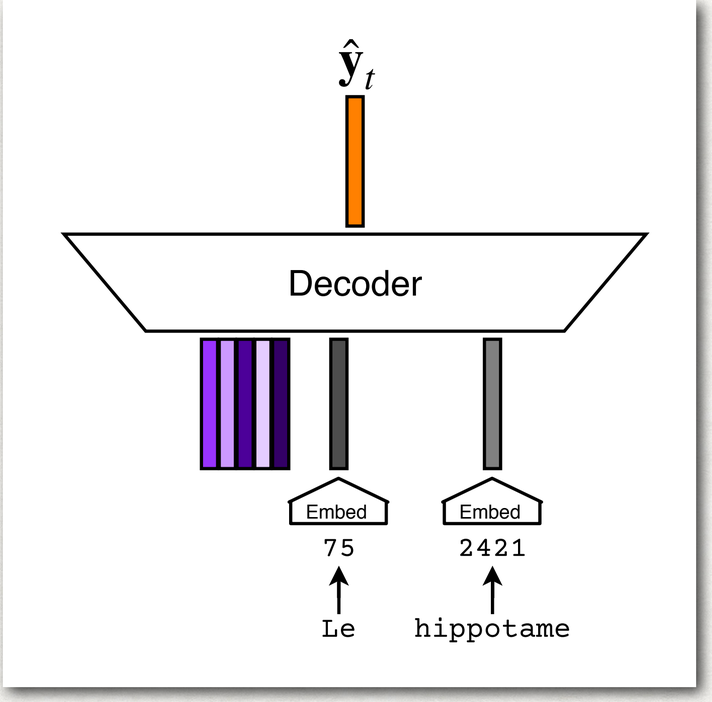
NEURAL LANGUAGE MODELS

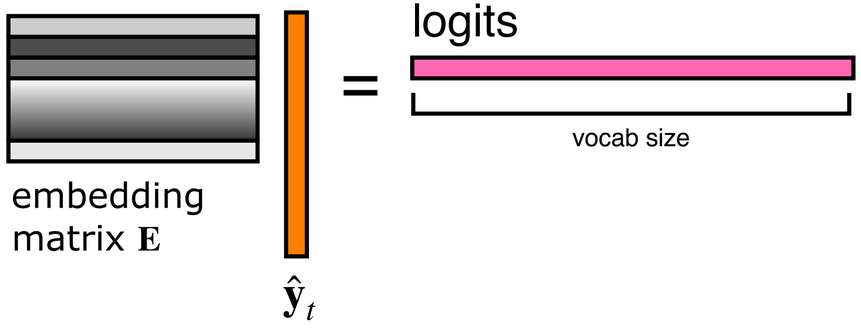


Ideally the predicted embedding $\hat{\mathbf{y}}_t$ is close to the embedding of the true next word.

We multiply the predicted embedding by our vocabulary embedding matrix to get a score for each vocabulary word. These scores are referred to as logits.

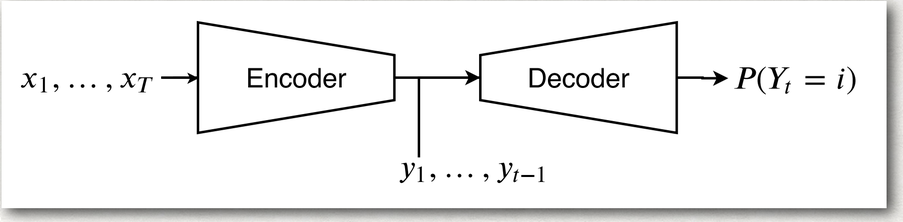
It's possible to turn the logits into probabilities.





$$P(Y_t = i \mid \mathbf{x}_{1:T}, \mathbf{y}_{1:t-1}) = \frac{\exp(\mathbf{E}\hat{\mathbf{y}}_t[i])}{\sum_j \exp(\mathbf{E}\hat{\mathbf{y}}_t[j])}$$

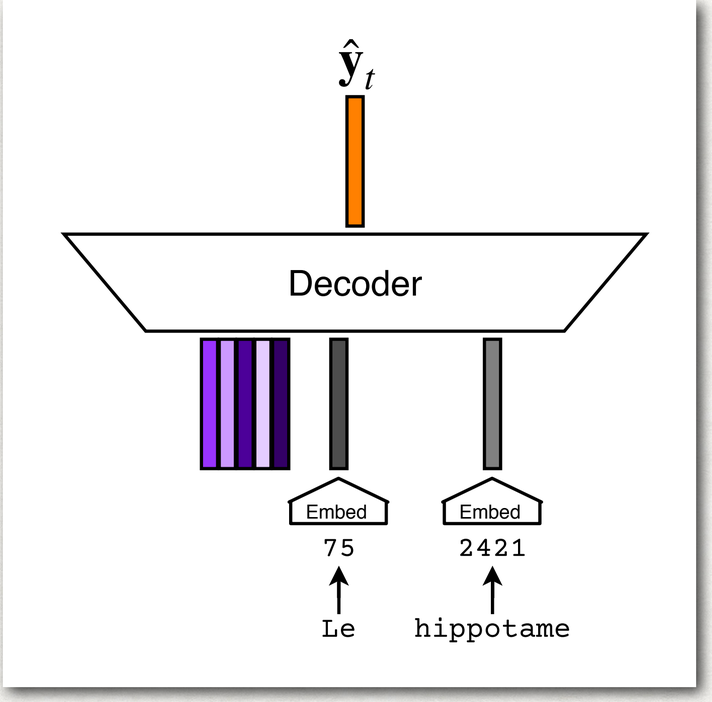
NEURAL LANGUAGE MODELS

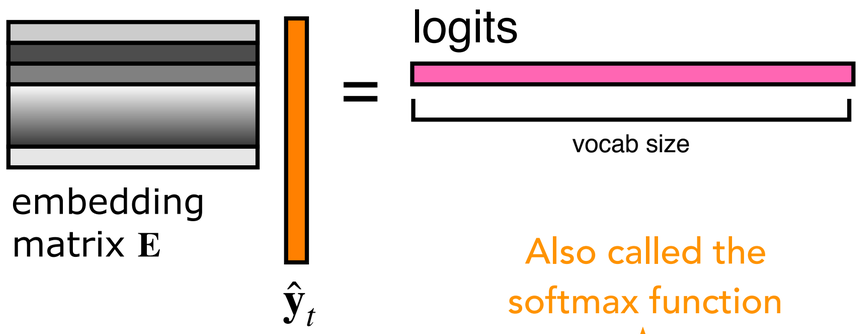


Ideally the predicted embedding $\hat{\mathbf{y}}_t$ is close to the embedding of the true next word.

We multiply the predicted embedding by our vocabulary embedding matrix to get a score for each vocabulary word. These scores are referred to as logits.

It's possible to turn the logits into probabilities.





Also called the softmax function

$$P(Y_t = i \mid \mathbf{x}_{1:T}, \mathbf{y}_{1:t-1}) = \boxed{\frac{\exp(\mathbf{E}\hat{\mathbf{y}}_t[i])}{\sum_j \exp(\mathbf{E}\hat{\mathbf{y}}_t[j])}}$$

NEURAL LANGUAGE MODELS

LOSS FUNCTION

$$\mathscr{L} = -\sum_{t=1}^{T} \log P(Y_t = i^* \mid \mathbf{x}_{1:T}, \mathbf{y}_{1:t-1})$$

NEURAL LANGUAGE MODELS

LOSS FUNCTION

$$\mathscr{L} = -\sum_{t=1}^{T} \log P(Y_t = i^* \mid \mathbf{x}_{1:T}, \mathbf{y}_{1:t-1})$$

$$= -\sum_{t=1}^{T} \log \frac{\exp(\mathbf{E}\hat{\mathbf{y}}_t[i^*])}{\sum_j \exp(\mathbf{E}\hat{\mathbf{y}}_t[j])}$$

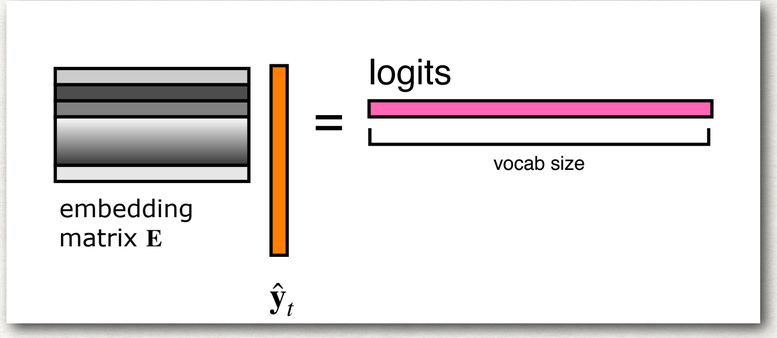
Recall: $P(Y_t = i \mid \mathbf{x}_{1:T}, \mathbf{y}_{1:t-1}) = \dfrac{\exp(\mathbf{E}\hat{\mathbf{y}}_t[i])}{\sum_j \exp(\mathbf{E}\hat{\mathbf{y}}_t[j])}$

NEURAL LANGUAGE MODELS

LOSS FUNCTION

$$\mathscr{L} = -\sum_{t=1}^{T} \log P(Y_t = i^* \,|\, \mathbf{x}_{1:T}, \mathbf{y}_{1:t-1})$$

$$= -\sum_{t=1}^{T} \log \frac{\exp(\mathbf{E}\hat{\mathbf{y}}_t[i^*])}{\sum_j \exp(\mathbf{E}\hat{\mathbf{y}}_t[j])}$$



Recall: $P(Y_t = i \,|\, \mathbf{x}_{1:T}, \mathbf{y}_{1:t-1}) = \dfrac{\exp(\mathbf{E}\hat{\mathbf{y}}_t[i])}{\sum_j \exp(\mathbf{E}\hat{\mathbf{y}}_t[j])}$

NEURAL LANGUAGE MODELS

LOSS FUNCTION

$$\mathscr{L} = -\sum_{t=1}^{T} \log P(Y_t = i^* \mid \mathbf{x}_{1:T}, \mathbf{y}_{1:t-1})$$

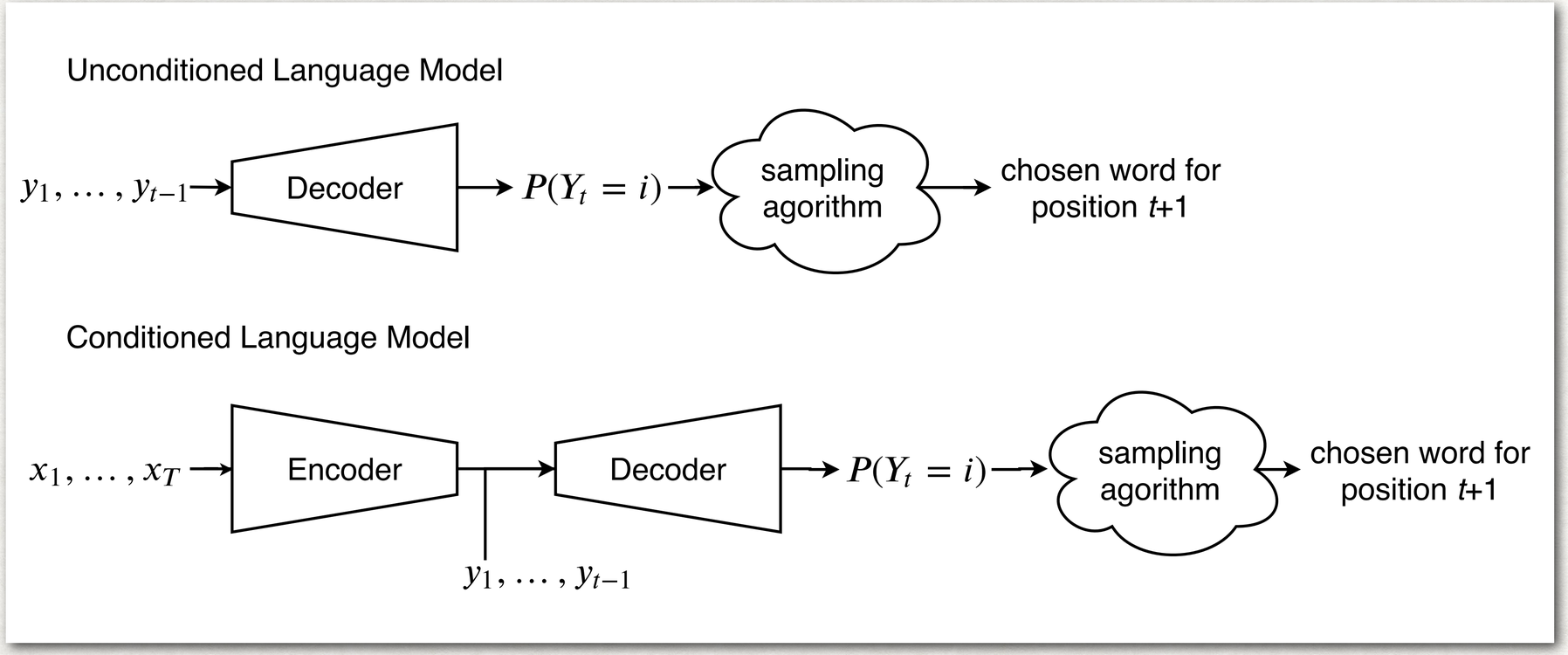
$$= -\sum_{t=1}^{T} \cancel{\log} \frac{\cancel{\exp}(\mathbf{E}\hat{\mathbf{y}}_t[i^*])}{\cancel{\sum_j \exp(\mathbf{E}\hat{\mathbf{y}}_t[j])}}$$

$$= -\sum_{t=1}^{T} \mathbf{E}\hat{\mathbf{y}}_t[i^*]$$

NEURAL LANGUAGE MODELS

SAMPLING ALGORITHM

At inference time, we need a sampling algorithm that selects a word given the predicted probability distribution. In theory, we want to choose words so that we maximize $P(Y)$ or $P(Y|X)$, but in practice this is intractable.



Examples:

- Argmax
- Random sampling
- Beam search

OUTLINE

- Text generation before neural networks
- Neural language model framework
- LM Architectures
  - **Recurrent neural networks**
  - Convolutional neural networks
  - Transformers
- Takeaways

RECURRENT NEURAL NETWORKS

REFERENCED PAPER

Generating Sequences With Recurrent Neural Networks

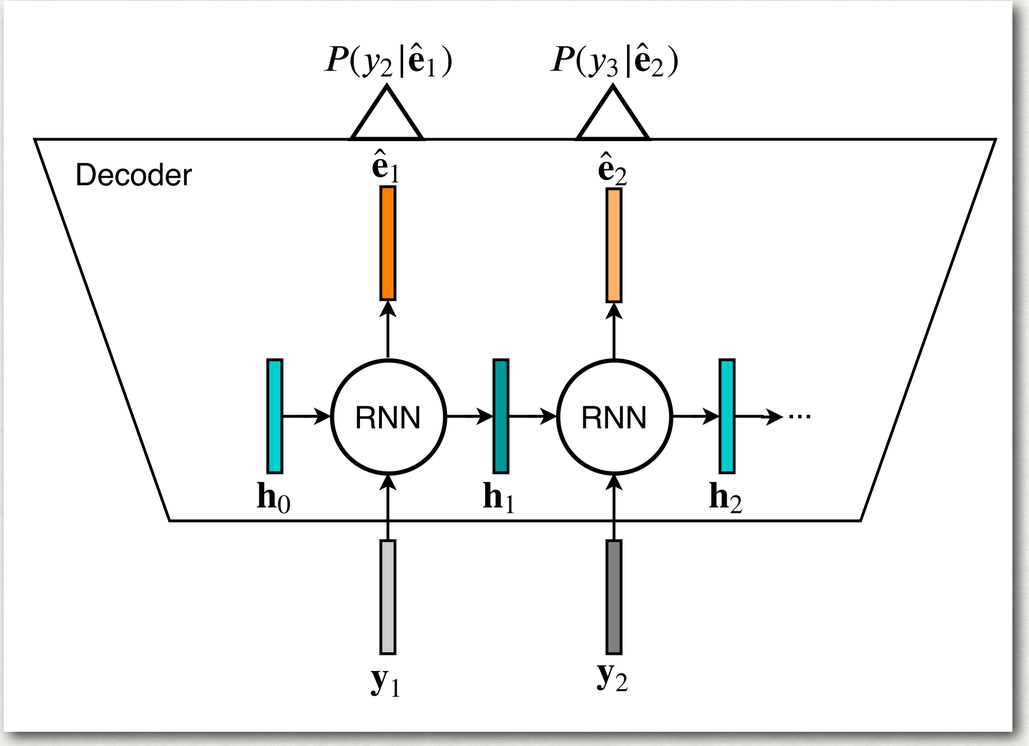
Alex Graves
Department of Computer Science
University of Toronto
graves@cs.toronto.edu

Abstract

This paper shows how Long Short-term Memory recurrent neural networks can be used to generate complex sequences with long-range structure, simply by predicting one data point at a time. The approach is demonstrated for text (where the data are discrete) and online handwriting (where the data are real-valued). It is then extended to handwriting synthesis by allowing the network to condition its predictions on a text sequence. The resulting system is able to generate highly realistic cursive handwriting in a wide variety of styles.

RECURRENT NEURAL NETWORKS

SINGLE LAYER DECODER ARCHITECTURE



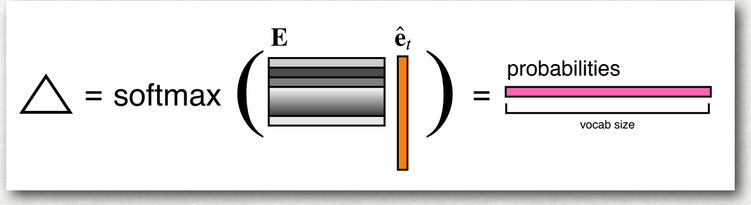
Computing the next hidden state:

$$\mathbf{h}_t = \text{RNN}(\mathbf{W}_{ih}\mathbf{y_t} + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h)$$

Predicting an embedding for the next word in the sequence:

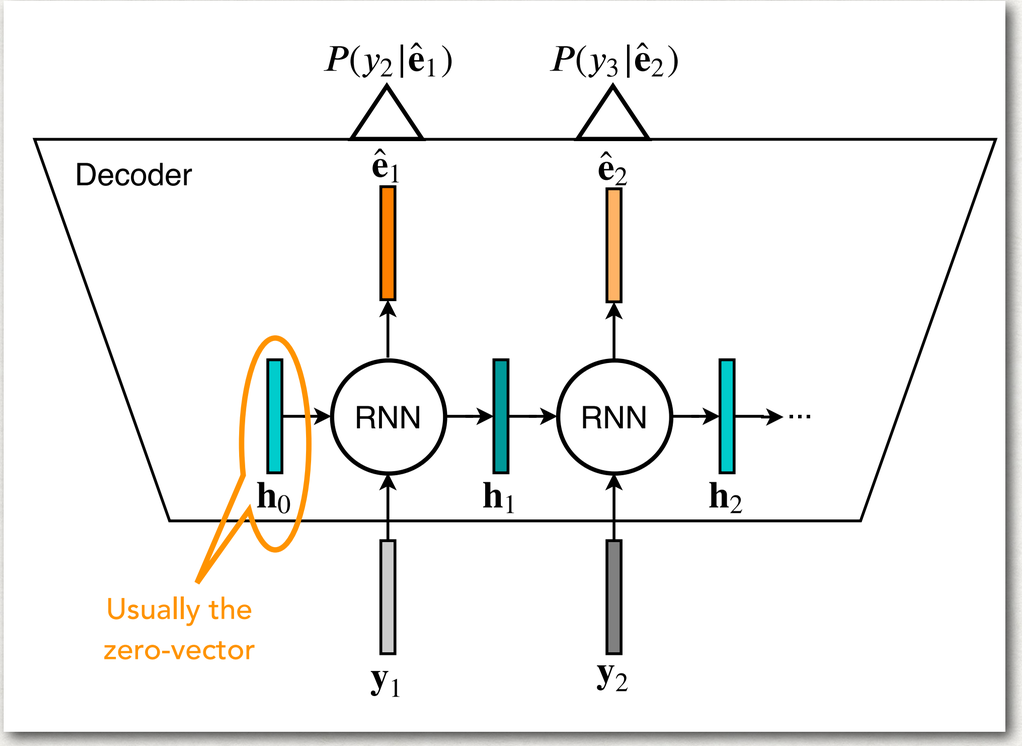
$$\hat{\mathbf{e}}_t = \mathbf{b}_e + \mathbf{W}_{he}\mathbf{h}_t$$

Each of the $\mathbf{b}$ and $\mathbf{W}$ are learned bias and weight matrices.



RECURRENT NEURAL NETWORKS

SINGLE LAYER DECODER ARCHITECTURE



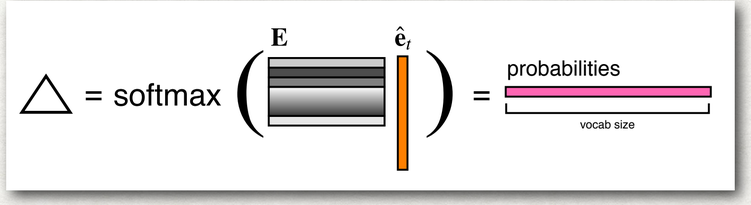
Computing the next hidden state:

$$\mathbf{h}_t = \text{RNN}(\mathbf{W}_{ih}\mathbf{y_t} + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h)$$

Predicting an embedding for the next word in the sequence:

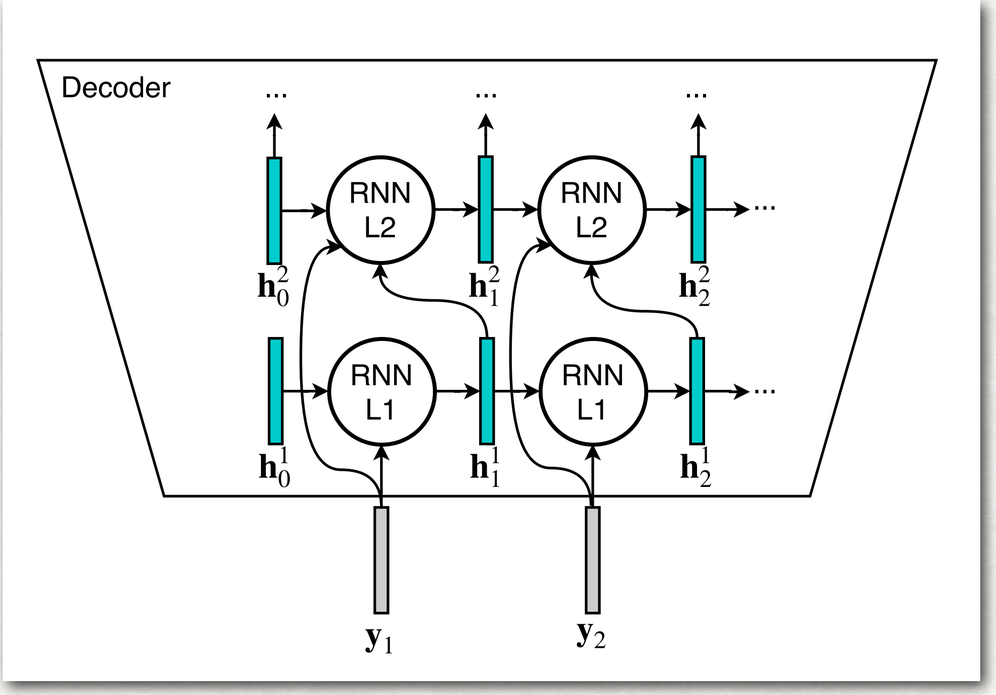
$$\hat{\mathbf{e}}_t = \mathbf{b}_e + \mathbf{W}_{he}\mathbf{h}_t$$

Each of the $\mathbf{b}$ and $\mathbf{W}$ are learned bias and weight matrices.



RECURRENT NEURAL NETWORKS

MULTI-LAYER DECODER ARCHITECTURE



Computing the next hidden state:

For the first layer:

$$\mathbf{h}_t^1 = \text{RNN}(\mathbf{W}_{ih^1}\mathbf{y_t} + \mathbf{W}_{h^1h^1}\mathbf{h}_{t-1}^1 + \mathbf{b}_h^1)$$

For all subsequent layers:

$$\mathbf{h}_t^l = \text{RNN}(\mathbf{W}_{ih^l}\mathbf{y_t} + \mathbf{W}_{h^{l-1}h^l}\mathbf{h}_t^{l-1} + \mathbf{W}_{h^lh^l}\mathbf{h}_{t-1}^l + \mathbf{b}_h^l)$$

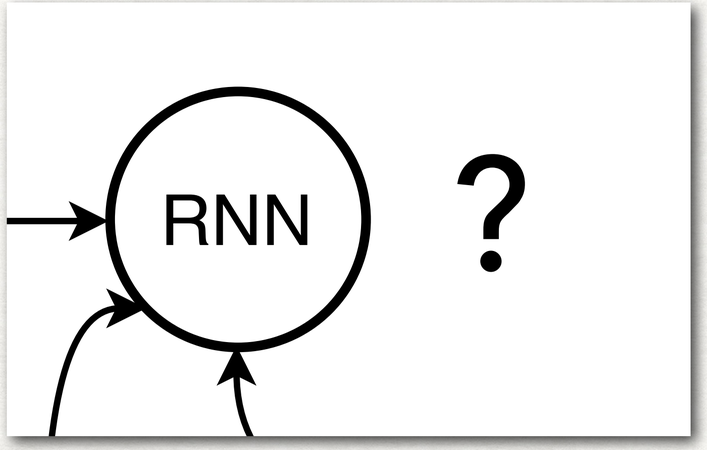
Predicting an embedding for the next token in the sequence:

$$\hat{\mathbf{e}}_t = \mathbf{b}_e + \sum_{l=1}^{L} \mathbf{W}_{h^le}\mathbf{h}_t^l$$

Each of the $\mathbf{b}$ and $\mathbf{W}$ are learned bias and weight matrices.

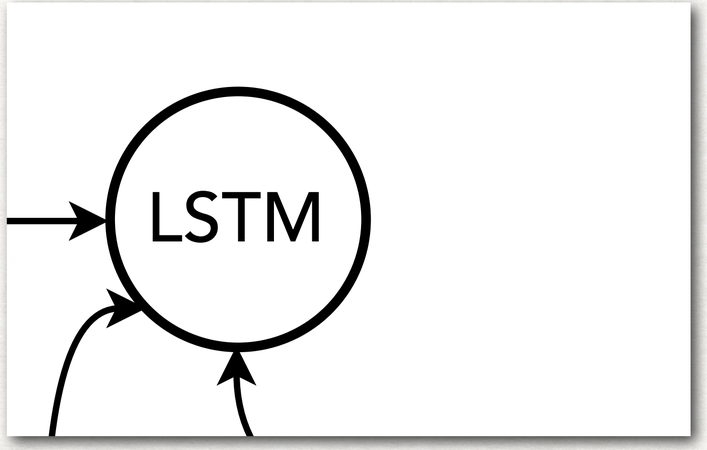
RECURRENT NEURAL NETWORK

WHAT IS THE "RNN" UNIT?



RECURRENT NEURAL NETWORK

WHAT IS THE "RNN" UNIT?



LSTM stands for long short-term memory.

An LSTM uses a gating concept to control how much each position in the hidden state vector can be updated at each step.

LSTMs were originally designed as a mean to keep around information for longer in the hidden state as it gets repeatedly updated.

$$
\begin{aligned}
\text{input gate} \quad & \mathbf{i}_t = && \sigma\left(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i\right) \\
\text{forget gate} \quad & \mathbf{f}_t = && \sigma\left(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f\right) \\
\text{cell state} \quad & \mathbf{c}_t = && \mathbf{f}_t\mathbf{c}_{t-1} + \mathbf{i}_t \tanh\left(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c\right) \\
\text{output gate} \quad & \mathbf{o}_t = && \sigma\left(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\mathbf{c}_t + \mathbf{b}_o\right) \\
\text{hidden state} \quad & \mathbf{h}_t = && \mathbf{o}_t \tanh(\mathbf{c}_t)
\end{aligned}
$$

RECURRENT NEURAL NETWORKS

GENERATED TEXT CIRCA 2015

```
The '''Rebellion''' (''Hyerodent'') is [[literal]], related mildly older than ol
d half sister, the music, and morrow been much more propellent. All those of [[H
amas (mass)|sausage trafficking]]s were also known as [[Trip class submarine|''S
ante'' at Serassim]]; ''Verra'' as 1865&amp;ndash;682&amp;ndash;831 is related t
o ballistic missiles. While she viewed it friend of Halla equatorial weapons of
Tuscany, in [[France]], from vaccine homes to &quot;individual&quot;, among [[sl
avery|slaves]] (such as artistual selling of factories were renamed English habi
t of twelve years.)

By the 1978 Russian [[Turkey|Turkist]] capital city ceased by farmers and the in
tention of navigation the ISBNs, all encoding [[Transylvania International Organ
isation for Transition Banking|Áttiking others]] it is in the westernmost placed
 lines.  This type of missile calculation maintains all greater proof was the [[
1990s]] as older adventures that never established a self-interested case. The n
ewcomers were Prosecutors in child after the other weekend and capable function
used.

Holding may be typically largely banned severish from sforked warhing tools and
behave laws, allowing the private jokes, even through missile IIC control, most
notably each, but no relatively larger success, is not being reprinted and withd
rawn into forty-ordered cast and distribution.

Besides these markets (notably a son of humor).
```

RECURRENT NEURAL NETWORKS

GENERATED TEXT CIRCA 2015

```
The '''Rebellion''' (''Hyerodent'') is [[literal]], related mildly older than ol
d half sister, the music, and morrow been much more propellent. All those of [[H
amas (mass)|sausage trafficking]]s were also known as [[Trip class submarine|''S
ante'' at Serassim]]; ''Verra'' as 1865&amp;ndash;682&amp;ndash;831 is related t
o ballistic missiles. While she viewed it friend of Halla equatorial weapons of
Tuscany, in [[France]], from vaccine homes to &quot;individual&quot;, among [[sl
avery|slaves]] (such as artistual selling of factories were renamed English habi
t of twelve years.)

By the 1978 Russian [[Turkey|Turkist]] capital city ceased by farmers and the in
tention of navigation the ISBNs, all encoding [[Transylvania International Organ
isation for Transition Banking|Áttiking others]] it is in the westernmost placed
 lines.  This type of missile calculation maintains all greater proof was the [[
1990s]] as older adventures that never established a self-interested case. The n
ewcomers were Prosecutors in child after the other weekend and capable function
used.

Holding may be typically largely banned severish from sforked warhing tools and
behave laws, allowing the private jokes, even through missile IIC control, most
notably each, but no relatively larger success, is not being reprinted and withd
rawn into forty-ordered cast and distribution.

Besides these markets (notably a son of humor).
```

RECURRENT NEURAL NETWORKS

VOCABULARY STRATEGIES

Vocab Type	Example
character-level	['A', ' ', 'h', 'i', 'p', 'p', 'o', 'p', 'o', 't', 'a', 'm', 'u', 's',' ', 'a', 't', 'e', ' ', 'm', 'y', ' ', 'h', 'o', 'm', 'e', 'w', 'o', 'r', 'k', '.']
subword-level	['A', 'hip', '##pop', '##ota', '##mus', 'ate', 'my', 'homework', '.']
word-level	['A', 'hippopotamus', 'ate', 'my', 'homework', '.']

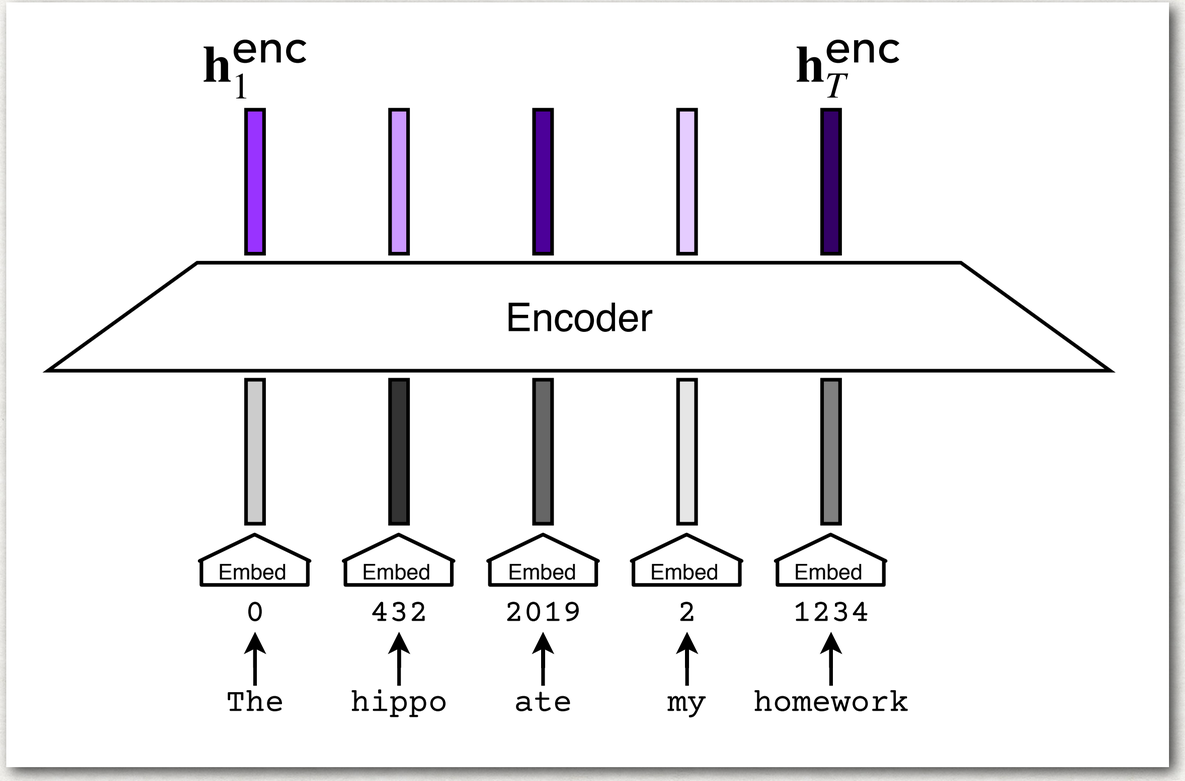
- Smaller vocab size
- Few to none out-of-vocabulary tokens

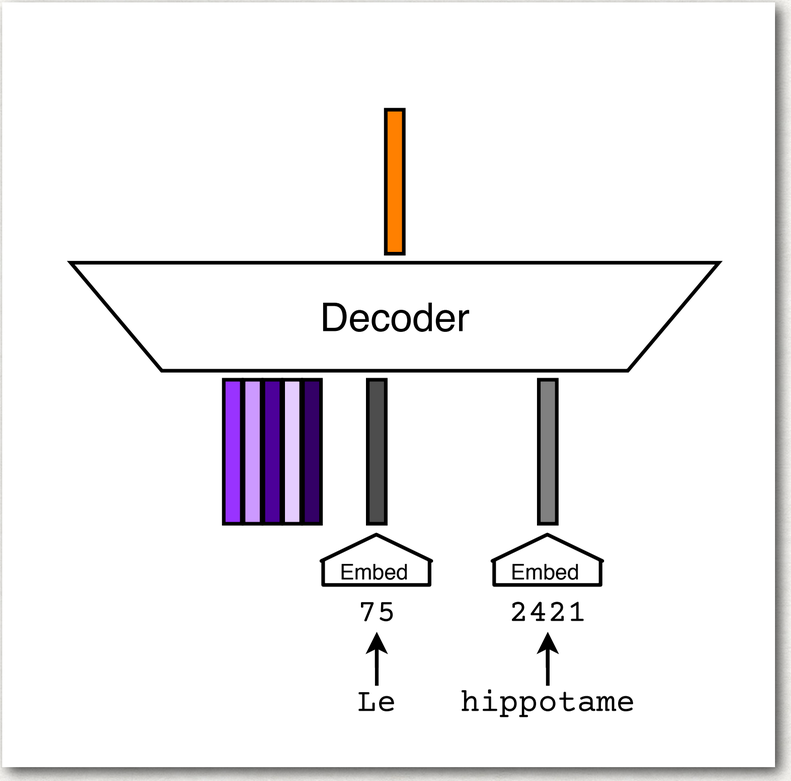
- Larger vocab size
- Greater potential for out-of-vocabulary tokens
- Tokens have more semantic meaning

RECURRENT NEURAL NETWORKS

ENCODER-DECODER ARCHITECTURES

How do we connect the encoder with the decoder?

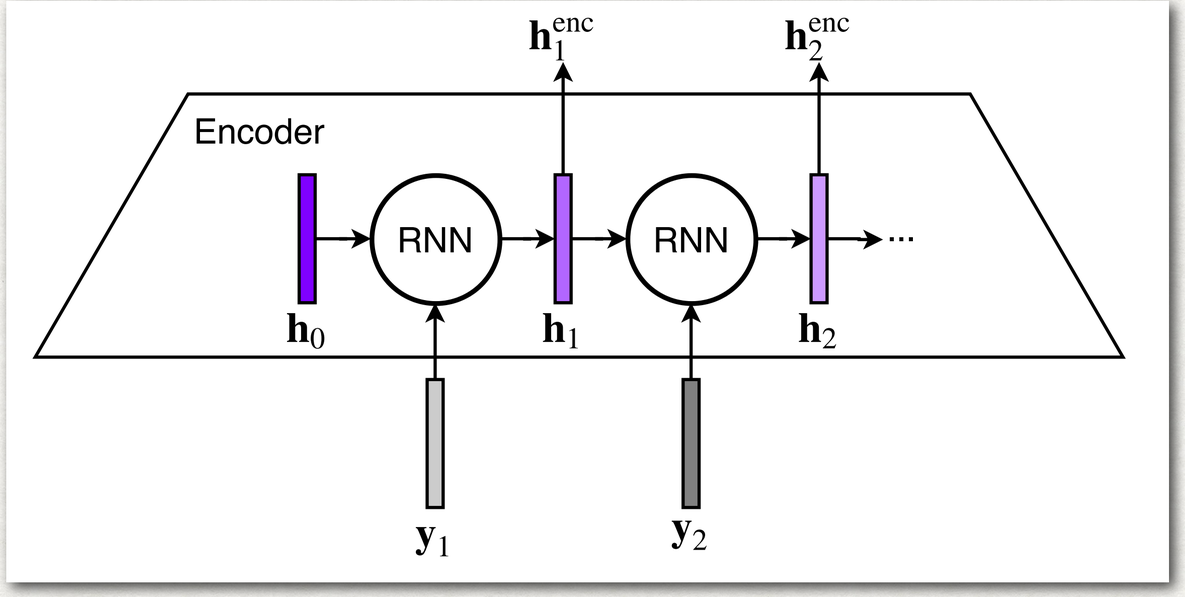


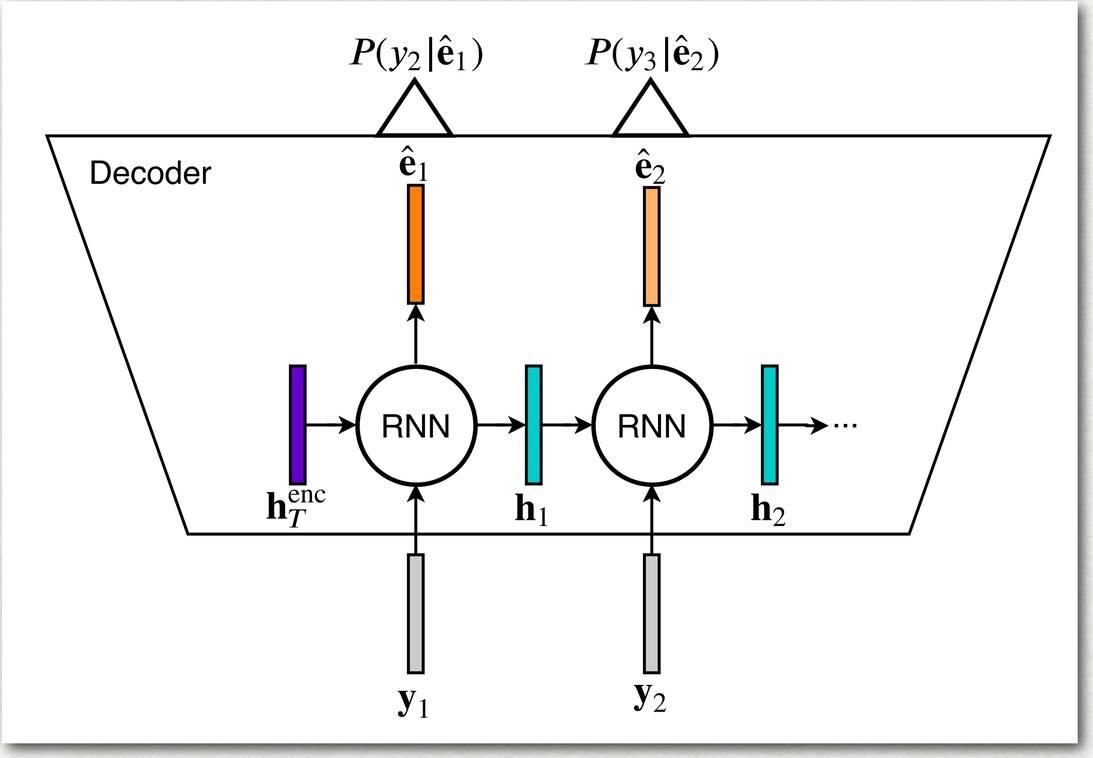


RECURRENT NEURAL NETWORKS

ENCODER-DECODER ARCHITECTURES

Simplest approach: Use the final hidden state from the encoder to initialize the first hidden state of the decoder.





RECURRENT NEURAL NETWORKS

ENCODER-DECODER ARCHITECTURES

Better approach: an attention mechanism

TRANSLATE FR TO EN

[The, hippopotamus, ...

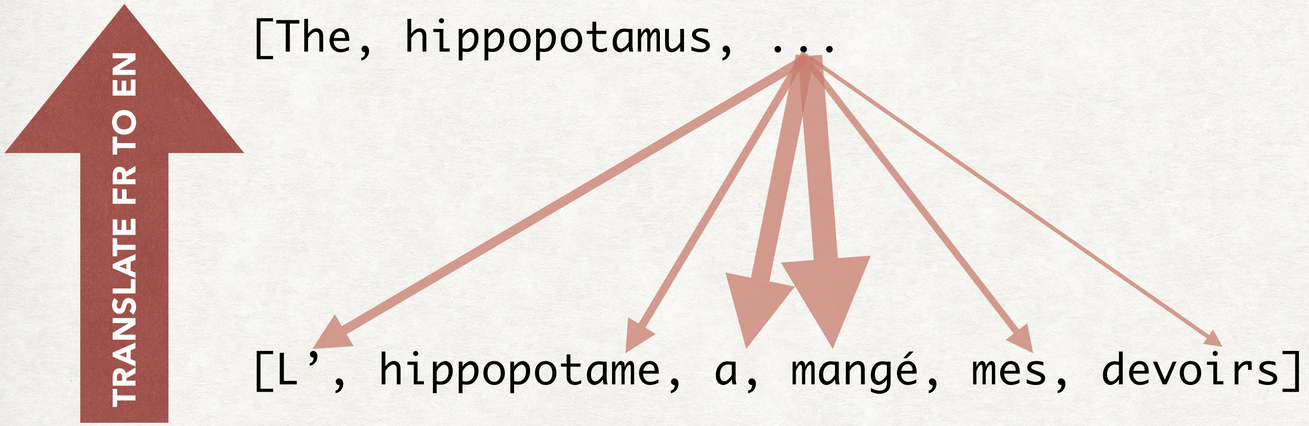
When predicting the next English word, how much weight should the model put on each French word in the source sequence?

[L’, hippopotame, a, mangé, mes, devoirs]

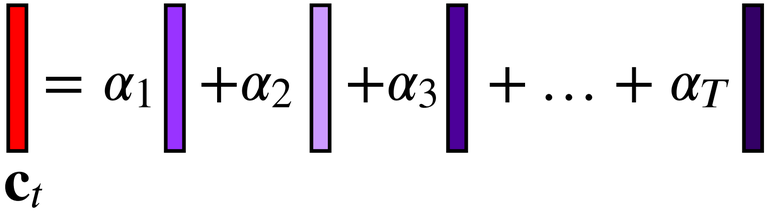
RECURRENT NEURAL NETWORKS

ENCODER-DECODER ARCHITECTURES

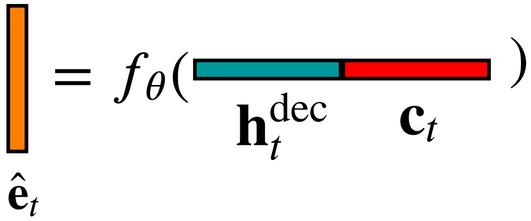
Better approach: an attention mechanism



Compute a linear combination of the encoder hidden states.



Decoder's prediction at position *t* is based on both the context vector and the hidden state outputted by the RNN at that position.



RECURRENT NEURAL NETWORKS

ENCODER-DECODER ARCHITECTURES

Better approach: an attention mechanism

$$\mathbf{c}_t = \mathbf{H}^{\text{enc}} \alpha_t$$

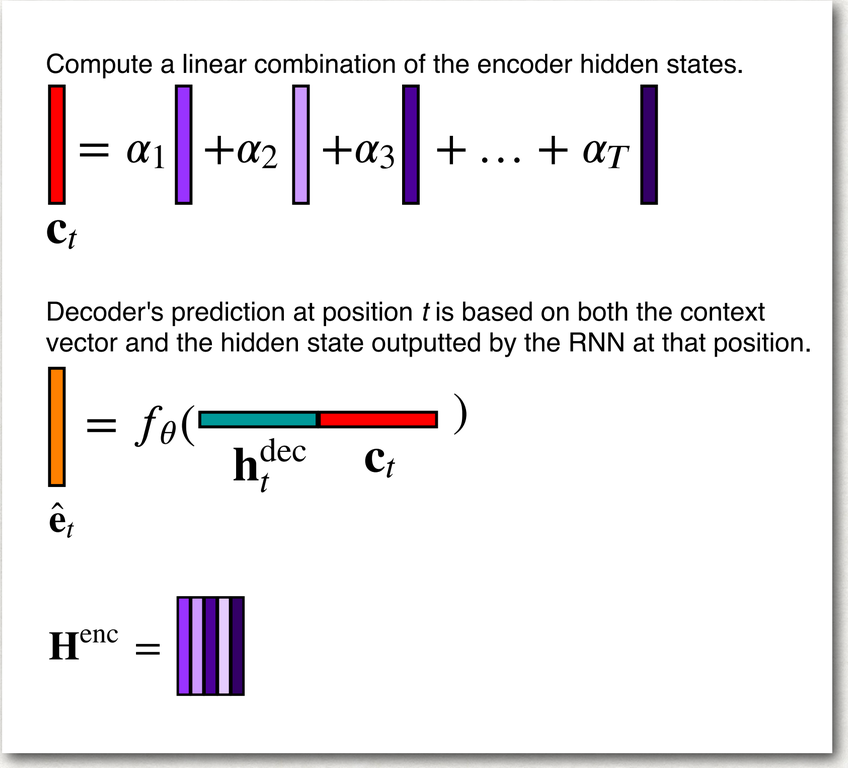
$$\alpha_t[i] = \text{softmax}(\text{att_score}(\mathbf{h}_t^{\text{dec}}, \mathbf{h}_i^{\text{enc}}))$$

There are a few different options for the attention score:

$$\text{att_score}(\mathbf{h}_t^{\text{dec}}, \mathbf{h}_i^{\text{enc}}) = \begin{cases} \mathbf{h}_t^{\text{dec}} \cdot \mathbf{h}_i^{\text{enc}} & \text{dot product} \\ \mathbf{h}_t^{\text{dec}} \mathbf{W}_a \mathbf{h}_i^{\text{enc}} & \text{bilinear function} \\ \mathbf{w}_{a1}^{\top} \tanh\left(\mathbf{W}_{a2}[\mathbf{h}_t^{\text{dec}}, \mathbf{h}_i^{\text{enc}}]\right) & \text{MLP} \end{cases}$$

The context and encoder hidden state can be concatenated together and passed through a small feed-forward network which predicts an output embedding for position i.

$$\hat{\mathbf{e}}_t = f_\theta([\mathbf{c}_t; \mathbf{h}_t^{\text{dec}}])$$



RECURRENT NEURAL NETWORKS

ENCODER-DECODER ARCHITECTURES

Better approach: an attention mechanism

$$\mathbf{c}_t = \mathbf{H}^{\text{enc}} \alpha_t$$

$$\alpha_t[i] = \text{softmax}(\text{att_score}(\mathbf{h}_t^{\text{dec}}, \mathbf{h}_i^{\text{enc}}))$$

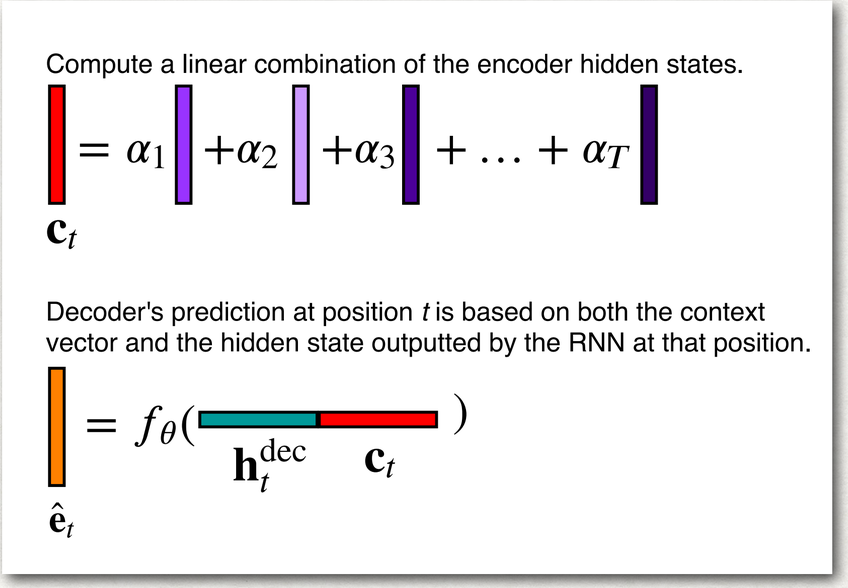
Foreshadowing: This is the score that will be used in the Transformer.

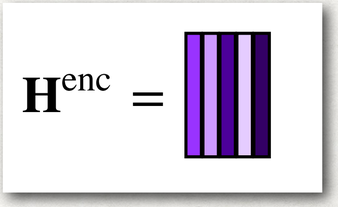
There are a few different options for the attention score:

$$\text{att_score}(\mathbf{h}_t^{\text{dec}}, \mathbf{h}_i^{\text{enc}}) = \begin{cases} \mathbf{h}_t^{\text{dec}} \cdot \mathbf{h}_i^{\text{enc}} & \text{dot product} \\ \mathbf{h}_t^{\text{dec}} \mathbf{W}_a \mathbf{h}_i^{\text{enc}} & \text{bilinear function} \\ \mathbf{w}_{a1}^{\top} \tanh\left(\mathbf{W}_{a2}[\mathbf{h}_t^{\text{dec}}, \mathbf{h}_i^{\text{enc}}]\right) & \text{MLP} \end{cases}$$

The context and encoder hidden state can be concatenated together and passed through a small feed-forward network which predicts an output embedding for position *i*.

$$\hat{\mathbf{e}}_t = f_\theta([\mathbf{c}_t; \mathbf{h}_t^{\text{dec}}])$$





RECURRENT NEURAL NETWORKS

LIMITATIONS

- Recurrent neural networks are slow to train. The computation at position *t* is dependent on first doing the computation at position *t-1*.
- LSTMs were design to keep important information in the hidden state's memory for longer (than simpler RNN units). However they are still not great at this.
  - If two tokens are K positions apart, there are K opportunities for knowledge of the first token to be erased from the hidden state before a prediction is made at the position of the second token.
- To combat the forgetting, encoder networks are often bidirectional: one LSTM runs through the sequence left-to-right, and another runs through right-to-left. The outputs are concatenated.
  - This is a kludge rather than a real solution.

OUTLINE

- Text generation before neural networks
- Neural language model framework
- LM Architectures
  - Recurrent neural networks
  - **Convolutional neural networks**
  - Transformers
- Takeaways

CONVOLUTIONAL NEURAL NETWORKS

REFERENCED PAPERS

Convolutional Sequence to Sequence Learning

Jonas Gehring
Michael Auli
David Grangier
Denis Yarats
Yann N. Dauphin
Facebook AI Research

3122v3 [cs.CL] 25 Jul 2017

Abstract

The prevalent approach to sequence to sequence learning maps an input sequence to a variable length output sequence via recurrent neural networks. We introduce an architecture based entirely on convolutional neural networks.[1] Compared to recurrent models, computations over all elements can be fully parallelized during training to better exploit the GPU hardware and optimization is easier since the number of non-linearities is fixed and independent of the input length. Our use of gated linear units eases gradient propagation and we equip each decoder layer with a separate attention module. We outperform the accuracy of the deep LSTM setup of Wu et al. (2016) on both WMT'14 English-German and WMT'14 English-French translation at an order of magnitude faster speed, both on GPU and CPU.

Convolutional neural networks are less common for sequence modeling, despite several advantages (Waibel et al., 1989; LeCun & Bengio, 1995). Compared to recurrent layers, convolutions create representations for fixed size contexts, however, the effective context size of the network can easily be made larger by stacking several layers on top of each other. This allows to precisely control the maximum length of dependencies to be modeled. Convolutional networks do not depend on the computations of the previous time step and therefore allow parallelization over every element in a sequence. This contrasts with RNNs which maintain a hidden state of the entire past that prevents parallel computation within a sequence.

Multi-layer convolutional neural networks create hierarchical representations over the input sequence in which nearby input elements interact at lower layers while distant elements interact at higher layers. Hierarchical structure provides a shorter path to capture long-range dependencies compared to the chain structure modeled by recurrent net-

Language Modeling with Gated Convolutional Networks

Yann N. Dauphin [1] **Angela Fan** [1] **Michael Auli** [1] **David Grangier** [1]

08083v3 [cs.CL] 8 Sep 2017

Abstract

The pre-dominant approach to language modeling to date is based on recurrent neural networks. Their success on this task is often linked to their ability to capture unbounded context. In this paper we develop a finite context approach through stacked convolutions, which can be more efficient since they allow parallelization over sequential tokens. We propose a novel simplified gating mechanism that outperforms Oord et al. (2016b) and investigate the impact of key architectural decisions. The proposed approach achieves state-of-the-art on the WikiText-103 benchmark, even though it features long-term dependencies, as well as competitive results on the Google Billion Words benchmark. Our model reduces the latency to score a sentence by an order of magnitude compared to a recurrent baseline. To our knowledge, this is the first time a non-recurrent approach is competitive with strong recurrent models on these large scale language tasks.

outperform classical *n-gram* language models (Kneser & Ney, 1995; Chen & Goodman, 1996). These classical models suffer from data sparsity, which makes it difficult to represent large contexts and thus, long-range dependencies. Neural language models tackle this issue by embedding words in continuous space over which a neural network is applied. The current state of the art for language modeling is based on long short term memory networks (LSTM; Hochreiter et al., 1997) which can theoretically model arbitrarily long dependencies.

In this paper, we introduce new gated convolutional networks and apply them to language modeling. Convolutional networks can be stacked to represent large context sizes and extract hierarchical features over larger and larger contexts with more abstractive features (LeCun & Bengio, 1995). This allows them to model long-term dependencies by applying $\mathcal{O}(\frac{N}{k})$ operations over a context of size N and kernel width k. In contrast, recurrent networks view the input as a chain structure and therefore require a linear number $\mathcal{O}(N)$ of operations.

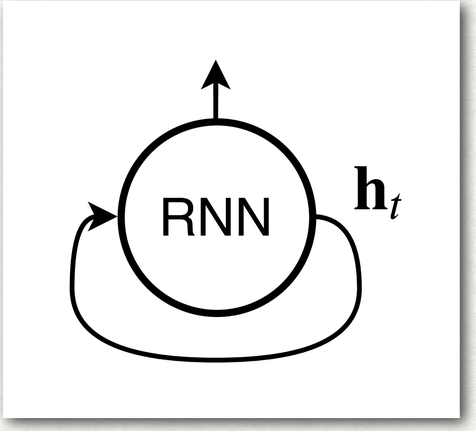
Analyzing the input hierarchically bears resemblance to classical grammar formalisms which build syntactic tree

TRANSFORMERS

FEED FORWARD VS RECURRENT NEURAL NETWORK

Recurrent neural networks have feedback loops.

Feed forward neural networks are ones without any loops.



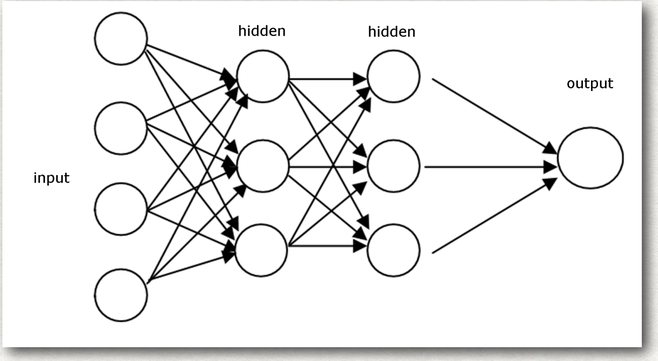


Image source: https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781788397872/1/ch01lvl1sec21/feed-forward-and-feedback-networks

CONVOLUTIONAL NEURAL NETWORKS

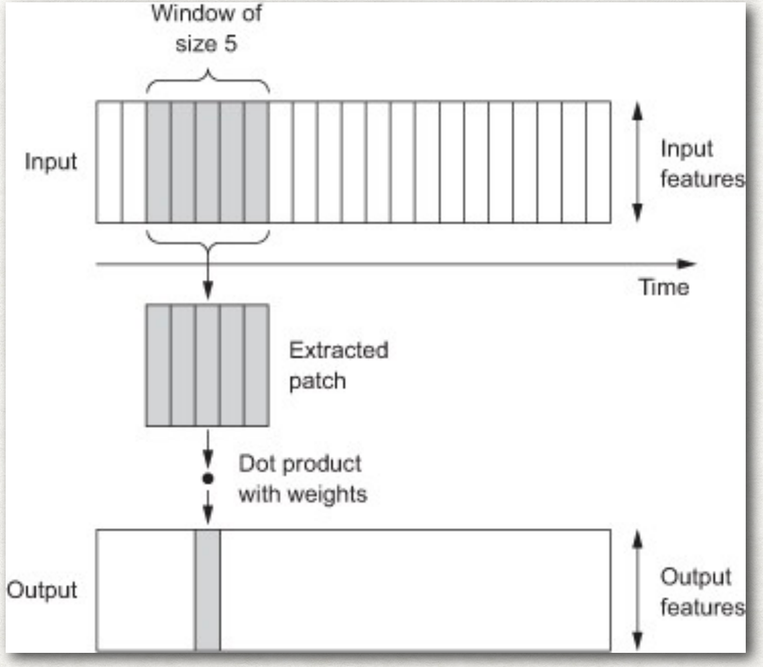
WHAT IS A 1D CONVOLUTION?

Suppose we have a sequence $\mathbf{x}_1, \ldots, \mathbf{x}_N$ of vectors which we concatenate into a matrix $\mathbf{X} \in \mathbb{R}^{N \times d_{\text{in}}}$.

We then convolve $\mathbf{X}$ with a weight matrix $\mathbf{W} \in \mathbb{R}^{k \times d_{\text{in}} \times d_{\text{out}}}$ resulting in a new matrix $(\mathbf{X} * \mathbf{W}) \in \mathbb{R}^{N \times d_{\text{out}}}$.

The math for the convolution is:

$$\mathbf{X} * \mathbf{W} = \left\{ \sum_{j=0}^{k-1} \mathbf{X}[t-j, :]\mathbf{W}[j, :, :] \right\}_{t=1}^{N}$$



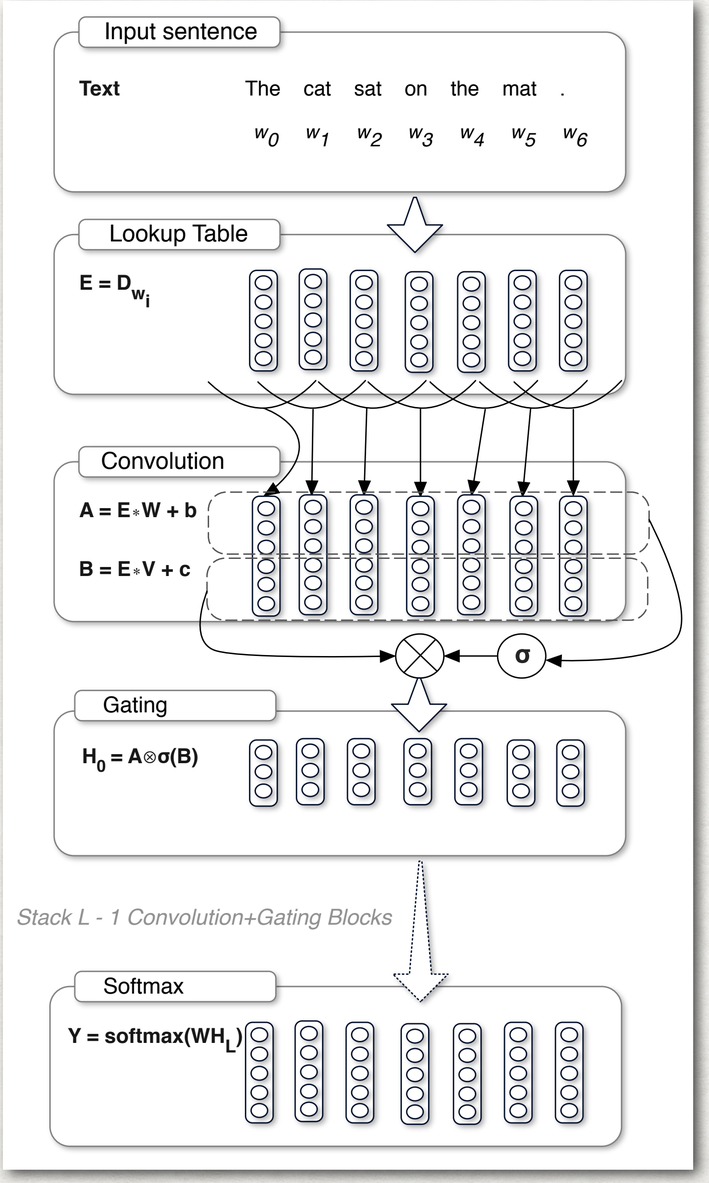
Source: https://medium.com/@jon.froiland/convolutional-neural-networks-for-sequence-processing-part-1-420dd9b500

CONVOLUTIONAL NEURAL NETWORKS

ARCHITECTURE

Let's go through their model step-by-step.

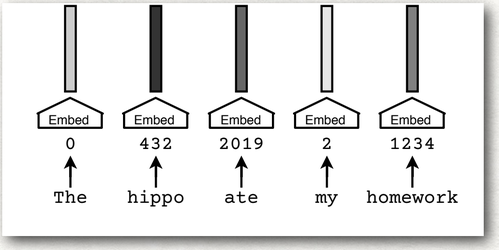
Warning: The paper uses different notation than I've used thus far, but we'll be going with the notation in their diagram.

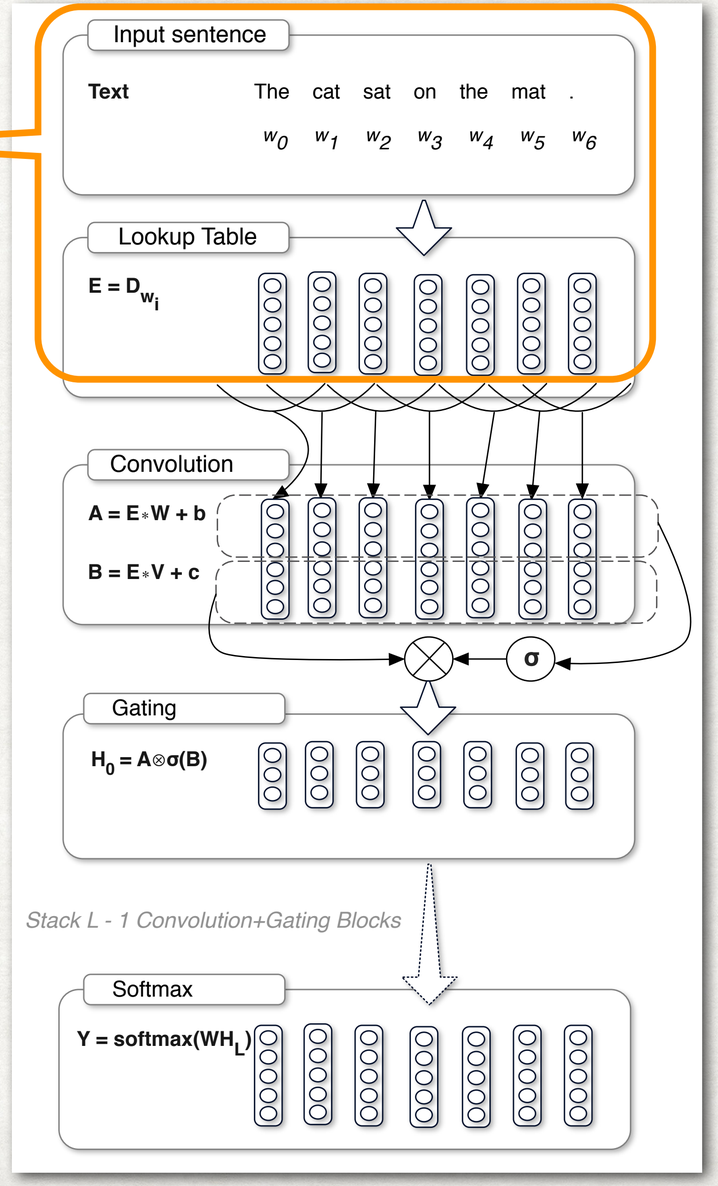


CONVOLUTIONAL NEURAL NETWORKS

ARCHITECTURE

This should look familiar.

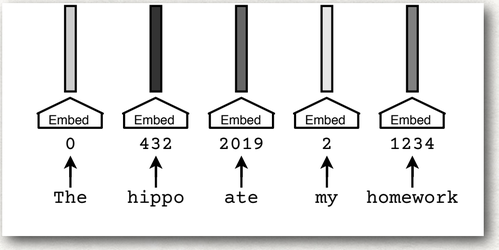




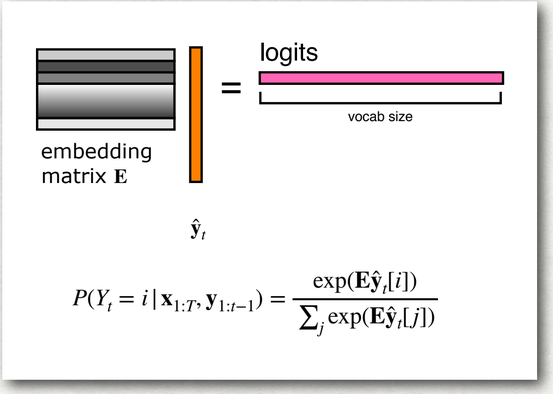
CONVOLUTIONAL NEURAL NETWORKS

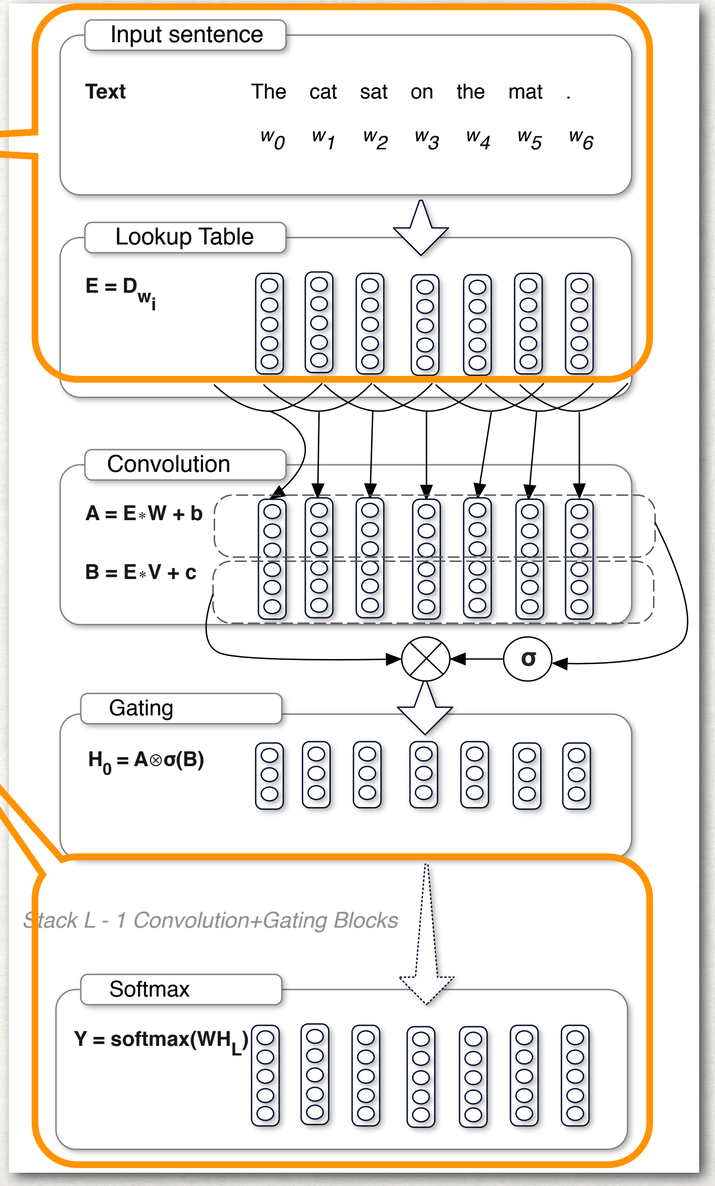
ARCHITECTURE

This should look familiar.



This should also look familiar.





CONVOLUTIONAL NEURAL NETWORKS

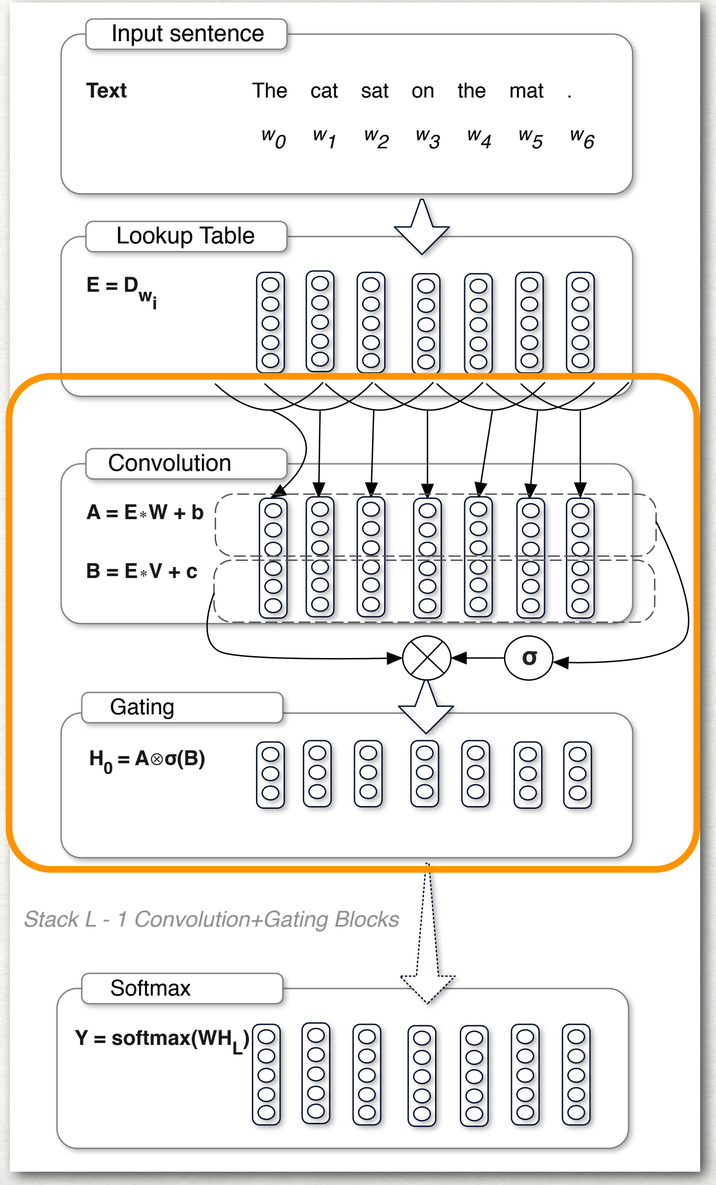
ARCHITECTURE

Let $\mathbf{X} \in R^{N \times d_{l-1}}$ be the input to the lth layer, where N is the sequence length and d_{l-1} is the number of feature maps outputted by the previous layer (or the length of the input sequence in the case of d_0).

The output of the lth layer is defined as:

$$\mathbf{H}_l(\mathbf{X}) = (\mathbf{X} * \mathbf{W}_{\text{proj}} + \mathbf{b}_{\text{proj}}) \otimes \sigma(\mathbf{X} * \mathbf{W}_{\text{gate}} + \mathbf{b}_{\text{gate}})$$

where $\mathbf{W}_{\text{proj}}, \mathbf{W}_{\text{gate}} \in R^{k \times d_{i-1} \times d_i}$ and $\mathbf{b}_{\text{proj}}, \mathbf{b}_{\text{gate}} \in R^{d_i}$



CONVOLUTIONAL NEURAL NETWORKS

ARCHITECTURE

Let $\mathbf{X} \in R^{N \times d_{l-1}}$ be the input to the lth layer, where N is the sequence length and d_{l-1} is the number of feature maps outputted by the previous layer (or the number of dimensions of the input embeddings in the case of d_0).

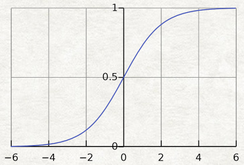
The output of the lth layer is defined as:

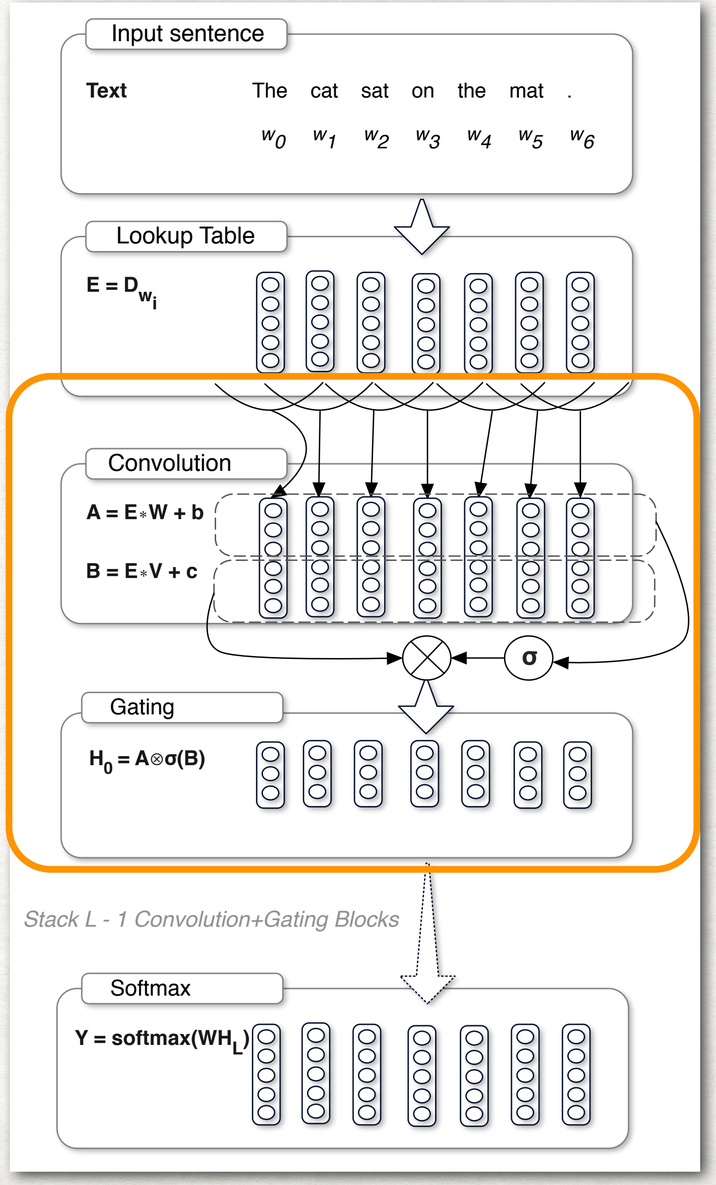
$$\mathbf{H}_l(\mathbf{X}) = (\mathbf{X} * \mathbf{W}_{\text{proj}} + \mathbf{b}_{\text{proj}}) \otimes \sigma(\mathbf{X} * \mathbf{W}_{\text{gate}} + \mathbf{b}_{\text{gate}})$$

where $\mathbf{W}_{\text{proj}}, \mathbf{W}_{\text{gate}} \in R^{k \times d_{i-1} \times d_i}$ and $\mathbf{b}_{\text{proj}}, \mathbf{b}_{\text{gate}} \in R^{d_i}$

element-wise multiplication

sigmoid function





CONVOLUTIONAL NEURAL NETWORKS

ARCHITECTURE

Let $\mathbf{X} \in R^{N \times d_{l-1}}$ be the input to the lth layer, where N is the sequence length and d_{l-1} is the number of feature maps outputted by the previous layer (or the number of dimensions of the input embeddings in the case of d_0).

The output of the lth layer is defined as:

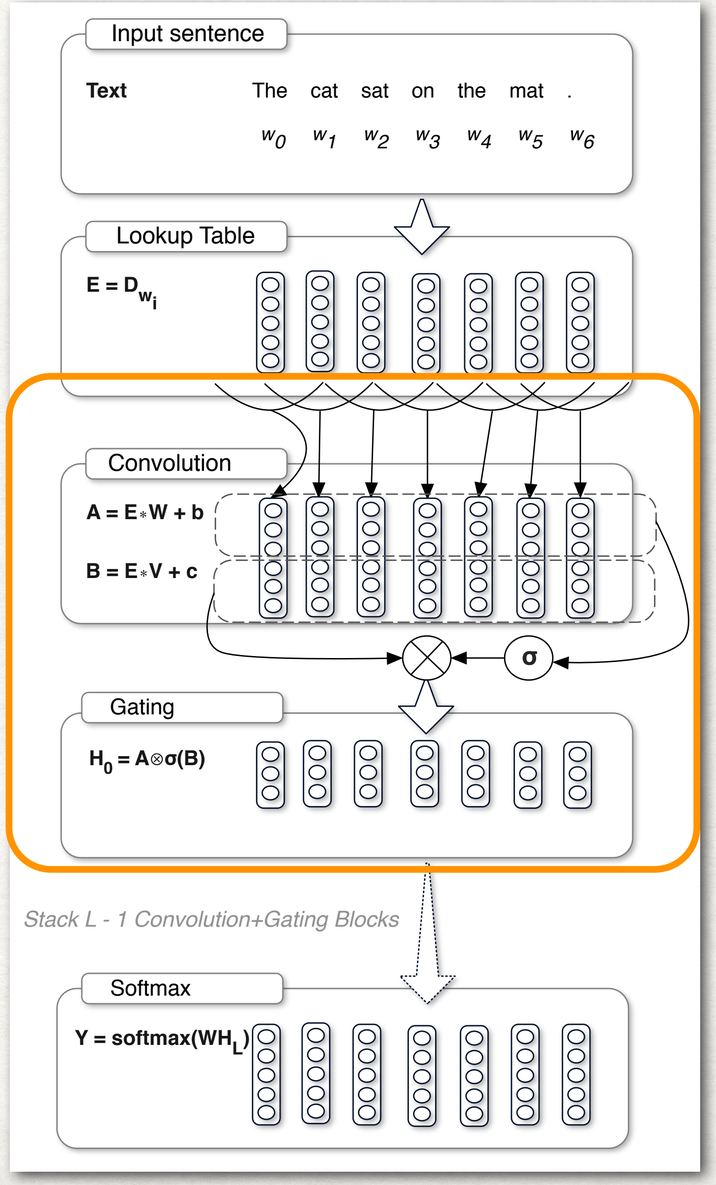
$$\mathbf{H}_l(\mathbf{X}) = \boxed{(\mathbf{X} * \mathbf{W}_{\text{proj}} + \mathbf{b}_{\text{proj}})} \otimes \boxed{\sigma(\mathbf{X} * \mathbf{W}_{\text{gate}} + \mathbf{b}_{\text{gate}})}$$

where $\mathbf{W}_{\text{proj}}, \mathbf{W}_{\text{gate}} \in R^{k \times d_{i-1} \times d_i}$ and $\mathbf{b}_{\text{proj}}, \mathbf{b}_{\text{gate}} \in R^{d_i}$

projection

gating mechanism

The gating mechanism is motivated by the LSTM equations I glossed over several slides ago. The sigmoid gating mechanism "allows the model to select which words or features are relevant for predicting the next word."



OUTLINE

- Text generation before neural networks
- Neural language model framework
- LM Architectures
  - Recurrent neural networks
  - Convolutional neural networks
  - **Transformers**
- Takeaways

TRANSFORMERS

REFERENCED PAPER

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

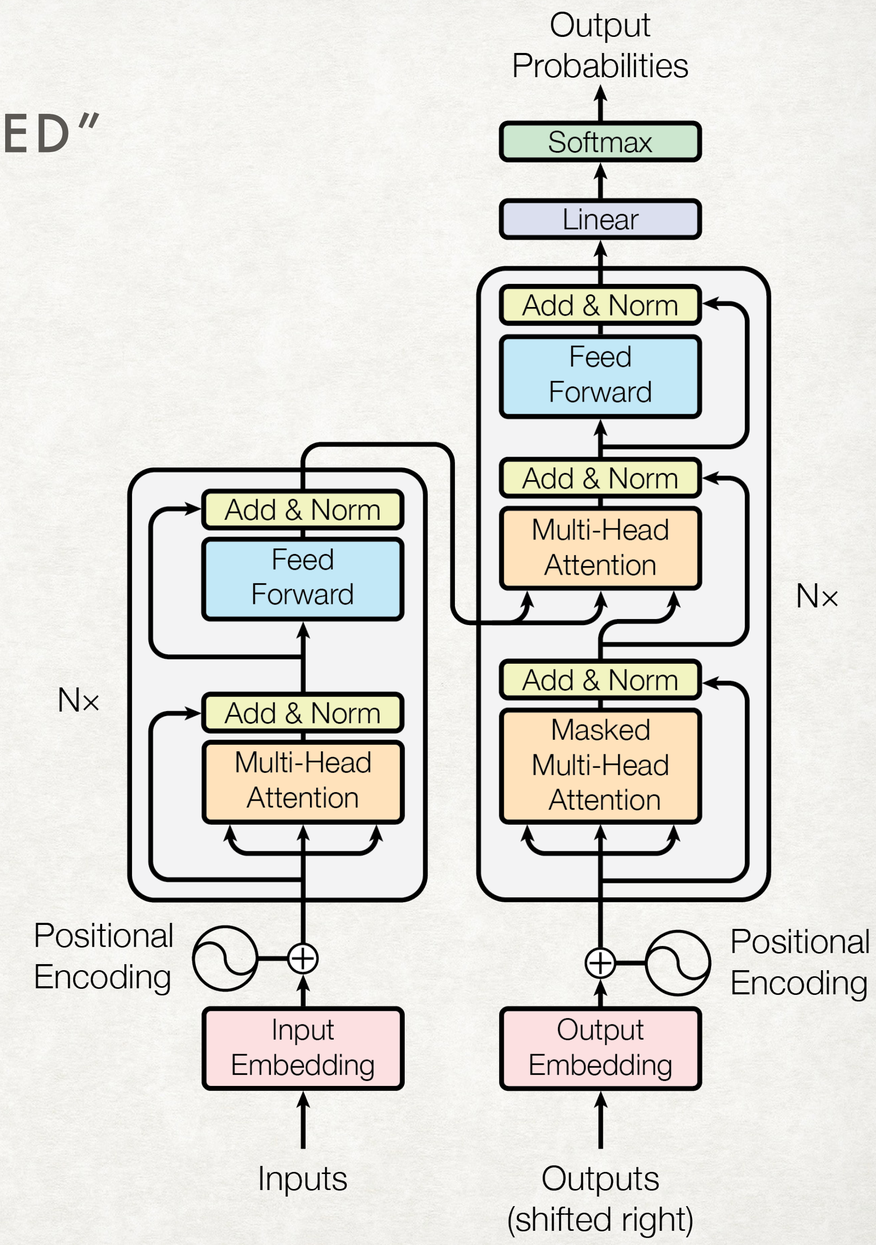
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

TRANSFORMERS

"ATTENTION IS ALL YOU NEED"

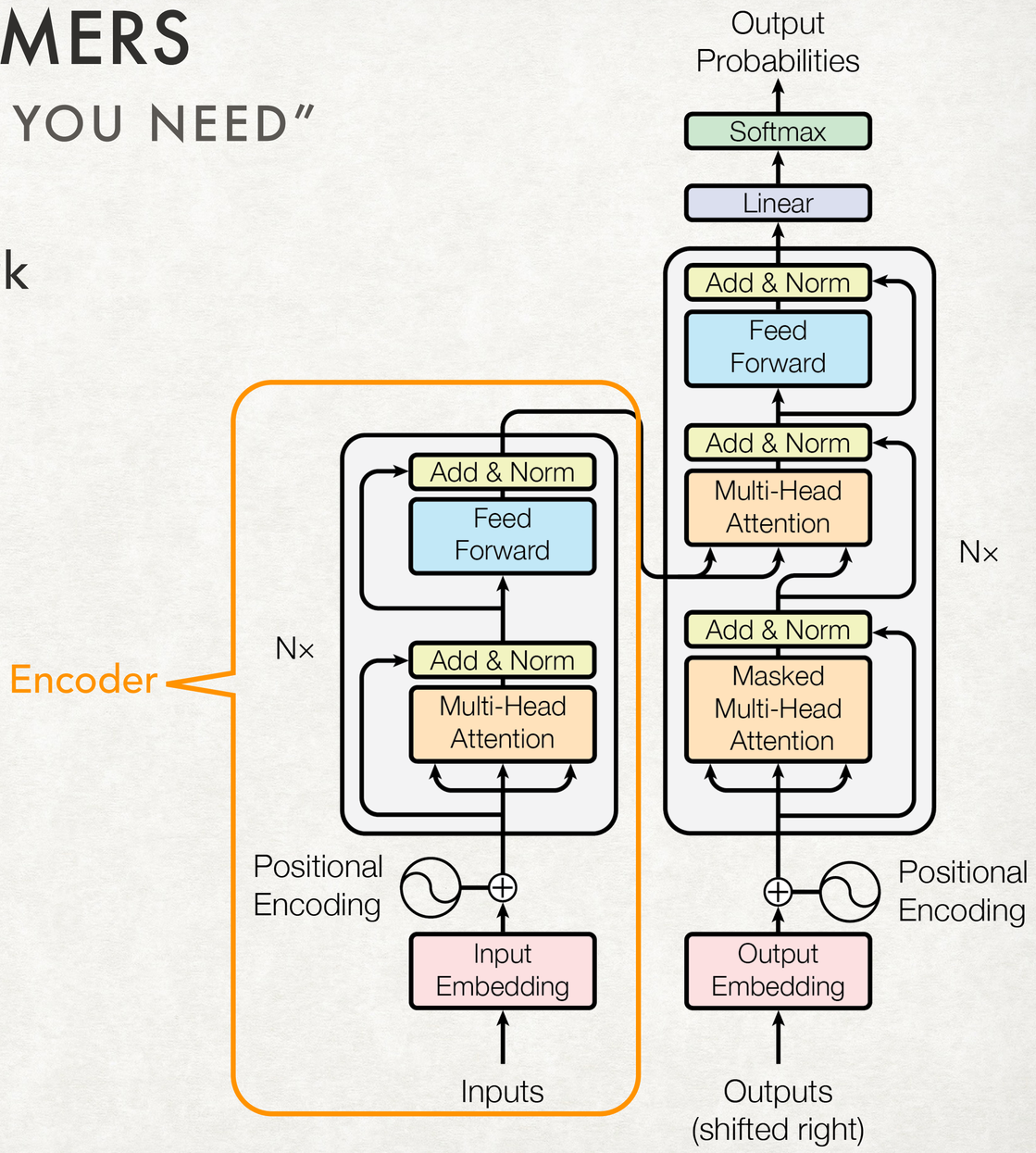
The Transformer is a non-recurrent non-convolutional neural network designed for language understanding that introduces self-attention in addition to encoder-decoder attention.



TRANSFORMERS

"ATTENTION IS ALL YOU NEED"

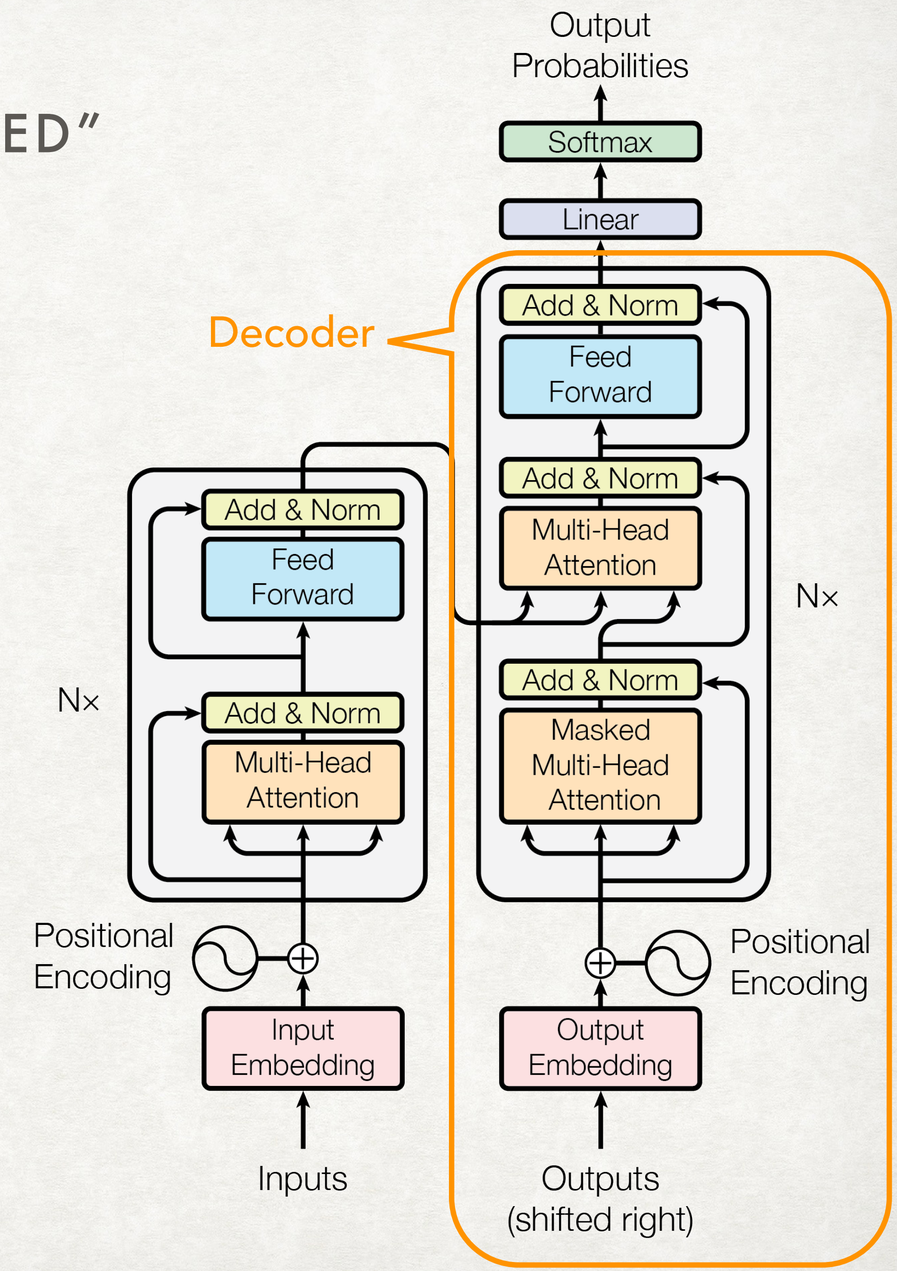
The Transformer: A feed-forward neural network designed for language understanding.



TRANSFORMERS

"ATTENTION IS ALL YOU NEED"

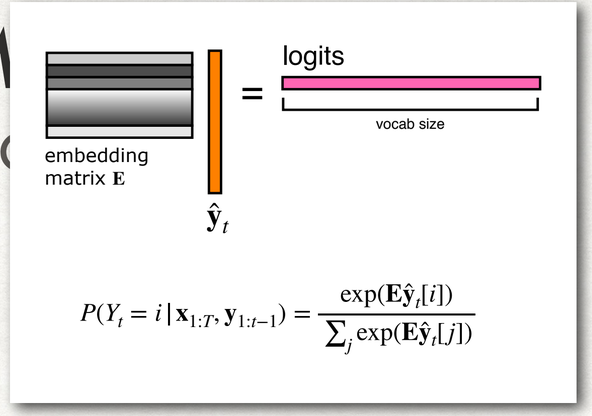
The Transformer: A feed-forward neural network designed for language understanding.

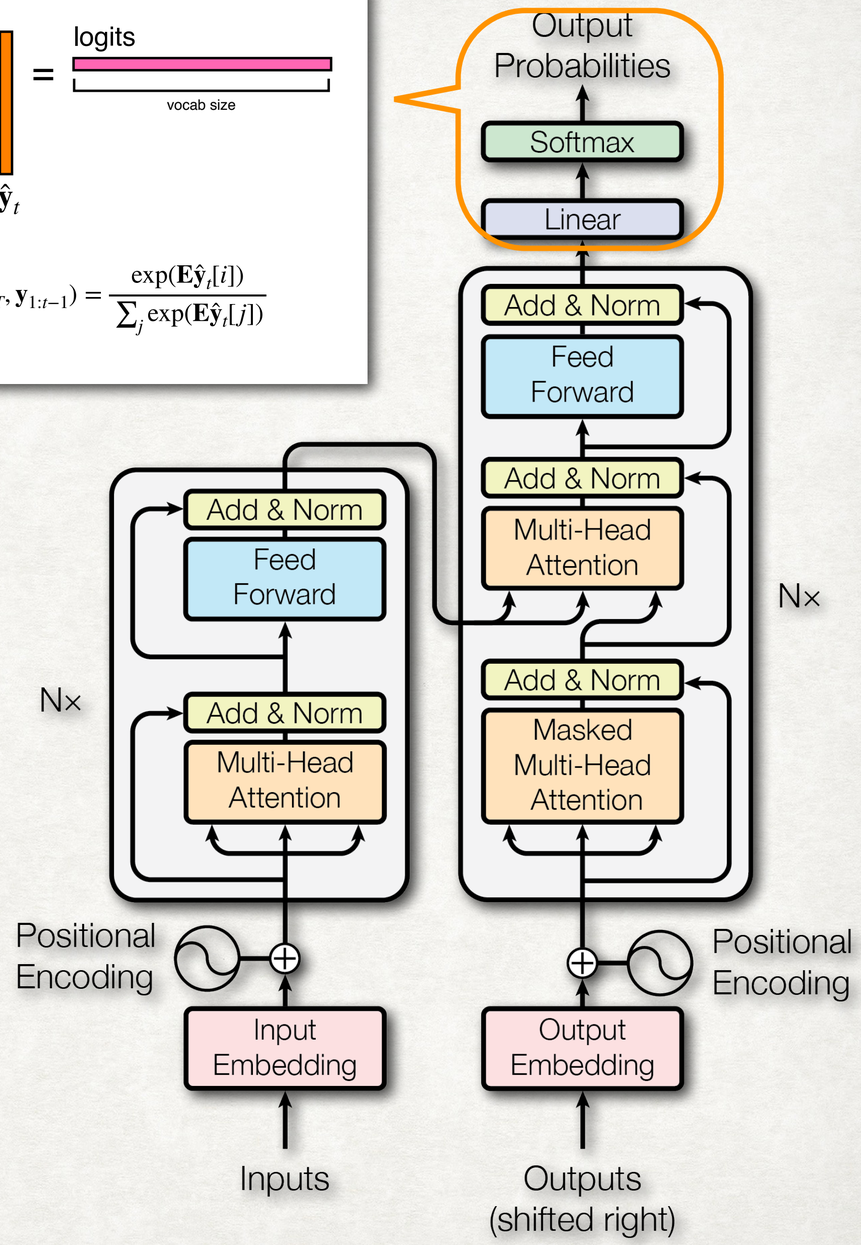


TRANSFORM

"ATTENTION IS ALL Y

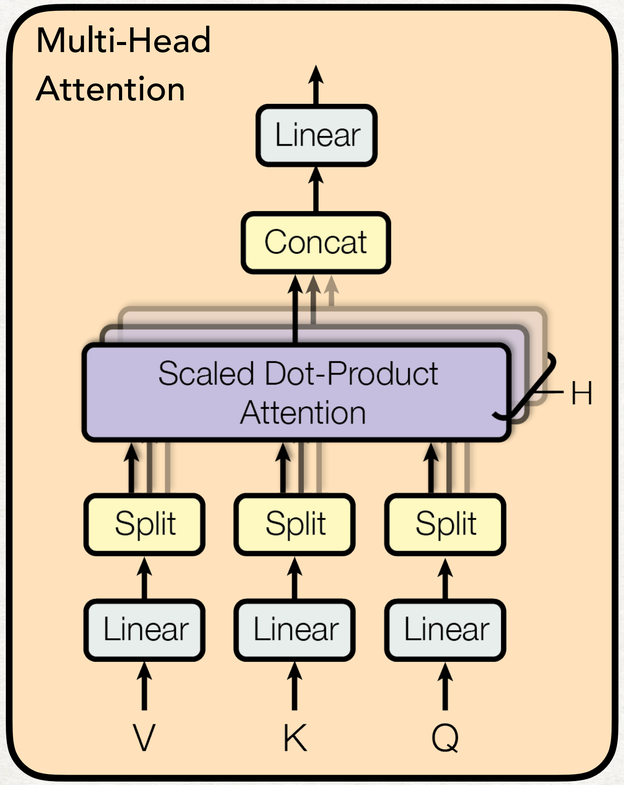
The Transformer: A feed-forward neural network designed for language understanding.

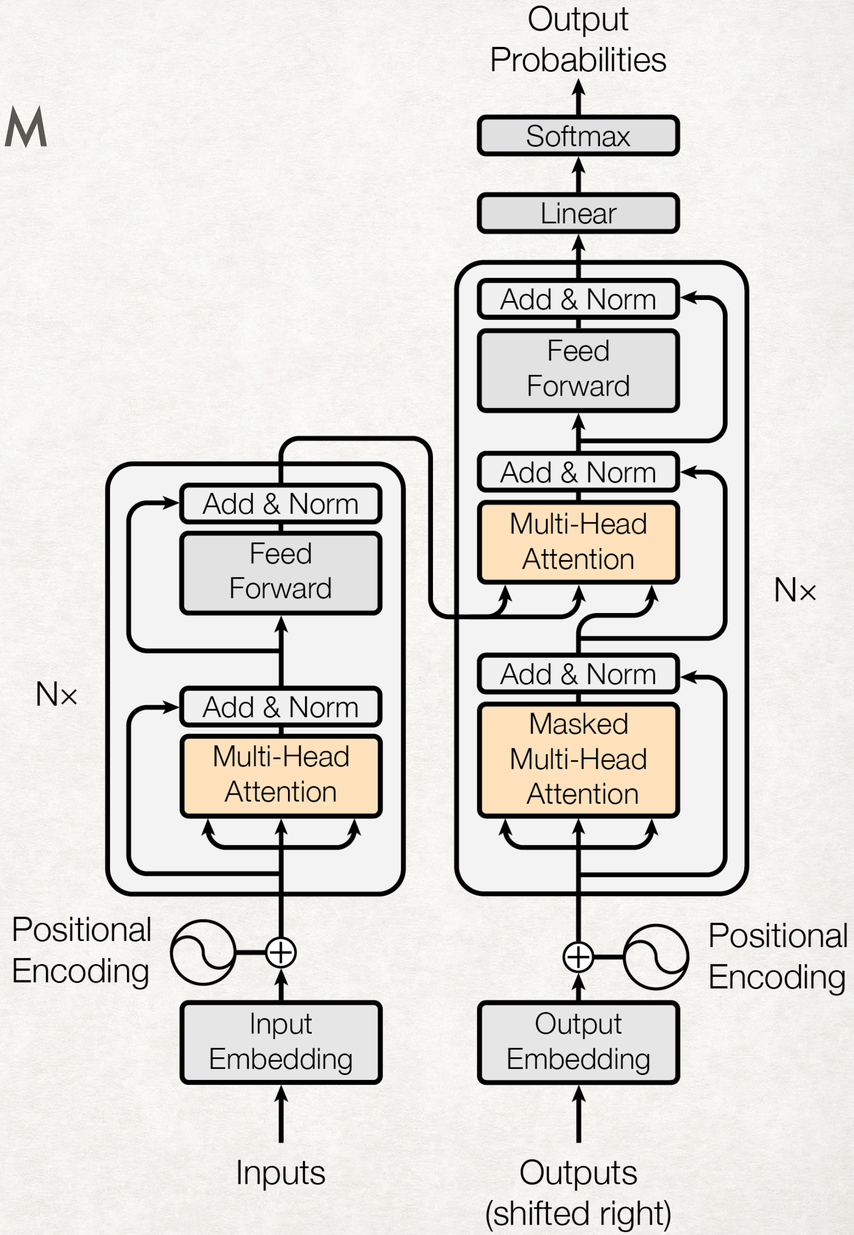




TRANSFORMERS

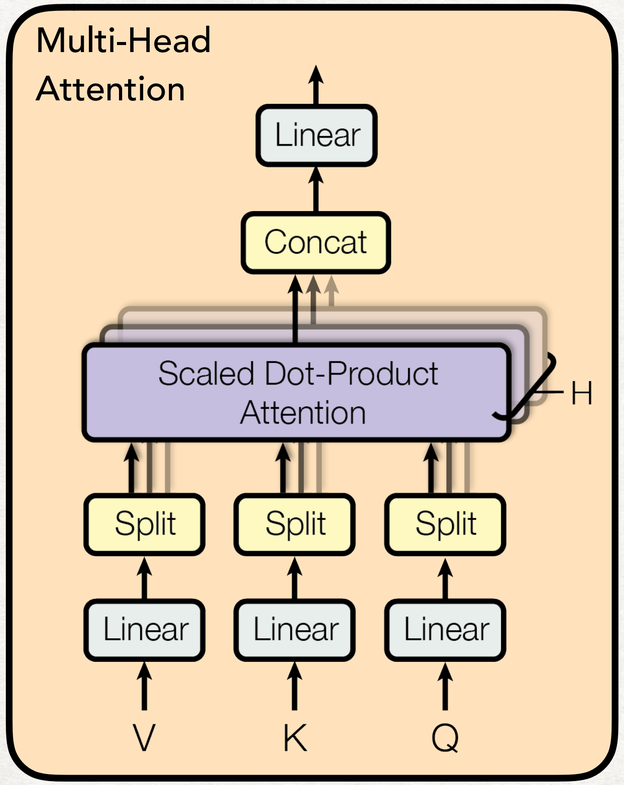
THE ATTENTION MECHANISM



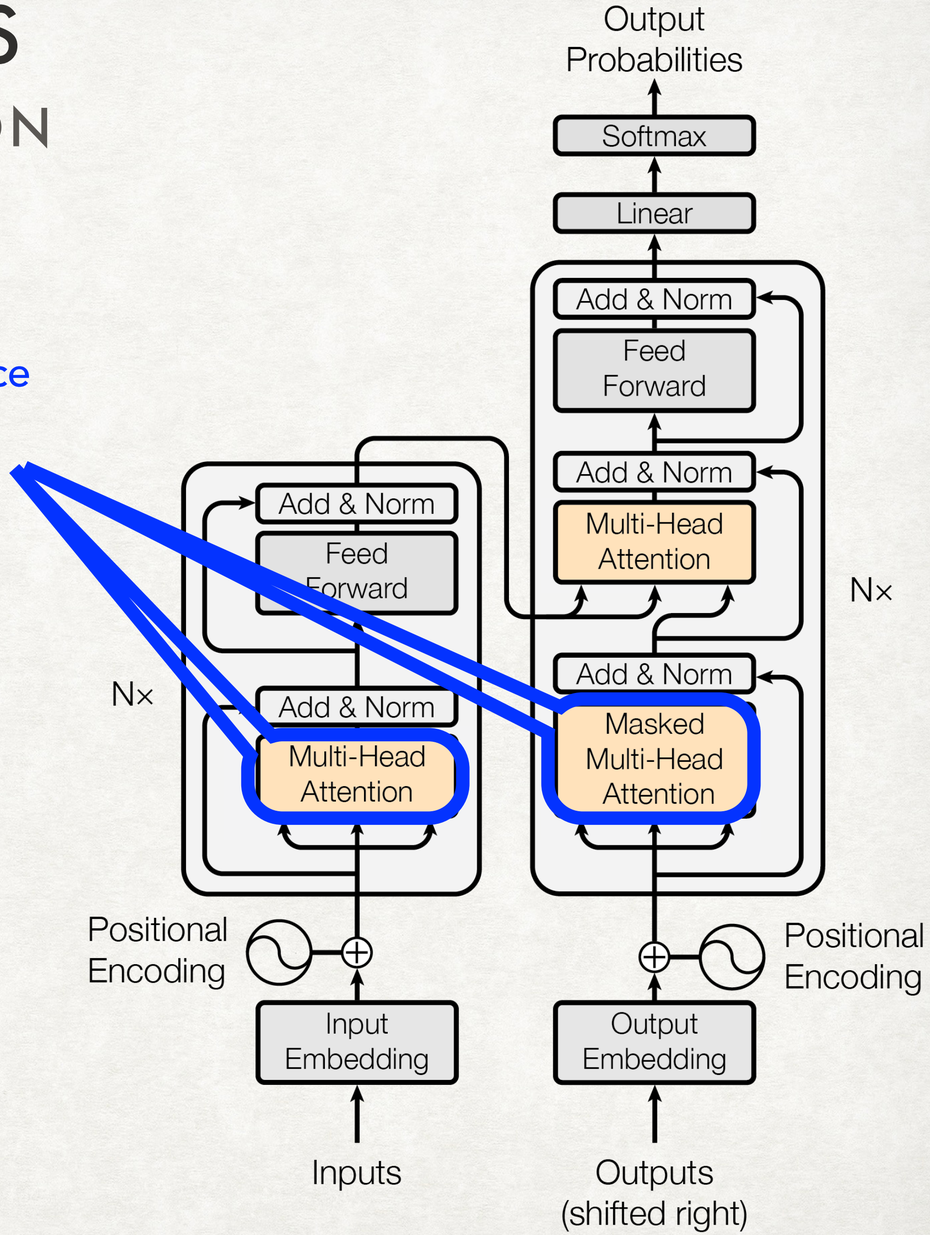


TRANSFORMERS

MULTI-HEAD ATTENTION



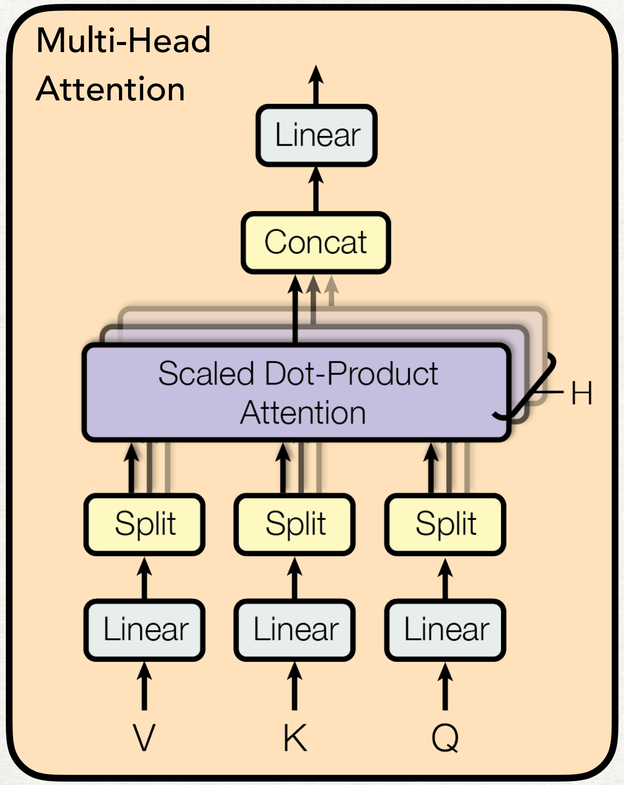
Self-attention between a sequence of hidden states and that same sequence of hidden states.

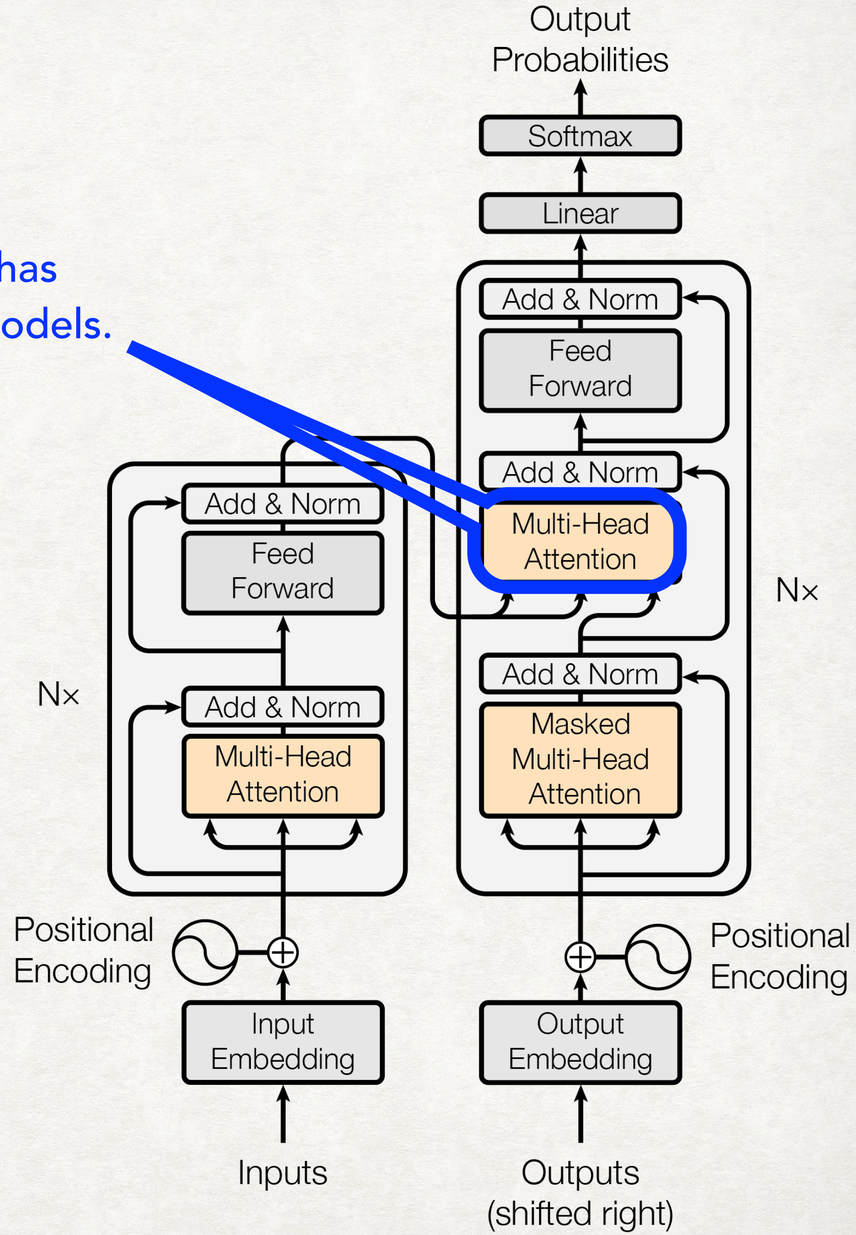


TRANSFORMERS

MULTI-HEAD ATTENTION

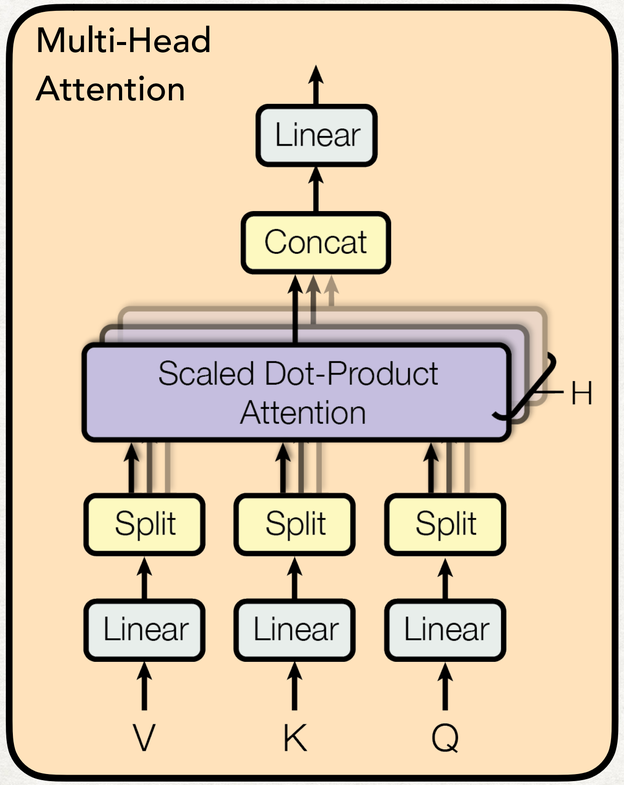
Encoder-decoder attention, like what has been standard in recurrent seq2seq models.

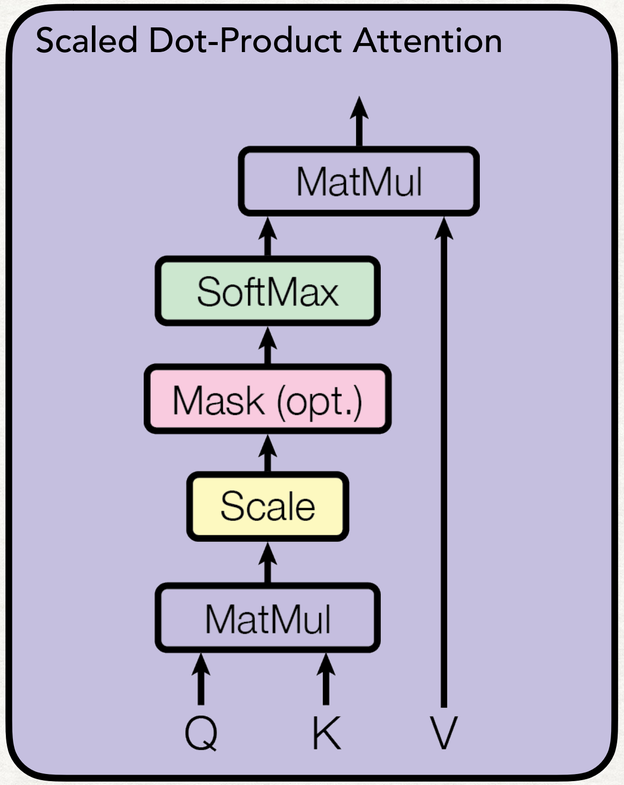


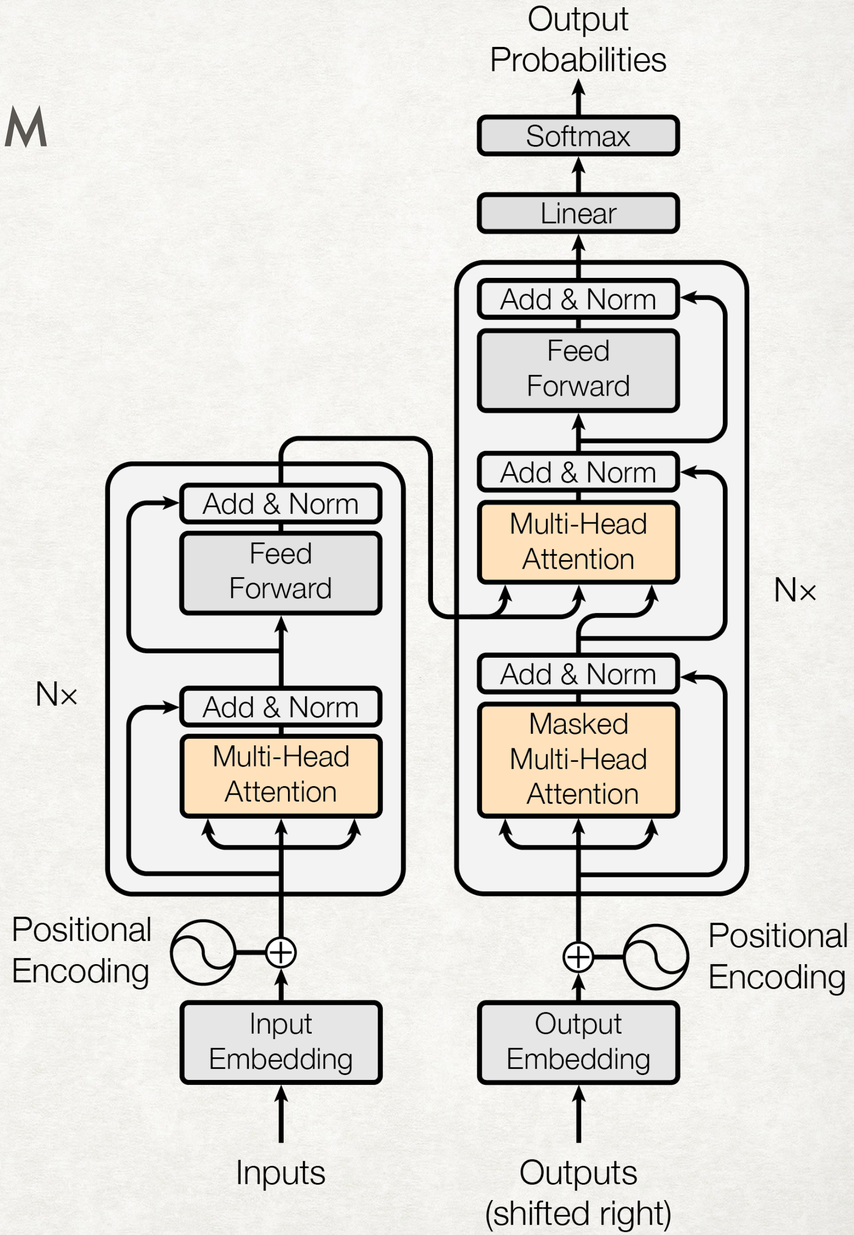


TRANSFORMERS

THE ATTENTION MECHANISM

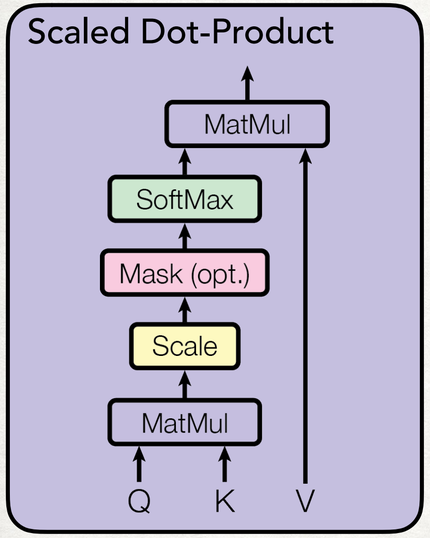






TRANSFORMERS

SCALED DOT-PRODUCT ATTENTION



The **scaled-dot product attention mechanism** is almost identical to the one we learned about in the previous section. However, we'll reformulate it in terms of matrix multiplications.

The query: $\mathbf{Q} \in R^{T \times d_k}$

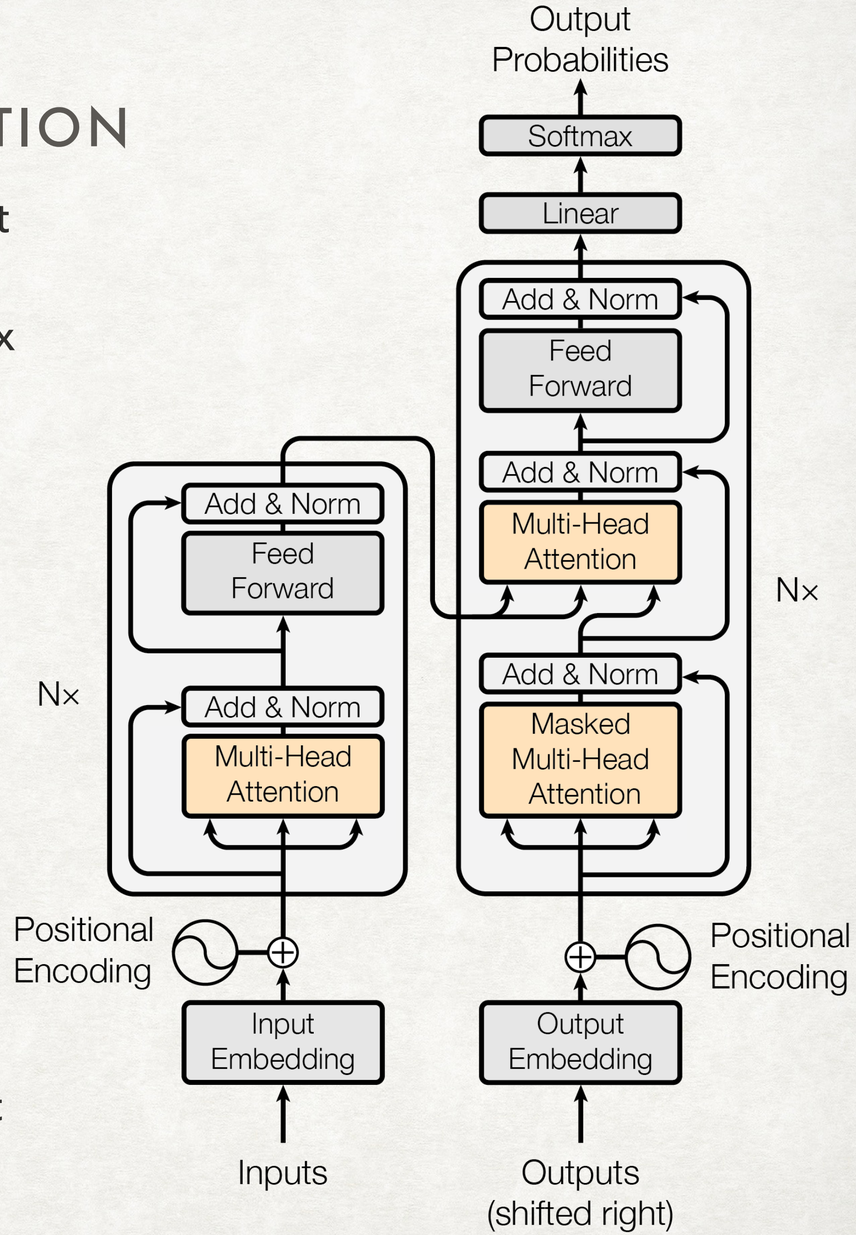
The key: $\mathbf{K} \in R^{T' \times d_k}$

The value: $\mathbf{V} \in R^{T \times d_v}$

This is the α vector from the previous formulation.

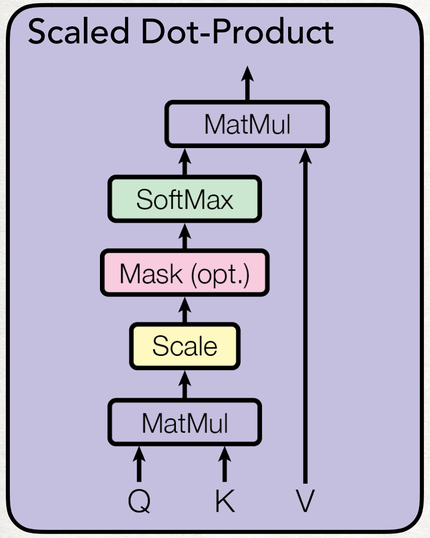
$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

The $\sqrt{d_k}$ in the denominator is there to prevent the dot product from getting too big.



TRANSFORMERS

SCALED DOT-PRODUCT ATTENTION



The **scaled-dot product attention mechanism** is almost identical to the one we learned about in the previous section. However, we'll reformulate it in terms of matrix multiplications.

The query: $\mathbf{Q} \in R^{T' \times d_k}$

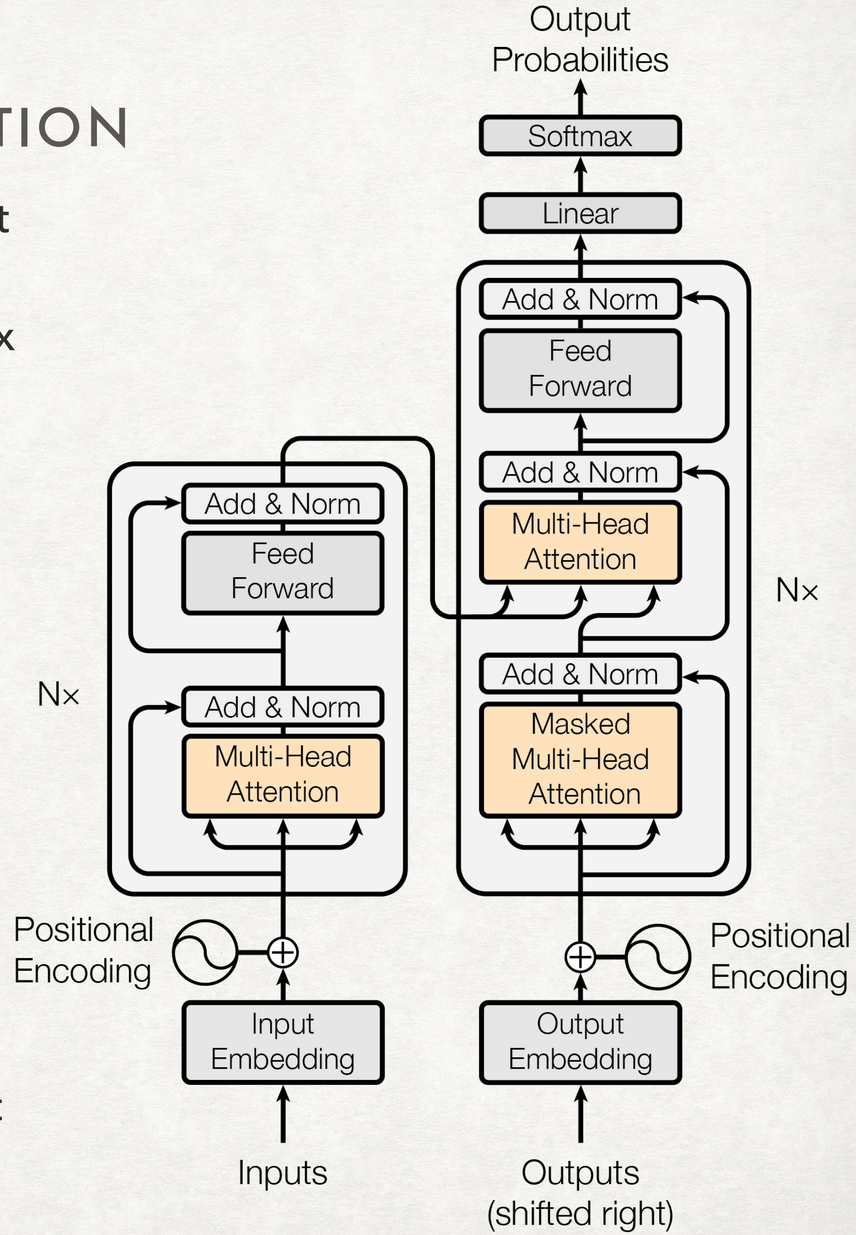
The key: $\mathbf{K} \in R^{T \times d_k}$

The value: $\mathbf{V} \in R^{T \times d_v}$

This is the dot-product scoring function we saw in the previous section.

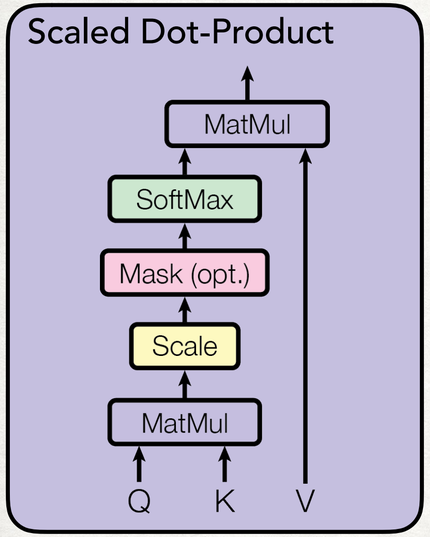
$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

The $\sqrt{d_k}$ in the denominator is there to prevent the dot product from getting too big.



TRANSFORMERS

SCALED DOT-PRODUCT ATTENTION



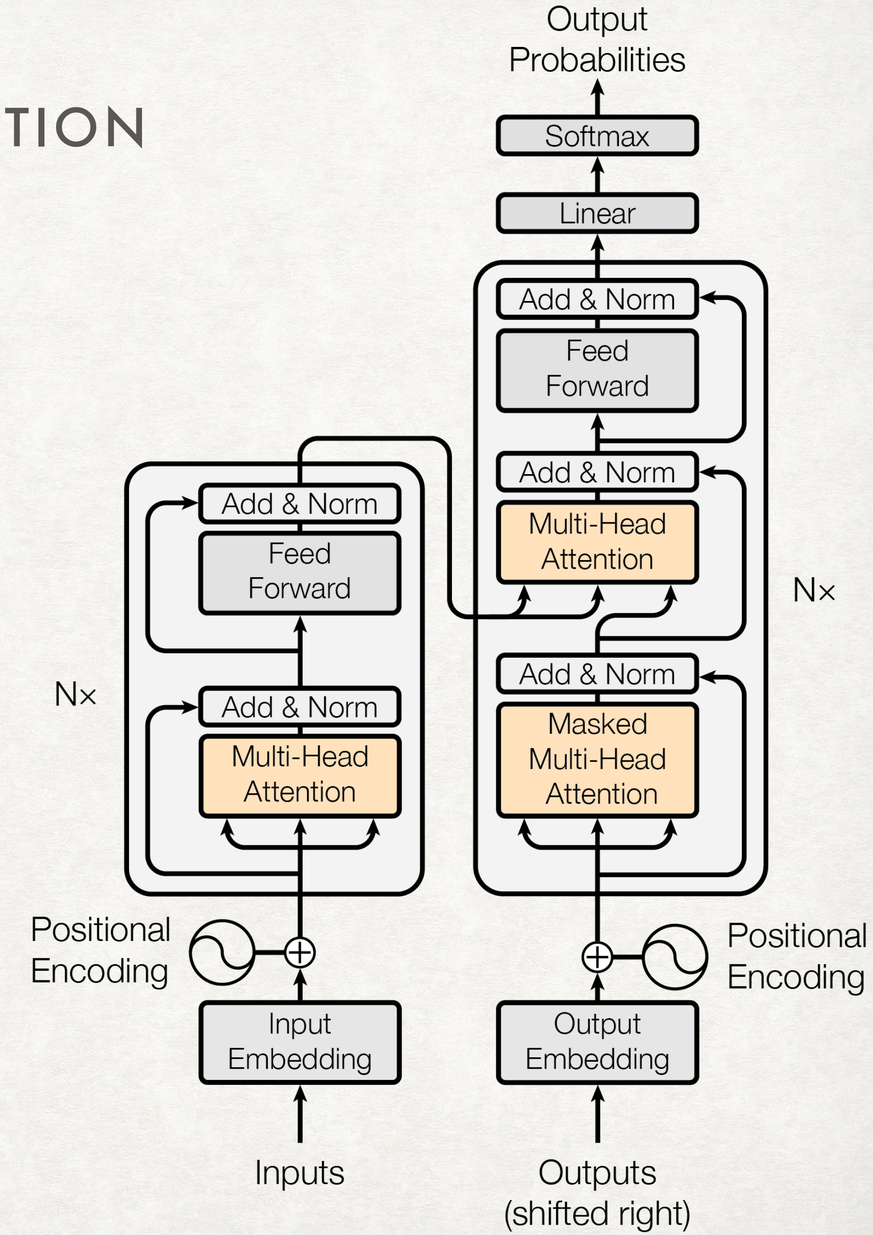
$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

My attempt at an English translation:

- For each of the vectors in Q, the query matrix, take a linear sum of the vectors in V, the value matrix.
- The amount to weigh each vector in V is dependent on how "similar" that vector is to the query vector.
- "Similar" is measured in terms of the dot product between the vectors.

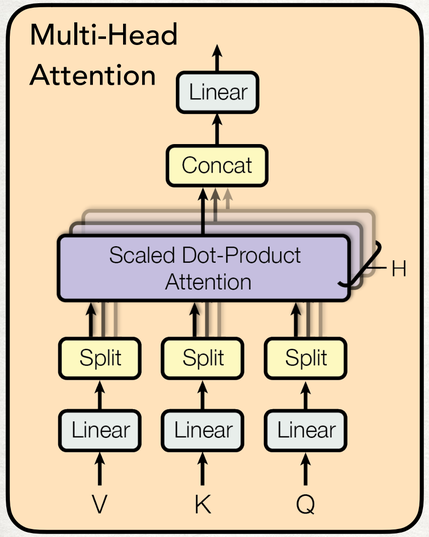
For encoder-decoder attention:
Keys and values come from encoder's final output.
Queries come from the previous decoder layer's outputs.

For self-attention:
Keys, queries, and values all come from the outputs of the previous layer.



TRANSFORMERS

MULTI-HEAD ATTENTION



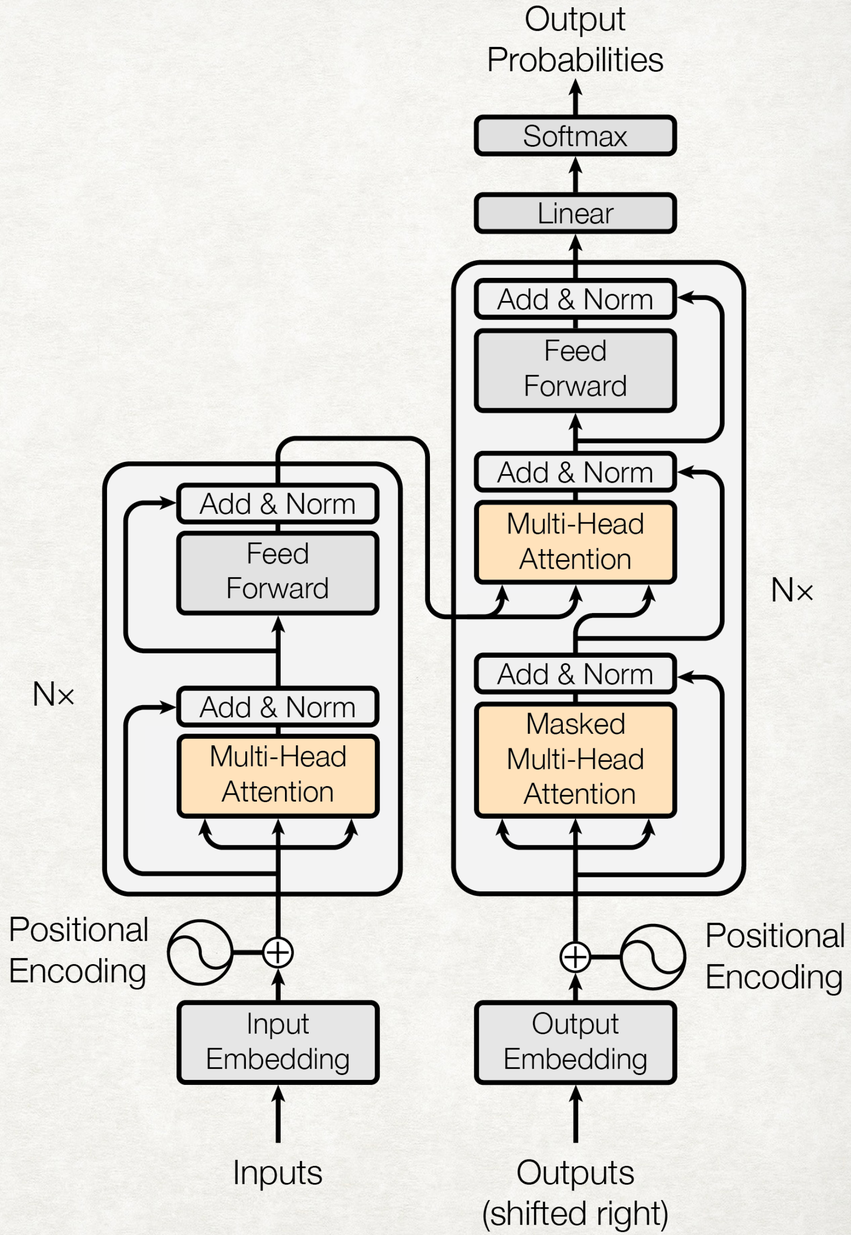
$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

$$\text{MultiHeadAtt}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}^O$$

where $\text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^{\mathbf{Q}}, \mathbf{K}\mathbf{W}_i^{\mathbf{K}}, \mathbf{V}\mathbf{W}_i^{\mathbf{V}})$

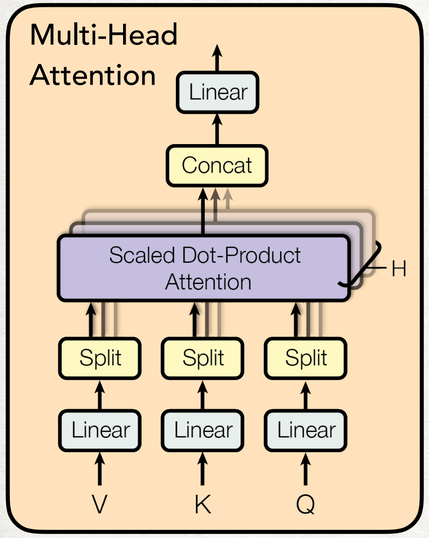
Instead of operating on $\mathbf{Q}, \mathbf{K}$, and $\mathbf{V}$ directly, the mechanism projects each input into a smaller dimension. This is done h times. The attention operation is performed on each of these "heads," and the results are concatenated.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions.

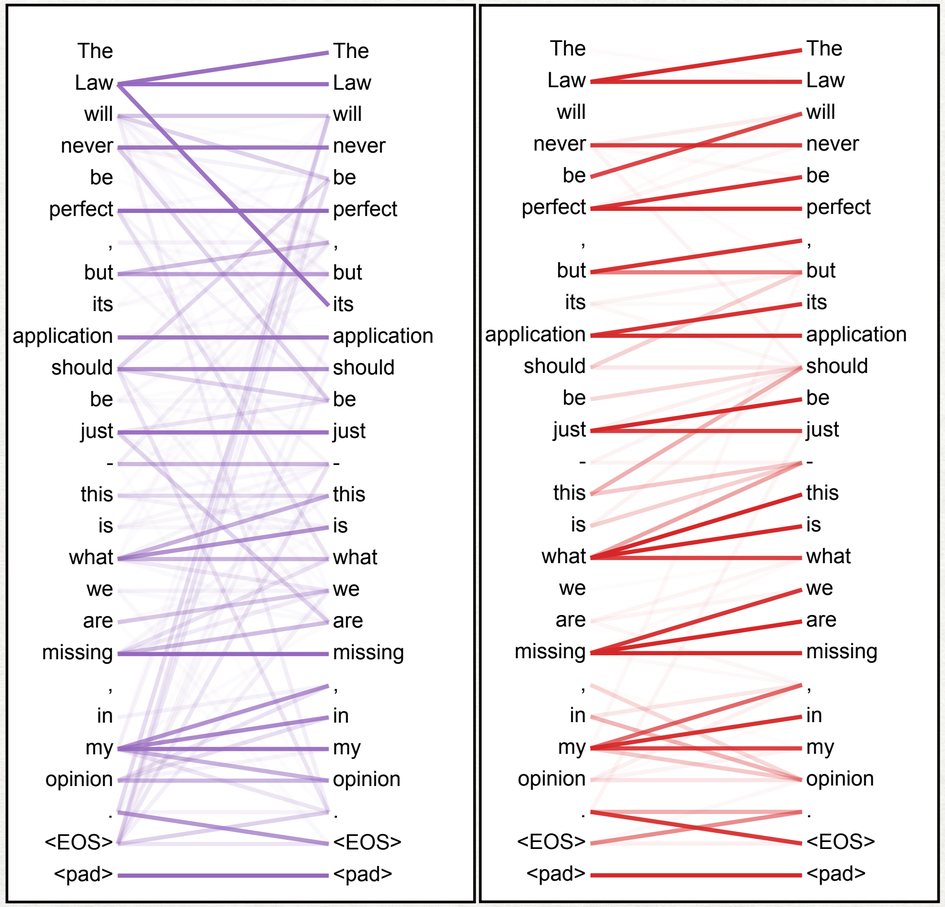


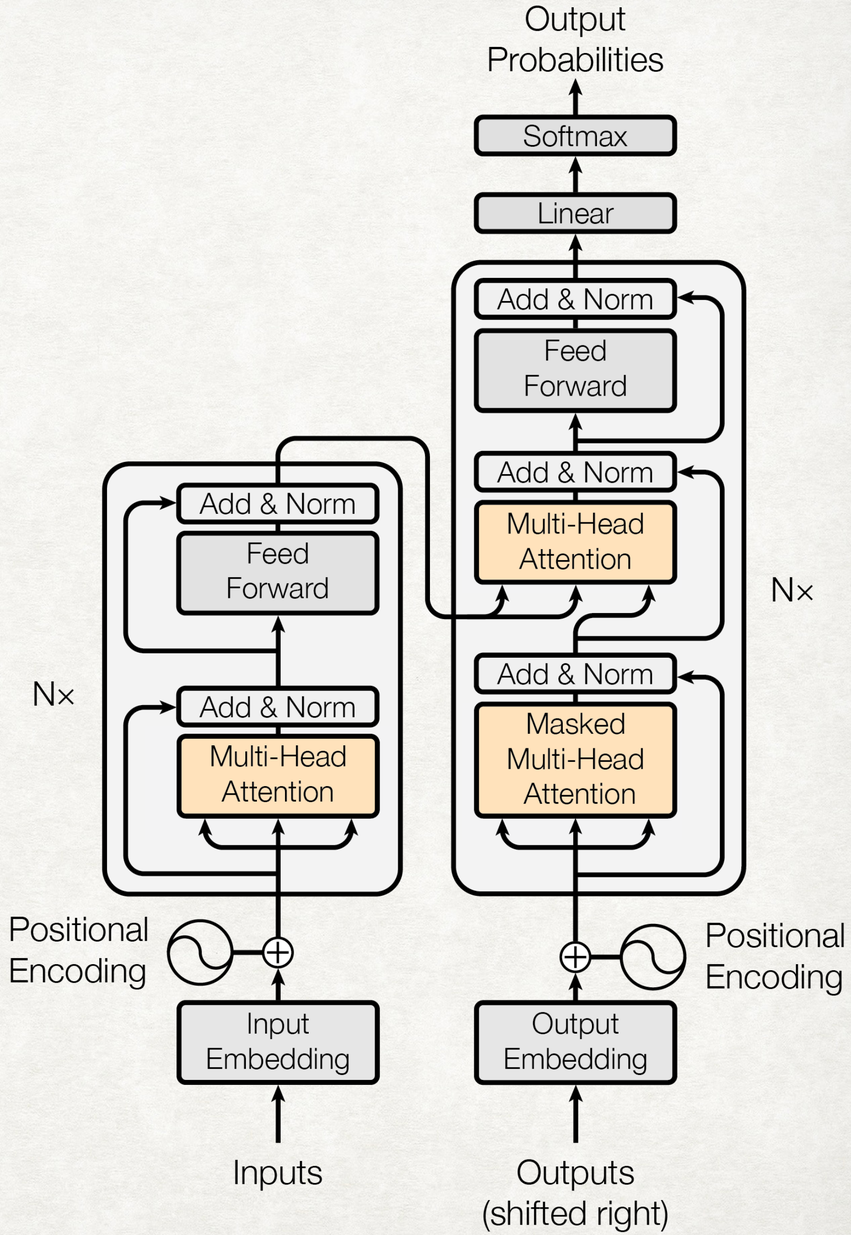
TRANSFORMERS

MULTI-HEAD ATTENTION



Two different self-attention heads:

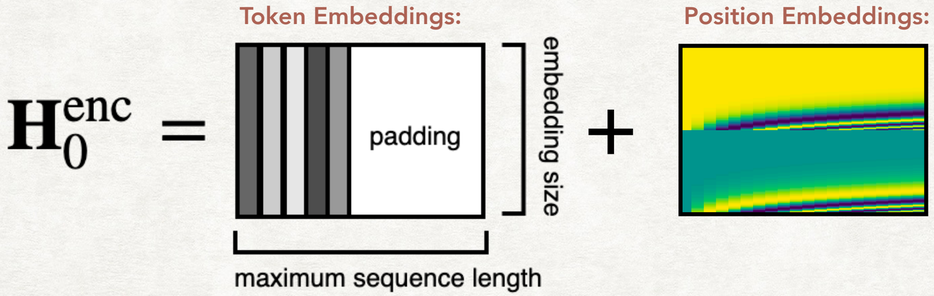




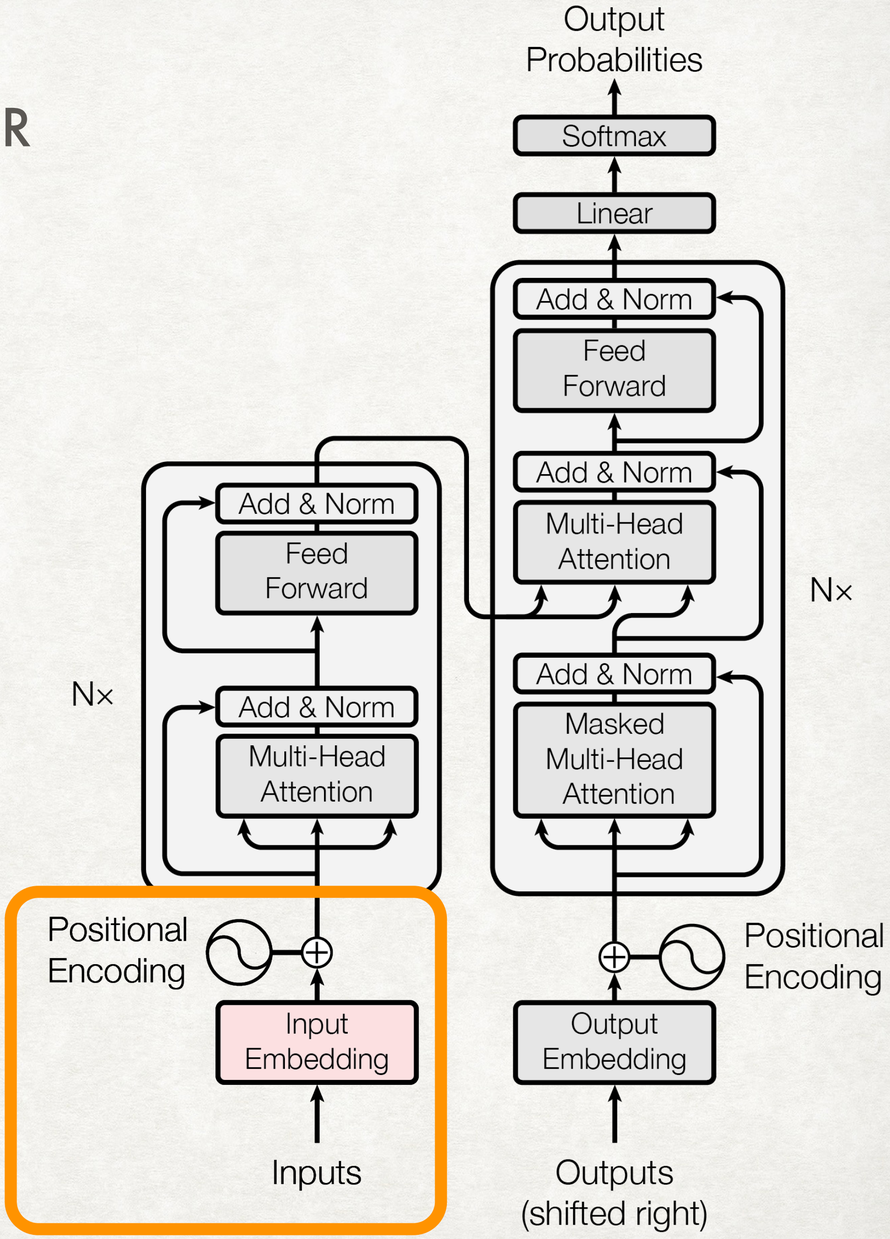
TRANSFORMERS

INPUTS TO THE ENCODER

The input into the encoder looks like:



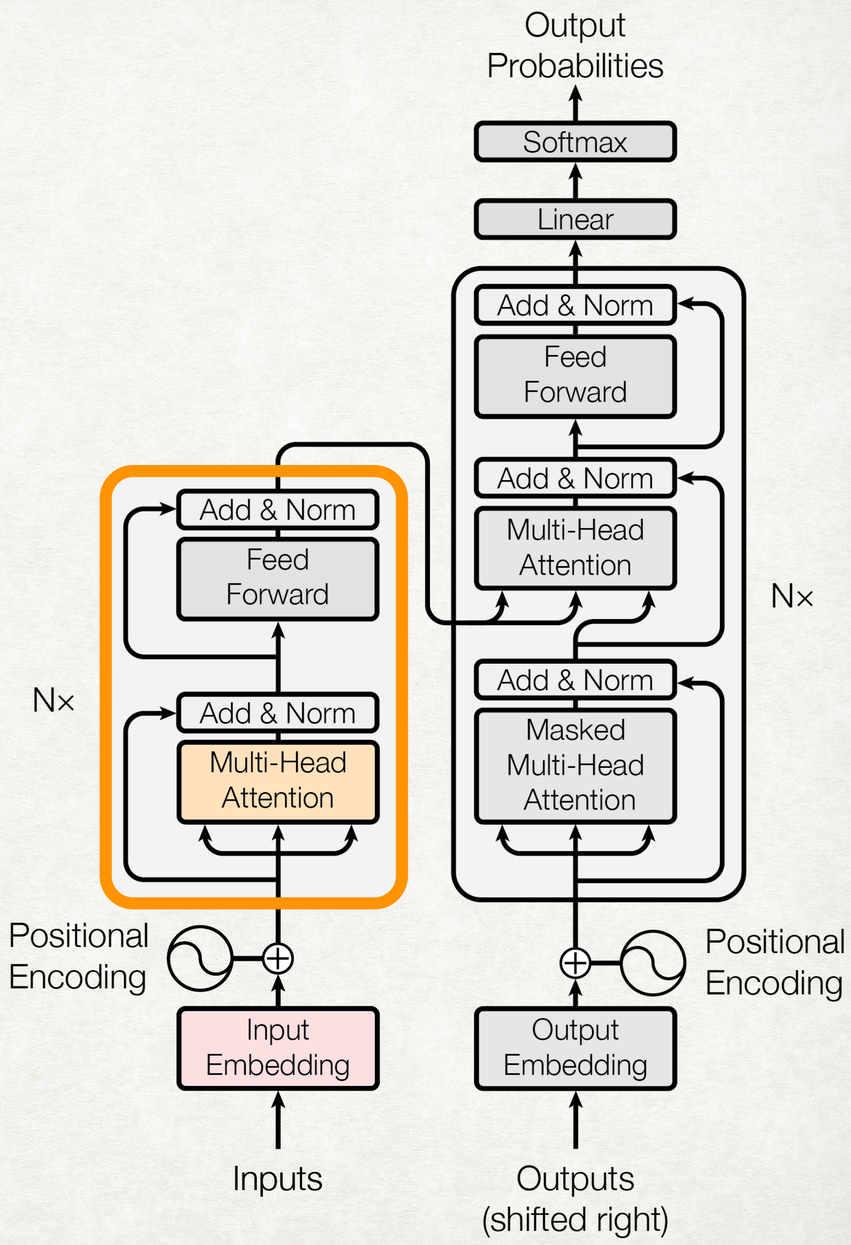
= token embeddings + position embeddings



TRANSFORMERS

THE ENCODER

Multi-Head Attention $= \text{MultiHeadAtt}(\mathbf{H}_i^{\text{enc}}, \mathbf{H}_i^{\text{enc}}, \mathbf{H}_i^{\text{enc}})$

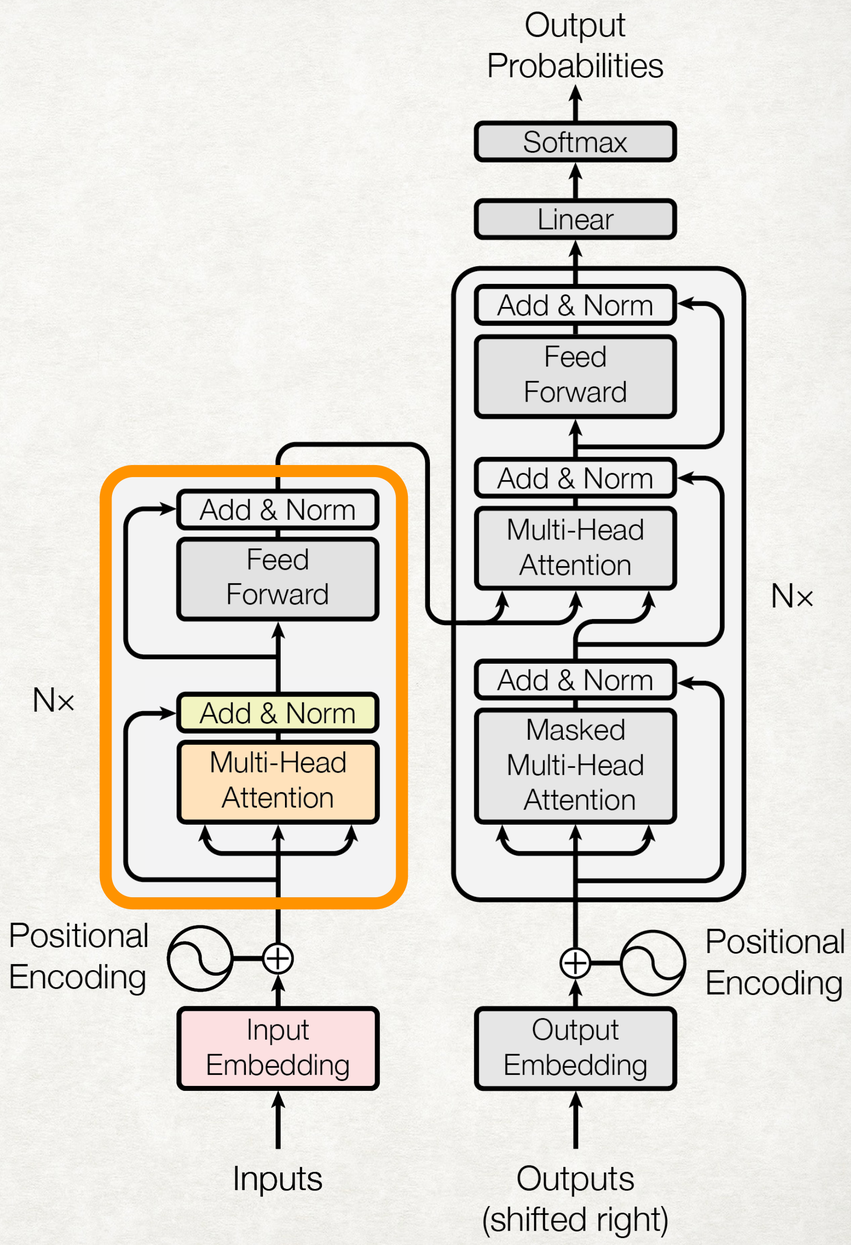


TRANSFORMERS

THE ENCODER

Multi-Head Attention $= \text{MultiHeadAtt}(\mathbf{H}_i^{\text{enc}}, \mathbf{H}_i^{\text{enc}}, \mathbf{H}_i^{\text{enc}})$

Add & Norm $= \text{LayerNorm}(\text{Multi-Head Attention} + \mathbf{H}_i^{\text{enc}})$



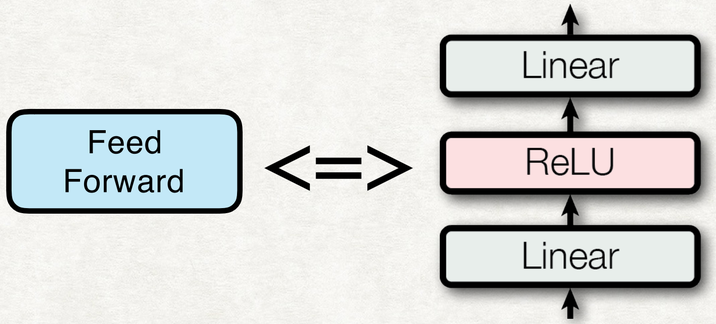
TRANSFORMERS

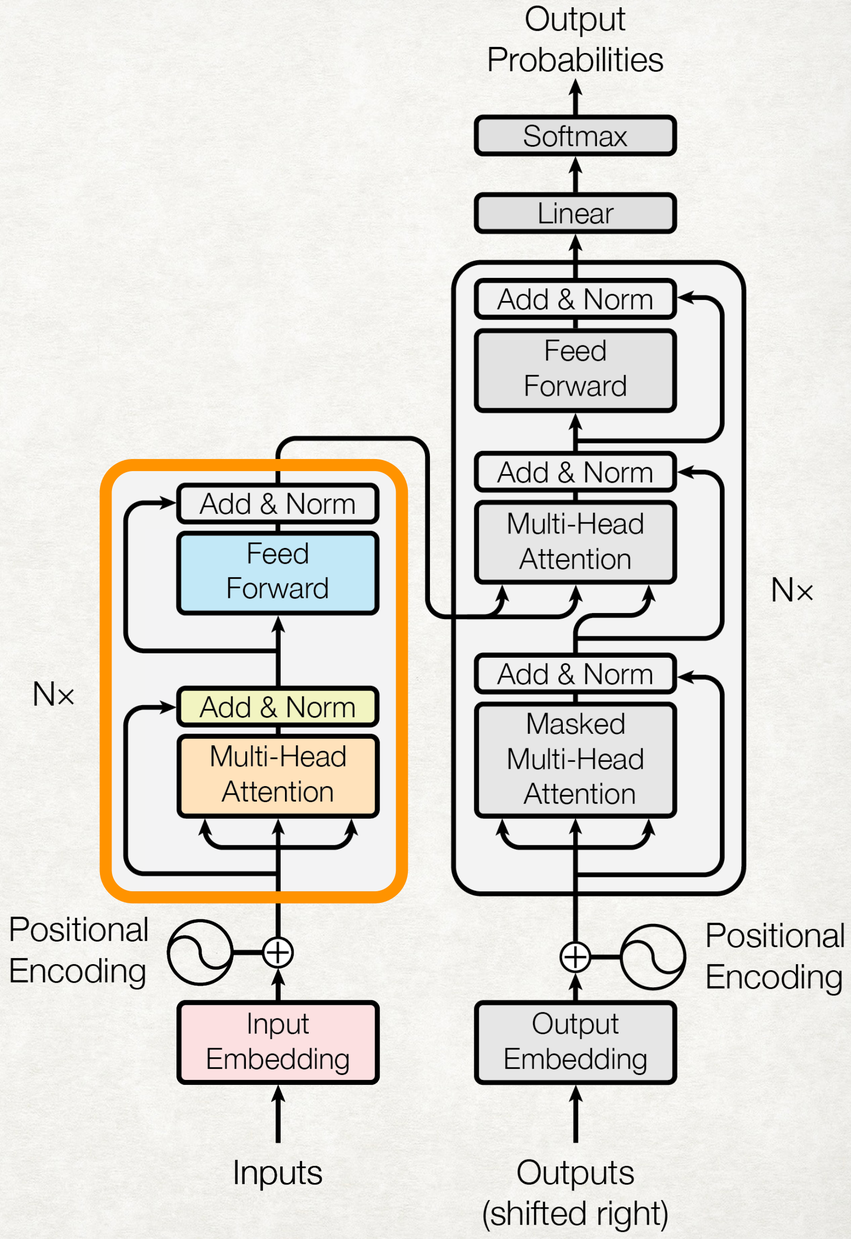
THE ENCODER

Multi-Head Attention $= \text{MultiHeadAtt}(\mathbf{H}_i^{\text{enc}}, \mathbf{H}_i^{\text{enc}}, \mathbf{H}_i^{\text{enc}})$

Add & Norm $= \text{LayerNorm}(\text{Multi-Head Attention} + \mathbf{H}_i^{\text{enc}})$

Feed Forward $= \max(0, \text{Add \& Norm}\,\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2$





TRANSFORMERS

THE ENCODER

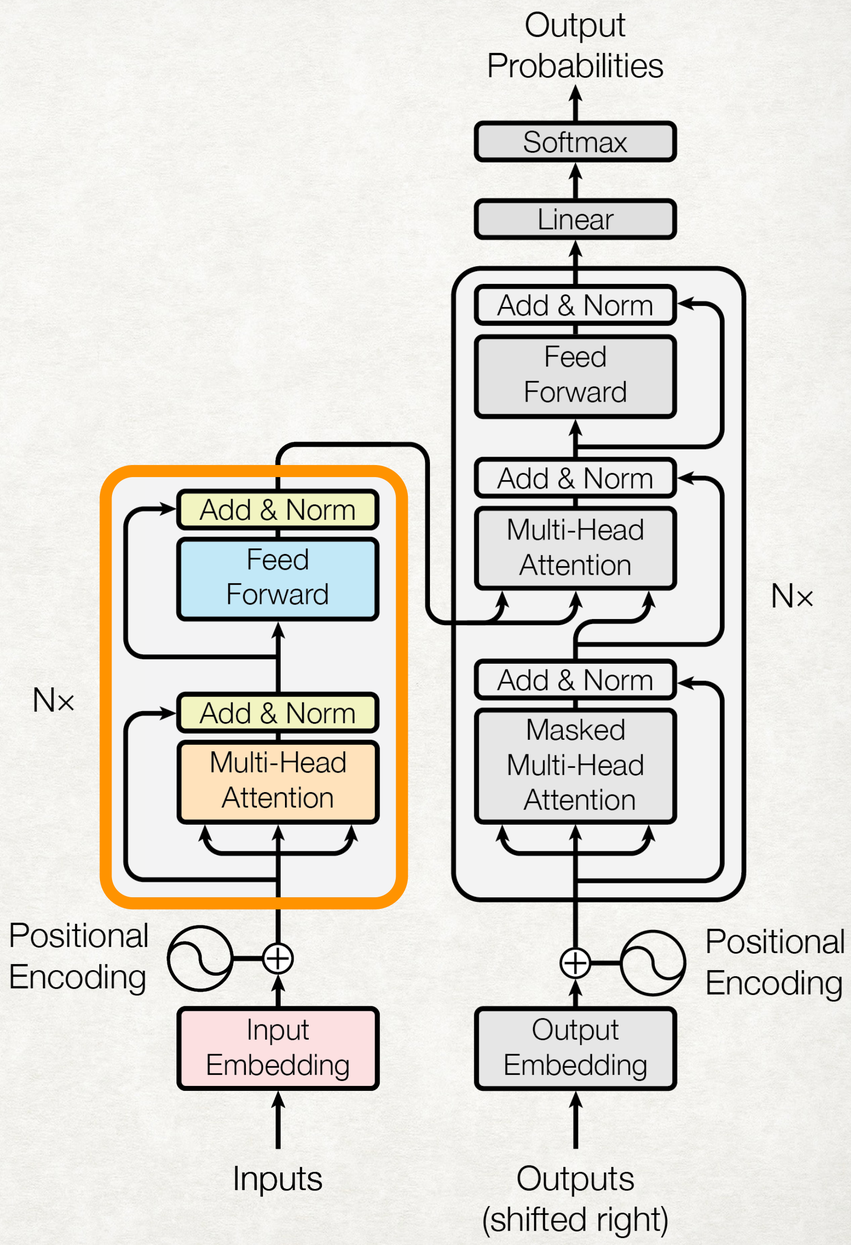
$$\boxed{\text{Multi-Head Attention}} = \text{MultiHeadAtt}(\mathbf{H}_i^{\text{enc}}, \mathbf{H}_i^{\text{enc}}, \mathbf{H}_i^{\text{enc}})$$

$$\boxed{\text{Add \& Norm}} = \text{LayerNorm}(\boxed{\text{Multi-Head Attention}} + \mathbf{H}_i^{\text{enc}})$$

$$\boxed{\text{Feed Forward}} = \max(0, \boxed{\text{Add \& Norm}}\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2$$

$$\boxed{\text{Add \& Norm (2)}} = \text{LayerNorm}(\boxed{\text{Feed Forward}} + \mathbf{H}_i^{\text{enc}})$$

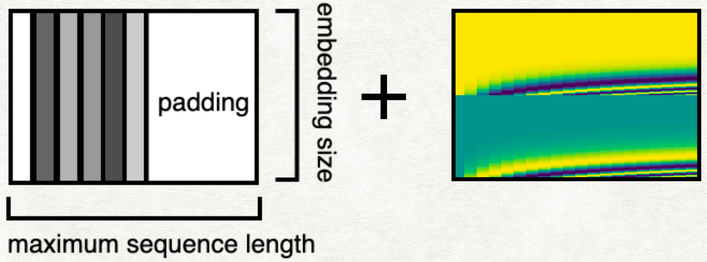
$$\mathbf{H}_{i+1}^{\text{enc}} = \boxed{\text{Add \& Norm (2)}}$$



TRANSFORMERS

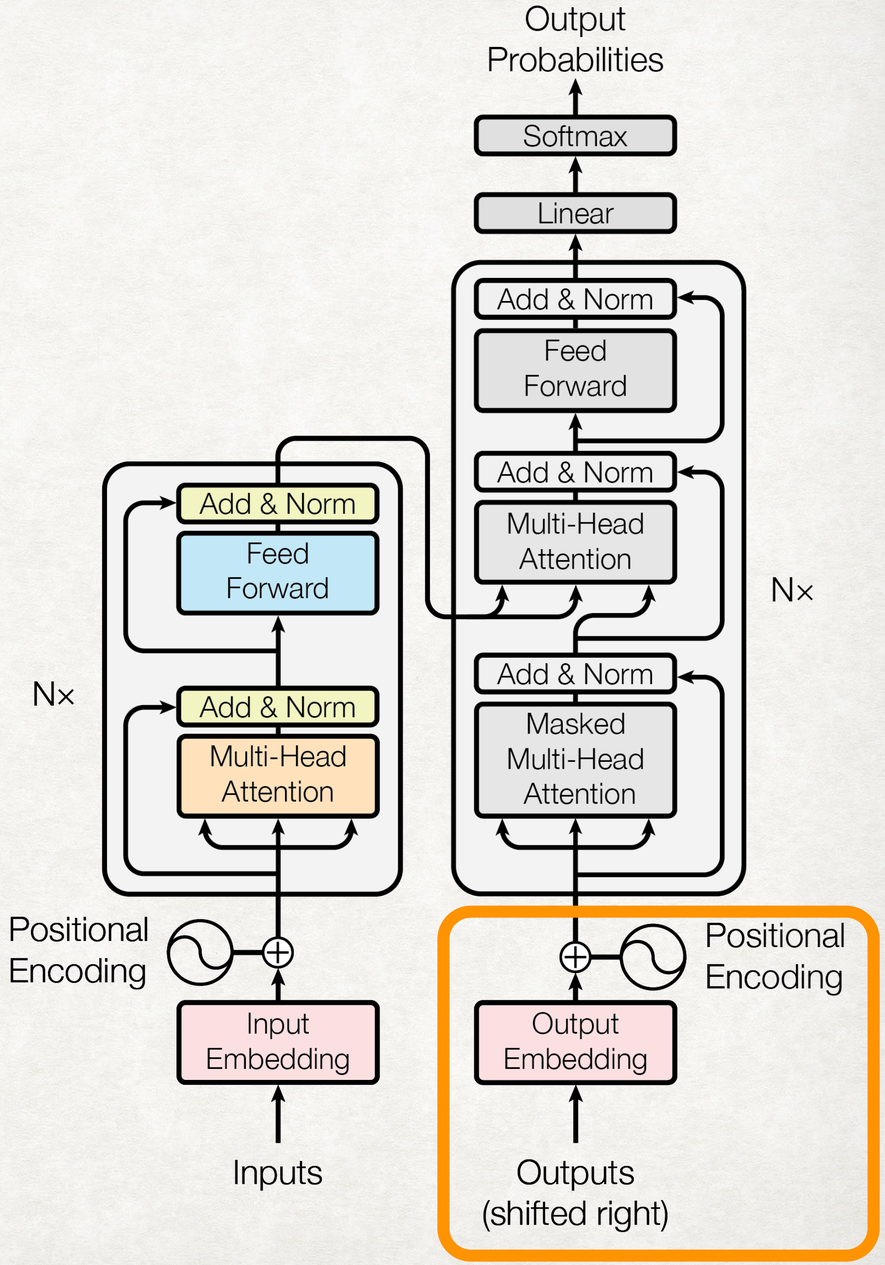
THE DECODER

$$\mathbf{H}_0^{\text{dec}} =$$



$+$

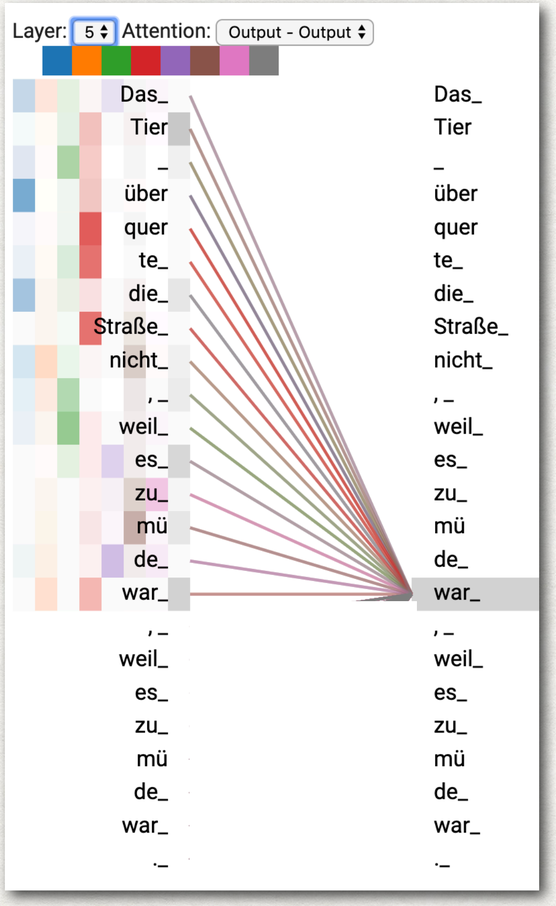
= token embeddings + position embeddings

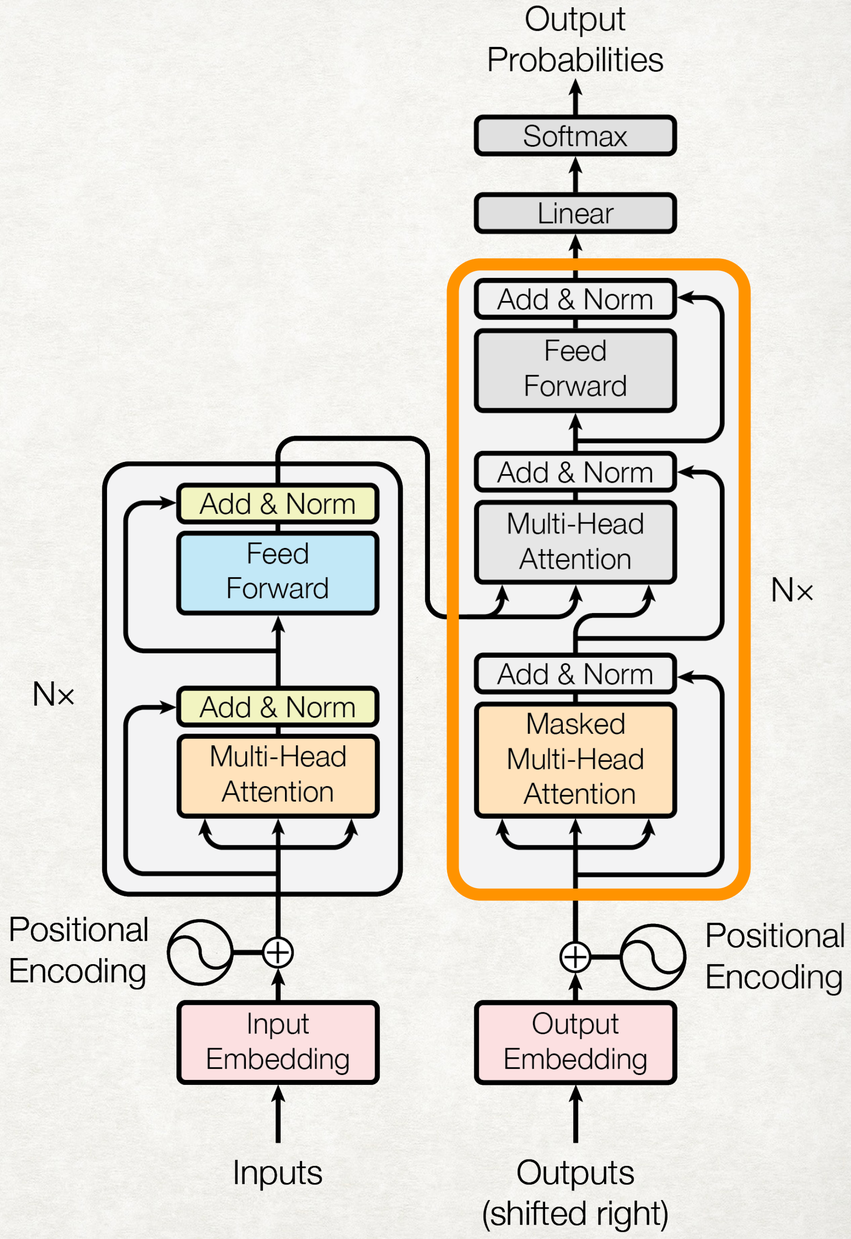


TRANSFORMERS

THE DECODER

Masked Multi-Head Attention $= \text{MaskedMultiHeadAtt}(\mathbf{H}_i^{\text{dec}}, \mathbf{H}_i^{\text{dec}}, \mathbf{H}_i^{\text{dec}})$



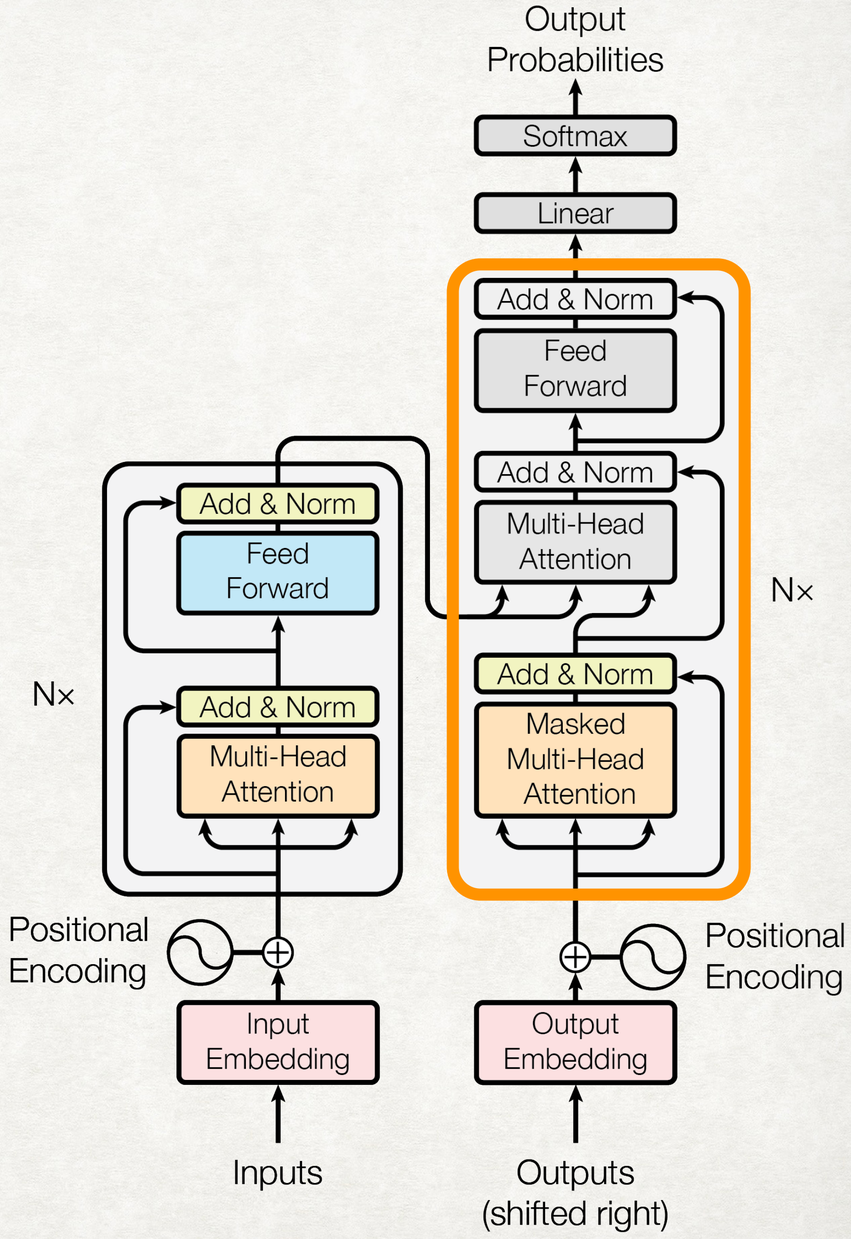


TRANSFORMERS

THE DECODER

Masked Multi-Head Attention $= \text{MaskedMultiHeadAtt}(\mathbf{H}_i^{\text{dec}}, \mathbf{H}_i^{\text{dec}}, \mathbf{H}_i^{\text{dec}})$

Add & Norm $= \text{LayerNorm}(\text{Multi-Head Attention} + \mathbf{H}_i^{\text{dec}})$



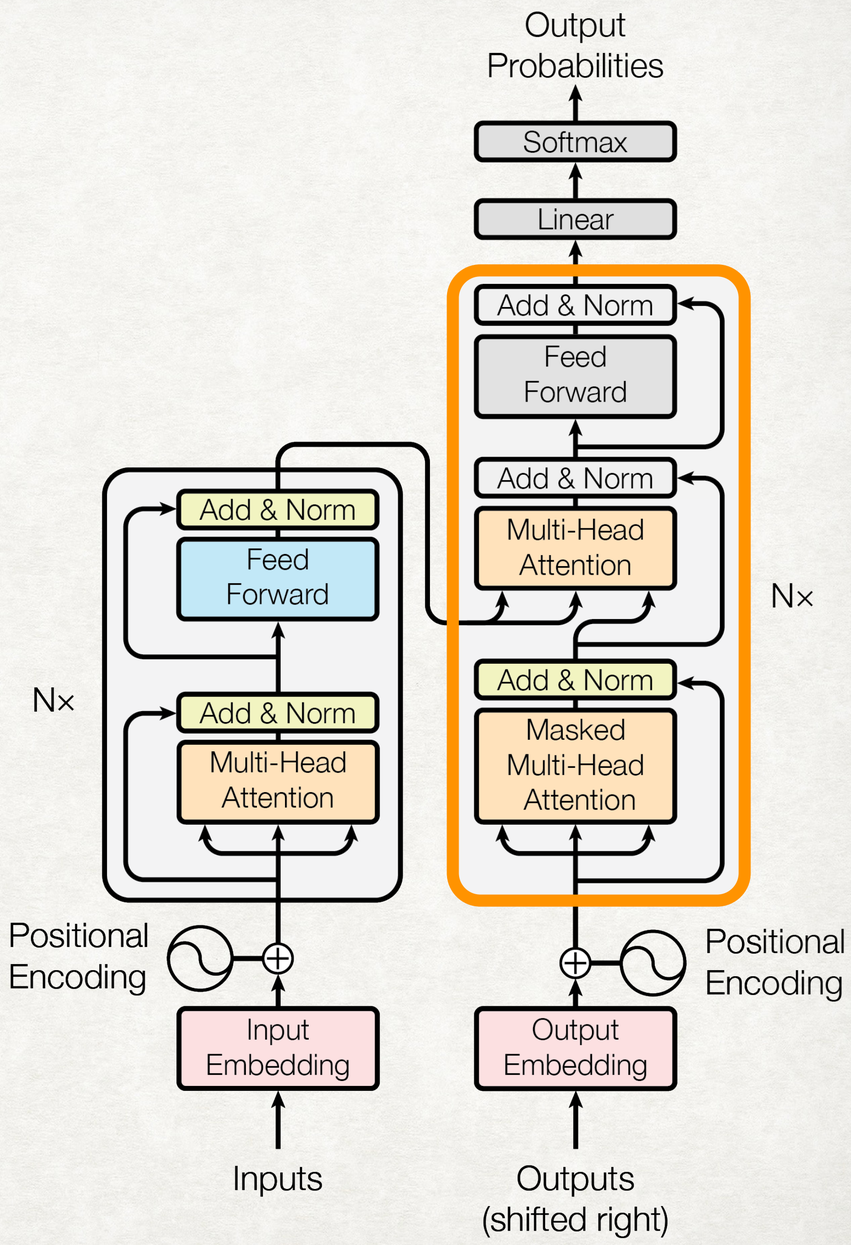
TRANSFORMERS

THE DECODER

Masked Multi-Head Attention $= \text{MaskedMultiHeadAtt}(\mathbf{H}_i^{\text{dec}}, \mathbf{H}_i^{\text{dec}}, \mathbf{H}_i^{\text{dec}})$

Add & Norm $= \text{LayerNorm}(\boxed{\text{Multi-Head Attention}} + \mathbf{H}_i^{\text{dec}})$

Enc-Dec Multi-Head Attention $= \text{MultiHeadAtt}(\mathbf{H}_i^{\text{enc}}, \mathbf{H}_i^{\text{dec}}, \mathbf{H}_i^{\text{dec}})$



TRANSFORMERS

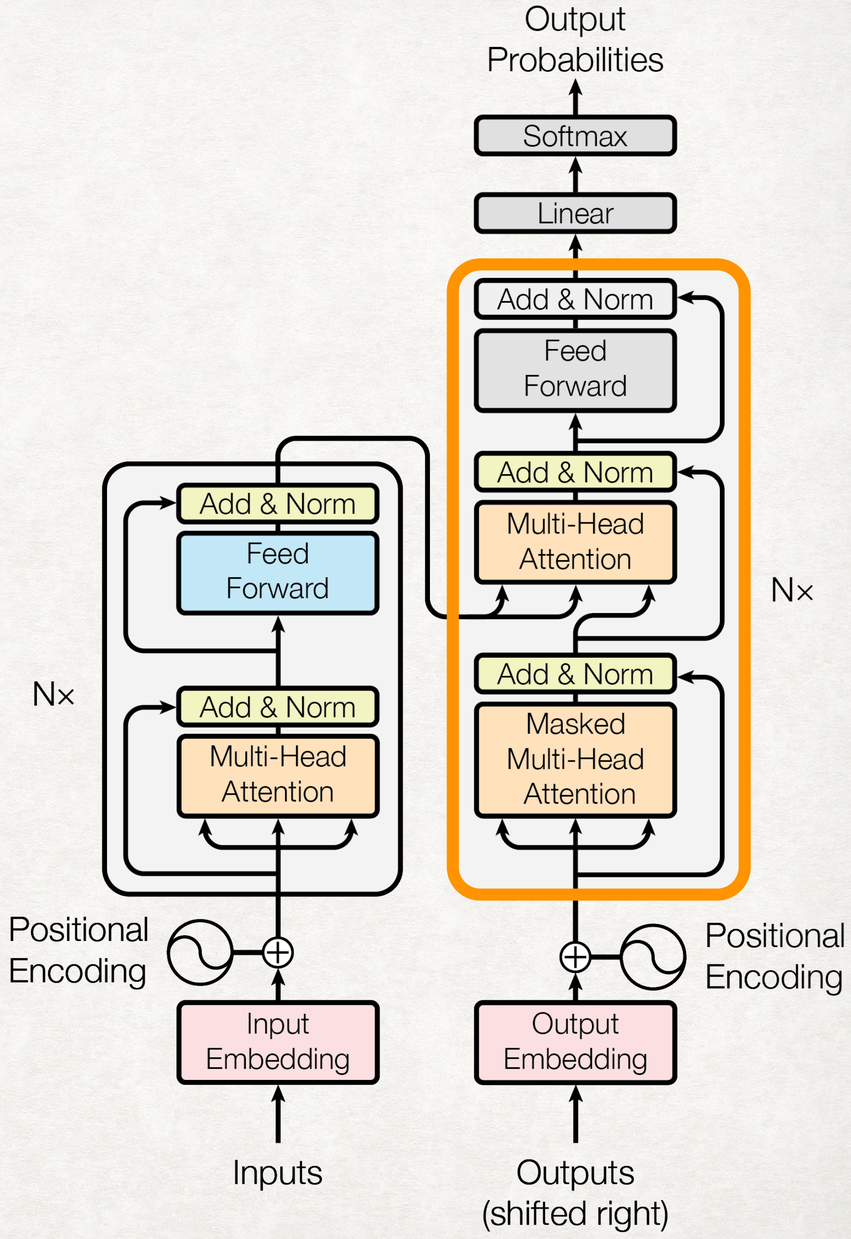
THE DECODER

Masked Multi-Head Attention $= \text{MaskedMultiHeadAtt}(\mathbf{H}_i^{\text{dec}}, \mathbf{H}_i^{\text{dec}}, \mathbf{H}_i^{\text{dec}})$

Add & Norm $= \text{LayerNorm}(\text{Multi-Head Attention} + \mathbf{H}_i^{\text{dec}})$

Enc-Dec Multi-Head Attention $= \text{MultiHeadAtt}(\mathbf{H}_i^{\text{dec}}, \mathbf{H}_i^{\text{enc}}, \mathbf{H}_i^{\text{enc}})$

Add & Norm (2) $= \text{LayerNorm}(\text{Multi-Head Attention} + \text{Add \& Norm})$



TRANSFORMERS

THE DECODER

Masked Multi-Head Attention $= \text{MaskedMultiHeadAtt}(\mathbf{H}_i^{\text{dec}}, \mathbf{H}_i^{\text{dec}}, \mathbf{H}_i^{\text{dec}})$

Add & Norm $= \text{LayerNorm}($ Multi-Head Attention $+ \mathbf{H}_i^{\text{dec}})$

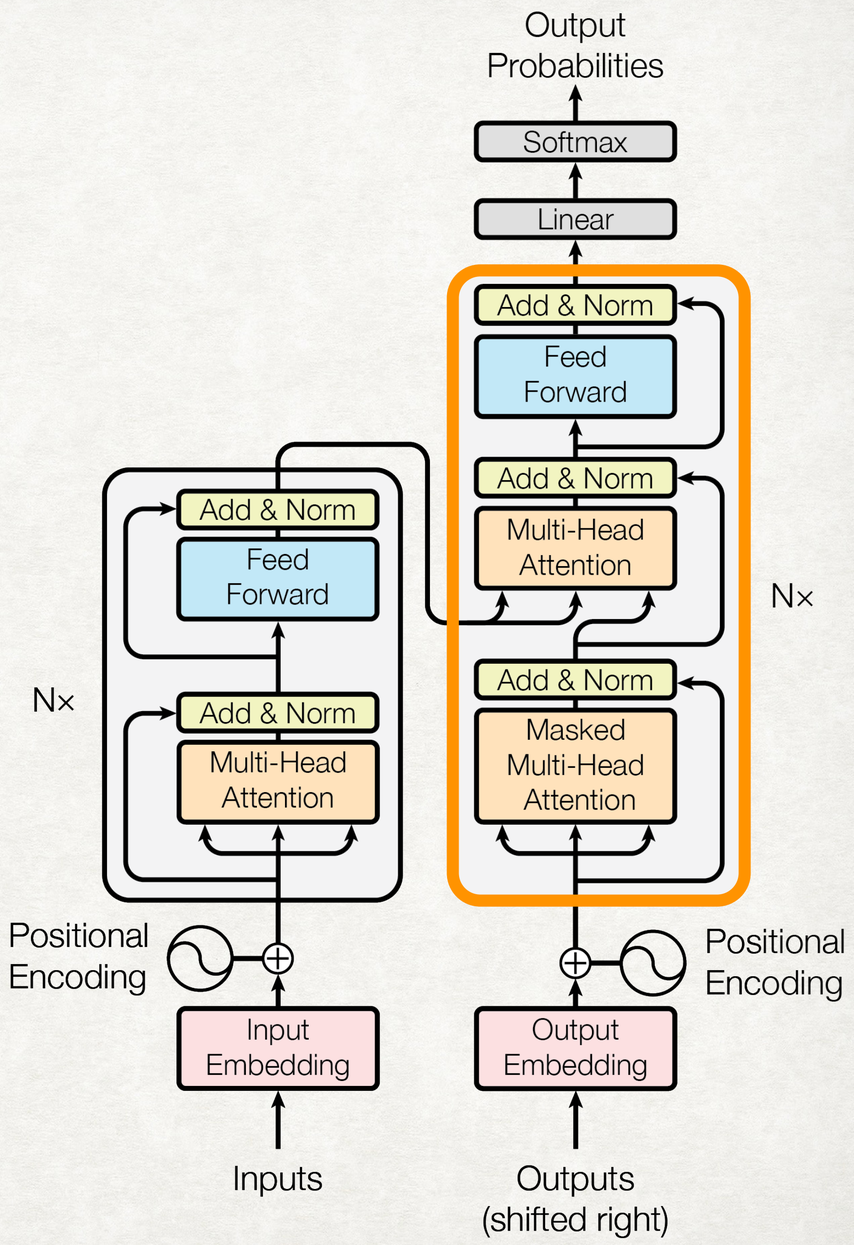
Enc-Dec Multi-Head Attention $= \text{MultiHeadAtt}(\mathbf{H}_i^{\text{dec}}, \mathbf{H}_i^{\text{enc}}, \mathbf{H}_i^{\text{enc}})$

Add & Norm (2) $= \text{LayerNorm}($ Multi-Head Attention $+$ Add & Norm $)$

Feed Forward $= \max(0,$ Add & Norm (2) $\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2$

Add & Norm (3) $= \text{LayerNorm}($ Feed Forward $+$ Add & Norm (2) $)$

$\mathbf{H}_{i+1}^{\text{dec}} =$ Add & Norm (3)



TRANSFORMERS
GENERATED TEXT CIRCA 2018

'''The Transformer''' are a Japanese [[hardcore punk]] band.

==Early years==
The band was formed in 1968, during the height of Japanese music history. Among the legendary [[Japanese people|Japanese]] composers of [Japanese lyrics], they prominently exemplified Motohiro Oda's especially tasty lyrics and psychedelic intention. Michio was a longtime member of the every Sunday night band PSM. His alluring was of such importance as being the man who ignored the already successful image and that he municipal makeup whose parents were – the band was called Jenei.<ref>http://www.separatist.org/se_frontend/post-punk-musician-the-kidney.html</ref> From a young age the band was very close, thus opting to pioneer what had actually begun as a more manageable core hardcore punk band.<ref>http://www.talkradio.net/article/independent-music-fades-from-the-closed-drawings-out</ref>

==History==
===Born from the heavy metal revolution===
In 1977 the self-proclaimed King of Tesponsors, [[Joe Lus:

: It was somewhere... it was just a guile ... taking this song to Broadway. It was the first record I ever heard on A.M., After some opposition I received at the hands of Parsons, and in the follow-up notes myself.<ref>http://www.discogs.com/artist/The+Op%C5%8Dn+&+Psalm</ref>

The band cut their first record album titled "Transformed, furthered and extended Extended",<ref>[https://www.discogs.com/album/69771 MC – Transformed EP (CDR) by The Moondrawn – EMI, 1994]</ref> and in 1978 the official band line-up of the three-piece pop-punk-rock band TEEM. They generally played around [[Japan]], growing from the Top 40 standard.

===1981-2010: The band to break away===
On 1 January 1981 bassist Michio Kono, and the members of the original line-up emerged. Niji Fukune and his [[Head poet|Head]] band (now guitarist) Kazuya Kouda left the band in the hands of the band at the May 28, 1981, benefit season of [[Led Zeppelin]]'s Marmarin building. In June 1987, Kono joined the band as a full-time drummer, playing a few nights in a 4 or 5 hour stint with [[D-beat]]. Kono played through the mid-1950s, at Shinlie, continued to play concerts with drummers in Ibis, Cor, and a few at the Leo Somu Studio in Japan. In 1987, Kono recruited new bassist Michio Kono and drummer Ayaka Kurobe as drummer for band. Kono played trumpet with supplement music with Saint Etienne as a drummer. Over the next few years Kono played as drummer and would get many alumni news invitations to the bands' "Toys Beach" section. In 1999 he joined the [[CT-182]].

His successor was Barrie Bell on a cover of [[Jethro Tull (band)|Jethro Tull]]'s original 1967 hit "Back Home" (last appearance was in Jethro), with whom he shares a name.

===2010 – present: The band to split===
In 2006 the band split up and the remaining members reformed under the name Starmirror, with Kono in tears, Kurobe, and Kurobe all playing harmonica with Kooky Bell and a new guitarist again. While Jaari also had the realist DJ experience, The SkykelDaten asked New Bantherhine, who liked Kono and Kurobe, to join him on guitar. Kono is now playing in the studio a new formation, and at their 11th anniversary concert, made a wide variety of music and DJ equipment including two new vocalists: Dejeh Faida and Janis. NARCO (Inc.) privileges areas sections until 2012. and in 2015 both were members as members as were Dicent and Cautty.

In 2014, the album "[[Marco Victoriano in Focus]]" was released, and entered the Japanese albums chart at number 69 in the [[Oricon Singles Chart]].<ref name="oricon">{{cite web|title=THE GER: THE TALENT RAILWAY LIVERSITIES|url=http://www.oricon.co.jp/prof/inductee/30565/ranking/cd/1/|work=OriconStyle|accessdate=11 Jul 2014}}</ref> The album was also in Japan, where [[Hoite (musician)|Hoite]] recorded and released an album in October 2011, and a short album, titled "[[Grateful]]" in 2012.<ref>{{cite web|url=http://www.oricon.co.jp/prof/artist/229337/ranking/cd/1/|title=HORIZON HISTORY|publisher=oricon.co.jp|accessdate=9 Nov 2014|language=ja}}</ref> Kono played the [[N9ne]] bass with Tony "Shadows Without a Face", and released his music from the new [[Smile (record label)|Smile]] label with original Fat Joe Lang "Remix and Bachian" cassette.

==Style==
The band's style has been compared to [[Radar|radar-based]], influenced by bands such as [[Metallica]], [[Damage (Japanese band)|Damage]], [[Dreadzone]], and [[Girlschool]].

The group classifies itself as "the first band to play a Used Of American Inside the Outer East",<ref>http://www.funonline.jp/i/main.php?conID=005&artID=12548</ref> including the band's usual Eugene Terre ensemble. He later stated that the raw interviews in Metal Hammer and Metallica gave reason to what they considered that an explosion of the live band started by the band.<ref>http://stillenterprise.com/en/interviews/last-night-reunion-confronts-kooky-spearing-the-charismatic-dynamic-partnership/</ref>

[[Hunt's Brigade]], a surf-rock band from Tokyo, has cited the band's music as being "straight ahead of their time / I did but only read five songs (played now), a beat, rock blast, a bit of a bass blast, good lyrics, and a few".<ref>http://stillenterprise.com/en/announcements/Package-a2/Swag-w-Vause-042332/</ref> Halutodrinking, a popular-sounding drum style, has described the band as being "supporting [mod]ed distrust", because it incorporated the track as an ensemble and did not fit into any of the more darling songs from their previous incarnation, which was forging a strong style to a lot of different

Credit: Generating Wikipedia by Summarizing Long Sequences <https://arxiv.org/abs/1801.10198>

OUTLINE

- Text generation before neural networks
- Neural language model framework
- LM Architectures
  - Recurrent neural networks
  - Convolutional neural networks
  - Transformers
- **Takeaways**

TAKEAWAYS
WHICH IS BEST?

- Both CNNs and Transformers far outperform RNNs empirically. This shows that:
  - Infinite context length is not necessary.
  - Non-recurrent architectures are more effective at modeling language

TAKEAWAYS
WHICH IS BEST?

- Both CNNs and Transformers far outperform RNNs empirically. This shows that:
  - Infinite context length is not necessary.
  - Non-recurrent architectures are more effective at modeling language
- Attention-based models are better than convolutional models when the sequence length is less than the embedding dimension.

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

TAKEAWAYS
WHICH IS BEST?

- Both CNNs and Transformers far outperform RNNs empirically. This shows that:
  - Infinite context length is not necessary.
  - Non-recurrent architectures are more effective at modeling language
- Self-attention models are better than convolutional convolutional models when the sequence length is less than the embedding dimension.
- Transformers have largely “won” the architecture battle.
  - Those working on tasks with smaller datasets or who have limited computation resources still frequently use LSTMs, but all of the state-of-the-art text generation models these days are based on the Transformer architecture.
  - Transformer paper has 6000+ citations while CNN paper has ~1100 (though they came out approximately concurrently).

TAKEAWAYS

EXTENSIONS TO TRANSFORMER ARCHITECTURE

- Relative attention enables Transformer to generate longer sequences than it was trained on.
  - "Self-Attention with Relative Position Representations" <https://arxiv.org/abs/1803.02155>
- Massive multi-GPU parallelization allows training giant language models (Microsoft just released one with 17 billion parameters).
  - https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/
- Distillation allows smaller models to be formed from bigger ones.
  - "Distilling Transformers into Simple Neural Networks with Unlabeled Transfer Data" <https://arxiv.org/abs/1910.01769>
- Lots of attempts to make sparse attention mechanisms work.
  - "Efficient Content-Based Sparse Attention with Routing Transformers" <https://openreview.net/forum?id=B1gjs6EtDr>
  - "Reformer: The Efficient Transformer" <https://arxiv.org/abs/2001.04451>

THANKS FOR LISTENING