Tuesday • October 26, 2021

The Transformer: Attention-Only Networks

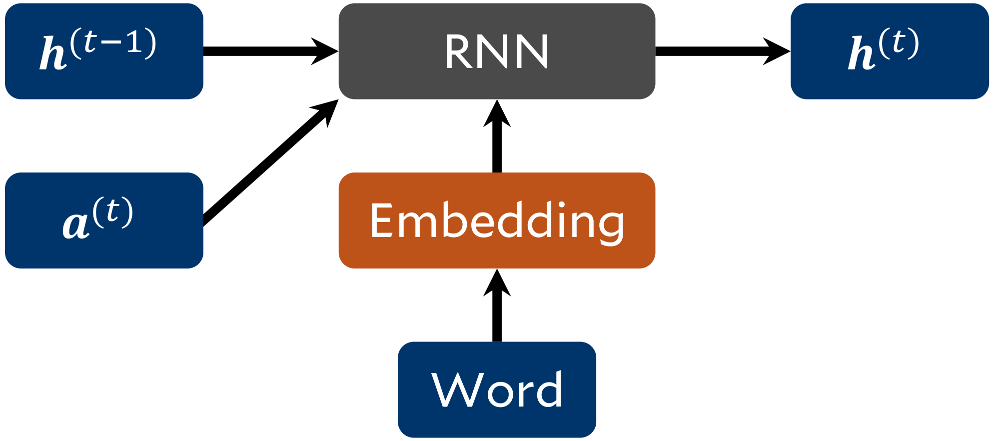
Yale

LING 380/780

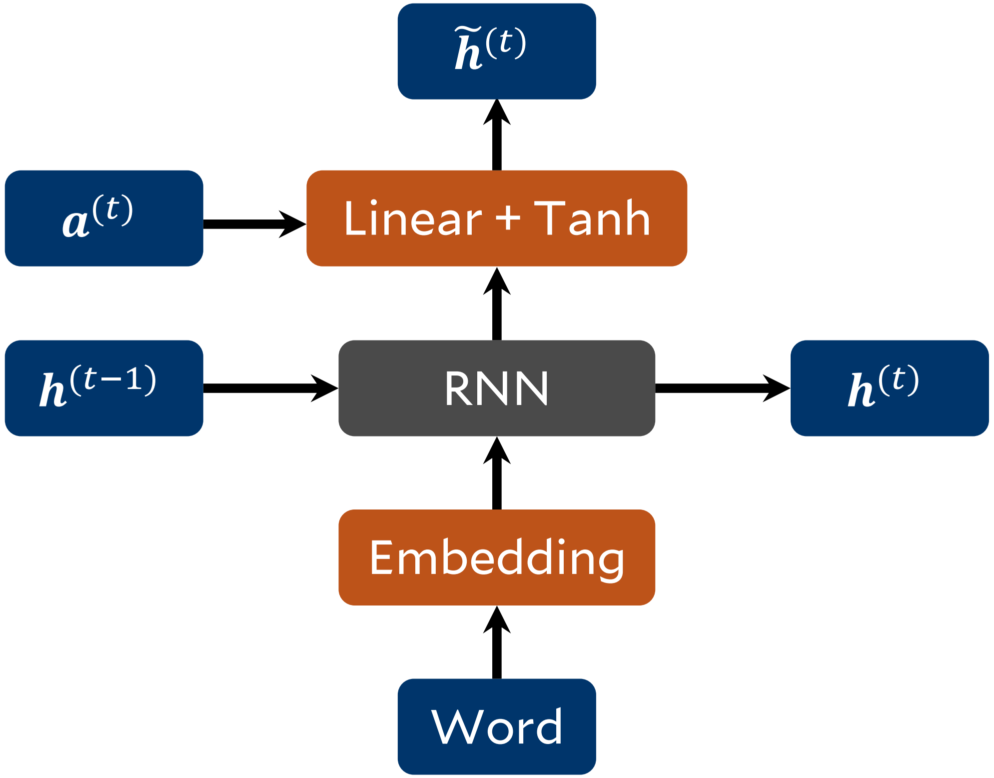
Neural Network Models of Linguistic Structure

RNN Decoder with Attention

Bahdanau-Style:



Luong-Style:



Computing Attention

Query Key

First, compute **attention scores**:

$$s_i^{(t)} = \text{score}(\boldsymbol{h}^{(d,T)}, \boldsymbol{h}^{(e,i)})$$

Then, convert the attention scores into **attention weights**:

$$\boldsymbol{\alpha}^{(t)} = \text{softmax}(\boldsymbol{s}^{(t)})$$

Finally, take a weighted average of encoder hidden states:

$$\boldsymbol{a}^{(t)} = \sum_i \alpha_i^{(t)} \boldsymbol{h}^{(e,i)}$$

Value

Computing Attention

First, compute **attention scores**:

$$s_i^{(t)} = \text{score}(\underset{\text{Query}}{\boldsymbol{q}^{(t)}}, \underset{\text{Key}}{\boldsymbol{k}^{(i)}})$$

Then, convert the attention scores into **attention weights**:

$$\boldsymbol{\alpha}^{(t)} = \text{softmax}(\boldsymbol{s}^{(t)})$$

Finally, take a weighted average of encoder hidden states:

$$\boldsymbol{a}^{(t)} = \sum_i \alpha_i^{(t)} \boldsymbol{v}^{(i)} \longleftarrow \text{Value}$$

Intuition behind Attention

- **Attention:** "looking up" previous information
- **Query** ($\boldsymbol{q}^{(t)}$): the thing that we want to look up
- **Keys** ($\boldsymbol{k}^{(i)}$): things that could be looked up
- **Values** ($\boldsymbol{v}^{(i)}$): the result of the lookup
- **Score** ($s_i^{(t)}$): How well does query $\boldsymbol{q}^{(t)}$ match key $\boldsymbol{k}^{(i)}$?

Computing Attention Scores

Computing attention scores

- Dot Product: $\mathrm{score}(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{q}^\top \boldsymbol{k}$
- Bilinear: $\mathrm{score}(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{q}^\top \boldsymbol{W} \boldsymbol{k}$
- MLP: $\mathrm{score}(\boldsymbol{q}, \boldsymbol{k}) = \mathrm{MLP}\left(\begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{k} \end{bmatrix}\right)$

Evaluating Machine Translation

- How do RNNs with attention compare to RNNs?
- What does it mean to translate well?

Evaluating Machine Translation

- **Example Sentence:** 猫はマットにいる。
- **Reference Translation:** The cat is on the mat.
- **Possible Machine Translations:**
  - The cat is on the mat.
  - The cats are on the mat.
  - The mat is where the cat is.
  - The the the the the the.

Evaluating Machine Translation

n-Gram Precision:

$$\frac{\text{\# of n}-\text{grams in the machine and reference translation}}{\text{\# of n}-\text{grams in the machine translation}}$$

Evaluating Machine Translation

- **Reference Translation:** The cat is on the mat.
- **Machine Translation:** The the mat.
- 1-gram precision: 100% (✔ The, ✔ the, ✔ mat, ✔ .)
- 2-gram precision: 67% (❌ The the, ✔ the mat, ✔ mat .)
- 3-gram precision: 50% (❌ The the mat, ✔ the mat .)
- 4-gram precision: 0% (❌ The the mat .)

Evaluating Machine Translation

Bilingual **E**valuation **U**nderstudy (BLEU) Score

$$\text{BLEU} = \text{length penalty} \cdot \left(\prod_{\text{n}=1}^{4} n\text{−gram precision} \right)^{1/4}$$

where

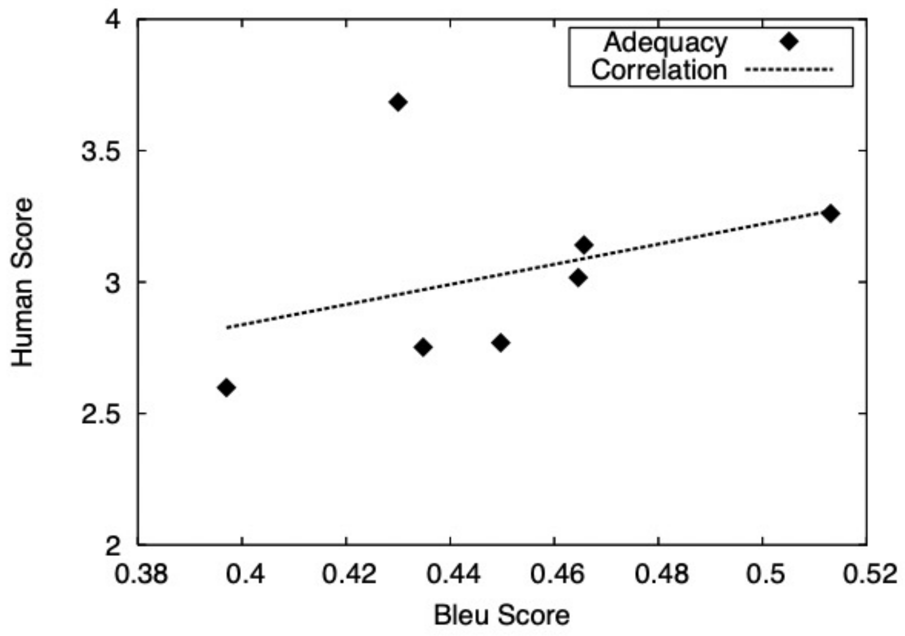
$$\text{length penalty} = \min(1, e^{1-\frac{\text{length of reference translation}}{\text{length of machine translation}}})$$

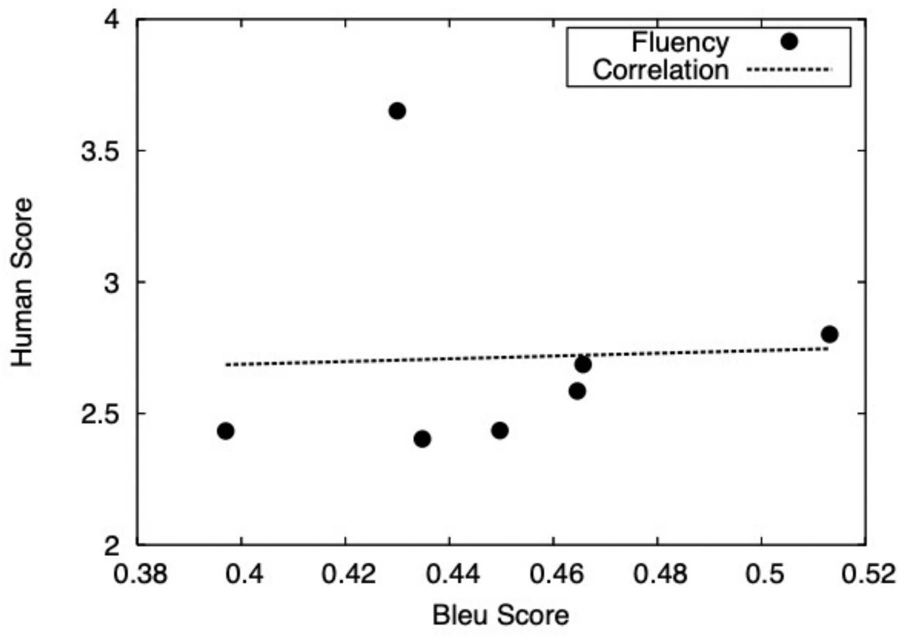
Evaluating Machine Translation

- **Example Sentence:** 猫はマットにある。
- **Reference Translation:** The cat is on the mat.
- **Possible Machine Translations:**
  - The cat is on the mat. **BLEU:** 100.0%
  - The cats are on the mat. **BLEU:** 43.5%
  - The mat is where the cat is. **BLEU:** 0.0%
  - The the the the the the. **BLEU:** 0.0%

Is BLEU Reasonable?

BLEU vs. Human Judgements (Callison-Burch et al., 2006)





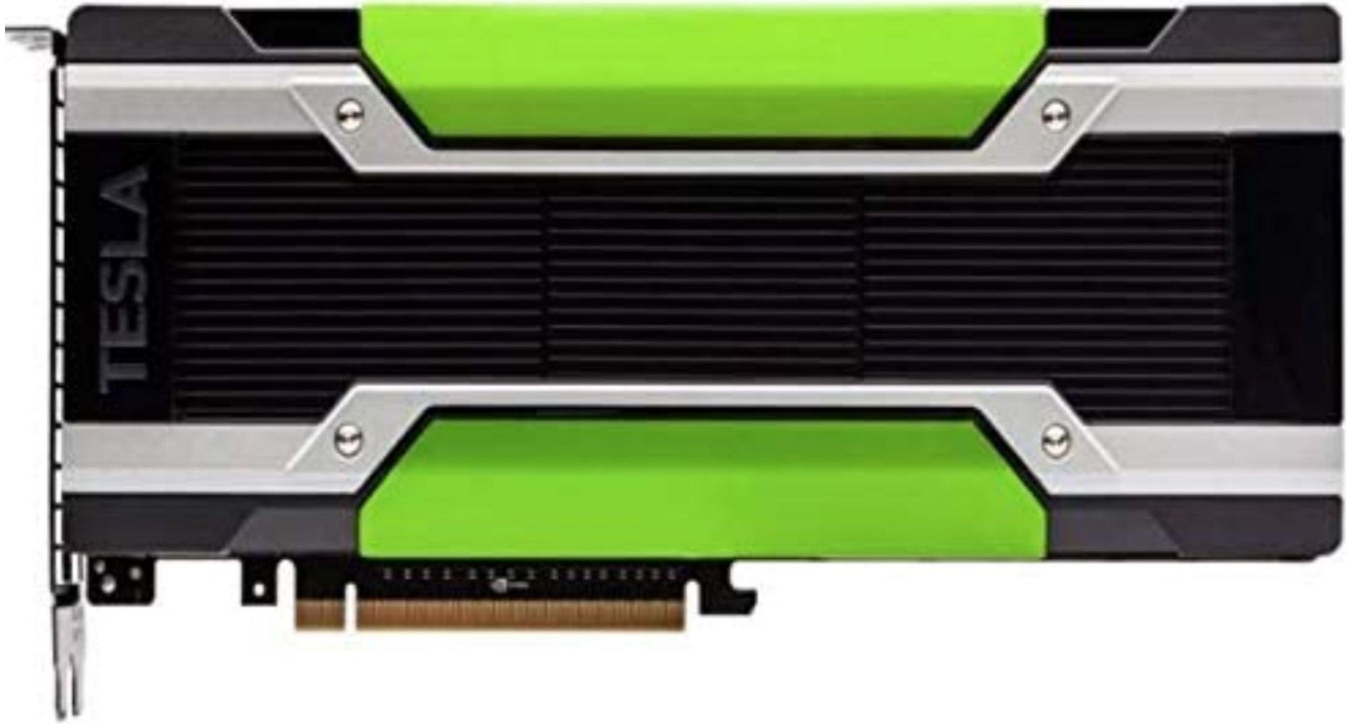
EN – FR Machine Translation Performance

Model	BLEU (%)
GRU, 50 Hidden (Bahdanau et al., 2015)	17.82
GRU, 50 Hidden + Bahdanau Attn. (ibid)	28.45
GRU, 1000 Hidden (Cho et al., 2014)	33.87
Google NMT: LSTM, 8 Layers, 1024 Hidden + Bahdanau Attn. (Wu et al., 2016)	39.92

Cost of Model Training

- The Google NMT model requires 9 days of training on 96 NVIDIA K80 GPUs.
- Yale's Grace high performance cluster only has 12 NVIDIA K80 GPUs.

Cost of Model Training



Nvidia Tesla K80 24GB GDDR5 CUDA Cores Graphic Cards

Visit the NVIDIA Store

48 ratings | 83 answered questions

8 Price Changes

Price: $397.99 & FREE Returns

S Best price S +

Pay $66.33/month for 6 months (plus S&H, tax) with 0% interest equal monthly payments when you're approved for an Amazon Prime Store Card.

May be available at a lower price from other sellers, potentially without free Prime shipping.

Brand	NVIDIA
Graphics Coprocessor	Nvidia Tesla
Video Output Interface	VGA
Graphics Processor	NVIDIA

Parallel Computation

- Graphics processing units (GPUs) are devices that perform parallel computation.
- Parallel algorithms perform multiple computations at the same time.

Example: Sequential Matrix Multiplication

Suppose $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{4\times 4}$. How do you compute $\boldsymbol{AB}$?

- Initialize $\boldsymbol{C} \leftarrow \boldsymbol{0}$.
- For $i, j \in \{1, 2, 3, 4\}$: *16 repetitions*
  - Set $C_{i,j} \leftarrow A_{i,1}B_{1,j} + A_{i,2}B_{2,j} + A_{i,3}B_{3,j} + A_{i,4}B_{4,j}$. *7 operations*
- Return $\boldsymbol{C}$.

Total time: 112 FLOPs (floating-point operations)

Example: Parallel Matrix Multiplication

- We can speed up matrix multiplication by doing it in parallel.
- Divide the matrices into 4 blocks:

$$\boldsymbol{AB} = \begin{bmatrix} \boldsymbol{A}_{:2,:2}\boldsymbol{B}_{:2,:2} + \boldsymbol{A}_{:2,3:}\boldsymbol{B}_{3:,:2} & \boldsymbol{A}_{:2,:2}\boldsymbol{B}_{:2,3:} + \boldsymbol{A}_{:2,3:}\boldsymbol{B}_{3:,3:} \\ \boldsymbol{A}_{3:,:2}\boldsymbol{B}_{:2,:2} + \boldsymbol{A}_{3:,3:}\boldsymbol{B}_{3:,:2} & \boldsymbol{A}_{3:,:2}\boldsymbol{B}_{:2,3:} + \boldsymbol{A}_{3:,3:}\boldsymbol{B}_{3:,3:} \end{bmatrix}$$

Example: Parallel Matrix Multiplication

- Initialize $\boldsymbol{C} \leftarrow \mathbf{0}$, $\boldsymbol{D} \leftarrow \mathbf{0}$.
- Fork the process into 8 threads: *12 operations each*
  - $\boldsymbol{C}_{:2,:2} \leftarrow \boldsymbol{A}_{:2,:2}\boldsymbol{B}_{:2,:2}$ $\quad$ $\boldsymbol{C}_{:2,3:} \leftarrow \boldsymbol{A}_{:2,:2}\boldsymbol{B}_{:2,3:}$ $\quad$ $\boldsymbol{C}_{3:,2:} \leftarrow \boldsymbol{A}_{3:,:2}\boldsymbol{B}_{:2,:2}$,
  - $\boldsymbol{C}_{3:,3:} \leftarrow \boldsymbol{A}_{3:,:2}\boldsymbol{B}_{:2,3:}$ $\quad$ $\boldsymbol{D}_{:2,:2} \leftarrow \boldsymbol{A}_{:2,3:}\boldsymbol{B}_{3:,:2}$ $\quad$ $\boldsymbol{D}_{:2,3:} \leftarrow \boldsymbol{A}_{:2,3:}\boldsymbol{B}_{3:,3:}$,
  - $\boldsymbol{D}_{3:,2:} \leftarrow \boldsymbol{A}_{3:,3:}\boldsymbol{B}_{3:,:2}$ $\quad$ $\boldsymbol{D}_{3:,3:} \leftarrow \boldsymbol{A}_{3:,3:}\boldsymbol{B}_{3:,3:}$
- Return $\boldsymbol{C} + \boldsymbol{D}$. *16 operations*

Total time: 28 FLOPs

Neural Networks and Parallel Computation

- Feedforward (i.e., non-recurrent) networks like MLPs can be trained very quickly using parallel computation with GPUs.
- RNNs cannot, since inputs must be processed one at a time.
- Solution: Create a feedforward architecture with attention!

Self Attention

The **self-attention head** is an attention layer that pays attention **to its own input.**

- Input: $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(n)}$
- Queries: $\boldsymbol{q}^{(t)} = \text{Linear}(\boldsymbol{x}^{(t)})$
- Keys: $\boldsymbol{k}^{(i)} = \text{Linear}(\boldsymbol{x}^{(i)})$
- Values: $\boldsymbol{v}^{(i)} = \text{Linear}(\boldsymbol{x}^{(i)})$

Self Attention

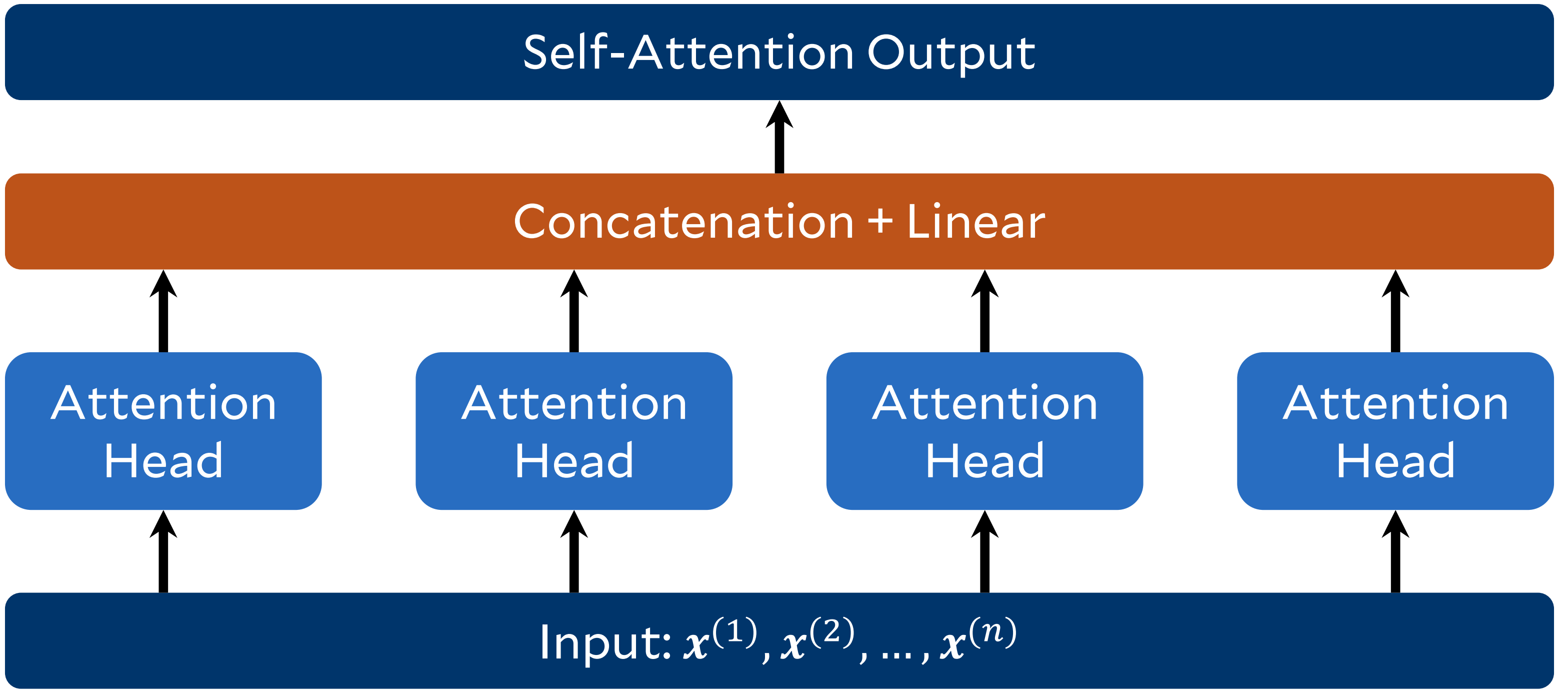
Attention scores are computed using **scaled dot-product attention**:

$$\mathrm{score}(\boldsymbol{q}, \boldsymbol{k}) = \frac{\boldsymbol{q}^\top \boldsymbol{k}}{\sqrt{d}}$$

where $\boldsymbol{q}, \boldsymbol{k} \in \mathbb{R}^d$.

Attention Heads

Multi-head attention combines several self-attention heads.



The Transformer Architecture

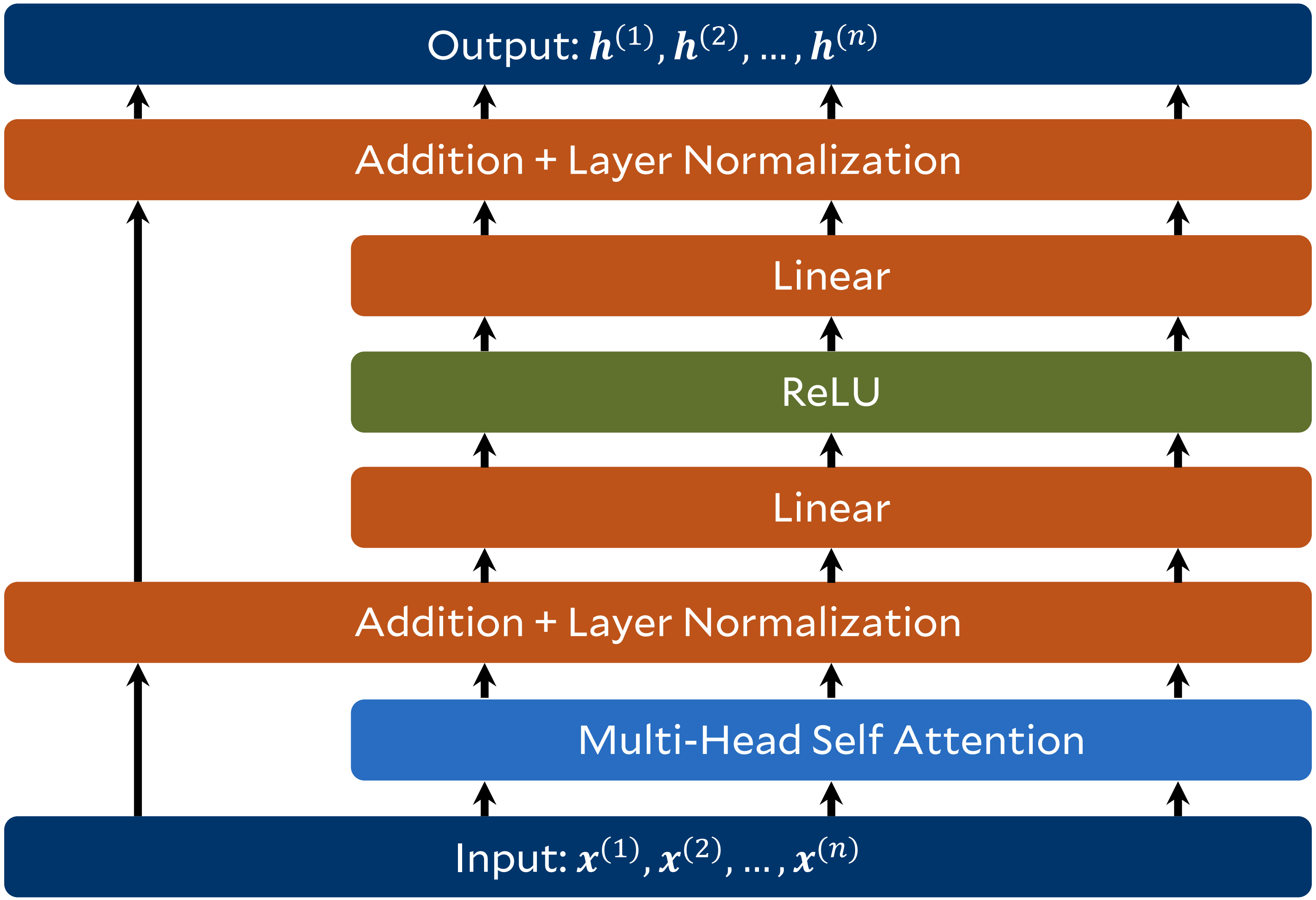
The **Transformer** is a self-attention-based architecture made up of **encoder blocks**.

Each encoder contains:

- Several attention heads
- **Layer normalization** layers

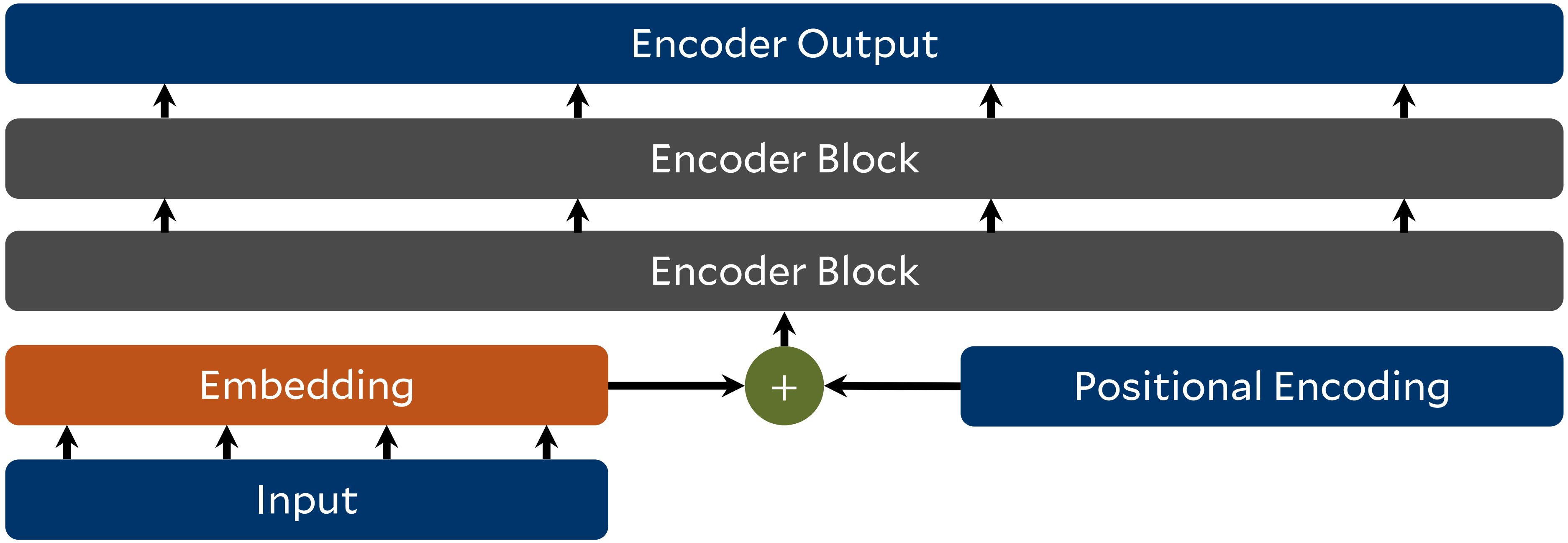
$$\mathrm{LayerNorm}(\boldsymbol{x}) = \gamma \frac{\boldsymbol{x} - \mathrm{mean}(\boldsymbol{x})}{\sqrt{\mathrm{var}(\boldsymbol{x}) + \varepsilon}} + \beta$$

Encoder Block



Transformer Encoder

The Transformer encoder combines word embeddings with a positional encoding before passing them through several encoder blocks.



Transformer Encoder

Positional Encoding: Given sequence $w_1, w_2, \ldots, w_n$, the representation of w_i is

$$\text{Embedding}(w_i) + \boldsymbol{p}^{(i)}$$

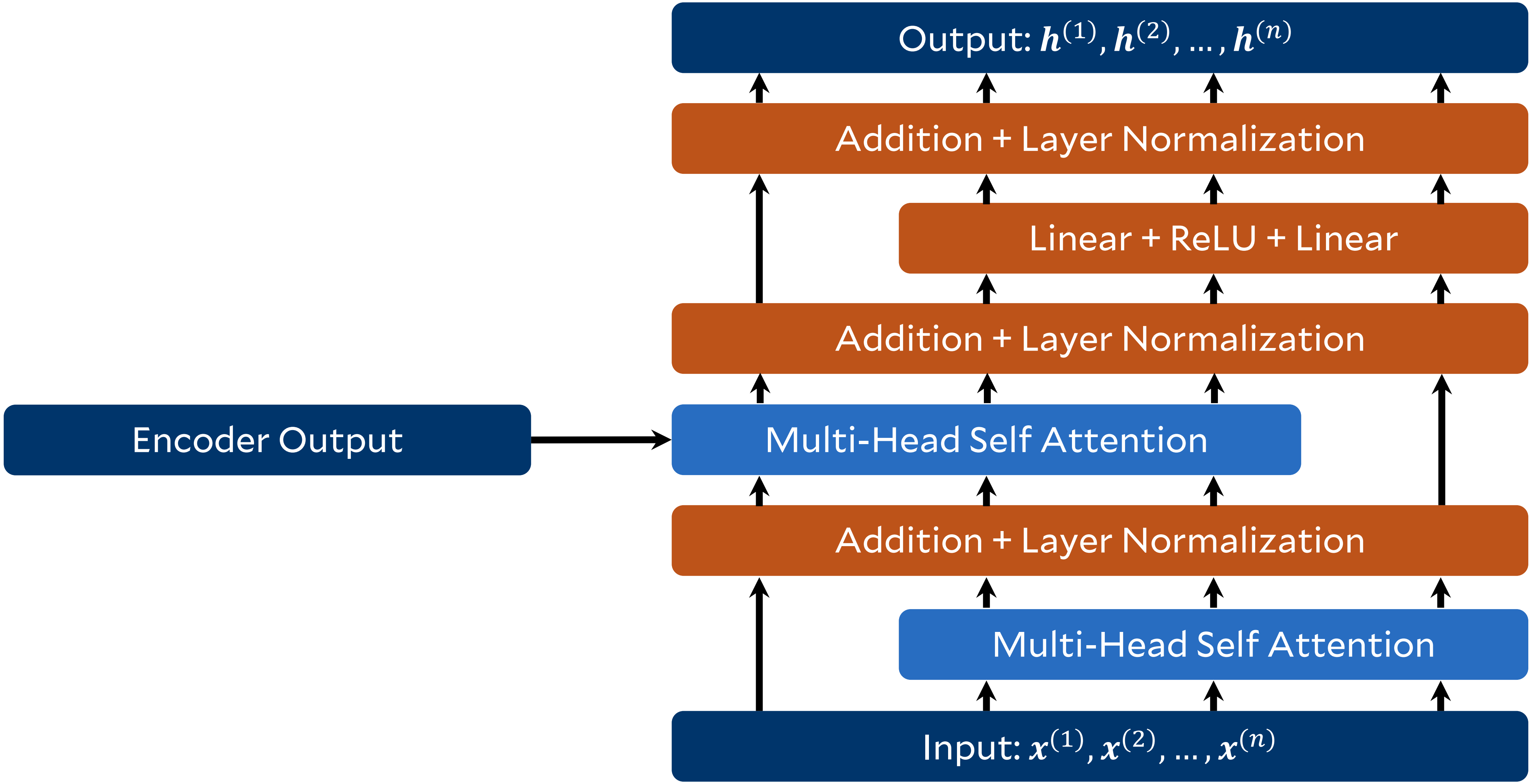
where the embedding and hidden size is h and

$$p_j^{(i)} = \begin{cases} \sin\left(i \cdot 10000^{-2j/h}\right), j \text{ is even} \\ \cos\left(i \cdot 10000^{-2j/h}\right), j \text{ is odd} \end{cases}$$

Decoder Block

The Transformer also has a decoder block for auto-regressive decoding.

Decoder Block



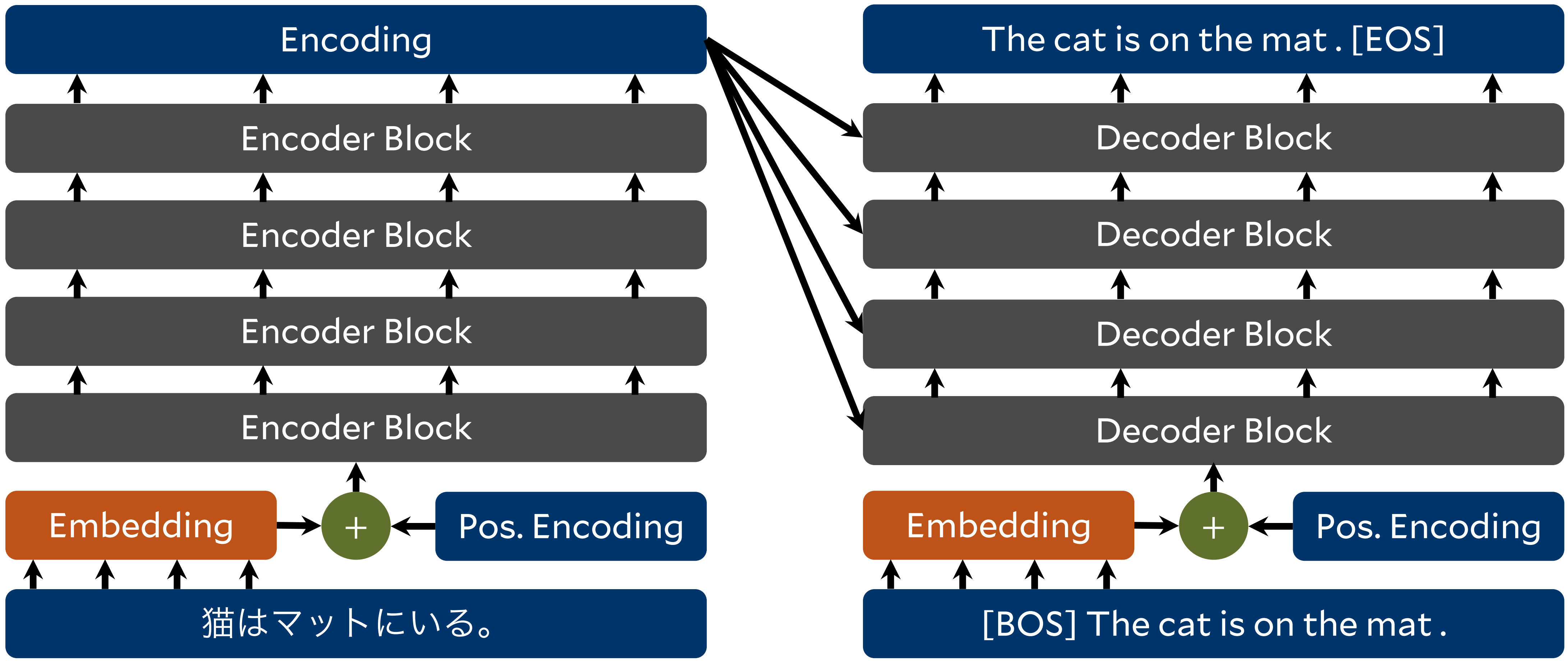
Decoder Block

- The first self-attention layer is **masked**: no position may attend to a future position. If $i \geq t$, then

$$s_i^{(t)} = -\infty$$

- In the second self-attention layer, keys and values are obtained from the encoder outputs $\boldsymbol{o}^{(i)}$:
  - Keys: $\boldsymbol{k}^{(i)} = \mathrm{Linear}(\boldsymbol{o}^{(i)})$
  - Values: $\boldsymbol{v}^{(i)} = \mathrm{Linear}(\boldsymbol{o}^{(i)})$

Transformer Encoder – Decoder



Teacher Forcing

- During training, the entire reference translation is provided to the decoder. This is called **teacher forcing**.
- Why is this okay?
- The decoder makes all predictions simultaneously, making training more parallel.
- The decoder is never trained to make predictions based on incorrectly predicted previous words.

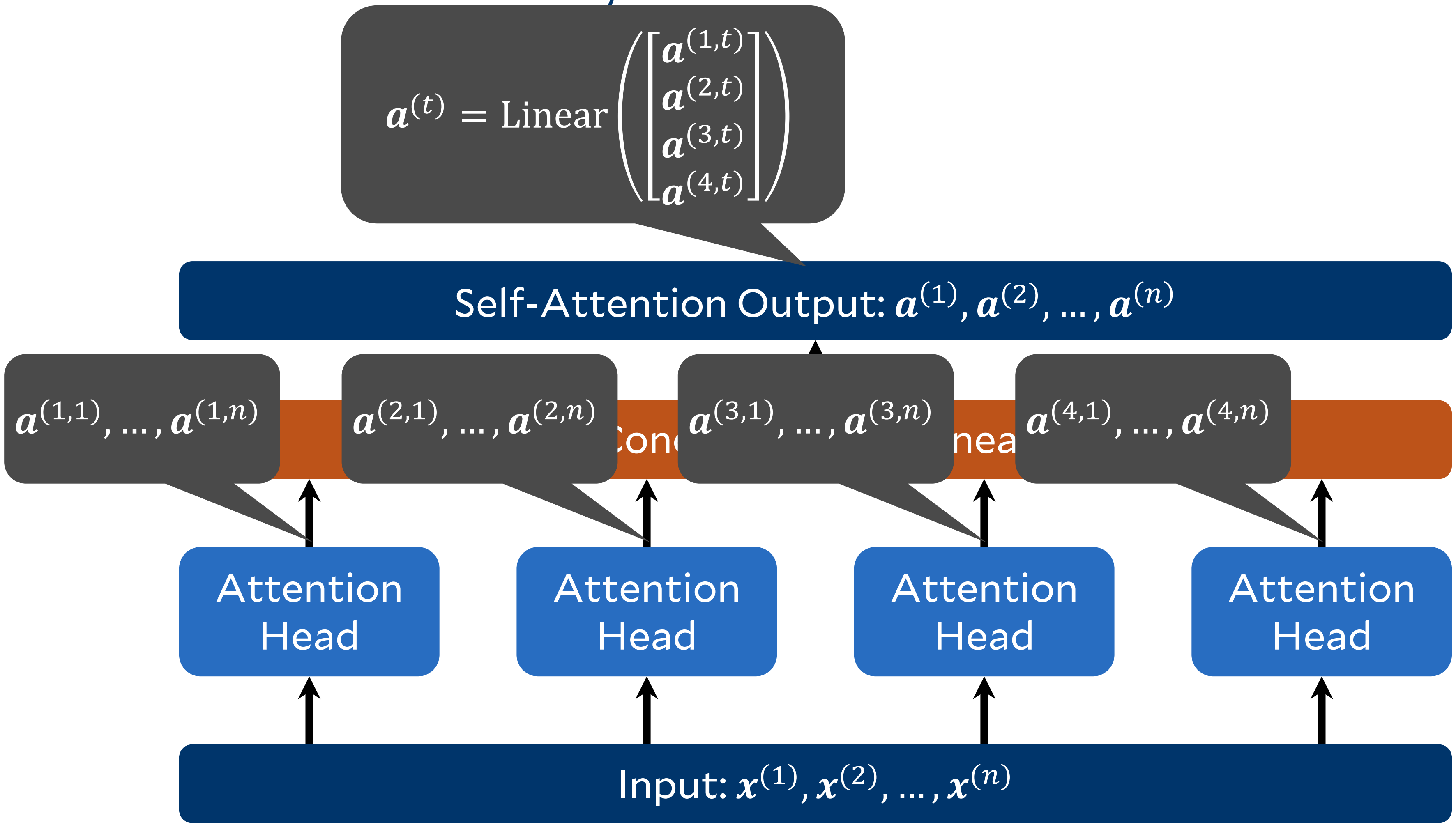
EN – FR Machine Translation Performance

Model	BLEU (%)
GRU, 50 Hidden (Bahdanau et al., 2015)	17.82
GRU, 50 Hidden + Bahdanau Attn. (ibid)	28.45
GRU, 1000 Hidden (Cho et al., 2014)	33.87
Google NMT: LSTM, 8 Layers, 1024 Hidden + Bahdanau Attn. (Wu et al., 2016)	39.92
Transformer, 512 Hidden, 8 Heads (Vaswani et al., 2017)	38.1
Transformer, 1024 Hidden, 16 Heads (ibid)	41.0

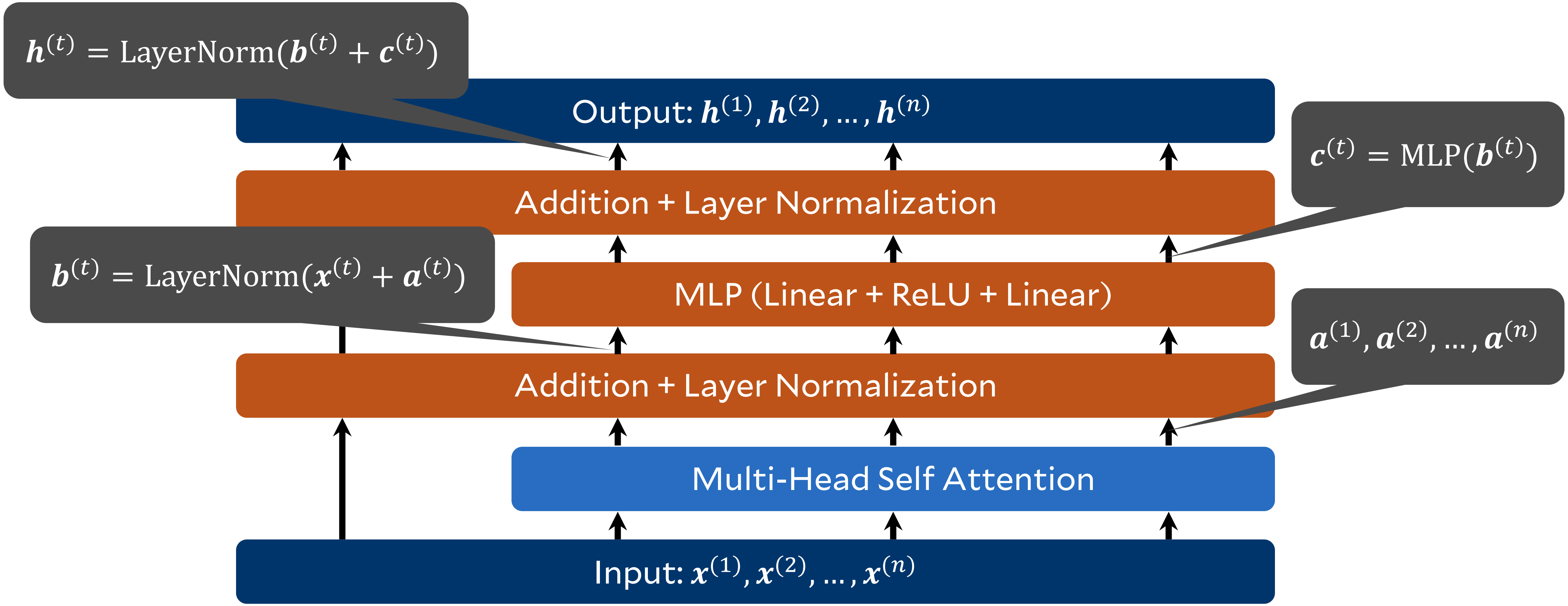
Self-Attention Head

- **Input:** $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(n)}$
- **Output:** $\boldsymbol{a}^{(1)}, \boldsymbol{a}^{(2)}, \ldots, \boldsymbol{a}^{(n)}$
- Queries:
  $$\boldsymbol{q}^{(t)} = \text{Linear}\left(\boldsymbol{x}^{(t)}\right) \in \mathbb{R}^d$$
- Keys:
  $$\boldsymbol{k}^{(i)} = \text{Linear}\left(\boldsymbol{x}^{(i)}\right) \in \mathbb{R}^d$$
- Values:
  $$\boldsymbol{v}^{(i)} = \text{Linear}(\boldsymbol{x}^{(i)})$$
- Attention Scores:
  $$s_i^{(t)} = \text{score}\left(\boldsymbol{q}^{(t)}, \boldsymbol{k}^{(i)}\right) = \left(\boldsymbol{q}^{(t)}\right)^{\top} \boldsymbol{k}^{(i)} / \sqrt{d}$$
- Attention Weights:
  $$\boldsymbol{\alpha}^{(t)} = \text{softmax}(\boldsymbol{s}^{(t)})$$
- Attention Vectors:
  $$\boldsymbol{a}^{(t)} = \sum_i \alpha_i^{(t)} \boldsymbol{v}^{(i)}$$

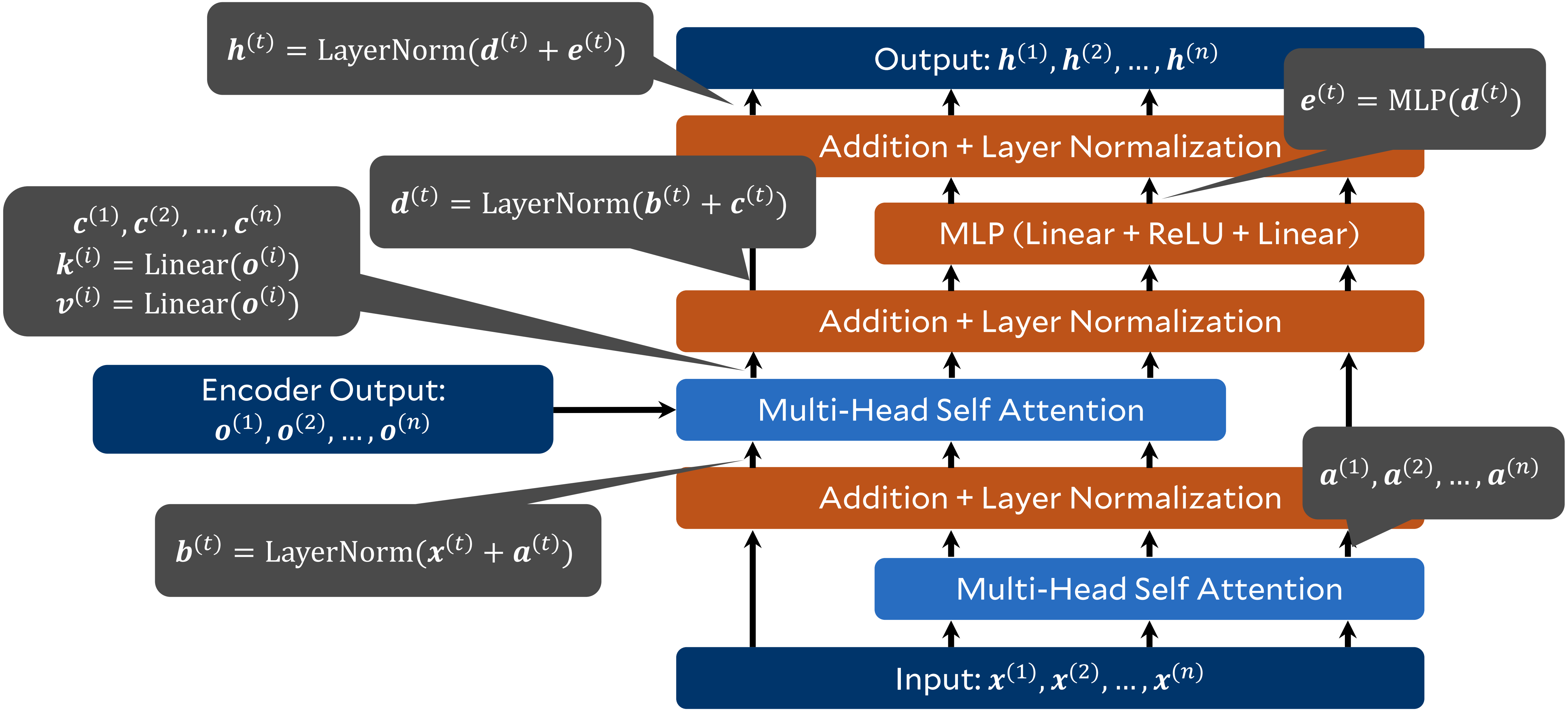
Self-Attention Layer



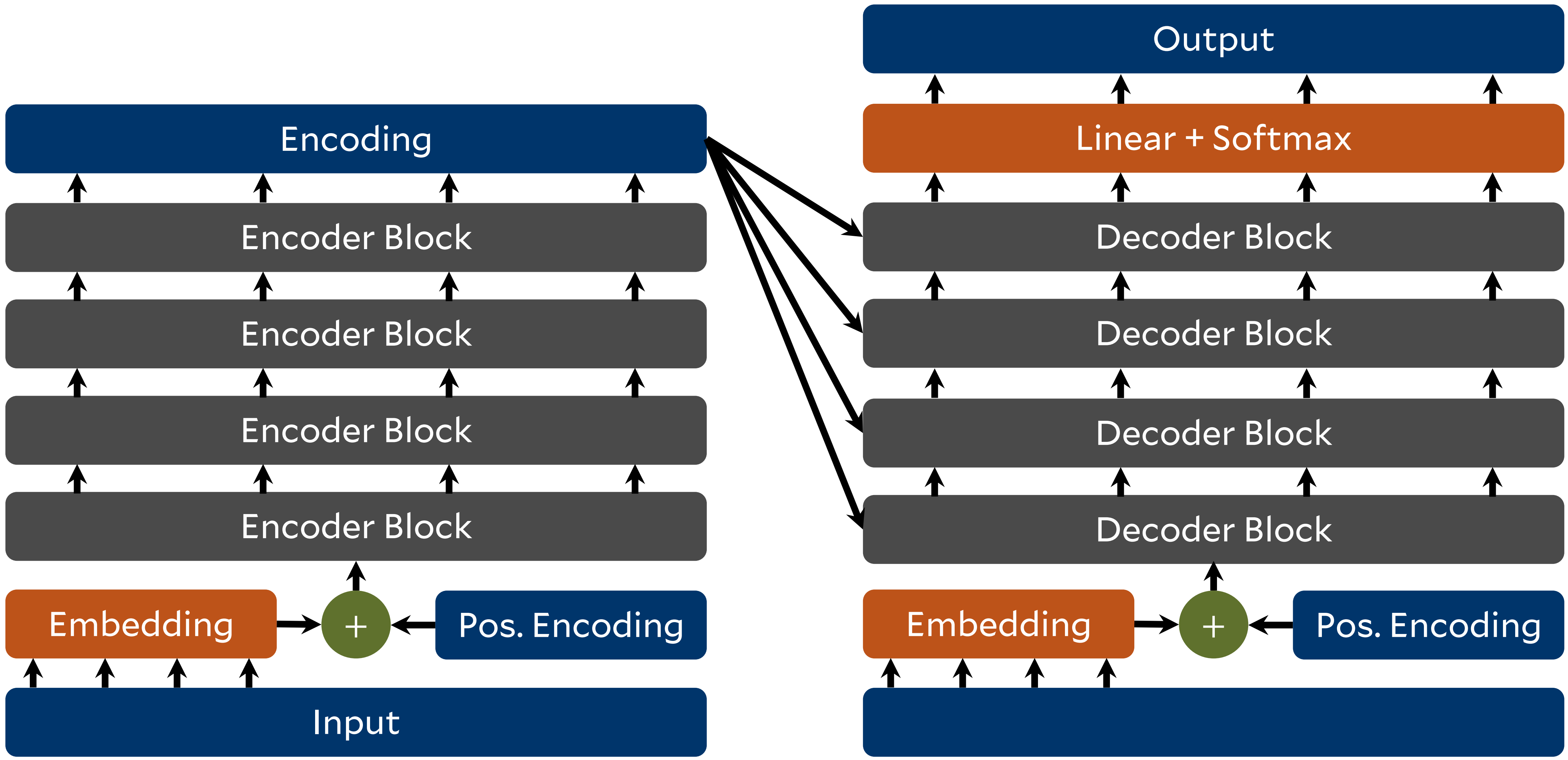
Encoder Block



Decoder Block



Transformer Encoder – Decoder



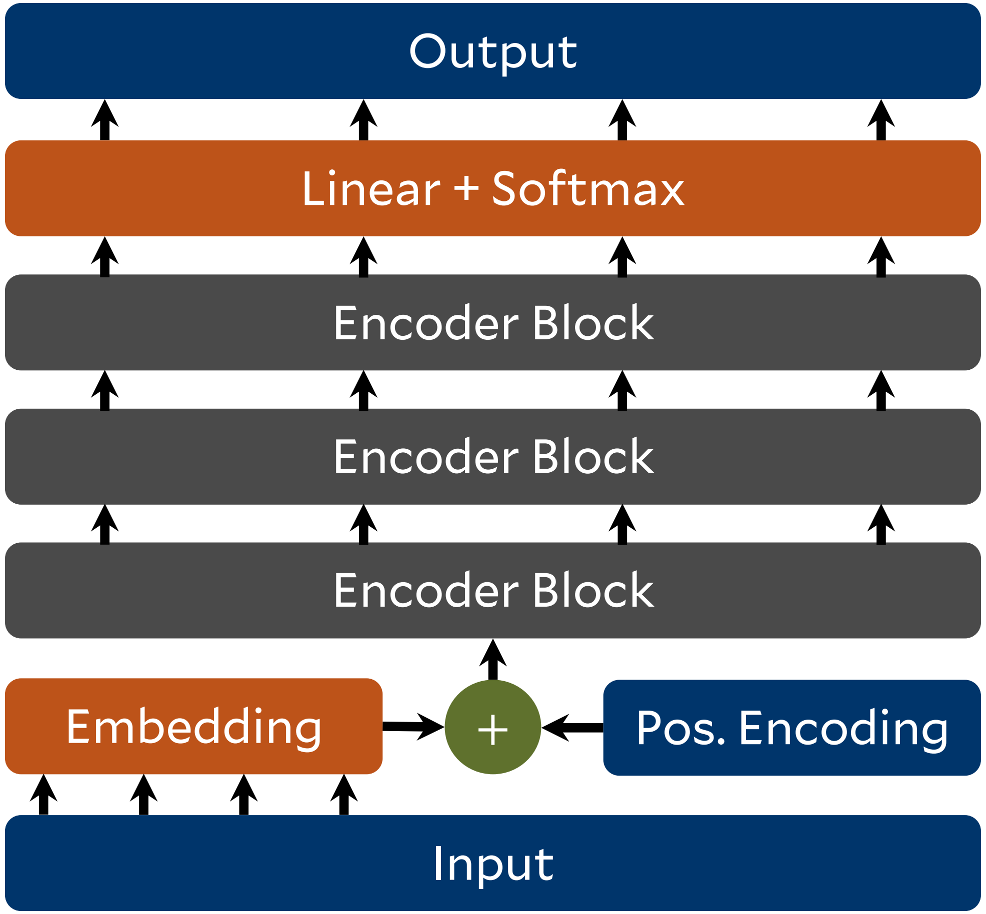
Types of Recurrent Networks

	Single Input	Sequence Input
Single Output	MLP	RNN Classifier
Sequence Output	RNN Generator	RNN Encoder–Decoder

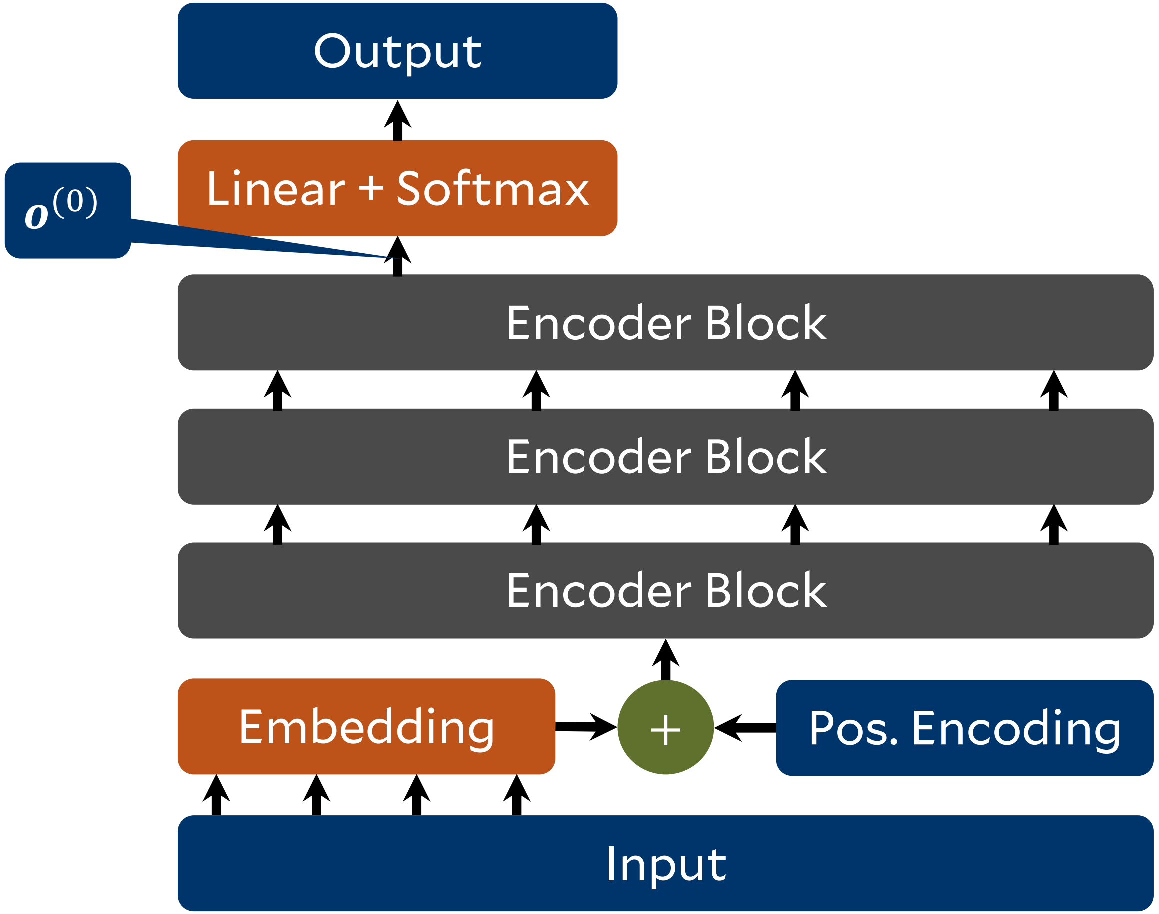
Types of Transformer Networks

	Single Input	Sequence Input
Single Output	MLP	Transformer Classifier
Sequence Output	Transformer Generator	Transformer Encoder–Decoder

Transformer Generator



Transformer Classifier



Transformer vs. RNN Training Time

Model	BLEU (%)	Time (PetaFLOPs)
Google NMT	39.92	140,000
Transformer (Base)	38.1	3,300
Transformer (Big)	41.0	23,000