Attention Mechanisms and Transformers

Applied Deep Learning — Chapter 10

Yin Yang

Foundation Books Project

Roadmap

Sequence-to-Sequence and the Bottleneck

The Attention Model

Self-Attention and Multi-Head Attention

The Transformer

Pretrained Transformers

Accuracy and Speed in Practice

Summary

Sequence-to-Sequence and the Bottleneck

From RNNs to Selective Reading

Many applied tasks are *input-output mapping*, not classification:

- machine translation: source sentence $\rightarrow$ target sentence
- image captioning: image $\rightarrow$ caption
- speech recognition: acoustic frames $\rightarrow$ characters

Common thread: selective use of information. A decoder may need "not" precisely when emitting a negation token.

Running examples in this chapter:

- math: machine translation
- applied: multi-label toxic-comment classification

Encoder–Decoder Architecture

Encoder maps source to representation:

$$z = f_\theta(x_{1:T_x}) \in \mathbb{R}^d.$$

Decoder factors target probability autoregressively:

$$p_\phi(y_{1:T_y} \mid x_{1:T_x}) = \prod_{t=1}^{T_y} p_\phi(y_t \mid y_{<t}, z).$$

Teacher forcing during training: condition on *true* previous tokens. At inference, the decoder must use its own outputs — errors can propagate.

The Fixed-Vector Bottleneck

Early seq2seq passed only $z = h_{T_x}$ to the decoder.

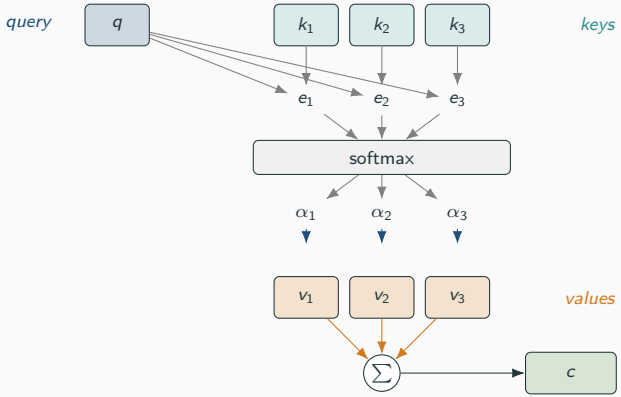
- One vector must support every output position.
- No explicit operation for "look again at source token 4 now."
- Pressure grows with sentence length and word-order divergence.

Bahdanau, Cho, Bengio (2015): let the decoder *search softly, differentiably* over source positions while generating each target word.

Attention is the operation that makes selective re-reading explicit.

The Attention Model

Query, Key, Value



$e_i = \mathrm{score}(q, k_i)$, $\alpha = \mathrm{softmax}(e)$, $c = \sum_i \alpha_i v_i$. Differentiable retrieval; trainable end-to-end.

A Hand Calculation

Let $q = (1, 0)$, keys $k_1 = (1, 0),\ k_2 = (0, 1),\ k_3 = (1, 1)$.

Dot-product scores: $e_1 = 1,\ e_2 = 0,\ e_3 = 1$. Then

$$\exp(e_i) \approx (2.718,\ 1.000,\ 2.718), \quad \text{sum} \approx 6.436,$$

$$\alpha \approx (0.422,\ 0.155,\ 0.422).$$

With values $v_1 = (2, 0),\ v_2 = (0, 3),\ v_3 = (1, 1)$:

$$c \approx 0.422(2, 0) + 0.155(0, 3) + 0.422(1, 1) \approx (1.27, 0.89).$$

q matches k_1 and k_3 equally; the context blends v_1 and v_3.

Score Functions: Additive vs. Multiplicative

Additive (Bahdanau):

$$e_i = w_a^\top \tanh(W_q q + W_k k_i + b_a).$$

Flexible; handles different d_q, d_k.

Multiplicative (Luong):

$$e_i = q^\top k_i \quad \text{or} \quad e_i = q^\top W_a k_i.$$

Scaled dot-product (Vaswani):

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

Scale before softmax so high-dimensional dot products don't saturate the softmax.

Masks Are Inputs, Not Decoration

Suppose scores $(2, 1, -4, 0)$ but position 4 is padding.

Replace padding score with $-\infty$ before softmax:

$$(2, 1, -4, -\infty) \;\rightarrow\; \exp \approx (7.389, 2.718, 0.018, 0).$$

Weights become $(0.730, 0.268, 0.002, 0.000)$.

- **Padding mask:** hide positions added for batching.
- **Causal mask:** hide future positions in autoregressive decoding.

Without masks, models can "learn" from padding artifacts.

Self-Attention and Multi-Head Attention

Self-Attention

Queries, keys, values all come from *the same* sequence $X \in \mathbb{R}^{T \times d}$:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V,$$

$$\text{SelfAttn}(X) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V.$$

Each token mixes information from every other token (subject to masks).

RNN: token 5 reaches token 20 only through 14 intermediate states. Self-attention: one matrix produces all pairwise reads at once.

Shape Walkthrough

$T = 5$ tokens, embedding dim $d = 4$, head dims $d_k = d_v = 3$.

$$X \in \mathbb{R}^{5\times 4}, \quad W_Q, W_K, W_V \in \mathbb{R}^{4\times 3}.$$

Then $Q, K, V \in \mathbb{R}^{5\times 3}$, and

$$QK^\top \in \mathbb{R}^{5\times 5}, \quad \text{softmax}(\cdot)\, V \in \mathbb{R}^{5\times 3}.$$

- Cost grows as T^2 per head: $T = 100 \Rightarrow$ 10,000 scores; $T = 500 \Rightarrow$ 250,000.
- Doubling max length more than doubles attention memory.

Maximum sequence length is one of the most consequential knobs.

Positional Information

Self-attention treats the input as a *set* of vectors.

Position must be injected somehow:

- sinusoidal positional encodings (original Transformer)
- learned absolute position embeddings (BERT)
- rotary position embeddings, RoPE (modern decoders)
- relative position biases

Different mechanisms, same applied requirement: the model must know token order.

Multi-Head Attention

Run M heads with different projections in parallel:

$$O^{(m)} = \text{Attention}(HW_Q^{(m)},\, HW_K^{(m)},\, HW_V^{(m)}),$$

$$\text{MHA}(H) = [O^{(1)}; \ldots; O^{(M)}]\, W_O.$$

- Each head can attend in a different learned similarity space.
- Heads are *not* reliably labeled ("the negation head").
- Practical motive: representational capacity, not interpretability.

The Transformer

A Transformer Block

Two sublayers, each with residual + normalization.

Encoder block:

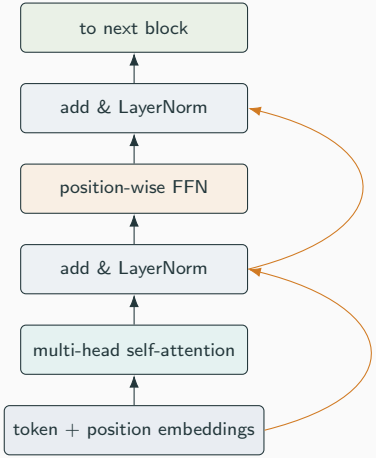
$$\tilde{H} = \text{LayerNorm}(H + \text{MHA}(H)),$$

$$H_{\text{out}} = \text{LayerNorm}(\tilde{H} + \text{FFN}(\tilde{H})).$$

- Attention mixes tokens together.
- FFN transforms each token independently.
- Residuals preserve information; LayerNorm stabilizes training.

Pre-norm variants move LayerNorm before each sublayer, but the two-job pattern is the same.

Encoder Block (schematic)



Orange: residual paths around attention and FFN.

Three Kinds of Attention

Attention type	Queries from	Keys/values from
Encoder self-attention	source sequence	source sequence
Decoder masked self-attention	target prefix	target prefix
Cross-attention	decoder states	encoder outputs

- Encoder-only model (e.g. BERT): self-attention only.
- Decoder-only model (e.g. GPT): causally masked self-attention.
- Encoder–decoder (e.g. T5, original Transformer): all three.

Training Speed vs. Generation Speed

Training:

- Teacher forcing supplies all target tokens.
- Causal mask hides future positions.
- All target positions can be processed in parallel.

Inference (autoregressive):

- Each new token depends on previously generated ones.
- Implementations cache previous keys and values (KV cache).
- Generation is still sequential, token by token.

This is why an encoder classifier finishes in one forward pass while a decoder must emit text step by step.

Pretrained Transformers

Match Architecture to Task Shape

Encoder-only (BERT, ModernBERT) reads input bidirectionally; strong for classification, retrieval, sentence embeddings.

Decoder-only (GPT family) autoregressive generation; foundation for modern LLMs.

Encoder–decoder (T5, BART) sequence-to-sequence: translation, summarization, text-to-text question answering.

First question: what must the model output? Toxic-comment classification needs label logits, not generated tokens.

Tokenizers Are Part of the Model

- Tokenizer maps text to IDs in the checkpoint's vocabulary.
- Subword units (BPE, SentencePiece) handle unfamiliar words.
- Special tokens, truncation, attention masks, sometimes token-type IDs.
- Changing tokenizer changes the meaning of every input ID.

Record the checkpoint and tokenizer together. Subword splits also explain why the same prompt can use different token counts under different models.

Fine-Tuning a Pretrained Encoder

Frozen feature extractor vs. end-to-end fine-tuning is a deliberate choice.

Conservative recipe for toxic-comment classification:

1. Start small: DistilBERT or ModernBERT-small.
2. Short max length until pipeline is verified.
3. Multi-label loss; track macro F1, micro F1, ROC-AUC.
4. Validate before any leaderboard submission.

Larger checkpoints are an experimental variable, not a guarantee.

Accuracy and Speed in Practice

Length 128 vs. 256 (DistilBERT, 20k subset)

Setting	Max len	Macro F1	Micro F1	ROC-AUC	Epoch
A	128	0.4735	0.7776	0.9735	1:15
B	256	0.4659	0.7708	0.9763	1:40

Length 256 wins ROC-AUC by 0.003 but loses on F1 and is slower.

Choose by truncation evidence. In this dataset 19.3% of comments truncate at 128, 6.9% at 256, 2.2% at 512 — so 128 is defensible if false negatives don't track length.

Where Speed Disappears

- **Padding:** dynamic padding to per-batch max; bucketing by length.
- **Batch size:** fits in memory, then gradient accumulation simulates larger batches.
- **Precision:** fp16 / bf16 mixed precision for speed and memory; record what was used.
- **Optimizer:** small LR (2×10^{-5} to 5×10^{-5}), AdamW, warmup, early stopping on validation.
- **Hardware:** CPU, older GPU, recent GPU + AMP — not comparable as raw timings.

Validation Discipline for Fine-Tuning

Pre-flight checks before launching:

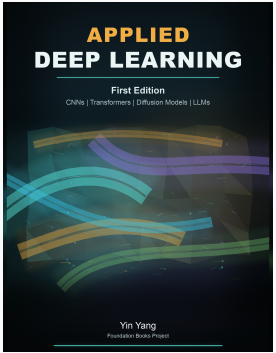
- Verify multi-label labels and shapes.
- Print package versions, checkpoint, tokenizer, seed.
- Inspect tokenized examples (longest *and* shortest).
- Use dynamic padding; record max length.
- Monitor validation loss *and* task metrics.
- Record runtime, hardware, precision, batch size.
- Change one factor at a time.
- Save best validation checkpoint.
- Touch the test set *only after* model selection.

Summary

Summary

- **Bottleneck:** one fixed encoder vector cannot serve every output step. Attention makes retrieval differentiable.
- **Q/K/V:** score, softmax, weighted average. Scale before softmax; masks are part of the computation.
- **Self-attention:** replaces the recurrent chain with one pairwise read. Position must be injected.
- **Transformer block:** attention mixes tokens, FFN transforms each token; residual + LayerNorm.
- **Three attention types:** encoder self, decoder masked self, cross.
- **Applied:** match architecture to task shape; pretrain + fine-tune; report quality *and* cost; validate before testing.

Next: from the Transformer to Large Language Models — tokens, context windows, decoding, KV caches, and the prompt-only baseline.



Applied Deep Learning

Chapter overview slides for the book by Yin Yang.

Book page	`foundation-books.github.io/applied-deep-learning`
Code	`github.com/foundation-books/applied-deep-learning-code`
Print	`amazon.com/dp/B0GZF7QRSH`
Kindle	`amazon.com/dp/B0GZF9221X`
License	CC BY-NC-ND 4.0

These free overview slides may be shared non-commercially with attribution and without modifications. Full explanations, exercises, and companion-code context are in the book and linked repository.