Attention as Bilinear Form

A Physicist's Guide to Transformer Attention

Tensor Calculus, Statistical Mechanics, and Differential Geometry

Tutorial Notes
January 2025

Contents

1 Introduction

The attention mechanism, introduced by Bahdanau et al. [1] and refined in the Transformer architecture by Vaswani et al. [2], has become the foundation of modern deep learning. While typically presented in matrix notation optimized for implementation, the underlying mathematical structure reveals deep connections to classical physics and differential geometry.

This tutorial recasts the attention mechanism in the language of tensor calculus, making explicit the geometric and statistical mechanical structure hidden in the standard formulation. Our goals are:

1. **Tensor Calculus**: Express attention using index notation with proper contravariant/covariant indices and Einstein summation convention
2. **Bilinear Forms**: Show that attention scores arise from a bilinear form with an implicit metric tensor
3. **Statistical Mechanics**: Interpret softmax as the Gibbs/Boltzmann distribution from thermodynamics
4. **Differential Geometry**: Understand the feature space as a Riemannian manifold
5. **Gradients**: Derive backpropagation formulas in index notation and verify against autodiff

1.1 Notation Conventions

Throughout this document, we use the following conventions:

Definition (Index Notation).

- **Superscripts** denote contravariant (column vector) indices: v^a, Q^{ia}
- **Subscripts** denote covariant (row vector/dual) indices: g_{ab}, u_a
- **Einstein summation**: Repeated indices (one up, one down) are summed: $v^a u_a = \sum_a v^a u_a$
- **Kronecker delta**: $\delta_b^a = \begin{cases} 1 \text{ if } a{=}b \\ 0 \text{ otherwise} \end{cases}$

The key index labels we use:

Index	Meaning
i	Query sequence position (n_q positions)
j	Key/value sequence position (n_k positions)
a, b	Feature/embedding dimensions (d_k or d_v)
h	Attention head index (H heads)
μ, ν	Pattern index in Hopfield networks

2 Part I: Foundations

2.1 Vectors and Dual Vectors

In physics, we distinguish between vectors and their duals (covectors). A vector v^a lives in a vector space V, while a covector u_a lives in the dual space V^*. The natural pairing between them is:

$$\langle u, v \rangle = u_a v^a \tag{1}$$

This is basis-independent. In a coordinate basis, this becomes the familiar dot product.

Definition (Metric Tensor). A **metric tensor** g_{ab} is a symmetric, positive-definite $(0, 2)$-tensor that defines an inner product on the vector space:

$$\langle u, v \rangle_g = u^a g_{ab} v^b \tag{2}$$

The metric allows us to:
1. **Lower indices**: $v_a = g_{ab} v^b$ (convert vector to covector)
2. **Raise indices**: $v^a = g^{ab} v_b$ (convert covector to vector)

where g^{ab} is the inverse metric satisfying $g^{ac} g_{cb} = \delta^a_b$.

2.2 Bilinear Forms

Definition (Bilinear Form). A **bilinear form** is a map $B : V \times W \to \mathbb{R}$ that is linear in both arguments:

$$B(\alpha u + \beta v, w) = \alpha B(u, w) + \beta B(v, w) \tag{3}$$

$$B(u, \alpha v + \beta w) = \alpha B(u, v) + \beta B(u, w) \tag{4}$$

In index notation with a matrix M_{ab}:

$$B(u, v) = u^a M_{ab} v^b \tag{5}$$

The connection to attention: the attention score between a query q and key k is precisely a bilinear form:

$$S = q^a g_{ab} k^b \tag{6}$$

where the metric g_{ab} encodes how we measure similarity in feature space.

2.3 Standard Metrics

Example. Euclidean metric: $g_{ab} = \delta_{ab}$ (identity matrix)

This gives the standard dot product: $\langle u, v \rangle = u^a \delta_{ab} v^b = u^a v_a$

Example. Scaled Euclidean metric: $g_{ab} = \frac{1}{\sqrt{d_k}} \delta_{ab}$

This is precisely the metric implicit in scaled dot-product attention! The $\frac{1}{\sqrt{d_k}}$ factor prevents dot products from growing too large in high dimensions.

Example. Learned bilinear metric: $g_{ab} = W_a^c W_{cb}$

Parameterizing the metric as $W^T W$ ensures positive semi-definiteness. This generalizes standard attention to learnable similarity functions.

3 Part II: The Attention Mechanism

3.1 Attention as Tensor Contraction

The attention mechanism operates on three inputs:

- **Queries** Q^{ia}: What we're looking for (shape: $n_q \times d_k$)
- **Keys** K^{ja}: What we're matching against (shape: $n_k \times d_k$)
- **Values** V^{jb}: What we retrieve (shape: $n_k \times d_v$)

The mechanism proceeds in three steps:

3.1.1 Step 1: Score Computation

Compute pairwise similarity between all queries and keys:

Definition (Attention Scores).

$$S^{ij} = \frac{1}{\sqrt{d_k}} Q^{ia} K^{ja} \tag{7}$$

Or with explicit metric: $S^{ij} = Q^{ia} g_{ab} K^{jb}$

where $g_{ab} = \frac{1}{\sqrt{d_k}} \delta_{ab}$.

The contraction over the feature index a computes a scalar similarity for each query-key pair. This is a **bilinear form** evaluated for all pairs.

In code (JAX with einsum):

```
# S^{ij} = Q^{ia} K^{ja} / sqrt(d_k)
S = jnp.einsum('ia,ja->ij', Q, K) / jnp.sqrt(d_k)
```

3.1.2 Step 2: Softmax Normalization

Convert scores to a probability distribution over keys:

Definition (Attention Weights).

$$A^{ij} = \frac{\exp(S^{ij})}{\sum_k \exp(S^{ik})} = \frac{\exp(S^{ij})}{Z^i} \tag{8}$$

where $Z^i = \sum_j \exp(S^{ij})$ is the **partition function** for query i.

The softmax is applied row-wise: each query i gets its own probability distribution over keys.

3.1.3 Step 3: Value Aggregation

Compute weighted sum of values:

Definition (Attention Output).

$$O^{ib} = A^{ij} V^{jb} \tag{9}$$

This contracts over the key index j, producing an output for each query.

3.2 Full Attention in One Equation

Combining all steps:

Theorem (Scaled Dot-Product Attention).

$$O^{ib} = \frac{\exp(Q^{ia} g_{ac} K^{jc})}{\sum_k \exp(Q^{ia} g_{ac} K^{kc})} V^{jb} \tag{10}$$

Or in matrix notation:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{11}$$

4 Part III: Statistical Mechanics

The softmax function is not merely a normalization trick—it is the **Gibbs distribution** from statistical mechanics, revealing deep connections to thermodynamics.

4.1 The Gibbs/Boltzmann Distribution

In statistical mechanics, a system with energy levels E_j at temperature T has probability:

Definition (Gibbs Distribution).

$$P(j) = \frac{\exp(-E_j/T)}{Z}, \quad \text{where } Z = \sum_j \exp(-E_j/T) \tag{12}$$

- T: Temperature (controls distribution sharpness)
- Z: Partition function (normalization)
- $\beta = 1/T$: Inverse temperature

For attention, we identify:

- **Scores as negative energies**: $S^{ij} = -E^{ij}$ (higher score = lower energy = preferred)
- **Temperature**: Standard attention uses $T = 1$

Thus:

$$A^{ij} = \frac{\exp(S^{ij}/T)}{\sum_k \exp(S^{ik}/T)} \tag{13}$$

4.2 Temperature Effects

The temperature parameter controls how "peaked" the attention distribution is:

Proposition. As temperature varies:

- $T \to 0$: **Hard attention** (argmax). Attention concentrates on highest-scoring key.
- $T = 1$: **Standard softmax**. Balanced distribution.
- $T \to \infty$: **Uniform attention**. All keys weighted equally.

The $\frac{1}{\sqrt{d_k}}$ scaling in standard attention can be interpreted as setting an effective temperature $T = \sqrt{d_k}$ when using unscaled scores.

4.3 Entropy and Information

The **Shannon entropy** of attention weights measures how diffuse or focused the attention is:

Definition (Attention Entropy).

$$H^i = -\sum_j A^{ij} \log A^{ij} \tag{14}$$

- $H = 0$: Delta distribution (all attention on one key)
- $H = \log(n_k)$: Uniform distribution (equal attention on all keys)

The **normalized entropy** $H^i / \log(n_k) \in [0, 1]$ provides a scale-independent measure.

4.4 Free Energy

The **free energy** combines energy and entropy:

Definition (Free Energy).

$$F^i = -T \log Z^i = -T \log \sum_j \exp(S^{ij}/T) \tag{15}$$

At temperature T, the free energy satisfies:

$$F = \langle E \rangle - T \cdot H \tag{16}$$

where $\langle E \rangle$ is the expected energy and H is the entropy.

Minimizing free energy balances:

1. **Low energy**: Attend to high-scoring keys
2. **High entropy**: Spread attention broadly (regularization)

This provides a principled way to understand attention with temperature.

5 Part IV: Differential Geometry

The feature space where queries and keys live can be understood as a Riemannian manifold, with the metric tensor defining geometry.

5.1 Riemannian Metrics

Definition (Riemannian Manifold). A **Riemannian manifold** (M, g) is a smooth manifold M equipped with a metric tensor $g_{ab}(x)$ at each point $x \in M$.

The metric defines:

- **Distances**: $ds^2 = g_{ab} dx^a dx^b$
- **Angles**: $\cos\theta = \frac{u^a g_{ab} v^b}{\|u\|_g \|v\|_g}$
- **Volumes**: $dV = \sqrt{\det g} dx^1 ... dx^n$

In attention, the feature space $\mathbb{R}^{d_k}$ is (implicitly) a Riemannian manifold with metric $g_{ab} = \frac{1}{\sqrt{d_k}} \delta_{ab}$.

5.2 Christoffel Symbols and Covariant Derivatives

For a general metric $g_{ab}(x)$ that varies with position, we need **covariant derivatives** to properly differentiate tensors:

Definition (Christoffel Symbols). The **Christoffel symbols of the second kind** are:

$$\Gamma^c_{ab} = \frac{1}{2} g^{cd} (\partial_a g_{bd} + \partial_b g_{ad} - \partial_d g_{ab}) \tag{17}$$

Definition (Covariant Derivative). For a contravariant vector v^b:

$$\nabla_a v^b = \partial_a v^b + \Gamma^b_{ac} v^c \tag{18}$$

For a covariant vector u_b:

$$\nabla_a u_b = \partial_a u_b - \Gamma^c_{ab} u_c \tag{19}$$

For the standard (constant) attention metric, $\Gamma^c_{ab} = 0$ and covariant derivatives reduce to ordinary derivatives.

5.3 Natural Gradient Descent

When optimizing over parameter space, the **Fisher information matrix** provides a natural Riemannian metric:

Definition (Fisher Information).

$$F_{ij} = \mathbb{E}\left[\frac{\partial \log p(x|\theta)}{\partial \theta^i} \frac{\partial \log p(x|\theta)}{\partial \theta^j} \right] \tag{20}$$

Natural gradient descent uses this metric:

$$\Delta\theta^i = -\eta \left(F^{-1}\right)^{ij} \frac{\partial L}{\partial \theta^j} \tag{21}$$

This is invariant under reparameterization and often converges faster than standard gradient descent.

6 Part V: Gradient Derivations

We now derive the gradients for backpropagation through attention, using index notation throughout.

6.1 Chain Rule in Index Notation

For a scalar loss L, the chain rule gives:

$$\frac{\partial L}{\partial Q^{kl}} = \frac{\partial L}{\partial O^{ib}} \frac{\partial O^{ib}}{\partial A^{mn}} \frac{\partial A^{mn}}{\partial S^{pq}} \frac{\partial S^{pq}}{\partial Q^{kl}} \tag{22}$$

We compute each factor.

6.2 Gradient Through Value Aggregation

Given $O^{ib} = A^{ij} V^{jb}$:

Proposition.

$$\frac{\partial O^{ib}}{\partial A^{mn}} = \delta^i_m \delta^j_n V^{jb} = \delta^i_m V^{nb} \tag{23}$$

$$\frac{\partial O^{ib}}{\partial V^{mn}} = A^{ij} \delta^j_m \delta^b_n = A^{im} \delta^b_n \tag{24}$$

Therefore:

$$\frac{\partial L}{\partial A^{mn}} = \frac{\partial L}{\partial O^{mb}} V^{nb} \tag{25}$$

In matrix form: $\partial L / \partial A = (\partial L / \partial O) V^T$

And:

$$\frac{\partial L}{\partial V^{mn}} = A^{im} \frac{\partial L}{\partial O^{in}} \tag{26}$$

In matrix form: $\partial L / \partial V = A^T (\partial L / \partial O)$

6.3 Gradient Through Softmax

The softmax Jacobian is:

Proposition.

$$\frac{\partial A^{ij}}{\partial S^{mn}} = \delta^i_m A^{ij} (\delta^j_n - A^{in}) \tag{27}$$

The gradient through softmax becomes:

Theorem (Softmax Gradient).

$$\frac{\partial L}{\partial S^{mn}} = A^{mn} \left(\frac{\partial L}{\partial A^{mn}} - \sum_j A^{mj} \frac{\partial L}{\partial A^{mj}} \right) \tag{28}$$

In compact form:

$$\frac{\partial L}{\partial S} = A \circ \left(\frac{\partial L}{\partial A} - \text{rowsum}\left(A \circ \frac{\partial L}{\partial A} \right) \right) \tag{29}$$

where $\circ$ is elementwise multiplication.

6.4 Gradient Through Score Computation

Given $S^{ij} = \frac{1}{\sqrt{d_k}} Q^{ia} K^{ja}$:

Proposition.

$$\frac{\partial S^{ij}}{\partial Q^{kl}} = \frac{1}{\sqrt{d_k}} \delta^i_k \delta^a_l K^{ja} = \frac{1}{\sqrt{d_k}} \delta^i_k K^{jl} \tag{30}$$

$$\frac{\partial S^{ij}}{\partial K^{kl}} = \frac{1}{\sqrt{d_k}} \delta^j_k Q^{il} \tag{31}$$

Therefore:

Theorem (Query Gradient).

$$\frac{\partial L}{\partial Q^{kl}} = \frac{1}{\sqrt{d_k}} \frac{\partial L}{\partial S^{kj}} K^{jl} \tag{32}$$

In matrix form: $\partial L / \partial Q = \frac{1}{\sqrt{d_k}} \cdot (\partial L / \partial S) K$

Theorem (Key Gradient).

$$\frac{\partial L}{\partial K^{kl}} = \frac{1}{\sqrt{d_k}} \frac{\partial L}{\partial S^{ik}} Q^{il} \tag{33}$$

In matrix form: $\partial L / \partial K = \frac{1}{\sqrt{d_k}} \cdot (\partial L / \partial S)^T Q$

6.5 Complete Backward Pass

Combining all the pieces:

Theorem (Attention Backward Pass). Given upstream gradient $\partial L / \partial O$:

1. $\partial L / \partial V = A^T (\partial L / \partial O)$
2. $\partial L / \partial A = (\partial L / \partial O) V^T$
3. $\partial L / \partial S = A \circ (\partial L / \partial A - \text{rowsum}(A \circ \partial L / \partial A))$
4. $\partial L / \partial Q = \frac{1}{\sqrt{d_k}} (\partial L / \partial S) K$
5. $\partial L / \partial K = \frac{1}{\sqrt{d_k}} (\partial L / \partial S)^T Q$

7 Part VI: Multi-Head Attention

Multi-head attention runs multiple attention operations in parallel with different learned projections.

7.1 Structure

Definition (Multi-Head Attention). For H heads with projection matrices $W_Q^h, W_K^h, W_V^h, W_O^h$:

$$Q^{hia} = Q^{ib} W_Q^{hba} \tag{34}$$

$$K^{hja} = K^{jb} W_K^{hba} \tag{35}$$

$$V^{hjc} = V^{jb} W_V^{hbc} \tag{36}$$

Per-head attention:

$$S^{hij} = \frac{1}{\sqrt{d_k}} Q^{hia} K^{hja} \tag{37}$$

$$A^{hij} = \text{softmax}_j\left(S^{hij}\right) \tag{38}$$

$$O^{hic} = A^{hij} V^{hjc} \tag{39}$$

Final output (sum over heads):

$$Y^{ia} = O^{hic} W_O^{hca} \tag{40}$$

The head index h creates independent attention patterns, allowing the model to attend to different aspects of the input simultaneously.

7.2 Head Diversity

Well-trained multi-head attention should have **diverse** heads that capture different relationships. We can measure diversity using:

$$\text{diversity} = 1 - \text{avg pairwise cosine similarity between heads} \tag{41}$$

8 Part VII: Attention Variants

8.1 Causal Masking

For autoregressive models, we prevent position i from attending to future positions $j > i$:

Definition (Causal Mask).

$$M^{ij} = \begin{cases} 1 \text{ if } j \leq i \\ 0 \text{ otherwise} \end{cases} \tag{42}$$

Applied as: $S^{ij} \leftarrow S^{ij} + (1 - M^{ij}) \cdot (-\infty)$

8.2 Learned Bilinear Attention

Instead of scaled dot-product, use a learned metric:

$$S^{ij} = Q^{ia} M^{ab} K^{jb} \tag{43}$$

where M^{ab} is a learnable parameter (or $M = W^T W$ for positive definiteness).

8.3 Relative Position Attention

Add relative position information to keys:

$$S^{ij} = \frac{1}{\sqrt{d_k}} Q^{ia} \left(K^{ja} + R^{(i-j)a} \right) \tag{44}$$

where R^{ka} is a learned embedding for relative position k.

9 Part VIII: Attention as Kernel Regression

The attention mechanism can be viewed through the lens of kernel methods.

9.1 Kernel Formulation

Define a kernel:

$$K(q,k) = \exp\left(\frac{q \cdot k}{\sqrt{d_k}}\right) \tag{45}$$

Then attention weights are:

$$A^{ij} = \frac{K(q_i, k_j)}{\sum_k K(q_i, k_k)} \tag{46}$$

And the output is:

$$o_i = \frac{\sum_j K(q_i, k_j) v_j}{\sum_j K(q_i, k_j)} \tag{47}$$

This is exactly **Nadaraya-Watson kernel regression**!

9.2 Linear Attention

For efficiency, we can approximate the kernel using feature maps φ:

$$K(q,k) \approx \varphi(q)^T \varphi(k) \tag{48}$$

This allows computing attention in $O(n)$ instead of $O(n^2)$:

$$o_i = \frac{\sum_j \varphi(q_i)^T \varphi(k_j) v_j}{\sum_j \varphi(q_i)^T \varphi(k_j)} = \frac{\varphi(q_i)^T \sum_j \varphi(k_j) v_j^T}{\varphi(q_i)^T \sum_j \varphi(k_j)} \tag{49}$$

The sums can be precomputed once and reused for all queries.

10 Part IX: Hopfield Networks and Attention

A remarkable connection exists between transformer attention and modern Hopfield networks [3].

10.1 Classical Hopfield Networks

Classical Hopfield networks store patterns ξ_μ in a weight matrix:

$$W_{ij} = \frac{1}{N} \sum_{\mu} \xi_\mu^i \xi_\mu^j \tag{50}$$

The energy function is:

$$E(x) = -\frac{1}{2} x^i W_{ij} x^j \tag{51}$$

Fixed points (local minima) correspond to stored patterns.

Problem: Capacity scales only as $M \approx 0.14N$ patterns.

10.2 Modern Hopfield Networks

Modern Hopfield networks [3] use an exponential energy:

Definition (Modern Hopfield Energy).

$$E(x) = -\text{lse}(\beta \cdot \text{patterns}^T x) + \frac{1}{2} \|x\|^2 \tag{52}$$

where $\text{lse}(z) = \log \sum_\mu \exp(z_\mu)$ is the log-sum-exp (smooth maximum).

The update rule is:

Theorem (Hopfield Update = Attention).

$$x_{\text{new}} = \text{patterns}^T \cdot \text{softmax}(\beta \cdot \text{patterns} \cdot x) \tag{53}$$

This is exactly the attention mechanism with:

- Query: current state x
- Keys: stored patterns
- Values: stored patterns
- Temperature: $1/\beta$

Key insight: Capacity scales **exponentially** as $M \approx \exp(d/2)$!

11 Part X: Worked Examples

11.1 Example 1: 2-Query, 3-Key Attention

Consider:

$$Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad K = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}, \quad V = \begin{pmatrix} 2 & 0 \\ 0 & 2 \\ 1 & 1 \end{pmatrix} \tag{54}$$

Step 1: Scores ($d_k = 2$, so $\frac{1}{\sqrt{d_k}} = \frac{1}{\sqrt{2}} \approx 0.707$)

$$S = \frac{1}{\sqrt{2}} QK^T = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \approx \begin{pmatrix} 0.707 & 0 & 0.707 \\ 0 & 0.707 & 0.707 \end{pmatrix} \tag{55}$$

Step 2: Softmax (row-wise)

For row 1: $[e^{0.707}, e^0, e^{0.707}] \approx [2.03, 1, 2.03]$, sum $= 5.06$

$$A_1 \approx [0.401, 0.198, 0.401] \tag{56}$$

Row 2 is symmetric: $A_2 \approx [0.198, 0.401, 0.401]$

Step 3: Output

$$O = AV = \begin{pmatrix} 0.401 & 0.198 & 0.401 \\ 0.198 & 0.401 & 0.401 \end{pmatrix} \begin{pmatrix} 2 & 0 \\ 0 & 2 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1.203 & 0.797 \\ 0.797 & 1.203 \end{pmatrix} \tag{57}$$

11.2 Example 2: Temperature Effects

For scores $S = [2, 1, 0]$, attention weights at different temperatures:

T	**Weights**	**Entropy**
0.5	[0.88, 0.11, 0.01]	0.42
1.0	[0.67, 0.24, 0.09]	0.80
2.0	[0.47, 0.32, 0.21]	1.02
∞	[0.33, 0.33, 0.33]	1.10

12 Appendix A: Notation Reference

Symbol	Meaning
Q^{ia}	Query tensor, position i, feature a
K^{ja}	Key tensor, position j, feature a
V^{jb}	Value tensor, position j, feature b
S^{ij}	Attention scores
A^{ij}	Attention weights (probabilities)
O^{ib}	Output tensor
g_{ab}	Metric tensor
δ_b^a	Kronecker delta
Γ_{ab}^c	Christoffel symbols
Z^i	Partition function for query i
H^i	Entropy for query i
T	Temperature
β	Inverse temperature $(1/T)$

13 Appendix B: Code Verification

All derivations in this document have been verified against JAX autodiff. The code is available in the accompanying `attn_tensors` Python package:

```
from attn_tensors.gradients import verify_gradients
import jax.numpy as jnp

Q = jnp.array([[1., 0.], [0., 1.]])
K = jnp.array([[1., 0.], [0., 1.], [1., 1.]])
V = jnp.array([[2., 0.], [0., 2.], [1., 1.]])

results = verify_gradients(Q, K, V)
print(results)  # {'dL_dQ': True, 'dL_dK': True, 'dL_dV': True, 'all_correct': True}
```

Bibliography

[1] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[2] A. Vaswani *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[3] H. Ramsauer *et al.*, "Hopfield networks is all you need," *arXiv preprint arXiv:2008.02217*, 2020.