Self-Attention

Scaled Dot-Product Attention Analysis

Visualizations, Scaling Analysis, and Gradient Flow

```
Attention(Q, K, V) = softmax(QK^T / sqrt(d_k)) V
```

SEED = 42 | NumPy-only implementation

From-Scratch ML Implementations

Summary of Findings

1. Attention weights form an (n x n) matrix where each row sums to 1,
   representing a probability distribution over key positions.

2. Causal masking produces a lower-triangular attention pattern,
   preventing positions from attending to future tokens.

3. Scaling by sqrt(d_k) is essential: without it, large d_k causes
   softmax saturation (near-binary weights with vanishing gradients).

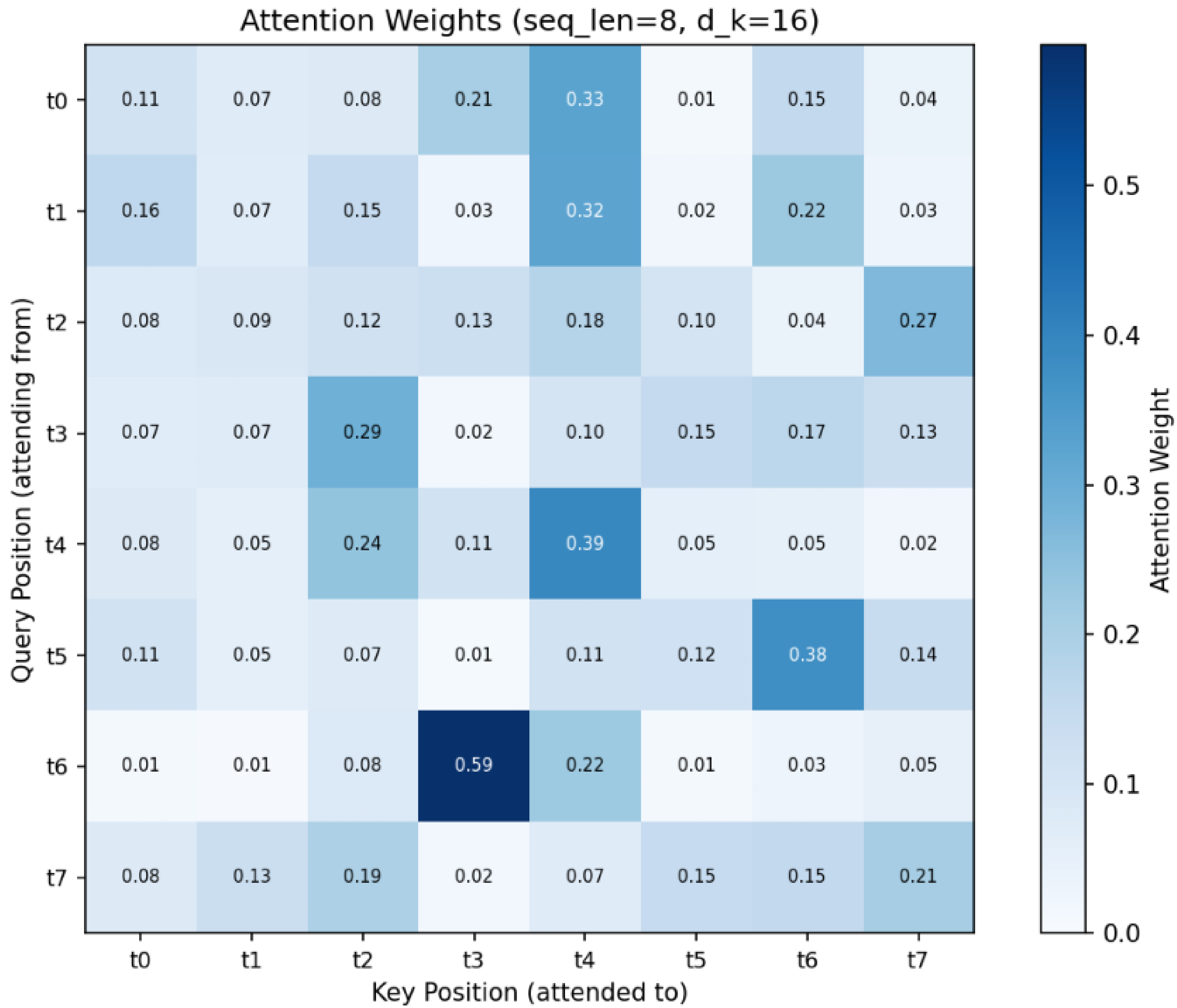
4. The attention matrix dominates memory at long sequences:
   at n=4096, it accounts for >98% of activation memory.

5. Compute is O(n^2 * d_k) for the attention core. At long sequences,
   attention core FLOPs dominate over linear projection FLOPs.

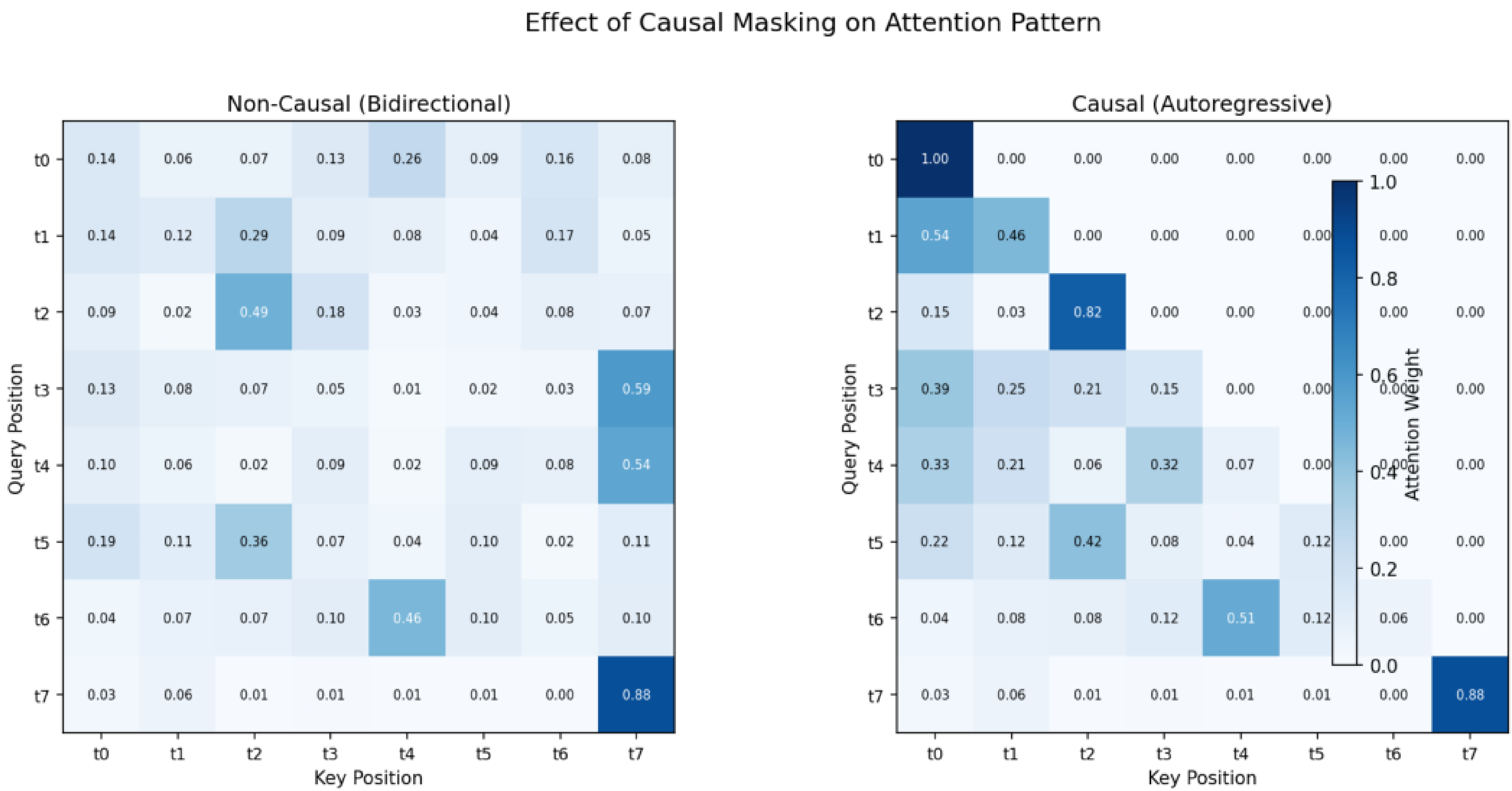
6. Gradient flow through attention is well-behaved with proper scaling.
   Causal masking causes asymmetric gradients across positions:
   earlier positions receive contributions from more downstream tokens.

7. These O(n^2) costs motivate Flash Attention (tiled computation),
   KV caching (avoid recomputing K/V), and GQA/MQA (share K/V heads).

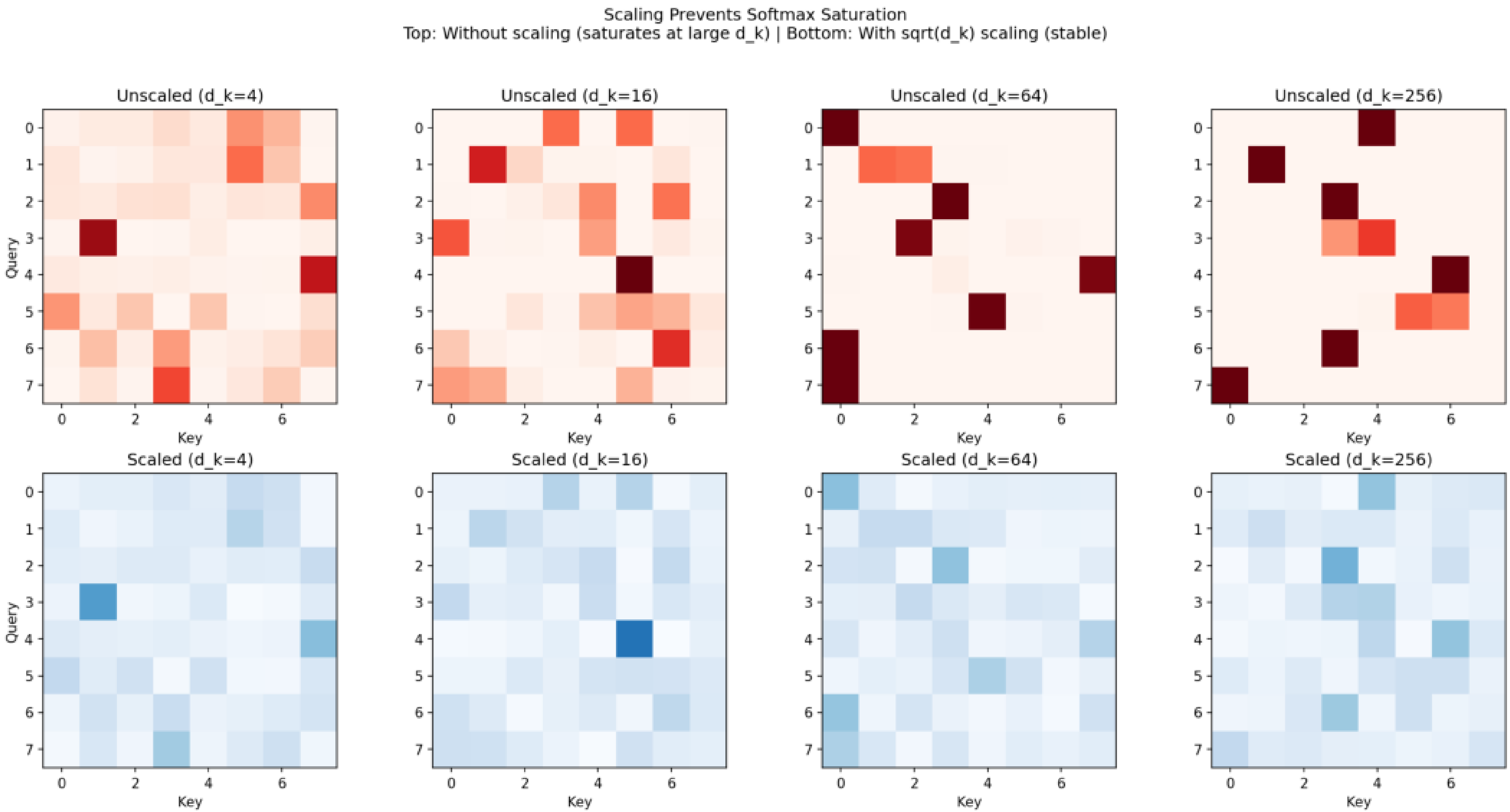
Attention Weights Heatmap



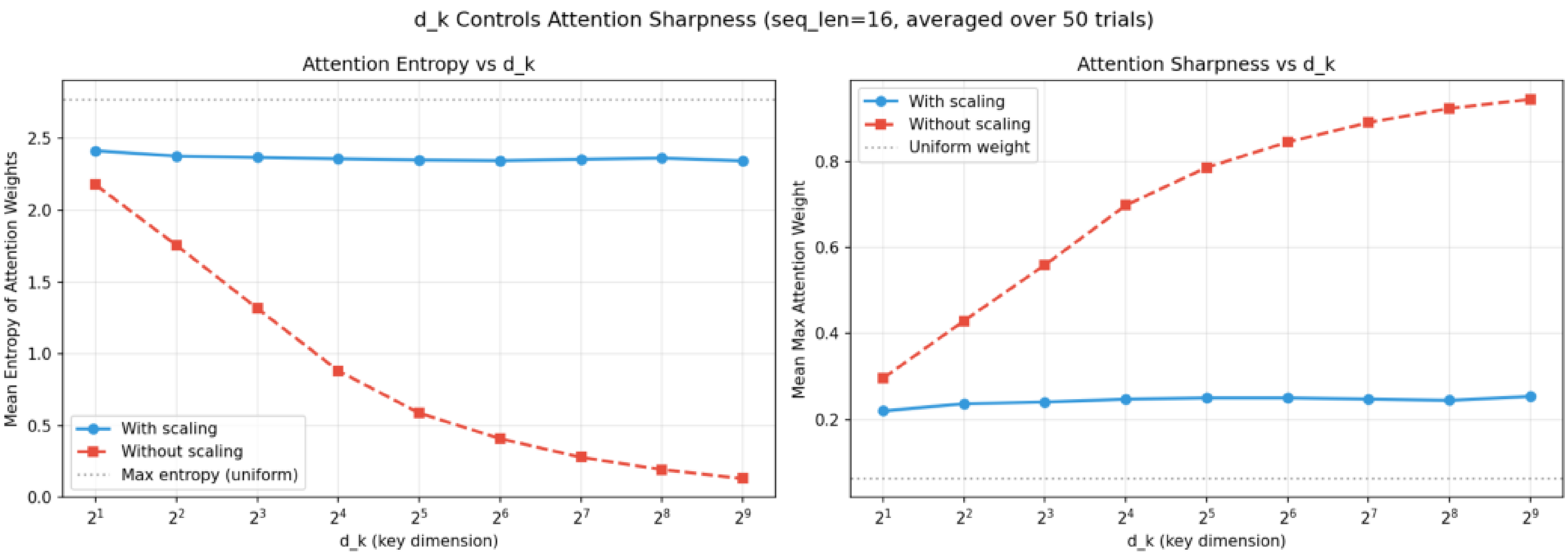
Causal vs Non-Causal Attention



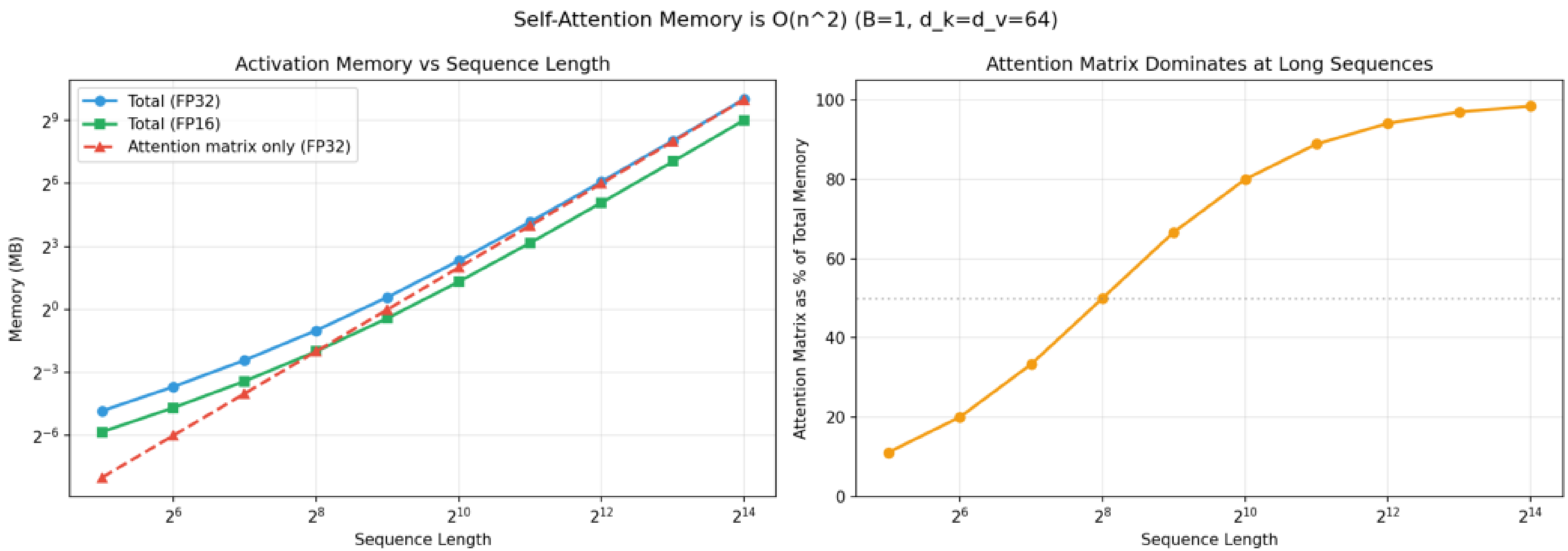
Effect of Scaling by sqrt(d_k)



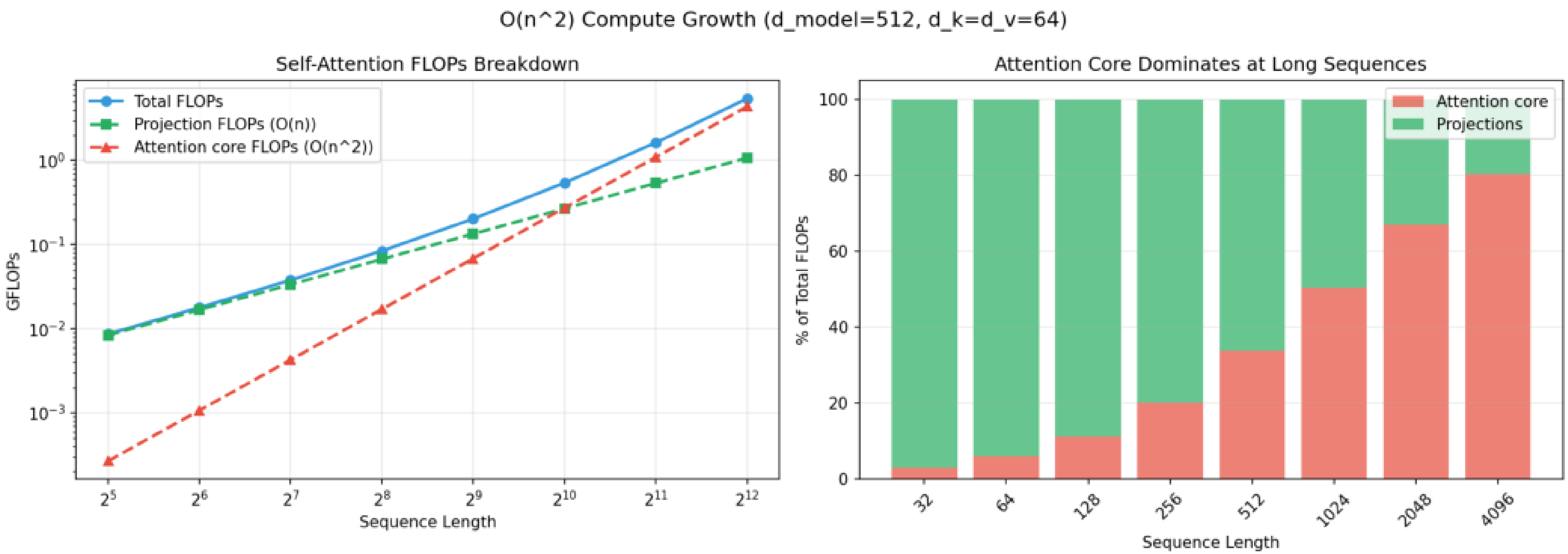
Attention Sharpness vs d_k Dimension



Quadratic Memory Scaling O(n^2)



FLOP Analysis and O(n^2) Growth



Gradient Flow Through Self-Attention