Contents

Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation (ALiBi)

Paper Link: https://arxiv.org/abs/2108.12409

Authors: Ofir Press, Noah A. Smith, Mike Lewis (University of Washington, Facebook AI Research, Allen Institute for AI)

Publication: ICLR 2022

Code: https://github.com/ofirpress/attention_with_linear_biases

---

Overview

ALiBi (Attention with Linear Biases) introduces a remarkably simple yet effective approach to position encoding that enables transformers to extrapolate to much longer sequences than they were trained on. Instead of adding positional embeddings to input tokens, ALiBi directly biases attention scores with a penalty proportional to the distance between tokens, creating an inductive bias toward recency.

Key Problem Addressed

Traditional position encoding methods face significant limitations:

1. **Length Extrapolation**: Models cannot handle sequences longer than training length
2. **Memory Requirements**: Positional embeddings consume additional memory
3. **Computational Overhead**: Position encoding adds computational cost
4. **Fixed Context**: Models struggle with variable sequence lengths

The fundamental question addressed: **How can a model achieve extrapolation at inference time for sequences longer than it saw during training?**

Core Innovation: Linear Bias Mechanism

ALiBi replaces traditional positional embeddings with a simple linear bias applied directly to attention scores:

Mathematical Formulation

Standard attention:

`softmax(q_i * k_j / √d)`

ALiBi attention:

`softmax(q_i * k_j / √d - m * |i - j|)`

Where: - `m` is a head-specific slope (negative value) - `|i - j|` is the distance between positions i and j - The bias `-m * |i - j|` penalizes distant tokens

Key Properties

1. Distance Penalty

- Attention scores decrease linearly with distance
- Closer tokens receive higher attention weights
- Farther tokens are naturally down-weighted

2. Head-Specific Slopes Different attention heads use different slopes:

`m_h = 1 / 2^(8h/H)`

Where H is the total number of heads, h is the head index.

3. No Learned Parameters

- Slopes are fixed, not learned during training
- Reduces parameter count compared to positional embeddings
- Eliminates need for position-specific parameters

Implementation Details

Attention Computation

```
def alibi_attention(q, k, v, slopes):
    # q, k, v: [batch, heads, seq_len, dim]
    # slopes: [heads] - head-specific slopes

    # Standard attention scores
    scores = torch.matmul(q, k.transpose(-2, -1)) / math.sqrt(dim)

    # Create distance matrix
    seq_len = q.size(-2)
    distance = torch.abs(torch.arange(seq_len)[:, None] - torch.arange(seq_len)[None, :])
    # Apply linear bias
    bias = slopes[:, None, None] * distance
    scores = scores - bias

    # Apply softmax
    attn = torch.softmax(scores, dim=-1)
    return torch.matmul(attn, v)
```

Slope Generation

```
def get_slopes(num_heads):
    # Generate slopes for each head
    slopes = []
    for i in range(num_heads):
        slope = 1.0 / (2 ** (8 * i / num_heads))
        slopes.append(-slope)  # Negative for penalty
    return torch.tensor(slopes)
```

Experimental Results

Length Extrapolation

- **Training Length**: 1024 tokens
- **Test Length**: 2048 tokens
- **Model Size**: 1.3B parameters
- **Performance**: Same perplexity as sinusoidal model trained on 2048 tokens

Efficiency Gains

- **Training Speed**: 11% faster than sinusoidal position embedding
- **Memory Usage**: 11% less memory consumption
- **Inference Speed**: No additional computational overhead

Benchmark Performance

- **WikiText-103**: Outperformed multiple strong position methods
- **Length Generalization**: Consistent performance across different sequence lengths
- **Multiple Scales**: Effective across different model sizes

Comparison with Other Methods

vs. Sinusoidal Position Embedding

- **Simplicity**: No position embeddings needed
- **Efficiency**: Faster training and inference
- **Extrapolation**: Better length generalization
- **Memory**: Lower memory requirements

vs. Learned Position Embedding

- **Parameter Count**: Fewer parameters (no position embeddings)

- **Flexibility**: Works with arbitrary sequence lengths
- **Generalization**: Better unseen length handling
- **Training**: More stable training dynamics

vs. Relative Position Encoding

- **Implementation**: Much simpler to implement
- **Computation**: Lower computational overhead
- **Scalability**: Linear complexity with sequence length
- **Effectiveness**: Comparable or better performance

Theoretical Analysis

Inductive Bias

ALiBi creates a strong inductive bias toward recency: - Recent tokens get higher attention weights - Distant tokens are systematically down-weighted - Natural modeling of language locality

Length Extrapolation Mechanism

The linear bias enables extrapolation because: - Penalty is proportional to distance - No fixed maximum sequence length - Consistent behavior across different lengths - Smooth degradation for very long sequences

Attention Pattern Analysis

ALiBi produces attention patterns that: - Focus on local context - Gradually decrease attention with distance - Maintain consistent behavior across sequence lengths - Preserve important long-range dependencies

Practical Advantages

1. Implementation Simplicity

```
# ALiBi is extremely simple to implement
attention_scores = attention_scores - slopes * distance_matrix
```

2. Memory Efficiency

- **No Position Embeddings**: Eliminates position embedding parameters
- **Reduced Memory**: Lower memory footprint during training
- **Scalability**: Memory usage doesn't grow with max sequence length

3. Training Efficiency

- **Faster Training**: 11% speed improvement
- **Stable Training**: More stable training dynamics
- **Better Convergence**: Improved convergence properties

4. Inference Flexibility

- **Arbitrary Lengths**: Handle any sequence length at inference
- **No Retraining**: Works immediately on longer sequences
- **Consistent Performance**: Maintains quality across lengths

Limitations and Considerations

1. Fixed Slopes

- Slopes are predefined, not learned
- May not be optimal for all tasks
- Limited adaptability to different domains

2. Linear Bias Assumption

- Assumes linear relationship between distance and relevance
- May not capture complex positional relationships
- Could be suboptimal for tasks requiring long-range precision

3. Recency Bias

- Strong bias toward recent tokens
- May hurt tasks requiring equal attention to all positions
- Could limit performance on some structured tasks

Impact on the Field

1. Widespread Adoption

- **Language Models**: Adopted in many modern LLMs
- **Open Source**: Integrated into popular frameworks
- **Production**: Used in real-world applications

2. Research Influence

- **Simplicity Principle**: Demonstrated power of simple solutions
- **Length Extrapolation**: Opened new research directions
- **Efficiency Focus**: Influenced efficiency-oriented research

3. Practical Applications

- **Long Document Processing**: Enabled longer context handling
- **Conversational AI**: Better handling of long conversations
- **Code Generation**: Improved performance on long code sequences

Key Insights

1. Simplicity Effectiveness

Simple linear bias can be more effective than complex position encoding schemes.

2. Distance-Based Attention

Direct penalization of distant tokens creates natural attention patterns.

3. Extrapolation Capability

Linear bias enables smooth extrapolation to unseen sequence lengths.

4. Efficiency Benefits

Removing position embeddings provides both speed and memory benefits.

Why Read This Paper

ALiBi is essential reading because:

1. **Practical Impact**: Enables better length handling in real applications
2. **Elegant Simplicity**: Demonstrates power of simple, principled approaches
3. **Efficiency Gains**: Provides concrete speed and memory improvements
4. **Foundational Method**: Widely adopted in modern language models
5. **Research Methodology**: Exemplifies rigorous experimental validation

Key Takeaways

1. **Linear Bias Works**: Simple distance-based penalties are highly effective
2. **Less is More**: Removing position embeddings can improve performance
3. **Extrapolation Possible**: Models can handle longer sequences than training data
4. **Efficiency Matters**: Simple methods often provide better efficiency
5. **Inductive Bias**: Recency bias is powerful for language modeling

ALiBi represents a paradigm shift toward simpler, more efficient position encoding methods that provide better length generalization capabilities. Its widespread adoption in modern language models demonstrates the practical value of elegant, theoretically grounded approaches to fundamental problems in transformer architecture design.