Data-Driven, Dynamic, Online, and Attention-Based Activation Functions for Deep Neural Networks

Shyamal Suhana Chandra

2025

Abstract

This paper presents a comprehensive framework for implementing data-driven, dynamic, online, and attention-based activation functions across multiple neural network architectures including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), Gated Recurrent Units (GRUs), Transformers, Hierarchical Transformers, Big Bird, and Mixture of Experts (MoE) models. Our implementation provides adaptive activation mechanisms that improve upon traditional fixed activation functions by incorporating data statistics, temporal dynamics, online learning capabilities, and attention mechanisms.

1 Introduction

Activation functions are fundamental components of neural networks, introducing non-linearity and enabling complex function approximation. Traditional activation functions such as ReLU, sigmoid, and tanh are static and do not adapt to the data distribution or temporal dynamics of the learning process. This work introduces four novel activation function paradigms:

- **Data-Driven Activations**: Adapt based on input data statistics
- **Dynamic Activations**: Evolve parameters over time during training
- **Online Activations**: Adapt in real-time to streaming data
- **Attention-Based Activations**: Use attention mechanisms to weight activations

2 Methodology

2.1 Data-Driven Activation Functions

Data-driven activation functions adapt their behavior based on the statistical properties of the input data. The activation function maintains running statistics (mean and variance) and uses adaptive weights to combine base activations with data-adaptive components:

$$f(x) = \alpha \cdot \text{GELU}(\bar{x}) + \beta \cdot w \cdot \text{Swish}(\bar{x}) \tag{1}$$

where $\bar{x} = \frac{x-\mu}{\sigma}$ is the normalized input, μ and σ are running statistics, and w are adaptive weights.

2.2 Dynamic Activation Functions

Dynamic activation functions evolve their parameters over time using momentum-based updates:

$$\theta_{t+1} = \theta_t + v_{t+1} \tag{2}$$

$$v_{t+1} = \gamma v_t + \eta \nabla_\theta \mathcal{L} \tag{3}$$

where γ is the decay rate and η is the update rate.

2.3 Online Activation Functions

Online activation functions adapt to streaming data using exponential moving averages and a sliding window buffer:

$$\mu_t = \lambda \mu_{t-1} + (1 - \lambda) x_t \tag{4}$$

$$\sigma_t^2 = \lambda \sigma_{t-1}^2 + (1 - \lambda)(x_t - \mu_t)^2 \tag{5}$$

where λ is the forgetting factor.

2.4 Attention-Based Activation Functions

Attention-based activations use multi-head attention to weight activation outputs:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{6}$$

$$f(x) = \alpha \cdot \text{GELU}(x) + \beta \cdot \text{Swish}(x \cdot \text{Attention}(x)) \tag{7}$$

3 Architecture-Specific Implementations

3.1 Convolutional Neural Networks

For CNNs, activations are applied to feature maps with spatial dimensions. The implementation supports channel-wise and spatial adaptive activations.

3.2 Recurrent Neural Networks

RNN implementations maintain hidden states and apply activations to the combined input and hidden state vectors.

3.3 LSTM and GRU

LSTM and GRU implementations integrate activations with gate mechanisms, applying different activation functions to gates and cell states.

3.4 Transformers

Transformer implementations apply activations in both self-attention blocks and feed-forward networks, with separate contexts for each component.

3.5 Hierarchical Transformers

Hierarchical transformers apply activations at multiple levels of abstraction, with each level using its own activation context.

3.6 Big Bird

Big Bird implementations use separate activation contexts for window attention, global attention, and random attention patterns.

3.7 Mixture of Experts

MoE implementations use a router to select and combine activations from multiple expert networks, each with its own activation function.

4 Implementation Details

The implementation is written in C/C++ for performance and portability. Key features include:

- Memory-efficient pooling for temporary allocations
- Modular architecture supporting multiple activation types
- Efficient forward and backward pass implementations
- Support for gradient computation

5 Experimental Results

[Experimental results and benchmarks would be included here in a complete paper]

6 Conclusion

We have presented a comprehensive framework for adaptive activation functions that can be applied across multiple neural network architectures. The implementation provides data-driven, dynamic, online, and attention-based activation mechanisms that improve upon traditional fixed activations.

7 Acknowledgments

This work was developed by Shyamal Suhana Chandra.

References

[1] Hendrycks, D., & Gimpel, K. (2016). Gaussian Error Linear Units (GELUs). arXiv preprint arXiv:1606.08415.

[2] Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. arXiv preprint arXiv:1710.05941.

[3] Vaswani, A., et al. (2017). Attention is all you need. Advances in neural information processing systems, 30.

[4] Zaheer, M., et al. (2020). Big bird: Transformers for longer sequences. Advances in Neural Information Processing Systems, 33.

[5] Shazeer, N., et al. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. arXiv preprint arXiv:1701.06538.