Data-Driven, Dynamic, Online, and Attention-Based Activation Functions

Implementation for CNNs, RNNs, LSTMs, GRUs, Transformers, and Beyond

Shyamal Suhana Chandra

2025

Overview

- Novel activation function paradigms for deep learning
- Four activation types: Data-Driven, Dynamic, Online, Attention-Based
- Support for multiple architectures: CNN, RNN, LSTM, GRU, Transformer, Hierarchical Transformer, Big Bird, MoE
- High-performance C/C++ implementation

Motivation

- Traditional activations (ReLU, sigmoid, tanh) are static
- Fixed activations don't adapt to data distribution
- Need for adaptive mechanisms that learn from data
- Attention mechanisms can improve activation quality

Data-Driven Activations

Key Features

- Adapt based on input statistics (mean, variance)
- Maintain running statistics with momentum
- Use adaptive weights to combine base and data-adaptive components
- Formula: $f(x) = \alpha \cdot \text{GELU}(\bar{x}) + \beta \cdot w \cdot \text{Swish}(\bar{x})$

Dynamic Activations

Key Features

- Parameters evolve over time during training
- Momentum-based updates: $v_{t+1} = \gamma v_t + \eta \nabla_\theta \mathcal{L}$
- Bounded parameter space for stability
- Combines multiple activation functions dynamically

Online Activations

Key Features

- Real-time adaptation to streaming data
- Exponential moving averages: $\mu_t = \lambda \mu_{t-1} + (1 - \lambda) x_t$
- Sliding window buffer for recent samples
- Forgetting factor controls adaptation rate

Attention-Based Activations

Key Features

- Multi-head attention mechanism
- Query-Key-Value attention: $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$
- Attention-weighted activation combination
- Temperature scaling for attention sharpness

Architecture Support

- **CNNs**: Channel and spatial adaptivity
- **RNNs**: Hidden state integration
- **LSTMs**: Gate-aware activations
- **GRUs**: Reset and update gates
- **Transformers**: Self-attention blocks
- **Hierarchical**: Multi-level activations
- **Big Bird**: Sparse attention patterns
- **MoE**: Expert-specific activations

Implementation Highlights

- **Language**: C/C++ for performance
- **Build System**: CMake for cross-platform support
- **Memory**: Efficient pooling for temporary allocations
- **Modularity**: Clean separation of concerns
- **Gradients**: Full backward pass support

Code Structure

Core Components

- `core/`: Base activation implementations
- `architectures/`: Architecture-specific wrappers
- `include/`: Public API headers
- `examples/`: Usage demonstrations

Usage Example

CNN with Data-Driven Activation

```
ddaf_context_t* ctx =
    ddaf_create_context(DDAF_TYPE_DATA_DRIVEN,
                        DDAF_ARCH_CNN, 0);
ddaf_cnn_init(ctx, 64, 32, 32);
ddaf_forward(ctx, input, output, size);
ddaf_backward(ctx, grad_output, grad_input, size);
```

Benefits

- **Adaptability**: Functions adapt to data characteristics
- **Performance**: Efficient C/C++ implementation
- **Flexibility**: Multiple activation types and architectures
- **Extensibility**: Easy to add new activation types

Future Work

- Benchmarking on standard datasets
- Optimization for specific hardware (GPU, TPU)
- Additional activation function variants
- Integration with popular deep learning frameworks

Conclusion

- Comprehensive framework for adaptive activations
- Support for major neural network architectures
- High-performance implementation
- Open for extension and improvement

Questions?

Thank you for your attention!