Smart HVAC Systems: RL Approach

Aniket Dixit
PhD Candidate, Computer Science
Tech University
June 15, 2025

Advisor: Prof. James Brusey

Committee: Prof. James Brusey, Prof. Matthew England

Research Overview

- **Problem**: Current sequential models struggle with long-range dependencies
- **Approach**: Novel attention mechanism with adaptive memory
- **Contribution**: 15% improvement over transformer baselines
- **Impact**: Applications in machine translation, summarization, and dialogue
- **Validation**: Comprehensive evaluation across 8 benchmark datasets

Core Innovation: Dynamic attention weights that adapt based on sequence complexity and context relevance

Problem Statement

Sequential Data Processing Challenges

Current Limitations

- **Computational Complexity**: Standard attention scales quadratically with sequence length
- **Memory Constraints**: Limited ability to maintain long-term context
- **Training Instability**: Gradient vanishing in very deep architectures
- **Domain Adaptation**: Poor generalization across different text domains

Research Questions

- How can we design attention mechanisms that scale efficiently?
- What architectural changes improve long-range dependency modeling?
- Can we achieve better performance with fewer parameters?

Hypothesis: Adaptive attention with hierarchical memory structures can overcome current limitations while maintaining computational efficiency

Reward Function

We model sequential data processing as learning a mapping function:

Objective Function:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \ell(f_\theta(X^{(i)}), Y^{(i)}) + \lambda\Omega(\theta)$$

Where f_θ represents our proposed attention-based architecture, ℓ is the loss function, and $\Omega(\theta)$ is a regularization term.

Methodology

Proposed Architecture: AdaptiveNet

Key Components

- **Multi-Head Adaptive Attention**: Dynamic attention weights
- **Hierarchical Memory Module**: Long-term context storage
- **Residual Gating**: Improved gradient flow
- **Layer-wise Learning Rates**: Optimized training dynamics

AdaptiveNet Architecture
[Insert detailed network diagra]

Innovation Details

The adaptive attention mechanism computes context-dependent weights:

$$\alpha_{ij} = \frac{\exp(e_{ij} \cdot \mathrm{adapt}(c_i))}{\sum_{k=1}^{n} \exp(e_{ik} \cdot \mathrm{adapt}(c_i))}$$

Training Algorithm

Algorithm 1: AdaptiveNet Training

Input: Training dataset $\mathcal{D}$, model parameters θ_0, learning rate η
Output: Optimized parameters θ^*

1. **Initialize** model parameters θ_0 and memory states M_0
2. **for** epoch = 1 to max_epochs **do**
3. &ention}(X, M)$
   ard pass: $\hat{Y} = \text{AdaptiveNet}(X, A,
4. **end for**

Theoretical Analysis

Theorem 1: Convergence Guarantee

Under standard assumptions of bounded gradients and Lipschitz continuity, the AdaptiveNet training algorithm converges to a stationary point with probability 1.

Proof Sketch:

1. Show that the adaptive attention preserves the contraction property
2. Apply stochastic approximation theory for the memory update rule
3. Use martingale convergence theorem for the parameter updates

Complexity Analysis

- **Time Complexity**: $O(n \log n)$ for sequence length n (vs. $O(n^2)$ for standard attention)
- **Space Complexity**: $O(n + m)$ where m is memory size
- **Parameter Efficiency**: 23% fewer parameters than comparable transformer models

Experimental Results

Experimental Setup

Datasets

- **Machine Translation**: WMT14 En-De, En-Fr
- **Text Summarization**: CNN/DailyMail, XSum
- **Question Answering**: SQuAD 2.0, Natural Questions
- **Dialogue**: PersonaChat, MultiWOZ

Baseline Models

- **Transformer-Base**: Standard attention mechanism
- **Linformer**: Linear attention approximation
- **Performer**: Fast attention via random features
- **Longformer**: Sparse attention patterns

Experimental Pipeline [Insert flowchart showing data preprocessing, model training, evaluation metrics, and statistical testing procedures]

Main Results

Performance Comparison Across Tasks
[Figure]

Key Findings

TASK	ADAPTIVENET	TRANSFORMER	IMPROVEMENT
Translation (BLEU)	**34.2**	29.8	**+14.8%**
Summarization (ROUGE-L)	**42.1**	38.7	**+8.8%**
QA (F1)	**89.3**	85.2	**+4.8%**
Dialogue (BLEU)	**28.6**	24.1	**+18.7%**

Ablation Study

Component Analysis

COMPONENT	PERFORMANCE IMPACT	STATISTICAL SIGNIFICANCE
Adaptive Attention	**+12.3%**	$p < 0.001$
Hierarchical Memory	+8.7%	$p < 0.001$
Residual Gating	+4.2%	$p < 0.01$
Layer-wise LR	**+2.8%**	$p < 0.05$

Critical Finding: The adaptive attention mechanism provides the largest performance gain, with hierarchical memory being the second most important component

Computational Efficiency

Speed and Memory Comparison [Insert line graphs showing training time, inference speed, and memory usage across different sequence lengths for all baseline models]

Qualitative Analysis

Attention Visualization

Attention Patterns [Insert heatmaps showing attention weights for sample sentences, demonstrating long-range dependencies and adaptive behavior]

Example Outputs

Translation Quality:

- **Our Model**: "This is a very complex scientific article"
- **Baseline**: "This is a complex scientific paper"

Key Improvements:

- Better preservation of semantic meaning
- More accurate handling of technical terms
- Improved coherence in long documents

Error Analysis

Failure Cases

- **Very Short Sequences** (<10 tokens): Adaptive mechanism adds unnecessary overhead
- **Highly Repetitive Text**: Memory module can get stuck in local patterns
- **Code-Switching**: Limited training data for multilingual scenarios

Robustness Testing

PERTURBATION TYPE	PERFORMANCE DROP	RECOVERY METHOD
Noise Injection	-8.2%	Data augmentation
Domain Shift	-12.5%	Few-shot adaptation
Adversarial	-15.3%	Adversarial training

Contributions & Impact

Primary Contributions

Technical Innovations

- **Novel Architecture**: First adaptive attention mechanism with hierarchical memory
- **Theoretical Framework**: Convergence guarantees for adaptive training dynamics

Scientific Impact

- **Performance**: State-of-the-art results on 6 out of 8 benchmark datasets
- **Efficiency**: 40% faster training, 60% faster inference than baselines

Broader Impact: This work enables deployment of advanced NLP models in resource-constrained environments while achieving superior performance

Future Directions

Short-term Goals (6-12 months)

- **Multimodal Extension**: Adapt architecture for vision-language tasks
- **Few-shot Learning**: Improve performance with limited training data
- **Hardware Optimization**: Develop GPU-optimized implementations

Long-term Vision (2-5 years)

- **Foundation Models**: Scale to billion-parameter models
- **Real-time Applications**: Deploy in production systems
- **Cross-lingual Transfer**: Universal language understanding

Collaboration Opportunities

- **Industry Partnerships**: Google, Microsoft, Meta AI research teams
- **Academic Networks**: Stanford HAI, MIT CSAIL, CMU LTI
- **Open Science**: Contributing to Hugging Face, PyTorch ecosystem

References

1. **Vaswani, A., et al.** (2017). Attention is All You Need. *Neural Information Processing Systems*, 5998-6008.
2. **Devlin, J., et al.** (2019). BERT: Pre-training of Deep Bidirectional Transformers. *NAACL-HLT*, 4171-4186.
3. **Wang, S., et al.** (2020). Linformer: Self-Attention with Linear Complexity. *arXiv preprint arXiv:2006.04768*.
4. **Choromanski, K., et al.** (2021). Rethinking Attention with Performers. *ICLR*.
5. **Beltagy, I., et al.** (2020). Longformer: The Long-Document Transformer. *arXiv preprint arXiv:2004.05150*.
6. **Brown, T., et al.** (2020). Language Models are Few-Shot Learners. *Neural Information Processing Systems*, 1877-1901.

Thank You

Questions & Discussion

Contact Information:

✉️ jane.smith@techuniversity.edu

🐦 @janesmith_ai

🌐 janesmith-research.com

Resources:

📁 Code: github.com/janesmith/adaptivenet

📄 Paper: arxiv.org/abs/2025.54321

🎥 Demo: adaptivenet-demo.com

Appendix: Implementation Details

Software Architecture

Core Dependencies

- **Framework**: PyTorch 2.0 + Transformers 4.28
- **Optimization**: AdamW with cosine scheduling
- **Distributed Training**: DeepSpeed ZeRO Stage 2

Hardware Configuration

- **Training**: 8× NVIDIA A100 (80GB) GPUs
- **Evaluation**: Single V100 (32GB) GPU
- **Storage**: High-speed NVMe SSD array

```
Hyperparameter Configuration:

Learning Rate: 1e-4 (adaptive attention), 5e-5 (other components)
Batch Size: 32 per GPU (256 total)
Sequence Length: 512 (training), 1024 (evaluation)
Memory Size: 256 slots
Attention Heads: 16
Hidden Dimension: 768
Dropout: 0.1
Weight Decay: 0.01
```

Appendix: Extended Results

Detailed Performance Analysis [Insert comprehensive results table with confidence intervals, statistical significance tests, and cross-validation scores for all experiments]

Statistical Analysis

- **Sample Sizes**: 10 random seeds × 3 dataset splits × 5 cross-validation folds
- **Significance Testing**: Paired t-tests with Bonferroni correction
- **Effect Sizes**: Cohen's d > 0.8 for all major improvements
- **Confidence Intervals**: 95% bootstrapped intervals reported

Computational Profiling

OPERATION	TIME (MS)	MEMORY (MB)	GPU UTILIZATION (%)
Forward Pass	12.3	2,840	87%
Backward Pass	18.7	3,120	92%
Memory Update	3.2	480	45%
Attention Computation	8.9	1,760	78%

Thank You

Questions & Discussion

Contact Information:

✉️ [your.email@university.edu]

🐦 [@your_twitter_handle]

🌐 [your-website.com]

Resources:

📂 Code: [github.com/username/repo]

📄 Paper: [arxiv.org/abs/xxxx.xxxxx]

🎥 Demo: [project-website.com]