Augmenting Optimizers with NARS: The NARS Learning Rate Scheduler

Hirsa Kia

May 2025

Abstract

We introduce the Non-Axiomatic Reasoning System (NARS) Learning Rate Scheduler, a novel approach that dynamically optimizes step sizes in gradient-based learning algorithms by bridging symbolic reasoning with numerical optimization. Our methodology implements Non-Axiomatic Logic, a framework designed for reasoning under uncertainty and incomplete knowledge, to continuously evaluate training metrics through a belief system with frequency-confidence truth values. Unlike conventional schedulers following predetermined patterns, our system demonstrates adaptive phase-specific behavior, deploying distinct strategies during exploration, exploitation, and fine-tuning stages of training. The scheduler processes gradient magnitudes, loss trajectories, and validation performance through formal syllogistic inference to derive optimal learning rate adjustments. Empirical evaluations across diverse optimization landscapes, the Rosenbrock function, Fashion MNIST, and CIFAR-10 datasets, demonstrate performance improvements, with loss value gain of 75% on Rosenbrock optimization and accuracy gains of 3.17% on Fashion MNIST and 1.12% on CIFAR-10. The scheduler's explicit reasoning mechanisms also provides interpretability compared to black-box adaptation techniques. This work establishes a conceptual bridge between symbolic reasoning and statistical optimization, opening promising research directions in interpretable, adaptive optimization for neural networks.

1 Introduction

The Non-Axiomatic Reasoning System (NARS) exhibits remarkable capabilities in deriving logical relationships between statements through experience-based learning Wang (2019). Despite these strengths, applying NARS to quantitative domains presents fundamental challenges: (1) its reasoning architecture prioritizes qualitative symbolic inference over numerical computation, and (2) it operates natively on symbolic propositions rather than continuous mathematical representations Wang (2006, 2013). This research addresses these limitations by developing a principled interface between NARS and mathematical optimization, a domain with broad applications across machine learning, engineering, and computational science Boyd und Vandenberghe (2004). To our knowledge, this constitutes the first systematic integration of Non-Axiomatic Logic with numerical optimization frameworks.

1.1 The Landscape of Optimization Methods

Contemporary optimization problems bifurcate into convex and non-convex domains. Convex optimization presents favorable mathematical properties, including guaranteed global optima, absence of deceptive local minima, and polynomial-time solvability for many formulations. In contrast, non-convex optimization introduces substantial challenges: exponential proliferation of local minima with increasing dimensionality, prevalence of saddle points in high-dimensional parameter spaces, and absence of universal solution guarantees Boyd und Vandenberghe (2004).

First-order methods have emerged as the predominant approach for addressing complex optimization challenges due to their computational efficiency (linear complexity per iteration), minimal assumptions (requiring only differentiability), and scalability to high-dimensional spaces Ruder (2016). These methods typically follow the canonical update rule:

$$\theta_{t+1} = \theta_t + \alpha_t d_t \tag{1}$$

where $\theta_t \in \mathbb{R}^n$ represents the parameter vector, d_t denotes the search direction (typically derived from gradient information), and α_t controls the step size. The convergence properties of this iterative process depend critically on the interplay between direction selection (d_t) and step size adaptation (α_t), with sub-optimal choices resulting in oscillatory behavior or slow convergence rates Goodfellow u. a. (2016).

1.2 Bridging Symbolic Reasoning and Numerical Optimization

NARS provides a unique framework for integrating symbolic reasoning with numerical optimization, sharing conceptual parallels with reinforcement learning through their emphasis on sequential decision-making, experience-driven adaptation, and resource-constrained rationality Wang (2013, 2006). While reinforcement learning has traditionally excelled in discrete action spaces such as game-playing or robotic control, recent advances have extended its application to adaptive optimization for neural network training, hyperparameter tuning, and black-box optimization problems Li u. a. (2017, 2010). This research positions NARS as a meta-reasoning layer that augments numerical optimization processes. By leveraging NARS's capabilities in dynamic priority management, context-sensitive rule application, and uncertain evidence integration, our approach enhances optimization through adaptive step size adjustment and flexible convergence criteria Wang (2019, 2013). This synergistic integration of symbolic reasoning with numerical methods enables more robust and efficient optimization, particularly in environments characterized by uncertainty, non-stationarity, and complex loss landscapes.

Contributions and Significance: This work makes several notable contributions to the fields of optimization and artificial intelligence. First, we demonstrate that symbolic reasoning systems can effectively guide numerical optimization processes through a novel interface between NARS and gradient-based learning. Second, we introduce a phase-aware learning rate adaptation mechanism that dynamically adjusts optimization strategies based on training context, outperforming conventional schedulers on benchmark tasks. Our empirical evaluation shows substantial improvements over baseline methods, with accuracy gains of 3.17% over OneCycle and 7.37% over fixed-rate approaches on Fashion MNIST, along with a 2.87% gain over OneCycle and 1.12% gain over fixed-rate approaches on the more complex CIFAR-10 dataset. The significance of this research extends beyond immediate performance improvements. By establishing a formal bridge between symbolic reasoning and numerical optimization, we open new avenues for interpretable, adaptive learning algorithms that combine the strengths of both paradigms. This integration addresses long-standing challenges in optimization, particularly for complex non-convex problems where traditional methods often struggle with plateaus, saddle points, and deceptive local minima. The explicit reasoning mechanisms of our approach provide transparency into the decision-making process, enabling practitioners to understand why particular learning rate adjustments were made, a significant advantage over black-box adaptation techniques.

Paper Organization: The remainder of this paper is organized as follows: Section 2 compares our approach with traditional learning rate schedulers, highlighting key conceptual and operational differences. Section 3 details the NARS Learning Rate Scheduler architecture, including its belief system, inference mechanisms, and adaptation strategies. Section 4 presents experimental results on the Rosenbrock function, Fashion MNIST and CIFAR10 dataset, demonstrating the scheduler's effectiveness across different optimization domains. Section 5 discusses conceptual connections between NARS and reinforcement learning, along with future research directions. Finally, Section 6 concludes with a summary of our findings and broader implications for adaptive optimization.

2 Related Work

Traditional Learning Rate Schedulers: Learning rate scheduling has been extensively studied in optimization literature. Static schedulers such as Step Decay and Exponential Decay Ruder (2016) operate according to predetermined patterns, applying fixed reductions at specific intervals without considering the actual training dynamics. These approaches rely on predefined heuristics rather than adapting to the specific optimization landscape encountered during training Goodfellow u. a. (2016). Cyclical approaches such as

Cyclical Learning Rate Smith (2017a) and Cosine Annealing with Warm Restarts Loshchilov und Hutter (2017) introduce periodic variations in learning rates. While these methods provide better exploration of the parameter space, they remain constrained by their predefined patterns and fixed schedules. Their recovery mechanisms rely on periodic restarts to escape plateaus rather than targeted interventions based on specific training signals.

Adaptive Optimization Methods: Adaptive optimizers such as Adam, RMSProp, and AdaGrad Kingma und Ba (2014) take a fundamentally different approach by adjusting parameter-specific learning rates based on accumulated gradient statistics. These methods maintain historical representations of gradient behavior to scale updates differently across the parameter space. While effective in many scenarios, their adaptation mechanisms remain implicit and operate at the parameter level rather than considering global training dynamics or phase-specific strategies. Recent research has explored reinforcement learning (RL) for learning rate scheduling Li u. a. (2017); Xu u. a. (2020). These approaches frame the scheduling problem as a sequential decision process, where an agent learns a policy for adjusting learning rates to maximize performance. RL schedulers encode experiences as numerical state representations and derive implicit policies through reward maximization, resulting in potentially powerful but less interpretable adaptation strategies.

Non-Axiomatic Reasoning Systems: Our work builds upon the theoretical foundations of Non-Axiomatic Reasoning System (NARS), which provides a framework for reasoning under uncertainty and incomplete knowledge. Wang Wang (2022) explores NARS's balance of non-axiomatic, experience-driven reasoning with axiomatic tendencies under the Assumption of Insufficient Knowledge and Resources (AIKR). This framework offers truth-values with frequency and confidence components, enabling explicit modeling of uncertainty in observations and decisions. The application of NARS to practical problems has been demonstrated in diagnostic systems. Wang et al. Wang u. a. (2020) describe a NARS-based General Diagnostic Model (GDM) that applies inference mechanisms such as abduction and induction to process uncertain data. Our learning rate scheduler shares conceptual similarities with this work, though specialized for optimization tasks. The decision-making aspects of our scheduler are related to Wang's work Wang (2022) on NARS-based decision frameworks, which proposes expectation-driven decision models for goal-oriented choices under uncertainty. While this work focuses on broader artificial general intelligence applications, our implementation demonstrates how these principles can be effectively applied to the specific domain of learning rate adaptation in gradient-based optimization.

3 NARS Learning Rate Scheduler

3.1 Relationship to Optimization Theory

The NARS Learning Rate Scheduler can be understood within the framework of classical optimization theory while extending it in several directions. In traditional first-order methods, step size adaptation follows theoretical convergence bounds derived from properties of the objective function, such as Lipschitz continuity of gradients Boyd und Vandenberghe (2004). The NARS approach differs by using empirical observations and logical inference rather than analytical guarantees.

Our scheduler addresses three fundamental challenges in optimization theory:

- **Non-stationarity**: By incorporating phase-specific adaptations, the system accounts for the changing nature of the optimization landscape as training progresses
- **Exploration-exploitation balance**: Through belief-based adjustment factors, the system explicitly models this tradeoff, with larger adjustments in early phases and more conservative strategies later
- **Transfer learning across optimization problems**: The context memory and procedural patterns enable the scheduler to apply successful strategies from similar contexts, implementing a form of meta-learning for optimization

Unlike analytical approaches that provide global convergence guarantees under restrictive assumptions, the NARS scheduler offers adaptive behavior based on local empirical evidence, making it potentially more effective for the complex, non-convex landscapes typical in deep learning.

3.2 Foundations of Non-Axiomatic Logic

Learning rate scheduling is critical for optimizing neural network training, as it controls the step size of parameter updates. While traditional schedulers like step decay or cosine annealing rely on predefined rules, the NARS Learning Rate Scheduler introduces a novel approach that leverages Non-Axiomatic Logic to dynamically adjust learning rates based on training dynamics. NARS provides a reasoning system designed to handle uncertainty and perform inference under incomplete information. Its key components include:

- **Terms** representing concepts, categorized as atomic (e.g., ”x”), compound (with connectives), or statements (with copulas)
- **Truth values** consisting of frequency (f) and confidence (c) in the range $[0, 1]$
- **Inference rules** enabling the system to derive new beliefs

The expectation of a truth value is calculated as $e = c \cdot (f - 0.5) + 0.5$, representing the belief's strength. NARS defines relationships between terms using copulas such as inheritance (“$--\ >$”), similarity (“$< - >$”), implication (“==>”), and equivalence (“<==>”).

NARS's inference mechanisms support several rules:

- **Revision** combines two truth values for the same term, weighting them by evidence: $f_{\text{new}} = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2}$, $c_{\text{new}} = \frac{w_1 + w_2}{w_1 + w_2 + 1}$
- **Deduction** derives $A \to C$ from statements $A \to B$ and $B \to C$ with $f = f_1 \cdot f_2$ and $c = c_1 \cdot c_2 \cdot f_1$
- **Abduction** derives $A \to B$ from $A \to C$ and $B \to C$ with $f = f_1$ and $c = c_1 \cdot c_2 \cdot f_2$
- **Induction** derives $B \to C$ from $A \to B$ and $A \to C$ with $f = f_2$ and $c = c_1 \cdot c_2 \cdot f_1$
- **Exemplification** derives $C \to B$ from $A \to B$ and $C \to A$ with $f = 1.0$ and $c = c_1 \cdot c_2 \cdot f_1 \cdot f_2$
- Compound term operations include **intersection** ($f = f_1 \cdot f_2$, $c = c_1 \cdot c_2$), **union** ($f = f_1 + f_2 - f_1 \cdot f_2$, $c = c_1 \cdot c_2$), and **negation** ($f = 1.0 - f$, $c = c$)

3.3 System Architecture

The architecture comprises several interconnected components:

- **NAL Engine**: Manages beliefs (statements with truth values), inference cycles (applying syllogistic rules), and action selection based on belief strengths
- **Training State Tracking**: Maintains fixed-length deques for learning rates, losses, gradient norms, and improvements, enabling temporal analysis of training dynamics
- **Context Memory**: Records successful and failed actions with their contexts, building an experiential knowledge base
- **Procedural Patterns**: Tracks effective learning rate ranges for specific training phases to inform future decisions
- **Detection Mechanisms**: Implements methods for detecting plateaus, oscillations, gradient trends, and performance improvements
- **Checkpointing**: Saves and restores model states based on performance metrics
- **Warmup**: Provides gradual learning rate increases during early training

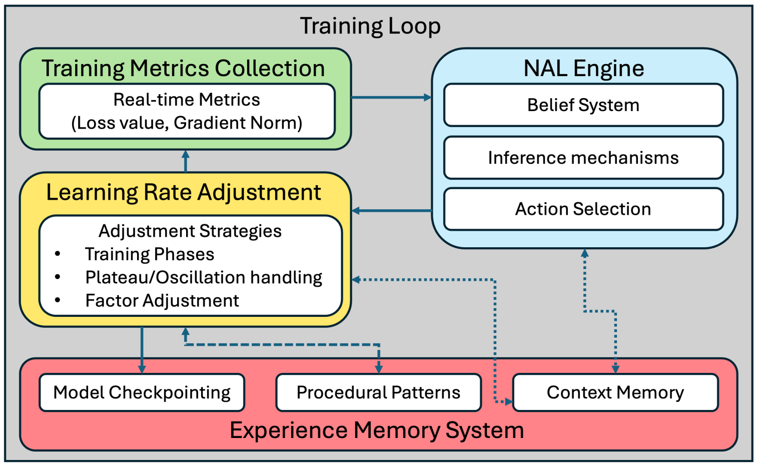


Figure 1: Architecture of the NARS Learning Rate Scheduler. The system's three main components within the training loop are: (1) Training Metrics Collection (green) for gathering real-time data; (2) NAL Engine (blue) for processing information through its belief system; and (3) Learning Rate Adjustment (yellow) for implementing strategies based on training phase and conditions. The Experience Memory System (red) supports these components through context memory storage, procedural patterns tracking, and model checkpointing.

3.4 Operational Workflow

The scheduler operates through a cycle of observation, inference, and adjustment:

3.4.1 Initialization

The system establishes the initial learning rate with minimum and maximum bounds, configures a warmup period, and references the model for checkpointing.

3.4.2 Observation Phase

The `observe` method collects metrics including:

- Improvement (difference between current and previous loss)
- Gradient norm (L2 norm of model gradients)
- Step ratio (proxy for weight update magnitude)
- Current loss
- Optional validation accuracy

These metrics populate history buffers and generate beliefs such as:

- "grad_large" ($f = 1$ if gradient norm > 0.1, $c = 0.8$)
- "plateau_detected" ($f = 1$ if loss changes are minimal, $c = 0.8$)
- "oscillating" ($f = 1$ if learning rate changes frequently, $c = 0.9$)
- "grad_decreasing" ($f = 1$ if gradient trend is negative, $c = 0.7$)
- "improvement_increasing" ($f = 1$ if improvement trend is positive, $c = 0.7$)

The system also adds phase-specific beliefs, distinguishing between early (epoch ≤ 3), mid ($3 <$ epoch ≤ 10), and late (epoch > 10) training phases.

3.4.3 Detection Mechanisms

The system includes specialized detection methods:

- `detect_plateau`: Identifies when loss has plateaued using relative changes below a threshold
- `detect_oscillation`: Detects oscillatory learning rate patterns by analyzing sign flips in recent changes
- `calculate_trends`: Computes linear trends in gradient magnitudes and improvement metrics
- `context_similarity`: Measures similarity between training contexts for experience-based decisions

3.4.4 Inference Phase

The NAL engine applies multiple types of inference through its `inference_cycle` method:

- Syllogistic rules (deduction, induction, abduction, exemplification) on inheritance statements
- Deduction on implication statements (e.g., $A ==> B$, $B ==> C$ yields $A ==> C$)
- Revision of beliefs based on new observations
- Conversion of statements (e.g., $A - - > B$ yields $B - - > A$ with adjusted truth values)

The engine then selects the optimal action based on belief expectations using the `select_best_action` method, which finds the action with the highest expectation value above a minimum confidence threshold.

3.4.5 Adjustment Phase

The `adjust_lr` method implements learning rate modifications based on inferred actions:

- During warmup (`warmup_counter < warmup_steps`), the rate increases linearly
- Early training (epoch ≤ 1, steps ¡ 100) applies gradual increases (factor ≤ 1.05)
- The system escapes minimum (2×) or maximum (0.5×) learning rates when indicated
- Strong signals trigger increases (factor 1.1–1.2) or decreases (factor 0.8–0.9) based on belief strengths
- Gentle decreases (factor 0.95) support convergence when `should_decrease_lr_gently` has high expectation
- Strategy changes reverse recent trends when progress stalls (`should_change_strategy` ¿ 0.7)
- Continuation maintains current trends with minor adjustments when `should_continue` ¿ 0.6
- Forced exploration occurs periodically (every 5th epoch with 50% probability) to break plateaus
- Historical best rates may be adopted during late training (epoch ¿ 15) based on context similarity

The new learning rate is constrained within [`lr_min`, `lr_max`] and recorded in history.

3.5 Context Memory and Experience Transfer

The scheduler implements sophisticated memory mechanisms:

- `create_context_vector`: Generates a representation of the current training state
- `remember_successful_action/remember_failed_action`: Records outcomes of actions in context
- `update_successful_ranges`: Tracks effective learning rate ranges for different training phases
- `get_best_lr_for_context`: Retrieves historically effective learning rates for similar contexts
- `update_best_state`: Saves model parameters when performance improves
- `restore_best_state`: Reverts to best checkpointed model state when needed

3.6 Integration with Training Loop

Integration with the training process occurs through the `train_with_enhanced_nal` function, which:

- Initializes the tuner with appropriate parameters
- Calls `on_epoch_start` at the beginning of each epoch
- Collects metrics from each training batch
- Calls `observe` and `adjust_lr` to update the learning rate
- Performs validation and updates the tuner with accuracy information
- Maintains comprehensive training history for analysis

Algorithm 1 NARS Learning Rate Tuner

```
NALLRTuner Class
function INITIALIZE(initial_lr, lr_min, lr_max, history_len)
    Create NALEngine instance
    Set learning rate parameters: initial_lr, lr_min, lr_max
    Initialize history queues: recent_lrs, loss_history, grad_history, improvement_history
end function
function OBSERVE(improvement, grad_norm, step_ratio, current_loss, accuracy)
    Store metrics in history
    Reset NAL engine
    Add primary and derived observations as beliefs
    Add core implications and training phase beliefs
    Run inference cycle
end function
function ADJUST_LR
    Get belief strengths for actions
    Determine action and factor based on belief strengths and training phase
    Apply special cases (warmup, boundary escape, forced exploration)
    Clip new LR to [lr_min, lr_max]
    Record action and context for memory
    return new_lr
end function
```

3.7 Advantages and Limitations

This approach offers several advantages:

- Adaptive reasoning about training dynamics through formal logical inference
- Experience-based learning through context memory and procedural patterns
- Robustness against noisy metrics through confidence-based belief revision
- Flexibility via phase-specific strategies and context-dependent adaptation
- Interpretability through explicit reasoning processes and documented decisions

However, limitations include:

- Increased computational overhead from the NAL engine and multiple detection mechanisms
- Sensitivity to hyperparameters such as confidence thresholds and detection window sizes
- Complexity in reasoning rule selection and belief formulation
- Performance variations across different model architectures and datasets

4 Results

In this section, we present experimental results evaluating the NARS Learning Rate Scheduler across three distinct optimization scenarios. We begin with the Rosenbrock function, a classical non-convex optimization benchmark with a narrow curved valley that challenges many optimizers. We then proceed to two deep learning image classification tasks of increasing complexity: Fashion MNIST (grayscale images) and CIFAR-10 (color images).

For each experiment, we compare three learning rate strategies:

- **NARS**: Our proposed approach using Non-Axiomatic Logic for dynamic learning rate adjustment based on training signals.
- **OneCycle**: A popular schedule that increases the learning rate from an initial value to a maximum, then decreases it following a cosine curve Smith (2017b). This approach is known for enabling faster convergence through controlled exploration of the loss landscape.
- **Fixed**: A constant learning rate throughout training, serving as a baseline to evaluate the effectiveness of adaptive strategies.

Experiment with Rosenbrock uses vanilla SGD and both experiments in Deep Learning were conducted using the Adam optimizer, with these scheduling strategies applied to the global learning rate. We maintained consistent hyperparameters across scheduling methods, focusing solely on the impact of learning rate adaptation mechanisms.

4.1 Rosenbrock Function Optimization

The Rosenbrock function ($f(x, y) = (a - x)^2 + b(y - x^2)^2$) serves as a challenging non-convex optimization benchmark, featuring a global minimum at $(x, y) = (a, a^2)$ where $f(x, y) = 0$. Using standard parameters ($a = 1$, $b = 100$), we evaluated the NARS scheduler against OneCycle Adam and Fixed Adam optimizers initialized at $(x, y) = (0, 1.0)$.

Quantitative results in Table 1 demonstrate the NARS scheduler's superior performance, achieving a final loss of 0.0091, an 85% improvement over Fixed Adam (0.0617) and substantially outperforming OneCycle Adam (0.0365).

Method	Final Loss	Improved wrt. Fixed (%)	Improved wrt. OneCycle (%)
NARS SGD	0.0091	85%	75%
OneCycle Adam	0.0365	40%	
Fixed Adam	0.0617		

Table 1: Loss and Improvement on Rosenbrock Function

Figure 2 illustrates the comparative performance across optimization metrics. As shown in Figure 2(a), NARS achieves consistently lower function values in later stages. Figure 2(b) reveals how NARS adaptively modulates the learning rate in response to the optimization landscape, starting with lower values before incrementally increasing. The trajectory visualization in Figure 2(c) demonstrates NARS's more efficient navigation through the characteristic narrow valley, avoiding the oscillatory behavior observed with fixed-rate methods.

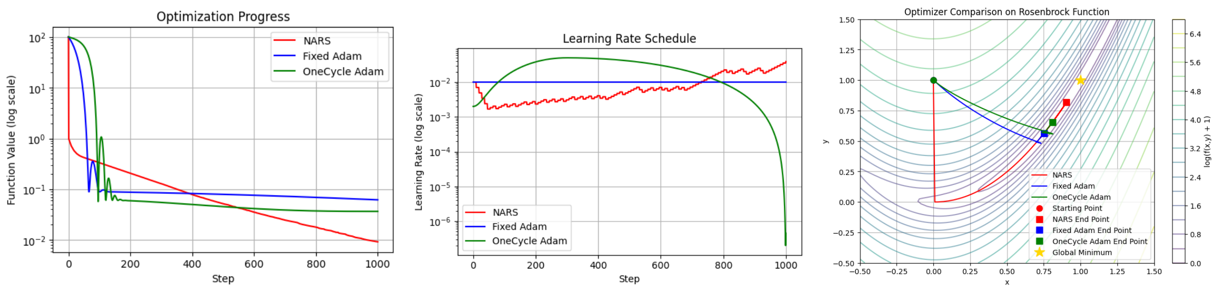


Figure 2: Rosenbrock function optimization comparison: a) Function value over optimization steps, b) Learning rate schedules, c) Optimization trajectories in parameter space. The NARS method (red) shows more direct convergence compared to Fixed Adam (blue) and OneCycle Adam (green).

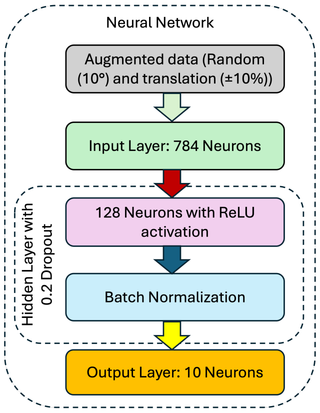
Although NARS is using SGD (which does not have the luxury of additinal help from momentum), it outperforms both the other methods. These results provide compelling evidence for the efficacy of the NARS approach in handling challenging non-convex optimization landscapes characterized by narrow convergence paths and pronounced curvature variations.

4.2 Deep Learning on Fashion MNIST Dataset

We evaluated the NARS scheduler on the Fashion MNIST dataset Xiao u. a. (2017), comprising 70,000 grayscale images (60,000 training, 10,000 testing) across 10 fashion categories at 28 × 28 pixel resolution. This dataset serves as an established benchmark for image classification tasks,. Fashion MNIST's consistent image dimensions, balanced class distribution, and inherent variance within categories make it particularly suitable for evaluating learning rate adaptation strategies, as the optimization landscape contains a mix of smooth and challenging regions.

4.2.1 Experimental Setup

We implemented a feedforward neural network with the architecture shown in Figure 3. The network consists of an input layer (784 neurons), a hidden layer (128 neurons with ReLU activation and 0.2 dropout), batch normalization, and an output layer (10 neurons). Data augmentation included random rotation (10°) and translation (±10%). Training proceeded with Adam optimization, cross-entropy loss function, initial learning rate of 1×10^{-4}, batch size 128, and 50 training epochs. We compared three learning rate strategies: NARS, OneCycle, and Fixed learning rate.


Neural Network
Augmented data (Random (10°) and translation (±10%))
Input Layer: 784 Neurons
Hidden Layer with 0.2 Dropout
128 Neurons with ReLU activation
Batch Normalization
Output Layer: 10 Neurons
Figure 3: Network Architecture for Fashion MNIST

4.2.2 Performance Analysis

As shown in Table 2, the NARS scheduler achieved 85.96% validation accuracy, outperforming OneCycle (82.79%) and Fixed (78.59%) schedulers. The NARS approach demonstrated superior generalization with a testing error of 0.3944, significantly better than OneCycle (0.4742) and Fixed (0.5907) methods. This represents a 3.17% accuracy improvement over OneCycle and a 7.37% improvement over the Fixed learning rate approach.

Learning Rate Scheduling	Accuracy	Training Error	Testing Error
NARS	85.96%	0.4773	0.3944
OneCycle	82.79%	0.0619	0.4742
Fixed	78.59%	0.7070	0.5907

Table 2: Learning rate adaptation (left) and validation accuracy progression (right) for Fashion MNIST training. NARS (yellow) maintains more stable intermediate learning rates compared to OneCycle's (red) large fluctuations and Fixed's (blue) constant value.

Figure 4 illustrates the dynamic learning rate adjustments (left) and resulting validation accuracy (right). The NARS scheduler maintained intermediate learning rates (10^{-2} range) throughout training (although we observed that it tries different values between epochs), while OneCycle followed a dramatic bell curve and Fixed remained at a constant low rate (10^{-4}). This strategic modulation resulted in consistently higher validation accuracy from early epochs, with NARS quickly reaching high accuracy levels and maintaining a clear advantage throughout the entire training process.

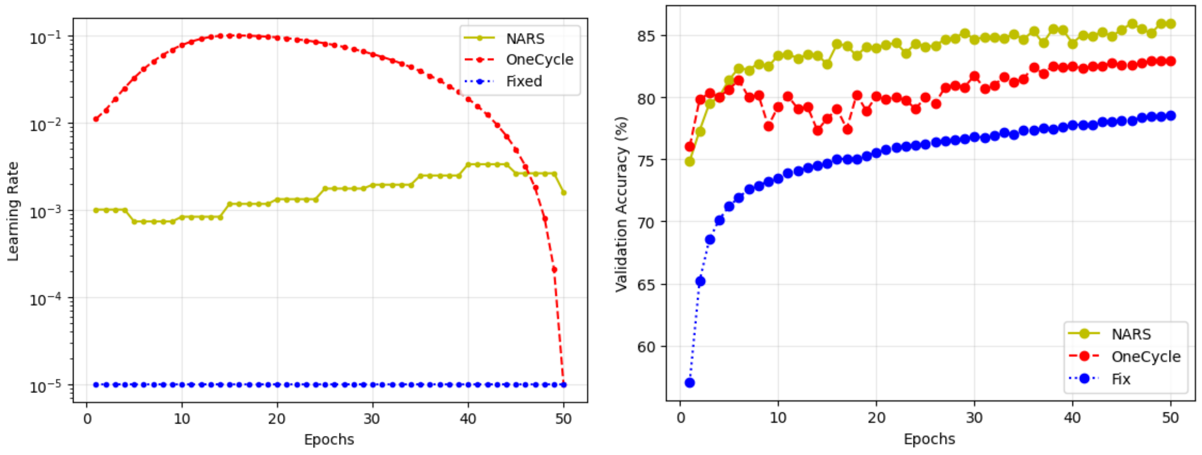


Figure 4: Learning rate adaptation (left) and validation accuracy progression (right) for Fashion MNIST training. NARS (yellow) maintains more stable intermediate learning rates compared to OneCycle's (red) large fluctuations and Fixed's (blue) constant value.

Figure 5 demonstrates the training and validation loss trajectories. The NARS scheduler exhibited faster initial reduction in validation loss and maintained consistently lower values throughout training. Despite OneCycle achieving the lowest training error (0.0619), its higher validation error suggests significant overfitting, a problem that NARS's adaptive approach effectively mitigated, leading to better generalization performance.

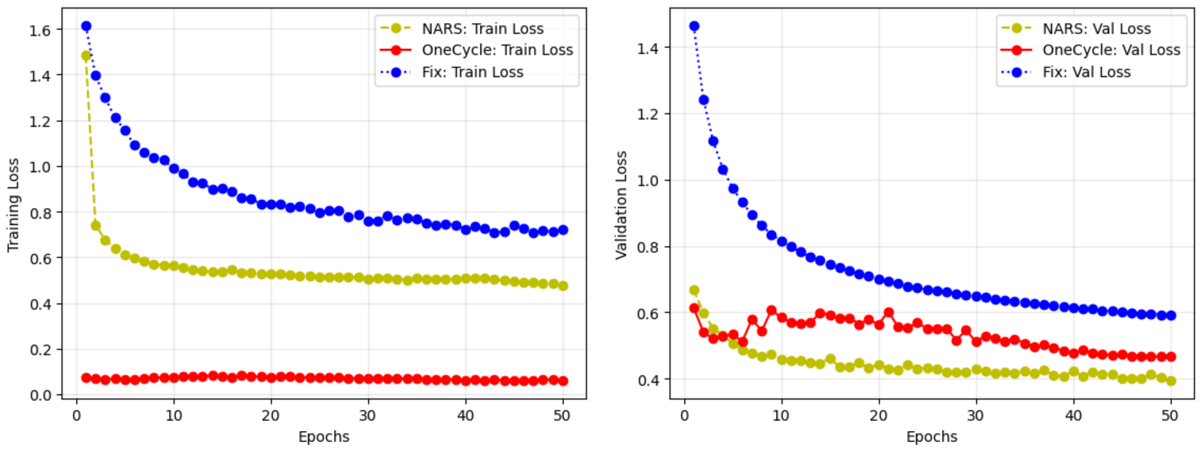
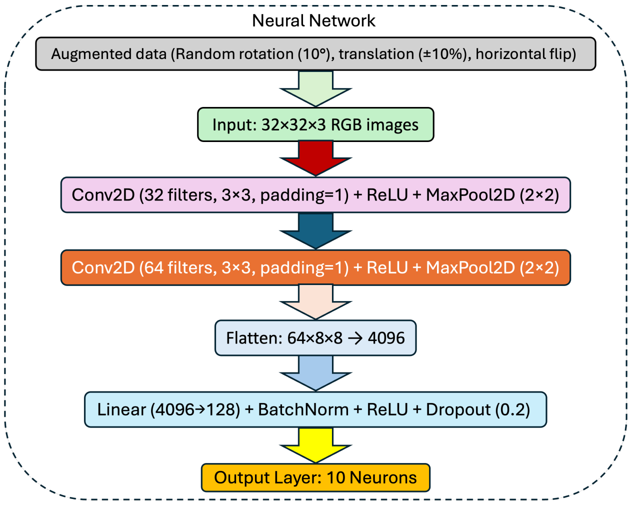


Figure 5: Training loss (left) and validation loss (right) comparison. NARS (yellow) demonstrates better generalization with lower validation loss than OneCycle (red) and Fixed (blue) schedulers.

4.3 Deep Learning on CIFAR10



Neural Network
Augmented data (Random rotation (10°), translation (±10%), horizontal flip)
Input: 32×32×3 RGB images
Conv2D (32 filters, 3×3, padding=1) + ReLU + MaxPool2D (2×2)
Conv2D (64 filters, 3×3, padding=1) + ReLU + MaxPool2D (2×2)
Flatten: 64×8×8 → 4096
Linear (4096→128) + BatchNorm + ReLU + Dropout (0.2)
Output Layer: 10 Neurons

Figure 6: Network Architecture for CIFAR10

We further evaluated our approach on CIFAR-10 Krizhevsky (2009), a more challenging benchmark comprising 60,000 color images (32×32 pixels) across 10 object categories. Unlike Fashion MNIST, CIFAR-10 introduces three color channels and greater intra-class variation, creating a more complex optimization landscape with potential for local minima and plateaus. This dataset presents a substantially different learning challenge, with its higher-dimensional input space and more nuanced feature relationships requiring effective navigation of the loss surface. By testing on both Fashion MNIST and CIFAR-10, we aimed to demonstrate the robustness of the NARS scheduler across varying levels of problem complexity.

4.3.1 Experimental Setup

We implemented a convolutional neural network architecture for the CIFAR-10 dataset as shown in Figure 6. The network consists of a feature extraction component with two convolutional blocks, followed by a classification component with fully connected layers. Each convolutional block contains a 2D convolution layer with ReLU activation and max pooling. The first convolution layer processes the RGB input (3 channels) with 32 filters, while the second layer expands this to 64 feature channels. Max pooling operations progressively reduce the spatial dimensions from the original 32×32 to 8×8.

The feature maps are then flattened and passed through a fully connected layer with 128 neurons, batch normalization, ReLU activation, and dropout regularization (rate=0.2) before the final classification layer with 10 output neurons corresponding to the CIFAR-10 classes. Data augmentation techniques including random rotation (10°), affine transformations (translations up to 10%), and horizontal flipping were applied during training to improve generalization. All images were normalized across the RGB channels with mean and standard deviation values of 0.5. The model was trained for 20 epochs using a batch size of 128 and an initial learning rate of 0.001.

4.3.2 Performance Analysis

As shown in Table 3, the NARS scheduler achieved 79.55% validation accuracy, outperforming OneCycle (76.68%) and Fixed (78.43%) schedulers. The NARS approach demonstrated generalization with a testing error of 0.6029, better than OneCycle (0.6772) and slightly better than Fixed (0.6037) methods. While the performance gap is narrower than in the Fashion MNIST experiment, NARS still demonstrates advantages on this more complex dataset with RGB images.

Learning Rate Scheduling	Accuracy	Training Error	Testing Error
NARS	79.55%	0.6954	0.6029
OneCycle	76.68%	0.0837	0.6772
Fixed	78.43%	0.6865	0.6037

Table 3: Performance comparison across learning rate schedulers on CIFAR-10

Figure 7 illustrates the dynamic learning rate adjustments (left) and resulting validation accuracy (right). The NARS scheduler showed faster early-epoch accuracy improvements, maintaining a lead throughout training. While the NARS eventually approached fixed learning rate performance, OneCycle consistently lagged behind. The learning rate plot shows how NARS maintained intermediate rates (10^{-2} range) with small adaptive adjustments, contrasting with OneCycle's dramatic bell curve and Fixed's constant low rate.

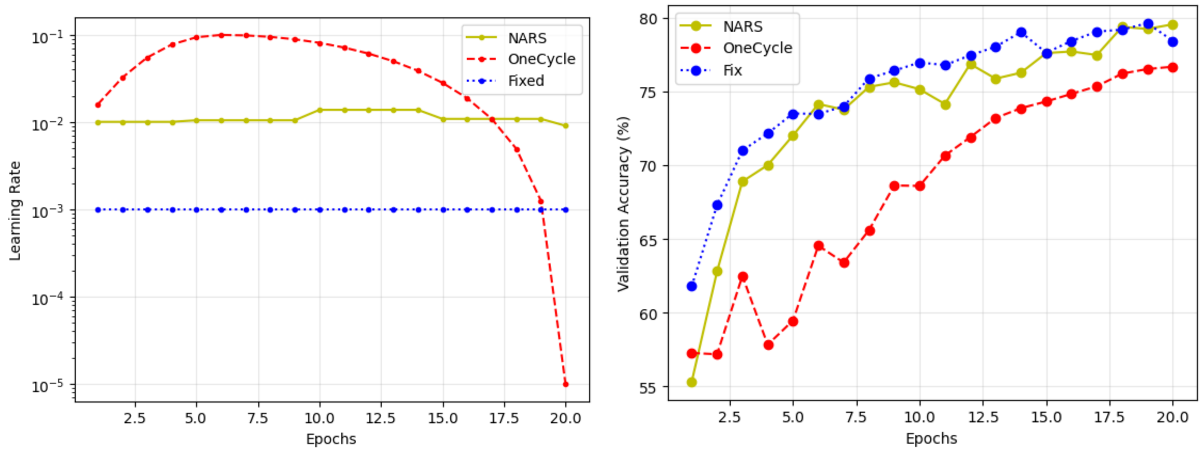


Figure 7: Learning rate adaptation (left) and validation accuracy progression (right) for CIFAR10 training. NARS (yellow) maintains more stable intermediate learning rates compared to OneCycle's (red) large fluctuations and Fixed's (blue) constant value.

Figure 8 demonstrates the training and validation loss trajectories. The NARS scheduler and Fixed approach show similar training loss curves, while OneCycle achieves dramatically lower training error. This suggests OneCycle is overfitting significantly to the training data, a problem effectively mitigated by NARS's adaptive approach that balances exploration and exploitation.

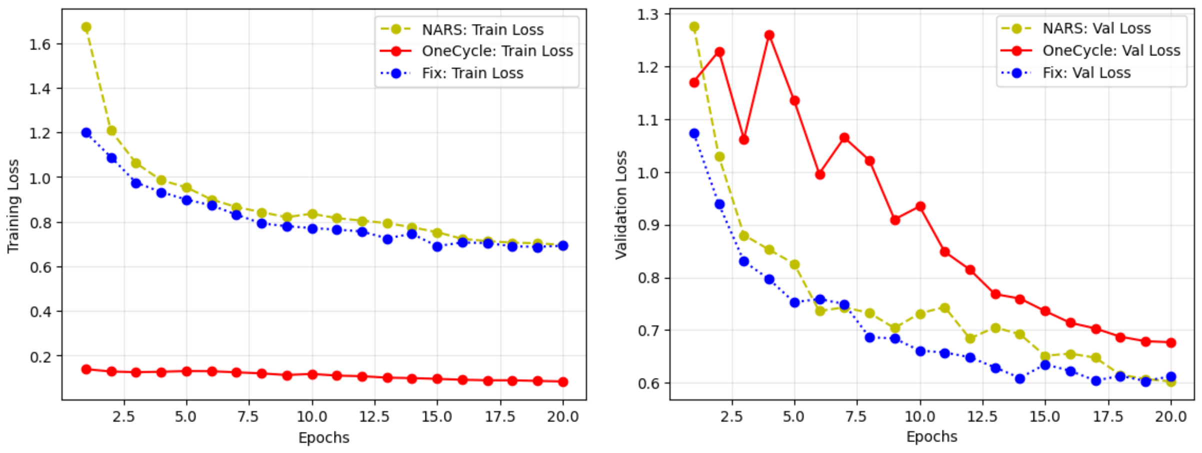


Figure 8: Training loss (left) and validation loss (right) comparison. NARS (yellow) demonstrates better generalization with lower validation loss than OneCycle (red) and Fixed (blue) schedulers.

The NARS scheduler demonstrated three distinct operational phases: (1) aggressive exploration in early epochs with rapid learning rate adjustments, (2) exploitation during middle epochs maintaining elevated but stable rates, and (3) fine-tuning in later epochs with minor rate adjustments. This phase-specific behavior, aligned with the principles outlined in Section 3, enabled more effective optimization across the training process. Notably, the NARS scheduler's ability to detect plateaus and oscillations allowed it to make context-appropriate adjustments that balanced exploration and exploitation more effectively than the comparison methods. Table 4 summarizes the performance of NARS over all the scenarios in comparison to the two more traditional alternatives. It is quite impressive that NARS managed to outperform in all the scenarios.

Scenario	NARS Advantage over OneCycle	NARS Advantage over Fixed
Rosenbrock	+75%	+82%
Fashion MNIST	+3.17%	+7.37%
CIFAR10	+2.87%	+1.12%

Table 4: NARS performance in all three scenarios

5 Discussion

5.1 Conceptual Parallels Between NARS and Reinforcement Learning

The striking conceptual parallels between Non-Axiomatic Reasoning Systems (NARS) and Reinforcement Learning (RL) reveal deep connections between symbolic reasoning and statistical learning frameworks that directly informed our implementation. Both paradigms fundamentally operate through experiential learning, continuously updating their internal models as they process new information. The parallels between the two fields was a source of inspiration in designing the NARS here. This connection manifests concretely in our NARS scheduler's implementation through several mechanisms:

- **Context Memory as Experience Replay**: Our scheduler's context memory system functions analogously to experience replay buffers in RL. Just as RL agents revisit and learn from past experiences, our `remember_successful_action` and `remember_failed_action` methods store training contexts paired with their outcomes, enabling the scheduler to retrieve historically effective learning rates for similar situations through `get_best_lr_for_context`. This direct implementation of experience-based learning bridges symbolic reasoning with statistical pattern recognition.

- **Truth Values as Uncertainty Quantification**: The frequency-confidence truth values in our implementation (such as `grad_large (f=1, c=0.8)` and `plateau_detected (f=1, c=0.8)`) serve the same functional role as uncertainty estimations in Bayesian RL. Both approaches maintain probabilistic beliefs about the environment and update these through evidence accumulation. The NAL engine's belief revision operations formalize this update process in a manner that preserves uncertainty while incorporating new evidence, directly comparable to Bayesian belief updates in probabilistic RL frameworks. However note that, in contrast to RL, we are not trying to fit any specific probability distribution to the environment.

- **Belief-Based Action Selection**: Our `select_best_action` method implements a form of expectation-maximization analogous to RL policy selection. By selecting the action with the highest expectation value above a minimum confidence threshold, the scheduler balances exploitation (high expectation) with a form of uncertainty-aware exploration (confidence thresholds). This mirrors how modern RL algorithms use upper confidence bounds or Thompson sampling to manage exploration-exploitation tradeoffs.

- **Belief-Based Action Selection**: Inspired by methods like ϵ-greedy, we added a exploration mechanism to that our system to make it more confident in making the correct action.

- **Phase-Specific Strategies as Policy Adaptation**: The scheduler's distinct behavioral phases (exploration in early epochs, exploitation in middle epochs, and fine-tuning in later epochs) implement a form of non-stationary policy optimization similar to curriculum learning in RL. Both approaches recognize that different training stages require different strategies, a principle explicitly encoded in our phase-specific beliefs and adjustment factors.

The temporal reasoning capabilities of both systems reveal perhaps the most profound connection. Our implementation's trend detection mechanisms (`calculate_trends`) extract temporal patterns from training dynamics, allowing the scheduler to project future performance based on current trajectories. This mirrors how temporal difference learning in RL builds predictive models of future rewards, with both systems addressing the fundamental challenge of temporal credit assignment, determining which actions lead to future benefits.

These concrete implementation parallels suggest that our NARS scheduler effectively operates as a symbolic reasoning layer atop a statistical optimization process, with the NAL engine serving as an explicit, interpretable policy framework for learning rate adaptation. This hybrid approach combines the transparency of rule-based systems with the adaptivity of statistical learning, demonstrating how symbolic reasoning can enhance numerical optimization through principled belief management and context-sensitive decision making.

5.2 Future Work

The NARS Learning Rate Scheduler represents a promising approach to optimization, but several avenues for future research remain unexplored. First, a comprehensive parameter sensitivity analysis would provide valuable insights into the system's behavior. The current implementation relies on numerous hyperparameters that could potentially be consolidated through mathematical relationships. Establishing formal connections between these parameters could yield a more elegant, symmetrical framework that reduces manual configuration while maintaining or improving performance.

Second, extending NARS to online learning environments presents an intriguing research direction. The conceptual parallels between NARS and reinforcement learning suggest potential for cross-fertilization of ideas. While reinforcement learning algorithms progressively refine policies through environmental interaction, NARS could offer a complementary approach through its explicit reasoning mechanisms. Investigating how NARS's logical inference capabilities perform in sequential decision problems could reveal novel optimization strategies for dynamic environments.

Third, enhancing NARS's robustness to uncertainty represents perhaps the most significant opportunity for advancement. A fundamental limitation in the current implementation is its tendency to accept observations at face value without sufficient skepticism. Real-world optimization problems frequently involve noisy or unreliable measurements, partial information, and occasionally deceptive signals. Developing mechanisms for NARS to evaluate the reliability of incoming information, perhaps through confidence-weighted belief revision or adversarial testing, would substantially increase its practical utility in noisy optimization domains. Such improvements would align NARS with robust optimization principles while preserving its distinctive logical reasoning framework.

Beyond these core improvements, several promising application domains could benefit from NARS-based optimization, such as Large Language Model Fine-tuning, Federated Learning and Multi-task and Continual Learning. These application domains represent fertile ground for extending the conceptual framework presented in this paper, potentially yielding specialized NARS-based optimizers tailored to the unique challenges of each domain. By continuing to refine the interface between symbolic reasoning and numerical optimization, we envision a new class of hybrid optimizers that combine the interpretability of logical systems with the adaptivity of statistical approaches, advancing the state-of-the-art in machine learning optimization.

6 Conclusion

The NARS Learning Rate Scheduler represents a significant advancement in the domain of adaptive optimization for neural network training. By integrating Non-Axiomatic Logic with practical training mechanisms, we have demonstrated a novel approach that dynamically adapts learning rates based on observed training dynamics, detected plateaus, and phase-specific strategies. Our experimental results show NARS performance across diverse tasks as mentioned in Table 4.

The key contribution of this work lies in establishing a principled bridge between symbolic reasoning and numerical optimization, leveraging NAL's belief system to handle uncertainty in the optimization process. This integration enables more sophisticated decision-making about learning rate adjustments than conventional heuristic approaches, while maintaining interpretability through explicit reasoning processes.

While our implementation demonstrates the efficacy of this approach, future work could focus on reducing computational overhead, formalizing mathematical relationships between hyperparameters, extending the framework to online learning environments, and enhancing robustness to noisy measurements. These improvements would further advance the practical applicability of NARS-based optimization across diverse machine learning domains.

This research opens new avenues for cross-fertilization between symbolic AI and statistical learning methods, suggesting that hybrid approaches incorporating formal reasoning with adaptive numerical optimization may offer substantial benefits for next-generation machine learning systems.

References

[Boyd und Vandenberghe 2004] Boyd, Stephen ; Vandenberghe, Lieven: *Convex Optimization.* Cambridge University Press, 2004

[Goodfellow u. a. 2016] Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep Learning.* MIT Press, 2016

[Kingma und Ba 2014] Kingma, Diederik P. ; Ba, Jimmy: Adam: A Method for Stochastic Optimization. In: *arXiv preprint arXiv:1412.6980* (2014)

[Krizhevsky 2009] Krizhevsky, Alex: Learning multiple layers of features from tiny images. 2009. – Forschungsbericht

[Li u. a. 2010] Li, Lihong ; Chu, Wei ; Langford, John ; Schapire, Robert E.: A Contextual-Bandit Approach to Personalized News Article Recommendation. In: *Proceedings of the 19th International Conference on World Wide Web*, 2010, S. 661–670

[Li u. a. 2017] Li, Lisha ; Jamieson, Kevin ; DeSalvo, Giulia ; Rostamizadeh, Afshin ; Talwalkar, Ameet: Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. In: *Journal of Machine Learning Research* 18 (2017), Nr. 185, S. 1–52

[Loshchilov und Hutter 2017] Loshchilov, Ilya ; Hutter, Frank: SGDR: Stochastic Gradient Descent with Warm Restarts. In: *International Conference on Learning Representations*, 2017

[Ruder 2016] Ruder, Sebastian: An Overview of Gradient Descent Optimization Algorithms. In: *arXiv preprint arXiv:1609.04747* (2016)

[Smith 2017a] Smith, Leslie N.: Cyclical Learning Rates for Training Neural Networks. In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017, S. 464–472

[Smith 2017b] Smith, Leslie N.: *Cyclical Learning Rates for Training Neural Networks.* 2017. – URL `https://arxiv.org/abs/1506.01186`

[Wang 2006] Wang, Pei: *Rigid Flexibility: The Logic of Intelligence.* Springer, 2006

[Wang 2013] Wang, Pei: *Non-Axiomatic Logic: A Model of Intelligent Reasoning.* World Scientific, 2013

[Wang 2019] Wang, Pei: NARS: A Reasoning System for Real-World Intelligence. In: *International Journal of Machine Consciousness* 6 (2019), Nr. 1, S. 1–13

[Wang 2022] Wang, Pei: Axiomatic Reasoning in NARS / Technical Report. 2022. – Forschungsbericht

[Wang u. a. 2020] Wang, Pei ; Power, Bill ; Li, Xiang: A NARS-Based Diagnostic Model / Technical Report 10, Temple University AGI Team. 2020. – Forschungsbericht

[Xiao u. a. 2017] Xiao, Han ; Rasul, Kashif ; Vollgraf, Roland: *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.* 2017. – URL `https://arxiv.org/abs/1708.07747`

[Xu u. a. 2020] Xu, Zhongwen ; Hasselt, Hado P. van ; Silver, David: Meta-Gradient Reinforcement Learning. In: *Advances in Neural Information Processing Systems* Bd. 33, 2020, S. 2396–2407