Aletheion: Fractal Epistemic Architecture for Large Language Models

Aletheion Research Collective

June 2024

Abstract

Large language models hallucinate facts, contradict themselves, and rarely express calibrated uncertainty—failure modes rooted in softmax's forced normalization. We introduce *epistemic softmax*, which augments logits with trainable confidence gates (Q_1, Q_2) and variance-aware optimization (VARO). Applied fractally to all transformer softmax instances—attention weights, head aggregation, output vocabularies—this yields *Aletheion*, an architecture where uncertainty propagates hierarchically. We formalize three implementation levels: output-only (Level 1), attention-aware (Level 2), and full fractal (Level 3). VARO training aligns epistemic confidence with ground-truth ambiguity via $L = L_{\text{CE}} + \lambda \|u - u^*\|_2^2$. Theoretical analysis shows (1) uncertainty composes monotonically across layers, (2) computational overhead is $< 5\%$ relative to transformers, and (3) calibration improves under VARO. We project Level 3 achieves 58% on TruthfulQA (vs. 40% baseline), expected calibration error of 0.06 (vs. 0.15), and uncertainty–error correlation of 0.8 (vs. 0.3). Aletheion reframes uncertainty as an architectural primitive, enabling models that know when they do not know—a critical step toward safe, reliable AI.

Notation

Throughout this paper, we use the following conventions:

Dimensions

- L: number of transformer layers
- H: number of attention heads per layer
- d: model hidden dimension (d_{model})
- d_k: dimension of keys/queries (typically d/H)
- d_v: dimension of values (typically d/H)
- n: sequence length
- V: vocabulary size
- k: gate MLP hidden dimension

Variables

- $h^{(l)} \in \mathbb{R}^{n \times d}$: hidden state at layer l
- $Q^{(l,h)}, K^{(l,h)}, V^{(l,h)} \in \mathbb{R}^{n \times d_k}$: query, key, value matrices for head h of layer l
- $a^{(l,h)} \in \mathbb{R}^{n \times n}$: attention logits for head h of layer l
- $p_{\text{att}}^{(l,h)} \in \mathbb{R}^{n \times n}$: attention weights after epistemic softmax
- $u_{\text{att}}^{(l,h)} \in [0,1]$: uncertainty from attention head h of layer l
- $o^{(l,h)} \in \mathbb{R}^{n \times d_v}$: output of attention head h in layer l
- $w^{(l)} \in \mathbb{R}^{H}$: head aggregation logits at layer l
- $u^{(l)} \in [0,1]$: aggregated uncertainty from layer l
- $u_{\text{final}} \in [0,1]$: final output uncertainty

Functions

- $Q_1 : \mathbb{R}^d \to [0,1]$: local uncertainty gate
- $Q_2 : \mathbb{R}^d \to [0,1]$: global consensus gate
- `EpSoftmax`: epistemic softmax operator (Algorithm 1)
- $f : [0,1]^{L+1} \to [0,1]$: uncertainty aggregation function

Losses

- $\mathcal{L}_{\text{CE}}$: cross-entropy loss
- $\mathcal{L}_{\text{VARO}}$: variance-adjusted ranking optimization loss
- $\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{VARO}}$: total loss

1 Introduction

Large language models (LLMs) deliver impressive generative capabilities yet remain unreliable in high-stakes settings. They hallucinate citations, contradict themselves across turns, flatter users even when prompted with false statements, and rarely admit uncertainty. These behaviors undermine safety, reliability, and trustworthiness in downstream deployments [6]. Contemporary mitigation strategies—retrieval augmentation, reinforcement learning from human feedback (RLHF), prompt engineering, and temperature heuristics—address symptoms but leave the architectural root cause intact.

1.1 The Problem with Modern LLMs

- **Hallucination:** Transformers confidently produce fabricated facts when the hidden state lacks evidence, leading to erroneous citations and reports.
- **Inconsistency:** Autoregressive decoding produces context-dependent contradictions because there is no persistent epistemic state that aggregates evidence across turns.
- **Sycophancy:** Preference optimization pushes models to agree with users instead of contesting falsehoods, reinforcing misinformation.
- **Inability to express doubt:** Softmax-based decoders must emit a normalized distribution, even when logits are uninformative, eliminating the option to say "I do not know."

1.2 Previous Approaches

Retrieval augmented generation, RLHF or DPO, prompt engineering, confidence calibration, and temperature tuning provide partial relief but do not model epistemic uncertainty within the network. Bayesian ensembles and Monte Carlo dropout offer uncertainty estimates yet remain post-hoc, costly, or incompatible with production-scale decoding [5, 8, 16].

1.3 Our Insight

Softmax appears throughout the transformer pipeline: attention weights, head aggregation, output vocabularies, mixture-of-experts gates, and auxiliary routing mechanisms [5]. Each instance forces a probability distribution even when the upstream representation encodes insufficient evidence. We observe that epistemic softmax—a composite of two gating signals (Q_1 and Q_2), a variance-adjusted ranking objective (VARO), and an exploration strategy—can replace any softmax invocation. The key question is: *what if this replacement is applied fractally across the entire network?*

1.4 Contributions

1. **Root-cause analysis:** We identify forced normalization via softmax as the shared trigger of five dominant failure modes in LLMs [6].
2. **Epistemic softmax primitive:** We define a differentiable operator that augments logits with explicit epistemic confidence while remaining compatible with transformer training pipelines.
3. **Fractal architecture:** We formalize the Aletheion principle—replace every softmax with epistemic softmax—and present implementation levels from output-only to full-stack integration.
4. **Training methodology:** We introduce the VARO objective for calibrating epistemic confidence and describe gradient flow through the new gates.
5. **Theoretical and experimental roadmap:** We analyze uncertainty propagation, computational overhead, and outline evaluation protocols for near-term validation.

2 Background

2.1 Transformer Architecture

Transformers encode tokens into contextual representations using multi-head self-attention, feed-forward networks, and layer normalization [22]. Given query, key, and value projections ($Q, K, V \in \mathbb{R}^{n \times d_k}$) per head, attention computes weights via scaled dot-product softmax and aggregates values accordingly. Feed-forward sublayers apply position-wise non-linear transformations, while residual connections and layer normalization stabilize training.

2.2 Softmax and Uncertainty

For logits $\mathbf{z} \in \mathbb{R}^m$, softmax produces $\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$. Transformers rely on softmax to generate attention scores, vocabulary distributions, and gating coefficients. However, forcing a probability distribution even under epistemic uncertainty masks the model's ignorance.

2.3 Epistemic vs. Aleatoric Uncertainty

Aleatoric uncertainty arises from inherent data noise, while epistemic uncertainty reflects ignorance reducible with more information. LLMs trained on static corpora primarily face epistemic uncertainty when encountering novel facts, adversarial prompts, or contradictory instructions; softmax conflates these modes by always returning a confident distribution.

2.4 Related Work

Bayesian neural networks, deep ensembles, Monte Carlo dropout, selective prediction, and conformal prediction provide valuable uncertainty estimates but are costly or post-hoc [1, 8, 16, 15, 20]. Calibration studies for LLMs rely on selective prediction or verbalized confidence. Our approach differs by embedding epistemic reasoning directly within the attention and decoding primitives, avoiding ensembling or expensive sampling [18, 14].

Consistency Training and Sycophancy Recent work by Google DeepMind [9] addresses sycophancy and jailbreaks through *consistency training*: augmenting training data with paraphrased prompts and penalizing inconsistent responses across paraphrases. This behavioral-level intervention reduces sycophancy without modifying the underlying architecture or providing explicit uncertainty estimates.

Aletheion is *complementary* to consistency training. While consistency training enforces paraphrase robustness at the training objective level, epistemic softmax provides architectural uncertainty quantification that operates at every decision point. Combined approaches—where $\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_1 \mathcal{L}_{\text{VARO}} + \lambda_2 \mathcal{L}_{\text{consistency}}$—may yield models that are both *calibrated* (via Aletheion's epistemic gates) and *behaviorally consistent* (via consistency training). We leave empirical validation of this combination to future work.

3 Failure Modes

We synthesize five dominant failure modes from operational evaluations [6]. Each stems from softmax-imposed certainty.

1. **Hallucination:** When the final hidden state lacks evidence for any candidate token, softmax still returns a peaked distribution, leading to fabricated facts or citations. Cross-entropy loss reinforces whichever hallucination receives accidental reinforcement, without penalizing unjustified confidence.

2. **Inconsistency:** Autoregressive decoding conditions on prior outputs, so early confident errors propagate. Softmax never signals "insufficient evidence," preventing the model from pausing or branching.

3. **Sycophancy:** RLHF incentivizes agreement with human raters. Softmax offers no mechanism to represent disagreement or uncertainty, so the model converges to high-confidence agreement even under contradictory evidence.

4. **Prompt brittleness:** Small paraphrases perturb token-level logits, and softmax amplifies minor logit differences into categorical preferences. Without uncertainty-aware smoothing, responses vary dramatically across prompts with equivalent semantics.

5. **Inability to express uncertainty:** The model cannot emit an "I do not know" distribution because softmax enforces confidence. Users misinterpret the resulting probabilities as certainty, even when the internal representations were ambiguous.

4 Epistemic Softmax

4.1 Motivation

Standard softmax treats logits as fully reliable. We seek an operator that preserves differentiability but factors epistemic uncertainty into every decision.

4.2 Components

Q_1 **(Local uncertainty):** A lightweight neural gate that maps the context of a softmax invocation—e.g., per-head query vectors—to $[0, 1]$. Low values indicate insufficient evidence at that locus.

Q_2 **(Global consensus):** Aggregates sibling contexts, such as attention heads or decoder layers, to estimate agreement. Disagreement implies epistemic uncertainty.

VARO (Variance-Adjusted Ranking Optimization): An auxiliary loss that penalizes confident errors and rewards calibrated confidence: $L_{\text{VARO}} = -\log p(y^*) + \lambda \operatorname{Var}(p)$.

Exploration strategy: Dynamically adjusts sampling temperature and decoding strategy based on the epistemic confidence score.

4.3 Algorithmic Definition

Algorithm 1 clarifies the gating mechanism and returned uncertainty signal.

The gating interpolates between a confident softmax distribution and a maximally uncertain uniform distribution. Returning p_{gated} and u makes explicit that epistemic softmax outputs both a calibrated distribution and an uncertainty scalar.

4.4 Properties

Epistemic softmax reduces to standard softmax when $Q_1 = Q_2 = 1$, outputs uniform distributions when $Q_1 = Q_2 = 0$, remains differentiable, and exposes explicit uncertainty $u = 1 - Q_1 Q_2$.

Algorithm 1 Epistemic Softmax

Require: logits z, context features c_{ctx}, gate networks Q_1, Q_2, base temperature τ_0, threshold τ_{thresh}

$q_1 \leftarrow Q_1(c_{\text{ctx}})$ ▷ local evidence gate
$q_2 \leftarrow Q_2(c_{\text{ctx}})$ ▷ cross-context consensus gate
$c \leftarrow \text{clip}(q_1 q_2, \varepsilon, 1)$ ▷ epistemic confidence
$\tau \leftarrow \tau_0 / c$ if $c < \tau_{\text{thresh}}$ else τ_0
$p \leftarrow \text{softmax}(z/\tau)$
$u_{\text{uniform}} \leftarrow \mathbf{1}/|p|$
$p_{\text{gated}} \leftarrow c \cdot p + (1 - c) \cdot u_{\text{uniform}}$
$u \leftarrow 1 - c$ ▷ epistemic uncertainty scalar
return p_{gated}, u

5 Fractal Architecture

5.1 Level 1: Output-Only

Let h_t denote decoder state, $z = Wh_t$ the logits, and $c^{(\text{out})}$ the context features (e.g., hidden state, attention summary). Epistemic softmax yields $(p_t, u_t) = \text{EpSoftmax}(z, c^{(\text{out})})$. Uncertainty u_t modulates decoding temperature and can trigger abstention policies.

5.2 Level 2: Attention + Output

Level 2 applies epistemic softmax to both attention mechanisms and output distributions.

Attention with Epistemic Gating

For layer l and head h, we first compute attention logits:

$$a^{(l,h)} = \frac{Q^{(l,h)}(K^{(l,h)})^\top}{\sqrt{d_k}} \in \mathbb{R}^{n \times n} \tag{1}$$

where $Q^{(l,h)} = h^{(l-1)} W_Q^{(l,h)}$ and $K^{(l,h)} = h^{(l-1)} W_K^{(l,h)}$ are the projected query and key matrices.

We then apply epistemic softmax to obtain gated attention weights:

$$(p_{\text{att}}^{(l,h)}, u_{\text{att}}^{(l,h)}) = \texttt{EpSoftmax}(a^{(l,h)}, c_{\text{att}}^{(l,h)}) \tag{2}$$

where $c_{\text{att}}^{(l,h)} = Q_{[:,0,:]}^{(l,h)}$ is the context vector (we use the first query position as representative context, though any pooling strategy works).

The gated attention is applied to values:

$$o^{(l,h)} = p_{\text{att}}^{(l,h)} \cdot V^{(l,h)} \in \mathbb{R}^{n \times d_v} \tag{3}$$

where $V^{(l,h)} = h^{(l-1)} W_V^{(l,h)}$.

Head Aggregation with Epistemic Gating

After computing outputs from all H heads, we aggregate them using a second epistemic gate. First, concatenate head outputs:

$$\text{head\_concat}^{(l)} = [o^{(l,1)} \,||\, o^{(l,2)} \,||\, \cdots \,||\, o^{(l,H)}] \in \mathbb{R}^{n \times d} \tag{4}$$

To determine how to weight each head, we compute aggregation logits via a learned MLP:

$$w^{(l)} = \text{MLP}_{\text{agg}}(\text{mean}(\text{head\_concat}^{(l)})) \in \mathbb{R}^H \tag{5}$$

where MLP_{agg} is a small feedforward network that outputs H scalar logits.

We construct the context for the head aggregation gate:

$$c^{(l)}_{\text{head}} = \text{mean}_{\text{seq}}(\text{head\_concat}^{(l)}) \in \mathbb{R}^d \tag{6}$$

(mean pooling over the sequence dimension).

Apply epistemic softmax to obtain head mixing weights:

$$(p^{(l)}_{\text{head}}, u^{(l)}_{\text{head}}) = \texttt{EpSoftmax}(w^{(l)}, c^{(l)}_{\text{head}}) \tag{7}$$

where $p^{(l)}_{\text{head}} \in \mathbb{R}^H$ is a probability distribution over heads.

The final layer output is the weighted combination:

$$h^{(l)}_{\text{attn}} = \sum_{h=1}^{H} p^{(l)}_{\text{head},h} \cdot o^{(l,h)} \in \mathbb{R}^{n \times d} \tag{8}$$

Layer Uncertainty Aggregation

The combined uncertainty for layer l aggregates uncertainties from all heads and the head mixing gate:

$$u^{(l)} = \max\left(\max_{h \in [H]} u^{(l,h)}_{\text{att}},\ u^{(l)}_{\text{head}}\right) \tag{9}$$

This conservative aggregation ensures that if *any* head or the aggregation is uncertain, the layer reflects that uncertainty.

Complete Layer Forward Pass

The complete forward pass for layer l is:

$$h^{(l)}_{\text{attn}} = \text{LayerNorm}\left(h^{(l-1)} + \text{MultiHeadAttn}_{\text{epistemic}}(h^{(l-1)})\right) \tag{10}$$

$$h^{(l)} = \text{LayerNorm}\left(h^{(l)}_{\text{attn}} + \text{FFN}(h^{(l)}_{\text{attn}})\right) \tag{11}$$

where $\text{MultiHeadAttn}_{\text{epistemic}}$ incorporates all the epistemic gating described above.

5.3 Level 3: Full Fractal

Level 3 replaces every softmax invocation—mixture-of-experts routers, adaptive span controllers, key-value selection—with epistemic softmax. Each module exports an uncertainty scalar; the layer exposes $(y^{(l)}, u^{(l)})$. Uncertainty composition follows a monotone aggregation function f:

$$u_{\text{final}} = f\big(u^{(1)}_{\text{att}}, \ldots, u^{(L)}_{\text{att}}, u^{(1)}_{\text{head}}, \ldots, u^{(L)}_{\text{head}}, u_{\text{out}}\big). \tag{12}$$

Choices include max (conservative), mean (smooth), or a learned aggregator trained to predict downstream errors.

5.4 Fractal Pseudocode

Algorithm 2 Fractal Epistemic Transformer (Forward Pass)

Require: Token sequence $x = (x_1, \ldots, x_n)$
Ensure: Probability distribution p_{out}, uncertainty u_{final}

$h^{(0)} \leftarrow \text{Embed}(x) + \text{PositionalEncoding}(x)$

for $l = 1$ to L **do**
  // Multi-head attention with epistemic gating
  for $h = 1$ to H **do**
    $Q^{(l,h)} \leftarrow h^{(l-1)} W_Q^{(l,h)}$
    $K^{(l,h)} \leftarrow h^{(l-1)} W_K^{(l,h)}$
    $V^{(l,h)} \leftarrow h^{(l-1)} W_V^{(l,h)}$

    // Compute attention logits
    $a^{(l,h)} \leftarrow (Q^{(l,h)} (K^{(l,h)})^\top) / \sqrt{d_k}$

    // Apply epistemic softmax to attention
    $c_{\text{att}}^{(l,h)} \leftarrow Q_{[:,0,:]}^{(l,h)}$ ▷ use first query as context
    $(p_{\text{att}}^{(l,h)}, u_{\text{att}}^{(l,h)}) \leftarrow \texttt{EpSoftmax}(a^{(l,h)}, c_{\text{att}}^{(l,h)})$

    // Apply gated attention to values
    $o^{(l,h)} \leftarrow p_{\text{att}}^{(l,h)} \cdot V^{(l,h)}$
  end for

  // Aggregate heads with epistemic gating
  $\text{head\_concat} \leftarrow [o^{(l,1)} \,||\, \cdots \,||\, o^{(l,H)}]$
  $w^{(l)} \leftarrow \text{MLP}_{\text{agg}}(\text{mean}(\text{head\_concat}))$ ▷ produces H logits
  $c_{\text{head}}^{(l)} \leftarrow \text{mean}_{\text{seq}}(\text{head\_concat})$ ▷ aggregated context
  $(p_{\text{head}}^{(l)}, u_{\text{head}}^{(l)}) \leftarrow \texttt{EpSoftmax}(w^{(l)}, c_{\text{head}}^{(l)})$

  // Weighted head combination
  $h_{\text{attn}}^{(l)} \leftarrow \sum_{h=1}^{H} p_{\text{head},h}^{(l)} \cdot o^{(l,h)}$

  // Apply residual + FFN
  $h_{\text{attn}}^{(l)} \leftarrow \text{LayerNorm}(h^{(l-1)} + h_{\text{attn}}^{(l)})$
  $h^{(l)} \leftarrow \text{LayerNorm}(h_{\text{attn}}^{(l)} + \text{FFN}(h_{\text{attn}}^{(l)}))$

  // Layer uncertainty
  $u^{(l)} \leftarrow \max(\max_h u_{\text{att}}^{(l,h)}, u_{\text{head}}^{(l)})$
end for

// Output distribution with epistemic gating
$\text{logits} \leftarrow h^{(L)} W_{\text{vocab}}$
$c_{\text{out}} \leftarrow \text{mean}_{\text{seq}}(h^{(L)})$
$(p_{\text{out}}, u_{\text{out}}) \leftarrow \texttt{EpSoftmax}(\text{logits}, c_{\text{out}})$

// Final uncertainty aggregation
$u_{\text{final}} \leftarrow \max(u^{(1)}, \ldots, u^{(L)}, u_{\text{out}})$
return $p_{\text{out}}, u_{\text{final}}$

5.5 Uncertainty Propagation

For a transformer with L layers, conservative deployment adopts

$$u_{\text{final}} = \max\big(\max_{l} u_{\text{att}}^{(l)}, u_{\text{out}}\big). \tag{13}$$

Learned aggregators can be implemented as small monotone networks that take concatenated uncertainties and output a calibrated scalar.

6 Training with VARO

6.1 Supervisory Signal u^*

Training requires a target uncertainty u^*:

1. **Data ambiguity:** For examples with multiple valid labels, assign $u^* = 1 - 1/|\mathcal{Y}|$.
2. **Head variance:** Estimate u^* using variance of attention head outputs: $u^* = \sigma^2(\{z_h\})/(\sigma^2(\{z_h\})+1)$.
3. **Distributional distance:** Detect out-of-distribution tokens via density models or embedding distances, mapping high distances to high u^*.
4. **Self-consistency probes:** Monte Carlo decoding disagreement supplies additional targets during fine-tuning.

6.1.1 Practical Implementation Strategy

The choice of u^* depends on the training phase and available supervision:

Phase 0-1: Pre-training (no labeled uncertainty) Use **Method 2 (Head Variance)**:

$$u^* = \frac{\sigma^2(\{z_h\})}{\sigma^2(\{z_h\}) + 1} \tag{14}$$

where z_h are logits from different attention heads. This is computed automatically during forward pass and requires no external labels.

Implementation:

```
heads_logits = [head_1.logits, ..., head_H.logits]  # [H, B, T, V]
variance = torch.var(heads_logits, dim=0)            # [B, T, V]
u_star = variance / (variance + 1)                   # normalize to [0,1]
```

Phase 2: Fine-tuning with labeled data Use **Method 1 (Data Ambiguity)**:

For examples with multiple valid labels $Y = \{y_1, \ldots, y_k\}$:

$$u^* = 1 - \frac{1}{|Y|} \tag{15}$$

Example: Question "What is the capital of the Netherlands?"

- If dataset has both "Amsterdam" (official capital) and "The Hague" (seat of government):

- $Y = \{\text{Amsterdam}, \text{The Hague}\}$, thus $|Y| = 2$
- Therefore $u^* = 1 - 1/2 = 0.5$ (high ambiguity)

Implementation:

```
if len(valid_labels) > 1:
    u_star = 1.0 - 1.0/len(valid_labels)
else:
    u_star = 0.0  # unambiguous example
```

Phase 3: Out-of-Distribution Detection Use **Method 3 (Distributional Distance)**:

$$u^* = \min\left(1, \frac{d(x, X_{\text{train}})}{d_{\max}}\right) \tag{16}$$

where $d(x, X_{\text{train}})$ is the distance from input x to the nearest training example.

Implementation using embedding distance:

```
emb_x = encoder(x)                              # current input embedding
emb_train = encoder(X_train_sample)             # sample from training set
distances = torch.cdist(emb_x, emb_train)       # pairwise distances
min_dist = torch.min(distances)
u_star = torch.clamp(min_dist / d_max, 0, 1)
```

Phase 4: Post-training validation Use **Method 4 (Self-Consistency)**:

Generate K responses, measure disagreement:

$$u^* = 1 - (\text{agreement rate}) \tag{17}$$

where agreement rate $= \frac{\text{count of most common response}}{K}$.

Implementation:

```
responses = [model.generate(prompt, temp=T) for _ in range(K)]
unique_responses = set(responses)
agreement_rate = max(responses.count(r) for r in unique_responses) / K
u_star = 1.0 - agreement_rate
```

Combining methods In practice, use a weighted combination:

$$u^* = w_1 \cdot u^*_{\text{variance}} + w_2 \cdot u^*_{\text{ambiguity}} + w_3 \cdot u^*_{\text{distance}} \tag{18}$$

where weights $\{w_i\}$ depend on available supervision signals and sum to 1.

6.2 Loss and Gradient Flow

The total loss is

$$L = L_{\text{CE}}(p_{\text{gated}}, y^*) + \lambda \|u - u^*\|_2^2. \tag{19}$$

Gradients propagate through the gates:

$$\frac{\partial L}{\partial z} = \frac{\partial L_{\text{CE}}}{\partial z} + \lambda \frac{\partial u}{\partial z} 2(u - u^*), \tag{20}$$

$$\frac{\partial L}{\partial Q_i} = \frac{\partial L}{\partial u} \frac{\partial u}{\partial Q_i}, \quad i \in \{1, 2\}. \tag{21}$$

Because $u = 1 - Q_1 Q_2$, both gates receive gradients whenever predicted uncertainty misaligns with supervision.

6.3 Training Phases

1. **Phase 0: Baseline pretraining.** Train a standard transformer with cross-entropy until convergence.
2. **Phase 1: Gate warm-start.** Insert Q_1, Q_2 modules with outputs initialized near 1; freeze them for T_w steps while continuing baseline training.
3. **Phase 2: VARO activation.** Unfreeze gates, enable VARO with schedule λ_t, and introduce uncertainty targets u^*.
4. **Phase 3: Epistemic decoding.** Use u to control temperature, abstention, retrieval triggers, and self-consistency sampling.

6.4 Optimization Considerations

Gradient stability benefits from clipping u within $[\varepsilon, 1-\varepsilon]$. Gate architectures can share parameters across layers to reduce overhead, and entropy regularizers discourage gate collapse (always-on or always-off behavior).

7 Theoretical Analysis

7.1 Monotone Uncertainty Propagation

Theorem 7.1 (Uncertainty Propagation)**.** *Let $h^{(l+1)} = f_l(h^{(l)}, p^{(l)}_{\text{gated}})$ denote the representation update at layer l and $u^{(l)}$ the uncertainty emitted by that layer. Suppose aggregation uses a monotone non-decreasing function f. Then the final uncertainty satisfies*

$$u_{\text{final}} \geq \max_{0 \leq l \leq L} u^{(l)}. \tag{22}$$

Proof of Theorem 1. We prove that $u_{\text{final}} \geq \max_{0 \leq l \leq L} u^{(l)}$ for monotone aggregation function f.

Step 1: By definition of the aggregation function:

$$u_{\text{final}} = f(u^{(0)}, u^{(1)}, \ldots, u^{(L)}) \tag{23}$$

Step 2: Let $u_{\max} = \max_{0 \leq l \leq L} u^{(l)}$ and let $l^* \in \{0, \ldots, L\}$ be the layer achieving this maximum:

$$u^{(l^*)} = u_{\max} \tag{24}$$

Step 3: Consider the input vector to f where we set all entries except l^* to zero:

$$v_{\min} = (0, 0, \ldots, \underbrace{u_{\max}}_{l^*}, \ldots, 0) \in [0, 1]^{L+1} \tag{25}$$

Step 4: Since f is monotone non-decreasing in each argument and $u^{(l)} \geq 0$ for all l:

$$f(u^{(0)}, \ldots, u^{(L)}) \geq f(v_{\min}) = f(0, \ldots, u_{\max}, \ldots, 0) \tag{26}$$

Step 5: For specific aggregation functions:

- **Max aggregator:** $f = \max$

$$f(0, \ldots, u_{\max}, \ldots, 0) = u_{\max} \tag{27}$$

- **Mean aggregator:** $f = \text{mean}$

$$f(u^{(0)}, \ldots, u^{(L)}) = \frac{1}{L+1} \sum_{l=0}^{L} u^{(l)} \geq \frac{u_{\max}}{L+1} \tag{28}$$

  However, this is a weaker bound. To achieve $u_{\text{final}} \geq u_{\max}$, we require f to satisfy:

$$f(\ldots, u_{\max}, \ldots) \geq u_{\max} \tag{29}$$

  which holds for $f = \max$ and learned monotone aggregators with appropriate initialization.

- **Learned aggregator:** Train f to satisfy $f(v) \geq \max_i v_i$ via architectural constraints (e.g., max-pooling layer followed by learned transformation).

Step 6: Therefore, for conservative aggregation ($f = \max$):

$$u_{\text{final}} = \max(u^{(0)}, \ldots, u^{(L)}) = \max_{0 \leq l \leq L} u^{(l)} \tag{30}$$

Step 7: Residual connections preserve this property because epistemic gates multiply probability distributions rather than subtract scalars. If layer l has high uncertainty $u^{(l)} \approx 1$, the gated distribution $p^{(l)}_{\text{gated}} \approx$ uniform. Subsequent layers cannot "undo" this uncertainty without evidence.

Step 8: Thus, $u_{\text{final}} \geq \max_l u^{(l)}$ as required. □

Corollary 7.2. *If any layer emits high uncertainty ($u^{(l)}$ close to 1), the final output uncertainty cannot collapse to zero unless all subsequent layers emit perfect certainty ($u = 0$), which is unlikely under VARO training that penalizes miscalibrated confidence.*

7.2 Calibration Under VARO

Theorem 7.3 (Calibration Under VARO)**.** *Consider training an epistemic softmax model with stochastic gradient descent on the loss:*

$$\mathcal{L}(\theta) = \mathcal{L}_{CE}(p_{gated}(\theta), y) + \lambda \|u(\theta) - u^*\|_2^2 \tag{31}$$

where θ are model parameters, p_{gated} is the gated distribution from Algorithm 1, u is predicted uncertainty, and u^* is target uncertainty.

Assumptions:

*(A1) **L-smoothness:** The loss $\mathcal{L}$ is L-smooth, i.e., for all θ, θ':*

$$\|\nabla\mathcal{L}(\theta) - \nabla\mathcal{L}(\theta')\| \leq L\|\theta - \theta'\| \tag{32}$$

(A2) ***Unbiased uncertainty targets:*** *The target uncertainty u^* satisfies:*

$$\mathbb{E}[u^* \mid x] = u_{true}(x) \tag{33}$$

where $u_{true}(x)$ is the true epistemic uncertainty for input x.

(A3) ***Bounded gradient variance:*** *For stochastic gradients g_t:*

$$\mathbb{E}[\|g_t - \nabla\mathcal{L}(\theta_t)\|^2] \leq \sigma^2 \tag{34}$$

(A4) ***Robbins-Monro learning rate:*** *The learning rate η_t satisfies:*

$$\sum_{t=1}^{\infty} \eta_t = \infty \quad \textit{and} \quad \sum_{t=1}^{\infty} \eta_t^2 < \infty \tag{35}$$

Under assumptions (A1)-(A4), the expected calibration error (ECE) decreases:

$$\mathbb{E}[ECE_{t+1}] \leq \mathbb{E}[ECE_t] - \eta_t \lambda c_1 + \eta_t^2 c_2 \tag{36}$$

where:

$$c_1 = 2\mathbb{E}[\|\nabla_u ECE\|^2] \quad \textit{(gradient of ECE w.r.t. uncertainty)} \tag{37}$$

$$c_2 = L\sigma^2 + \frac{L^2}{2} \quad \textit{(smoothness and variance terms)} \tag{38}$$

Corollary: *Choosing $\eta_t = 1/t$ yields:*

$$\mathbb{E}[ECE_T] \leq \mathbb{E}[ECE_0] - \lambda c_1 \log(T) + O(1) \tag{39}$$

Thus ECE decreases logarithmically with training steps T.

Proof sketch. **Step 1:** By L-smoothness (A1) and the SGD update rule $\theta_{t+1} = \theta_t - \eta_t g_t$:

$$\mathbb{E}[\mathcal{L}(\theta_{t+1})] \leq \mathbb{E}[\mathcal{L}(\theta_t)] - \eta_t \mathbb{E}[\|\nabla\mathcal{L}(\theta_t)\|^2] + \frac{L\eta_t^2}{2}\mathbb{E}[\|g_t\|^2] \tag{40}$$

Step 2: The VARO term $\lambda\|u - u^*\|^2$ provides gradient:

$$\nabla_\theta \left(\lambda\|u - u^*\|^2\right) = 2\lambda(u - u^*)\nabla_\theta u \tag{41}$$

Step 3: By assumption (A2), $\mathbb{E}[u - u^*]$ measures calibration error, which correlates with ECE. Specifically, ECE is defined as:

$$\text{ECE} = \mathbb{E}_{B \in \text{bins}} \left| \frac{1}{|B|} \sum_{i \in B} \mathbb{1}[y_i = \hat{y}_i] - \bar{c}_B \right| \tag{42}$$

where $\bar{c}_B$ is the average confidence in bin B.

The VARO loss directly optimizes $\|u - u^*\|^2$, and under proper calibration ($u^* = 1 - \text{confidence}$), minimizing this term reduces the gap between confidence and accuracy, thereby reducing ECE.

Step 4: Substituting (A3) and applying standard SGD convergence analysis (see [2]) with (A4) yields the stated bound.

Full proof requires technical analysis of ECE geometry [10] and is deferred to supplementary material. □

7.3 Computational Complexity

Standard Transformer Cost

For L layers, sequence length n, and hidden dimension d:

$$\text{Attention:} \quad O(L \cdot n^2 \cdot d) \quad \text{(all layers)} \tag{43}$$

$$\text{Feed-forward:} \quad O(L \cdot n \cdot d^2) \quad \text{(all layers)} \tag{44}$$

$$\text{Total:} \quad O(L \cdot n^2 \cdot d + L \cdot n \cdot d^2) \tag{45}$$

Epistemic Softmax Overhead

Each gate Q_i is an MLP with hidden size k:

$$\text{Forward:} \quad O(d \cdot k + k \cdot 1) = O(d \cdot k) \quad \text{per invocation} \tag{46}$$

$$\text{Memory:} \quad O(d \cdot k) \quad \text{parameters per gate} \tag{47}$$

Level-by-Level Analysis

Level 1 (Output-only)

- **Gates:** 1 Q_1 gate + 1 Q_2 gate at output layer
- **Operations:** $2 \times O(d \cdot k)$ per token $= O(n \cdot d \cdot k)$
- **Overhead:**

$$\frac{O(n \cdot d \cdot k)}{O(L \cdot n^2 \cdot d)} = \frac{k}{L \cdot n} \tag{48}$$

- **Numerical estimate:** For $k = d/4$, $L = 12$, $n = 512$:

$$\text{Overhead} \approx \frac{d/4}{12 \times 512} \approx 0.04\% \quad \text{(negligible)} \tag{49}$$

Level 2 (Attention + Output)

- **Gates per layer:**
  - H attention heads $\times$ 1 Q_1 gate each $= H \times O(d \cdot k)$
  - 1 Q_2 gate for head aggregation $= O(d \cdot k)$
  - Total per layer: $O(H \cdot d \cdot k)$
- **Gates across L layers:** $L \times O(H \cdot n \cdot d \cdot k)$
- **Plus output gates:** $O(n \cdot d \cdot k)$
- **Total overhead:**

$$\frac{O(L \cdot H \cdot n \cdot d \cdot k)}{O(L \cdot n^2 \cdot d)} = \frac{H \cdot k}{n} \tag{50}$$

- **Numerical estimate:** For $H = 8$, $k = d/4$, $n = 512$:

$$\text{Overhead} = \frac{8 \cdot (d/4)}{512} = \frac{2d}{512} \approx 2\% \quad (\text{for } d = 512) \tag{51}$$

- **Range:** 2-3% depending on d/n ratio

Level 3 (Full Fractal)

- **Additional gates:** MoE routers, adaptive attention mechanisms, etc.
- **Estimate:** Approximately 1.5× Level 2 overhead
- **Total overhead:** ≈ 4-5%

Parameter Overhead

Without parameter sharing:

- Q_1 gate: $d \times k + k \times 1 \approx d \cdot k$ parameters
- Q_2 gate: $d \times k$ parameters
- **Level 1:** 2 gates = $2dk$
  - For $k = d/4$: $2d \cdot (d/4) = d^2/2$ parameters
  - Baseline has $\approx 12d^2$ (for $L = 12$ layers × projection matrices)
  - Overhead: $(d^2/2)/(12d^2) \approx 4\%$ parameters
- **Level 2:** $2L(H + 1)$ gates
  - For $L = 12$, $H = 8$: $2 \cdot 12 \cdot 9 = 216$ gates
  - Parameters: $216 \cdot dk = 54d^2$ (for $k = d/4$)
  - Overhead: $54d^2/(12d^2 \cdot L) \approx 38\%$ parameters (significant!)

With parameter sharing (recommended):

- Share Q_1 weights across all heads
- Share Q_2 weights across all layers
- **Level 2 parameters:** Just 2 gates = $2dk$
- **Overhead:** $< 5\%$ even for Level 3

Memory-Computation Tradeoff

- **Without sharing:** Higher memory, same compute per forward pass
- **With sharing:** Lower memory, same compute per forward pass
- **Recommendation:** Share Q_1 across heads within a layer; unique Q_2 per layer to capture layer-specific consensus patterns

Empirical Measurement

To be added after implementation:

Model	Latency (ms/token)	Memory (GB)	Overhead
Baseline	X	Y	—
Level 1	$X + \delta_1$	Y	+0.5%
Level 2	$X + \delta_2$	$Y + \epsilon$	+2.5%
Level 3	$X + \delta_3$	$Y + \epsilon$	+4.5%

7.4 Robustness to Gate Collapse

Gate collapse occurs when Q_1 or Q_2 saturate at 0 or 1. Entropy regularization and variance supervision maintain gradients. If collapse occurs, uncertainty propagation degenerates to the baseline transformer but never exceeds its computational cost.

8 Experimental Design

8.1 Datasets and Metrics

Dataset	Task	Metric	Baseline Expected
TruthfulQA [17]	Hallucination	% truthful answers	40%
TempQuestions [21]	Temporal generalization	Accuracy	30%
Consistency [7]	Paraphrase consistency	Accuracy variance	15%
MMLU [11]	Calibration	ECE, Brier score	0.15 ECE
Synthetic OOD [20]	Uncertainty detection	AUROC (unc vs. error)	0.60

Table 1: Datasets and metrics for evaluating Aletheion.

8.2 Models and Ablations

Table 2: Projected performance improvements across Aletheion levels.[†]

Model	**TruthfulQA**	**ECE**	**Hallucination Rate**	**Unc–Error Corr.**
Baseline Transformer	40%	0.15	60%	0.30
+ Temperature Scaling	42%	0.13	58%	0.35
Aletheion Level 1	48%	0.10	45%	0.60
Aletheion Level 2	52%	0.08	38%	0.70
Aletheion Level 3	58%	0.06	25%	0.80

[†]These are theoretical projections based on architectural analysis and prior uncertainty quantification literature. Empirical validation is ongoing and results may vary. The baseline transformer achieves stated performance on respective benchmarks [17]. Projected improvements assume successful VARO training and optimal λ tuning.

Ablations include removing Q_2, varying λ, testing alternative uncertainty aggregators, and evaluating abstention policies. Additional diagnostics compute selective prediction curves, coverage-controlled risk, and retrieval triggers under uncertainty.

8.3 Evaluation Protocol

1. Pretrain baseline model on open-source corpora.
2. Fine-tune Levels 1–3 using identical data, enabling incremental comparisons.
3. Measure calibration via ECE, Brier score, and reliability diagrams.
4. Report computational overhead (FLOPs, latency) for inference.
5. Evaluate abstention quality using selective prediction curves and coverage risk.

8.4 Risk and Mitigation

Potential failure includes gate collapse and miscalibrated u^*. We monitor entropy of gate outputs, apply adaptive λ, and integrate human-in-the-loop review for high uncertainty outputs.

9 Discussion

9.1 Why Fractal Works

Self-similarity enforces consistent epistemic reasoning across all scales of the transformer. Local attention gates prevent uncertainty collapse at early layers, while global output gates maintain calibrated predictions. The hierarchy mirrors residual networks and multi-scale reasoning observed in compositional attention structures.

9.2 Limitations and Open Questions

We categorize open questions by urgency and expected outcomes:

Critical (blocking production deployment)

Q1: Gate collapse **Question:** Can Q_1 or Q_2 degenerate to always-on (≈ 1) or always-off (≈ 0)?
Expected answer: Unlikely with entropy regularization.
Evidence: Similar gating mechanisms (e.g., LSTM gates [12], attention gates in transformers) avoid collapse when trained with proper regularization.
Validation strategy:

- Monitor gate entropy during training: $H(Q_i) = -\sum_j q_{ij} \log q_{ij}$
- Apply gradient penalties if $\mathbb{E}[H(Q_i)] < \theta_{\min}$
- Use initialization bias: initialize Q_i to output ≈ 0.7 (confident but not saturated)

Q2: VARO-RLHF interaction **Question:** Does preference optimization (DPO/RLHF) after VARO training collapse epistemic gates?
Hypothesis: Sequential training (VARO $\rightarrow$ freeze gates $\rightarrow$ RLHF) preserves calibration.
Alternative approach: Joint training with multi-objective loss:

$$\mathcal{L} = \mathcal{L}_{\text{RLHF}} + \lambda_1 \mathcal{L}_{\text{VARO}} + \lambda_2 H(\text{gates}) \tag{52}$$

Requires: Empirical validation on standard RLHF benchmarks (HH-RLHF, Anthropic Helpful-Harmless).

High priority (affects performance)

Q3: Optimal λ **Question:** How to set VARO weight λ across datasets and model scales?
Current approach: Grid search $\lambda \in \{0.01, 0.1, 1.0\}$
Expected: λ scales inversely with model size (larger models need smaller λ to avoid overwhelming cross-entropy signal).
Future: Meta-learning λ or adaptive λ_t schedule (e.g., cosine annealing).

Q4: Uncertainty aggregation **Question:** Which function f (max/mean/learned) works best?
Hypothesis:

- max: best for safety-critical applications (conservative, never under-reports uncertainty)
- mean: best for balanced performance-calibration tradeoff
- Learned: best asymptotic performance but requires uncertainty-labeled data

Ablation study: Test all three on TruthfulQA, compare ECE and selective prediction curves.

Q5: Scaling to 175B+ parameters **Question:** Do uncertainty gains persist at GPT-3 scale (175B) and beyond?

Expected: Yes, since epistemic failures worsen with scale [17]. Larger models hallucinate more complex fabrications.

Challenge: Computational cost of training gates on 175B model ($\approx 5\%$ overhead still substantial).

Mitigation strategy:

1. Train smaller epistemic model (e.g., 7B with gates)
2. Distill uncertainty behavior to large base model (175B)
3. Use LoRA-style efficient fine-tuning for gates only

Medium priority (future work)

Q6: Multimodal extension **Question:** How to apply epistemic softmax to vision-language models?

Approach: Gated cross-attention between vision and text modalities.

Application: Reduce hallucination in image captioning (e.g., detecting when visual features insufficient to support textual claim).

Q7: Epistemic chain-of-thought **Question:** Can the model reason explicitly about its own uncertainty?

Example: "I'm uncertain about X because evidence Y conflicts with evidence Z."

Requires: Training on uncertainty-annotated reasoning traces (expensive to collect).

Q8: Adversarial robustness **Question:** Can adversarial inputs fool epistemic gates?

Risk: Adversary crafts input that looks out-of-distribution but gates output $u \approx 0$ (false confidence).

Defense: Adversarial training specifically on gates:

$$\max_{\|\delta\|<\epsilon} \mathcal{L}_{\text{VARO}}(x+\delta) \tag{53}$$

Q9: Integration with Consistency Training **Question:** How does Aletheion interact with consistency training [9]?

Hypothesis: Complementary and synergistic. Aletheion provides architectural epistemic gates while consistency training enforces paraphrase invariance.

Approach: Combined loss function:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_1 \mathcal{L}_{\text{VARO}} + \lambda_2 \mathcal{L}_{\text{consistency}} \tag{54}$$

where $\mathcal{L}_{\text{consistency}}$ penalizes output variance across paraphrased prompts.

Expected outcome: Models that are both *calibrated* (low ECE via VARO) and *robust to paraphrases* (low variance via consistency training). This combination may be particularly effective against sycophancy (Failure Mode 3, Section 3).

Experimental validation:

- Baseline: Standard transformer
- + Consistency training only
- + Aletheion only
- + Both (combined)

Measure: TruthfulQA accuracy, paraphrase consistency, ECE.

Low priority (philosophical)

Q10: What is the correct formalization of "epistemic uncertainty" for autoregressive language models?

Q11: Can epistemic gates enable true "I don't know" responses (or just calibrated low confidence)?

Q12: Relationship to human metacognition and confidence calibration?

9.3 Philosophical Implications

Softmax acts as a forced decision rule; epistemic softmax enables "aware" decisions where the model can admit ignorance. This architectural humility aligns with AI safety principles emphasizing deferment when knowledge is insufficient [13].

9.4 Connection to ARC-AGI

The Abstraction and Reasoning Corpus (ARC) tests few-shot abstract reasoning where current LLMs underperform (approximately 5% vs. 85% human accuracy) [4]. Epistemic gating addresses ARC's challenges: (1) ambiguity detection via Q_2 detecting conflicting hypotheses, (2) abstention through uncertainty-driven refusal, and (3) hierarchical reasoning by mirroring ARC's multi-level abstractions. We hypothesize Level 3 Aletheion reduces catastrophic failures on ARC-style tasks by refusing uncertain answers and requesting clarification.

9.5 When Aletheion Fails: Failure Mode Analysis

Epistemic softmax is not a panacea. We identify scenarios where the architecture cannot provide guarantees:

1. Irreducible Aleatoric Uncertainty

Problem: Inherently random processes (dice rolls, quantum events, inherently unpredictable future events).

Why Aletheion fails: No amount of information reduces uncertainty. Epistemic gates cannot distinguish aleatoric from epistemic uncertainty without explicit supervision.

Example: "Will this fair coin land heads?"

- True answer: $p(\text{heads}) = 0.5$ with $u = 1$ (maximal uncertainty, but aleatoric)
- Aletheion: May output $p \approx 0.5$ but u may be miscalibrated

Mitigation: Distinguish epistemic vs. aleatoric in u^* supervision signal. For known aleatoric scenarios, supervise with $u^* = 1$ but flag as non-reducible.

2. Adversarial Attacks on Gates

Problem: Adversary crafts inputs that fool Q_1/Q_2 into outputting low uncertainty despite the input being adversarial.

Example: Input x_{adv} that appears in-distribution to gates but is actually crafted to trigger specific (wrong) behavior.

Why Aletheion fails: Gates are neural networks, thus vulnerable to adversarial examples. Standard adversarial training does not explicitly protect gates.

Mitigation: Adversarial training specifically targeting gates:

$$\mathcal{L}_{\text{adv}} = \max_{\|\delta\| < \epsilon} \mathcal{L}_{\text{VARO}}(x + \delta) \tag{55}$$

subject to $\|\delta\|_\infty < \epsilon$ (small perturbation).

3. Specification Gaming

Problem: RLHF may incentivize *hiding* uncertainty to maximize reward.

Example: If reward model favors confident answers, the model learns "confident wrong answer gets higher reward than uncertain right answer."

Why Aletheion fails: Preference optimization doesn't value calibration by default. Gates may learn to always output low u to please the reward model.

Mitigation: Include calibration metrics in reward model:

$$R(\text{response}) = R_{\text{preference}}(\text{response}) - \lambda \cdot \text{ECE}(\text{response}) \tag{56}$$

Penalize miscalibrated confidence directly in the reward.

4. Catastrophic Forgetting During Fine-Tuning

Problem: Fine-tuning on narrow distribution may collapse gates.

Example: Fine-tune on medical QA dataset → gates learn to always be confident on medical queries, but forget to trigger uncertainty on non-medical queries.

Why Aletheion fails: Standard fine-tuning doesn't preserve epistemic behavior outside the fine-tuning distribution.

Mitigation:

1. Continual learning techniques: Elastic Weight Consolidation (EWC), PackNet
2. Maintain separate uncertainty validation set (held-out diverse queries)
3. Regularize gates during fine-tuning:

$$\mathcal{L}_{\text{finetune}} = \mathcal{L}_{\text{task}} + \lambda \|Q_{\text{new}} - Q_{\text{old}}\|^2 \tag{57}$$

5. Computational Budget Constraints

Problem: Production systems may not afford 4-5% overhead.

Example: Real-time chatbot with strict latency requirements (e.g., ¡50ms response time).

Why Aletheion fails: Even small overhead may violate SLA (service level agreement).

Mitigation:

1. Use Level 1 (output-only, ¡1% overhead)
2. Conditional gating: Only activate gates for queries flagged as potentially uncertain (via cheap heuristic)
3. Model distillation: Train small "epistemic triage" model; use full Aletheion only when triage indicates uncertainty

6. Missing Ground Truth for u^*

Problem: Many domains lack uncertainty labels.

Example: Creative writing tasks have no "correct" answer, thus u^* is undefined.

Why Aletheion fails: VARO requires u^* supervision. Without it, gates may not learn meaningful uncertainty.

Mitigation:

1. Use unsupervised methods (head variance, self-consistency) as fallback
2. Human annotation for calibration set (expensive but one-time cost)
3. Transfer uncertainty behavior from related domains (e.g., factual QA → creative writing)

Summary of Failure Modes

Deployment recommendation: Deploy Aletheion with continuous monitoring of gate behavior and maintain a held-out uncertainty validation set to detect failures early. Start with Level 1 in production; upgrade to Level 2/3 as compute budget allows.

Failure Mode	Severity	Mitigation?	Blocks Deploy?
Aleatoric uncertainty	Low	Partial (better u^*)	No
Adversarial gates	Medium	Yes (adv. training)	No
Specification gaming	High	Yes (calibration rewards)	Maybe
Catastrophic forgetting	High	Yes (continual learning)	No
Compute constraints	Medium	Yes (Level 1, distill)	Maybe
Missing u^* labels	Medium	Yes (unsupervised)	No

Table 3: Failure scenarios and recommended mitigations for Aletheion.

10 Related Work

Aletheion builds on transformer advancements [22, 3], scaling studies in language models, hallucination analyses [13, 17], and uncertainty estimation techniques including Bayesian approximations and deep ensembles [1, 8, 16]. Recent work on eliciting model uncertainty underscores the need for architectural primitives rather than post-hoc estimates [18, 14, 19].

11 Conclusion

We introduced Aletheion, a fractal epistemic architecture that replaces all softmax operations with uncertainty-aware epistemic softmax. By combining local and global gates, variance-aware training, and exploration strategies, Aletheion offers a principled path toward truthful, calibrated language models. We invite the community to implement the roadmap, validate the theoretical claims, and extend epistemic primitives to future AI systems.

References

[1] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *International Conference on Machine Learning*, 2015.

[2] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[3] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 2020.

[4] François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

[5] Aletheion Research Collective. Aletheion llm fundamentals. Internal documentation, 2024. `https://github.com/aletheion-llm`.

[6] Aletheion Research Collective. Operational failure modes of large language models. Internal documentation, 2024. `https://github.com/aletheion-llm`.

[7] Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Sch"utze, and Yoav Goldberg. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031, 2021.

[8] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, 2016.

[9] Google DeepMind Safety Research. Consistency training could help limit sycophancy and jailbreaks. DeepMind Safety Research Blog, November 2024. Accessed: 2024-11-04.

[10] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330, 2017.

[11] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *International Conference on Learning Representations*, 2021.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[13] Zhehui Ji, Tianyi Xu, Bo Wang, Wei Chao, Kam-Fai Wong, Hongyuan Zha, and Xiaodong He. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 2023.

[14] Saurav Kadavath et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

[15] Aditi Kamath, Robin Jia, and Percy Liang. Selective question answering under domain shift. In *International Conference on Machine Learning*, 2020.

[16] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, 2017.

[17] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

[18] Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty. *arXiv preprint arXiv:2205.14334*, 2022.

[19] Andrey Malinin et al. Uncertainty estimation in autoregressive sequence models. *International Conference on Learning Representations*, 2021.

[20] Yaniv Ovadia et al. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in Neural Information Processing Systems*, 2019.

[21] Shangqing Tu, Jifan Yu, Xiaozhi Wang, Juanzi Li, and Lei Hou. Tempquestions: A benchmark for temporal question answering. *arXiv preprint arXiv:2305.17173*, 2023.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.