# Dual Embedding Cross-Check (DEC): A Lightweight Method for Improving LLM Reliability Using Static Semantic Anchors

Raghvendran Kumar / Raghav Kumar — Independent Researcher

Raghav Kumar raghavk.azp@outlook.com / 091 9677-311-245

Created at & on: Chennai, India | Dec 12, 2025
Last Update: Chennai, India | Dec 13, 2025

# Abstract

Large Language Models (LLMs) have achieved remarkable capability, yet they continue to suffer from semantic drift and hallucination—particularly in long-form reasoning, open-ended generation, and long-context tasks. Existing mitigation strategies such as retrieval augmentation, self-consistency sampling, classifier-based hallucination detection, and uncertainty estimation are often costly, model-dependent, or insufficiently interpretable.

This paper introduces **Dual Embedding Cross-Check (DEC)**, a lightweight, external, and model-agnostic method that evaluates semantic stability by comparing an LLM's internal contextual embedding trajectory against a parallel trajectory constructed from static word embeddings (e.g., Word2Vec, GloVe, FastText). DEC computes a geometric drift score via cosine divergence between the two semantic paths, where elevated drift serves as an early indicator of instability or hallucination.

Beyond detection, DEC functions as an **external executive-control layer**, analogous to a prefrontal cortex, providing metacognitive oversight of reasoning without modifying model weights or architecture. The framework further enables **geometric interpretability**, producing visual semantic maps that localize drift at the token and segment level, offering insight into where and how reasoning deviates.

Importantly, DEC provides immediate benefits when applied to already-trained large-scale systems, including improved effective accuracy, identification of inefficient reasoning paths, drift-aware inference control, and cost-aware optimization—without retraining. By stabilizing semantic trajectories rather than increasing parameter count, DEC suggests a path toward more reliable, interpretable, and compute-efficient LLM systems, potentially allowing smaller models to approach the reliability of significantly larger ones.

# Introduction

Large Language Models continue to scale, achieving impressive generative, reasoning, and multimodal abilities. However, increased capability has not eliminated a core weakness: semantic drift leading to hallucination. When an LLM's internal reasoning path deviates from factual or semantic grounding, outputs degrade in subtle or catastrophic ways.

Modern solutions attempt to reduce hallucination through:

- Retrieval-Augmented Generation (RAG)
- Multi-sample self-consistency
- Temperature adjustments
- RLHF / RLAIF
- Domain-specific classifiers
- Specialized alignment models

These techniques either incur significant compute overhead, require retraining, or fail to detect drift until after hallucination has already occurred.

## A Missing Ingredient: External Reference Stability

LLMs operate in contextual embedding spaces. Every token is encoded relative to the prompt, attention pattern, and internal state—meaning the "coordinates" of a word shift continuously.

Static embeddings (e.g., Word2Vec), by contrast, encode stable semantic meaning independent of context.

This raises a critical insight:

If a model's dynamic embedding path diverges significantly from a stable reference semantic path, it signals an impending hallucination.

## Contribution

This paper introduces Dual Embedding Cross-Check (DEC), a new paradigm for model verification that:

1. Extracts the LLM's internal token embeddings
2. Maps each token to its nearest static embedding vector

3. Constructs two semantic trajectories
4. Computes a drift score using cosine divergence
5. Flags output instability when drift exceeds a threshold

DEC is:

- Model-agnostic
- Zero retraining
- Extremely cheap
- Interpretable
- Effective before hallucinations fully manifest

Most importantly, DEC provides a pathway for small models to behave more reliably without additional compute, offering potential efficiency gains across training, inference, and deployment.

# Background

## Contextual vs. Static Embeddings

Contextual embeddings (LLMs):

- Dynamic
- Dependent on prompt and internal state
- High dimensional but unstable under long reasoning
- Can drift subtly due to attention dilution

Static embeddings (Word2Vec/GloVe/FastText):

- Fixed vectors
- Stable global semantic structure
- Preserve word-level relationships
- Immune to prompt-induced drift

## Semantic Drift

Semantic drift occurs when the LLM's embedding trajectory strays from a semantically reasonable direction. Drift is often the precursor to hallucination. DEC's premise is that drift can be quantified.

## Cosine Similarity as Trajectory Comparison

Cosine similarity offers a robust measure of directional consistency between:

- LLM's sequence embedding
- Static-compass sequence embedding

When the cosine similarity drops, it indicates internal instability.

# The DEC Method

DEC constructs two parallel semantic paths for every generated sequence.

Imagine the LLM as a GPS system: detailed, powerful, but occasionally confused in poor coverage conditions.
Static embeddings act as the compass: simple, stable, always pointing in the same direction.

When GPS and compass disagree sharply, you know you're off-course.

## Formal Definition

**3.1 Formal Definition**

Given tokens $t_1, \ldots, t_n$:

**LLM contextual embeddings**

$$E_{LLM}(t_i)$$

**Static embeddings (nearest-neighbor mapped)**

$$E_{static}(t_i)$$

**Paths**

$$P_{LLM} = (E_{LLM}(t_1), \ldots, E_{LLM}(t_n))$$

$$P_{static} = (E_{static}(t_1), \ldots, E_{static}(t_n))$$

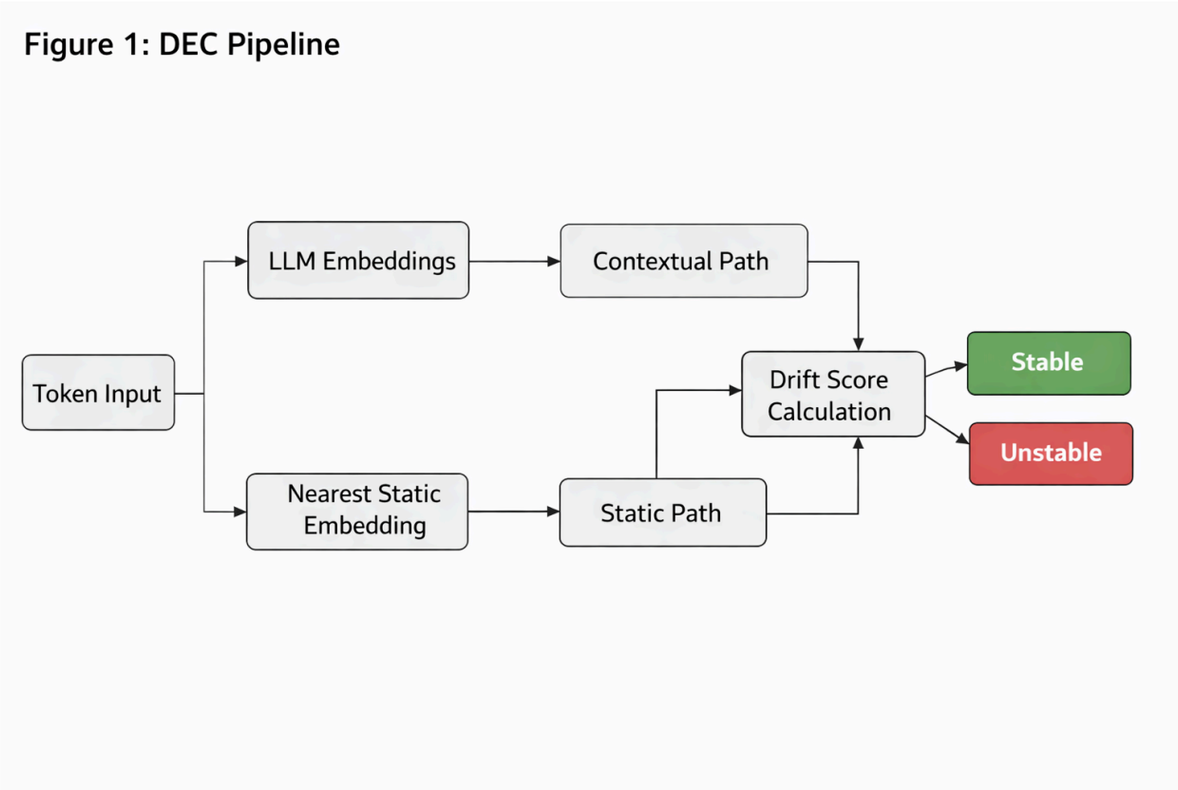
**Drift Score**

$$D = 1 - \cos(P_{LLM}, P_{static})$$

When $D > \theta$ (threshold), the output is considered unstable.

# Algorithm

Figure 1: DEC Pipeline (diagram description)




A block diagram showing:

- Token → LLM embedding
- Token → nearest static embedding
- Two parallel paths
- Drift score calculator
- "Stable / Unstable" decision layer

# Pseudocode

```
def dec_compare(llm_embeddings, static_vectors, threshold=0.25):
    mapped = []
    for vector in llm_embeddings:
        nearest = find_closest_word_vector(vector, static_vectors)
        mapped.append(nearest)

    llm_path = aggregate(llm_embeddings)
    static_path = aggregate(mapped)
```

```
    drift = cosine_distance(llm_path, static_path)

    if drift > threshold:
        return "Potential Hallucination", drift

    return "Stable", drift
```

# Benefits of DEC

- **Zero retraining**

  Works with any model: GPT, LLaMA, Mistral, Phi, Claude-style, etc.

- **Extremely low compute**

  Most work is nearest-neighbor lookup; can be optimized using FAISS or HNSW.

- **Real-time drift detection**

  Signals hallucination *before* it happens.

- **Stabilizes smaller models**

  DEC can make a 20B model behave more reliably like a 70–80B model by providing grounding during inference.

- **External and independent**

  DEC does not rely on:

  - logits
  - attention weights
  - uncertainty estimation
  - prompt constraints

  It is a separate measurement system — a compass distinct from the GPS.

- **Increased interpretability**

  Drift curves visually illustrate when a model veers off course.

# What DEC Visualizes (Beyond Drift Curves)

- Exact token-level drift

  Because DEC compares dynamic and static vectors per token, you can produce:

  - a heatmap of which tokens diverged the most
  - a timeline of semantic instability
  - segment-specific drift spikes

  This directly highlights:

  - which phrase caused the model to derail
  - where context got diluted
  - which reasoning step was unstable

  No current hallucination method gives this clarity.

- Geometric trajectory plots (LLM vs Static Manifold)

  Imagine this diagram:

  - The LLM's embedding path is a curvy, wandering blue line
  - The static embedding path is a straight, stable red line
  - Divergence is the widening gap between them

  This is essentially a geometric map of the model's "thought path."

  This becomes interpretability gold.

  You can:

  - inspect where the curve bends incorrectly
  - detect loops / oscillations
  - compare drift shapes across prompts
  - visualize attention disruptions indirectly

This is extremely useful for debugging reasoning.

- Multi-dimensional projections (PCA / UMAP)

  When you reduce both embedding paths to 2D or 3D:

  - You get smooth trajectories when the model is stable
  - Jagged, chaotic paths when the model hallucinates

  A visualization could literally show:

  - a smooth arc (good reasoning)
  - a sharp zigzag (context break)
  - spirals (self-contradiction loops)

  This is interpretability at a level that current LLM systems don't provide.

- Segment Clustering

  If you cluster segments of reasoning:

  - Stable segments cluster tightly
  - Hallucination-prone segments scatter far away

  This reveals:

  - which parts of the answer were on solid footing
  - which were unstable or fabricated

  Imagine being able to point at a part of the answer and say:
  "This paragraph is highly unreliable — the vector path drifted 3× more than baseline."

  That is far beyond just "hallucination = yes/no."

## A future possibility: “Semantic Drift Maps”

DEC naturally enables visual drift maps, like weather maps for LLM reasoning. Layers could include:

- drift intensity
- drift direction
- drift velocity (how fast the embedding diverges)
- drift curvature

This would give unprecedented insight into:

- pathological reasoning
- misleading context
- misalignment in hidden layers

It becomes a real interpretability instrument.

## Why this is better than attention-head visualization

Attention maps show *where* the model attends, not *whether the reasoning is stable*.

DEC shows:

- the geometry
- the trajectory
- the semantic deformation
- the divergence zone

In other words:

- ❌ Attention shows pointing
- ✔ DEC shows drift
- ✔ DEC shows trajectory
- ✔ DEC shows semantic consistency

**These are far more useful indicators of hallucination.**

# Comparison to Existing Methods

## Retrieval-Augmented Generation (RAG)

- Heavy infra
- Requires data availability
- Mitigates hallucination but does not detect drift

DEC = no retrieval needed.

## Self-Consistency

- Requires multiple samples → expensive
- Still internal to model

DEC = single pass, external signal.

## Uncertainty Estimation

- LLM confidence is poorly calibrated

DEC = directly measures semantic divergence.

## Classifier-Based Detection

- Requires training data
- Domain-dependent

DEC = universal, domain-agnostic.

### Table 1: Comparison of Hallucination Mitigation Methods

| Method | Extra Compute | Real-Time? | External? | Detects Drift? | No Retraining? |
|---|---|---|---|---|---|
| RAG | High | No | No | Weak | No |
| Self-Consistency | Very High | No | No | No | Yes |
| Uncertainty | Low | Yes | No | Weak | Yes |
| Classifier | Medium | Yes | No | Weak | No |
| DEC (proposed) | Low | Yes | Yes | Yes | Yes |

### Evaluation Strategy (Planned)

While this work primarily focuses on introducing the DEC framework, the method naturally lends itself to empirical validation. Planned evaluation strategies include:

- **Correlation analysis** between DEC drift scores and factual correctness on hallucination benchmarks (e.g., TruthfulQA subsets)

- **Token-level drift visualization** to identify whether drift precedes incorrect or speculative generations

- **Before/after comparisons** using drift-aware truncation or regeneration to measure changes in effective accuracy and verbosity

- **Long-context stress tests** to observe cumulative drift behavior over extended reasoning sequences

These evaluations can be conducted using open-source LLMs and static embedding spaces on modest hardware, making DEC reproducible and accessible for independent validation.

# Implications for Training and Scaling

1. Faster, cheaper training

   DEC could act as:

   - A training-time regularizer
   - Drift penalty for RLHF/RLAIF
   - A debugging tool for long-context training failures

2. Smaller models can achieve higher reliability

   This is the big one.

   If a 20B model can maintain semantic stability comparable to an 80B model, we unlock:

   - cheaper inference
   - lighter deployment
   - more accessible LLM tech

3. Reduced catastrophic errors

   Real-time drift monitoring prevents spirals into nonsense.

# Further Potential and Roadmap

1. **DEC as an Executive-Control Layer (Artificial Prefrontal Cortex)**
   Large Language Models exhibit strong generative and associative capabilities, but they lack an explicit mechanism for semantic self-regulation. Once a reasoning trajectory begins to drift, the model has limited ability to detect or correct that deviation internally.

In human cognition, this role is served by the **prefrontal cortex (PFC)**, which performs executive functions such as monitoring coherence, detecting errors, suppressing impulsive reasoning, and maintaining goal alignment. DEC plays an analogous role for LLMs.

In the DEC framework, the LLM acts as the **generative core**, producing dynamic contextual embeddings, while DEC functions as an **external executive monitor**, continuously evaluating semantic stability by comparing the model's internal trajectory against a stable semantic reference space.

This separation enables:

- early detection of unstable reasoning
- prevention of runaway semantic drift
- maintenance of coherence over long contexts
- metacognitive oversight without modifying the model itself

Unlike internal confidence heuristics or attention-based signals, DEC operates from an **independent coordinate system**, making it robust to internal biases and calibration errors. This positions DEC as a lightweight, externalized form of metacognition — an executive layer that supervises reasoning rather than generating it.

## 2. Geometric Interpretability via DEC Visualization and Semantic Legends

Beyond producing a scalar drift score, DEC enables **rich geometric visualization of reasoning trajectories**, offering interpretability that goes significantly beyond existing attention-based or classifier-based methods.

By projecting both the LLM embedding path and the static embedding path into lower-dimensional spaces (e.g., via PCA or UMAP), DEC makes semantic behavior visually inspectable. Divergence between paths highlights where and how reasoning begins to destabilize.

To support interpretability, DEC visualizations can incorporate:

- **color gradients** representing drift severity
- **shape markers** indicating token roles (e.g., entities, verbs, numerics)
- **token-level annotations** identifying text segments responsible for divergence
- **directional arrows** encoding drift velocity and acceleration
- **region labels** separating stable reasoning from speculative or hallucinatory zones

Geometric patterns themselves carry semantic meaning. For example:

- smooth trajectories indicate stable reasoning
- sharp turns signal abrupt topic shifts
- oscillations suggest internal contradiction
- outward spirals correspond to runaway hallucination

Together, these legends and keys transform DEC outputs into **semantic maps of reasoning**, enabling researchers and engineers to diagnose failure modes, understand instability mechanisms, and identify optimization opportunities with visual clarity.

## 3. Positive Area-of-Effect on Existing and Large-Scale Systems

A key advantage of DEC is that it is **non-invasive**: it does not alter model weights, architecture, or training data. As a result, DEC provides immediate benefits when applied to already-deployed large-scale systems, including models with tens or hundreds of billions of parameters.

Even in large models, semantic drift still occurs — though often more subtly — manifesting as unnecessary verbosity, circular reasoning, overconfident speculation, or gradual goal dilution in long contexts. DEC is particularly effective at detecting these fine-grained instabilities.

When applied to existing systems, DEC yields several positive area-of-effect (AoE) benefits:

- **Identification of inefficient reasoning paths**, enabling removal of semantically redundant or unproductive computation
- **Improved effective accuracy**, by preventing derailment rather than adding new knowledge
- **Drift-aware inference control**, such as selective regeneration or early termination of unstable spans
- **Cost-aware optimization**, allocating additional compute only when semantic instability is detected
- **Safer agent behavior**, reducing cascading failures in long-horizon or tool-using systems
- **Diagnostic insight**, guiding future fine-tuning, pruning, or distillation efforts

Importantly, these benefits accrue **without retraining**, making DEC attractive as a deployment-layer reliability and efficiency enhancement. While DEC does not increase a model's raw knowledge capacity, it improves controllability, predictability, and semantic efficiency — properties that become increasingly valuable as model size and deployment scale grow.

*Taken together, these 3 extensions position DEC not merely as a hallucination detector, but as a general-purpose executive-control and interpretability layer. By providing semantic oversight, geometric diagnostics, and non-invasive optimization benefits, DEC complements model*

*scaling rather than competing with it, offering a path toward more reliable and compute-efficient AI systems.*

# Limitations

- Static embeddings struggle with rare words or new jargon
- Multilingual applications may require language-specific static vectors
- DEC is a detector, not a corrector (though it can guide correction)

# Future Extensions

- Multi-anchor DEC using multiple static embedding spaces
- Token importance weighting
- Integration with RAG for hybrid grounding
- DEC-enhanced prompt engineering
- Drift-aware decoding strategies

# Conclusion

Dual Embedding Cross-Check (DEC) introduces a new layer of reliability to LLMs by comparing their internal dynamic embedding trajectories against a stable semantic anchor derived from static embeddings. This simple, elegant approach detects semantic drift early, reduces hallucination risk, improves interpretability, and may allow smaller models to approach the reliability of significantly larger ones — all without retraining, heavy infrastructure, or model modification.

And if we get this right... Maybe one day the GPUs and RAM can finally go back to the gamers.

This paper was written with the assistance of LLMs and AI. I learn about AI by interacting with them. I am not formally trained. I am a self-taught individual that has been naturally shaped into a systems thinker as a result of my life experiences.

To connect with me please reach out to me on: raghavk.azp@gmail.com / outlook.com