

# Predicting and Suppressing Repetitive Degeneration via Hidden-State Risk Estimation

Logan Matthew Napolitano

---

## Abstract

Repetitive degeneration—the tendency of autoregressive language models to fall into loops or repeat phrases—remains a persistent failure mode in long-form generation. We demonstrate that this degeneration is **predictable from internal hidden states** before it occurs. We train a lightweight classifier ( $\sim 50\text{K}$  parameters) on transformer hidden representations to predict imminent token repetition, achieving  $F1 > 0.96$  and up to **125 $\times$  separation** between risk scores for tokens that will repeat versus those that will not. Using this signal, we introduce a decode-time intervention that applies adaptive penalties only when repetition risk is high, reducing repetition rate by **48.4%** while improving lexical diversity by **16.7%** (Distinct-2). We further document systematic failures of five attention-gating approaches on pretrained models, providing actionable negative results. Our findings suggest that repetition is not a stochastic artifact but a **predictable internal regime** exploitable without architectural modification. Code and weights are available at <https://github.com/Loganwins/HolonomyTransformer>.

---

## 1. Introduction

Autoregressive language models suffer from repetitive degeneration during long-form generation (Holtzman et al., 2020; Welleck et al., 2020). This manifests as:

- **Token-level loops:** Repeating the same word or phrase
- **Structural loops:** Cycling through similar sentence patterns
- **Semantic stagnation:** Failing to advance the discourse

Existing mitigations include:

Method	Mechanism	Limitation
Repetition penalty	Divide logits by constant	Uniform, not adaptive
Frequency penalty	Penalize by occurrence count	No predictive signal
Unlikelihood training	Train against repetition	Requires retraining base model
Contrastive decoding	Compare to smaller model	Additional model overhead

All these methods apply **uniform** or **reactive** interventions. None attempt to **predict** when repetition will become problematic.

## Contributions

1. **Empirical finding:** Hidden states strongly predict imminent repetition ( $F1 > 0.96$ ,  $125\times$  risk separation)
  2. **Method:** Decode-time adaptive penalty requiring no architectural changes
  3. **Negative results:** Systematic documentation of five failed attention-gating approaches
  4. **Artifacts:** Open-source code, trained weights, and reproduction instructions
- 

## 2. Related Work

### 2.1 Repetition in Neural Text Generation

Holtzman et al. (2020) identified repetition as a fundamental failure mode of likelihood-maximizing decoding. Welleck et al. (2020) introduced unlikelihood training to penalize repetition at training time. Fu et al. (2021) analyzed repetition through the lens of token frequency.

### 2.2 Decoding Strategies

Nucleus sampling (Holtzman et al., 2020), typical decoding (Meister et al., 2023), and contrastive decoding (Li et al., 2023) modify the sampling distribution to improve generation quality. Our work is complementary: we provide a **learned signal** that can inform any decoding strategy.

### 2.3 Probing Hidden Representations

Linear probes have revealed that hidden states encode syntactic structure (Hewitt & Manning, 2019), factual knowledge (Meng et al., 2022), and uncertainty (Kadavath et al., 2022). We extend this line by showing hidden states encode **repetition risk**.

---

## 3. Method

### 3.1 Problem Formulation

Given a sequence of tokens  $x_1, \dots, x_t$  and corresponding hidden states  $H = \{h_1^{(l)}, \dots, h_t^{(l)}\}_{l=1}^L$  across  $L$  layers, we aim to predict:

$$y_t = 1[x_t \in \{x_{t+1}, \dots, x_{t+W}\}]$$

where  $W$  is the repetition window (we use  $W = 32$ ). That is: will token  $x_t$  appear again in the next  $W$  tokens?

### 3.2 Risk Predictor Architecture

The predictor is deliberately minimal ( $\sim 50K$  parameters):

**RiskPredictor:**

```
fiber_proj[l]: Linear(d_model → d_fiber)    # Per-layer, l ∈ [1,L]
layer_weights: Parameter(L)                # Learned aggregation
predictor: MLP(d_fiber → d_control → 1)    # Risk logit
```

#### Forward pass:

1. Project each layer's hidden state:  $f_t^{(l)} = \text{fiber\_proj}_l(h_t^{(l)})$
2. Aggregate across layers:  $f_t = \sum_l \text{softmax}(\text{layer\_weights})_l \cdot f_t^{(l)}$
3. Predict risk:  $r_t = \sigma(\text{predictor}(f_t))$

#### Hyperparameters:

Parameter	Value
$d_{model}$	4096
$d_{fiber}$	16
$d_{control}$	64
$L$ (layers)	32
$W$ (window)	32
Total params	$\sim 50K$

### 3.3 Training

We train on WikiText-2 (Merity et al., 2017) with binary cross-entropy loss:

$$\mathcal{L} = -\frac{1}{T} \sum_{t=1}^T [w_+ \cdot y_t \log(r_t) + (1 - y_t) \log(1 - r_t)]$$

where  $w_+ = \min(n_{neg}/n_{pos}, 10)$  corrects for class imbalance.

#### Training configuration:

Parameter	Value
Optimizer	AdamW
LR (predictor)	1e-4
LR (LoRA)	2e-5
Batch size	4
Gradient accumulation	8
Steps	5000

The base model is frozen except for LoRA adapters (Hu et al., 2022) on attention projections.

### 3.4 Decode-Time Intervention

At generation time, for each step:

1. Compute logits  $z_t$  and hidden states  $H_t$
2. Predict risk  $r_t$  from  $H_t$
3. If  $r_t > \tau$  (threshold):
  - Identify recent tokens:  $R = \{x_{t-W}, \dots, x_{t-1}\}$
  - Apply penalty:  $z_t[v] \leftarrow z_t[v] - r_t \cdot \lambda$  for  $v \in R$
4. Sample from adjusted distribution

#### Inference hyperparameters:

Parameter	Default
$\tau$ (threshold)	0.1
$\lambda$ (penalty scale)	3.0
Temperature	0.8

## 4. Negative Results: Attention Gating Failures

Before arriving at decode-time intervention, we attempted five approaches to modify attention patterns directly. All failed. We document these as actionable negative results.

## 4.1 Multiplicative Gating

**Approach:**  $\text{attn\_out} = \text{attn\_out} \times g$  where  $g \in [0.1, 0.9]$

**Result:** Immediate incoherence. Pretrained attention patterns expect specific magnitude distributions; multiplicative scaling destroys this.

## 4.2 Log-Space Score Modification

**Approach:**  $\text{scores} = \text{scores} + \log(g + \epsilon)$

**Result:** Gates converged to 0.499 uniformly. Mathematical inevitability:  $\text{softmax}(x + c) = \text{softmax}(x)$ . Adding constants to all scores has no effect post-softmax.

## 4.3 Normalized Gating

**Approach:**  $g_{\text{norm}} = (g - \mu_g) / \sigma_g$

**Result:** NaN during generation. Single-token sequences have undefined standard deviation, causing division by zero at inference.

## 4.4 Causal EMA Statistics

**Approach:** Maintain running statistics with exponential moving average for consistent normalization.

**Result:** Training metrics appeared healthy (low loss, meaningful gates). Generation degraded progressively—coherent at step 100, gibberish by step 4000. The model learned training-specific compensations that failed under autoregressive generation.

## 4.5 Extended Training

**Approach:** Continue training to 5000+ steps hoping for adaptation.

**Result:** Complete collapse to invalid byte sequences. Extended training exacerbated overfitting.

## 4.6 Analysis

The fundamental issue: **pretrained attention patterns are finely tuned over billions of tokens**. Any modification creates distribution shift that propagates through subsequent layers. The model's knowledge is encoded in inter-component relationships, not just weights.

**Conclusion:** Decode-time intervention at the sampling stage avoids this problem entirely by leaving internal computations unchanged.

---

# 5. Experiments

## 5.1 Risk Prediction Performance

We evaluate the predictor's discrimination ability:

Step	F1	P(risk   repeat)	P(risk   ¬repeat)	Separation															
-----	-----	-----	-----	-----		3000	0.96	0.946	0.076	12.4×		4000	0.99						
0.997	0.014	71×				<b>5000</b>	<b>0.99+</b>	<b>0.998</b>	<b>0.008</b>	<b>125×</b>		6000	0.99+		0.999				
0.021	48×																		

Peak separation occurs at step 5000. Further training reduces discrimination, consistent with overfitting to training-specific patterns.

**Key finding:** The predictor achieves near-perfect separation. Tokens that will repeat receive risk scores averaging 0.998; tokens that will not average 0.008.

5.2 Generation Quality

We evaluate on 5 open-ended prompts with 500 tokens each:

Metric	Baseline	+ Intervention	Δ
Repetition Rate	33.9%	17.5%	<b>-48.4%</b>
Distinct-2	0.836	0.976	<b>+16.7%</b>

**Repetition Rate:** Fraction of tokens appearing in previous 32-token window.

**Distinct-2:** Ratio of unique bigrams to total bigrams (Li et al., 2016).

5.3 Qualitative Analysis

**Baseline failure mode (procedural generation):**

┆ "Step 1: ... Step 2: ... Step 2a: ... Step 2b: ... Step 2c: ..."

The model becomes trapped in nested sub-steps, never advancing.

**With intervention:**

┆ "Step 1: Establish framework. Step 2: Implement core logic. Step 3: Validate outputs..."

The model maintains forward progress through the procedure.

5.4 Ablations

Configuration	Rep. Rate	Distinct-2
Baseline (no intervention)	33.9%	0.836
Fixed penalty ( $\lambda=1.2$ , no prediction)	28.1%	0.891
Adaptive penalty (ours)	<b>17.5%</b>	<b>0.976</b>

The learned, adaptive penalty outperforms uniform penalty, confirming the value of the predictive signal.

## 6. Limitations

1. **Single model:** Evaluated only on LLaMA-3.1-8B. Generalization to other architectures is untested.
2. **Single dataset:** Trained on WikiText-2. Domain transfer is unverified.
3. **No tuned baselines:** We did not extensively tune standard repetition penalties for comparison. The 48.4% improvement is against unpenalized generation.
4. **Computational overhead:** Risk prediction adds  $\sim 10\%$  to generation time.
5. **Narrow definition:** We predict token-level repetition, not semantic redundancy or higher-order loops.

## 7. Discussion

### 7.1 Repetition as Predictable Regime

The  $125\times$  separation suggests repetition is not stochastic noise but a **predictable internal state**. The model "knows" it is about to repeat before emitting the token. This signal is:

- Extractable with minimal parameters ( $\sim 50K$ )
- Present across layers (learned aggregation helps)
- Anticipatory, not retrospective

### 7.2 Why Decode-Time Works

Modifying attention failed because pretrained representations are brittle to distribution shift. Decode-time intervention succeeds because:

- Base model computations remain unchanged
- Only the sampling distribution is modified
- The intervention is minimal and targeted

### 7.3 Implications

If repetition is predictable, other degeneration modes may be as well:

- Hallucination
- Contradiction
- Topic drift
- Incoherence

A suite of lightweight predictors could enable **multi-signal decoding control** without architectural changes.

---

## 8. Conclusion

We demonstrated that repetitive degeneration in language models is strongly predictable from hidden states ( $F1 > 0.96$ ,  $125\times$  separation). A decode-time intervention using this signal reduces repetition by 48.4% while improving diversity. We documented five failed attention-gating approaches, establishing that sampling-stage control is more robust than architectural modification for pretrained models.

Our results suggest repetition is a detectable internal regime, not an inevitable artifact—opening possibilities for learned, adaptive generation control.

---

## References

- Fu, Z., Lam, W., So, A. M. C., & Shi, B. (2021). A theoretical analysis of the repetition problem in text generation. AAAI.
- Hewitt, J., & Manning, C. D. (2019). A structural probe for finding syntax in word representations. NAACL.
- Holtzman, A., Buys, J., Du, L., Forbes, M., & Choi, Y. (2020). The curious case of neural text degeneration. ICLR.
- Hu, E. J., et al. (2022). LoRA: Low-rank adaptation of large language models. ICLR.
- Kadavath, S., et al. (2022). Language models (mostly) know what they know. arXiv.
- Li, J., Galley, M., Brockett, C., Gao, J., & Dolan, B. (2016). A diversity-promoting objective function for neural conversation models. NAACL.



Li, X. L., et al. (2023). Contrastive decoding: Open-ended text generation as optimization. ACL.

Meister, C., et al. (2023). Locally typical sampling. TACL.

Meng, K., et al. (2022). Locating and editing factual associations in GPT. NeurIPS.

Merity, S., Xiong, C., Bradbury, J., & Socher, R. (2017). Pointer sentinel mixture models. ICLR.

Welleck, S., et al. (2020). Neural text generation with unlikelihood training. ICLR.

---

## Appendix A: Reproduction

**Code:** <https://github.com/Loganwins/HolonomyTransformer>

**Weights:** <https://huggingface.co/LoganResearch/Adaptive-Repetition-Controller>

### Training:

```
bash

cd HolonomyTransformer
python training/cfhot_risk_v2.py
```

### Inference:

```
python

from huggingface_hub import hf_hub_download
import torch

# Load predictor
predictor = torch.load(
    hf_hub_download("LoganResearch/Adaptive-Repetition-Controller", "risk_predictor.pt")
)
```

---

## Appendix B: Per-Prompt Results

Prompt	Base Rep.	+Ours	Base D2	+Ours
Philosophy	5.3%	27.0%	1.000	0.964
Creation myth	54.1%	0.0%	0.622	1.000
Reality	30.7%	23.6%	0.933	0.943
Happiness	51.4%	24.3%	0.682	0.972
Mind-body	28.1%	12.5%	0.944	1.000
<b>Average</b>	<b>33.9%</b>	<b>17.5%</b>	<b>0.836</b>	<b>0.976</b>

### Appendix C: Hyperparameter Sensitivity

$\lambda$ (penalty)	Rep. Rate	Distinct-2	Coherence
1.0	25.2%	0.912	High
3.0	17.5%	0.976	High
5.0	14.1%	0.984	Medium
10.0	8.3%	0.991	Low

Higher penalties reduce repetition but can harm coherence.  $\lambda \in [3, 5]$  balances both.