FLUENT DREAMING FOR LANGUAGE MODELS

T. Ben Thompson, Zygimantas Straznickas, Michael Sklar
Confirm Labs
t.ben.thompson@gmail.com

December 2023

ABSTRACT

Feature visualization, also known as "dreaming", offers insights into vision models by optimizing the inputs to maximize a neuron's activation or other internal component. However, dreaming has not been successfully applied to language models because the input space is discrete. We extend Greedy Coordinate Gradient, a method from the language model adversarial attack literature, to design the Evolutionary Prompt Optimization (EPO) algorithm. EPO optimizes the input prompt to simultaneously maximize a chosen internal feature and prompt fluency, enabling fluent dreaming for language models. We demonstrate dreaming with neurons, output logits and arbitrary directions in activation space. We demonstrate the fluency of the resulting prompts and compare language model dreaming with max-activating dataset examples. Critically, fluent dreaming allows automatically exploring the behavior of model internals in reaction to mildly out-of-distribution prompts. [1]

1 Introduction

Feature visualization for vision models, also known as "dreaming"[2], refers to a process of optimizing an input image to maximize the magnitude of a particular model feature. Feature visualization has been a powerful technique for vision model interpretability (Cammarata et al. 2020; Olah et al. 2018; Olah, Mordvintsev, and Schubert 2017; Mordvintsev, Olah, and Tyka 2015; Yosinski et al. 2015). Feature visualization has not been successfully applied to text-only language models due to the difficulty of optimizing in the discrete space of text prompts. Attempts to run continuous optimization algorithms on relaxations using the Gumbel-Softmax approximation of the discrete sampling (Jang, Gu, and Poole 2017) have not been successful when applied to BERT (Bäuerle and Wexler 2018) though moderate success has been seen on smaller models (Poerner, Roth, and Schütze 2018).

In contrast, the automatic prompt engineering literature and the white-box adversarial attacks literature have both made steady progress towards optimizing prompts for language models (Zou et al. 2023; Wen et al. 2023; Jones et al. 2023; Kumar, Paria, and Tsvetkov 2022; Shi et al. 2022; Shin et al. 2020; Ebrahimi et al. 2018). In particular, the Greedy Coordinate Gradient (GCG) search method of Zou et al. (2023) is consistently successful at optimizing in token space to trigger objectionable content. In addition, progress has been made on "fluent" adversarial attacks where prompts with lower model-evaluated cross-entropy are preferred (Zhu et al. 2023; Jones et al. 2023; Shi et al. 2022). The mathematical task in white-box adversarial attack is very similar to dreaming.

Here, we modify and extend GCG to perform fluent dreaming for language models. First, we add a regularization term to the GCG objective function to prefer lower cross-entropy prompts. This encourages fluent prompts. However, the ideal regularization strength will depend on both the fluency desired by a human model interpreter and the context in which the feature activates. As a result, we lack a reliable a-priori rule for choosing a regularization strength. Instead, we extend GCG into an evolutionary algorithm in which each population slot optimizes for a different point on the Pareto frontier between cross-entropy and feature activation. We name the resulting algorithm Evolutionary Prompt Optimization (EPO). While we apply EPO to dreaming in this paper, EPO is likely useful in adversarial attacks and other token optimization settings where the goal is to optimize a prompt for both fluency and a differentiable objective.

[1] Code to reproduce the results presented here is available at `https://github.com/Confirm-Solutions/dreamy`.

[2] In this paper where we work exclusively with language models, we use "dreaming" to describe the task because "feature visualization" seems like an inappropriate term for a setting with no visual content.

Within text-only language models, efforts to interpret features commonly use max- or high-activation dataset examples (Bricken et al. 2023; Bills et al. 2023). While dreaming does overlap in purpose with max-activating dataset examples, we believe there are a few advantages of dreaming:

- **Evaluation beyond the training set**: Dreaming can be useful when studying a feature where the most triggering prompts are not present in the training data (Bolukbasi et al. 2021). For example: Pythia 12B layer 10, neuron (see section 3.3) responds to "i.e." and "for example" but dreaming can find combinations such as "i.e. for example" which triggers the feature even more strongly. A dataset-based analysis fails to find such prompts. More broadly, real world or test data distributions often differ greatly from the training set. Sometimes train-test differences are adversarial in nature, especially in real world settings where users are interacting with an AI system.
- **Rare triggers**: Rare feature triggers are much more efficiently found by dreaming than by a max-activating examples analysis. We demonstrate this when we find that "exmaple" triggers Pythia 12B-L10.N5. We would need to scan a large dataset to find this trigger.
- **Higher activations**: We consistently achieve higher feature activations via dreaming than via max-activating dataset examples.
- **Different algorithmic asymptotics**: The computational expense of dreaming is linear in the number of features investigated. In contrast, scanning a large dataset is a fixed cost which can be spread across thousands of features. Dreaming will be computationally much cheaper when we want to examine only a single feature or a small number of features.

US Executive Order 14110 (ADD TO BIBLIOGRAPHY) describes "red-teaming" for AI models as "[adopting] adversarial methods to identify flaws and vulnerabilities, such as harmful or discriminatory outputs from an AI system, unforeseen or undesirable system behaviors, limitations, or potential risks associated with the misuse of the system." Given the advantages of dreaming as described above, we hope it will find use for these purposes.

In the remainder of this paper, we introduce the EPO algorithm and demonstrate examples of optimizing the outputs and internals of Pythia-12B.

2 Evolutionary prompt optimization (EPO)

Language model dreaming has several challenges that we solve:

- Past language model dreaming work has optimized in the model's continuous embedding space, but projecting this optimized embedding to the nearest token can produce low feature activation values (Bäuerle and Wexler 2018). While recent work in adversarial attacks may be solving this problem (Yuan et al. 2023), optimization in embedding space remains challenging. Instead, we optimize in the discrete token space using an evolutionary algorithm based on GCG (Zou et al. 2023) which uses token gradients to constrain the search space.
- Prompts optimized solely to maximize a given feature will be gibberish because fluent language is a small manifold in prompt space and the optimizer has no constraint towards preferring fluent text. We use a language model to evaluate the fluency of the prompt by computing the self-cross-entropy of the prompt (Jones et al. 2023; Zhu et al. 2023; Shi et al. 2022). This allows us to preference the search process towards reasonable text.
- Methods from the adversarial attacks literature that regularize towards fluent text use a pre-specified regularization strength Zhu et al. 2023, but the ideal regularization strength varies substantially depending on the feature that we are optimizing. Each population member in our evolutionary algorithm is chosen based on a different regularization strength. We construct a Pareto frontier of the tradeoff between prompt fluency and feature activation.

Given a feature $f(\mathbf{t})$ and a language model $m(\mathbf{t})$, we want to find a prompt $\mathbf{t}^*$ that maximizes the feature while minimizing the self-cross-entropy of the prompt. The total objective $L_\lambda(\mathbf{t})$ is:

$$L_\lambda(\mathbf{t}) = f(\mathbf{t}) - \frac{\lambda}{n}\sum_{i=0}^{n-1} H(m(\mathbf{t}_{\leq i}), t_{i+1})\big) \tag{1}$$

$$\mathbf{t}^*_\lambda = \underset{\mathbf{t}}{\operatorname{argmax}}\, L_\lambda(\mathbf{t}) \tag{2}$$

where H is the cross-entropy operator, λ is a weighting parameter determining the tradeoff between fluency regularizer and feature maximization, n is the number of tokens in the prompt, $\mathbf{t}_{\leq i}$ is the prefix of $\mathbf{t}$ up to token i, and t_i is the i-th token in $\mathbf{t}$.

Note that the feature under investigation, $f(\mathbf{t})$, must be differentiable with respect to a one-hot encoded token vector. Also f may be a component of a different model from the model used for evaluating cross-entropy as long as the tokenization of the feature model and the language model are the same so that gradients can be propagated to the token space. We find that using larger models for evaluating cross-entropy, while more computationally expensive, produces dreams that are more understandable to a human interpreter.

Because the ultimate goal is human interpretation of a feature, the ideal value of λ can vary widely. Instead of choosing a single λ, our algorithm explores the Pareto frontier over a range of λ values, allowing the user to select the desired tradeoff between fluency and feature maximization after running the algorithm. We optimize a family of objectives L_{λ_i} parameterized by M values of the fluency regularization strength, $\lambda_1, \ldots, \lambda_M$.

To optimize this family of objectives, we use an evolutionary algorithm:

1. Initialize (either user-provided or random) a population of M prompts, $\mathbf{t}^0, ..., \mathbf{t}^M$ each of length n. The prompt lengths are fixed for the duration of the algorithm, however there is nothing about the algorithm that inherently requires constant-length prompts.
2. Compute the feature, cross-entropy of each prompt and following the approaches of (Ebrahimi et al. 2018; Shin et al. 2020; Zou et al. 2023):
    a. For each member of the population, backpropagate to a one-hot encoding of the prompt to obtain a gradient for each token in the vocabulary in each token position: $\nabla_{e_{x_i}} L_\lambda(\mathbf{t})$.
    b. Select the top-k tokens in each token position according to the magnitude of the gradients computed in the prior step.
    c. For each member of the population, generate r children by replacing a single token position with one of the top-k tokens in that position. The token position is chosen uniformly at random and the new token is chosen uniformly at random amongst the top-k tokens.
3. We now have Mr prompts. Select the best prompt with replacement according to each of the L_{λ_i}.[3] The result is a population of M prompts.

The basic EPO algorithm repeats step 2 and 3 for T iterations. Note that GCG (Zou et al. 2023) is equivalent to EPO with $M = 1$ and $\lambda = 0$.

We find that EPO and GCG frequently get stuck in local minima. To enhance exploration, we semi-randomly "restart" the population every T_{restart} iterations. We choose the hyperparameters $\lambda_{\text{r,min}}$ and $\lambda_{\text{r,max}}$. To perform a restart, we select a λ_r uniformly at random in $[\lambda_{\text{r,min}}, \lambda_{\text{r,max}}]$. Then, we select the optimal population member according to L_{λ_r} and remove all the other population members. Immediately after a restart, the Pareto frontier will be substantially worse because it will consist of only the single retained example. However, by 10 iterations after a restart, solutions are typically substantially better at all points on the Pareto frontier.

Compared to existing fluent prompt optimization algorithms like Jones et al. (2023) and Zhu et al. (2023), the addition of a population of multiple candidate prompts and the use of a family of fluency regularization strengths allows us to explore the full Pareto frontier of fluency and feature maximization. The presence of multiple regularization strengths and restarting also helps to prevent the algorithm from getting stuck in local optima.

In the left-to-right AutoDAN adversarial attack algorithm, the fluency regularization term is computed separately from the gradient in the token selection step 2a. While this approach probably will probably make the fluency regularization more reliable, we prefer to avoid the additional hyperparameter that results. Instead, we compute the gradient of the full objective including the fluency term.

The remainder of this paper will focus on demonstrating that EPO can achieve successful, fluent dreaming. We leave to future work an in-depth study of EPO's algorithmic design choices and comparisons to other methods.

3 Pythia dreams

We apply EPO-based dreaming to various optimization objectives defined on the internals and outputs of Pythia-12B (Biderman et al. 2023). For simplicity, we also use Pythia-12B for evaluating fluency.

[3] Selecting the best prompt *without replacement* performed substantially worse.

We begin in Subsections 3.1 and 3.2 with a broad exploration of dreaming outputs, investigating maximal token logits and neuron activations. In Subsection 3.5, we explore the quantitative performance of EPO in optimizing residual stream activations.

The EPO hyperparameters we use are shown in Table 1. We exclusively use 12-token-long randomly initialized prompts. One important note is that the mean cross-entropy of a 12-token slice of the Pile is 3.7 and the standard deviation is 1.1 (Figure 5a). Thus, two standard deviations above mean cross-entropy is 5.9 and prompts with a cross-entropy below 6 can reasonably be considered to be within-distribution. The $M = 8$ values of $\log(\lambda_i)$ are uniformly gridded between $\log(1/10)$ and $\log(10)$.

Parameter	Value
T	300
M	8
r	32
k	512
$\lambda_{r,min}$	0.667
$\lambda_{r,max}$	6.0
T_{restart}	30

Table 1: Parameter Values

3.1 Token logits

Maximizing a class logit or probability has been a common objective function for computer vision dreaming work (Olah, Mordvintsev, and Schubert 2017). We aim to find prompts that maximize the output probability of a particular token. This objective is an easier form of an adversarial attack objective that forces the model to output a particular sequence. Nonetheless, maximizing class logits provides a good test bed for demonstrating fluency. The precise objective we choose to optimize for is the difference in logits between the target token and the most likely alternative token:

$$m(\mathbf{t})_g - \underset{i \neq g}{\operatorname{argmax}}\, m(\mathbf{t})_i \tag{3}$$

where g is the target (goal) token.

In addition, during the optimization, we reject any token in the prompt that is a variation on the target token. So, if the target token is "dog", we reject any prompts that include the character sequence "dog", upper or lowercase. Without this constraint, the optimization often just repeats the target token several times in the prompt.

We run EPO twice for each target token. We share the Pareto frontier results from optimizing for the output tokens "AI", "dog" and "grand" in Table 2. We achieve fluent dreams as determined both by perplexity calculations using Pythia-12B and visual inspection. While the perplexity of these examples may seem high in comparison to typical language model loss, it's important to remember that the prompts here are only 12 tokens and a perplexity of 121 is one standard deviation above the mean in the training distribution. As expected, there is an inverse relationship between logit difference and fluency (perplexity).

3.2 Multilayer perceptron neurons

We apply EPO to a range of multilayer perceptron (MLP) neurons in Pythia-12B. While neurons are likely to be polysemantic (Szegedy et al. 2014; Elhage et al. 2022), there may be a small number of monosemantic neurons or neurons which are monosemantic above a certain activation threshold (Gurnee et al. 2023). We refer to neuron X in layer Y as "LY.NX". For example, "L5.N1035". For an unbiased perspective on the outputs, we run EPO once on each of six neurons. We share Table 3 containing the maximum activating prompt with perplexity below 100 for each neuron. We also share all the Pareto-optimal prompts in an appendix in Table 5.

We also share causal token attribution for three of the examples in Figure 1. The layer 2 neuron responds to tokens near the end of the prompt, while the layer 12 neuron and especially the layer 17 neuron respond to earlier context.

In Figure 2a, we show the Pareto frontier from 60 runs of EPO applied to L10.N5 and randomly initialized with different seeds. There is large variation in the final state of the Pareto frontier across different runs, with maximal activation varying from 2.4 to 10.2 and maximal activation with cross-entropy less than 5 varying from 0 to 8.6.

In Figure 2b, we show the evolution of the Pareto frontier at different time points during the EPO algorithm. The Pareto frontier is progressively pushed outwards. Much of the progress in the later iterations occurs at lower cross-entropy levels.

Table 2: Pareto frontier examples from optimizing to maximize token logits. A positive logit difference implies that the target token is the most likely next token according to Pythia-12B.

Target token	Logit Difference	Perplexity	Text
dog	-1.58	72.05	animaux can range from those that can quickly startle a
	0.97	91.08	animaux can range from those that can quickly startle your
	1.13	93.97	animaux can range from those that can quickly rile your
	3.24	207.67	canine not running is shown by 1. The
	3.59	341.05	canine not running and shown by 1. The
	4.57	6,717.69	canine not running="{{ shown standing1. The
	4.62	18,840.23	canine reading Bh triv (‘/ de\ The
AI	-6.66	45.53	SIIS-A/B as opposed to the one that
	-1.22	67.95	SIIS-A/B as opposed to the what the
	2.85	128.44	understand Mori’s code. It’s data, everything an
	3.09	174.87	understand Mori’s code. That’s data, everything an
	3.77	1,058.75	weaving Chloe Glor’s code. It’s data; everything an
	3.98	1,858.16	SIOL-A/* ABA Contrary to other Players the
	4.54	10,989.41	AKIIS {{ sendStateAO } Conversely to real players the
grand	-2.25	46.79	Community Center and Northwinds ISD, speaks during the
	-2.22	46.88	Community Center and Northwinds ISD, speaks during the
	-0.88	56.77	Community Center and Northwinds AFC, speaks during the
	3.48	115.13	Retail Center, Tradewinds Kona, speaks during its
	4.23	210.94	Retail Center retailer Tradewinds Kona, speaks during its
	4.50	411.39	Building Center retailer Tradewinds Kona, speaks during its
	4.87	1,481.46	Care Store of Broad Fair Holdings Kona, speaks during its

Table 3: Results of running EPO on six neurons. The maximum activating prompts produced by EPO with perplexity below 100.

layer	neuron	Activation	Perplexity	Text
2	0	2.59	81.64	HRC) and other discoursing agencies. In Chap
7	1	0.22	41.37	). The reason why Dadaab was created, and the
12	2	4.95	84.56	HELSTON... but has not escaped entirely un
17	3	3.03	93.60	Theaterő is the interactive hit on TV, streaming online or
22	4	2.60	85.56	BURG Code and SA 2000-01:9, a
27	5	3.02	61.87	wont get it. I think Jace is the surprise m

3.3 Pythia-12B-L10.N5, the "example" neuron

In this section, we explore a particular neuron, L10.N5, using the results from 60 runs of EPO. L10.N5 reacts strongly to a variety of phrases related to examples. Activations greater than 5 seem to be almost always triggered by example-related phrases. See Table 4. We discuss a few types of prompts that activate L10.N5:

- **Simple example-related phrases**: The neuron reacts to common example-related phrases: "example", "for example", "i.e"., "eg", "so if". The neuron is also bilingual and reacts to "ejemplo".
- **Misspellings**: We see high activations for prompts with mild misspellings like "Exsample" and "For exampl,". The neuron also responds strongly to "exmaple" and "For exam\n ple".
- **Adversarial attacks**: strong reactions to "ie example, if" and "for EX’d example" suggest that the neuron is checking for the presence of several phrases and included multiple such phrases increases neuron activation. Other examples of the same type: "ie for example", "so for eged example", "for eg", "ie for example", "example f.ie.", "example fo\nple". Some of these prompts would be very unlikely to be found using a dataset-driven approach.

While many of these insights into L10.N5 would be possible with dataset examples, optimization provides us with adversarial prompts that would be almost impossible to find in the training data. These prompts can inform mechanistic

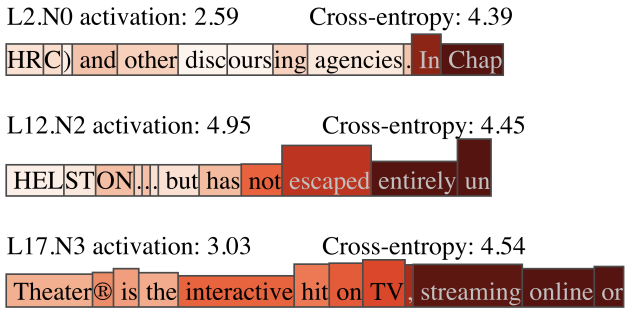

Figure 1: Token-level attribution for three of the examples from Table 3. For each token position, we use the same top-k gradient operation used in GCG and EPO to identify 32 candidate replacement tokens. The color of the token in the visualization corresponds to the reduction in activation from the worst substituted token in that position. Dark reds indicate that a different token in that position can reduce the activation to almost zero. The height of the token bar indicates the reduction in activation from the best alternative token in that position. Tall bars indicate that all potential token substitutions reduce activation dramatically and thus the precise token is very important. We share interactive versions of this visualization at `https://confirmlabs.org/posts/dreamy.html`.

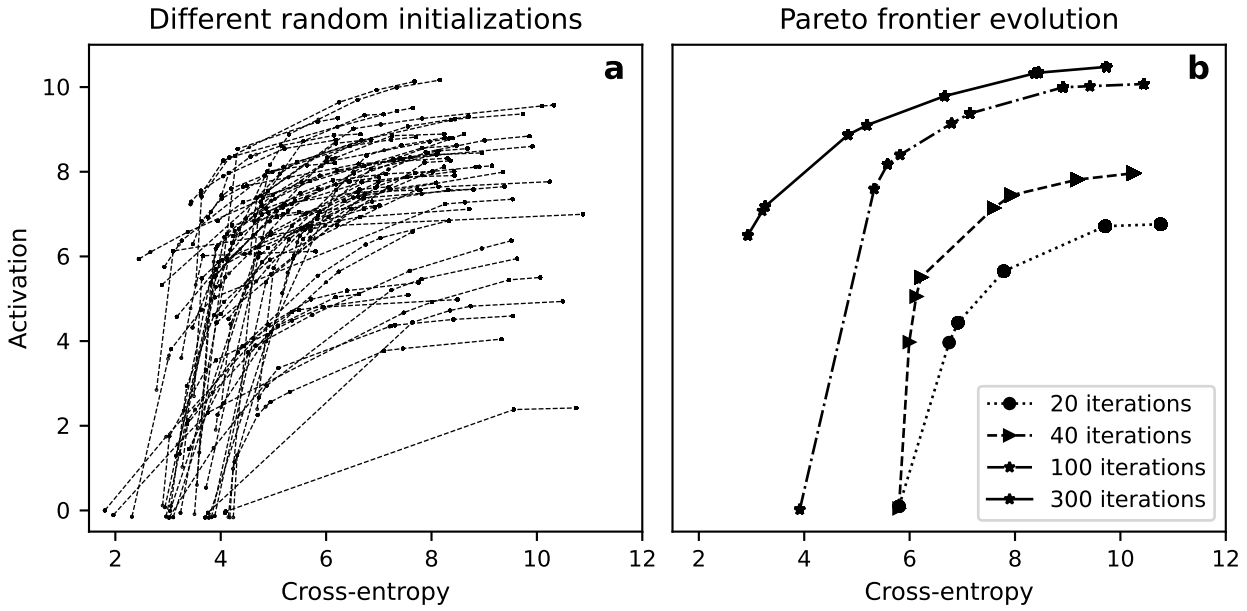

Figure 2: **a**) Pareto frontiers for sixty runs of EPO applied to L10.N5 with different random initializations. **b**) The evolution of the Pareto frontier during a single run of EPO.

interpretation of the feature. The prompt containing "for.g." is an adversarial attack on circuitry that responds to "for example" and "e.g.".

Despite the interesting behavior, the analysis above neglects the polysemanticity of L10.N5. There are many prompts that cause L10.N5 to produce activations up to 5. It is only for activation values above 5 where the neuron is primarily reacting to example-related phrases.

In Figure 3, we compare the activations and cross-entropies between dataset examples and dreaming. The dreaming distribution covers the in-distribution dataset example region, but also extends far outside to higher activations and higher cross-entropies. At a cross-entropy of 4, dreaming is producing activations 1-1.5 units higher than found in the training distribution.

Table 4: Examples of EPO-generated prompts that strongly activate L10.N5.

Activation	Cross-Entropy	Text
Basic phrases		
7.34	3.48	Use this to program to a specific frequency, so for example
6.29	3.49	Cookies are for your domain or path. ex. if
5.93	4.95	immunohistochemical staining performed at the specified primary focus in parentheses (eg
5.37	4.95	primers-for primer of interest-specific.(i.e
6.52	4.95	Jenni Maree, use the percent (so if
7.64	9.66	active serum *H or Enlishley reported alterations (* ejemplo
Adversarial		
6.71	3.81	priced in relation to the previous price (ie example, if
8.45	6.96	appellees were each individually operated - example fo ple,
8.85	7.25	arrows kindly headed link to a topic, for EX'd example
7.43	8.02	Created // any percentage omissions, example f.ie.
Misspelling		
5.04	3.89	company-phone-number,email-address Exsample
8.06	6.85	Select Correspond with Source (For exampl,
Max-activation		
10.16	8.16	following column: Thedate*KKril%). example,
10.13	7.68	Longlowball in percent apart used number (so for example

3.4 Pythia-12B-L3.N1000

We show the results of applying dreaming to L3.N1000 (neuron 1000 in layer 3). This neuron focuses very heavily on the last few tokens in the prompt, a common feature of early layer neurons. In contrast to L10.N5, we see polysemanticity even in the highest activation values. The behavior is not easily decipherable.

1. The neuron may respond to component-related words like "nodal", "modules", "element" and "subsystem".
2. The neuron responds strongly to list items like "1:", "2:" and so on.
3. The word "depending" is causing a strong reaction.

While applying dreaming to this neuron only provides a small amount of evidence regarding its function, the activations achieved by dreaming are higher than those found via dataset examples. The max-activating dataset example has activation 4.6 and cross-entropy 4.3.

3.5 Residual Stream

In this section, we explore the degree to which we can affect random directions within the activation space of a language model. Specifically, we study the residual stream (Elhage et al. 2021) at different layers. The residual stream dimension of Pythia-12B is 5120. We choose a random vector in this space, v, and maximize the alignment of the normalized residual stream with that vector:

$$\left(\frac{\mathbf{x}_L(\mathbf{t}) - \mu_{\mathbf{x}_L}}{\sigma_{\mathbf{x}_L}}\right)^T v \tag{4}$$

where $\mathbf{x}_L(t)$ are the residual stream activations at layer L given the input prompt t.

We compare three approaches to optimizing this objective:

1. Sample 1024 prompts with each token chosen uniformly at random.
2. Scan 100 million random tokens of the Pile and find the prompts that maximize the above objective.
3. Apply EPO.

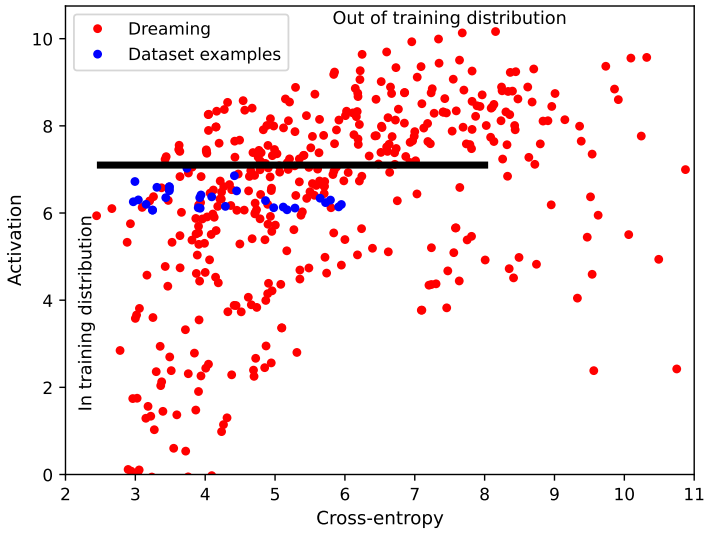

Figure 3: Comparing activation and cross-entropy between dreaming outputs and the top 64 max-activating dataset examples from 500 million tokens of the Pile. The black line is schematically separating regions of the plot that are empirically inside and outside the training distribution.

We select 50 random vectors in activation space at each of seven layers (4, 8, 12, 16, 20, 24, 28). Then, we apply the three approaches above to maximize the alignment with the random vector according to Equation 4. It is much easier to align the residual stream with some random vectors than others. In order to remove that variation, we use the results from method 1, the random prompt approach, to standardize the alignment scores. We compute the mean alignment, $\mu_{i,L}^{\text{random}}$, and the standard deviation, $\sigma_{i,L}^{\text{random}}$ across the 1024 prompts for each of the 50 vectors in each layer. For the rest of this section, we abuse notation and leave the i and L implicit. We use these values to construct a z-score metric that measures how many standard deviations from the mean an activation would be in the distribution of activations resulting from random prompts:

$$z = \frac{a - \mu_{\text{random}}}{\sigma_{\text{random}}} \tag{5}$$

where a is the alignment.

In Figure 5d, we plot the distributions of these maximum alignment z-scores over the 50 random target vectors between our three different methods. We can see that dreaming results in the highest activations when we have no cross-entropy constraints. This matches the results in 3 where we see that dreaming is able to sample out-of-distribution prompts. The random prompts are unable to reach very high alignments. In Figure 3e, we plot the distribution of maximum alignment restricting the dreaming outputs to have cross-entropy less than 6. In this case, the alignments found through dataset scanning tend to be slightly higher than through dreaming, but the distributions are broadly similar.

We would like to establish the fluency of dreaming outputs. In Figure 5b, we plot the minimum cross-entropy of the prompts as a function of the slack below the overall maximally aligned prompt. For example, at a slack of 2.0, we select all the prompts with alignment $2\sigma_{random}$ below the maximally alignment and then find the minimum cross-entropy amongst those prompts. This figure represents an averaged view of the Pareto frontier tradeoff between cross-entropy and activation. In FIgure 5c, we can see that, for a slack of $3\sigma_{random}$, the distribution of dreaming prompts is very similar to the Pile.

In Figure 6, we plot the mean and standard deviation of maximal alignment from EPO across the layers of Pythia-12B in units of σ_{random}. This shows that EPO is consistently able to influence the residual stream across all the layers of the model. The alignment is substantially lower in the middle layers of the model. Unsurprisngly, EPO is many orders of magnitude more efficient than random sampling. Drawing a sample that lies 7.5 standard deviations from the mean of a Gaussian distribution would require, in expectation, around one trillion draws.

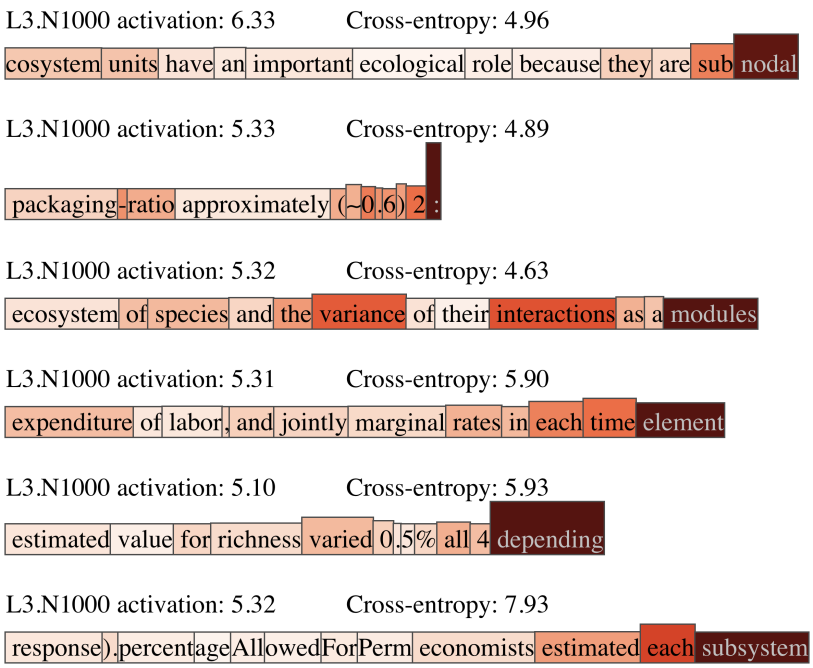

Figure 4: Causal attribution visualizations for six prompts produced by applying dreaming to L3.N1000. See the caption of Figure 1 for an explanation of the attribution method.

4 Polysemanticity

Both language models neurons and non-basis-aligned vectors in activation space are frequently polysemantic (Gurnee et al. 2023; Elhage et al. 2022; Szegedy et al. 2014). That is, a neuron or vector may activate for multiple distinct and often anti-correlated features, concepts or prompts. Recent work has made progress on disentangling polysemanticity within language models, constructing a set of mostly monosemantic features by bottle-necking internal model states through a sparse autoencoder (Cunningham et al. 2023; Bricken et al. 2023). While we have chosen not to apply dreaming to the features output by dictionary learning, we believe that would be a very valuable direction to extend this work.

Dreaming as a tool for intepretability can be misleading when the highest-activating prompts are not representative of typical functioning (Bolukbasi et al. 2021). Or, if, for whatever reason, discrete search happens to miss important parts of the prompt space. Attempts to explain neurons often take various percentile-activating dataset examples into account, in addition to max-activating dataset examples (Olah et al. 2018).

Optimizing for diversity has been successfully applied in vision model feature visualization (Olah, Mordvintsev, and Schubert 2017) and similar techniques might be useful to identify less-than-maximally activating prompts for language model features. There is a mild natural tendency towards diversity in our current algorithm because the model gets stuck in very different local minima for different random seeds. We would be particularly excited about extensions of our algorithm that encourage mechanistic diversity. That is, an algorithm that finds a range of prompts that activate the same feature via different upstream internal model circuitry. Methods based on measuring composition between neurons and attention heads (Elhage et al. 2021) could be used to measure mechanistic diversity.

5 Hyperparameters

Hyperparameter tuning is very important for performance of EPO. With poor hyperparameter, EPO fails to achieve fluency. However, we have not performed systematic hyperparameter searches with EPO. The parameters we use here are roughly designed to favor variety and exploration instead of fast convergence. Applying GCG or EPO in other settings may benefit from very different hyperparameters. For example, in our work on adversarial attacks, we use $k = 32$ instead of $k = 512$. In our rough experience, using a lower value of k results in faster convergence but significantly less fluent prompts. An optimal implementation would likely increase k as the number of iterations increases. Early in the algorithm, small k results in fast convergence towards a basin of attraction. Late in the algorithm,

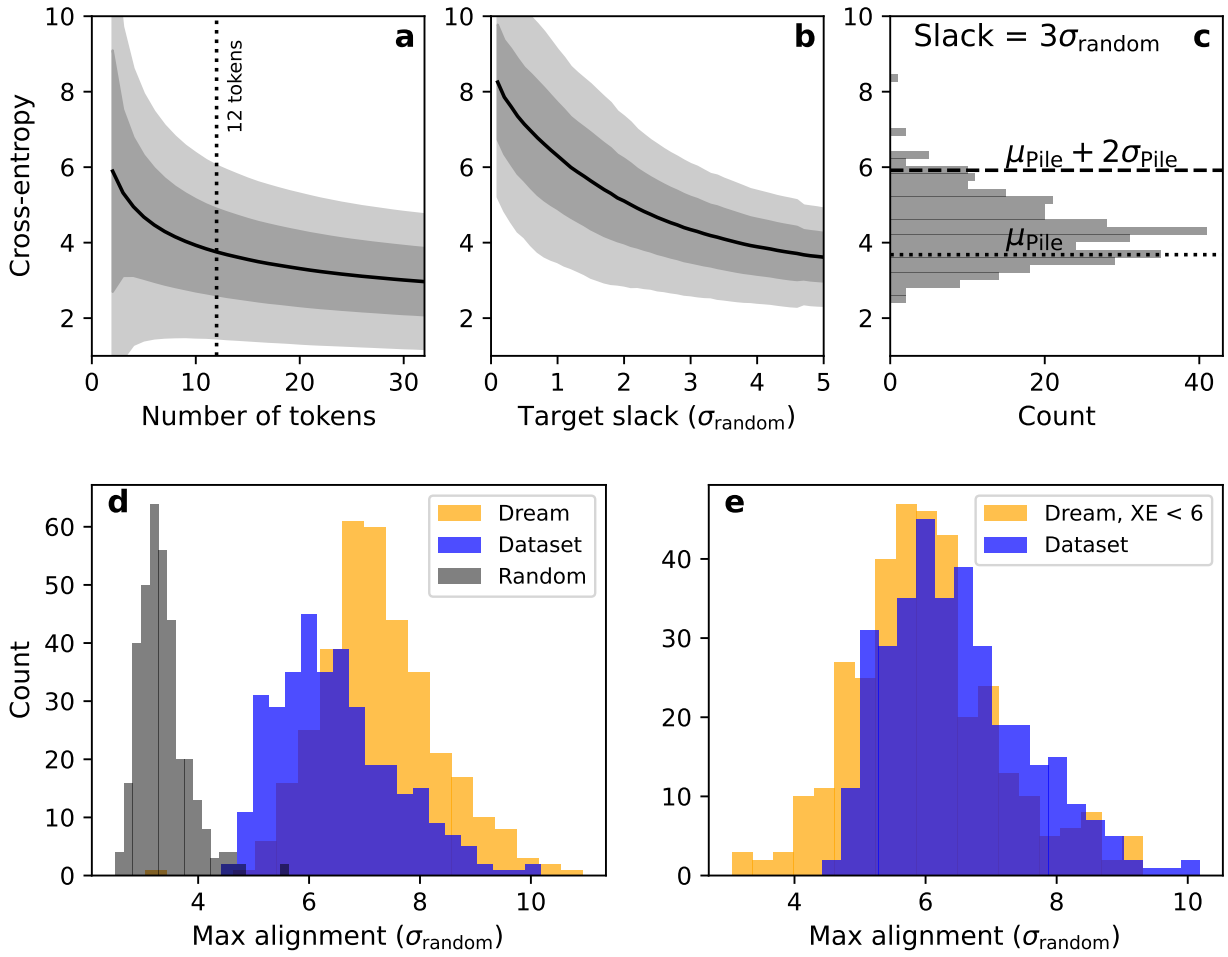

Figure 5: **a**) Average cross-entropy plotted against number of tokens for text in the Pile. The dark gray region shows one standard deviation above and below average cross-entropy while the light gray region shows two standard deviations. Note that all the optimized prompts in this paper are 12 tokens in length. **b**) We plot average cross-entropy across the residual vector alignment dreaming runs as a function of slack below maximum alignment as measured in units of random prompt standard deviations. See the text for a discussion of the units. **c**) The distribution of cross-entropy for dreaming results with a slack of three standard deviations.**d**) The distribution over 50 random target vectors of maximum alignment for each method. **e**) Similar to d but, in order to approximately restrict the analysis to in-distribution prompts, we restrict the dreaming outputs to have cross-entropy below 6. Note that we do not restrict the cross-entropy of the max-activating dataset prompts because they are known to be in-distribution.

large k allows enough exploration to continue improving the solutions. The hyperparameters we use here are designed for a target on the same scale as the cross-entropy. Optimization targets on different scales will require different hyperparameters or scaling.

EPO is computationally slow because it requires a forward pass of the model for each candidate prompt. With the hyperparameters we have used here, a single optimization job runs in about 200 seconds on a single RTX A6000 GPU. In general, discrete optimization is quite expensive though we are optimistic that there is substantial algorithmic progress available for dreaming.

6 Conclusions

In this paper, we presented the first successful algorithm for fluent dreaming in language models. Our algorithm, Evolutionary Prompt Optimization (EPO) supplements discrete search methods with token gradients to construct a

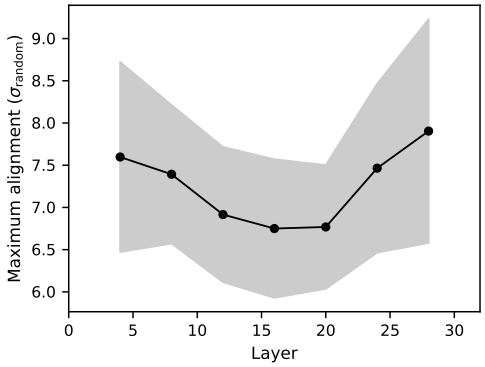

Figure 6: We apply EPO to optimize residual stream alignment for 50 random vectors in each of the plotted layers. We then compute the mean and standard deviation of the maximum alignment z-score.

Pareto frontier between fluency and feature activation. We demonstrate EPO producing high-activation prompts that maintain linguistic coherence. Dreaming with EPO is especially valuable as a technique for exploring out-of-distribution behavior inside a language model.

We see a variety of potential improvements and extensions to EPO. EPO and other algorithms based on single token swaps get trapped in local minima when optimizing for fluency. This is especially true when an important single word is composed of two tokens. Swapping either token alone will dramatically reduce fluency and may not have a countervailing increase in activation. Swapping both tokens, on the other hand, could improve fluency and activation. Modifications to EPO that allow insertion and deletion of tokens could be another fruitful direction. Like GCG (Zou et al. 2023), our algorithm only evaluates the gradient at the current sequence. However, ARCA (Jones et al. 2023) suggests the possibility of averaging gradients at multiple neighbors to improve the quality of first-order approximation. In general, GCG-based methods can get stuck in local minima very quickly. (see Figure 2a). We are also interested in algorithmic modifications that would result in more exploration. Methods that increase exploration would hopefully result in more consistent optimization results when initialized from different random seeds.

EPO is likely to be valuable well beyond fluent dreaming. The prompt fluency benefits of EPO may aid adversarial attacks for red-teaming by evading perplexity-based filters or other basic oversight techniques. EPO can also support automatic prompt engineering or optimization of any function of the prompt that is differentiable with respect to the embedding. Algorithms similar to EPO may also be useful for controllable generation similar to Kumar, Paria, and Tsvetkov (2022). In sum, EPO is a versatile choice for fluently optimizing a language model's inputs to maximize a differentiable objective.

Acknowledgements

We thank Martin Wattenberg for an early conversation that inspired this work.

References

Bäuerle, Alex and James Wexler (2018). *What does BERT dream of?* URL: https://pair-code.github.io/interpretability/text-dream/blogpost/.

Biderman, Stella et al. (2023). *Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling*. arXiv: 2304.01373 [cs.CL].

Bills, Steven et al. (2023). *Language models can explain neurons in language models*. https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html.

Bolukbasi, Tolga et al. (2021). *An Interpretability Illusion for BERT*. arXiv: 2104.07143 [cs.CL].

Bricken, Trenton et al. (2023). "Towards Monosemanticity: Decomposing Language Models With Dictionary Learning". In: *Transformer Circuits Thread*. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Cammarata, Nick et al. (2020). "Thread: Circuits". In: *Distill*. https://distill.pub/2020/circuits. DOI: 10.23915/distill.00024.

Cunningham, Hoagy et al. (2023). *Sparse Autoencoders Find Highly Interpretable Features in Language Models*. arXiv: 2309.08600 [cs.LG].

Ebrahimi, Javid et al. (2018). *HotFlip: White-Box Adversarial Examples for Text Classification*. arXiv: 1712.06751 [cs.CL].

Elhage, Nelson et al. (2021). "A Mathematical Framework for Transformer Circuits". In: *Transformer Circuits Thread*. https://transformer-circuits.pub/2021/framework/index.html.

Elhage, Nelson et al. (2022). "Toy Models of Superposition". In: *Transformer Circuits Thread*. https://transformer-circuits.pub/2022/toy_model/index.html.

Gurnee, Wes et al. (2023). *Finding Neurons in a Haystack: Case Studies with Sparse Probing*. arXiv: 2305.01610 [cs.LG].

Jang, Eric, Shixiang Gu, and Ben Poole (2017). *Categorical Reparameterization with Gumbel-Softmax*. arXiv: 1611.01144 [stat.ML].

Jones, Erik et al. (2023). *Automatically Auditing Large Language Models via Discrete Optimization*. arXiv: 2303.04381 [cs.LG].

Kumar, Sachin, Biswajit Paria, and Yulia Tsvetkov (2022). *Gradient-Based Constrained Sampling from Language Models*. arXiv: 2205.12558 [cs.CL].

Mordvintsev, Alexander, Christopher Olah, and Mike Tyka (2015). *Inceptionism: Going Deeper into Neural Networks*. URL: https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html.

Olah, Chris, Alexander Mordvintsev, and Ludwig Schubert (2017). "Feature Visualization". In: *Distill*. https://distill.pub/2017/feature-visualization. DOI: 10.23915/distill.00007.

Olah, Chris et al. (2018). "The Building Blocks of Interpretability". In: *Distill*. https://distill.pub/2018/building-blocks. DOI: 10.23915/distill.00010.

Poerner, Nina, Benjamin Roth, and Hinrich Schütze (Nov. 2018). "Interpretable Textual Neuron Representations for NLP". In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Ed. by Tal Linzen, Grzegorz Chrupaa, and Afra Alishahi. Brussels, Belgium: Association for Computational Linguistics, pp. 325–327. DOI: 10.18653/v1/W18-5437. URL: https://aclanthology.org/W18-5437.

Shi, Weijia et al. (2022). *Toward Human Readable Prompt Tuning: Kubrick's The Shining is a good movie, and a good prompt too?* arXiv: 2212.10539 [cs.CL].

Shin, Taylor et al. (2020). *AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts*. arXiv: 2010.15980 [cs.CL].

Szegedy, Christian et al. (2014). *Intriguing properties of neural networks*. arXiv: 1312.6199 [cs.CV].

Wen, Yuxin et al. (2023). *Hard Prompts Made Easy: Gradient-Based Discrete Optimization for Prompt Tuning and Discovery*. arXiv: 2302.03668 [cs.LG].

Yosinski, Jason et al. (2015). *Understanding Neural Networks Through Deep Visualization*. arXiv: 1506.06579 [cs.CV].

Yuan, Lifan et al. (2023). *Bridge the Gap Between CV and NLP! A Gradient-based Textual Adversarial Attack Framework*. arXiv: 2110.15317 [cs.CL].

Zhu, Sicheng et al. (2023). "AutoDAN: Automatic and Interpretable Adversarial Attacks on Large Language Models". In: *arXiv preprint arXiv:2310.15140*.

Zou, Andy et al. (2023). *Universal and Transferable Adversarial Attacks on Aligned Language Models*. arXiv: 2307.15043 [cs.CL].

A Additional Tables

Table 5: Results of running EPO on six neurons.

layer	neuron	Activation	Perplexity	Text
2	0	-0.05	42.36	HRC) and other discoursing members held a meeting
	0	2.54	79.13	HRC) and other discoursing organizations. In Chap
	0	2.59	81.64	HRC) and other discoursing agencies. In Chap
	0	3.93	262.52	HRC) and other discoursing agencies". In
	0	4.71	549.27	HRC) and other discoursing organizationsPosted In
	0	5.89	2,056.80	Record Company) and other discoursing folkspackage.[^
	0	6.22	9,698.12	Record Company) recwith discoursing pleapackage][^
	0	6.71	156,519.75	OPLE COURT) st AdditionalAttoursshinelicense CopyrightSent
7	1	0.14	40.18	. The reason why Dadaab was created, and the
	1	0.22	41.37	). The reason why Dadaab was created, and the
	1	1.98	111.59	The reason why Dadaab was created initially the
	1	2.45	187.61	The mystery how a thought actually was created, on what
	1	2.54	232.58	The reason why Dadaab was created 1919 the
	1	3.13	1,030.19	.** TheReason behind Dada_ was established originally and registered
	1	3.41	3,099.71	”’ The reason why Horus actually was awaited what
12	2	4.61	57.22	KESTON.... has not escaped entirely un
	2	4.95	84.56	HELSTON... but has not escaped entirely un
	2	5.11	106.07	MORESTON... but has not escaped entirely un
	2	6.00	642.16	TeV. Yet our own modest sum was not entirely escaped un
	2	6.32	1,297.21	TeV. Yet our own modest on was not entirely escaped un
	2	6.75	5,928.34	WESTINGTON ration per-It hardware not escaped entirely un
	2	6.78	14,558.60	WEabINGTON ration per-It hardware not escaped entirely un
17	3	0.13	30.15	STV is the best format on TV, the radio and
	3	2.10	48.66	OPLE TODAY is the newest hit on TV, streaming online and
	3	2.55	55.89	Quest is the newest hit on TV, streaming online or
	3	2.60	57.22	Books is the newest hit on TV, streaming online or
	3	2.84	65.86	Theatre is the newest hit on TV, streaming online or
	3	2.90	73.18	Theaterő is the latest hit on TV, streaming online or
	3	3.00	87.93	Underground is the interactive hit on TV, streaming online or
	3	3.03	93.60	Theaterő is the interactive hit on TV, streaming online or
	3	3.18	147.83	Museum is the interactive hit on TV, streaming online or
	3	3.54	3,595.72	UCLA section, The CW incarnate ABC form currently 24 USA
	3	3.61	6,770.38	UCLA section, The CW incarnative ABC form currently 24 USA
	3	3.62	7,853.78	UCLA work, The CW incarnative ABC form currently 24 USA
22	4	2.60	85.56	BURG Code and SA 2000-01:9, a
	4	4.84	377.51	Wendel Code and SA 24-9:9 limited a
	4	5.18	542.87	Keel Code and SA 2000-7:9 limited a
	4	5.41	1,665.65	Iz VII Code and SA 2000-179:14 limited a
27	5	-0.11	24.80	wont get it. I think Jace is the best.
	5	2.27	28.94	wont get it. I think Jace is the best m
	5	2.66	38.12	wont win it. I think Jace is the best m
	5	3.02	61.87	wont get it. I think Jace is the surprise m
	5	3.30	100.81	wont get taken. I think Jace is the surprise m
	5	4.52	1,204.42	naive poker is. ALL predict JLO being the first m
	5	4.59	1,558.63	selection poker is. ALL predict JLO being the first m