Introduction:

Deep Dream:

A landmark result in mechanistic interpretability research was DeepDream. Unfortunately, there is no similarly successful approach for language model interpretability. Previous attempts to optimize a sentence in order to maximally activate any given neuron often produce gibberish text - https://pair-code.github.io/interpretability/text-dream/blogpost/

Other work has shown that in e.g. sentiment analysis, optimizing for a positive or negative sentiment can tell you what types of words and sentence combinations the model thinks contribute to sentiment. However, when this optimization is applied to internal neurons, incoherent sentences are often generated.

An important lesson from deep dream is that **careful regularization techniques are needed** in order to get images that look recognizable. Therefore, we work on finding a suitable regularization to the search.

Large Language model Interpretability

Why do we want sentences that maximally activate a neuron? Previous work from **OpenAI** has demonstrated one possible approach to generating explanations for how neurons behave, given sentences along with per token activation information. - https://openai.com/research/language-models-can-explain-neurons-in-language-models

However, this approach often generates explanations that are less accurate than human raters, and overall the vast majority of the explanations are not descriptive. A wholly grail would be generating sentences that clearly demonstrate the use of a neuron.

Our Contributions:

1. We create a novel regularization technique to constrain neural activation optimization, which leads to more coherent sentences.
2. We explain how to measure the effectiveness of this approach in generating sentences that allow GPT-4 to come up with better explanations of neurons.

Further work we build on is the logit lens. In that work, they are able to decode embedded vectors throughout the model by using the embedding matrix. We similarly use the embedding matrix to

Deep dream works on images, however, baseline implementations of deep dream produce garbage. We show in the appendix that optimizing the embeddings from an auto-encoder produces much more realistic images on MNIST than directly optimizing the image.

Our Method

1. Instead of allowing the optimization to occur over the entire 768 dimension embedding space, we restrict the optimization to occur over a k dimensional space where k ranges from 10-100.
2. We train an encoder (a separate gpt2) and decoder (both a 6 layer transformer from scratch and gpt2)
3. Generate random sentences or grab sentences from the training corpus, embed them in the latent space, optimize this embedding to activate the target neuron, then decode the embedding with the aute-encoder and the transpose of the embedding matrix.

Experiments

We experiment on distillgpt-2, and utilize colab notebooks with a 16GB gpu. Furthermore, we focus our attention on activations in the MLP layer.

Baseline

For our baseline, we implement the following:

1. Start with a random, GPT-4 generated sentence
2. Embed the sentence using the embedding matrix, and make this a tensor with an optimizer on it.
3. Pick an arbitrary neuron in the network, and optimize the embeddings to increase the mean activation of this neuron across all tokens in the embedded sentence.
    a. We find that optimizing the sigmoid of the activation leads to slightly better sentences.

Starting sentence:

```
I'm sorry for the misunderstanding, but as an AI developed by OpenAI, I don't have direct access to individual sentences or documents from my training data.
```

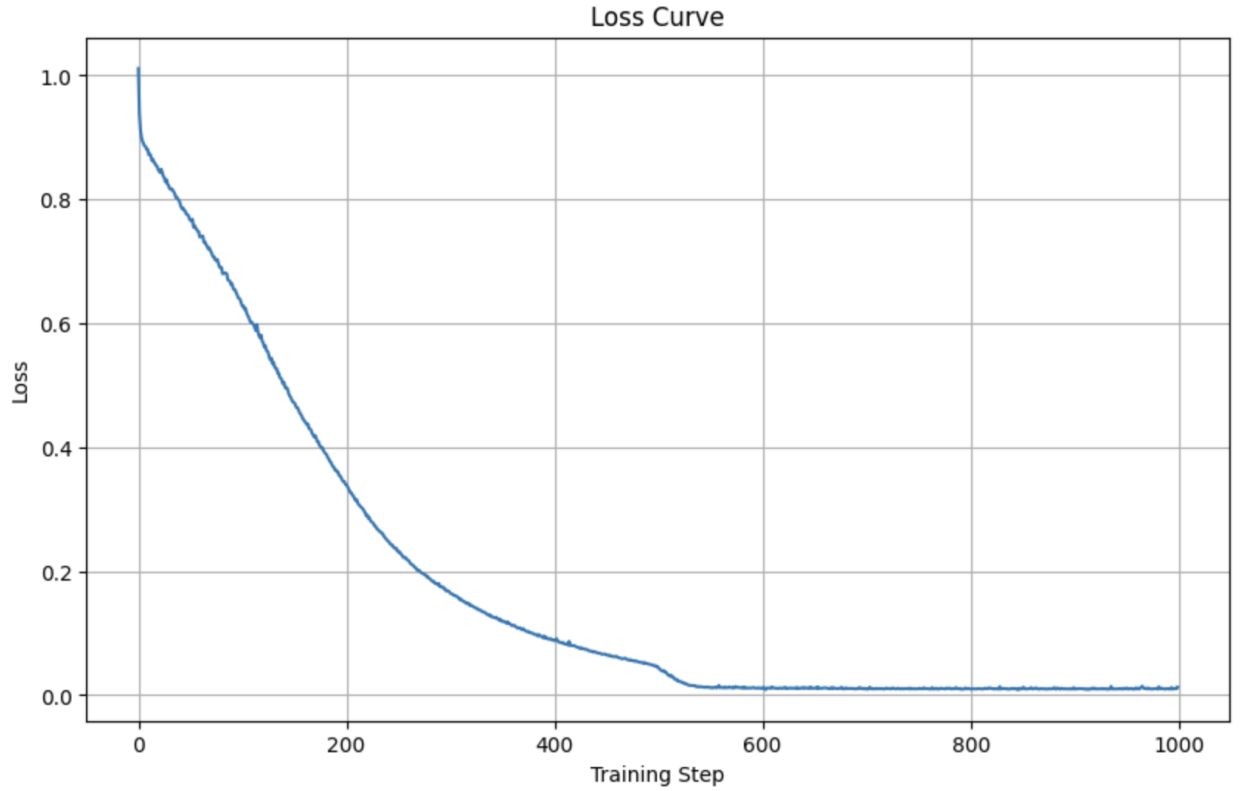
Transformed sentence to activate e.g. neuron 2 in layer 5 of the MLP.

```
cue am�� XIIIhes misunderstandinghes butafterair laptus-[ OpenAIsoevermy cannotno
```

One thing we noticed from the baseline method is that Neuron 2 in layer 1 of the MLP, often sentences include the word “Download” in several places when starting from a random GPT-4 generated sentence.

Our Method

Autencoder loss going down in embedding space



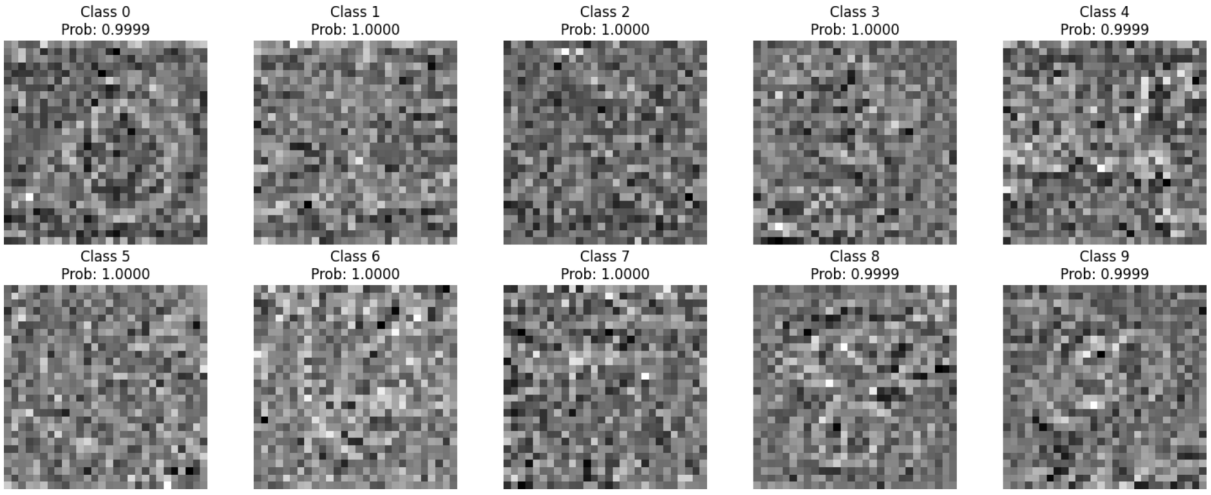
Unfortunately auto-encoder doesn't successfully reconstruct sentences, preventing downstream possibilities.

Future work:

1. Training better autoencoders
2. Placing the auto-encoder in intermediate layers, not just after the first embedding layer.
3. Instead of searching with gradient descent, search over tokens restricted to only the most plausible tokens as judged by the language model itself.

Appendix:

Before auto-encoder optimization constraint applied:



After Auto-encoder optimization constraint applied: