Homework 1 Report

maochuan 2300013218

March 30, 2025

1 Coding Summary

This section briefly summarizes the algorithms and implementations in Part One, including code completion, debugging experiences, and key ideas.

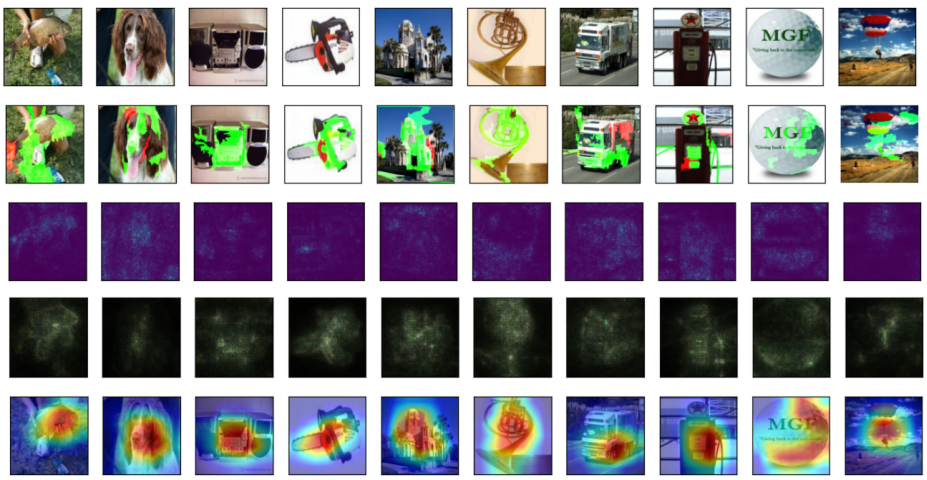

Figure 1: experiment results

1.1 LIME

I used the `lime_image` module to obtain a `LimeImageExplainer` instance. The `explain_instance` method automatically selects images in the local region and assigns labels to them (by default, using a batch size of 10). It then trains a small surrogate model to determine which segments contribute the most to the classification.

Initially, I normalized the image using incorrect mean and standard deviation values, which caused the model to predict incorrect labels. I mistakenly believed that the goal was to make the explainer justify misclassifications. However, the program still threw errors. Eventually, I discovered that the explainer can only explain labels that rank within the top k probabilities.

1.2 Saliency Map

The model output is backpropagated, and gradients are computed using automatic differentiation. Since we process 10 images as a batch, calling `loss.backward()` directly does not work. Instead, I used:

```
output.backward(torch.ones_like(output))
```

to compute the gradient.

1.3 SmoothGrad

$$\text{SmoothGrad}(x) = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial f(x_i)}{\partial x_i} \tag{1}$$

Compared to the standard Saliency Map algorithm, the SmoothGrad algorithm runs for multiple epochs, adding Gaussian noise to the image at each iteration. Using a fixed standard deviation for the noise distribution, I generated multiple saliency maps. Finally, I computed their mean and normalized the result.

1.4 Grad-CAM

In Grad-CAM, for a specific layer and a specific class, class `ActivationsAndGradients` record activations during the forward pass and gradients during the backward pass. The importance weights are computed by taking the mean of the gradients across the spatial dimensions. The final class activation map is obtained using the ReLU function:

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \tag{2}$$

$$L_{\text{grad-cam}}^c = \text{ReLU}\left(\sum_k \alpha_k^c A^k \right) \tag{3}$$

The codebase supports computing class activation maps (CAMs) for multiple layers. However, in our tests, we only considered one layer. To adapt to the dimensions of tensor, I used `unsqueeze()`, `expand_dims()` and resized it accordingly. When resizing the CAM from 7×7 to 224×224, I added a `target_size` parameter to the `get_cam_image` function to ensure proper alignment.

2 Paper Reading Report

Paper: ***Sparse Autoencoders Find Highly Interpretable Features in Language Models***

2.1 Summary

In the domain of explainable models, one of the key challenges is **polysemanticity**, which arises due to **Superposition**—a phenomenon where individual neurons activate for multiple unrelated features. This work demonstrates that it is possible to resolve

superposition in language models using a scalable, unsupervised method based on sparse dictionary learning.

The authors introduce sparse autoencoders as a means to interpret internal activations in language models. By training an autoencoder with an ℓ_1 penalty, they recover a set of *dictionary features* that exhibit greater interpretability and monosemanticity compared to traditional bases, such as neurons, PCA, or ICA directions. Experimental results indicate that these learned features not only achieve higher automated interpretability scores but also enable more precise causal interventions, providing insight into the internal representations of large language models.

2.2 Claims and Evidence

The paper makes several key claims, each substantiated by rigorous experimental evidence:

- **Improved Interpretability:** Sparse autoencoders yield dictionary features that are more interpretable than raw neuron activations and other dimensionality reduction techniques. This claim is supported by quantitative autointerpretability scores computed using GPT-based methods, which show that the learned dictionary features align more closely with human-interpretable concepts.
- **Causal Localization:** The learned dictionary features facilitate a finer-grained localization of causal effects in language model behavior. For example, in the Indirect Object Identification (IOI) task, activation patching experiments demonstrate that modifying a small number of dictionary features is sufficient to induce significant behavioral changes in model predictions. The paper quantitatively compares the number of patched features required to achieve a target KL divergence against alternative representations, showing a clear advantage for sparse dictionary features.
- **Scalability and Applicability:** The approach is shown to scale effectively to large language models and can be applied to different network components, such as residual streams and, with some limitations, MLP layers. The authors validate this claim by conducting experiments on multiple model sizes, including Pythia-70M and Pythia-410M, and by demonstrating the method's applicability across various layers of the transformer architecture.

2.3 Methods and Evaluation Criteria

The methodology relies on sparse dictionary learning, implemented through the following key components:

- **Sparse Autoencoder:** The model reconstructs activation vectors x by enforcing a sparse latent representation c. The encoding process is defined as:

$$c = \text{ReLU}(Mx + b) \tag{4}$$

  and the reconstruction is given by:

$$\hat{x} = M^T c. \tag{5}$$

- **Loss Function:** The training objective minimizes a loss function that balances reconstruction fidelity with sparsity:

$$L(x) = |x - \hat{x}|_2^2 + \alpha |c|_1, \tag{6}$$

where α controls the trade-off between reconstruction accuracy and sparsity in the learned representation.

- **Evaluation:** Interpretability is assessed using an automated protocol based on GPT-generated descriptions of dictionary features. By using these descriptions as prompts and analyzing their activation patterns, the authors obtain interpretability scores for individual dictionary features. These scores are then compared against various baselines, revealing an explicit improvement in interpretability, particularly in the earlier transformer layers. Furthermore, the study employs activation patching to assess the causal role of learned features, demonstrating that dictionary features allow for targeted interventions with fewer patches to achieve a given KL divergence threshold.

2.4 Experimental Designs or Analyses

The experimental design consists of the following components:

- **Data Collection:** Activation vectors are sampled from language models (e.g., Pythia-70M and Pythia-410M) while processing large-scale corpora such as the Pile.

- **Training Procedure:** Sparse autoencoders are trained on these activation vectors, varying the dictionary size and sparsity coefficient to explore different trade-offs.

- **Hyperparameter Selection:** The authors systematically vary two key hyperparameters: R, the dictionary expansion factor, and α, the regularization strength. By training multiple dictionary models with different parameter settings, they mitigate randomness and strengthen the credibility of their findings.

- **Comparisons with Baselines:** The sparse dictionary features are compared against alternative representations such as PCA and ICA using both quantitative metrics (autointerpretability scores) and qualitative case studies. Additionally, KL divergence is used to evaluate the effectiveness of activation patching. Results indicate that modifying a small number of dictionary features achieves a closer match to the target model's behavior compared to patching PCA components or features from non-sparse ($\alpha = 0$) dictionaries.

- **Causal Analysis:** The study leverages activation patching in the IOI task to investigate the causal effects of individual dictionary features. The results show that a small set of dictionary features is sufficient to modulate the model's predictions, highlighting their localized and semantically meaningful nature.

- **Case Studies:** The authors select specific dictionary features that correspond to human-understandable concepts and perform three detailed analyses to infer their semantic meanings:

- **Activation Histograms:** The distribution of feature activations across tokens is examined to identify semantically coherent clusters.
- **Feature Ablation:** The impact of removing specific features from the residual stream is tested. For instance, ablation of the "apostrophe" feature primarily reduces the logit probability of the following "s" (as in possessive constructions and contractions like "let's").
- **Causal Dependency Trees:** The authors construct a causal dependency tree to visualize cross-layer relationships among dictionary features, providing further insight into their functional roles in the model.

2.5 Other Strengths and Weaknesses

Strengths:

- The proposed approach addresses a core challenge in mechanistic interpretability—polysemanticity arising from superposition—by isolating monosemantic features through the use of sparse autoencoders with a single hidden layer.
- The method significantly outperforms naive baselines (e.g., PCA, ICA) in terms of automated interpretability scores, indicating a more human-aligned feature representation.
- The approach is scalable and computationally efficient compared to full retraining of large models, making it viable for practical use in analyzing frontier AI systems.
- The study provides both quantitative and qualitative analyses, which together strengthen the evidence for improved interpretability and causal localization in the internal activations of language models.

Weaknesses:

- There are inherent limitations to the single-layer autoencoder approach. In particular, the reconstruction loss remains non-zero, which raises concerns about whether all true latent features can be fully captured by such a shallow model.
- It remains unclear whether enumerating all features is a viable strategy for ensuring AI safety, as the approach may miss critical aspects of the model's internal representation.
- The method performs less robustly in later layers and on MLP sublayers, with many features remaining inactive, suggesting that further refinement is needed for non-residual components.
- The trade-off between sparsity and reconstruction accuracy might limit the method's effectiveness in capturing the complex, high-dimensional features present in later layers of the model.
- Interpretability, as a concept, lacks universally agreed-upon metrics, making it challenging to rigorously evaluate and compare the proposed method against baselines. The study is therefore partially qualitative and builds on prior work in the field, which reduces the contribution of the work to a certain extent.