4.2 Recovering known features with 1d probes

If we expect that a specific feature (e.g sentiment, language identification) should be discovered by a high quality autoencoder, then one metric of autoencoder quality is to check whether these features are present. Based on this intuition, we curated a set of 61 binary classification datasets (details in Table 1). For each task, we train a 1d logistic probe on each latent using the Newton-Raphson method to predict the task, and record the best cross entropy loss (across latents).[12] That is:

$$\min_{i,w,b} \mathbb{E}\left[y \log \sigma\left(wz_i + b\right) + (1-y) \log\left(1 - \sigma\left(wz_i + b\right)\right)\right] \tag{4}$$

where z_i is the ith pre-activation autoencoder latent, and y is a binary label.

Results on GPT-2 small are shown in Figure 6a. We find that probe score increases and then decreases as k increases. We find that TopK generally achieves better probe scores than ReLU (Figure 23), and both are substantially better than when using directly residual stream channels. See Figure 32 for results on several GPT-4 autoencoders: we observe that this metric improves throughout training, despite there being no supervised training signal; and we find that it beats a baseline using channels of the residual stream. See Figure 33 for scores broken down by component.

This metric has the advantage that it is computationally cheap. However, it also has a major limitation, which is that it leans on strong assumptions about what kinds of features are natural.

4.3 Finding simple explanations for features

Anecdotally, our autoencoders find many features that have quickly recognizable patterns that suggest explanations when viewing random activations (Section E.1). However, this can create an "illusion" of interpretability [Bolukbasi et al., 2021], where explanations are overly broad, and thus have good recall but poor precision. For example, Bills et al. [2023] propose an automated interpretability score which disproportionately depends on recall. They find a feature activating at the end of the phrase "don't stop" or "can't stop", but an explanation activating on all instances of "stop" achieves a high interpretability score. As we scale autoencoders and the features get sparser and more specific, this kind of failure becomes more severe.

Unfortunately, precision is extremely expensive to evaluate when the simulations are using GPT-4 as in Bills et al. [2023]. As an initial exploration, we focus on an improved version of Neuron to Graph (N2G) [Foote et al., 2023], a substantially less expressive but much cheaper method that outputs explanations in the form of collections of n-grams with wildcards. In the future, we would like to explore ways to make it more tractable to approximate precision for arbitrary English explanations.

To construct a N2G explanation, we start with some sequences that activate the latent. For each one, we find the shortest suffix that still activates the latent.[13] We then check whether any position in the n-gram can be replaced by a padding token, to insert wildcard tokens. We also check whether the explanation should be dependent on absolute position by checking whether inserting a padding token at the beginning matters. We use a random sample of up to 16 nonzero activations to build the graph, and another 16 as true positives for computing recall.

Results for GPT-2 small are found in Figure 25a and 25b. Note that dense token patterns are trivial to explain, thus $n = 2048, k = 512$ latents are easy to explain on average since many latents activate extremely densely (see Section E.5)[14]. In general, autoencoders with more total latents and fewer active latents are easiest to model with N2G.

We also obtain evidence that TopK models have fewer spurious positive activations than their ReLU counterparts. N2G explanations have significantly better recall (>1.5x) and only slightly worse precision (>0.9x) for TopK models with the same n (resulting in better F1 scores) and similar L_0 (Figure 24).

[12] This is similar to the approach of Gurnee et al. [2023], but always using $k = 1$, and regressing on autoencoder latents instead of neurons.

[13] with at least half the original activation strength

[14] This highlights an issue with our precision/recall metrics, which care only about binarized values. We also propose a more expensive metric which uses simulated values Section 4.4 and addresses this issue.

(a) A feature with precision = 0.97, recall = 0.56

Recall Eval Sequences:

\n\nTherefore she's not going to play pranks but to give a sweet time-a rendezvous

choice. [Warning: Ripe with Wine Puns]\n\nVital Lacerda (September

. [Warning: Developed with Evolutionary Puns]\n\nMailbags\n\nTrial &

al6606 Luke.A.M mepwn12 Thorny [B]Available voice actors,

his picture of President Putin.\n\nPresident Prankster\n\nDmitry Rogozin,

Precision Eval Sequences:

away:\n\nItching.\n\nPrickling.\n\nThese are not all of the

% 70% 99%\n\n71 NoPwn1ntended i7-950 3.4

ottonee Whimsicott M 0 Prankster / Infiltrator Prankster / Inf

of Chimera players using one-dimensional teams: Prank

of Chimera players using one-dimensional teams: Prankster SporeSeed, Shell Smash, Poison

(b) A feature with precision = 0.06, recall = 0.05

Recall Eval Sequences:

\nMeal 1 06h30 Oats, double serving of Whey, 6

's.\n\n04Jul65 to 01Aug65\n\n1/3 un

at\n\n2013-11-16 09:47:18 (25) chat: <DiamondCard

0000 - 002600076] 119 pages 5: [00260

, calm it2013-11-11 20:00:43 (25) chat: It's vanity

Precision Eval Sequences:

1 Tue Mar 13 22:37:59 EST 2001 i686 Locale: LANG=C

Kiwi76 added 01:08 - Apr 28\n\nCheers Clive and hopefully you can see

.aspx?key=248858886|000100&CMC=&PN=&Is

.aspx?key=248858886|000101&CMC=&PN=&Is

key=248858886|000102&CMC=&PN=&Is

Figure 7: Qualitative examples of latents with high and low precision/recall N2G explanations. Key: Green = Ground truth feature activation, Underline = N2G predicted feature activation

4.4 Explanation reconstruction

When our goal is for a model's activations to be interpretable, one question we can ask is: how much performance do we sacrifice if we use only the parts of the model that we can interpret?

Our downstream loss metric measures how much of the performance we're capturing (but our features could be uninterpretable), and our explanation based metric measures how monosemantic our features are (but they might not explain most of the model). This suggests combining our downstream loss and explanation metrics, by using our explanations to simulate autoencoder latents, and then checking downstream loss after decoding. This metric also has the advantage that it values both recall and precision in a way that is principled, and also values recall more for latents that activate more densely.

We tried this with N2G explanations. N2G produces a simulated value based on the node in the trie, but we scale this value to minimize variance explained. Specifically, we compute $E[sa]/E[s^2]$, where s is the simulated value and a is the true value, and we estimate this quantity over a training set of tokens. Results for GPT-2 are shown in Figure 8. We find that we can explain more of GPT-2 small than just explaining bigrams, and that larger and sparser autoencoders result in better downstream loss.

4.5 Sparsity of ablation effects

If the underlying computations learned by a language model are sparse, one hypothesis is that natural features are not only sparse in terms of activations, but also in terms of downstream effects [Olah et al., 2024]. Anecdotally, we observed that ablation effects often are interpretable (see our visualizer). Therefore, we developed a metric to measure the sparsity of downstream effects on the output logits.

At a particular token index, we obtain the latents at the residual stream, and proceed to ablate each autoencoder latent one by one, and compare the resulting logits before and after ablation. This process leads to V logit differences per ablation and affected token, where V is the size of the token vocabulary. Because a constant difference at every logit does not affect the post-softmax probabilities, we subtract at each token the median logit difference value. Finally, we concatenate these vectors together across some set of T future tokens (at the ablated index or later) to obtain a vector of

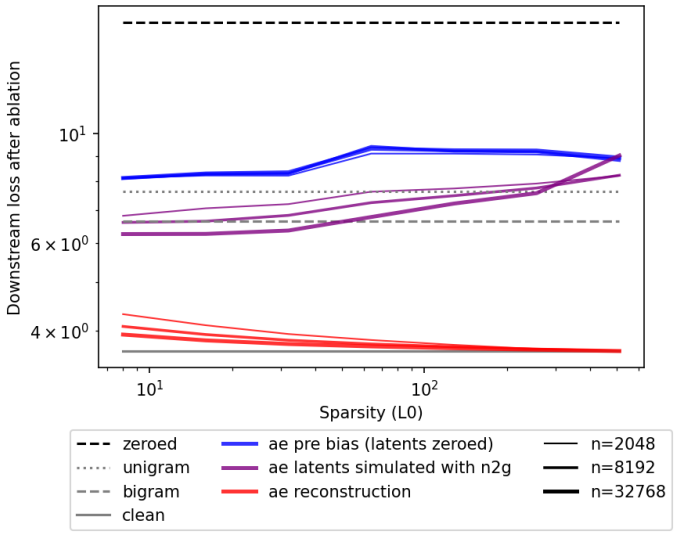

Figure 8: Downstream loss on GPT-2 with various residual stream ablations at layer 8. N2G explanations of autoencoder latents improves downstream loss with larger n and smaller k.

$V \cdot T$ total numbers. We then measure the sparsity of this vector via $(\frac{L_1}{L_2})^2$, which corresponds to an "effective number of vocab tokens affected". We normalize by $V \cdot T$ to have a fraction between 0 and 1, with smaller values corresponding to sparser effects.

We perform this for various autoencoders trained on the post-MLP residual stream at layer 8 in GPT-2 small, with $T = 16$. Results are shown in Figure 6b. Promisingly, models trained with larger k have latents with sparser effects. However, the trend reverses at $k = 512$, indicating that as k approaches $d_{\text{model}} = 768$, the autoencoder learns latents with less interpretable effects. Note that latents are sparse in an absolute sense, having a $(\frac{L_1}{L_2})^2$ of 10-14% , whereas ablating residual stream channels gives 60% (slightly better than the theoretical value of $\sim \frac{2}{\pi}$ for random vectors).

5 Understanding the TopK activation function

5.1 TopK prevents activation shrinkage

A major drawback of the L_1 penalty is that it tends to shrink all activations toward zero [Tibshirani, 1996]. Our proposed TopK activation function prevents activation shrinkage, as it entirely removes the need for an L_1 penalty. To empirically measure the magnitude of activation shrinkage, we consider whether different (and potentially larger) activations would result in better reconstruction given a fixed decoder. We first run the encoder to obtain a set of activated latents, save the sparsity mask, and then optimize only the nonzero values to minimize MSE.[15] This refinement method has been proposed multiple times such as in k-SVD [Aharon et al., 2006], the relaxed Lasso [Meinshausen, 2007], or ITI [Maleki, 2009]. We solve for the optimal activations with a positivity constraint using projected gradient descent.

This refinement procedure tends to increase activations in ReLU models on average, but not in TopK models (Figure 9a), which indicates that TopK is not impacted by activation shrinkage. The magnitude of the refinement is also smaller for TopK models than for ReLU models. In both ReLU and TopK models, the refinement procedure noticeably improves the reconstruction MSE (Figure 9b), and the downstream next-token-prediction cross-entropy (Figure 9c). However, this refinement only closes part of the gap between ReLU and TopK models.

[15] unlike the "inference-time optimization" procedure in Nanda et al. [2024]