Using Sparse Autoencoders to Identify Danish Language Context Neurons in a Language Model

Martin Kirkegaard (marki@itu.dk) & Markus Sibbesen (mksi@itu.dk)

January 16, 2025

Abstract

When analyzing the inner activations of transformer based Large Language Models (LLMs), one would ideally want one neuron to correspond to one interpretable feature. This is, however, made difficult by the phenomenon of *superposition*, where neural networks fit in more features than there are neurons, by each feature being represented by a combination of multiple neurons, causing each neuron to represent multiple features (meaning each neuron is *polysemantic*). *Sparse Autoencoders* (SAEs) have emerged as a promising method for separating features into interpretable *monosemantic* neurons.

In this paper, we fit an SAE on a different size models from the Pythia suit and use it to identify neurons representing the feature of Danish language text. First, we use *Classification probes* to investigate a range of different size language models, to find which model (and which layer) would be a good candidate for identifying Danish language features. We find layer 16 of the 1.4 billion parameter model to be a suitable choice, fit an SAE on the multi layer perceptron activations and use it to identify 6 neurons overrepresented in Danish text.

We measure the performance of the model both on next token prediction and a range of downstream tasks in both English and Danish and find that artificially lowering the activation of the Danish language neurons impacts the performance on Danish tasks considerably more than English.

While these results provide evidence that the neurons identified are really Danish-specific, we advise caution in interpreting the results. More research is needed regarding how they behave on other circumstances, e.g. in the context of other languages, and how they interact with the rest of the neural network, before we can pin down their exact function.[1]

1 Introduction

It is largely a mystery how modern transformer-based Large Language Models (LLMs) work. After being trained on immense datasets[2], they have gained considerable capabilities in not only natural language processing tasks (Qin et al. 2023), but also programming (Hadi et al. 2024) and mathematical reasoning (Yuan et al. 2023), but what goes on inside the models which allows them to do so, is difficult to figure out. This has particular relevance, since language models have been shown to exhibit gender bias (Jentzsch et al. 2022), ethnic bias (Ahn et al. 2021) and general attitudes influenced disproportionally by western value-systems (Atari et al. 2023). They also generally perform better in some languages than others, and fixing that requires additional fine-tuning, which is often limited by data availability (Etxaniz et al. 2024). Knowing the internal workings of the models may provide alternative avenues to alleviate such issues directly.

[1]The code can be found on our GitHub

[2]For example, the largest of Meta's LLama 3 models was on trained on more than 15 trillion tokens (Dubey et al. 2024)

The research field of mechanistic interpretability seeks to develop methods to analyze the internals of machine learning models, to hopefully be able to reverse-engineer them (Olah 2022). One of the methods developed, Sparse Autoencoders (SAEs), is used to "disentangle" internal representations in models, so we can hone in on discrete features that can be understood.

We will attempt to use SAEs to find representations of Danish language in a range of differently sized language models from the Pythia suit (Biderman et al. 2023) and if successful, test how modifying them can change the model's performance. We hope that we can thereby improve the model's capabilities in Danish.

The plan for our investigations can be summarized as:

- Use probing to identify which layer in which model contains has largest degree of separation between Danish and English data.
- Gridsearch on a smaller SAE in order to perform hyperparameter tuning.
- Train a SAE on the chosen model and layer, with the chosen hyperparameters, in order to retrieve Danish neurons.
- Perform ablation/amplification of the neurons.
- Conduct downstream tasks before and after treatment in order to see effect.

2 Background

An interesting feature of large language models is how well they perform cross-linguistically. Granted, some models are trained explicitly for multilingual use (Le Scao et al. 2023; Lin et al. 2022), but even allegedly monolingual LLMs often achieve remarkable performance in non-english contexts (Armengol-Estapé et al. 2021). Part of the explanation is of course, that the datasets used to train English language models also include other languages (e.g the dataset used for training Llama 3 includes 8% non-english tokens (Dubey et al. 2024)), but in addition, it is possible that some neurons are shared between languages. That is, upon an input inquiring about (for example) a topic in mathematics, many of the model's internal activations may be similar, no matter if the question is asked in English, German or Arabic. After all, if the model has learned patterns useful for solving math problems from English data, leveraging those patterns when encountering similar inputs in other languages would improve performance.

There are some preliminary results suggesting such an explanation. Wendler et al. (2024) investigated internal activations of the Llama-2 models when given a translation task and found that the hidden activations go through distinct "stages" during a forward pass, from hard-to-interpret latent representations orthogonal to all languages, to "concepts", which lie closest to English tokens and finally to activations aligned with the tokens in the desired language. Tentatively, they found the models internal representations to consist of non-linguistic concepts, which nonetheless lie closest to English, which are then put into to the correct language for the output.

This hypothesis raises the possibility of finding "context neurons", neurons that are active when processing input in particular languages (or indeed, other categories of text, such as programming languages), which could then inform the model's final translation. This may have already been successfully achieved: Gurnee et al. (2023) introduced the method of *sparse probing* (more on this in section 4.1) and used it on EleutherAI's Pythia models (Biderman et al. 2023), to find individual neurons strongly associated with text in French, Go code and US Patent

Office documents, respectively. Building upon their methodology, Quirke et al. (2023) found German context neurons in the pythia-70m model, studied how they developed during training and demonstrated a decrease in model performance when they were ablated.

While this approach of finding meaningful neurons has the potential to significantly improve the interpretability of language models, it is unfortunately not always so simple. The above method works for finding distinct neurons that represent a feature, such as a text being in German. There is, however, reasons to doubt that features are usually represented so straightforwardly. The following sections explain the problems of *polysemanticity* and *superposition* in neural networks and a potential solution for interpretability: Sparse Autoencoders.

2.1 Superposition

To understand superposition and why it presents issues, it is useful and take a step back and ask how neural networks even store information. We generally think of the information in neural networks to consist of directions in activation space, which correspond to *features*. Exactly how to define "feature" is not all that simple, but in this paper, we will use it to mean a characteristic in the data, that the model stores, which is interpretable to a human. A classic example comes from Mikolov (2013), who introduced *word2vec*, a technique for embedding words as vectors, and found that these word embeddings, despite being trained in an unsupervised manner, had captured a large amount of information that goes beyond syntactic regularities. For example, he identified that vectors for "king" and "queen" were related in such a manner that `vector("king") - vector("man") + vector("woman") = vector("queen")`. In this example `vector("man")` and `vector("woman")` comprise general directions in embedding space corresponding to masculinity and femininity, respectively.

The task of finding interpretable neurons can now be reformulated as finding interpretable *directions* in activation space. These directions may align with the neurons, such that one neuron corresponds to one feature (such neurons are called *monosemantic*, see figure 1, left), or they may be misaligned such that each direction is a linear combination of neurons (here, the neurons would be *polysemantic*). For interpretability, monosemantic neurons are of course desirable, but since each direction in activation space is in principle equivalent, a linear combination of neurons, corresponding to a distinct direction in activation space, could be just as interpretable, given that each feature direction is orthogonal.

Elhage et al. (2022) trained small neural networks and investigated their hidden activations, finding that not only were the neurons polysemantic, they also represented more distinct features than there were neurons! Figure 1 (right) illustrates this situation, where the number of features exceeds the number of neurons, which the authors call *superposition*. The obvious disadvantage of this, from the model's perspective, is that each feature direction is not orthogonal, causing interference between them. In figure 1 (right), this can be seen in that increasing the value of the cyan feature would inescapably also increase the value of the purple feature. The authors find that two preconditions are necessary for this to be a worthwhile trade-off for the model:

- The model has to have a non-linear activation function.
- The data has to have sparse features, meaning that only a small share of features will be present for any given datapoint.

They hypothesize that it is less problematic for two *sparse* features to interfere with each other since they will co-occur infrequently. In a well-known example, a single neuron in the vision model InceptionV1 responded to both cat faces and the fronts of cars, specifically the whiskers of the cat and the shiny front of the car (Olah et al. 2020). These two features should

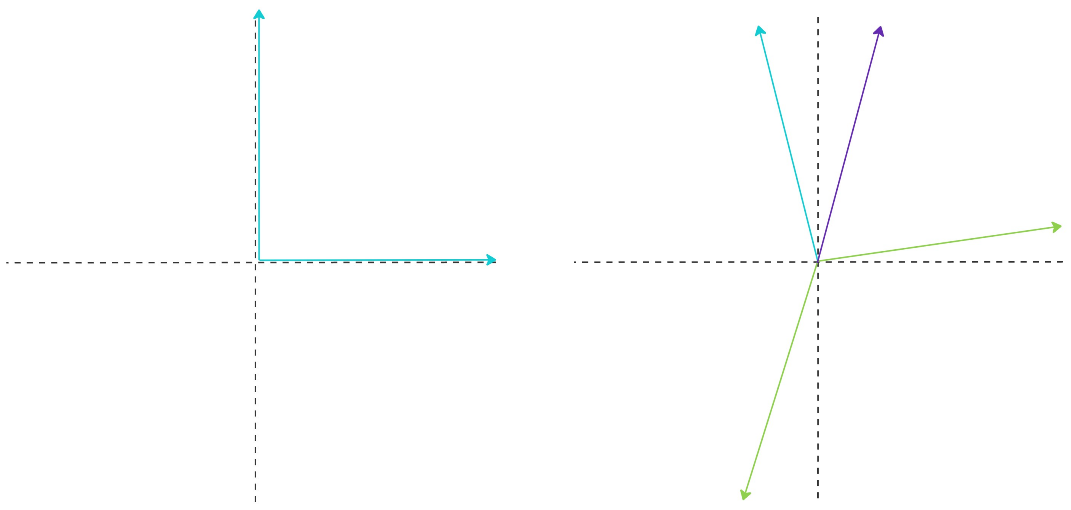


Figure 1: The axis of each plot represents values for a given neuron, meaning if you move up and down y-axis, then you change the value of the neuron corresponding to said axis. The arrows illustrates hypothetical features, with the left figure showing no superposition and perfectly interpretable features, and the right figure having multiple features along vertical axis, indicating that the neuron imposes superposition

mostly not appear together and therefore it is fine, in terms of performance, for the network to impose superposition.

The issue of superposition gets amplified in larger models due to a non-intuitive feature of high-dimensional space. It turns out that, while the number of orthogonal vectors that can be represented in a n-dimensional space grows linearly with n, the number of *almost orthogonal* vectors grows exponentially (Elhage et al. 2022). For our case, this means that inside an LLM, the features represented do not need to be neatly arranged in such a way that neurons correspond to features, with around the same number of features as neurons. Instead, there is a vastly higher number of features, each represented by a subtly different combination of neurons. Interpreting them then requires disentangling the huge web of interconnected features. *Sparse Autoencoders* have emerged as a promising method for doing just that.

2.2 Sparse Autoencoders

Elhage et al. (2022) suggest a novel way of thinking of models exhibiting superposition. Here the model with n neurons is actually performing a lossy compression of a hypothetical larger model with $m > n$ monosemantic neurons. The larger model, if it existed, would then be straightforwardly interpretable, with one neuron corresponding to one feature. Sparse Autoencoders (SAEs) can be seen as a way of reconstructing this larger model. Unlike regular autoencoders (figure 2, right), which compress the data in a large vector into a lower dimensionality, an SAE (figure 2, left) expands the dimensionality of the vector in its hidden layer. However, this would be trivial if it were not for the sparseness constraint, which ensures that only a limited amount of neurons in the hidden layer will be active for a given input.

Much success has been had recently in using SAE to find interpretable features in both small toy models and large production models. Bricken et al. (2023) analyzed a small 1-layer

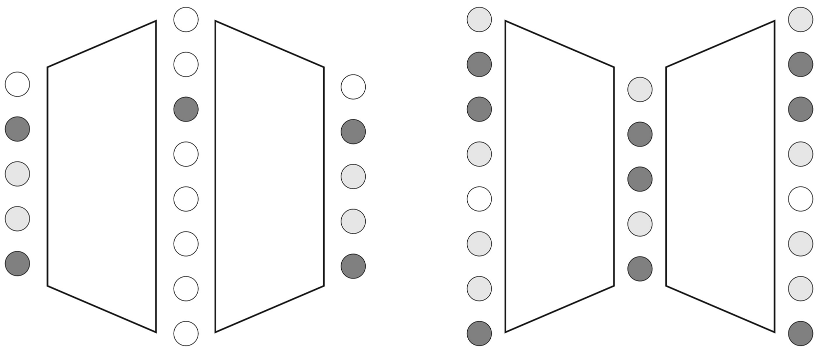


Figure 2: Left is a sparse autoencoder, right is a regular autoencoder

transformer model and identified features associated with (among other things) Arabic text and DNA-sequences. The same team later scaled up their approach to analyze the LLM Claude Sonnet and found features related to brain science, transit infrastructure, and the Golden Gate Bridge (Templeton et al. 2024). Notably, the features seemed to be causally relevant to the model's output, which the authors demonstrated with a modified version of their model, which would lead any conversation towards the topic of the Golden Gate Bridge. Similarly, Cunningham et al. (2023) used an automated approach to evaluate features identified from a small open-source language model (from the Pythia suite of models released by Biderman et al. (2023)). They found that SAE's identify features significantly more interpretable than singular neurons.

3 Models and Data

We chose to focus on the Pythia models (Biderman et al. 2023), a suite of LLMs specifically trained for research purposes. It includes multiple different sizes, ranging from 14 million parameters to 12 billion parameters. They were trained on The Pile (Gao et al. 2020), a mostly English language dataset and achieve performance comparable to GPT-3 13B on the LAMBADA benchmark (Papers with Code 2024). Because of computational constraints, we restrict ourselves to only look at a subset of all the Pythia models. The full model list can be seen in figure 3. The model architecture consists of multiple stacking blocks called GPTNeoXLayer that contain multihead attention followed by a Multi Layer Perceptron (MLP), with the smallest pythia-14m having 6 blocks and the largest pythia-12b having 36 (Biderman et al. 2023). Since Gurnee et al. (2023) had success in finding context neurons in the middle layer of the MLPs, we will also focus on those, ignoring the multi head attention[3].

3.1 Parallel Data

For training our sparse autoencoder and later for identifying Danish neurons, we use parallel Danish-English datasets. We reason that for the SAE to be incentivized to encode the presence of Danish language in a neuron, it should be trained on a dataset where Danish language is prominent. Since parallel data includes 50% Danish sentences, this is suitable for our purposes. For identifying neurons that are active more often in Danish contexts than English, parallel

[3]From here on, when we refer to layer i of the model, we are in particular talking about the MLP in layer i, not the attention.

data, which includes the same sentences in Danish and English, is also suitable, since it isolates language as the only difference between the Danish and English data.

3.1.1 NeuLab-TedTalks

This dataset is a collection of human-translated TED-talks into more than 100 different languages (Tiedemann 2012), with the Danish section containing 48.5k sentences. It was used to train both our probes (section 4.1) and our final SAE. Additionally, it was used for identifying Danish neurons. As for the split, we use 70% of the total dataset for training and the rest for testing.

3.1.2 ELRC_2923

Ideally, if we have identified monosemantic Danish neurons in our SAE, it should also neatly divide Danish and English text in different datasets, with different content. We therefore also analyze their activations with ELRC_2923, a small dataset comprising parallel Danish and English translations of statements about the European Union's response to the COVID-19 pandemic.

3.2 Data for Downstream Tasks

We want to measure the effect of the Danish neurons on the model, so we perform an experiment, in which we vary the activation of the neurons, and measure the impact on performance in downstream tasks. The downstream tasks used are a Danish sentiment analysis task, using **angry-tweets**, a Danish language acceptability task using **ScaLa**, a subset of the Scandeval evaluation suite (Nielsen et al. 2024) and finally **Tweet Sentiment Extraction** an English sentiment analysis dataset.

3.2.1 angry-tweets

The dataset consists of anonymized Danish tweets, each labeled with a sentiment category: positive, neutral, or negative. The dataset has a total of 3.484 tweets divided into training and test splits, with the test set making up 20%. The label distribution is as follows: 37% negative, 35% neutral, and 28% positive (Pauli et al. 2021).

3.2.2 ScaLA

The task for this dataset is to predict whether or not a sentence is grammatically correct. The data has sentences which are labeled incorrect was automatically corrupted using a ruleset[4] created by the authors of Scandeval. The dataset has a total of 5352 sentences with the test set having 20% of the total rows and a perfect 50/50 correct and incorrect label distribution.

3.2.3 Tweet Sentiment Extraction

Like angry-tweets, this dataset is a Twitter sentiment analysis dataset, consisting of 27.5k English tweets and a corresponding sentiment label with the following label distribution: 40.5% of the data is neutral, 31.5% is positive and 28% is negative (Maggie et al. 2020). We used a 20% test split and a subset of the full data due to compute limit.

[4] GitHub containing the ScaLa creation script

4 Preliminary Experiments

A notable downside of sparse autoencoders is how computationally expensive they can be to train. This is due to the simple fact that their parameter count scales quadratically with the base model's hidden size and linearly with the expansion factor. For this reason, we perform some preliminary analysis to find the optimal model, location and hyperparameters, which would be most likely to yield interpretable Danish language features.

4.1 Probing

Since we are considering 7 different models, each with anywhere from 6 to 32 layers, there are a many different points on which to attach our SAE. Computational limitations make it unfeasible for us to train our SAE on all points, so a preliminary study of the general structure of the models, to find *where* the relevant linguistic information is utilized, seemed in order. Here, we chose to train much less computationally expensive *Classification Probes* (Alain 2016) on each layer of each model of interest.

Probing is a technique to identify where certain information resides within a given model. The setup uses a model (probe) that takes activations from the model you want to analyze as input and predicts the information you want to locate, which in our case is its Danish capabilities. While probing has some theoretical limitation (outlined by Belinkov (2022)), including questions about the causal properties of the information identified by them, the aforementioned success of sparse probing by Gurnee et al. (2023) and Quirke et al. (2023) in finding linguistic context neurons suggests optimism in their utility for our purposes.

4.1.1 Methodology

For each layer on each model, we train a probe $\sigma_\beta(\mathbf{x})$ on the internal activation vector $\mathbf{x} \in \mathbb{R}^n$. The probe is defined as:

$$\sigma_\beta(\mathbf{x}) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n)}}$$

where β_0 is a bias, $\beta_i x_i$ is the probe's weight and the corresponding neuron activation from the original model. The probe is then trained by minimizing the expected binary cross-entropy loss

$$\mathcal{L} = y \cdot \log(\sigma_\beta(\mathbf{x})) + (1 - y) \cdot \log(1 - \sigma_\beta(\mathbf{x}))$$

where $y \in \{0, 1\}$ is the ground truth for the language $(0 = \mathit{Danish}, 1 = \mathit{English})$

The method used by Gurnee et al. (2023) limits the probes' flexibility by imposing a sparseness constraint on its weights, requiring that at most k coefficient be active at a time. This allows them to hone in on features represented by either a single (monosemantic) at least very few neurons. Since we would rather not assume that the Danish features we are looking for are represented by (nearly) monosemantic neurons, we instead limit the flexibility of our probes by adding an L2 regularisation term to the loss function:

$$L_2 = \lambda \cdot \sqrt{\sum_{i=0}^{n} \beta_i^2}$$

where λ is hyperparameter that sets the strength of the regularization. Intuitively, for our SAE to be able to identify a direction (or very few distinct directions) in activation space encoding the presence of Danish language, the presence of Danish should be retrievable by our probe using very few distinct directions and therefore a low value of the L2 regularization term. A probe achieving a high accuracy in classifying Danish or English text using the activations from layer l, even with a high λ-value, would thereby be evidence that that layer l encodes the presence of Danish language (or at least non-English language, more on this in section 6.2) using few distinct directions in activations space, which an SAE would be able identify.

4.1.2 Results

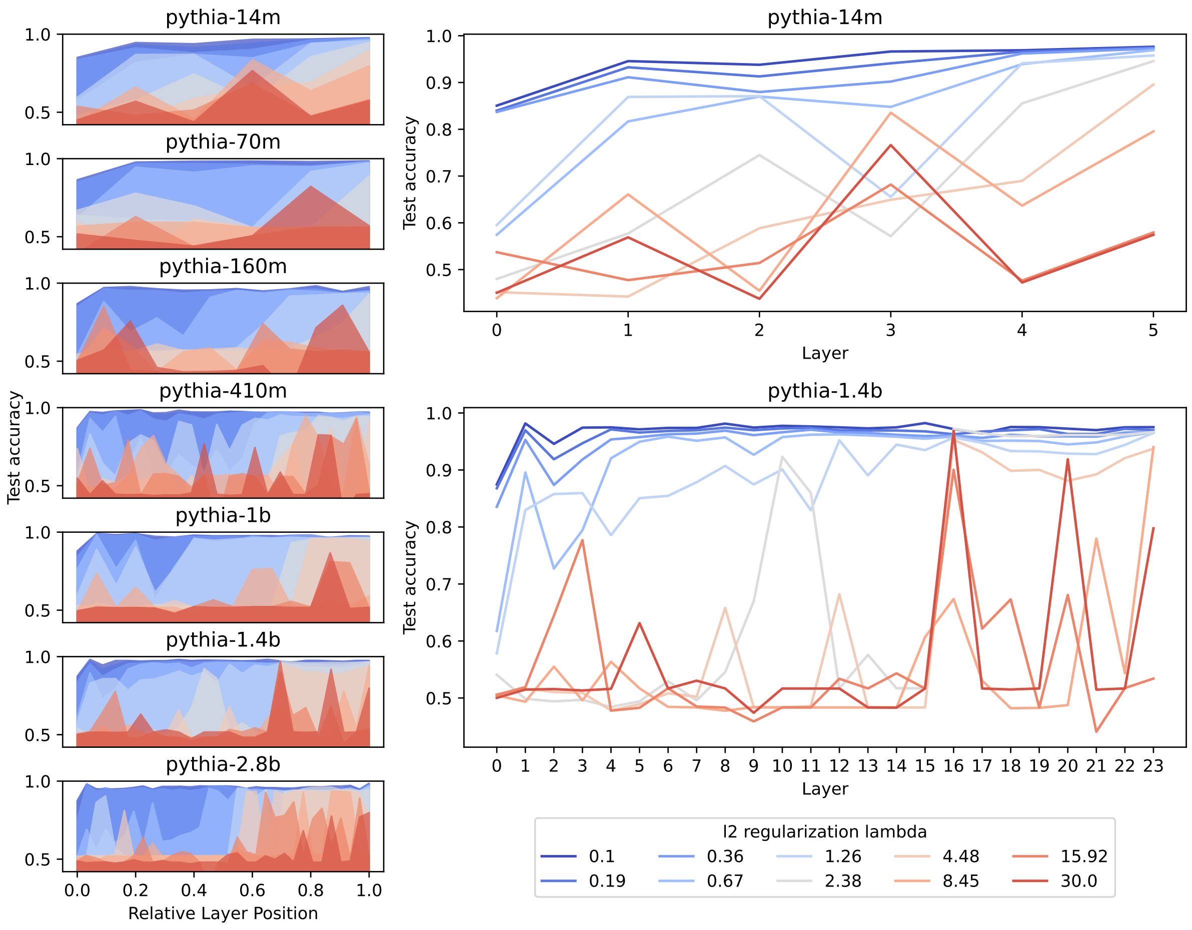


Figure 3: Test accuracy for probes trained on 7 of the Pythia models at different levels of regularization. Pythia-14m and Pythia-1.4b highlighted.

For each model and layer, we trained 10 classification probes with different λ-value. The results are shown in figure 3 for different size models, with the 14m parameter and 1.4b parameter models being highlighted. We find that with low regularization factors, the probes achieve high accuracy on all layers, but as we increase the λ-value, it drops starkly in the early layers, but stays high for longer in the later layers. On the larger models, we observe a notable "phase transition" somewhere between the middle layer and three fourths into the model. This seems to fit somewhat with the work of Wendler et al. (2024), who find that representations that can be straightforwardly interpreted linguistically appear quite suddenly around halfway through the model.

Qualitatively, the most noticeable single layer rise occurs on layer 16 of pythia-1.4b, which achieves upwards of 90% accuracy even with high regularization factors, notably higher than the layers immediately before *and* after (see figure 3, bottom right). We therefore chose to focus on training our SAE on the activations from this layer.

4.2 SAE Hyperparameter Tuning

In addition to which layer to investigate in the original model, there are also many options regarding how to set up the SAE. We look primarily to the implementations of Bricken et al. (2023), Cunningham et al. (2023) and Gao et al. (2024) for inspiration, but between those, there are many design differences in their respective approaches.

Common to all implementations is the general structure of a sparse autoencoder, consisting of an encoding weight matrix $W_e \in \mathbb{R}^{m\times n}$, a decoding weight matrix $W_d \in \mathbb{R}^{n\times m}$, an encoding bias $\mathbf{b}_e \in \mathbb{R}^m$ and a non linear activation function ϕ. The SAE is then trained by minimizing the expected mean square error (MSE) reconstruction loss

$$\mathcal{L} = ||\mathbf{x} - \hat{\mathbf{x}}||_2^2, \quad \hat{\mathbf{x}} = W_d\phi(W_e\mathbf{x} + \mathbf{b}_e)$$

The main difference between the three implementations is how they go about ensuring sparsity in the hidden layer. Gao et al. (2024) do it directly, by using a *topk* activation function, which, similar to the sparse probing in Gurnee et al. (2023), restricts the latent representation to only the k highest activations, setting the rest to zero. Cunningham et al. (2023) and Bricken et al. (2023) use a more standard RELU activation function, but adds an L1 regularization term on the latent activations to the loss function.

The number of bias terms differ between them: Cunningham et al. (2023) use just an encoder bias as described above, while Bricken et al. (2023) and Gao et al. (2024) use an additional pre-encoder bias $\mathbf{b}_p \in \mathbb{R}^n$, which they subtract before encoding and then re-add after decoding:

$$\hat{\mathbf{x}} = W_d\phi(W_e(\mathbf{x} - \mathbf{b_p}) + \mathbf{b}_e) + \mathbf{b}_p$$

Finally, Cunningham et al. (2023) use the same weight matrix for encoding and decoding ($W_d = W_e^T$), while Bricken et al. (2023) and Gao et al. (2024) use two independent matrices.

They all give different arguments for their choices, ranging from the theoretical to the anecdotal to actual systematic tests. The most principled approach in this case was to perform a gridsearch over the various options to identify the one that performed best, see the appendix A for the full search space.

4.2.1 Methodology

The first issue with comparing different model configurations is the metric with which to compare them. It is of course possible for any of the model configurations to achieve perfect reconstruction accuracy, given that we relax the sparseness constraints. We therefore need to consider both the sparsity, i.e. how few neurons are active for each token input, and the reconstruction loss.

If not for the sparseness constraint, Gao et al. (2024) found the neurons in the hidden layer would become dense, meaning a given neuron is active for a larger proportion of token inputs, making it harder to identify feature-specific neurons. However, selecting the model with the lowest density alone is not sufficient, as exclusively minimizing density can lead to solutions where the overall number of “alive” neurons is drastically reduced (Gao et al. 2024), effectively removing the sparse component of the SAE and reducing it to a regular autoencoder, akin to the model in (figure 2, right). Alive neurons should therefore also be a desirable metric.

Secondly, the different approaches to ensure sparsity also present an issue: with the *topk* activation function, sparsity is directly controlled through a hyperparameter, while in the regularization-based approach, the hyperparameter only indirectly ensures sparsity. We therefore need to measure the sparsity of the RELU based models at different levels of regularization, to find a fair level for comparison.

We perform a gridsearch of different hyperparameters using a SAE with expansion factor 4 on layer 3 of the smallest Pythia model (pythia-14m). We found evidence of this layer, similar to layer 16 of pythia-1.4b, encoding Danish language more-so than surrounding layers (see figure 3). For each combination of

- pre-encoder bias *or* no preencoder bias,
- $W_d = W_e^T$ *or* $W_d \neq W_e^T$,
- *topk* activation function *or* RELU + L1-regularization,
- 6 different k-values/L1-λs,

we train for one epoch on the text from the NeuLab-TedTalks dataset (see section 3.1.1).

For each combination, we measure the MSE loss, sparsity (The number of non-zero neurons per token. For the topk models, this is equal to k) and active neurons overall (the number of non-zero neurons per batch).

4.2.2 Results

Since MSE loss and sparsity trade-off against eachother, it would hopefully be possible to determine a pareto frontier, finding an optimal hyperparameter configuration for each value. This is what we find in our gridsearch (see left plot in figure 4). We find the largest discrepancies between the activation functions, namely topK, and RELU with topK consistently having a more sparse latent space at each fixed MSE score. Unlike what Bricken et al. (2023) found, we do not see a clear performance difference from using pre-encoder bias, and keeping an independent encoder-decoder matrix shows no clear impact either.

As noted in 2.2, it is crucial to check for dead neurons since we want the encoder to maintain the sparseness property. After all, maybe the topk SAEs simply use fewer neurons overall, which would be undesirable for an SAE. (Figure 4, right) illustrates the active neurons per batch across various MSE values. Here we see that in fact, the topK SAEs for the most part have more active neurons per batch than the RELU SAEs.

Because of these findings, we chose to use a topK activation function, but since we did not see a large difference either way in the other two parameters, we chose to follow the recommendations of Bricken et al. (2023) and Gao et al. (2024) and use pre-encoder bias and separate weight matrices for encoder and decoder. This info and additional training parameters can be seen in table 4.

5 Main Experiments

We train an SAE on layer 16 of pythia-1.4b, using the hyperparameters described in section 4.2.2. Because of high computational costs, we had to limit ourselves to an expansion factor of 8, meaning we have 8192 input neurons and 65536 hidden neurons. We set $k = 32$, which seems quite small, but Gao et al. (2024) report good results from an SAE with 131072 hidden neurons

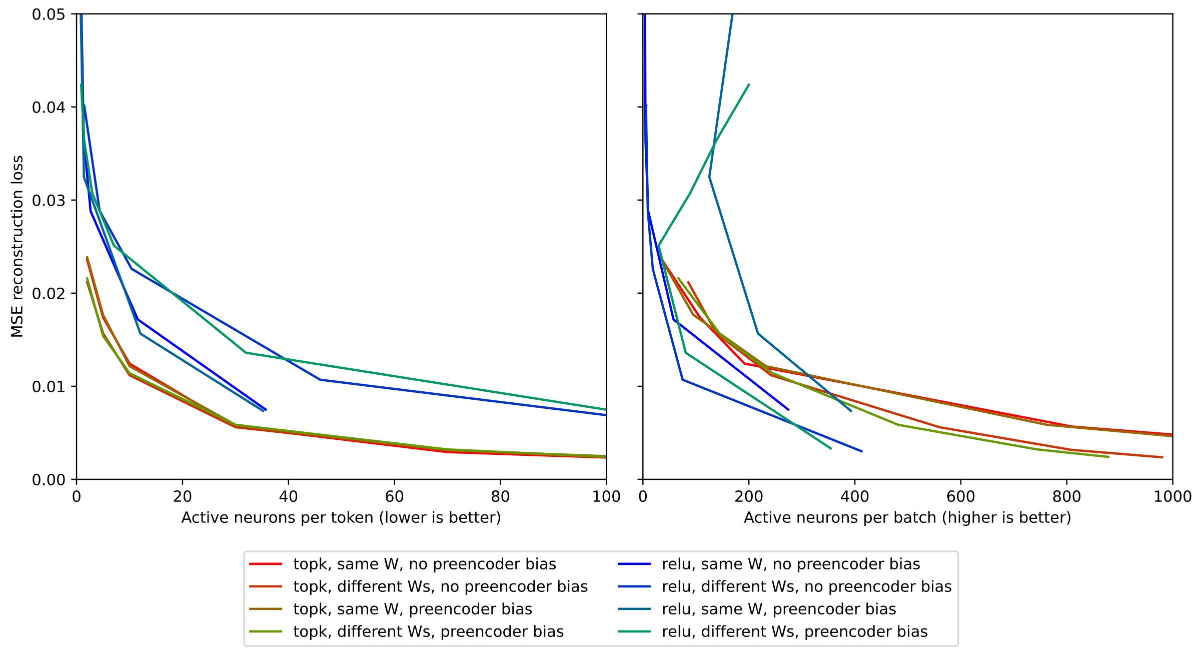


Figure 4: Gridsearch results for different model architectures displaying the trade off between MSE and active neurons, with the left plot showing active neurons per token and the right plot showing per batch. Each line shows the loss and active neurons, as we vary the k and λ-values, respectively (see appendix A.)

and $k = 32$. We train on the NeuLab-TedTalks dataset described in section 3.1.1 using an A100 on Google Colab.

5.1 Qualitative analysis of language specific neuron

To identify Danish-specific neurons, we seek to find neurons that are active (that is, are among the k neurons with the highest activation) on a large amount of Danish tokens, but a small amount of English ones. On the test set of NeuLab-TedTalks, we identify 6 neurons, which are active on more than 50% of tokens in the Danish sentences, but less than 1% of tokens in the English sentences. We plot their pre-activations (their values before the topK activation function) in figure 5. Here we see that for all of them, their pre-activations are (expectedly) significantly higher on Danish tokens than English. In particular, neuron 26335 shows remarkably little overlap between the two languages (figure 5). For English tokens the pre-activations all center around 0 whereas the values for Danish are mostly distributed between 1-5, showing neuron 26335 exhibits a clear distinction between Danish and English tokens

We wanted to illustrate how the neuron behaved on sample Danish and English text, so we computed its activation and pre-activation on Danish and English parallel text from ELRC_2923 (see section 3.1.2). In figure 6 (top) we see that neuron 26335 is highly active post activation function and in figure 6 (bottom) the neuron is completely absent, meaning at no point is it in the topk highest activations.

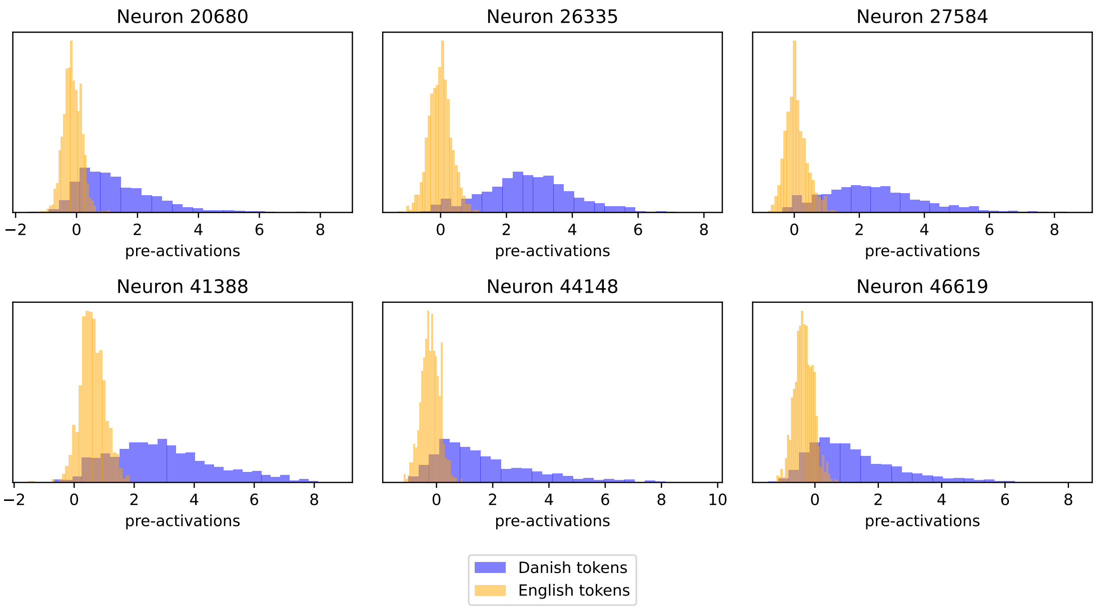
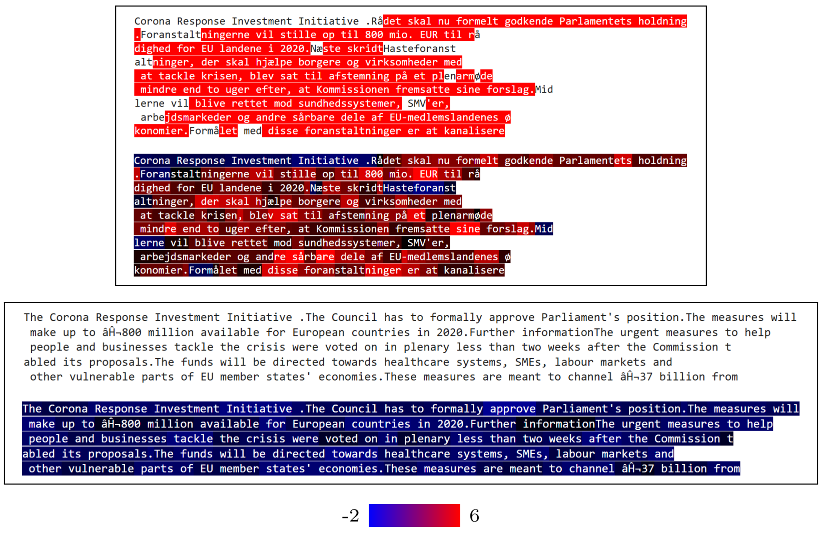


Figure 5: Histogram of top neurons' pre-activations for Danish and English tokens.



```
Corona Response Investment Initiative .Rådet skal nu formelt godkende Parlamentets holdning
.Foranstaltningerne vil stille op til 800 mio. EUR til rå
dighed for EU landene i 2020.Næste skridtHasteforanst
altninger, der skal hjælpe borgere og virksomheder med
 at tackle krisen, blev sat til afstemning på et plenarmøde
 mindre end to uger efter, at Kommissionen fremsatte sine forslag.Mid
lerne vil blive rettet mod sundhedssystemer, SMV'er,
 arbejdsmarkeder og andre sårbare dele af EU-medlemslandenes ø
konomier.Formålet med disse foranstaltninger er at kanalisere
```

```
Corona Response Investment Initiative .Rådet skal nu formelt godkende Parlamentets holdning
.Foranstaltningerne vil stille op til 800 mio. EUR til rå
dighed for EU landene i 2020.Næste skridtHasteforanst
altninger, der skal hjælpe borgere og virksomheder med
 at tackle krisen, blev sat til afstemning på et plenarmøde
 mindre end to uger efter, at Kommissionen fremsatte sine forslag.Mid
lerne vil blive rettet mod sundhedssystemer, SMV'er,
 arbejdsmarkeder og andre sårbare dele af EU-medlemslandenes ø
konomier.Formålet med disse foranstaltninger er at kanalisere
```

```
The Corona Response Investment Initiative .The Council has to formally approve Parliament's position.The measures will
 make up to âĤ¬800 million available for European countries in 2020.Further informationThe urgent measures to help
 people and businesses tackle the crisis were voted on in plenary less than two weeks after the Commission t
abled its proposals.The funds will be directed towards healthcare systems, SMEs, labour markets and
 other vulnerable parts of EU member states' economies.These measures are meant to channel âĤ¬37 billion from
```

```
The Corona Response Investment Initiative .The Council has to formally approve Parliament's position.The measures will
 make up to âĤ¬800 million available for European countries in 2020.Further informationThe urgent measures to help
 people and businesses tackle the crisis were voted on in plenary less than two weeks after the Commission t
abled its proposals.The funds will be directed towards healthcare systems, SMEs, labour markets and
 other vulnerable parts of EU member states' economies.These measures are meant to channel âĤ¬37 billion from
```

Figure 6: The top figure shows which Danish tokens have neuron 26335 included in the topk activations, meaning tokens in non-red has not made the neuron fire. Underneath is displayed the pre-activation values across different tokens for the neuron. The bottom figure shows the same, but for English sentences

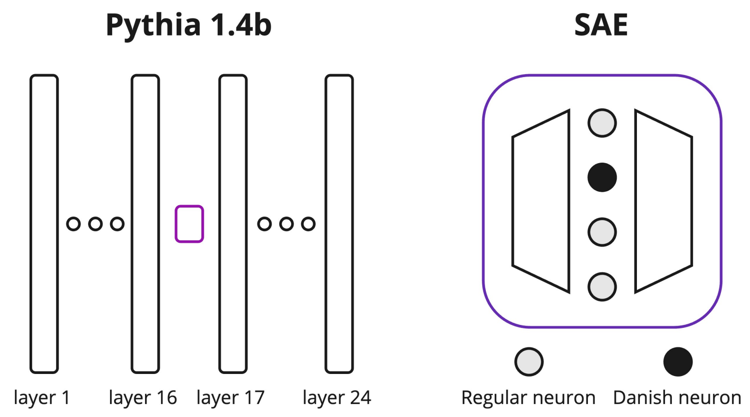


Figure 7: Experiment setup with the SAE being inserted between layer 16 and 17, and an identified Danish neuron being highlighted

5.2 Ablation studies

To test the effect of the Danish neurons on the model, we measure the model's performance as we vary their activations. This is done by inserting our trained SAE on layer 16 of the model, replacing the output with our reconstruction (see figure 7). If the SAE has a sufficiently low reconstruction error, then the language model should in principle act as if the SAE is not present at all since the input and output of the SAE would be close to the same. Now, we can isolate the effect of the Danish neurons by manually varying their activations in the SAE.

First, we measure the cross-entropy loss for next token prediction on the Danish and English parts of the parallel ELRC_2923 dataset, respectively, as we fix the activation of the 6 neurons at different levels (figure 8). We find that the optimal fixed value for the neurons lies close to their mean pre-activation value (as seen in figure 5), which is to be expected, but that lowering the value has an outsized negative influence on performance on Danish text in particular, compared with English text. Notably, the loss is slightly lowered by fixing the value at around 4 compared to remaining unmodified (see Appendix C).

Secondly, we fine-tune the model to three different downstream tasks with different fixed values for the Danish language neurons:

- A Danish language sentiment analysis task using twitter data.
- A Danish language acceptability task.
- An English language sentiment analysis task.

Looking at figure 9 we see that the model performs best on the two sentiment analysis tasks, with the English version being overall slightly better, and significantly worse on the grammatical task ScaLA. When the neurons are fixed at -8 we see a much larger decrease in performance on the Danish tasks compared to the English task. This finding along with the results in figure 8 could indicate we have identified a set of neurons that is relevant to performance in Danish in particular. For a table overview see appendix D.

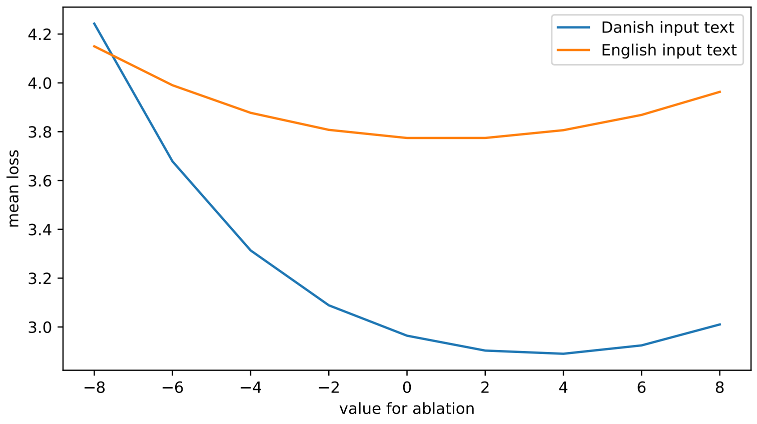


Figure 8: Cross-Entropy next token prediction loss for different fixed values of the Danish neurons.

6 Discussion and Limitations

6.1 Subpar Performance on Downstream Tasks

While the results in figure 9 show a notable difference on account of the intervention, the downstream performance of the model in all configurations are not great. Particularly in the ScaLA-task, which is binary, and has evenly distributed labels, one would expect higher scores than 50% for a fine-tuned model.

Overall, the reason could be due to the dataset used to train the Pythia suite being primarily an English dataset (Gao et al. 2020) and as such, there may not even be a good representation for the Danish language in the last hidden layer. As seen in table 1 the Danish output of the model is not of high quality, with the English output 2 being noticeably better. This would empirically indicate that the model is better at English than Danish.

6.2 Are the Neurons Specific to Danish?

Additionally, one might worry that the clear separation between languages (as seen in figure 5) is not actually a distinction between Danish and English but between English and not English. For example, upon inspecting the tokens used in the model, it becomes apparent that Danish words are composed of significantly more subword-tokens compared to English. This could constitute a feature that is easy for the model to optimize for, which is nevertheless not specific to Danish. An additional language could have been used to test this hypothesis.

6.3 Compute limits

Bricken et al. (2023) recommends a scaling factor in the range [8;256], but we were limited by the hardware available. With an expansion factor of 8, we were forced to use the largest available GPU on Google Colab, and if we wanted to increase the expansion factor to more than the lower bound of what (Bricken et al. 2023) recommends, a more sophisticated solution would probably be necessary.

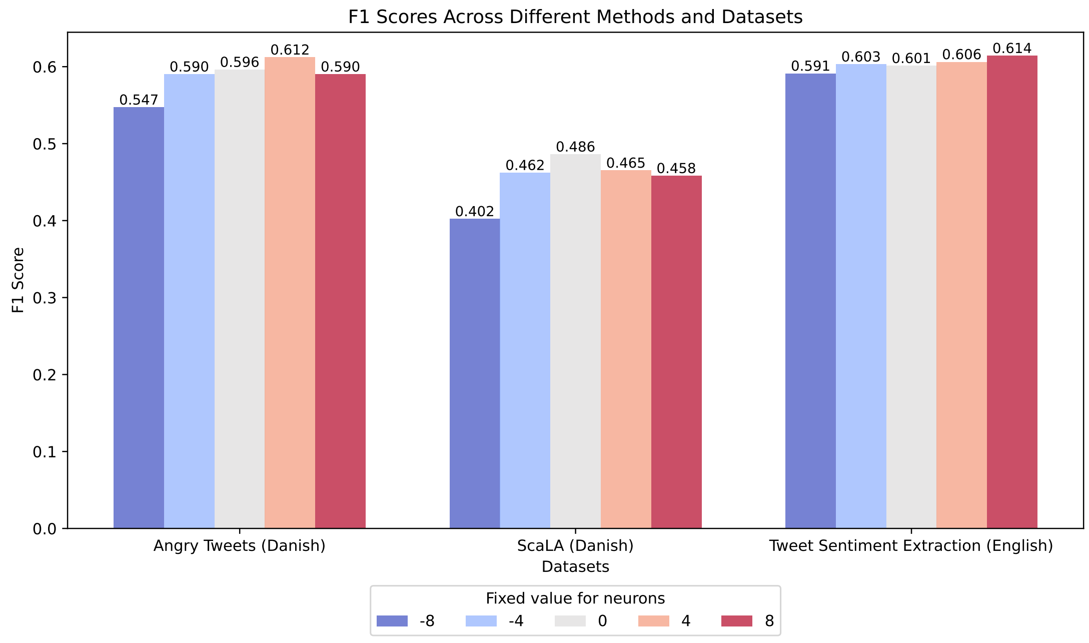


Figure 9: Results for three different downstream tasks as we vary the activation of the 6 Danish neurons.

Prompt	Response
En gang for længe siden var der	en del, der var blevet udsat
Da han var ude og fiske så han	var ute og fiske.
Hver torsdag spiser hun	en kop kaffe.
Zooen har fået en ny	kandidat, og det er en af de mest

Table 1: EleutherAI's **pythia-1.4b** prompted with Danish sentences

6.4 Limits of our SAE Approach

Templeton et al. (2024) suggest a limitation to the approach of using MLP activations as inputs to the SAE. They believe many features in LLMs exhibit *cross-layer* superposition, a phenomenon where features are represented across layers. They propose to instead use residual stream activations in the SAE as a way to remedy this, since the residual stream gathers the outputs of all preceding blocks. This would also enable a larger expansion factor, due to the residual stream being smaller.

Prompt	Response
Once upon a time there was a	man who lived in a house. He had a wife and two
When he went fishing he saw	a big fish, and he caught it. He took it home and
Every thursday she eats	a big breakfast and then she goes to the gym. She is a very
The zoo got a new	name, and the new name was "The Zoo of the Future."

Table 2: EleutherAI's **pythia-1.4b** prompted with English sentences

References

Tiedemann, Jörg (2012). "Parallel data, tools and interfaces in OPUS." In: *Lrec*. Vol. 2012. Citeseer, pp. 2214–2218.

Mikolov, Tomas (2013). "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* 3781.

Alain, Guillaume (2016). "Understanding intermediate layers using linear classifier probes". In: *arXiv preprint arXiv:1610.01644*.

Gao, Leo et al. (2020). "The Pile: An 800GB Dataset of Diverse Text for Language Modeling". In: *arXiv preprint arXiv:2101.00027*.

Maggie, Phil Culliton, and Wei Chen (2020). *Tweet Sentiment Extraction*. https://kaggle.com/competitions/tweet-sentiment-extraction. Kaggle.

Olah, Chris et al. (2020). "Zoom In: An Introduction to Circuits". In: *Distill*. https://distill.pub/2020/circuits/zoom-in. DOI: 10.23915/distill.00024.001.

Ahn, Jaimeen and Alice Oh (Nov. 2021). "Mitigating Language-Dependent Ethnic Bias in BERT". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Ed. by Marie-Francine Moens et al. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 533–549. DOI: 10.18653/v1/2021.emnlp-main.42. URL: https://aclanthology.org/2021.emnlp-main.42.

Armengol-Estapé, Jordi, Ona de Gibert Bonet, and Maite Melero (2021). "On the multilingual capabilities of very large-scale English language models". In: *arXiv preprint arXiv:2108.13349*.

Pauli, Amalie Brogaard et al. (May 2021). "DaNLP: An open-source toolkit for Danish Natural Language Processing". In: *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*. Ed. by Simon Dobnik and Lilja Øvrelid. Reykjavik, Iceland (Online): Linköping University Electronic Press, Sweden, pp. 460–466. URL: https://aclanthology.org/2021.nodalida-main.53.

Belinkov, Yonatan (2022). "Probing classifiers: Promises, shortcomings, and advances". In: *Computational Linguistics* 48.1, pp. 207–219.

Elhage, Nelson et al. (2022). "Toy Models of Superposition". In: *Transformer Circuits Thread*. https://transformer-circuits.pub/2022/toy$_{m}odel/index.html$.

Jentzsch, Sophie and Cigdem Turan (July 2022). "Gender Bias in BERT - Measuring and Analysing Biases through Sentiment Rating in a Realistic Downstream Classification Task". In: *Proceedings of the 4th Workshop on Gender Bias in Natural Language Processing (GeBNLP)*. Ed. by Christian Hardmeier et al. Seattle, Washington: Association for Computational Linguistics, pp. 184–199. DOI: 10.18653/v1/2022.gebnlp-1.20. URL: https://aclanthology.org/2022.gebnlp-1.20.

Lin, Xi Victoria et al. (2022). "Few-shot learning with multilingual generative language models". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 9019–9052.

Olah, Chris (2022). "Mechanistic Interpretability, Variables, and the Importance of Interpretable Bases". In: *Transformer Circuits Thread*. URL: https://www.transformer-circuits.pub/2022/mech-interp-essay.

Atari, Mohammad et al. (2023). “Which humans?” In.

Biderman, Stella et al. (2023). “Pythia: A suite for analyzing large language models across training and scaling”. In: *International Conference on Machine Learning*. PMLR, pp. 2397–2430.

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.

Cunningham, Hoagy et al. (2023). “Sparse autoencoders find highly interpretable features in language models”. In: *arXiv preprint arXiv:2309.08600*.

Gurnee, Wes et al. (2023). “Finding neurons in a haystack: Case studies with sparse probing”. In: *arXiv preprint arXiv:2305.01610*.

Le Scao, Teven et al. (2023). “Bloom: A 176b-parameter open-access multilingual language model”. In.

Qin, Chengwei et al. (2023). “Is ChatGPT a General-Purpose Natural Language Processing Task Solver?” In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1339–1384.

Quirke, Lucia et al. (2023). “Training dynamics of contextual n-grams in language models”. In: *arXiv preprint arXiv:2311.00863*.

Yuan, Zheng et al. (2023). “Scaling relationship on learning mathematical reasoning with large language models”. In: *arXiv preprint arXiv:2308.01825*.

Dubey, Abhimanyu et al. (2024). “The llama 3 herd of models”. In: *arXiv preprint arXiv:2407.21783*.

Etxaniz, Julen et al. (2024). “Latxa: An open language model and evaluation suite for Basque”. In: *arXiv preprint arXiv:2403.20266*.

Gao, Leo et al. (2024). “Scaling and evaluating sparse autoencoders”. In: *arXiv preprint arXiv:2406.04093*.

Hadi, Muhammad Usman et al. (2024). “Large language models: a comprehensive survey of its applications, challenges, limitations, and future prospects”. In: *Authorea Preprints*.

Nielsen, Dan Saattrup, Kenneth Enevoldsen, and Peter Schneider-Kamp (2024). “Encoder vs Decoder: Comparative Analysis of Encoder and Decoder Language Models on Multilingual NLU Tasks”. In: *arXiv preprint arXiv:2406.13469*.

Papers with Code (2024). *Language Modelling on LAMBADA — State-of-the-Art Performance*. `https://paperswithcode.com/sota/language-modelling-on-lambada`. Accessed: 2024-12-09.

Templeton, Adly et al. (2024). “Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet”. In: *Transformer Circuits Thread*. URL: `https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html`.

Wendler, Chris et al. (2024). “Do llamas work in english? on the latent language of multilingual transformers”. In: *arXiv preprint arXiv:2402.10588*.

Appendix

A Gridsearch table

These are the parameters searched for. Note: when we use topK no lambda value is searched, and likewise for the k-value when using RELU.

Parameter	**Value**
Pre-Encoder Bias	True/False
Same W	True/False
k	2, 5, 10, 30, 70, 100
λ	$5 \times 10^{-4}, 1 \times 10^{-3}, 2 \times 10^{-3}, 3 \times 10^{-3}, 4 \times 10^{-3}, 5 \times 10^{-3}$

Table 3: Gridsearch Parameters

B Training details

A small table of the parameters used when training the SAE.

Batch size	32 sentences
Epoch	1
Expansion factor	8
Activation function	topK
Optimizer	Adam
Independent weight matrix	False
Pre encoder bias	True

Table 4: Training details

C Next token prediction results (cross entropy loss)

Fixed value for neuron	-8	-6	-4	-2	0	2	4	6	8	No modification
Danish input text	4.24	3.68	3.31	3.09	2.96	2.90	2.89	2.92	3.01	2.94
English input text	4.15	3.99	3.88	3.81	3.77	3.77	3.81	3.87	3.96	3.77

D Downstream task results (f1 score)

Fixed value for neuron	-8	-4	0	4	8	No modification
Angry Tweets (Danish)	0.547	0.590	0.596	0.612	0.590	0.607
ScaLa (Danish)	0.402	0.462	0.486	0.465	0.458	0.470
English Tweets	0.591	0.603	0.601	0.606	0.614	0.609