Do not prune but interpret weights that did not change

Jorrit Kruthoff
School of Natural Sciences,
Institute for Advanced Study,
Princeton, 08540 New Jersey, USA
jorritkruthoff@gmail.com

Abstract

We empirically show that language models have a tiny set of weights that did not change after finetuning. Such weights, which we call *zero modes*, are essential for the model's functionality. They can be linked to various mechanisms that are ought to be learned during pretraining, such as induction heads or a neuron firing on the first sentence of an input. We make these claims concrete in one and two layer toy models trained on general code and finetuned on Python and three popular open-source pretrained-finetuned pairs of models. We find many similarities between the toy and large language models. For the LLMs, we not only show how the zero modes impact their performance on a wide variety of tasks, but also how they can affect hidden states and help select interesting (and universal) neurons.

1 Introduction

In 1989 Yann LeCun, John Denker and Sara Solla argued in [1] that parts of a neural network can be pruned without hurting performance. Since then, many techniques have been developed to expand this to the realm of large language models [2–4]. Notions such as the lottery ticket hypothesis [5] were established and put to the test in various experiments. In particular, it was shown that a small fraction of the weights could be retained without much degradation of down-stream performance. Alongside this development, it was also found that ablating certain layers has no effect because there are other parts of the model that compensate for the removal. This was dubbed the *hydra effect* [6]. Similar conclusions were also reached in [7].

Pruning models can help significantly with efficiency and optimal training strategies. On the other hand, with an eye towards interpretability [8–11], one can also invert the problem and ask whether there is a small sets of non-embedding parameters that are crucial for the behaviour of the model. For instance, parameters that are crucial for a few basic behaviours of language models, such as recognizing sentences or induction heads. Identifying such parameters is difficult as one would need a rather abstract notion of what is means for a parameter to be important or not. Various metrics/definitions in this direction have been proposed in [12–14]. Typically these metrics requires a forward pass or a single backward pass and so only give limited information about the parameters, especially when performed with a small dataset.

We instead propose to look at pairs of models, a pretrained model and its finetuned derivative. This provides a more global notion of how models move in the space of parameters and could provide more information as to what the various weights of a model are related to. Essentially, when we finetune a model, we try to get the pretrained model to respond in a specific and structured way, making sure the acquired knowledge in the pretraining stage is consistently and logically transferred to the model's output in response to a user query. Inasmuch, one would expect the finetuned model to have a set of weights in common with the pretrained model. These weights would then correspond to features not trained for while finetuning and are perhaps 'rudimentary' or 'fundamental' and are essential for some basic behaviour of the model.

1.1 Our contributions

We will show that such weights can indeed exist not only in tiny one and two-layer models, but also in large LLM models with billions of parameters. We will call weights that did not change during finetuning *zero modes*. Typically, they form a small fraction of the total number of weights and are crucial for the model's performance and behaviour. Analyzing these weights shows that they can be linked to a host of features/mechanisms one would intuitively think should have been learned in the pretraining stage. In particular our contributions are as follows:

- We train one and two-layer versions of LlaMA on general code and fine-tune on Python code. These models show 15% to 24% of the weights are zero modes when using BF16 precision. For the two-layer model, ablating these zero modes causes the loss to increase by 3-3.5x (compared to 1.1-2.5x for ablating a random (equally-sized) set). For FP16 only 2-3% are zero modes and their impact is less. For FP32 there are no zero modes.[1]
- In two-layer models, we find induction heads whose zero modes are crucial for their functionality.
- Pretrained/finetuned pairs LlaMA 2 7B/LlaMA 2 7B chat, Mistral 7B/Zephyr 7B and Gemma 7B/Gemma 7B IT also have zero modes. For the former two, 2-6% of the weights are zero modes, whereas for the last pair it is around 0.5%. These zero modes are again crucial for their performance as compared to random ablations of equal size.
- For these three pairs we focus on particular layers to isolate the most impactful zero modes. For the Mistral series we show there is a single (monosemantic) neuron that fires on the beginning of sequences and whose activation is massively impacted by the ablations. For the LlaMA series, the zero modes of layer 2 are causing the model to being unable to do few-shot learning. The Gemma series is more subtle as it has much less zero modes (factor 10 less), but its last layer's down projection quintessential for the model to not get trapped in producing the same hidden state.
- For Zephyr 7B β the zero modes of sets of the deepest layers are crucial for model performance, in particular open-ended generation.
- Our zero mode ablation studies also helps identifying surprisingly many interpretable neurons. Many of which are syntactical and *universal* for the models tested. We also make connections with the outliers in [15], the distribution of weights and the structure of the latent space.
- Sometimes ablations of zero modes can be healed, sometimes they are beyond repair.

1.2 Methodology

To come to these findings, we followed the following general workflow for our experiments. We start by obtaining a pretrained model and its finetuned partner. This could be either done entirely from scratch (see the toy models in Sec. 2 and 3) or by using existing language models. Once we have these models we obtain the zero modes by creating a binary mask which is 0 when the weights agree and 1 elsewhere. We construct such a mask for each model component and use it to form new models that have various amounts of the zero modes knocked out [2]. We then also construct models that have a random and equally sized set of weights knocked out. This gives three (or more) models that can then be compared.

For the toy models, we study very simple tasks, such as continuing a sequence of text or a repeated sequence of integers/random tokens. For the LLMs we can do more extensive benchmarking and our primary focus will be MMLU [16] and MT bench [17]. MT bench is extremely helpful/essential to access the ablations, because they can be rather subtle and malfunctions might only arise in open-ended generations.

[1] There could be weights that are much closer to the pretrained model weights than others, but we did not pursue that here. We mostly focussed on BF16 precision as this is the precision typically used for popular open-source LLMs.

[2] Here we mean zero ablating the zero modes. We also experimented with setting weights to the mean, but that didn't change our results and so we will stick to zero ablation. It is also important to note that since the number of zero modes is very small, these ablations are a tiny change to the distribution of the weights.

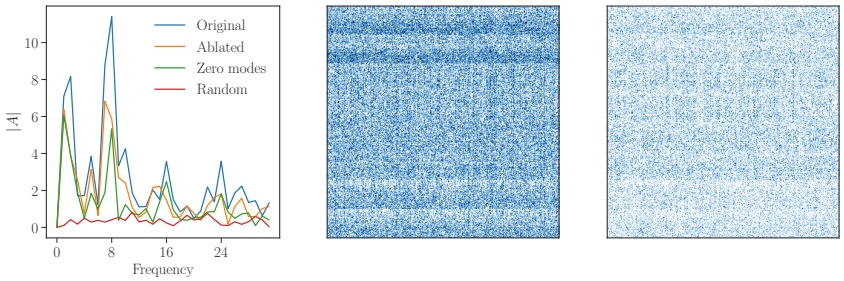

Figure 1: **Left:** Spectrum of W_K averaged over the second dimension and mean subtracted (abs amplitude as a function of frequency). We compare the original model, ablated model, the zero modes themselves and a random set of weights. We see the distribution of zero modes follows clearly the distribution of the full weight matrix W_K. **Middle:** Visualization of $|W_K|$ in which an entry is 1 if it is larger than the abs mean and zero otherwise. **Right:** Distribution of the zero modes, which shows the regularity in the y-direction as well.

Besides these benchmarks, we also study model internals (whenever they are useful) and look at firing patterns of neurons and attention maps.

2 One layer toy model

We will start the discussion with a very simple toy model: a one-layer version of LlaMA trained on 6B tokens of the algebraic stack of proof-pile [18] and finetuned it on just Phython code. See App. A for more details on the training/finetuning setup and model architecture.. We use this small model to show the existence of zero modes, study their evolution, impact on the loss and distribution. We will first focus on models trained using BF16 precision. Fig. 12 shows the loss curves.

Distribution Comparing weights of the finetuned and base model we find around 15% of the weights to be the same. This is rather large and we will see that in typical LLMs this is much smaller. To better understand their distribution, we consider the weights in the attention, W_K and W_Q. The weight matrices of the attention block in the transformer are structured according to heads and rotary embedding. This can be seen by the independence of the attention heads, which results in $2h$ independent blocks of size $d_{\text{head}}/2 \times d_{\text{model}}$. The factor of 2 is there due to the rotary embedding. It can also be seen in the weight matrices themselves through an FT. For instance, in our case we have $h = 4$, so we expect a fundamental frequency 8. Indeed, if we take W_K and take the average over the second dimension and perform a Fourier transform, we find the spectrum as shown in Fig. 1.[3] This shows that the distribution of the original and zero-modes ablated weight matrices (here only shown for W_K) is very similar. Showing zero-modes have some structure to them that is similar to the rest of the weights.

Robustness Besides their distribution, one can also ask how robust the zero modes are. For instance, we found that different learning rate schedules or with or without weight decay, 90+% of the zero modes remained the same. Here we kept the finetuning dataset fixed, but it would be interesting to vary that as well.

Dynamics As a function of steps, the number of zero modes decays, but does so differently depending on the learning rate schedule, see Fig. 13. Typically with cosine annealing there will be more zero modes than with a constant learning rate.[4] Furthermore, across the 7 different weight matrices, there are around 50-64% of the zero modes that didn't change at all during finetuning. Other zero modes typically bounce between a few different values before returning to their initial value.

[3] It is important to mention that the regularity is not the result of a 'per head' initialization; we initialize the all the weights at once with zero mean and a fixed std, preventing a bias to be created there.

[4] It would be interesting to see whether this correlates with other benefits cosine annealing versus has over constant learning rate.

learning rate	original model	ablated model	ablated model random
constant	2.61 ± 0.10	4.36 ± 0.08	3.14 ± 0.09
cosine	2.61 ± 0.10	4.83 ± 0.08	3.60 ± 0.09

Table 1: Changes in loss after zero-ablating the zero modes in the pretrained model. Zero modes obtained either from an SFT run using constant learning rate or cosine scheduler. It is clear the loss goes up significantly when we ablate the zero modes, much more than just a random ablation of the weights.

Performance To understand how these weights impact the performance of the model, we look at the change in loss on a test set (the test set of the algebraic stack of [18]) after zero-ablating the zero modes. From Tab. 1 we see that the zero modes have a bigger impact on the loss than just ablating an equal-sized set of randomly picked weights.

As noted in [9], a one-layer model is good at doing skip trigram statistics. We see this also in our model, however, since we also have an MLP it is not as straighforward. Typically the model obtains the correct token after the MLP and not already after the attention as was seen in [9]. From our experiments, these skip trigram relations are gone in the ablated model and the text makes much less sense.

Precision The models that we trained are all trained with weight decay. It is thus rather confusing that even with this regularization, there can be weights that do not change during finetuning. So unless there is some magical cancellation between the weight decay part and ordinary Adam update, we should not expect such zero modes to appear. This suggests that perhaps the precision of the training is important for the development of these zero modes. We indeed find that when we train with full precision (FP32), there are virtually no zero modes anymore after finetuning. This remains true even when weight decay has been turned off. For FP16 we do see zero-modes, but much less than in the BF16 case; around 2% of the weights. Their impact on the performance is also less than in the BF16 case.

It thus seems that the emergence of weights that don't change over the course of finetuning is something due to the fact that the quantisation is such that many weights would change value, but not after quantisation. The impact of BF16 seems to be largest. This is interesting as it a rather popular precision to perform finetuning with.

3 Two layer toy model

One layer models are interesting, but typically only encode very simple statistics and e.g. will not be able to make more complex associations, such as those found in [8]. Starting with two layers this behaviour can occur. In this section we discuss two-layer models and find they again behave in the same way as far as their zero modes are concerned. The training and finetuning setup is the same as before. See Fig. 12 for the loss curves. In contrast to the one-layer models, we will find that there are zero modes that are directly related to the functionality of induction heads [8, 9].

Distribution The distribution of the zero-modes again follows the distribution of the original weights (as viewed as deviations from the mean), see Fig. 2 for W_K of the second layer. We see again a fundamental frequency of 8 with the higher harmonics as well for W_Q and W_K. It is interesting to note that we found this periodicity primarily in the second layer.

Dynamics The amount of zero modes decays, but does not go to zero and reaches around 15-24% (depending on the component and layer) at the end of fine-tuning. See Fig. 14.

Performance Ablating the zero modes again results in a large performance drop, much more so than from just ablating a random set, see Tab. 2. We also studied ablating only zero modes in the first and second layer. The first layer's ablations appear much more impactful. In Tab. 7 we ablated the QK, OV and MLP seperately, showing the MLP in the first layer has most impact on the loss. We will study a potential reason below.

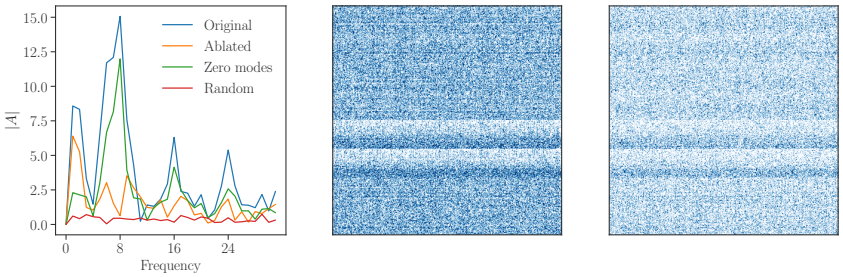

Figure 2: Spectral properties and distribution of zero modes in W_K for BF16 precision. **Left:** Spectrum of W_K averaged over the second dimension and mean subtracted (abs amplitude as a function of frequency). We compare the original model, ablated model, the zero modes themselves and a random set of weights. We see the distribution of zero modes follows clearly the distribution of the full weight matrix W_K. **Middle:** Visualization of $|W_K|$ in which an entry is 1 if it is larger than the abs mean and zero otherwise. **Right:** Distribution of the zero modes, which shows the regularity in the y-direction as well. We also see an interesting linear gradient (in the y-direction) for head $1:2$.

	ablations				
lr schedule	none	all	all rand.	L0	L1
cosine	$1.81_{\pm 0.08}$	$6.55_{\pm 0.06}$	$4.38_{\pm 0.08}$	$6.15_{\pm 0.09}$	$4.52_{\pm 0.08}$
const.	$1.81_{\pm 0.08}$	$5.54_{\pm 0.07}$	$3.00_{\pm 0.09}$	$5.08_{\pm 0.07}$	$3.50_{\pm 0.09}$

Table 2: Changes in loss after zero-ablating the zero modes in the pretrained two-layer model. Basemodel: BF16, SFT: BF16. It is clear the loss goes up significantly when we ablate the zero modes, much more than just a random ablation of the weights.

Precision Moving to FP16 instead of BF16, we see a lot less zero-modes, around 2-3%. However, these zero modes do again result in a larger increase in loss as compared to ablating an equally-sized random set. This is however for the model pretrained using BF16 and to understand the role of precision better we also studied an FP16 pretrained model. The amount of zero modes are around 2-3%, similar to the BF16 pretrained model and FP16 FTed model. Thus FP16 in either pretraining or FT causes much less zero modes. The performance impact is also less, see Tabs. 3.

3.1 Induction heads

As we mentioned in the beginning of this section, the interesting circuit that can emerge in two layer models is an induction head. These circuits were found in [8, 9] and various arguments were given for their existence, in particular in relation with in-context learning tasks. The formation of these induction heads was shown to correlate with a phase change in the loss curve. In Fig. 12 we can also see this for the two layered models but not for a single layered model. It seems natural that the induction heads are important circuits that are beneficial to keep *untouched* while finetuning.

To test this hypothesis, we study the finetuned model (with its ablated versions) on two variations of an in-context task where the model is given a set of repeated strings. In one variation these strings are four digit integers and in the second we use random tokens decoded to strings. First we identify the induction head in the original unablated model, after which we construct four more models to infer the impact of the zero modes. These models are: 1) the zero-mode ablated model, 2) the random ablated model, 3) ablated zero modes in the induction head and 4) random ablation in induction head. We analyze the accuracy on the first generated token by taking the logits[5] after the embedding and the two attentions and MLPs. Besides that we also study the attention maps of each head to infer where the next token gets most of its attention from.

We find that induction heads are present in our pretrained model and are located in head $1:2$, which is clear from the attention maps in Fig. 3. This can be seen by the lower diagonals having large non-zero values. Mechanically this means that head $1:2$ is responsible for paying attention to the correct token (in both the random token string or random integer string task). The question then is

[5]Obtained by passing the hidden states through a LayerNorm and unembedding.

		ablations				
PT/FT	lr schedule	none	all	all rand.	L0	L1
BF16/FP16	cosine	$1.81_{\pm 0.09}$	$2.45_{\pm 0.10}$	$1.93_{\pm 0.09}$	$2.18_{\pm 0.10}$	$1.97_{\pm 0.10}$
BF16/FP16	const.	$1.81_{\pm 0.09}$	$2.07_{\pm 0.09}$	$1.88_{\pm 0.09}$	$1.94_{\pm 0.09}$	$1.90_{\pm 0.09}$
FP16/FP16	cosine	$1.81_{\pm 0.08}$	$1.98_{\pm 0.08}$	$1.86_{\pm 0.08}$	$2.06_{\pm 0.08}$	$1.99_{\pm 0.08}$
FP16/BF16	cosine	$1.81_{\pm 0.08}$	$2.18_{\pm 0.09}$	$1.93_{\pm 0.08}$	$2.00_{\pm 0.08}$	$1.94_{\pm 0.08}$

Table 3: Changes in loss after zero-ablating the zero modes in the pretrained two-layer model. It is clear the loss goes up significantly when we ablate the zero modes, more than just a random ablation of the weights.

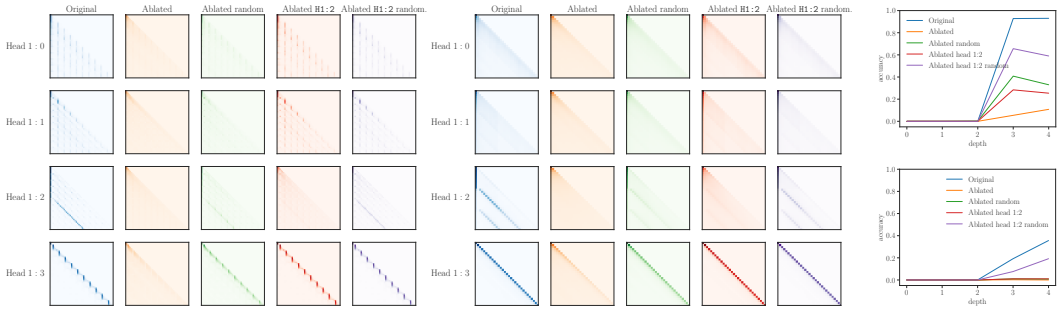

Figure 3: **Left**: Attention pattern for layer 1 for repeated digit task. **Middle:** Attention pattern for layer 1 for repeated random tokens task. Attention patterns are average over 500 examples. **Right top**: Accuracy on the repeated digit task. The horizontal axis is depth: 0: after embedding, 1: after 1st attention, 2: after first MLP, 3: after 2nd attention and 4: after 2nd MLP. The random ablation in head $1:2$ has almost no effect, but ablating the zero modes has! **Right bottom**: Same as right top, but for the repeated random token task. The first column for the original model shows that head $1:2$ ensures attention of the correct token is payed towards the final generated token. The accuracy also shows a clear jump after the second attention. These both provide evidence that head $1:2$ is an induction head.

whether this attention pattern has also caused the model to select the correct next token already after the attention map. This is indeed the case as we show in the accuracy plots in the left two plots (top for the repeated digit task and bottom for the repeated random token task). At depth 3, which means after the second attention block, when passing the hidden states through the unembedding to make a prediction (greedily) there is an rather significant jump (blue curve) in the accuracy, indicating the model has selected the correct hidden state after the second attention. These are the two requirements (at least in the emperical definition in [8, 9]) for this head to be an induction head.

Interestingly, the attention maps also show that the induction head is absent in the ablated models, except for the randomly ablated ones, since the large values on the lower diagonals have completely vanished and the accuracies have dropped significantly. In particular, after ablating the zero modes in head $1:2$, the accuracy reduces to random guessing. In contrast, a random ablation, leaves much of the accuracy intact. We also considered an inverted ablation, where we keep just the zero modes in the induction head, but keep the rest of the weights in all other heads/components/layers the same, see Fig. 16. This means setting around 80% of the weights in that head to zero! The result of that ablation keeps most of the accuracy intact, showing that the zero modes are doing most of the heavy lifting.

These observations provide suggestive evidence that these zero modes are important for the functioning of induction heads. [6] In terms of the attention patterns, we also see a clear deterioration, see Tab. 4, where we also included some other ablation studies.

[6] For the random token string task the accuracy drop is significant and can probably be attributed to the fact that in BF16 there are simply too many zero-modes. Indeed, for FP16, there are much less zero modes and we see a clear gap between the ablated and random ablated models.

	ablations						
task	none	all	all rand.	L0	L1	H1:2	H1:2 rand.
toks	$0.49_{\pm 0.15}$	$0.07_{\pm 0.01}$	$0.16_{\pm 0.05}$	$0.09_{\pm 0.04}$	$0.11_{\pm 0.02}$	$0.13_{\pm 0.02}$	$0.34_{\pm 0.07}$
ints	$0.53_{\pm 0.07}$	$0.04_{\pm 0.003}$	$0.14_{\pm 0.03}$	$0.04_{\pm 0.01}$	$0.07_{\pm 0.006}$	$0.07_{\pm 0.006}$	$0.28_{\pm 0.05}$

Table 4: Sum of attention scores that the final token receives from relevant positions (meaning positions of tokens that we expect to be generated) for a variety of ablations and the two tasks discussed so far. We also give the average std over those tokens. In general, ablations of zero modes have a big impact on the attention scores.

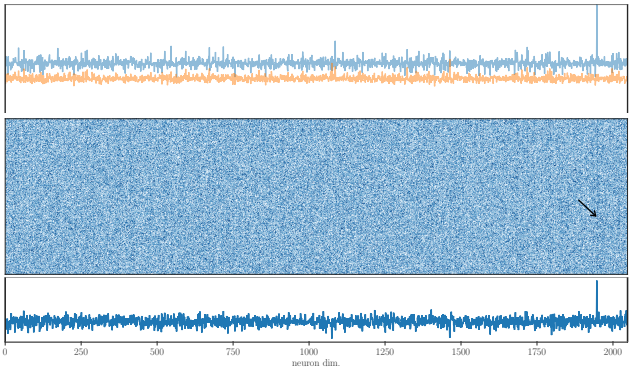

Figure 4: Gate projection of layer 0. **top:** L^1 norm of the ablated (orange) and unablated (blue) weights weights (divided by the dimension). **middle:** binarized weights according to the criterion whether it is bigger or smaller than the average. **bottom:** the (normalized) difference between the ablated and unablated weights. The black arrow indicates the neuron dimension 1947, which is most affected by the ablation of zero modes. Taking the average along the hidden dimensions also reveals that neuron 1947 has disproportionately many values above average.

We therefore see a clear causal effect between the zero modes in head $1:2$ and performance on the in-context learning tasks. Combined with the fact that head $1:2$ is an induction head (according to the empirical definition), it provides very suggestive evidence that these zero modes are most that matter for this induction head.

Finally, going back to the distribution of the weights and zero modes in Fig. 2, we see that the induction head (head $1:2$) has an interesting density profile for the weights, as in we see a almost linear gradient appearing in the y-direction for just that head. Actually, this occurs in many other models as well, as we will see below. Is there a reason the induction heads have profile for the weights? If we look at the OV circuit as well, we don't see this gradient in the density and the simplest explanation for this gradient would then be because of the rotary embedding. Nevertheless, this also immediately raises the question, why then does this occur in some heads and not in others? Maybe it only occurs in induction heads? But then in one-layer models, which cannot have induction heads, we also observed these types of weight profiles. What do these correspond to?

3.2 Neurons

Finally, the last piece of the network we can dive into a bit more are the neurons and see how the zero modes influence them. We would like to find neurons that correspond to features that should have been learned during pretraining. This turns out not to be too complicated. In layer 0, the neuron that has the largest change in activation value is one that fires on the BOS token. After ablation it fires less on this token and also starts activating on other (unrelated) tokens. This neuron is neuron 0:1947 and can actually be discovered just by looking at the weights as it is the neuron dimension that is most affected by the zero modes, see Fig. 4. This makes sense, as the BOS token is always there in the pretraining and so its representation in the model should have converged or at least should not have changed after finetuning.

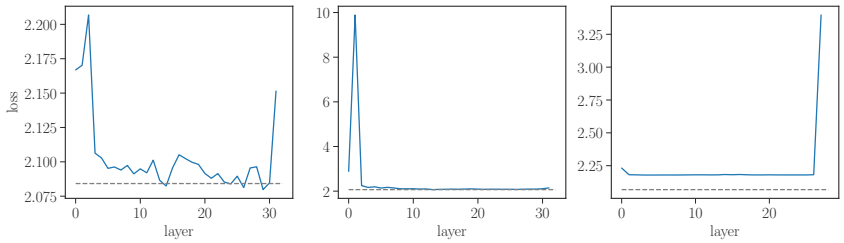

Figure 5: C4 loss on 1500 english examples for three 7B models. **Left**: LlaMA 7B chat, **Middle**: Zephyr 7B β, **Right**: Gemma 7B.

We will encounter this particular neuron also in the large language models we will discuss below. This is reasonable, because this is common to all models and their training procedure. It is interesting though that the zero modes seem to precisely capture this universality.

4 Large Language models

In the previous section we saw that in BF16, there are a substantial amount of weights that don't change during finetuning. The two layer model displayed that these zero modes in the case of an induction head are quintessential for its functionality. In this section we make a leap in model complexity and analyze the impact of zero modes on LLMs. We will show that again the zero modes are very important for the model's performance. Some cases even show a complete breakdown of producing any sensible output. We will study these models with the zero modes knocked out and compare against the original model and a model with random (equally sized) sets of weights knocked out.

We will first study global aspects and go more into details, such as specific layers or specific weight matrices of a particular layer in the next section. The focus will be on the LLaMA 2 7B family [3], Mistral 7B family (and Zephyr 7B) [4, 19] and Gemma 7B family [20].

4.1 Global aspects

In the previous section we saw that zero modes follow the same distribution as the rest of the weights. This remains true for the LLMs we tested, see Fig 17 for a sample. However, we find much less zero modes, around 2 to 6 % for LlaMA and Mistral, and for Gemma 7B less than 1%. The number of zero modes throughout layers does not change significantly.

For the Mistral series, we also have access to a SL'ed version used to construct Zephyr 7B β. Comparing zero modes between the base model, SL'ed model and Zephyr (DPO), we find substationally more zero modes (the difference between the weights is sparse at around 0.26 on average). This is probably expected because the DPO process changes more detailed nuances. It is also true that not all zero modes between the base and DPO'ed model are the same as the zero modes encountered via comparing the base with the SL'ed model and the SL'ed model with the DPO'ed model. Zero modes thus change but return to the intial value after some training steps, just as we saw in the toy models.

The performance of the models is heavily impacted by the zero modes. Taking 1500 english examples from the C4 dataset [21], we can already see this. For instance taking the ablating the zero-modes per layer we see great variance in the loss on this sample, see Fig. 5. For LlaMA 7B chat layers 0, 1, 2 and 31 are causing significant increases in the loss, for Zephyr 7B β this happens for layer 1 and for Gemma 7B layer 27 is the main culprit.

The severe performance degradation can also be seen by benchmarking for downstream tasks using for instance the MMLU test and MT bench. See Tab. 5. The ablated models all show a significant drop in both MMLU and MT bench, with Zephyr even incapable of doing anything once the zero-modes in layer 1 are ablated. It is also worth pointing out that the zero-modes of Gemma 7B only comprised around 0.5% of the weights and so the drop on MT bench is particularly surprising. In Fig. 6 we plot the accuracy after ablating zero modes in each layer separately. There is typically one early layer

		MT bench (GPT4)			
Model	zero mode ablation	MMLU (5)	Turn 1	Turn 2	Average
LLaMA 2 7B chat	none	45.71	5.88	5.71	5.79
	full	25.11	1.0	1.0	1.0
	full rand.	43.33	6.27	4.54	5.40
Zephyr 7B β	none	59.78	7.09	6.58	6.84
	full	0.00	1.0	1.0	1.0
	full rand.	52.03	5.64	4.58	5.11
Gemma 7B	none	63.37	6.97	5.39	6.18
	full	57.54	1.53	1.28	1.40
	rull rand.	62.56	6.86	5.98	6.42

Table 5: Three 7B models tested on 5-shot MMLU and MT-bench (judge is GPT-4). One can see the significant performance drop when the zero modes are ablated.

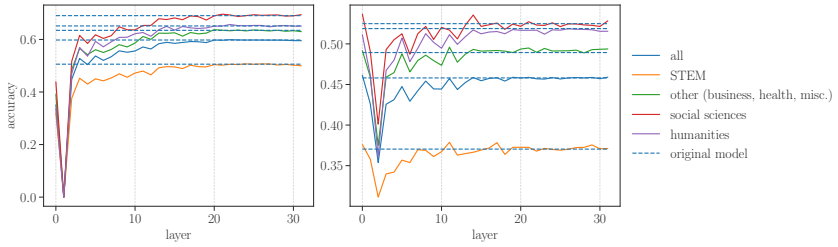

Figure 6: Accuracy on the MMLU test after ablating zero modes in each layer. **Left**: Zephyr 7B β, **Right**: LLaMA 2 7B chat. For Gemma 7B there was almost no variation, keeping almost all MMLU scores the same. It is interesting to note that the ranking between the five categories is the same for all models we tested.

whose zero modes explain to a large extend the drop in performance. For LLaMA that is layer 2 and for Zephyr (Mistral) layer 1.

We end this subsection with two additional observations about some global aspects of the model. *Outliers*, as found in [15] are specific feature dimensions where hidden states have large coefficients. Turning these feature dimensions off has severe consequences on the model's performance and so they might be related to what we see for the performance drop. To study this[7], we constructed hidden states for 128K tokens taken from the RedPajama dataset [22]. For Zephyr 7B β we see a massive reduction on the number of outliers after ablation, for LlaMA there is also a fluctuation in the number of outliers, but it is less severe. For Gemma 7B there is no significant change. See Fig. 7. Zero modes are thus not necessarily related to the outliers observed in [15].

[7]We focussed here on the criterion in [15] that coefficients of the hidden states need to have magnitude of at least 6.0. We did not incorporate the other constraints.

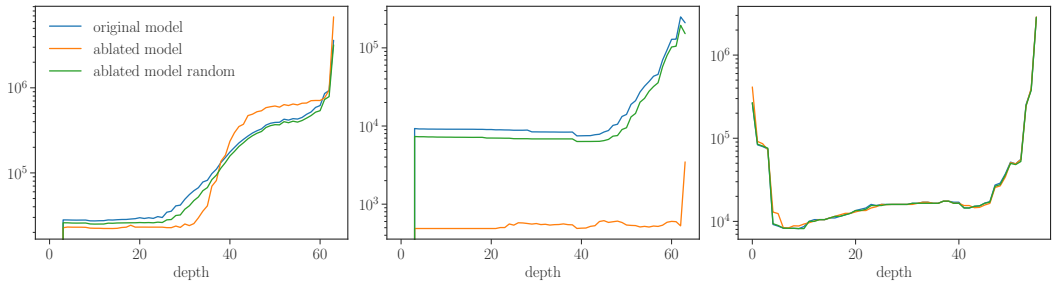

Figure 7: **Left**: Outliers of LLaMA 2 7B chat, **Middle:** of Zephyr 7B β and **Right**: Gemma 7B and their zero-mode-ablated counter parts. We also plotted what a random ablation would do.

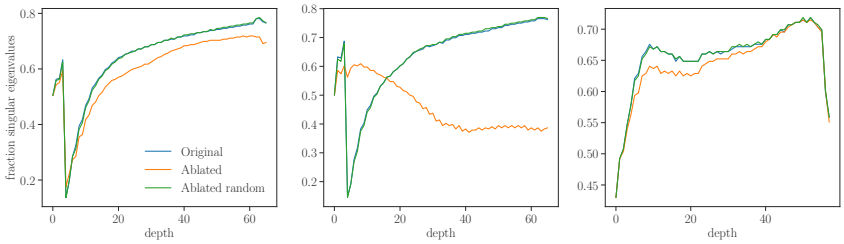

Figure 8: Fraction of singular values needed to explain 90% of the variance for 500 contexts of length 256 from the RedPajama dataset. Singular values are averaged over the 500 contexts. **Left**: LLaMA 2 7B chat, **Middle:** Zephyr 7B β and **Right**: Gemma 7B.

Another observation about the latent space is that some of the models we studied have transitions in their depth, in which the necessary hidden dimensions needed is massively reduced. More formally, we take 500 example contexts of length 256 of the RedPajama dataset [22] and construct the hidden states after each component of the network (so after attention and MLP). We then compute its singular values[8] and ask how many we need to explain 90% of the variance and average that over the 500 examples. This gives us an estimate of how many dimensions are being employed. In Fig. 8 we show how the fraction of dimensions employed develops as a function of depth (here we generate only a single token). We see that the zero modes can have a large impact here, but the perhaps more interesting observation is the sudden drop after the first layer. This drop co-occurs with the outliers appearing suddenly as we see in Fig. 7. It would be interesting to understand this better.

4.2 Transferability of zero modes

One question regarding the zero modes is whether they are a property of the base model or whether the zero modes one gets from finetuning on different sets is different. This question also touches upon our hypothesis that the zero modes are related to circuits/mechanisms/concepts established during pretraining and hence we would expect the zero modes to be important regardless of how one finetunes.

To study this question, we consider two models derived from LlaMA 2 7B, which are different from the chat version Meta released and we have been discussing so far. One of them was a model we finetuned on 100k entries of the Orca dataset [23, 24] [9], we will refer to this model as Mila 7B [10] and the other is Lemma 7B [18]. For both models, if we ablate the zero modes obtained by comparing the base and RLHF LLaMA 2 7B pair, we again see a profound drop in performance. Interestingly, if we check the zero modes of Mila 7B, we get a huge set; around 30% of the weights did not change. Ablating all of those will definitely destroy the model. A substantial amount (40-50%) of the zero modes of the pure LlaMA pair are also among these zero modes. Another interesting case is CodeLlaMA, since that model is further pretrained. It has far less zero modes, around 0.1% and are less impactful.

The zero modes of the Mistral series have an equally bad impact on Mistral 7B v0.1 (the base model) as the DPO'ed version.

5 Case studies

We will now study the models mentioned above in a bit more detail, focussing on particular aspects of each. Determining precisely what the zero modes are responsible for is inherently a difficult. The failures could be the result of many factors and the zero-modes could result in subtle changes in each layer but have a big effect when ablated together. Furthermore, earlier layers typically have a bigger impact as the subtle changes there can lead to big errors in later layers. We will see examples of this

[8] The data matrix has size $\text{depth} \times N \times L \times H$ with $\text{depth} = 66, 58$ (for Gemma 7B), $N = 500$, $L = 256$ and $H = 4096, 3072$(for Gemma 7B). The SVD is computed on the last two dimensions and so yield 256 singular eigenvalues for each context and depth.

[9] Due to memory constraints we limited the context window limited to 2048 tokens.

[10] I am not affilated with the institute.

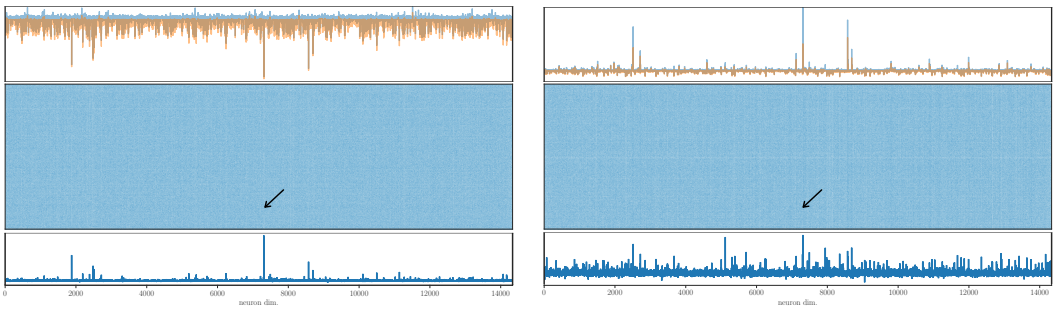

Figure 9: **Left**: Down projection of layer 1, **Right:** Gate projection of layer 1. The top plots are the L^1 norm of the ablated (orange) and unablated (blue) weights weights (divided by the dimension). The middle are the binarized weights according to the criterion whether it is bigger or smaller than the average. The bottom ones are the (normalized) difference between the ablated and unablated weights. The black arrow indicates the neuron dimension 7310, which is most affected by the ablation of zero modes.

below. The general philosophy here will be to try to find a specific layer that has most impact, e.g. as we saw in Fig. 6. Furthermore, we will also focus on very rudimentary tasks such as the ability of repeating sequences of integers or random tokens. We will compare neuron activations, attention patterns and logit distributions to get more insight into the role/relevance of the zero modes.

LlaMA 2 7B, Zephyr 7B β and Gemma 7B have interesting malfunctions once the zero-modes are ablated. For LlaMA 2 7B we will see that many in-context learning tasks are heavily impacted and for Zephyr 7B β the model has a *single* (monosemantic) neuron that fires on the first sentence. Perturbing its activation can break the model. In fact, we will see that this neuron occurs in all models and is very similar to the neuron we encountered in our two-layer toy model. These observations again support the idea that zero-modes are correlated with concepts or mechanisms that should have been learned during pretraining.

The case studies below also focus on different aspects and so the test/evals we do are different for each case study.

5.1 Zephyr 7B β

In Fig. 5 and 6, we saw layer 1 has the most impact for Zephyr 7B β. The ablation of these zero modes actually causes the model to output an endless repetition of the same token. It turns out that this is due to the zero modes in the gate and down projections. Ablating the zero modes either of these components is enough to kill the model.

The weights of the gate and down projection are interesting to visualize[11] and comparing the unablated and ablated weights. See Fig. 9. The first thing one notices is that there is one particular neuron that is impacted most by the zero modes. We will come back to this neuron below. Second, there seems to be some regularity in these binarized weight matrices, which can be seen from a Fourier transform having large amplitudes $|A_{k_x k_y}|$ where either k_x or k_y is zero[12].

Before moving on to analyzing layer 1 in more detail, we also looked at ablating zero modes of many deep layers as it was suggested that some of these layers are perhaps not that useful [7]. We find that the deeper layers' zero modes might not have a big impact on benchmarks like MMLU, they do

[11]To visualize we found it useful, as before, to convert the absolute value of the weights to a binary matrix with the criterion whether a given weight was bigger or smaller than the average (of the abs of the weights). This might seem arbitrary, but it turns out to provide a very clear visualization of the density of weights that are 'large' and 'small' and it turns out to produce some regularities, just as we saw for the one and two layer models.

[12]This regularity is interesting. For the attention weights this is perhaps expected because of the head structure, but the MLP does not have this imposed structure. It can of course inherit something from that, but we don't see that. It could be some notion of independence between the neuronic dimensions or hidden dimensions, but it would be interesting to understand that better.

have a tremendous effect on open-ended generation as shown by the severe performance drop of MT bench, see Tab. 8.

5.1.1 Neuron 1:7310 : A neuron that fires on the first sentence

Moving on to neurons, from Fig. 9 it is clear that neuron 1:7310 appears to be interesting regarding our zero mode ablation. This was however, purely motivated from the weights themselves and typically when we analyze neurons we study firing rates and/or activations. These are also heavily impacted by our zero modes. Let us focus on the original model and the model with the zero modes in the gate projection of layer 1 knocked out. Neuron 1:7310 is the neuron in layer 1 whose activations and firing rate are most impacted. It is mostly monosematic[13] and fires on the beginning-of-sequence (BOS) token (whatever string that might correspond to as this is model dependent)[14] and the first new line token `\n`. In other words, it fires on the first line. This again seems to be a feature that will be learned already during pretraining, since every element in the batch starts with an BOS token and has most likely also `\n` somewhere in the context.

The ablated model now freaks out when it encounters the BOS token at the start of the sequence as its dedicated neuron is gone and as a result it cannot produce any coherent text (it is surprising the effect is so severe). For instance, consider this question from MTBench,

> *A tech startup invests $8000 in software development in the first year, and then invests half of that amount in software development in the second year.What's the total amount the startup invested in software development over the two years?*

The model replies with an endless stream of the token *bekan*. This particular stream actually occurs in many situations.

The same malfunction also occurs for the base model Mistral 7B v0.1. If, however, we get rid of the BOS token and feed this to the model, the generation is totally fine with MTbench scoring 6.27 and 6.79 for the original and ablated model. Putting a BOS token at a random spot in the context also has no effect; the ablations really only have an effect when it is the first (or in some cases second) token. We can also entirely heal the behaviour of the ablated model easily by scaling up the activation of neuron 1:7310 by a factor so as to bring it back to the unablated model's level. The MTbench score goes from 1.00 to 6.70. We will also discuss healing via retraining below in Sec. 6.

It makes sense that the model is actually so sensitive to this particular activation. Since everything that is fed to the model has the BOS token to start with, one explanation would be that it quickly overfits to this and builds the rest of its functionality around it. One therefore should expect large variations when messing around with weights corresponding to this aspect of the inputs.

5.1.2 Other interesting neurons

So far, we mostly looked at an individual neuron that appeared to be very monosemantic (at least within the distribution of contexts we studied). One can also wonder whether other interesting neurons also pop up when comparing the ablated and original model. Weights are of course rather different than neurons as they require an input and it is typically their activation patterns we are interested in. These patterns are not only impacted by weights in the current layer, but also all previous components and hence dissecting the effect of our zero modes is much harder[15]. Nevertheless, these zero modes do have the effect of changing hidden states and activations throughout the network. Furthermore, since these patterns drastically change when ablating zero modes, the other advantage is that one has a natural way of selecting particular neurons by determining which activations or firing rates changed most. It turns out that some of these comparisons yield rather interpretable neurons. See Sec. E.1 for a list of some specific examples. In general there seem to be multiple neurons that fire on the same tokens, such as the 'first sentence neuron' we discussed above, seems to be present in other layers as well (and other models). This is an example of a duplicate neuron as found in [25]. There are also neurons that are the inverse of the 'first sentence neuron', i.e. they fire on everything but

[13] Pure monosemanticity is hard.

[14] If this token is present. If it is not, then it fires on another tokens. Such a situation is rather out-of-distribution however.

[15] One reason why one might think this is not that hard, is because the zero modes form a tiny set and a random set has not nearly as much impact.

model	0-shot	1-shot	5-shot
Original	66.85	68.59	72.61
Ablated	55.49	52.88	49.17
Ablated L2	66.69	67.96	69.30
Ablated L2 random	66.30	68.75	72.61

Table 6: Results LlaMA 2 7B chat on the Winogrande dataset as a function of the number of shots. The fully ablated model has difficulty performing well on this dataset and is also not helped by prompting it with multiple shots. The impact of few-shot prompting is also far less for the model with just the zero modes in layer 2 ablated.

the BOS and first \n token. Besides these more syntactical neurons, we also found neurons firing on particular concepts, such as medicine/biology or Latex. Some of these neurons appear to be rather monosemantic.

5.2 LlaMA 2 7B chat

LlaMA is an interesting case study, because it is not completely wiped out by knocking out the zero modes. The overall performance does drop significantly (MMLU reduces to chance) and MTbench is at its minimum. However, if we test for more basic tasks, like repeating integers, it will still produce a set of integers, but just not the correct sequence. Concretely, giving the model the sequence

```
7059 4536 5221 1060 7758 7059 4536 5221 1060 7758
```
(1)

the completion of the unablated and ablated model are, respectively,

Unablated: `7059 4536 5221 1060 7758` (2)

Ablated: `4536 21 106 75 10 7 5 2 1` (3)

This clearly shows the model is put rather far off distribution by just ablating a tiny set of weights. If we ablate either the MLPs, QKs or OVs, we see that QKs have by far the most impact. This hints perhaps at some attention/in-context learning/few-shot prompting is malfunctioning.

To study these potential failure modes, we focus on whether few-shot prompting still works by studying the Winogrande evaluation [26] and arithmetics [2]. Furthermore, in the appendix we also mention some result on the copying task discussed for the toy models. For those tasks we also observe that induction heads are heavily impacted in the ablated model.

From the MMLU results in Fig. 6, we see that layer 2's zero modes are most impactful. Indeed the total MMLU score with just the zero modes in this layer ablated is 35.39. The MTbench score remains relatively high at 5.59. Focussing on this layer and doing a few-shot analysis of the Winogrande evaluation, we obtain the results in Tab. 6. The zero mode ablations have big impact on the performance and the ability to learn in-context. Ablating just layer 2 shows that few-shot prompting has much less effect as compared to the original or random set. This is also true for the arithmetics task as shown in Fig. 10, especially for sums with more than two digits. Here we did not include the results for the fully ablated model as it would not given anything sensible at all. The interesting feature about this plot is that whereas sums with only one or two digits are performed on par with the original model, once more than two digits (left plot) are involved, the model has a lot of difficulty. For 0-shot all three models perform the same and so the zero modes don't do much there, but more shots do not help the model that has the zero modes in layer 2 knocked out. It is remarkable that knocking out such a tiny set of weights can have profound consequences, especially since the random ablation didn't make any dent.

5.2.1 Neuronic analysis

Just like the toy models and Zephyr, LlaMA also has a neuron (actually multiple) that fire on the BOS token and something that signifies the end of the first sentence. It could be a period or \n. Ablating all zero modes reveals that in the early layers, there are neurons that have this behaviour. In particular these are neurons 1:7890, 2:1661 and 4:2618[16]. They are all the most impacted neurons after the

[16]These neurons are also the neurons that receive the highest Wanda score [12]. In general a sizable fraction of the zero modes gets a high Wanda score, but the variation per component in each layer varies a lot. Hence, pruning with the Wanda metric seems to be sensible from the point of view of our zero modes.

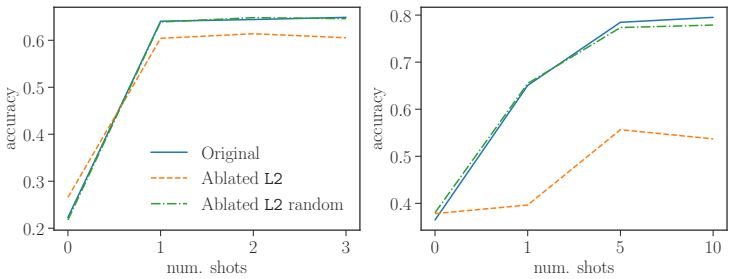

Figure 10: Accuracy on arithmetics (task taken from [2]). **Left**: Sums with 2 or less digits, **Right:** sums with more than 2 digits.

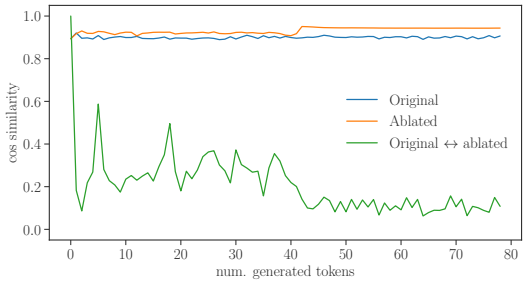

Figure 11: Cosine similarity of successive hidden states (blue for the original model, orange for the ablated model) and between hidden states of the ablated and original model (green). The query is: 'Write a Python function that computes the n-th Fibonacci number.'

ablation in terms of activations. If we focus just on the model with layer 2's zero modes knocked out, we find that there are neurons that previously were monosemantic (or at least to some extend) are now highly polysemantic and fire on a lot more tokens. For instance, take neurons 2:9905. This neuron was firing on the BOS token and some other tokens (including a `\n` or period), but the firing is sparse. After the ablation it has a very dense firing pattern. In App. F we documented a few more interesting examples we encountered.

5.3 Gemma 7B

Finally, we study the base model Gemma 7B. The model dimensions are slightly different from the other two models discussed and so is an interesting case to analyze as well. From the C4 losses shown in Fig. 5 it is clear that the last layer's zero modes have the most impact on the loss. Indeed, ablating only this layer results in a MT bench score of 2.24 (compared to 6.02 for a random ablation of that layer). MMLU on the other hand is not affected by this. One can even isolate the problem more and show that the zero modes in the down projection are causing all the harm.

Upon inspection the MT bench responses from the model with the zero modes of the down projection of layer 27 ablated, we see that the first few generated tokens (which could be multiple sentences) make sense, but after some number of tokens, the model gets stuck in generating the exact same token over and over again. This token (could also be a few) remains the same, not only after the end of the model, but also after every component within. If we study the hidden states of a few queries, the hidden states of the original and ablated model stay close after every generation, but are mutually rather orthogonal, see Fig. 11.

This suggests the zero modes in this down projection weight matrix are relevant for keeping the hidden states in check while generating [17]. It is not unreasonable that this is again something that is learned during pretraining and should remain the same after finetuning.

[17] It is also important to notice that the down projection is the final weight matrix before the final layer norm and unembedding.

5.3.1 Neuronic analysis

Just as in Zephyr and LlaMA, we can find neurons that fires on the BOS token. For instance, neuron 1:18657. The ablated model's version has a completely different activation pattern and pops up when comparing it with unablated models. It also does not just fire on the BOS token, but also on unrelated other stuff and so is (more) polysemantic. Perhaps a sparse auto-encoder, such as the ones used in [11], can be used to disentangle these different activations. Another set of interesting neurons are in the last layer. Neurons 27:3959, 27:392, 27:5490, which fire on everything but the BOS token. We also saw these neurons for the other two models and are thus suggesting they might be universal.

6 Healing zero modes

Typically when pruning a model, the models needs to be retrained so as to heal the damage done [7,27], but is not strictly necessary as for instance discussed in [12]. In our discussion so far, we have looked at a small set of weights that do substantial damage. It is therefore rather opposite to the pruning literature where large sets are ablated, but one can still ask whether these tiny prunings can be healed as well or not[18].

6.1 Two layer toy model

As a toy problem lets consider the BF16 pretrained model discussed in Sec. 3 and try to heal the ablated model. We took around 100M tokens from the pretraining dataset and ran a training run with that (using the same hyperparameters, but lowering warmup to 1000 steps). This completely healed the model, bringing the loss down to the one at the end of training of the unablated model. We also see the induction heads appearing again. The fact that the healing works here is no surprise, because the damage done by these zero modes is rather severe (removing functionality of an induction head) and because the measure for how broken the model is is just the value of the loss. There are not really more in depth tests one could do to access the damage that is left, apart from the in-context learning tasks discussed in Sec. 3, which do not show a large deviation from the performance of the original model.

6.2 Large language model

Healing is much more interesting and tricky for more complex models. As an example, let us look at the Mistral series. We saw the zero modes in the gate projection are crucial and so will *solely* heal this projection. For instance, let us take Zephyr 7B β with the zero modes of the gate projection of layer 1 ablated and retrain it on the dataset used to go from Mistral 7B v0.1 to the SFT version [19, 28]. This improved the model, bringing MMLU to 59.66 and MT bench to 6.80. The model thus appears to be healed and does not freak out by the BOS token anymore. Internally, we can also see that the model is back at the original level, since the activations have a small difference with the original model. However, despite neuron 1:7310 seemed to be the most impacted by the ablation of the zero modes, the healing has not tried to increase the activation of this neuron. Instead, it increased it for neuron 1:8572 in layer 1, which fires as well on the BOS and the first `\n` token (and which was also heavily impacted by the ablation, but not as bad as neuron 1:7310). Inspecting other activations, reveals some activation patterns in later layers that are interesting and naively one would expect being healed. For instance, 30:68 in the original model fires on the first `\n` and BOS token, but after the ablation it *just* fires on the first `\n` token. The healed model restored the firing pattern so it fires on both the BOS and first `\n` token. Nevertheless, scaling the activation of this neuron manually didn't have the same effect (we saw this did work for neuron 1:7310).

An aspect of this healing process that is worth mentioning is that the healing dataset we used here does not put any BOS token (as can be seen how the chat template is implemented here). This might explain why it did not heal the model by changing neuron 1:7310's activation. This also makes it particularly curious why healing worked at all and the healing is actually much more subtle. Recall that when we query the model with a prompt without any BOS token, the model would respond normal and its MT bench score would be high (close to original). Hence, the objective would not

[18] Of course, we have used zero-ablation to construct our ablated models, which from an weight initialization point of view is rather strange, because one would never set weights to zero at initialization.

be dramatically high (the training loss is indeed not experiencing a large drop in the initial part of retraining) and so the incentive to change the faulty activations is much less, which is perhaps why the activation of neuron 1:7310 did not change much. The damage the retraining therefore heals is much more subtle and far less obvious.

Despite the partial success for Zephyr, if we do the same for the base model Mistral 7B (v0.1), where the ablation has the same effect as it had on Zephyr, the healing did not work and the model would still freak out by the presence of the BOS token. Again, the BOS token was not included. If we include a BOS token explicitly in the SFT training we can heal the model, bringing MMLU back to 62.24, but MTbench remains stuck at 2.34. Upon inspection of the MT bench generations, we see that the generation in most cases starts out well (which probably also explains why MMLU is strong after healing), but degrades quickly after some number tokens have been generated. Again this shows that healing is perhaps not as trivial as expected.

7 Conclusion & Discussion

We have shown that language models (tiny ones or large ones) have weights that have not changed after finetuning. Their impact on the model's performance is profound, making them an interesting but tiny set of weights to study. Using various ablations studies, we showed that these weights can be related to induction heads or neurons encoding basic syntactic/linguistic properties. We end with a few comments.

Duplicate and universal neurons We noticed that in some models there are quite a few neurons that fire on the same tokens. This was also seen in [25]. This suggests there is some redundancy in these networks still. The neurons that we found using our comparisons were often also duplicates and so suggests their activations are already converged in pretraining. This was also a suggestion in [25] although there work focussed on pretrained models and so it is slightly different here.

Besides the neurons we found being duplicated inside models, they also often appear to be present in other models as well, i.e such neurons appear to be *universal neurons*. This makes sense, because there are many features that are the same in the training data, but it is interesting that from our comparisons rather similar types of neurons appeared. For instance, we saw many neurons associated with the BOS or end of sentence tokens and it thus makes sense the weights associated with them have converged in the pretraining stage.

Pruning is dangerous Despite the increasing body of literature that studies pruning (see [7] for a nice and recent overview), showing its potential, it is not entirely clear this method is without danger. We studied ablating specific weights that clearly cannot be knocked out, but above all, one needs an extensive set of benchmarks to show the efficacy of pruning. In particular, benchmarks with open-ended generation such as MT bench are important to add to this as we have seen a good score on MMLU does not guarantee a good MT bench score. This also holds after applying a healing strategy.

Backdoors The fact that Mistral could not be healed entirely and still could freak out by the presence of the BOS token, could be seen as analogous to a backdoor, albeit a rather uncontrolled one. It does not output coherent text, whereas typically in the discussion of backdoors one would want that. In our discussion, the zero modes ablated Mistral models open up a backdoor to producing non-sensible outputs with the BOS token as trigger. Despite the differences, the difficulty of removing the BOS 'backdoor' could perhaps be analogous to [29].

Limitations We have shown a correlation between zero modes and certain model behaviours. The initial premise was to relate the zero modes to concepts and mechanisms learned during pretraining. We showed some evidence for such a correlation. A more extensive study where one finetunes on different datasets and see whether the zero modes correspond to a skill already acquires during pretraining in all such cases. Such a study would also include what we would actually mean with 'basic behaviours' a pretrained language model is supposed to possess.

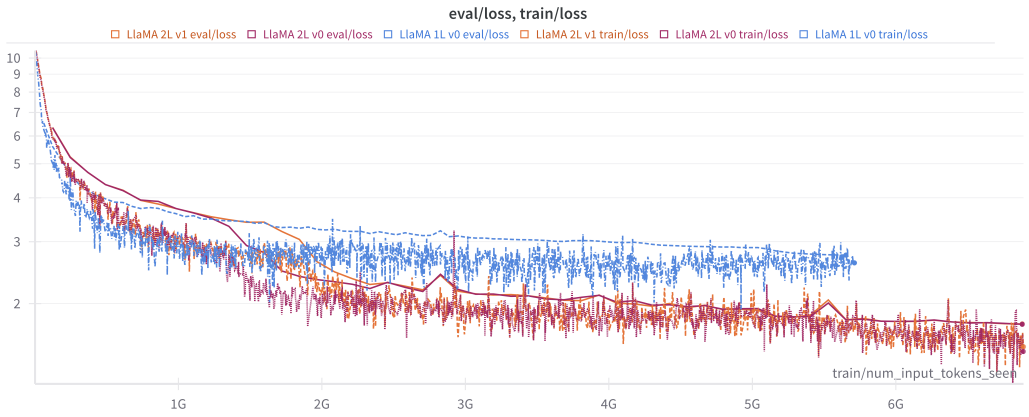

Figure 12: Eval and train loss for the one and two-layer LlaMA models. The version 0 curves are for the models trained with BF16 precision and the v1 is FP16.

Acknowledgement

I am supported by NSF grant PHY-2207584.

A Training details toy models

A.1 Pretraining

The toy models discussed in sections 2 and 3 we studied one and two-layer LlaMA 2 style transformers trained on code []. The architectural details can be found here. In short, the architecture consists of a standard multiheaded attention block (with pre LayerNorm) and an MLP of the form

$$\texttt{MLP}(x) = W_{\text{down}} \left(\texttt{SiLU}(W_{\text{gate}} x) \odot W_{\text{up}} x \right). \tag{4}$$

Hence each layer has seven different weight matrices. For both models we used $n_{\text{heads}} = 4 = n_{\text{kv heads}}$, $d_{\text{model}} = 512$, $d_{\text{ff}} = 2048$ and a context length of 1024. The tokenizer is the standard LlaMA tokenizer on huggingface. The dataset is the train subset of the algebraic stack of EleutherAI's Proof pile 2 [18] and can be accessed https://huggingface.co/datasets/EleutherAI/proof-pile-2. Both models were trained using AdamW with $\lambda = 0.05$ and a cosine learning rate scheduler with maximum learning rate $\eta = 10^{-4}$. We used 5000 warmup steps and trained for 5.7B tokens and 6.9B tokens for the one- and two-layer model, respectively. The batch size is 120. See Fig. 12.

One thing to notice about these training curves is that for the two-layer models, we see a substantial drop in the loss around 1.5 - 2B tokens. This is probably the same loss drop as seen in [8] and could be attributed to the development of induction heads, but more extensive analysis would be needed to be conclusive.

A.2 Finetuning

For finetuning we considered the test set of the algebraic stack and filtered it on just Python code. This dataset formed our finetuning dataset and consists of around 150M tokens. We used AdamW for 6 epochs with cosine annealing using 50 warmup steps and $\eta = 10^{-6}$ as the maximum learning rate. The batch size is 10. As discussed in the main text we used both no weight decay or $\lambda = 0.05$ and FP16 or BF16 precision.

B In-context learning task

We considered two tasks:

	ablations				
precision	none	QK L0	OV L0	MLP L0	Gate L0
BF16, cosine	1.81 ± 0.09	3.63 ± 0.08	4.16 ± 0.08	5.48 ± 0.09	2.36 ± 0.09

Table 7: Changes in loss after zero-ablating the zero modes in the pretrained two-layer model with precision. Zero modes obtained either from an SFT run using cosine scheduler. It is clear the loss goes up significantly when we ablate the zero modes, much more than just a random ablation of the weights.

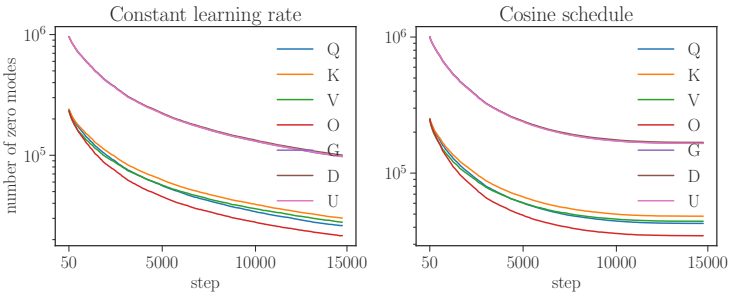

Figure 13: Dynamics of number of zero modes over the course of finetuning of one layer model. **Left:** Constant learning rate $\eta = 10^{-6}$. **Right:** Cosine schedule, $\eta_{\max} = 10^{-6}$, $\eta_{\min} = 0$ and 50 warmup steps.

1. A string of 5 four digit integers repeated two times, so 10 integers in total.
2. A string of 10 random tokens, repeated three times. We did not filter the tokens we draw from on rarity.

C More results on the toy models

In this appendix, we provide some more details on the toy models. In Fig. 13 we plotted the number of zero modes as a function of finetuning steps for two different learning rate schedules. The cosine flattens off, just because the learning rate becomes very small there and weights barely change. We checked that for more finetuning steps, the constant learning rate schedule also plateaus or at least starts decaying very slowly.

In Fig. 14 we plotted the same but now for the two layer model with both the pretraining and finetuning with BF16 precision. It is interesting to note that the zero modes in W_O are consistently less than the rest in the attention (this was also the case in the one layer model). There is also a clear distinction between the first and second layer.

In Figs. 15 and 16 we plotted the attention patterns and accuracy after each component for the two in-context learning tasks with different ablations. Especially the one where we ablate everything but the zero modes of head 1:2, clearly showing that we can retain much of the accuracy, despite $\sim 80\%$ of the weights set to zero.

D More results on global aspects

Here we report a sample of the distribution of the weights and their zero modes. We looked at the W_Q part, see Fig. 17, but one can also see similar structure occurring during to the multiheadedness in the other attention matrices. The W_O matrix also inherits a head structure as it is always paired with W_V, which has a head structure.

From the figure it is clear that the zero modes, despite being very faint, follow the same head structure of the rest of the weights and are thus not merely a random set of the weights.

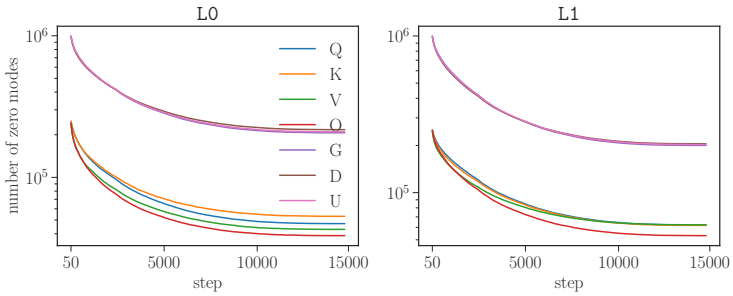

Figure 14: Evolution of zero modes computed by comparing the pretrained model (BF16) and SFT checkpoints in BF16 precision. **Left:** Layer 0. **Right:** layer 1

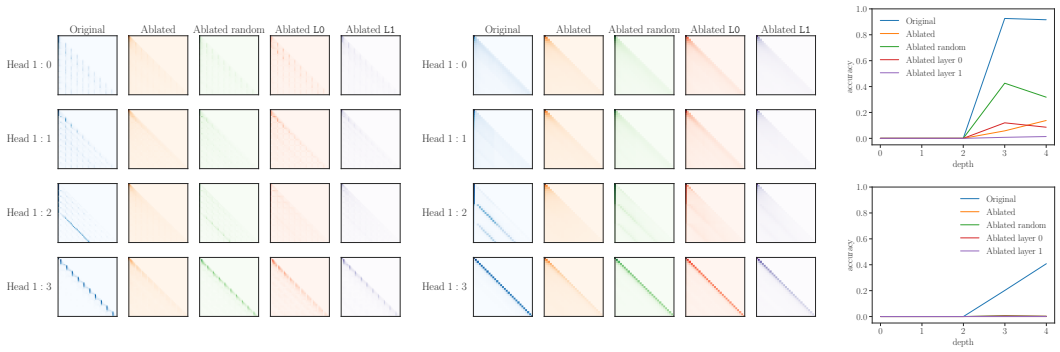

Figure 15: Ablations study, comparing original (blue), ablated (orange), ablated random (green), ablated L0 (red) and ablated L1 (purple). **Left**: Attention pattern for layer 1 for repeated digit task. **Middle:** Attention pattern for layer 1 for repeated random tokens task. Attention patterns are average over 500 examples. **Right top**: Accuracy on the repeated digit task. The horizontal axis is depth: 0: after embedding, 1: after 1st attention, 2: after first MLP, 3: after 2nd attention and 4: after 2nd MLP. **Right bottom**: Same as right top, but for the repeated random token task. The first column for the original model shows that head $1:2$ ensures attention of the correct token is payed towards the final generated token. The accuracy also shows a clear jump after the second attention. These both provide evidence that head $1:2$ is an induction head.

E More results on Zephyr 7B β case study

Below we list a few more details about the analysis of Zephyr 7B β. In Tab. 8 we present some results on other ablations of layer 1 and some deeper layers. This shows that the down and gate projections of layer 1 are the most important. Furthermore, since the experiments in the main text did not involve the chat template, we discuss that also in this table, but only for MTBench. The performance with and without the chat template is significant, especially for the ablated model where we knocked out the zero modes in the gate projection of layer 1. There the model can do open-ended generation and the main reason being that with the chat template, there is no BOS token to worry about.

We also looked at ablating the complement of layer 1 (so take out everything in layer one that isn't a zero mode). This had severe consequences and caused the model to break. This makes sense because this means setting 90%+ of the weights to zero. However, doing it only to the gate projection in layer 1 the effect is much less severe. This is interesting because despite loosing 56M parameters, the MMLU score dropped by 10 procent points and the open-ended generation remained at a reasonable level (one point drop). Most likely with a little healing this ablation can be completely restored.

Finally, the zero mode ablation of many deeper layers is in general also not a good idea, see last three rows in 8. This clearly shows that while the MMLU score might be good, open-ended generation is essentially destroyed.

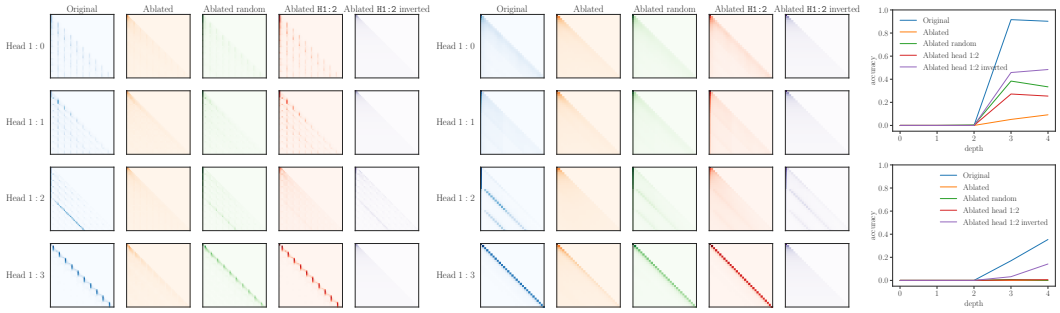

Figure 16: Ablations study, comparing original (blue), ablated (orange), ablated random (green), ablated `H1:2` (red) and ablated all but zero modes in `H1:2` (purple).**Left**: Attention pattern for layer 1 for repeated digit task. **Middle:** Attention pattern for layer 1 for repeated random tokens task. Attention patterns are average over 500 examples. **Right top**: Accuracy on the repeated digit task. The horizontal axis is depth: 0: after embedding, 1: after 1st attention, 2: after first MLP, 3: after 2nd attention and 4: after 2nd MLP. **Right bottom**: Same as right top, but for the repeated random token task. The first column for the original model shows that head $1:2$ ensures attention of the correct token is payed towards the final generated token. The accuracy also shows a clear jump after the second attention. These both provide evidence that head $1:2$ is an induction head.

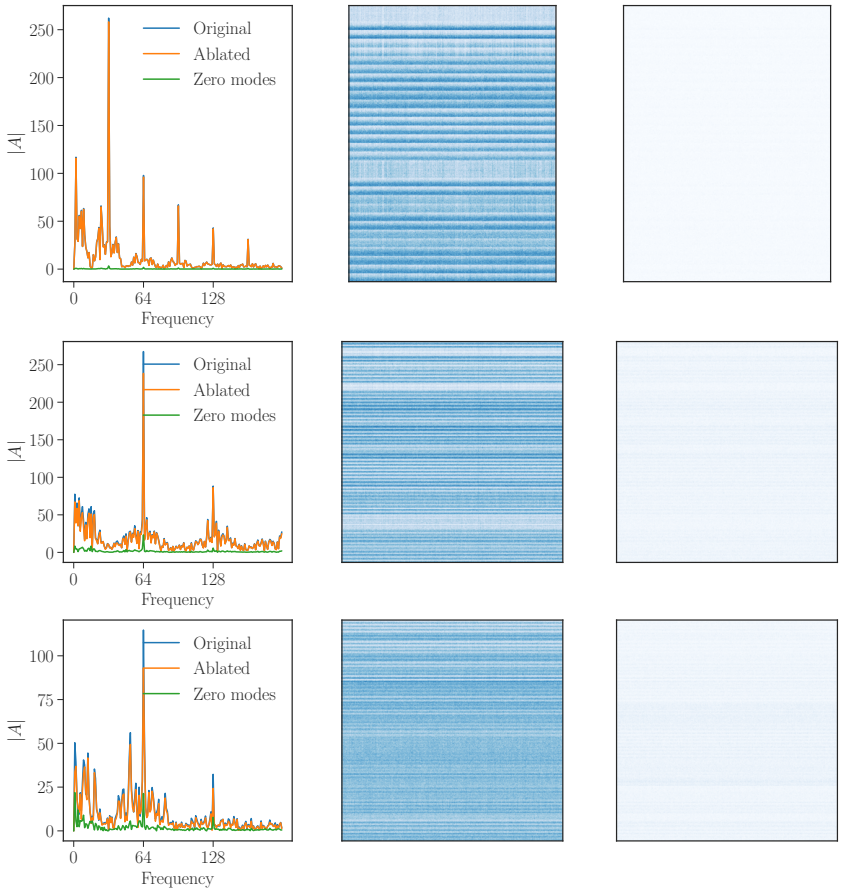

Figure 17: Distribution of weights (again visualized by converting the weights to a binary one as we did in toy models) for the W_Q matrices of **Top**: Gemma 7B layer 16, **Middle**: Llama 7B chat layer 2, and **Bottom**: Zephyr 7B β layer 22. In each panel the left is a fourier spectrum taken of the mean subtracted weights averaged over the second dimension. The middle and right are the binarized weights and zero modes, respectively.

		MT Bench (GPT4)		
Ablation	MMLU (5)	Turn 1	Turn 2	Average
None	59.78	7.09	6.58	6.84
with CT	-	7.68	6.15	7.13
Ablated layer 1	0.00	-	-	-
Ablated layer 1 random	59.14	7.23	6.10	6.67
Ablated layer 1 gate	0.00	-	-	-
Ablated layer 1 gate random	59.74	7.05	6.33	6.69
Ablated layer 1 gate with CT	-	5.86	6.06	5.95
Ablated layer 1 up	58.84	6.39	5.92	6.16
Ablated layer 1 down	0.00	-	-	-
Ablated complement layer 1	27.75	1.13	1.08	1.10
Ablated complement layer 1 gate	50.25	6.35	5.81	6.08
Ablated layers 16-31	44.72	2.01	1.37	1.69
Ablated layers 21-31	59.12	4.26	3.97	4.12
Ablated layers 21-31 random	59.79	6.76	6.2	6.48

Table 8: More detailed analysis of Zephyr 7B β tested on MMLU (five shot) and MT bench. CT = chat template. The dash on MMLU is because we did not perform the MMLU test with the chat template. The dashes for MT bench are there because when the MMLU score is exactly 0, the model cannot output anything useful, so the MT bench score will also be its minimum.

E.1 A list of Neurons and their firing pattern

Here we have gathered a small list of neurons we found in Zephyr 7B β that were most impacted by the ablation of zero modes. We looked at neurons whose activation changed most and which three neurons had their firing rate changed most. This gives us four neurons to look at for each layer, so a total of 128 neurons per comparison. These neurons are not necessarily all distinct, but does form a small selection of (as it turns out) interesting neurons.

The activations were obtained by sampling 100 context from RedPajama 1B sample and truncating the length to 128 tokens. This gives a decent set of activations to see some interesting patters, but we do not claim monosemanticity of some of the neurons, despite it being very suggestive. A more in depth analysis with many more contexts would need to be done to confidently conclude that. We see our method as exploratory and narrow down potentially interesting neurons for further study.

We have gathered a short list of neurons and a description of their activation pattern in Tab. 9.

F More results on LlaMA 2 7B chat case study

F.0.1 A list of Neurons and their firing pattern

Below we provide a short list of neurons that we found in the ablation process. We gather this data in the same was as for Zephyr 7B β, see Tab. 10

This list provides some insight into what types of neurons are most effected by the ablations, but we do not claim that their firing description as stated above is exhaustive. Some neurons that appear to be duplicates, could well be firing on some context that was not in our sample. In fact, neuron 9:3792 has a very clean activation on the BOS and `\n` or period, but it also fired on a piece of Swedish text.

G More results on Gemma 7B case study

Prompt used for Fig. 11:

```
Write a Python function that computes the n-th Fibonacci number.
```

Generated response of Gemma 7B for 80 tokens:

```
<bos>Write a Python function that computes the n-th Fibonacci number.\n\nThe n-th
    Fibonacci number can be computed as the sum of the (n-1)-th and the (n-2)-th
```

neuron	firing description
1:7310, 1:8572, 9:3244, 29:2017, 7:13558, 12:1068	BOS and first \n token.
3:9372	first token after BOS and first token of next newline/sentence.
6:4540, 12:1404, 16:12347	First token after BOS token.
16:11227, 24:11862	Period symbol or tokens with that symbol included, like '."'
10:4436	years, millennium, century, centuries, decade, digits representing years, ...
11:2372	Symbols like '(', ')', '$', '.', '"', '-', comma, etc.
13:360, 13:6430, 15:2538, 19:11788, 30:11995,	Everything but BOS and first \n token.
13:4054, 31:2187	BOS token.
16:4792, 18:10766	Words starting with 'p'.
17:4835	Tokens that start with a 'd' or 'D' or it fires on tokens that follow such a token.
17:14210	Words starting with 'm'.
18:9506, 19:7340	'in' plus few tokens after
18:10797	Conjugations of the verb 'to be'.
19:13571	Gerunds.
20:3086	Personal pronouns.
20:9931	Comma or tokens with a comma in it, like '),'.
22:10069	Possesive adjectives.
30:7579	First \n token.

Table 9: A set of neurons whose activations had the most absolute change or largest change in frequency of firing between the original and ablated model (in this case all zero modes were ablated). We added a short description on what tokens / concepts the specific neuron(s) fired in the second column.

neuron	firing description
0:7374	first token signifying end of sequence, like \n or a period.
1:7890, 2:1661, 4:2618, 9:3792, 26:868, 30:3721	BOS token and the end of a sentence, like \n or a period.
1:5994	BOS token
2:8665	Polysemantic, but dominant activation is on first token after BOS.
6:781	{we, We, We', our, your, me, my, I, I'} plus a few tokens that come after.
6:9458, 7:7431, 8:8862, 9:3209	Unrelated, but firing on the same and they are successive.
12:2521	Phrases that happen in the future, i.e. things like 'going to ...', 'will ...', 'would ...', etc.
13:10439	'it' or 'It' followed by a verb.
20:9315	Plural nouns or verbs.

Table 10: A set of neurons whose activations had the most absolute change or largest change in frequency of firing between the original and ablated model (in this case all zero modes were ablated). We added a short description on what tokens / concepts the specific neuron(s) fired in the second column.

```
    Fibonacci number. The first two Fibonacci numbers are 0 and 1.\n\n<code>>>> f(0)
    \n0\n>>> f(1)\n1\n>>> f(2)\n1\n>>> f(3)\n2\n>>> f
```

Generated response of Gemma 7B with the zero modes of down_{27} ablated, again for 80 tokens:

```
<bos>Write a Python function that computes the n-th Fibonacci number.\n\nAnswer:\n\
    ndef fibonacci(n): if n == 0: return fibonacci(1) elif n == fibonacci(n-1):
    return fibonacci(n-1)+fibonacci(n)\nelse exorbit exorbit exorbit exorbit
    exorbit exorbit exorbit exorbit exorbit exorbit exorbit exorbit exorbit exorbit
     exorbit exorbit exorbit exorbit exorbit exorbit exorbit exorbit exorbit
    exorbit exorbit exorbit exorbit exorbit exorbit exorbit exorbit exorbit exorbit
     exorbit exorbit exorbit exorbit exorbit
```

References

[1] Y. LeCun, J. Denker, and S. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems*, D. Touretzky, ed., vol. 2. Morgan-Kaufmann, 1989. https://proceedings.neurips.cc/paper_files/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems* **33** (2020) 1877–1901.

[3] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288* (2023) .

[4] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, *et al.*, "Mistral 7b," *arXiv preprint arXiv:2310.06825* (2023) .

[5] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635* (2018) .

[6] T. McGrath, M. Rahtz, J. Kramar, V. Mikulik, and S. Legg, "The hydra effect: Emergent self-repair in language model computations," *arXiv preprint arXiv:2307.15771* (2023) .

[7] A. Gromov, K. Tirumala, H. Shapourian, P. Glorioso, and D. A. Roberts, "The unreasonable ineffectiveness of the deeper layers," *arXiv preprint arXiv:2403.17887* (2024) .

[8] C. Olsson, N. Elhage, N. Nanda, N. Joseph, N. DasSarma, T. Henighan, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, D. Drain, D. Ganguli, Z. Hatfield-Dodds, D. Hernandez, S. Johnston, A. Jones, J. Kernion, L. Lovitt, K. Ndousse, D. Amodei, T. Brown, J. Clark, J. Kaplan, S. McCandlish, and C. Olah, "In-context learning and induction heads," *Transformer Circuits Thread* (2022) . https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

[9] N. Elhage, N. Nanda, C. Olsson, T. Henighan, N. Joseph, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, N. DasSarma, D. Drain, D. Ganguli, Z. Hatfield-Dodds, D. Hernandez, A. Jones, J. Kernion, L. Lovitt, K. Ndousse, D. Amodei, T. Brown, J. Clark, J. Kaplan, S. McCandlish, and C. Olah, "A mathematical framework for transformer circuits," *Transformer Circuits Thread* (2021) . https://transformer-circuits.pub/2021/framework/index.html.

[10] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, "Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned," *arXiv preprint arXiv:1905.09418* (2019) .

[11] T. Bricken, A. Templeton, J. Batson, B. Chen, A. Jermyn, T. Conerly, N. Turner, C. Anil, C. Denison, A. Askell, R. Lasenby, Y. Wu, S. Kravec, N. Schiefer, T. Maxwell, N. Joseph, Z. Hatfield-Dodds, A. Tamkin, K. Nguyen, B. McLean, J. E. Burke, T. Hume, S. Carter, T. Henighan, and C. Olah, "Towards monosemanticity: Decomposing language models with dictionary learning," *Transformer Circuits Thread* (2023) . https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[12] M. Sun, Z. Liu, A. Bair, and J. Z. Kolter, "A simple and effective pruning approach for large language models," *arXiv preprint arXiv:2306.11695* (2023) .

[13] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Advances in neural information processing systems* **28** (2015) .

[14] N. Lee, T. Ajanthan, and P. H. Torr, "Snip: Single-shot network pruning based on connection sensitivity," *arXiv preprint arXiv:1810.02340* (2018) .

[15] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale," *Advances in Neural Information Processing Systems* **35** (2022) 30318–30332.

[16] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," *arXiv preprint arXiv:2009.03300* (2020) .

[17] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, *et al.*, "Judging llm-as-a-judge with mt-bench and chatbot arena," *Advances in Neural Information Processing Systems* **36** (2024) .

[18] Z. Azerbayev, H. Schoelkopf, K. Paster, M. D. Santos, S. McAleer, A. Q. Jiang, J. Deng, S. Biderman, and S. Welleck, "Llemma: An open language model for mathematics," *arXiv preprint arXiv:2310.10631* (2023) .

[19] L. Tunstall, E. Beeching, N. Lambert, N. Rajani, K. Rasul, Y. Belkada, S. Huang, L. von Werra, C. Fourrier, N. Habib, *et al.*, "Zephyr: Direct distillation of lm alignment," *arXiv preprint arXiv:2310.16944* (2023) .

[20] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, *et al.*, "Gemma: Open models based on gemini research and technology," *arXiv preprint arXiv:2403.08295* (2024) .

[21] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv e-prints* (2019) , `arXiv:1910.10683`.

[22] T. Computer, "Redpajama: An open source recipe to reproduce llama training dataset," 2023. `https://github.com/togethercomputer/RedPajama-Data`.

[23] S. Mukherjee, A. Mitra, G. Jawahar, S. Agarwal, H. Palangi, and A. Awadallah, "Orca: Progressive learning from complex explanation traces of gpt-4," 2023.

[24] S. Longpre, L. Hou, T. Vu, A. Webson, H. W. Chung, Y. Tay, D. Zhou, Q. V. Le, B. Zoph, J. Wei, and A. Roberts, "The flan collection: Designing data and methods for effective instruction tuning," 2023.

[25] W. Gurnee, T. Horsley, Z. C. Guo, T. R. Kheirkhah, Q. Sun, W. Hathaway, N. Nanda, and D. Bertsimas, "Universal neurons in gpt2 language models," *arXiv preprint arXiv:2401.12181* (2024) .

[26] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi, "Winogrande: An adversarial winograd schema challenge at scale," *Communications of the ACM* **64** no. 9, (2021) 99–106.

[27] X. Ma, G. Fang, and X. Wang, "Llm-pruner: On the structural pruning of large language models," *Advances in neural information processing systems* **36** (2023) 21702–21720.

[28] N. Ding, Y. Chen, B. Xu, Y. Qin, Z. Zheng, S. Hu, Z. Liu, M. Sun, and B. Zhou, "Enhancing chat language models by scaling high-quality instructional conversations," 2023.

[29] E. Hubinger, C. Denison, J. Mu, M. Lambert, M. Tong, M. MacDiarmid, T. Lanham, D. M. Ziegler, T. Maxwell, N. Cheng, *et al.*, "Sleeper agents: Training deceptive llms that persist through safety training," *arXiv preprint arXiv:2401.05566* (2024) .