Neurons in Large Language Models: Dead, N-gram, Positional

Anonymous ACL submission

Abstract

We analyze a family of large language models in such a lightweight manner that can be done on a single GPU. Specifically, we focus on the OPT family of models ranging from 125m to 66b parameters and rely only on whether an FFN neuron is activated or not. First, we find that the early part of the network is sparse and represents many discrete features. Here, many neurons (more than 70% in some layers of the 66b model) are "dead", i.e. they never activate on a large collection of diverse data. At the same time, many of the alive neurons are reserved for discrete features and act as token and n-gram detectors. Interestingly, their corresponding FFN updates not only promote next token candidates as could be expected, but also explicitly focus on removing the information about triggering them tokens, i.e., current input. To the best of our knowledge, this is the first example of mechanisms specialized at removing (rather than adding) information from the residual stream. With scale, models become more sparse in a sense that they have more dead neurons and token detectors. Finally, some neurons are positional: them being activated or not depends largely (or solely) on position and less so (or not at all) on textual data. We find that smaller models have sets of neurons acting as position range indicators while larger models operate in a less explicit manner.

1 Introduction

The range of capabilities of language models expands with scale and at larger scales models become so strong and versatile that a single model can be integrated into various applications and decision-making processes (Brown et al., 2020; Kaplan et al., 2020; Wei et al., 2022; Ouyang et al., 2022; OpenAI, 2023; Anil et al., 2023). This increases interest and importance of understanding the internal workings of these large language models (LLMs) and, specifically, their evolution with scale. Unfortunately, scaling also increases the entry threshold for interpretability researchers since dealing with large models requires a lot of computational resources. In this work, we analyze a family of OPT models up to 66b parameters and deliberately keep our analysis very lightweight so that it could be done using a single GPU.

We focus on neurons inside FFNs, i.e. individual activations in the representation between the two linear layers of the Transformer feedforward blocks (FFNs). Differently from e.g. neurons in the residual stream, FFN neurons are more likely to represent meaningful features: the elementwise nonlinearity breaks the rotational invariance of this representation and encourages features to align with the basis dimensions (Elhage et al., 2021). When such a neuron is activated, it updates the residual stream by pulling out the corresponding row of the second FFN layer; when it is not activated, it does not update the residual stream (Figure 5).[1] Therefore, we can interpret functions of these FFN neurons in two ways: (i) by understanding when they are activated, and (ii) by interpreting the corresponding updates coming to the residual stream.

First, we find that in the first half of the network, many neurons are "dead", i.e. they never activate on a large collection of diverse data. Larger models are more sparse in this sense: e.g., in the 66b model more that 70% of the neurons in some layers are dead. At the same time, many of the alive neurons in this early network part are reserved for discrete features and act as indicator functions for tokens and n-grams: they activate if and only if the input is a certain token or an n-gram. The role of the updates coming from these token detectors to the residual stream is also surprising: at the same time as they promote next token candidates (which is to be expected according to Geva et al. (2021, 2022)), they are *explicitly targeted at removing information*

[1] Since OPT models have the ReLU activation function, the notion of "activated" or "not activated" is trivial and means non-zero vs zero.

about current input, i.e. their triggers. This means that in the bottom-up processing where a representation of the current input token gets gradually transformed into a representation for the next token, current token identity is removed by the model explicitly (rather than ends up implicitly "buried" as a result of additive updates useful for the next token). As far as we are aware, this is the first example of mechanisms specialized at removing (rather than adding) information from the residual stream.

Finally, we find that some neurons are responsible for encoding positional information regardless of textual patterns. Similarly to token and n-gram detectors, many of these neurons act as indicator functions of position ranges, i.e. activate for positions within certain ranges and do not activate otherwise. Interestingly, neurons often collaborate: their indicated positional ranges are often in agreement so that together they efficiently cover all possible positions and no neuron is redundant. In a broader picture, positional neurons question the key-value memory view of the FFN layers stating that "each key correlates with textual patterns in the training data and each value induces a distribution over the output vocabulary" (Geva et al., 2021, 2022). Neurons that rely on position regardless of textual pattern indicate that FFN layers can be used by the model in ways that *do not fit the key-value memory view*. Overall, we argue that the roles played by these layers are still poorly understood.

To sum up, we find neurons that:

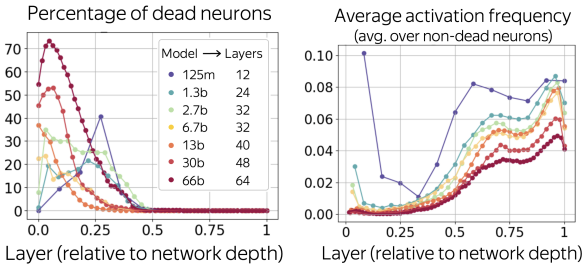
- are "dead", i.e. never activate on a large diverse collection of data;
- act as token- and n-gram detectors that, in addition to promoting next token candidates, explicitly remove current token information;
- encode position regardless of textual content which indicates that the role of FFN layers extends beyond the key-value memory view.

Larger models have more dead neurons and token detectors and are less focused on absolute position.

2 Data and Setting

Models. We use OPT (Zhang et al., 2022), a suite of decoder-only pre-trained transformers that are publicly available. We use model sizes ranging from 125M to 66B parameters and take model weights from the HuggingFace model hub.[2]

[2] https://huggingface.co/models



(a) (b)

Figure 1: (a) Percentage of "dead" neurons; (b) average neuron activation frequency among non-dead neurons.

Data. We use data from diverse sources containing development splits of the datasets used in OPT training as well as several additional datasets. Overall, we used (i) subsets of the validation and test part of the Pile (Gao et al., 2020) including Wikipedia, DM Mathematics, HackerNews, (ii) Reddit (Baumgartner et al., 2020; Roller et al., 2021), (iii) code data from Codeparrot[3].

For the experiments in Section 3 when talking about dead neurons, we use several times more data. Specifically, we add more data from Wikipedia, DM Mathematics and Codeparrot, as well as new domains from the Pile[4]: EuroParl, FreeLaw, PubMed abstracts, Stackexchange.

Overall, the data used in Section 3 has over 20M tokens, in the rest of the paper – over 5M tokens.

Single-GPU processing. We use only sets of neuron values for some data, i.e. we run only forward passes of the full model or its several first layers. Since large models do not fit in a single GPU, we load one layer at a time keeping the rest of the layers on CPU. This allows us to record neuron activations for large models: all the main experiments in this paper were done on a single GPU.

3 Dead Neurons

Let us start from simple statistics such as neuron activation frequency (Figure 1).

Many neurons are "dead". First, we find that many neurons never activate on our diverse data, i.e. they can be seen as "dead". Figure 1a shows that the proportion of dead neurons is very substantial: e.g., for the 66b model, the proportion of dead neurons in some layers is above 70%. We also see that larger models are more sparse because (i) they

[3] https://huggingface.co/datasets/codeparrot/codeparrot-clean

[4] https://huggingface.co/datasets/EleutherAI/pile

have more dead neurons and (ii) the ones that are alive activate less frequently (Figure 1b).

Only first half of the model is sparse. Next, we notice that this kind of sparsity is specific only to early layers. This leads to a clear distinction between the first and the second halves of the network: while the first half contains a solid proportion of dead neurons, the second half is fully "alive". Additionally, layers with most dead neurons are the ones where alive neurons activate most rarely.

Packing concepts into neurons. This difference in sparsity across layers might be explained by "concept-to-neuron" ratio being much smaller in the early layers than in the higher layers. Intuitively, the model has to represent sets of encoded in a layer concepts by "spreading" them across available neurons. In the early layers, encoded concepts are largely shallow and are likely to be discrete (e.g., lexical) while at the higher layers, networks learn high-level semantics and reasoning (Peters et al., 2018; Liu et al., 2019; Jawahar et al., 2019; Tenney et al., 2019; Geva et al., 2021). Since the number of possible shallow patterns is not large and, potentially, enumerable, in the early layers the model can (and, as we will see later, does) assign dedicated neurons to some features. The more neurons are available to the model, the easier it is to do so – this agrees with the results in Figure 1. Differently, the space of fine-grained semantic concepts is too large compared to the number of available neurons which makes it hard to reserve many dedicated neuron-concept pairs.[5]

Are dead neurons completely dead? Note that the results in Figure 1a can mean one of the two things: (i) these neurons can never be activated (i.e. they are "completely dead") or (ii) they correspond to patterns so rare that we never encountered them in our large diverse collection of data. While the latter is possible, note that this does not change the above discussion about sparsity and types of encoded concepts. On the contrary: it further supports the hypothesis that models assign dedicated neurons to specific concepts.

4 N-gram-Detecting Neurons

Now, let us look more closely into the patterns encoded in the lower model halves and try to understand the nature of the observed above sparsity. Specifically, we analyze how neuron activations depend on an input n-gram. For each input text with tokens $x_1, x_2, ..., x_S$, we record neuron activations at each position and if a neuron is activated (i.e., non-zero) at position k, we say that the n-gram $(x_{k-n+1}, \ldots, x_k)$ *triggered* this neuron.

In Sections 4.1-4.4 we talk about unigrams (i.e., tokens) and come to larger n-grams in Section 4.5.

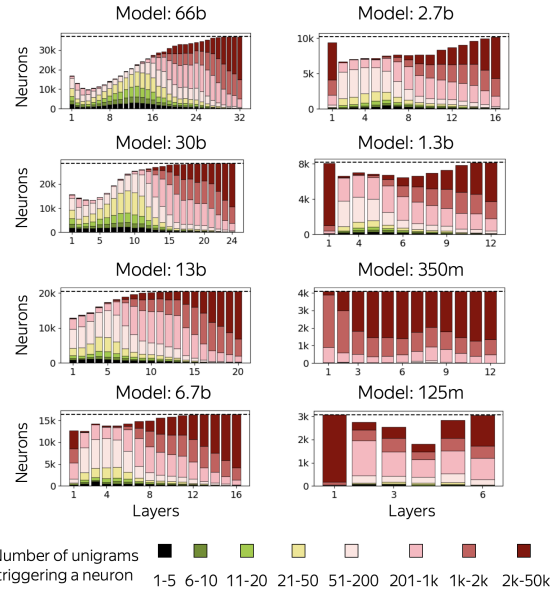


Figure 2: Neurons categorized by the number of unigrams (i.e., tokens) able to trigger them. First half of the network, alive neurons only.

4.1 Number of N-grams Triggering a Neuron

First, let us see how many n-grams are able to trigger each neuron. For each neuron, we evaluate the number of n-grams that cover at least 95% of its activations. For the bottom model half, Figure 2 shows how neurons in each layer are categorized by the number of covering them n-grams (we show unigrams here and larger n-grams in Appendix A).

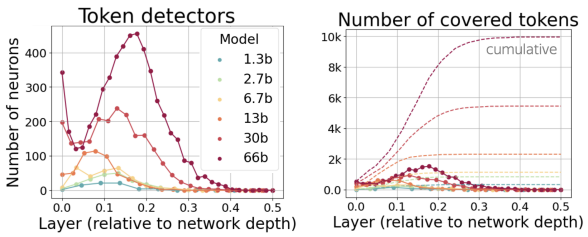
As anticipated, neurons in larger models are covered by fewer n-grams. Also, the largest models have a substantial proportion of neurons covered by as few as 1 to 5 tokens. This agrees with our hypothesis: the model spreads discreet shallow patterns across specifically dedicated neurons.[6]

4.2 Token-Detecting Neurons

Presence of neurons that can be triggered by only a few (e.g., 1-5) tokens point to the possibility that some neurons act as token detectors, i.e. activate

[5]There can, however, be a few specialized neurons in the higher layers. For example, BERT has neurons responsible for relational facts (Dai et al., 2022).

[6]Note that the 350m model does not follow the same pattern as all the rest: we will discuss this model in Section 6.



(a) (b)

Figure 3: (a) Number of token-detecting neurons; (b) number of tokens that have a detecting them neuron: solid line – per layer, dashed – cumulative over layers.

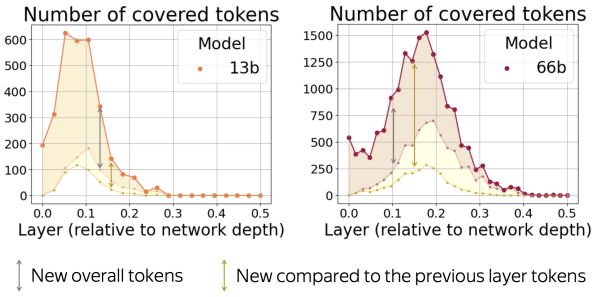


Figure 4: Number of tokens covered in each layer with indicated (i) new overall, and (ii) new compared to the previous layer tokens.

if and only if the input is one of the corresponding tokens, regardless of the previous context. To find such neurons, we (1) pick neurons that can be triggered by only 1-5 tokens, (2) gather tokens that are *covered* by this neuron (if the neuron activates at least 95% of the time the token is present), (3) if altogether, these covered tokens are responsible for at least 95% of neuron activations.[7]

Figure 3a shows that there are indeed a lot of token-detecting neurons. As expected, larger models have more of them and the 66b model has overall 5351 token detectors. Note that each token detector is responsible for a group of tokens that, in most of the cases, are variants of the same word (e.g., with differences only in capitalization, the space-before-word special symbol, morphological form, etc.). Figure 6 (top) shows examples of token groups detected by these neurons.

4.3 Ensemble-Like Behaviour of the Layers

Now, let us look at "detected" tokens, i.e. tokens that have a specialized detecting them neuron. Figure 3b shows the number of detected tokens in each layer and cumulatively over layers. We see that, e.g., the 66b model focuses on no more than 1.5k tokens in each layer but covers over 10k tokens overall. Thus, across layers, token-detecting neurons are responsible for largely differing tokens. Indeed, Figure 4 shows that in each following layer, detected tokens mostly differ from all the tokens covered below. This points to an ensemble-like (as opposed to sequential) behavior of the layers: layers collaborate in a divide-and-conquer-style manner which allows larger models to cover many tokens and use their capacity more effectively.

Originally, such an ensemble-like behavior of deep residual networks was found in computer vision (Veit et al., 2016). For transformers, previous evidence includes simple experiments showing that dropping or reordering layers does not hurt performance much (Fan et al., 2020; Zhao et al., 2021).

4.4 Token Detectors Suppress Their Triggers

Finally, let us explain the role of token-detecting neurons by interpreting how they update the residual stream. Throughout the layers, token representation in the residual stream gets transformed from the token embedding for the current input token[8] to the representation that encodes the next token. This happens via additive updates coming from attention and FFN blocks in each layer. Whenever an FFN neuron is activated, the corresponding row of the second FFN layer (multiplied by this neuron's value) is added to the residual stream (Figure 5). By projecting this FFN row onto vocabulary, we can interpret this update (and, thus, the role of this neuron) in terms of its influence on the output distribution encoded in the residual stream.

Current token suppression: implicit or explicit? Previously, this influence was understood only in terms of the top projections, i.e. tokens that are promoted (Geva et al., 2021, 2022). This reflects an existing view supporting implicit rather than explicit loss of the current token identity over the course of layers. Namely, the view that the current identity gets "buried" as a result of updates useful for the next token as opposed to being removed by the model explicitly. In contrast, we look not only at the top projections but also at the bottom: if these projections are negative, the corresponding tokens are suppressed by the model (Figure 5).

Explicit token suppression in the model. We find that often token-detecting neurons *deliberately suppress the tokens they detect*. Figure 6 shows

[7] We exclude the begin-of-sentence token from these computations because for many neurons, this token is responsible for the majority of the activations.

[8] For OPT, along with an absolute positional embedding.

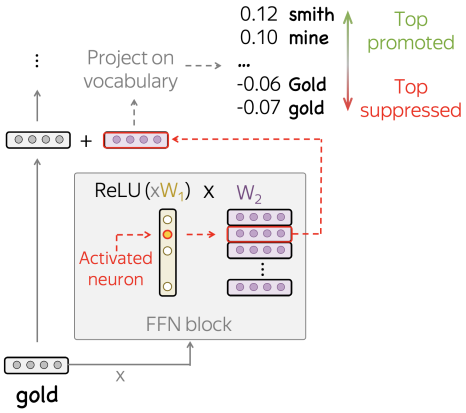


Figure 5: Intuition behind concept suppression: we look not only at the top projections of an FFN update on vocabulary but also at the bottom. The concepts that are added with a negative value are suppressed.

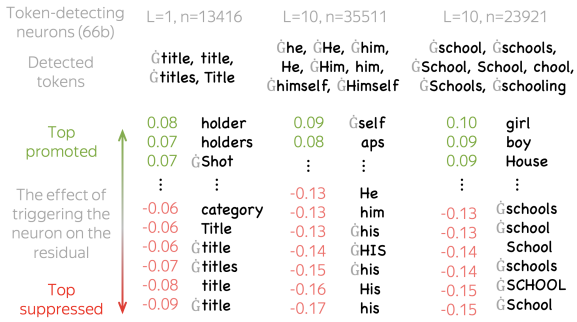


Figure 6: Examples of the top promoted and suppressed tokens for token-detecting neurons (Ġ is a special symbol denoting the space before word – in the OPT tokenizers, it is part of a word); OPT-66b model.

examples of these neurons along with the top promoted and suppressed concepts. While the top promoted concepts are in line with previous work (they are potential next token candidates as in Geva et al. (2021, 2022)), the top suppressed concepts are rather unexpected: they are exactly the tokens triggering this neuron. This means that vector updates coming from neurons play two different roles at the same time: (i) point in the direction of the next token candidates and (ii) point away from the tokens triggering the neuron. In total, for over 80% of token-detecting neurons their updates point in the negative direction from the triggering them tokens (although, the triggering tokens are not always at the very top suppressed concepts as in Figure 5).

To sum up, we show that models can have mechanisms targeted at removing information; future work can explore this further.

4.5 Beyond Unigrams

In Appendix A, we show results for bigrams and trigrams that mirror our observations for unigrams: (i) larger models have more specialized neurons, (ii) in each layer, models cover mostly new n-grams. Interestingly, for larger n-grams we see a more drastic gap between larger and smaller models.

5 Positional Neurons

When analyzing dead neurons (Section 3), we also noticed some neurons that, consistently across diverse data, never activate except for a few first token positions. This motivates us to look further into how position is encoded in the model and, specifically, whether some neurons are responsible for encoding positional information.

5.1 Identifying Positional Neurons

Intuitively, we want to find neurons whose activation patterns are defined by or, at least, strongly depend on token position. Formally, we identify neurons whose activations have high mutual information with position. For each neuron, we evaluate mutual information between two random variables:

- act – neuron is activated or not ($\{Y, N\}$),
- pos – token position ($\{1, 2, \ldots, T\}$).

Formal setting. We gather neuron activations for full-length data (i.e., $T = 2048$ tokens) for Wikipedia, DM Mathematics and Codeparrot. Let $fr_n^{(pos)}$ be activation frequency of neuron n at position pos and fr_n be the total activation frequency of this neuron. Then the desired mutual information is as follows:[9]

$$I(act, pos) = \frac{1}{T} \cdot \sum_{pos=1}^{T} \Bigg[fr_n^{(pos)} \cdot \log \frac{fr_n^{(pos)}}{fr_n} + (1 - fr_n^{(pos)}) \cdot \log \frac{1 - fr_n^{(pos)}}{1 - fr_n} \Bigg].$$

Choosing the neurons. We pick neurons with $I(act, pos) > 0.05$, i.e. high mutual information with position – this gives neurons whose activation frequency depends on position rather than content. Indeed, if e.g. a neuron is always activated within certain position range regardless of data domain, we can treat this neuron as responsible for position; at least, to a certain extent.

5.2 Types of Positional Neurons

After selecting positional neurons, we categorize them according to their activation pattern, i.e. activation frequency depending on position (Figure 7).

[9] For more details, see appendix B.1.

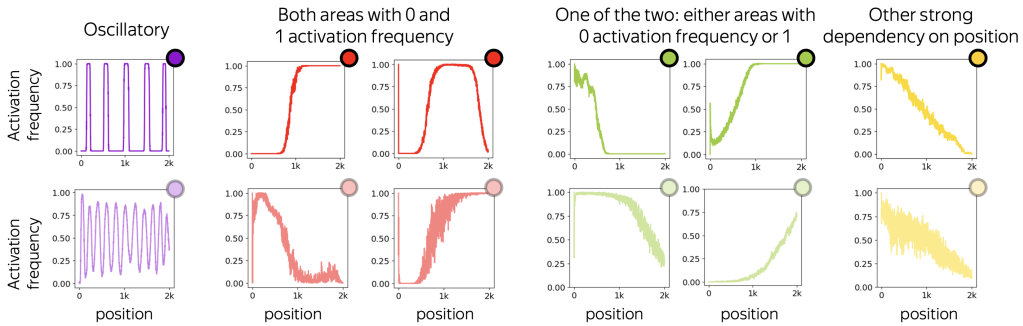


Figure 7: Types of positional neurons. Top row – "strong" pattern, bottom row – "weak" pattern.

Oscillatory. These neurons are shown in purple in Figure 7. When such a pattern is strong (top row), the activation pattern is an *indicator function* of position ranges: a neuron is activated if and only if the position falls into a certain set. Since the activation pattern does not change across data domains, it is defined solely by position and not the presence of some lexical or semantic information.

Both types of activation extremes. These are the neurons whose activation pattern is not oscillatory but still has intervals where activation frequency reaches both "activation extremes": 0 (never activated) and 1 (always activated). Most frequently, such a neuron is activated only for positions less than or greater than some value and not activated otherwise. Similarly to oscillatory neurons, when such a pattern is strong (Figure 7, top row), it is also (almost) an indicator function.

Only one type of activation extremes. Differently from the previous two types, activation patterns for these neurons can reach only one of the extreme values 0 or 1 (Figure 7, green). While this means that they never behave as indicator functions, there are position ranges where a neuron being activated or not depends solely on token position.

Other. Finally, these are the neurons whose activation patterns strongly depend on position but do not have intervals where activation frequency stays 0 or 1 (Figure 7, yellow). Typically, these activation patterns have lower mutual information with position than the previous three types.

Strong vs weak pattern. We distinguish "strong" and "weak" versions of each type and show this with color intensity (Figure 7, top vs bottom rows). For the first three neuron types, the difference between strong and weak patterns lies in whether on the corresponding position ranges activation frequency equals 0 (or 1) or close, but not equals, 0 (or 1). For the last type, this difference lies in how well we can predict activation frequency at some position knowing this value at the neighboring positions (informally, "thin" vs "thick" graph).

5.3 Positional Neurons Across the Models

For each of the models, Figure 8 illustrates the positional neurons across layers.

Small models encode position more explicitly. First, we notice that smaller models rely substantially on oscillatory neurons: this is the most frequent type of positional neurons for models smaller than 6.7b of parameters. In combination with many "red" neurons acting as indicator functions for wider position ranges, the model is able to derive token's absolute position rather accurately. In contrast, larger models do not have oscillatory neurons and rely on more generic patterns shown with red- and green-colored circles. We can also see that from 13b to 66b, the model loses two-sided red neurons and uses the one-sided green ones more. This hints at one of the qualitative differences between smaller and larger models: while the former encode absolute position more accurately, the latter ones are likely to rely on something more meaningful than absolute position. This complements recent work showing that absolute position encoding is harmful for length generalization in reasoning tasks (Kazemnejad et al., 2023). Differently from their experiments with same model size but various positional encodings, we track changes with scale. We see that, despite all models being trained with absolute positional encodings, stronger models tend to abstract away from absolute position.

Positional neurons collaborate. Interestingly, positional neurons seem to collaborate to cover the full set of positions. For example, let us look more closely at the 10 strongly oscillatory neurons in the second layer of the 125m model (Figure 8, dark

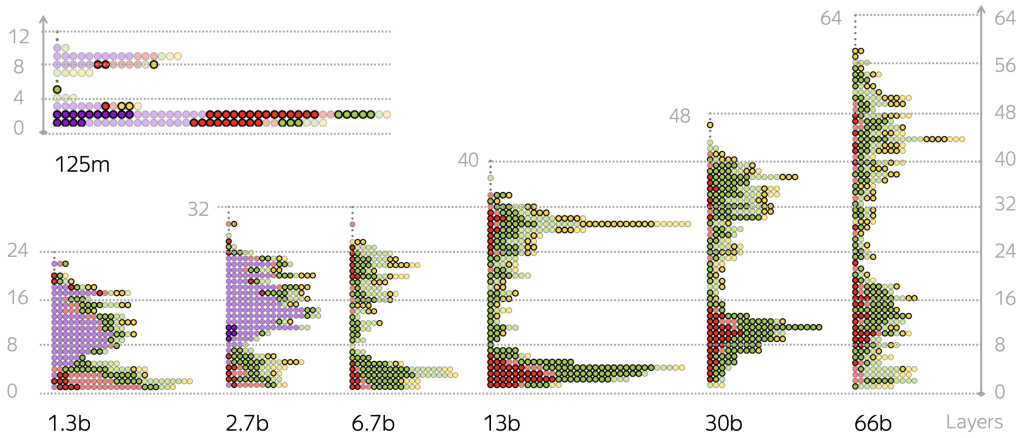


Figure 8: Positional neurons in each of the models. Each circle corresponds to a single neuron, colors and their intensity correspond to the types of patterns shown in Figure 7.

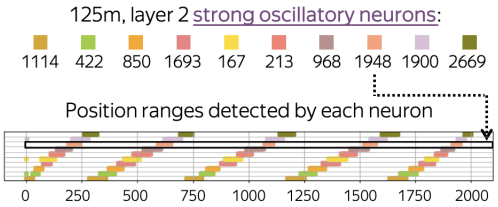


Figure 9: Position ranges indicated by strong oscillatory neurons in the second layer of the 125m model.

purple circles). Since they act as indicator functions, we can plot position ranges they indicate. Figure 9 shows that (i) indicated position ranges are similar up to a shift, (ii) the shifts are organized "perfectly": together, these ten neurons cover all positions and none of these neurons is redundant.

The two stages within the model. Finally, Figure 8 reveals two stages of up-and-downs of positional information within the model: roughly, the first third of the model and the rest. Interestingly, preferences in positional patterns also change between the stages: e.g., preference for "red" neurons changes to oscillatory purple patterns for the 1.3b and 2.7b models, and "red" patterns become less important in the upper stage for the 13b and 30b models. Note that the first third of the model corresponds to the sparse stage with the dead neurons and n-gram detectors (Sections 3, 4). Therefore, we can hypothesize that in these two stages, positional information is first used locally to detect shallow patterns and then more globally to use longer contexts and help encode semantic information.

Previously, the distinct bottom-up stages inside language models were observed in Voita et al. (2019a). The authors explained how the way representations gain and lose information across the layers is defined by the training objective and why, among other things, positional information should (and does) get lost. This agrees with our results: while there are many positional patterns in the second stage, they are weaker than in the first stage.

5.4 Positional Neurons are Learned Even Without Positional Encoding

Recently, it turned out that even without positional encoding, autoregressive LMs still learn positional information (Haviv et al., 2022). We hypothesize that these "NoPos" models encode position via positional neurons. To confirm this, we train 125m models with and without positional encodings and compare the types of their positional neurons.

Setup. We trained 125m models with the OPT setup but smaller training dataset: OpenWebText corpus (Gokaslan and Cohen, 2019), an open clone of the GPT-2 training data (Radford et al., 2019). This dataset contains 3B tokens (vs 180B for OPT).

Positional neurons without positional encoding. We see that, indeed, the model without positional encoding also has many strong positional patterns (Figure 10). Note, however, that the NoPos model does not have oscillatory neurons which, in combination with other positional neurons, allow encoding absolute position rather accurately. This means that the NoPos model relies on more generic patterns, e.g. "red" neurons encoding whether a position is greater/less than some value.

Oscillatory neurons require longer training. Finally, we find that oscillatory patterns appear only with long training. In Appendix B.3 we show positional patterns learned by the baseline 125m model trained for 50k, 150k and 300k steps and see that all models have very strong positional patterns, but only the last of them has oscillatory neurons.

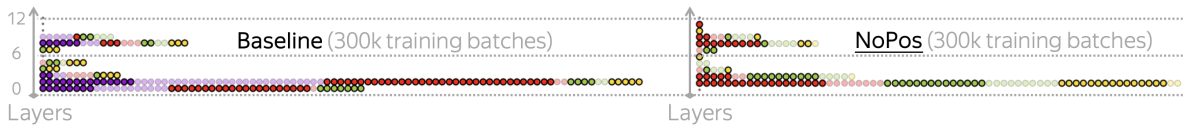


Figure 10: Positional neurons in 125m models: baseline vs model without positional encoding.

5.5 Doubting FFNs as Key-Value Memories

Currently, it is widely believed that FFNs in transformer-based language models operate as key-value memories. Specifically, "each key correlates with textual patterns in the training examples, and each value induces a distribution over the output vocabulary" (Geva et al. (2021, 2022); Dai et al. (2022); Meng et al. (2022), etc.). While in Section 4.4 we confirmed that this is true for some of the neurons, results in this section reveal that FFN layers can be used by the model in ways that *do not fit the key-value memory view*. Namely, activations of strong positional neurons are defined by position regardless of textual content, and the corresponding values do not seem to encode meaningful distributions over vocabulary – the role of these neurons is different from matching textual patterns to next token candidates. Therefore, the roles played by FFN layers are still poorly understood.

6 The 350m Model: The Odd One Out

Note that the 350m model does not follow the same pattern as the rest: it does not have dead neurons and its neuron activations do not seem to be sparse with respect to triggering them n-grams.[10] This might be explained by the difference in implementation: the 350m model applies LayerNorm after attention and FFN blocks, while all the other models – before.[11] Apparently, even minor implementation details can affect interpretability. Indeed, previous work also tried choosing certain modeling aspects to encourage interpretability, e.g. activation function (Elhage et al., 2022), sparse softmax variants (Martins and Astudillo (2016); Niculae and Blondel (2017); Peters et al. (2019); Correia et al. (2019); Martins et al. (2020)), or explicit modular structure (Andreas et al. (2016); Hu et al. (2018); Kirsch et al. (2018); Khot et al. (2021)).

7 Additional Related Work

Historically, neurons have been a basic unit of analysis. Early works started from convolutional networks first for images (Krizhevsky et al., 2012) and later for text classification (Jacovi et al., 2018). Similar to our work, Jacovi et al. (2018) also find n-gram detectors; although, for small convolutional text classifiers this is an almost trivial observation compared to Transformer-based LLMs. For recurrent networks, interpretable neurons include simple patterns such as line lengths, brackets and quotes (Karpathy et al., 2015), sentiment neuron (Radford et al., 2017), and others (Bau et al., 2019). For BERT, Dai et al. (2022) find that some FFN neurons store factual knowledge. Larger units of analysis include attention heads (Voita et al. (2018, 2019b); Clark et al. (2019); Kovaleva et al. (2019); Baan et al. (2019); Correia et al. (2019), etc), feed-forward layers (Geva et al., 2021, 2022) and circuits responsible for certain tasks (Wang et al., 2022; Geva et al., 2023; Hanna et al., 2023).

8 Implications and Conclusions

Overall, neurons in LLMs can (i) be dead (never-activating), (ii) act as token- and n-gram detectors, (iii) encode position regardless of textual content. Note that differently from most of the previous influential mechanistic interpretability work, we experiment with an entire family of models (instead of a single model as in Geva et al. (2021); Wang et al. (2022) among others) and consider very large models up to 66b. We also provide a way to analyze large models with very limited resources – we believe this is of high value to academic community. Finally, our main findings are not only about the presence of certain neurons in the OPT models but also about high-level conclusions regarding current beliefs in the community. Specifically, (1) information can be explicitly removed (rather than added) from the residual stream, (2) positional neurons question the key-value memory view of FFNs, (3) we explain how LMs trained without positional information still encode position, (4) we show that minor architecture changes can significantly influence interpretability, among others. On top of that, our analysis can easily be extended to other models: e.g., for other models later work validated our findings regarding positional neurons and suppressed concepts (Gurnee et al., 2024).

[10] There are, however, positional neurons (Appendix B.2).

[11] https://github.com/huggingface/transformers/blob/main/src/transformers/models/opt/modeling_opt.py

Limitations

The experiments in this paper are restricted to the OPT model family, which incorporate the ReLU activation function. As a result, the results may not be generalizable to alternative architectures and activation functions. The analysis itself, however, can be extended to other models in a way we explained above.

In our study, we define the term "neurons" to refer to the activations within the intermediate representation inside the feed-forward layers. While we provide rationale for focusing on these particular representations, it is worth noting that we don't explore other representations within the model.

Additionally, it is important to emphasize that the largest model used in this work is a 66B parameters model. Nevertheless, current Large Language Models exceed that parameter count, and may potentially exhibit different properties.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. 2023. Palm 2 technical report.

Joris Baan, Maartje ter Hoeve, Marlies van der Wees, Anne Schuth, and Maarten de Rijke. 2019. Understanding multi-head attention in abstractive summarization.

Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2019. Identifying and controlling important neurons in neural machine translation. In *International Conference on Learning Representations*, New Orleans.

Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. 2020. The pushshift reddit dataset.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. 2019. Adaptively sparse transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2174–2184, Hong Kong, China. Association for Computational Linguistics.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

Nelson Elhage, Tristan Hume, Catherine Olsson, Neel Nanda, Tom Henighan, Scott Johnston, Sheer ElShowk, Nicholas Joseph, Nova DasSarma, Ben Mann, Danny Hernandez, Amanda Askell, Kamal Ndousse, Jones, , Dawn Drain, Anna Chen, Yuntao Bai, Deep Ganguli, Liane Lovitt, Zac Hatfield-Dodds, Jackson Kernion, Tom Conerly, Shauna Kravec, Stanislav Fort, Saurav Kadavath, Josh Jacobson, Eli Tran-Johnson, Jared Kaplan, Jack Clark, Tom Brown, Sam McCandlish, Dario Amodei, and Christopher Olah. 2022. Softmax linear units. Https://transformer-circuits.pub/2022/solu/index.html.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The pile: An 800gb dataset of diverse text for language modeling.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus.

Wes Gurnee, Theo Horsley, Zifan Carl Guo, Tara Rezaei Kheirkhah, Qinyi Sun, Will Hathaway, Neel Nanda, and Dimitris Bertsimas. 2024. Universal neurons in gpt2 language models.

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2023. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model.

Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. 2022. Transformer language models without positional encodings still learn positional information. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1382–1390, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ronghang Hu, Jacob Andreas, Trevor Darrell, and Kate Saenko. 2018. Explainable neural computation via stack neural module networks. In *Proceedings of the European conference on computer vision (ECCV)*.

Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg. 2018. Understanding convolutional neural networks for text classification. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 56–65, Brussels, Belgium. Association for Computational Linguistics.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks.

Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Payel Das, and Siva Reddy. 2023. The impact of positional encoding on length generalization in transformers.

Tushar Khot, Daniel Khashabi, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2021. Text modular networks: Learning to decompose tasks in the language of existing models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1264–1279, Online. Association for Computational Linguistics.

Louis Kirsch, Julius Kunze, and David Barber. 2018. Modular networks: Learning to decompose neural computation. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*

and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4365–4374, Hong Kong, China. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

André F. T. Martins and Ramón F. Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1614–1623. JMLR.org.

Pedro Henrique Martins, Zita Marinho, and André F. T. Martins. 2020. Sparse text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4252–4273, Online. Association for Computational Linguistics.

Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*.

Vlad Niculae and Mathieu Blondel. 2017. A regularized framework for sparse and structured neural attention. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

OpenAI. 2023. Gpt-4 technical report.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Ben Peters, Vlad Niculae, and André F. T. Martins. 2019. Sparse sequence-to-sequence models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1504–1519, Florence, Italy. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y-Lan Boureau, and Jason Weston. 2021. Recipes for building an open-domain chatbot. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 300–325, Online. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Andreas Veit, Michael J Wilber, and Serge Belongie. 2016. Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019a. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4396–4406, Hong Kong, China. Association for Computational Linguistics.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-aware neural machine translation learns anaphora resolution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1264–1274, Melbourne, Australia. Association for Computational Linguistics.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019b. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2022. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models.

Sumu Zhao, Damian Pascual, Gino Brunner, and Roger Wattenhofer. 2021. Of non-linearity and commutativity in bert.

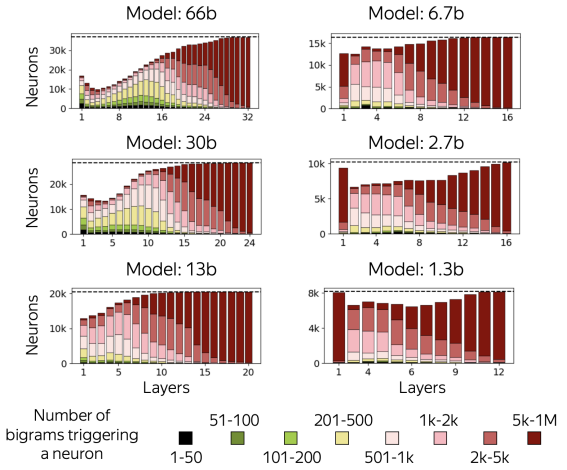


Figure 11: Neurons categorized by the number of bigrams able to trigger them. First half of the network, alive neurons only.

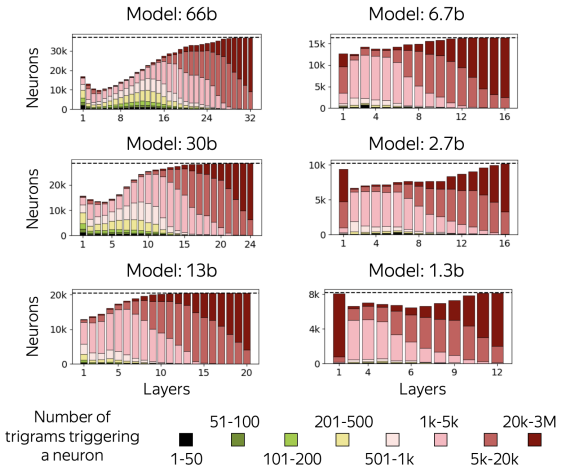


Figure 12: Neurons categorized by the number of trigrams able to trigger them. First half of the network, alive neurons only.

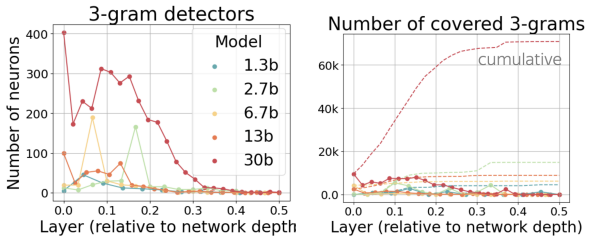
A N-gram-Detecting Neurons

A.1 Number of N-grams Triggering a Neuron

Figure 11 shows how neurons in each layer are categorized by the number of covering them bigrams, Figure 12 – trigrams. As expected, neurons in larger models are covered by less n-grams.

A.2 Trigram-Detecting Neurons

Similarly to token-detecting neurons in Section 4.2, we also find neurons that are specialized on 3-grams. Specifically, we (1) pick neurons that are covered by only 1-50 trigrams, (2) gather trigrams that are covered by this neuron (if the neuron activated at least 95% of the time the trigram is present), (3) if altogether, these covered trigrams are responsible for at least 95% of neuron activa-



(a) (b)

Figure 13: (a) Number of trigram-detecting neurons; (b) number of trigrams that have a detecting them neuron: solid line – per layer, dashed – cumulative over layers.

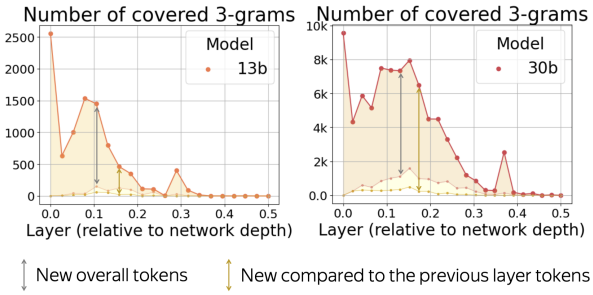


Figure 14: Number of trigrams covered in each layer with indicated (i) new overall, and (ii) new compared to the previous layer tokens.

tions.

Figure 13 shows the results. Overall, the results further support our main observations: larger models have more neurons responsible for n-grams. Interestingly, when looking at trigrams rather than tokens, at 30b of parameters we see a drastic jump in the number of covered n-grams. This indicates that one of the qualitative differences between larger and smaller models lies in the expansion of the families of features they are able to represent.

A.3 Ensemble-Like Layer Behavior

Figure 14 shows the number of covered trigrams in each layer. We see that in each layer, models cover largely new trigrams.

B Positional Neurons

B.1 Mutual Information

For each neuron, we evaluate mutual information between two random variables:

- act – neuron is activated or not ($\{Y, N\}$),
- pos – token position ($\{1, 2, \ldots, T\}$).

Formal setting. We gather neuron activations for full-length data (i.e., $T = 2048$ tokens) for Wikipedia, DM Mathematics and Codeparrot. Let $fr_n^{(pos)}$ be activation frequency of neuron n at position pos and fr_n be the total activation frequency of this neuron.

Then the desired mutual information is as follows:

$$I(act, pos) =$$

$$= \sum_{act} \sum_{pos=1}^{T} p(pos)p(act|pos) \cdot \log \frac{p(act|pos)}{p(act)} =$$

Since we only feed full-length texts, all positions appear with the same frequency: $p(pos) = 1/T$.

$$= \frac{1}{T} \cdot \sum_{act \in \{Y,N\}} \sum_{pos=1}^{T} p(act|pos) \cdot \log \frac{p(act|pos)}{p(act)} =$$

$$= \frac{1}{T} \cdot \sum_{pos=1}^{T} p(act = Y|pos) \cdot \log \frac{p(act = Y|pos)}{p(act = Y)} +$$

$$\frac{1}{T} \cdot \sum_{pos=1}^{T} (1 - p(act = Y|pos)) \cdot \log \frac{1 - p(act = Y|pos)}{1 - p(act = Y)} =$$

$$= \frac{1}{T} \cdot \sum_{pos=1}^{T} \left[fr_n^{(pos)} \cdot \log \frac{fr_n^{(pos)}}{fr_n} + \right.$$

$$\left. (1 - fr_n^{(pos)}) \cdot \log \frac{1 - fr_n^{(pos)}}{1 - fr_n} \right].$$

B.2 Positional Neurons for the 350m Model

The results are shown in Figure 15.

B.3 Oscillatory Neurons Require Longer Training

Figure 16 shows positional patterns learned by the baseline 125m model trained for 50k, 150k and 300k training batches. We see that all models have very strong positional patterns, but only the last of them has oscillatory neurons. Apparently, learning absolute position requires longer training time.

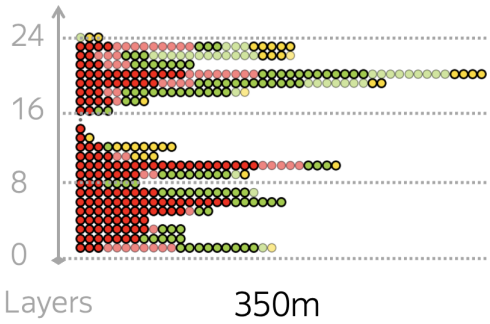


Figure 15: Positional neurons in the 350m model. Each circle corresponds to a single neuron, colors and their intensity correspond to the types of patterns shown in Figure 7.

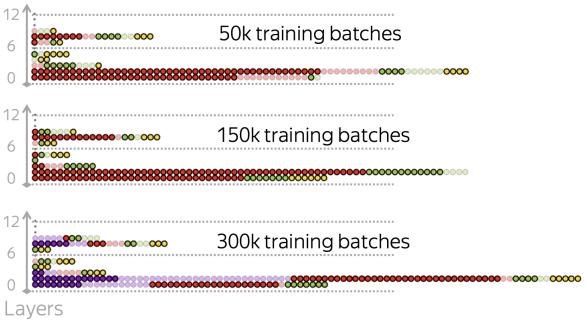


Figure 16: Positional neurons in the base 125m model trained with 50k, 150k and 300k batches.