23rd International Conference on Modeling & Applied Simulation, 001
21st International Multidisciplinary Modeling & Simulation Multiconference

2724-0037 © 2024 The Authors.
doi: 10.46354/i3m.2024.mas.001

Where are the arithmetic capabilities of a large language model located?

Andreas Stöckl[1,*] and Oliver Krauss[1]

[1]Digital Media Department, University of Applied Sciences Upper Austria, Softwarepark 13, Hagenberg, 4232, Austria

[*]Corresponding author. Email address: andreas.stoeckl@fh-hagenberg.at

Abstract

Language models are intended to process and generate text. In the extensive training process, however, they also develop arithmetic skills and the skills required to write programming code. In this work, we investigate whether it is possible to identify the areas in the neurons of these models responsible for a specific skill. For this purpose, we consider arithmetic tasks and let a language model solve them by completing and extracting the activation states of the neurons via synthetically generated datasets. We then try to reconstruct the results from individual groups of neurons using regression models to find the relevant groups for solving the tasks. Linear regression models, regression trees, and support vector regression are used to uncover possible relationships. We identify that neuron pairs, not individual neurons, in the LLM can be identified as responsible for specific arithmetic behavior. We also find that several distinct pairs of neurons in the GPT2 XL model are responsible for arithmetic capabilities, indicating a redundant encoding of these capabilities. In the future, this can lead to smaller models being extracted from larger ones for specific tasks.

Keywords: Large Language Models, Explainable AI, Decision Making

1. Introduction

We investigate in which layers and neurons the arithmetic capabilities of large language models are located. In the future, we hope to extend this work to find the programming capabilities of large language models (LLMs). Identifying these areas can improve the LLM's explainability. It could also allow direct neuron activation or extraction of neuron groups from an LLM for specialized tasks, such as modifying and repairing source code.

Such research is of fundamental importance in AI research, as explainability in AI servers as a basis for ethical and fair AI systems Dalal et al. (2023). It can also serve as a basis for pruning and regularization of LLMs Lehmler et al. (2023), leading to more efficient and specialized models.

For this purpose we try to answer the research question *Where are the arithmetic capabilities of a large language model located?*. For this we select the task of *counting* to find *where are sequential arithmetic capabilities located?*. We look at *addition* and the slightly more complex *multiplication* to ask *where are commutative and associative arithmetic capabilities located?*.

Language models generate text by continuing sequences of characters. They calculate the probability of the following characters from the statistical model they learned during model building and training. Today, calculating these probabilities is usually approximated with a neural network (Bengio et al., 2000).

Recently, the Transformer has established itself as the model architecture of choice (Vaswani et al., 2017). The method initially introduced for machine translation has almost completely replaced the previously predominant recursive architecture with long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997). Coupled with the idea of pretraining and subsequent fine-tuning (Howard and Ruder, 2018) (Radford et al., 2018), the ex-

cellent scalability of the architecture has led to the breakthrough of so-called large language models.

With the GPT2 model (Radford et al., 2019), it became increasingly clear that these models develop capabilities in the pretraining phase for which they were not explicitly trained but can still compete in many tasks with supervised, trained models created specifically for these tasks. Further scaling of model size and data volume led to more and more capabilities of the models (Brown et al., 2020), which were better adapted to questions and chat applications using fine-tuning techniques (Ouyang et al., 2022). Current models go beyond pure language models by integrating other modalities, such as images, into the pretraining (OpenAI, 2023). The explainability of these models is an active area of research, but many unanswered questions remain. An overview can be found in (Zhao et al., 2023).

This article investigates how healthy abilities acquired in the pretraining phase can be assigned to individual neurons or groups of neurons, so-called expert units. To this end, we investigate the arithmetic abilities of GPT2 models for which the model architecture and parameters are openly available after training. Starting from a synthetic dataset of arithmetic tasks, we analyze the activation of neurons during execution. Using different regression models, we analyze whether the results of the arithmetic tasks can be calculated from the activation values.

This publication is organized as follows. In section 2 we show the current state of the art of capability localization in LLMs. In section 3 explain how we generated our dataset to apply our own approach and explain the approach itself. Results are discussed in section 4. We draw conclusions in section 5 and give an outlook of future research directions.

2. State of the Art

The expert units in deep learning were first investigated in image analysis (Bau et al., 2017; Fong and Vedaldi, 2018). In large language models, discovering the "sentiment neuron" in creating a language model based on positive and negative product reviews was the first important milestone (Radford et al., 2017). A single neuron whose activation could control whether a generated review has a positive or negative sentiment was identified.

In (Suau et al., 2020) and (Cuadros et al., 2022), GPT2 models were also examined for the presence of linguistic concepts, such as the concept "bird" or others from WordNet (Miller, 1995), and based on classification tasks for the presence of the concept, it was decided which neurons are responsible for the concept. Typically, groups of 60 neurons were identified and activated by intervention in the neural network to condition the generation on a specific concept.

(Dalal et al., 2023) apply a symbolic reasoning approach, specifically Concept Induction, which was originally designed for the Semantic Web. They manage to attribute specific labels from image detection to individual neurons in a convolutional neuronal network.

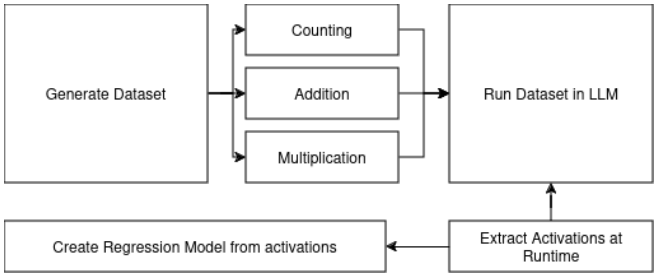

Figure 1. Process of our approach consisting of generating the dataset, extracting the activations at runtime and building a regression model

(Lehmler et al., 2023) concentrate on more complex tasks in image classification. Thus they do not look at individual neurons but rather activation patterns in the entire network by modelling networks as stochastic processes. By using neurscience techniques they extract spiking activity and use an arrival process following the Poisson distribution in a deep neural network.

When analyzing whether the truth value of a statement in a language model is represented in the internal states, a classification model for the truth value of statements was fitted in (Azaria and Mitchell, 2023) based on the activation values of different layers in the OPT-6.7B (Zhang et al., 2022) and LLAMA2-7b (Touvron et al., 2023b) models. The classification accuracy was tested on a data set created in the thesis with statements on cities, animals, inventions, companies, and scientific facts. A feed-forward network with three layers and (256, 128, 64) neurons was used as the regression model. For the OPT-6.7B model, the best value of 0.70 for layer 20 of 32 was achieved across all statements. For the LLAMA2-7b model from Meta's LLAMA series (Touvron et al., 2023a), almost 0.83 accuracy was achieved for layer 16 of 32. The other layers achieved lower values. The accuracy values differed significantly for the different types of statements. Overall, it can be seen that the middle layers or just below were the most suitable for reconstructing the truth values. The study did not attempt to localize the information on the truth of the statements in groups of individual neurons. Since even with a complex neural network as a classifier and the information from entire layers, only modest classification accuracy was achieved, finding a "truth neuron" seems problematic.

3. Materials and Methods

We generated a synthetic data set with arithmetic tasks as a first task of Figure 1. This dataset was then tested with the GPT2 models small and XL.

3.1. Generating the Data set

We construct a synthetic data set consisting of arithmetic tasks for the evaluation. To achieve this, we create text sequences that end with an arithmetic task to be completed by the model. Then, the model has to add a token that

generates the expected following number, the task result.

To ensure that we identify token activation independent of the token order or preceding text, we are using the Python package Loreipsum[1] to simulate a variable context by generating a sentence of varying length before each actual task that we want to solve. Some example sentences can be seen here:

- Sed modi velit etincidunt non numquam ut.
- Neque magnam dolorem voluptatem.

3.1.1. *Counting*

We investigate counting as an arithmetic operation. The model should provide the next token in a sequence of numbers. E.g. given the sequence "1 2 3" the expected result is "4". In this case, we also do not advise the model to complete the count.

For this, we generated a dataset consisting of 1000 counting exercises. Examples are:

- Quisquam magnam ipsum sit dolor. 1 2 3 4 5 6 → 7
- Velit ipsum sed voluptatem modi tempora tempora. 1 2 3 → 4
- Est velit aliquam dolore non velit neque. 1 2 3 4 5 6 7 8 9 10 11 → 12
- Sed modi velit etincidunt non numquam ut. 1 2 3 4 5 6 7 8 → 9

3.1.2. *Addition*

The second task is a simple addition of two real valued numbers. After the random context, we set the language model to simple addition tasks, also advising the model to conduct a calculation.

Examples of the test set, generated with 100 samples contain:

- Velit neque dolorem neque consectetur. Calculate 2 + 3 =
- Labore neque modi non. Calculate 4 + 5 =
- Dolorem tempora aliquam ipsum. Calculate 7 + 1 =

3.1.3. *Multiplication*

The third task is a simple multiplication of two real-valued numbers. After the random context, we set the language model to simple multiplication tasks with small numbers below 10.

Examples from the test set of 100 samples contain:

- Etincidunt est tempora dolorem tempora. Calculate 5 ∗ 6 =
- Sit adipisci non aliquam voluptatem. Calculate 3 ∗ 3 =
- Tempora adipisci dolorem velit porro. Calculate 7 ∗ 2 =

3.2. Implementation of the Model

We examine the GPT2 models available via Huggingface and the Transformer library (Wolf et al., 2020). We load the weight using the NanoGPT implementation [2] based on Pytorch (Paszke et al., 2019). The experiments were taken with the temperature set to 0.8 and top-K sampling (Fan et al., 2018) to 1.

To extract the activations of the neurons in the LLM at inference time, we use TorchLens (Taylor and Kriegeskorte, 2023).

We start with the small model with 124 million parameters, for which we only test counting. We inspect several linear layers of the transformer model and store the activation values at each generation process.

For the counting example, we obtain 1000 vectors and target values from the set 4, 5, ..., 12, 13. The small model can consistently conduct the counting operation, so we did not extend the experiment to the XL model for counting.

We use the XL model with 1.5 billion parameters for the addition and multiplication tasks and obtain 100 vectors and target values from the set 3, 4, 5, ... In addition to multiplication, the XL model had to be used because the small GPT2 model cannot consistently perform these two operations with successful outcomes.

We implement different regression models with the Sci-Kit Learn (Pedregosa et al., 2011) package to predict the results from the activation vectors. For this purpose, the regression model is trained with 750 examples, tested with 250 in the counting case, trained with 75 examples, and tested with 25 in the addition case. Besides linear regression, we try regression trees (Loh, 2011) and support vector regression (Awad et al., 2015).

4. Results and Discussion

We present the results tested on GPT2 small, with a vector of 1000 samples for counting and GPT2 XL for addition and multiplication with 100 samples each.

4.1. Counting

A linear regression model with all 50257 features from the last linear layer can reproduce the labels with a R^2 value of over 0.98. This is unsurprising since the last layer contains all the neural network information.

Suppose we try the same with activating the last linear layer of the attention blocks with 768 activations. In that case, a linear regression model cannot reconstruct the label with a reasonable R^2 value.

In the last linear layer, we now try to determine whether individual neurons (groups) can predict the result using linear regression. First, we try to predict the counting result with each 50257 neuron from the activation values. However, no R^2 values above 0.17 can be achieved, and therefore, no helpful prediction can be made.

Now, we want to find out whether there are pairs that can be used to infer the result with linear regression. Since the number of pairs with over a billion possibilities does

1 https://pypi.org/project/lorem-text/)

2 https://github.com/karpathy/nanoGPT

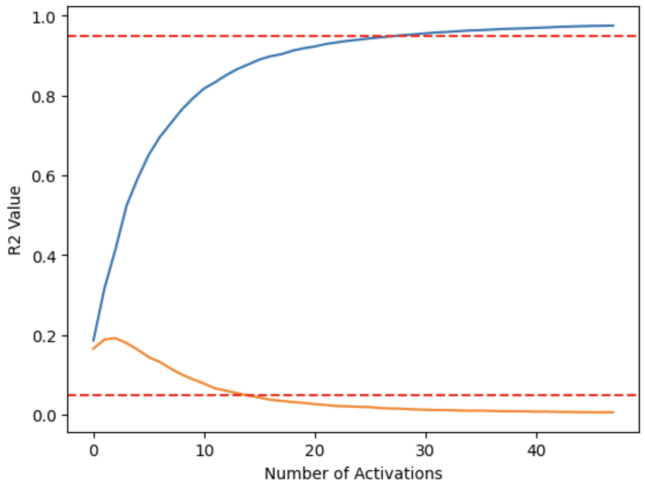

Figure 2. Mean and standard deviation of the R^2 value of 1000 random neuron groups over the size of the groups for counting.

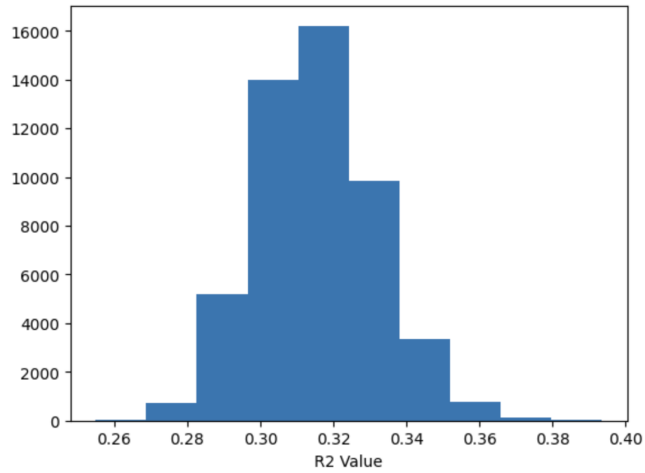

Figure 3. R^2 of single neurons for addition task with GPT2 small.

not allow us to try out all possibilities, we first try to find pairs using random selection. Some pairs that lead to R^2 values above 0.90 can be found.

In 500,000 randomly selected neuron pairs, 12 pairs with R^2 values above 0.87 can be found. The best one at an R^2 of 0.91.

We have also attempted to solve the optimization problem systematically using heuristic optimization methods. However, attempts with simulated annealing (Rutenbar, 1989) and genetic optimization (Srinivas and Patnaik, 1994) with different hyperparameters did not yield good results.

We repeat the same experiments with groups of 3 neurons. These only lead to insignificantly better R^2 values than with two neurons—the best one at an R^2 of 0.93. The ability to continue a sequence of numbers thus appears to be concentrated in neuron pairs.

We tested the same with regression trees (Loh, 2011) and support vector regression (Awad et al., 2015) with similar results but much longer computing time. So, we worked with linear regression for the rest.

Therefore, the information the language model uses for counting can already be found in individual neuron pairs—not just a single one, but several. The information, therefore, appears to be redundant in the language model.

To test how many neurons are needed in the mean to predict the result, we sample 1000 neuron groups of size k ranging from 1 to 50 and calculate the mean of R^2 over the 1000 samples. Figure 2 shows the mean and the standard deviation of the 1000 samples.

4.2. Addition

We look at the 100 addition tasks with the small model and try linear regression with individual neurons. For single neurons, the best value we achieve here is 0.39. Figure 3 shows the distribution of values for all individual neurons.

Looking at the generated responses, it becomes clear why the prediction from the activations cannot work. The calculations that the small model makes are usually wrong.

In contrast to simple counting, the small GPT2 model with 124 million parameters could not perform the additions correctly. The same applies to the medium (345 million parameters) and large version (774 million parameters) of GPT2. However, the XL version, with over 1.5 million parameters, has achieved this capability. But only for small numbers below 10. This is an example of the "emerging capabilities" of language models, as discussed in (Wei et al., 2022).

Therefore, analogous to the previous counting tasks, we try to find whether single neurons or neuron groups responsible for the calculations can be found. To achieve this, we again use linear regression and the data set created in section 3.1.

We check the ability of all individual neurons to predict the result of the addition tasks. The neuron with the best value provides an R^2 of 0.966, but it is not the only one with high values. Figure 4 shows the distribution of R^2 values across all neurons.

Since we can choose from a large number (50257) of characteristics for the linear regression, could it be that such a high R^2 is only achieved by pure chance?

To check this, we create a matrix of dimension (50257,100) with random numbers between 0 and 1 and calculate the linear regression with each column of this matrix against the labels of our addition task.

The maximum R^2 we achieve in this way is 0.26, and the distribution of all values can be seen in Figure 5.

Therefore, the good predicted values of the linear regression cannot have arisen by chance. There must, therefore, be information about the additions to the activations.

4.3. Multiplication

We now test the XL model with multiplication tasks from section 3.1. Linear regression with single neurons yields

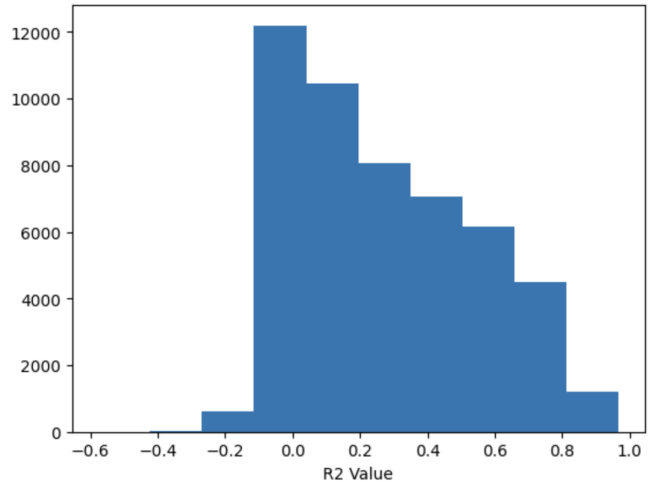

Figure 4. Distribution of R^2 values of the single neuron predictions with GPT2 XL.

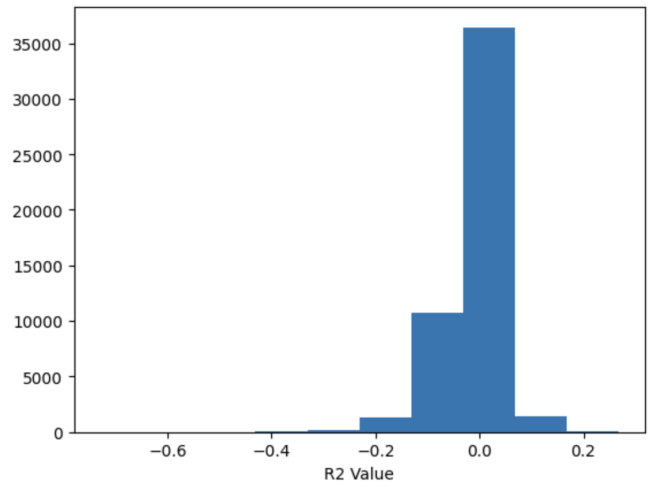

Figure 5. Distribution of R^2 values of regression with a random matrix using GPT2 XL

a value of R^2 = 0.89 as the best value, and Figure 6 shows the distribution of R^2 values across all neurons. These are slightly worse than the addition tasks. More errors occur when generating the multiplication tasks. The XL model does not master the multiplication tables without errors. This lack of capabilities can be seen when you compare Figure 6 with Figure 4. For groups of 2 neurons, an insignificantly better value of 0.93 was found.

5. Conclusions

To answer *Where are the arithmetic capabilities of a large language model located?*, while in the work of (Radford et al., 2017), a single neuron was responsible for the sentiment of the text, and (Cuadros et al., 2022) identified groups of about 60 neurons for linguistic concepts such as "bird," this study indicated that simple arithmetic skills

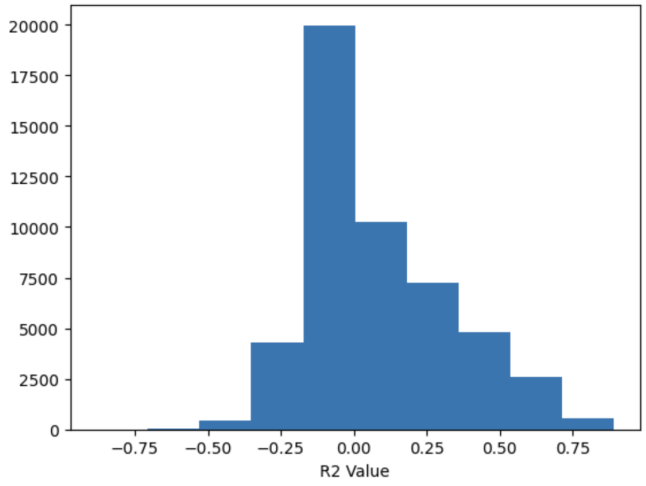

Figure 6. R^2 of single neurons for multiplication task

such as counting, addition, and multiplication could be reconstructed from the activation of a single neuron or two neurons of the last two layers with high precision. But only if the model is large enough for its capabilities. So, the small GPT2 model can do counting but cannot do calculations, but the XL version can.

While the answer to *where are sequential arithmetic capabilities located?* was that several pairs of redundant neurons were responsible, for *where are commutatitve and associative arithmetic capabilities located?* the answer is the same, but is only reflected in larger language models.

In the last two layers of the Transformer Model, not only can one specific neuron predict the result with high accuracy, but many neurons have that capability. That differs from the sentiment neuron's findings in (Radford et al., 2017). The information for the capabilities is stored redundantly in the network.

To be able to test the other arithmetic abilities, it makes sense to switch to more powerful language models with more parameters (Touvron et al., 2023a), (Jiang et al., 2023), (Penedo et al., 2023), as these have also developed more abilities. It would be exciting to investigate whether groups with a few neurons can also be found for other arithmetic abilities, like calculating with larger numbers or solving equations.

It would also be interesting to observe how the abilities develop and can be localized during training. It is expected that neurons representing the abilities can only be identified at an advanced stage of the process. For this purpose, checkpoints should be saved during training and used to conduct regression tests, as in this work.

In the future it would be interesting to extend this approach beyond arithmethic capabilities, for example specific tasks in image processing, such as segmentation Praschl (2021).

References

Awad, M., Khanna, R., Awad, M., and Khanna, R. (2015). Support vector regression. *Efficient learning machines: Theories, concepts, and applications for engineers and system designers*, pages 67–80.

Azaria, A. and Mitchell, T. (2023). The internal state of an llm knows when its lying. *arXiv preprint arXiv:2304.13734*.

Bau, D., Zhou, B., Khosla, A., Oliva, A., and Torralba, A. (2017). Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549.

Bengio, Y., Ducharme, R., and Vincent, P. (2000). A neural probabilistic language model. *Advances in neural information processing systems*, 13.

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

Cuadros, X. S., Zappella, L., and Apostoloff, N. (2022). Self-conditioning pre-trained language models. In *International Conference on Machine Learning*, pages 4455–4473. PMLR.

Dalal, A., Sarker, M. K., Barua, A., Vasserman, E., and Hitzler, P. (2023). Understanding cnn hidden neuron activations using structured background knowledge and deductive reasoning.

Fan, A., Lewis, M., and Dauphin, Y. (2018). Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*.

Fong, R. and Vedaldi, A. (2018). Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8730–8738.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

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

Lehmler, S. J., ur Rehman, M. S., Glasmachers, T., and Iossifidis, I. (2023). Understanding activation patterns in artificial neural networks by exploring stochastic processes.

Loh, W.-Y. (2011). Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(1):14–23.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

OpenAI (2023). Gpt-4 technical report.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Penedo, G., Malartic, Q., Hesslow, D., Cojocaru, R., Cappelli, A., Alobeidli, H., Pannier, B., Almazrouei, E., and Launay, J. (2023). The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*.

Praschl, C. (2021). Multi-resolution localization of individual logs in wooden piles utilizing yolo with tiling on client/server architectures. In *Proceedings of the 33rd European Modeling amp; Simulation Symposium*, EMSS 2021. CAL-TEK srl.

Radford, A., Jozefowicz, R., and Sutskever, I. (2017). Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Rutenbar, R. A. (1989). Simulated annealing algorithms: An overview. *IEEE Circuits and Devices magazine*, 5(1):19–26.

Srinivas, M. and Patnaik, L. M. (1994). Genetic algorithms: A survey. *computer*, 27(6):17–26.

Suau, X., Zappella, L., and Apostoloff, N. (2020). Finding experts in transformer models. *arXiv preprint arXiv:2005.07647*.

Taylor, J. and Kriegeskorte, N. (2023). Extracting and visualizing hidden activations and computational graphs of pytorch models with torchlens. *Scientific Reports*, 13(1):14375.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023a). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

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

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B.,

Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. (2022). Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. (2022). Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., Wang, S., Yin, D., and Du, M. (2023). Explainability for large language models: A survey. *arXiv preprint arXiv:2309.01029*.