# Exploring attention output similarity for duplication pruning of BLOOM and OPT

**Alan Ispani**
alai@itu.dk

**Johan Laursen**
jocl@itu.dk

## Abstract

In this thesis, we investigate the effects of structural pruning using attention output similarity for the BLOOM and OPT large language models (LLMs). Our hypothesis is that attention heads showing nearly identical attention patterns can be pruned without any notable drop in model accuracy. This hypothesis is examined by using two different kinds of pruning strategies: pruning by zeroing out (masking) a head's weights and pruning by duplication of similar heads. We show that duplication exhibits worse performance than masking weights for all experiments. Additionally, our results show that pruning based on the similarity of attention outputs does not outperform other current structural pruning methods at pruning percentages lower than 50%. However, our method outperforms the state of the art at 75% pruning on OPT. Code can be found at https://github.com/johanlaursen/explainabloomity

## 1 Introduction

In 2018, ~1-2% of worldwide electricity was used by computers. Just two years later in 2020, this number had jumped to roughly 4-6%, highlighting an extreme increase in energy consumption [1]. Over the past few years, competition for building and training bigger and more complex AI models - particularly LLMs - has intensified. With it, the rush towards obtaining large quantities of Graphics Processing Units (GPUs) is also increasing[2]. A study by Villalobos et al. (2022) about the changes in model sizes from 1950 to 2022 shows that between 1950 and 2018, models grew slowly and steadily. However, 2018 to 2022 was a turning point where model sizes began to expand at an increasing rate which greatly surpassed the growth witnessed during the previous seven decades. This trend is worrying, as the energy requirements and carbon footprint related to the extensive use of GPUs has significant environmental costs (Weidinger et al., 2022). However, there is potential to mitigate these impacts through various techniques to improve efficiency without significantly sacrificing performance (Treviso et al., 2023). This work focuses on structural pruning, a technique that effectively reduces the size of AI models by removing components that contribute the least to their performance.

In this thesis, we examine the efficacy of attention-based pruning in reducing model size without significantly reducing performance. We explore various ways to identify similar and superfluous attention heads in the OPT (Zhang et al., 2022) and BLOOM (Le Scao et al., 2023) language models, and examine how different similarity metrics for clustering attention heads, i.e. Euclidean distance and cosine similarity, affect the process of pruning. Furthermore, we examine the performance of two pruning methods: masking, which sets weights of similar attention heads to zero, and duplication, which replaces the weights of a head with those of a different, similar head. Our research questions are: *Can pruning attention heads based on similarity of attention output in large language models reduce their size without compromising performance? Additionally, can this similarity be used to improve performance by duplicating attention heads instead of masking them?*

## 2 Background

### 2.1 Transformers

Transformers (Vaswani et al., 2017) have contributed to the rapid growth of the field of natural language processing and are the backbone of large language models. LLMs are designed to understand and generate human language by learning

[1] https://penntoday.upenn.edu/news/hidden-costs-ai-impending-energy-and-resource-strain

[2] https://www.nytimes.com/2023/08/16/technology/ai-gpu-chips-shortage.html

statistical patterns from large amounts of text data, allowing them to perform tasks such as text completion, translation, summarization, and question answering. Transformers use a novel mechanism known as self-attention, which weighs the importance of different words in a text relative to each other and is able to capture word dependencies over long context windows. This self-attention mechanism is computed within so-called *attention heads*, each of which independently attends to different parts of the input sequence.

### 2.1.1 Self-attention

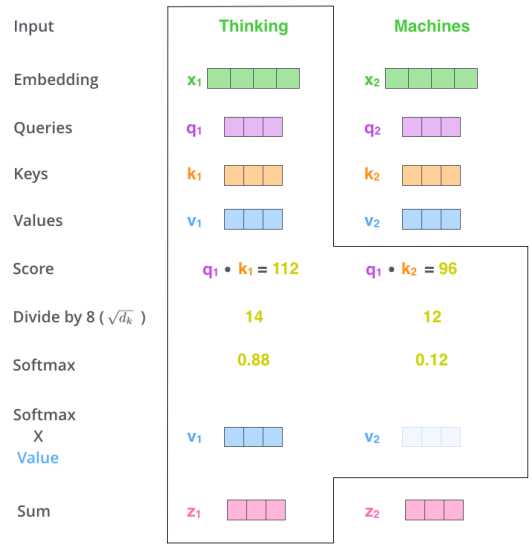



Figure 1: Self attention visualisation of a single attention head for the input "Thinking Machines". Source: Alammar (2018)

Figure 1 illustrates the steps necessary to compute attention for a single attention head. Initially, each word in the input sequence is converted into a vector using an embedding layer, combined with positional encodings to retain word order information. For each word, three vectors are computed through learned linear transformations: query (Q), key (K), and value (V) vectors, all derived from the input embeddings and sharing the same dimensionality. The attention score for a pair of words is calculated by taking the dot product of the query vector of one word with the key vector of another. These scores are then scaled by the square root of the key vectors' dimensionality to stabilise gradients during training. The attention scores are passed through a softmax function to convert them into probabilities, indicating the relative importance of each word in the context of the query word. Each word's value vector is then multiplied by these attention probabilities, and the results are summed to produce a new representation for each word, allowing the model to focus on relevant parts of the input when generating an output.

In practice, all the attention computations for a given input are calculated simultaneously via matrix multiplication for efficiency. The matrix of outputs is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

To summarise, self-attention updates the vector representations of the input words, which allows the model to capture the relationships between words over long context windows.

### 2.1.2 Multi-Headed Attention

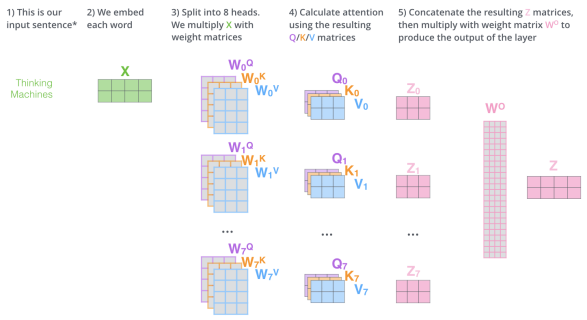



Figure 2: From self-attention to multi-headed attention. Source: Alammar (2018)

A single self-attention operation happens within one attention head, but these models are designed to capture many types of relationships, so multiple self-attention operations are run in parallel in a process called multi-headed attention. Each head in a multi-headed attention block has its own set of query, key, and value vectors, as can be seen in Figure 2. The input embeddings are projected into multiple sets of these vectors, one set per head, with each head performing the self-attention mechanism independently. For example, in a sentence like "The quick brown fox jumps over the lazy dog", one attention head might focus on the relationship between "quick" and "fox," while another head might focus on the relationship between "fox" and "jumps," allowing the model to understand both adjective-noun and noun-verb relationships simultaneously. The outputs from all attention heads are then concatenated and passed through a Feed-Forward Neural Network (FFNN), combining the different perspectives learned by each head

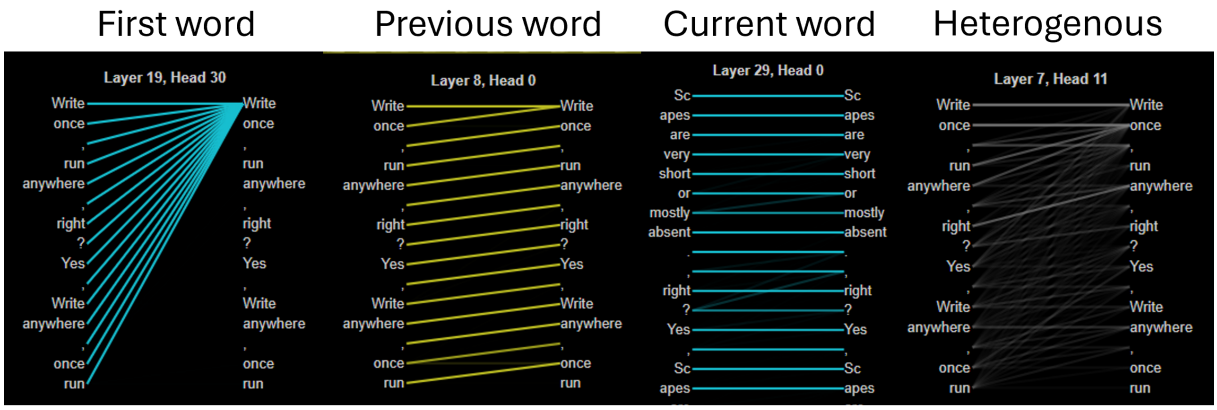



Figure 3: Examples of most common attention patterns in the BLOOM and OPT models found by qualitative analysis of different prompts taken from our benchmarks. A line connecting two tokens means that the token on the left is attending to the one on the right and the strength of the line is the magnitude of the attention. Visualised using BertViz (Vig, 2019)

.

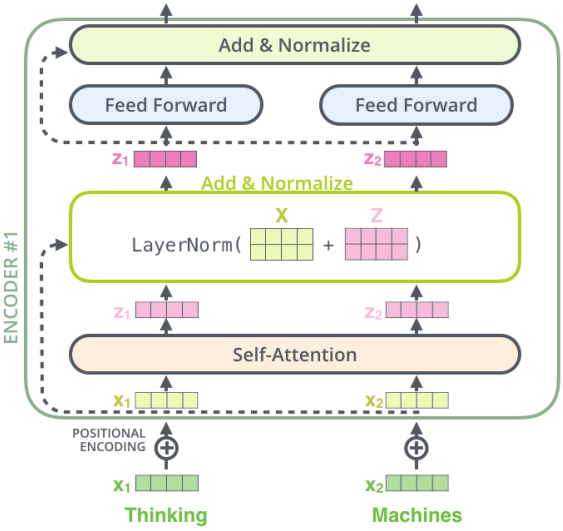



Figure 4: Residual connection (dashed arrow) from self-attention input to self-attention output. Source: Alammar (2018)

into one unified representation. This combination of multi-headed attention and FFNN constitutes a Transformer block. LLMs typically have many of these blocks stacked into layers, so the output of one block becomes the input of the next.

One important feature of Transformers is the existence of skip connections between parts of the Transformer block. This results in a so-called *residual stream* (Elhage et al., 2021). The residual stream is simply the sum of the output of all the previous layers and the original embedding. This means that the input of each component is directly added to its output, allowing the information to bypass the component's processing while still being included in the overall computation. It is the sum of the output of all the previous layers and the original embedding. The residual stream can be thought of as a communication channel since it doesn't do any processing itself and all layers communicate through it. The residual stream has a linear structure: every layer performs an arbitrary linear transformation to "read in" information from the residual stream at the start, and performs another arbitrary linear transformation before adding to "write" its output back into the residual stream. The dashed arrow in Figure 4 shows this process.

Elhage et al. (2021) show that attention heads operate completely in parallel and each separately add their output into the residual stream. This independent and additive property of attention heads has implications for our pruning methodology, which is explained in section 3.

### 2.1.3 Attention Maps

The result of the softmax calculation in Equation 1 before multiplying with the Value (V) matrix represents the extent to which each token attends to every other token in the input sequence. It can be visualised in an attention map as can be seen in Figure 3.

Many existing approaches try to explain transformer language models by analysing the knowledge encoded in the attention maps (Vig, 2019; Park et al., 2019; Hoover et al., 2020; Jaunet et al., 2021). However, there has been debate over the validity of attention-based explanations, with Brunner et al. (2020) and Bai et al. (2021) showing that raw attention contains redundant information that

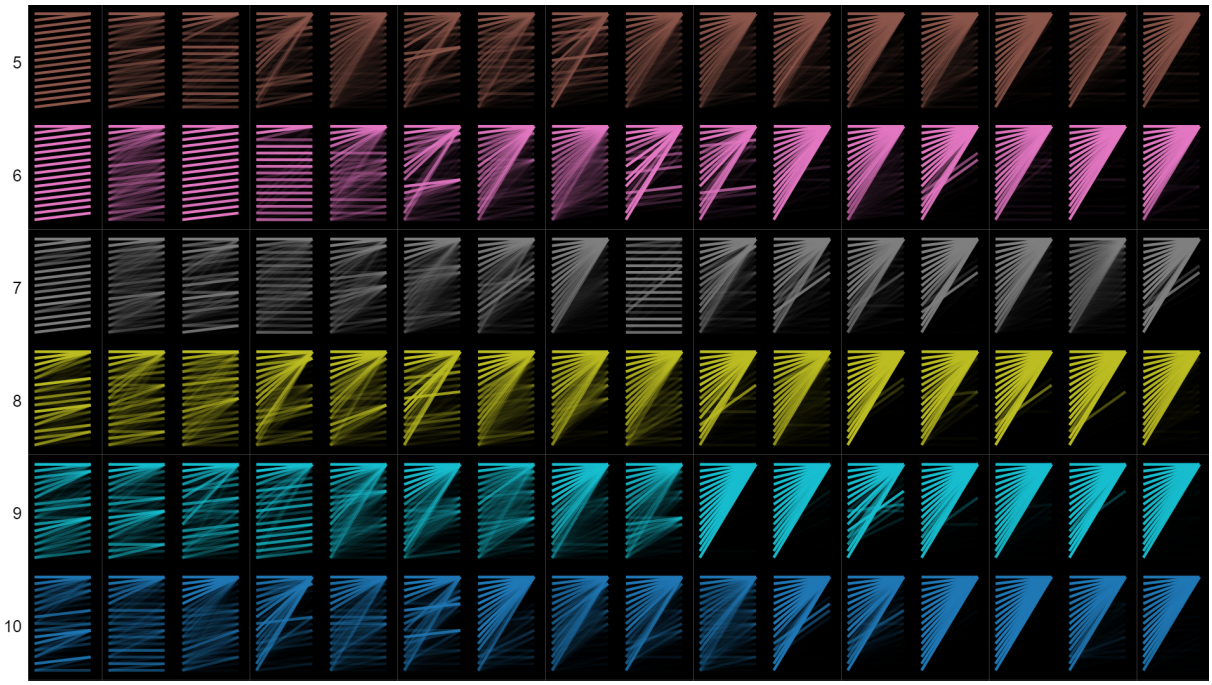



Figure 5: Example attention visualisation for BLOOM-560m, layers 5-10.

reduces its reliability in explanations.

Vig and Belinkov (2019) hypothesise that the structure of attention is closely tied to the pre-training objectives. This connection helps explain why Transformer Decoder LLMs (which are trained using left-to-right language modelling) often show attention patterns that focus on predicting the next token in the sequence. This observation becomes especially clear when comparing these patterns to those found in the BERT model by Kovaleva et al. (2019). Since BERT's pre-training objective involves masked language modelling and next sentence prediction, its attention patterns differ, displaying block patterns (where tokens only focus on other tokens within the same sentence) and vertical patterns (where all tokens focus on the separator between sentences).

In previous work, researchers have evaluated the importance of individual attention heads in Transformer models. Michel et al. (2019) show that in many cases, several heads can be removed from the BERT model without performance loss, and that some layers can even be reduced to only one head. Kovaleva et al. (2019) analysed the attention patterns in the BERT Transformer Encoder model, revealing redundancy in the information encoded by different heads. They demonstrated that the same attention patterns are consistently repeated across various language tasks and showed that not all attention heads are necessary for performance. Similarly, Vig and Belinkov (2019) studied the structure of attention in the generative Transformer GPT-2 and identified specific patterns associated with syntactic and semantic properties, further supporting the idea that some heads may be more important than others.

In our work, we performed a qualitative exploratory analysis of attention maps in the BLOOM and OPT models, ranging from 560 million to 13 billion parameters, which showed that similar attention patterns arise regardless of model type or parameter size. Figure 3 shows the four observed attention patterns: first word, previous word, current word and heterogeneous (everything else). The first three patterns produce attention maps that are nearly identical when visually inspecting them. Figure 5 gives an example of the frequency of these attention patterns in BLOOM-560M. Our hypothesis, as a result of the insights from previous work and our exploratory analysis, is that many redundant heads exist in LLMs that can safely be removed without significantly impacting performance.

### 2.2 Pruning

Pruning is a technique used to reduce the size of a machine learning model at the cost of performance. Pruning is crucial for improving computational efficiency and reducing memory requirements. It involves removing weights that contribute the least to the models performance, thereby reducing model

size at minimal performance cost (NVIDIA, 2021).

Blalock et al. (2020) lays out a framework for defining pruning methods. A neural network *architecture* is a function family $f(x; \cdot)$. The architecture consists of the layout of the network and its operations, such as the self-attention, positional encoding, feed-forward blocks, etc. Example architectures include BLOOM-7b1 and OPT-13b. They define a neural network *model* as a particular parameterization of an architecture, i.e., $f(x; W)$ for specific parameters $W$. Neural network *pruning* entails taking as input a model $f(x; W)$ and producing a new model $f(x; M \odot W')$. Here $W'$ is set of parameters that may be different from $W$. $M \in \{0, 1\}^{|W'|}$ is a binary mask that fixes certain parameters to 0, and $\odot$ is the element-wise product operator. In practice, rather than using an explicit mask, pruned parameters of $W$ are fixed to zero or removed entirely. Which parameters are pruned is determined by a *saliency* score where the low scoring parameters are pruned before the higher scoring parameters.

Pruning a model has multiple benefits. Removing weights from a model allows it to be compressed, reducing the memory requirements, which means the pruned model can potentially be used on smaller devices. Pruning can also reduce the compute requirements allowing for increased efficiency and reduced energy consumption (NVIDIA, 2021). However, there is an important distinction to make between theoretical pruning benefits and benefits realised in practice. Many researchers publish papers with $x\%$ weights pruned, where the weights are not actually removed from the model but instead masked. Men et al. (2024), for example, test the performance of the potential pruned model without demonstrating the pruning benefits. In this thesis, we refer to theoretical pruning benefits and the percentage pruning we specify does not account for the percentage of the weights that are not chosen by our pruning method, e.g. if the attention mechanism accounts for $24\%$ of a model's parameters, then $50\%$ pruning is in actuality only pruning $12\%$ of all parameters. Furthermore, even pruned parameters that are zeroed out won't necessarily reduce compute by the amount pruned, as the weights of certain operations can account for a disproportionate amount of the overall floating point operations (Li et al., 2017).

Pruning methods vary in their choice of structure. Unstructured pruning involves removing individual weights in a model based on a saliency score. The most basic form is magnitude pruning, which prunes the weights with the smallest magnitude under the assumption that smaller weights are less important (LeCun et al., 1989). Other unstructured pruning methods use additional factors to calculate saliency such as neural activations (Sun et al., 2023) or gradients (Frantar and Alistarh, 2023).

Unstructured pruning can achieve high compression but is not hardware friendly and thus the efficiency gains are not realised (NVIDIA, 2021). N:M sparsity pruning (Zhou et al., 2021) fixes this by enforcing hardware friendly constraints in the weight matrices so that for every $M$ weights, $N$ are pruned. Thus a $50\%$ pruning using 4:8 structured sparsity can potentially double the inference efficiency of the pruned model.

The alternative to unstructured pruning is structured pruning, where entire components are removed by eliminating entire weight matrices from the model. For transformer based models, structural pruning is typically done by calculating a saliency score for each attention head and pruning the lowest scoring. Bansal et al. (2023) developed a method for ranking attention heads based on their gradient contributions towards a language modelling task and iteratively pruning the least important heads.

KAMMA et al. (2020) and Kim et al. (2020) applied structured pruning methods to Convolutional Neural Networks by unifying neurons with similar behaviours, allowing one neuron to emulate the functions of another. These methods achieved impressive performance without the need for fine-tuning or retraining, beating the traditional pruning strategy of masking pruned neurons to weights of zero.

Our work seeks to explore structured pruning of Transformer language models where the *score* is an attention similarity metric. Furthermore, we attempt to duplicate weights between attention heads with high similarity.

### 2.3 LLM Interpret

We use the LLM Interpret structural pruning method proposed by Bansal et al. (2023) as a baseline for comparison. This method computes gradient-based importance scores for each attention head based on its contribution to the model's performance on a given task, and subsequently prunes the least important heads.

To find these importance scores, the model processes a given task and calculates the loss. It then

performs backpropagation to obtain the gradients of the loss with respect to each attention head's output. These gradients are a measure of the sensitivity of the loss to changes in that output.

Given a dataset $\mathcal{D}$, the gradient-based importance score for an attention head $h$ is calculated using the formula:

$$IS_h(\mathcal{D}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})} \left| A^h([\mathbf{x};\mathbf{y}])^T \frac{\partial \mathcal{L}(\mathbf{y}|\mathbf{x})}{\partial A^h([\mathbf{x};\mathbf{y}])} \right| \quad (2)$$

where:

- $IS_h(\mathcal{D})$ is the importance score of attention head $h$.
- $\mathbb{E}_{(\mathbf{x},\mathbf{y})}$ represents the average over all input-output pairs $(\mathbf{x}, \mathbf{y})$ in the dataset.
- $A^h([\mathbf{x};\mathbf{y}])$ is the output of attention head $h$ when processing the input $\mathbf{x}$ and the expected output $\mathbf{y}$.
- $\mathcal{L}(\mathbf{y}|\mathbf{x})$ is the loss function, which measures how well the model's prediction matches the actual output $\mathbf{y}$ given the input $\mathbf{x}$.
- $\frac{\partial \mathcal{L}(\mathbf{y}|\mathbf{x})}{\partial A^h([\mathbf{x};\mathbf{y}])}$ is the gradient of the loss with respect to the output of attention head $h$.

The term inside the expectation calculates the element-wise product of the attention head's output and its gradient, taking the absolute value to capture sensitivity. The importance score for a given head is then averaged over the entire dataset.

## 3 Methodology

The method developed in this thesis aims to achieve two things. The first is a structured pruning method where the saliency score is similarity of attention maps between attention heads. The second builds on the first by creating a *duplication pruning* method which duplicates weights from one head to another that is highly similar. Our methodology section will cover the duplication pruning (subsection 3.1), base language models (subsection 3.2), evaluation (subsection 3.3), attention-guided pruning implementation (subsection 3.4), pruning percentages (subsection 3.5), and random pruning baseline (subsection 3.6).

### 3.1 Duplication pruning

The core hypothesis in our method is that attention heads with similar attention output are redundant and can thus be either removed or combined. The typical pruning approach involves removing the similar attention head by setting its weights to zero. Combining the attention heads, which we call *duplication pruning*, is done by copying the weights from one attention head's key, query and value matrices to another attention head. The intuition behind duplication pruning is that the difference between the layer output before and after pruning should be smaller when duplicating heads as compared to removing a head if the heads are similar. This approach to pruning is relatively unexplored as most pruning methods attempt to find the least important components to prune which don't necessarily have a similar but important counterpart.

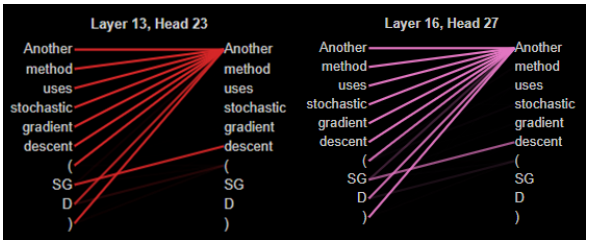



Figure 6: Example of heads with very similar attention patterns in the BLOOM-560m model.

Figure 6 shows an example of two heads that are very similar. Both heads put a significant amount of attention from the first initial of an acronym to the token that helps predict what the next initial would be. In this example, "SG" attends to "descent", as the model needs to predict the next token "D". Our assumption is that if these two heads were in the same layer then it wouldn't be necessary to have both, so one of them could be duplicated and the other removed to improve model efficiency.

The difference between the theoretical and practical pruning value of the *duplication* method is more significant than for most methods, which simply require a pruning mask to be converted to zeros in the pruned model's parameters. Our implementation of the method's duplication of weights provides only theoretical pruning benefits and requires potential restructuring of the architecture. In order for our *duplication pruning* to provide value, the weights would not be copied but instead be set to zero with a duplication map provided to the model's forward pass. This duplication map would then inform the model to use a duplicated head's Key, Query and Value states instead of the removed heads and thus avoid all computations required to calculate self-attention for the removed heads.

| Benchmark | Example prompts | Objective |
|---|---|---|
| PAWS-X | The Tabaci River is a tributary of the River Leurda in Romania, right? Yes, The Leurda River is a tributary of the River Tabaci in Romania.<br>The Tabaci River is a tributary of the River Leurda in Romania, right? No, The Leurda River is a tributary of the River Tabaci in Romania. | Loglikelihood rolling |
| Hellaswag | Roof shingle removal: A man is sitting on a roof. He is using wrap to wrap a pair of skis.<br>Roof shingle removal: A man is sitting on a roof. He is ripping level tiles off.<br>Roof shingle removal: A man is sitting on a roof. He is holding a rubik's cube.<br>Roof shingle removal: A man is sitting on a roof. He starts pulling up roofing on a roof. | Loglikelihood |
| BLiMP Ellipsis | Brad passed one big museum and Eva passed several.<br>Brad passed one museum and Eva passed several big. | Loglikelihood Rolling |
| BLiMP Subject-Verb Agreement | This goose isn't bothering Edward.<br>This goose weren't bothering Edward. | Loglikelihood Rolling |
| Arc Easy | Which of the following statements best explains why magnets usually stick to a refrigerator door? The refrigerator door is smooth.<br>Which of the following statements best explains why magnets usually stick to a refrigerator door? The refrigerator door contains iron.<br>Which of the following statements best explains why magnets usually stick to a refrigerator door? The refrigerator door is a good conductor.<br>Which of the following statements best explains why magnets usually stick to a refrigerator door? The refrigerator door has electric wires in it. | Loglikelihood |

Table 1: Example prompts for our benchmarks. The differences between prompts for a single example are underlined. Loglikelihood objective is log probability of getting the underlined target text given the starting context. Loglikelihood rolling is the log probability of getting the entire sequence. Note that different benchmarks have a different random accuracy due to number of choices for each sample.

### 3.2 Base language models

Our method was primarily tested on the Open Pre-trained Transformer (OPT) model (Zhang et al., 2022) which is an open sourced large language model developed by Meta AI. Since the baseline method was implemented for OPT, developing and testing our method on the same model expedited our development process.

The model comes in various sizes, ranging from 125M to 175B parameters. When it was released, OPT-175B had state of the art performance comparable to GPT-3. The 13B sized model was picked for pruning as a trade off between computational requirements and performance. OPT-13B has 40 layers and 40 heads with a dimension size of 5120. 24% of the weights are contained in the key, query and value matrices, and these are the weights pruned in our masking and duplication methods.

To check whether any observed effects of the proposed duplication pruning may be model-specific, we also evaluate it on BLOOM-7b1, an open-access LLM with 7B parameters. While similar in architecture to OPT, it has two key differences:

- While OPT uses learned positional embeddings, BLOOM uses ALiBi positional embeddings (Press et al., 2022). This embedding scheme adds a bias to the attention scores based on the distance between keys and queries. Since our pruning strategy relies on attention scores, this is a fundamental difference.
- While OPT was trained predominantly on English text, BLOOM is deeply multilingual, as it was trained on the ROOTS corpus (Laurençon et al., 2022), which spans 46 different languages.

### 3.3 Evaluation

For evaluation we used the Language Model Evaluation Harness library (Gao et al., 2023). This library offers over 200 language model evaluation tasks and was also used by our baseline allowing for easy comparisons with our own method. Table 1 shows example prompts for each task. We used five different tasks to test the performance of our pruning method, so as to identify whether any performance differences with our method were task-specific. The rationale for including specific tasks is discussed below.

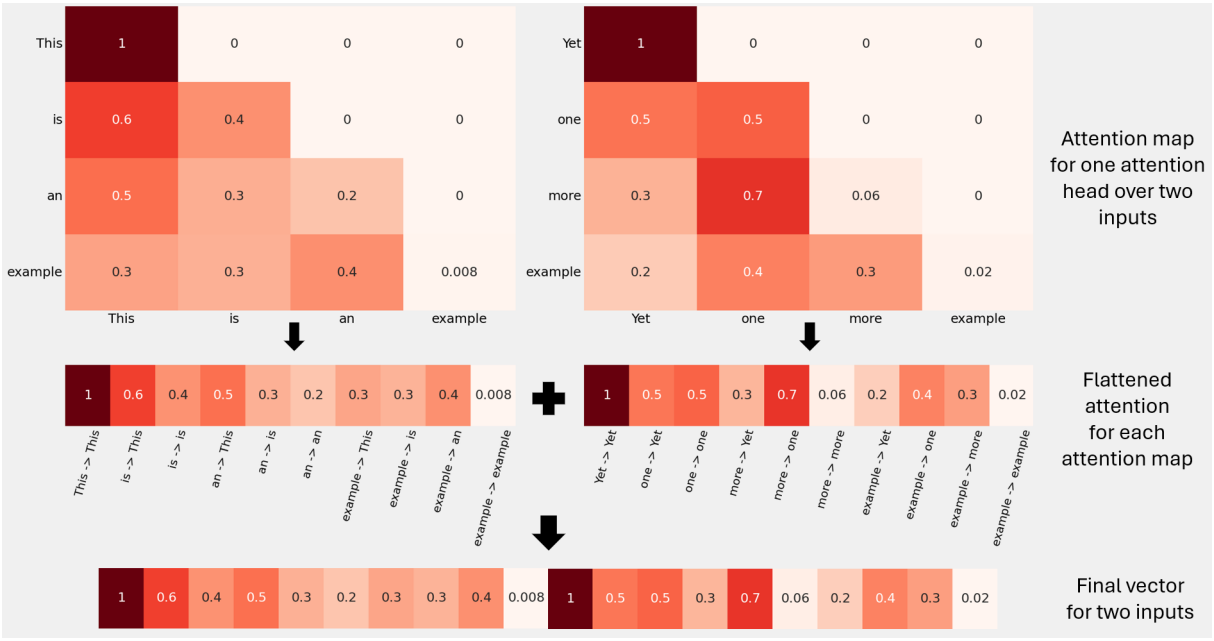



Figure 7: Example of how we convert attention maps from multiple inputs into one vector for each head.

**PAWS** (Paraphrase Adversaries from Word Scrambling) (Zhang et al., 2019) is an adversarial dataset for paraphrase identification. It presents two sentences to the model, which must determine if they paraphrase each other. This task is particularly challenging due to the subtle keyword changes that can significantly alter the meaning of the sentences.

**Hellaswag** (Zellers et al., 2019) comprises a collection of multiple-choice questions that require selecting the most appropriate continuation from a set of possible endings for a given scenario. The dataset includes adversarially chosen distractors that closely resemble the correct answer, thereby requiring subtle discrimination from the model. This task was included as there are often only a few tokens that are important for determining the correct answer, which makes the heads responsible for attending to those tokens critical for getting the right answer for the model. Whether or not these heads are pruned is an important performance factor for our pruning method. Hellaswag was chosen as the default metric in comparisons when only one metric is used.

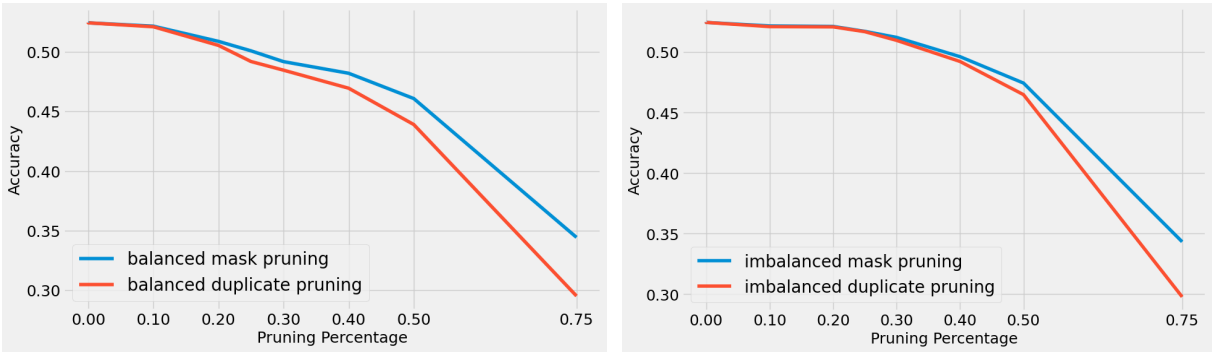
**ARC-Easy** (Clark et al., 2018) is a subset of the AI2 Reasoning Challenge (ARC) benchmark, focusing on elementary-level science questions. The model must choose the correct answer from a set of options, testing its ability to recall scientific facts. This task is relatively less complex but essential for verifying that our pruning method retains the model's factual knowledge.

**BLiMP** (Benchmark of Linguistic Minimal Pairs) (Warstadt et al., 2023) tests linguistic knowledge. Of the 67 individual datasets provided in BLiMP we used *ellipsis_n_bar_1* and *irregular_plural_subject_verb_agreement_1*. The ellipsis dataset tests the possibility of omitting expressions from a sentence, whereas the subject-verb agreement tests that the subjects and present tense verbs agree in number. Each dataset contains 1,000 same-length minimal pairs that contrast in grammatical acceptability for the specific grammatical phenomenon being tested.

By evaluating our pruning methods across these diverse tasks, we aim to ensure robust performance and identify any potential weaknesses specific to certain task types. Example prompts for our benchmarks can be seen in Table 1.

### 3.4 Attention-guided Pruning

For each task, we provide the model with 100 randomly selected samples and get the attention outputs for each head. We use multiple prompts to get a representative attention output for each head across different inputs. The resulting attention map outputs are flattened into a 1-dimensional vector for each attention head. This process is exemplified in Figure 7. The attention output for a single prompt and head is a triangular matrix where the length of the prompt is the dimension of the output. Given an average prompt length of 20 tokens,




(a) Balanced pruning

(b) Imbalanced pruning

Figure 8: Comparison of Hellaswag accuracy for masked vs. duplication pruning for both balanced and imbalanced pruning. Masking performs better than duplicating.

the attention vector for a single head is of length $\frac{20*21}{2} * 100 = 21000$ (number of elements in a 20 x 20 triangular matrix times number of prompts).

We then cluster the attention heads using a similarity metric, with each cluster containing only heads from a single layer. In most experiments described in this study we use cosine similarity, since it had the best performance (see subsection 4.5). Each pair of heads in a layer have a similarity score calculated and are then ranked across all layers. Since it doesn't make sense to compare the attention of heads across layers due the difference in their input distributions, this allows for per-layer similarity to be used as a global ranking. Each head is assumed to start in a cluster of its own, and pairs of heads are then selected from the top of the ranking. The clusters containing the pair are combined, and pairs are subsequently picked until the required number of heads to be pruned is reached.

Each cluster consists of a head to be kept and duplicated, as well as the heads to be pruned. The final number of clusters is equal to the number of heads not pruned, since there is always exactly one head kept per cluster. Therefore, a cluster of size one contains an attention vector that is not similar enough to any other attention vectors to be pruned or duplicated.

The head with the highest mean similarity to the other heads in each cluster is then selected for duplication, and its key, query, and value weight matrices are duplicated to the attention heads that are pruned.

### 3.5 Pruning percentages

Since pruning and evaluating is compute intensive, and given the large amount of different potential hyper parameter combinations, we needed to limit the number of pruning attempts. Consequently, our results were pruned in 10% increments up to 50% with an additional pruning at 75% to check our method's performance at high sparsity.

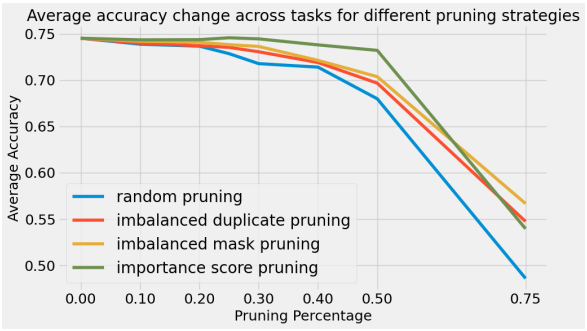


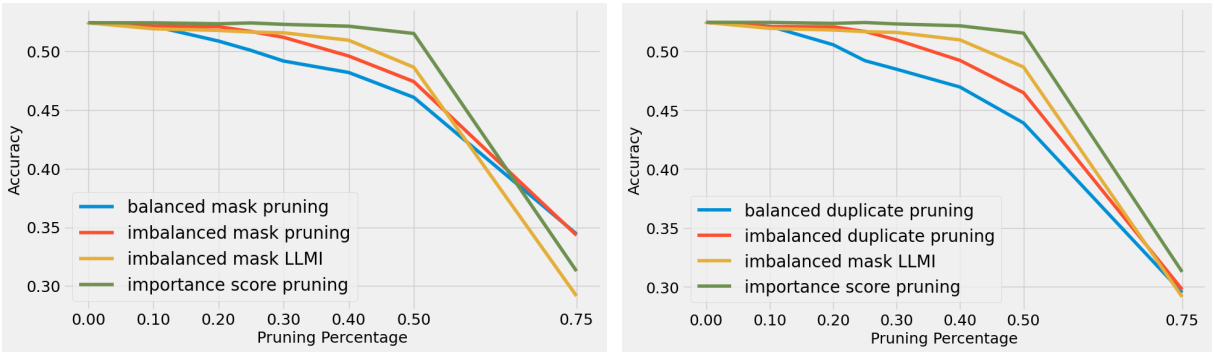

Figure 9: Our method vs. LLM Interpret compared to random pruning for OPT-13b. Both methods were pruned on Hellaswag. Results are average accuracy for BliMP, Hellaswag and ARC Easy

### 3.6 Random pruning baseline

Besides the LLM Interpret method, we include a random pruning baseline where the attention heads are randomly selected to be pruned. This was done by creating a list of all possible heads and then randomly selecting heads from the list until the desired pruning percentage was achieved. We did not duplicate the weights but instead set them to zero, as randomly selecting heads doesn't provide a way of selecting the source head for any duplication.

## 4 Results

In this section, we report the results of our experiments on attention-guided pruning of OPT vs baselines (subsection 4.1), comparing duplication pruning vs masked pruning (subsection 4.2) and balanced vs imbalanced pruning (subsection 4.3),




(a) Masked pruning

(b) Duplicate pruning

Figure 10: Comparison of Hellaswag accuracy for balanced vs. imbalanced pruning for both masked, duplicate and LLMI pruning. Imbalanced performs better than balanced.

its performance in BLOOM (subsection 4.4), the effect of the similarity metric choice on the method performance (subsection 4.5), and the data used for pruning (subsection 4.7).

### 4.1 Attention-guided pruning

Figure 9 shows the performance of our proposed attention-guided pruning against the baselines: the state-of-the-art LLM Interpret method (Bansal et al., 2023) as well as random pruning subsection 3.6. We find that attention-guided pruning has worse accuracy than LLM Interpret at all stages except for 75% sparsity, which indicates that our method may identify better heads to remove at extreme pruning percentages, where the gradient-based importance score is less informative.

### 4.2 Pruning with Duplication vs. Masking

To test our core hypothesis of duplication pruning being better than masked pruning, we did all our experiments using both. The only difference between the two methods is that masked pruning sets the weights of the pruned head to zero instead of copying weights from the original head.

Figure 8 shows that masking is slightly better than duplication pruning. This result does not support the idea that replacing the weights of a head with a similar head could reduce the error caused by pruning.

### 4.3 Balanced vs. Imbalanced Pruning

Using similarity for pruning means that we can only cluster within a layer, as we cannot compare attention across layers (since each layer has a different input distribution due to the non-linearities introduced by the feed forward components between attention blocks). We test three different pruning strategies:

- **Imbalanced**: calculates similarity between heads within a layer and then ranks them globally, pruning the highest scoring heads. This is the default strategy used and the one described in subsection 3.4.
- **Balanced**: prunes an equal number of heads from each layer based on similarity.
- **LLM Interpret Imbalanced**: observes how many heads the LLM Interpret method prunes per layer and uses that to inform our own pruning method, e.g. if LLM interpret prunes 8 heads in the first layer and 2 in the second, then our method will do the same. This allows us to see how head similarity and head importance score compare when having to select the same amount of heads per layer. The difference in number of heads kept per layer for 50% pruning can be seen in subsection 5.2.

Figure 10 shows that imbalanced is better than balanced. This is expected, as previous research shows that some layers have a higher concentration of important heads that significantly contribute to the model's performance on various language modelling tasks (Bansal et al., 2023). Therefore, pruning an equal number of heads from both 'important' and 'unimportant' layers is likely to lead to worse performance than an imbalanced approach. This figure also shows that using the LLM interpret pruned head distribution across layers increases the performance of our own pruning method at higher pruning percentages with the exception of 75% pruning where it significantly drops off. This

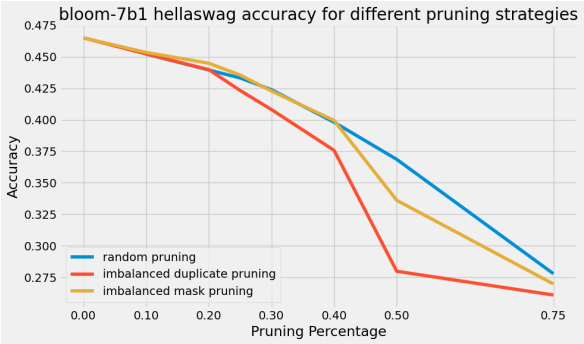



Figure 11: BLOOM-7b1 pruned on hellaswag - Hellaswag accuracy compared to random pruning.

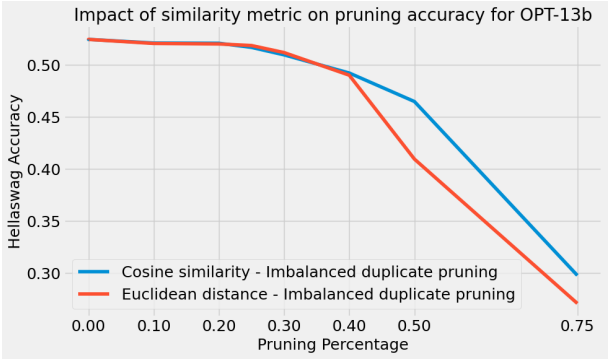



Figure 12: Impact of Euclidean vs cosine metric for choosing head similarity on imbalanced duplicate pruning Hellaswag accuracy. Cosine similarity has slightly better performance, although the difference is small.

better performance at 40% to 50% does not bridge the gap in performance with the baseline LLM interpret method.

### 4.4 Attention-guided Pruning for BLOOM

Figure 11 shows results when pruning on the BLOOM-7b1 model. The LLM Interpret method is not implemented for BLOOM, so our only comparison is to random pruning. As can be seen in the figure, our results are marginally better than random pruning for low pruning percentages and become worse than random for higher levels of sparsity. Similarly to the results for OPT (Figure 8), masked pruning gets better performance than duplication pruning, with a greater disparity in performance at higher pruning percentages.

These results indicate that our pruning method does not work for BLOOM. The two main differences between BLOOM and OPT are the usage of ALiBi positional encoding and the multilinguality of the data BLOOM was trained on. One or both of these factors somehow cause the similarity approach to pruning to be counter-productive, with worse than random results. How exactly this occurs requires further investigation.

### 4.5 Euclidean Distance vs Cosine Similarity

In addition to cosine similarity for scoring attention heads, we also tested Euclidean distance. Since all attention maps are taken from the same inputs, their vector lengths are constant, making the difference between these metrics a matter of orientation versus absolute positioning. Cosine similarity measures the angular difference between vectors, highlighting the direction of relationships between the attention maps. On the other hand, Euclidean distance calculates the straight-line distance between vectors, focusing on their exact positions. By comparing these two metrics, we aim to understand how different approaches to measuring similarity affect the performance of our pruning method.

In Figure 12 we compare the performance of the pruning method when using the two similarity metrics. The choice of similarity metric appears to have little to no importance until 40% of the model is pruned, and at higher sparsity cosine appears to be better. Since cosine is better it was used for all other experiments.

### 4.6 Impact of similarity on choosing the heads to prune

The final usage of the similarity metric is when selecting which head to keep in a cluster of heads. It is not clear how important it is to select the right head in the cluster. Thus we compare random selection vs. selection based on a a similarity metric. We use the cosine similarity for creating the cluster, and then randomly selecting which head to keep in each cluster. This could show how similar the heads are in each cluster. If they are very similar,

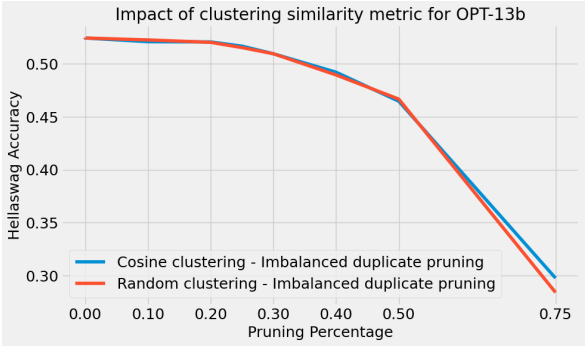



Figure 13: Similarity metric for clustering attention heads does not seem to impact pruning accuracy.

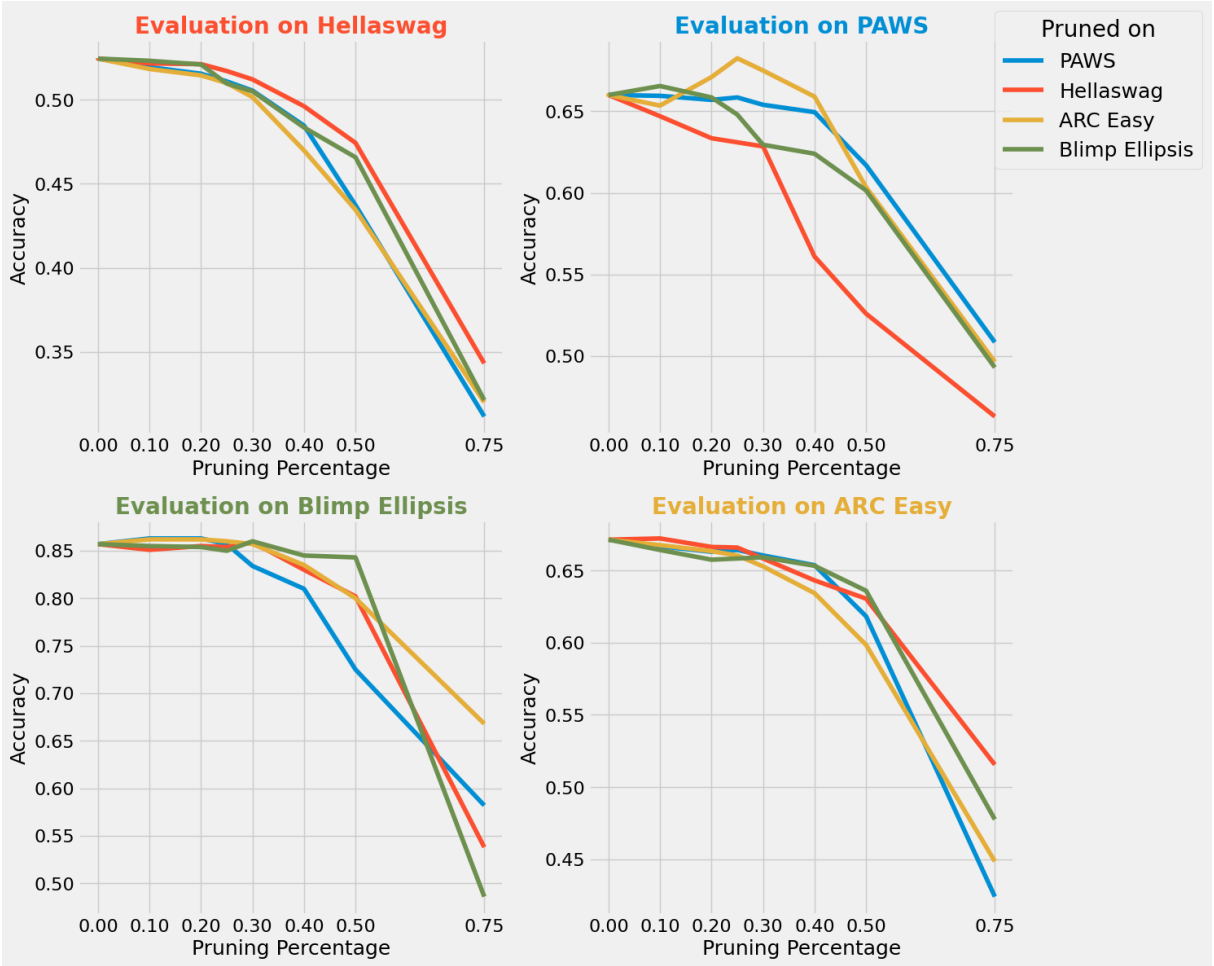



Figure 14: Cross-task transfer of attention head similarity as measured by impact of imbalanced mask pruning on accuracy of different tasks. Interestingly, models pruned on a task don't always perform better at that task, especially at higher levels of pruning.

then the selection of the head is not important and random selection will have similar performance. Alternatively, if the heads in a cluster are quite different, then selecting the correct head will likely have a larger impact on performance.

The results of this experiment can be seen in Figure 13. We observe no difference in performance until the 75% pruning mark. This seems to suggest that the clusters are generally homogeneous with few outliers. Random selection becomes worse at very high pruning percentages, which could be explained by the fact that there are fewer clusters at higher pruning percentages. Thus, the similarity requirement for being in a cluster with other heads is much lower, which could lead to a higher spread of similarities, and thus picking the head with the most similarity to the other heads in the cluster is more important.

### 4.7 Prompt sources

The prompts used for pruning the models we use for the comparisons in the previous sections were randomly selected from the Hellaswag dataset. Since these prompts are formatted with a specific structure, as can be seen in Table 1, our method might be biased by the prompt source. To investigate this possibility, several other pruning prompt sources were selected for comparison.

These prompt sources are the other evaluation tasks described in subsection 3.3. A comparison between different sources of prompts was made to see if the source of the prompt has any impact on performance and to check how pruning on one task generalises to others. The LLM Interpret pruning method also prunes for a specific task by selecting prompts from said task. In this way, the LLM interpret method can get better relative results on Hellaswag by using Hellaswag input. We reproduce this experiment as shown in Figure 14, which

shows that a model pruned using data from a given task, tends to perform better at that task than most models pruned on other tasks, although the performance gain is minimal and not consistent across pruning percentages.

## 5 Analysis

The results we presented in section 4 do not confirm our hypothesis that attention-guided pruning could be an effective way to reduce the amount of computation while preserving the performance of the model. In this section we will investigate the possible causes for this result:

- Pruned head overlap with LLM interpret (subsection 5.1) provides a potential explanation for why similarity pruning is significantly worse at 50%.
- Looking at layer importance at 50% and 75% (subsection 5.2) provides a potential explanation for our method's superior performance over LLM at 75%.
- Head similarity distribution analysis subsection 5.3 suggests our method may have some merit.
- Adjusted Rand index in subsection 5.4 supports the assumption that similar heads are consistently similar across many inputs, which is necessary for our method to work.
- Revisiting our exploratory qualitative analysis by removing the first token from the attention maps subsection 5.5 provides a possible explanation for why our pruning methods fail to get good results.

### 5.1 Pruned heads: comparison to gradient-based importance

Our methods so far have shown results that are better than random but significantly worse than LLM Interpret at all but the highest levels of sparsity. A possible explanation for our method's poor performance is that it is removing the most important heads according to other saliency metrics. We therefore create a comparison of the number of heads which LLM Interpret deems important and are pruned by our method (Figure 15). An identical pruning method to LLM Interpret would have zero percent of the top 800 important heads pruned up to and including the 50% mark as there are only 1600

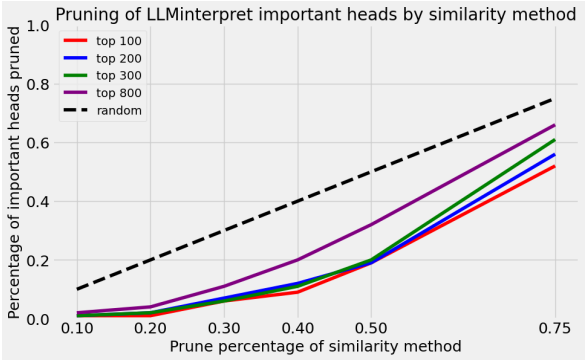



Figure 15: Percentage of top N important heads from LLM interpret pruned at each pruning percent of default pruning method

heads to prune in total. The figure shows that this is clearly not the case and that a significant portion of the important heads are pruned even at relatively low pruning percentages. Furthermore, a significant amount of LLM Interpet's most important heads are also pruned by our method (for example, they would only prune their top 100 heads once the pruning percentage reached 94%). However, we noticeably do not prune more important heads than random chance, which would suggest that using attention output similarity provides some benefit for head selection for pruning. Figure 16 shows the overlap of pruned heads between LLM Interpret and our method. It shows that there is significant amount of overlap even at low pruning percentages. This comparison would also suggest that there is some merit to using attention head output similarity as a pruning metric.

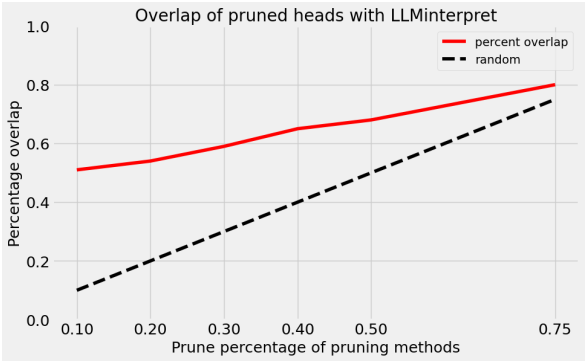



Figure 16: Pruned heads overlap between LLM Interpret and default pruning method

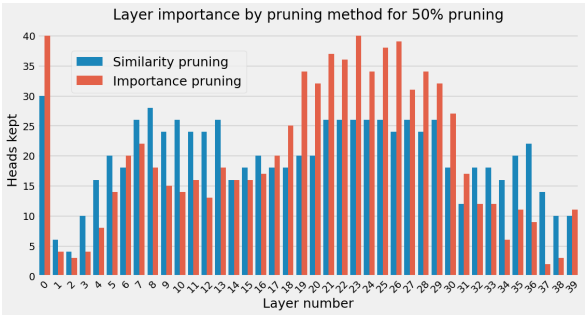



Figure 17: How many heads were kept per layer for OPT-13b at 50% pruning when pruning using Hellaswag prompts.

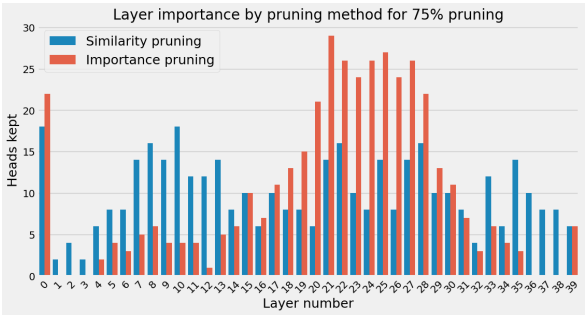



Figure 18: How many heads are kept for 75% pruning using Hellaswag prompts. Note that LLM interpret prunes all heads in several layers.

## 5.2 Layer importance

In subsection 4.3 we showed that using the LLM interpret distribution of pruned heads per layer to inform our own method we got increased performance at 50% pruning but the performance decreased significantly at 75% matching the LLM interpret baseline's decrease in performance. This suggests that the distribution of pruned heads across layers is important with the similarity method being better at 75%.

Figure 17 shows the distribution of important heads across layers for the OPT-13b model for the similarity metric pruning method and the LLM interpret pruning method at 50% pruning. Both methods clearly consider the first layer to be the most important and the layers immediately following to be the least important. The similarity pruning method has less variance across the middle layers as compared to the LLM interpret method where layers 19-30 are more important than the surrounding layers. This figure demonstrates that the similarity pruning method has similar layer priorities to LLM interpret yet is still quite different.

Figure 18 shows the distribution when the pruning percentage is increased to 75%. The variance in number of heads pruned per layer for the importance pruning method is more extreme at 75% pruning. A result of this extreme variance is that the LLM Interpret method prunes all heads in several layers (1-3 and 36-38). Due to the existence of the residual stream, pruning an entire layer does not destroy information from layers preceding the removed layer. However, considering that both LLM Interpret's and our method using LLM Interpret's distribution get worse results at 75%, the large difference in distribution of pruned heads between the two methods suggests that the difference in distribution might be a major factor.

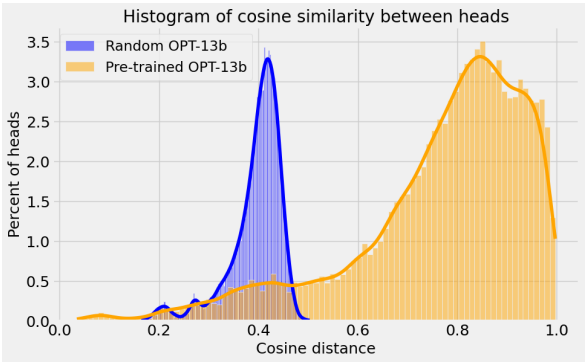
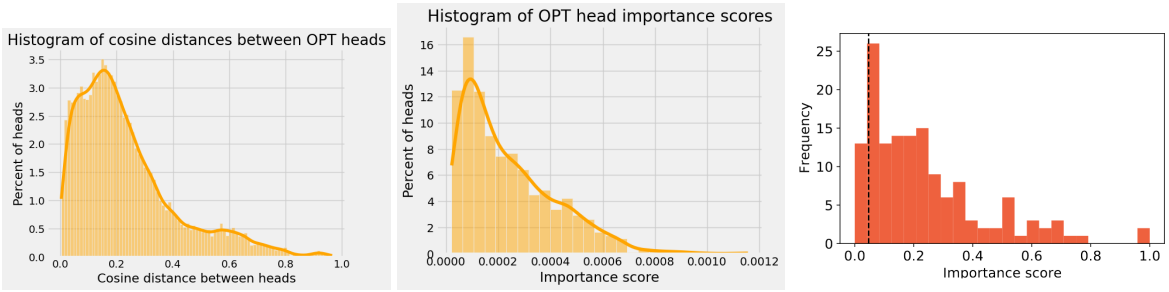



Figure 19: The attention vector of most heads in the pre-trained OPT-13b model have high cosine similarity when compared to a randomly initialised model, indicating that model training brings them closer together.

## 5.3 Head Similarity

Figure 19 shows the distribution of head similarities for the pre-trained OPT-13b and for a randomly initialised, untrained OPT-13b. We can see that the heads in a trained model tend to be more similar, but it is surprising to see that almost no heads are less similar than the randomly initialised ones, suggesting that the training brings most heads closer together, rather than bringing a subset of them closer while pushing others further apart. This has implications for our pruning method, as selecting the most similar pairs of heads to be duplicated/pruned becomes more difficult when most heads are clustered around a high level of similarity. This result is also reflected in Figure 5 with the large visual similarity between attention heads.

Figure 20 presents the head importance score distributions found by Bansal et al. (2023) for OPT and by Prasanna et al. (2020) for BERT. When these distributions are compared with the distribution of cosine distances between the attention vectors used in our method, a clear pattern emerges: both exhibit right-skewed, long-tailed distributions.




(a) Cosine distance distribution for our method

(b) Importance distribution for LLM Interpret

(c) BERT importance distribution. Source: Prasanna et al. (2020)

Figure 20: The gradient-based importance score distributions of BERT and OPT resemble the cosine distance distribution of head attention vectors found by our method. They are all right-skewed long-tailed distributions.

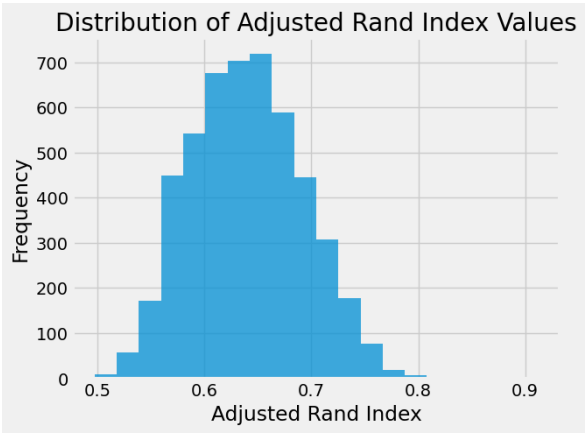



Figure 21: Rand index values for Opt-13b when pruning for 50%. Clusterings are never identical but always somewhat similar for different prompts.

This indicates a consistency in the saliency of attention heads across different models and methods, which suggests that there is potential to using cosine similarity as a saliency metric.

### 5.4 Adjusted Rand Index

One of our basic assumptions is that the attention output of similar heads is similar across many outputs. If this isn't the case, then the clustering becomes random and provides no value. To test this, we clustered the attention heads by similarity of the attention output for a single Hellaswag sample. We did this for 100 random samples and calculated the adjusted rand index (Rand, 1971) pairwise for the 100 prompts. The adjusted rand index is a similarity measure between two clusterings and is adjusted for chance where a score of -1 is completely different clusterings and 1 is identical clusterings. Figure 21 shows that the clusterings between two different prompts are always similar but never identical. This supports the idea that heads with similar attention outputs for a given prompt also have similar attention outputs for other prompts. Since the clusterings are never identical, despite the prompts being from the same source, it also suggests that the duplication pruning is likely to introduce errors since there is always enough difference in attention output to significantly change the clusterings. This may be a possible factor for why masked is always better than duplication.

### 5.5 No first token

Since our methods for pruning attention heads are significantly worse than LLM interpret, we revisited our initial motivation provided by the exploratory analysis of attention maps in subsubsection 2.1.3. One thing that stands out is the amount of attention in the maps that is targeting the first token, as seen in Figure 5. Since duplication is always worse than masked pruning, we believe a contributing factor is that similar looking attention maps have important subtle differences that are removed in duplication pruning. Removing the attention to the first token, as shown in Figure 22, reveals some of these differences. These subtle but likely important differences potentially mean that many similar looking heads are not duplicates of each other but merely appear similar. The similarity metrics that we used would put a large emphasis on the significant weight to the first token despite the smaller weights to other tokens likely being more important than they appear. This could be a factor for why our pruning methods performed poorly.

## 6 Discussion

The duplication and masked pruning of OPT performed worse than baseline up to 50% sparsity, with BLOOM performing worse than random. This contradicts research which shows that large lan-

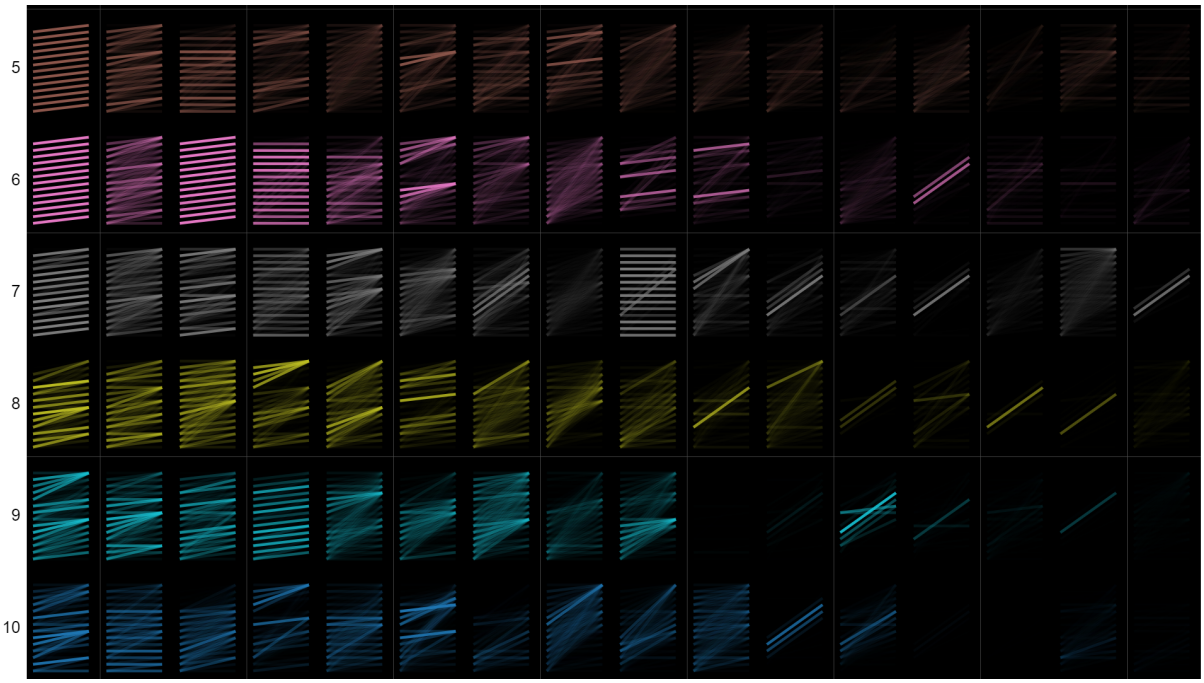



Figure 22: Attention visualisation for BLOOM-560m, layers 5-10 after removing all attention to the first token.

guage models can be pruned up to 50% with minimal performance loss ((Michel et al., 2019; Kovaleva et al., 2019; Bansal et al., 2023)). The large difference in results between OPT and BLOOM suggests that our method should be further tested on more models to determine the cause of this disparity. Special attention should be taken in selecting other multilingual models and models with different positional encodings, as these are the major differences between BLOOM and OPT.

The most interesting result from our experiments is that our method's performance at 75% was better than LLM Interpret's result on the OPT model. Our analysis in subsection 5.2 suggests the reason for this is the distribution of heads pruned per layer. Thus, one potential area of research would be pruning OPT to 50% using LLM interpret and then using our similarity method's pruned head distribution to inform LLM interpret about which remaining heads to prune. Alternatively, using our method at 75% pruning to determine the number of heads to prune at each layer and then constraining LLM Interpret to prune that amount of heads per layer, similar to the imbalanced mask LLMI method described in subsection 4.3. This could potentially improve LLM Interpret's method at high pruning percentages.

KAMMA et al. (2020) present a method for unifying Neurons and channels in CNNs through duplication and Kim et al. (2020) present a method for duplicating filters in CNNs. Both demonstrate that their methods are better than other current methods. Our duplication of attention heads using similarity of attention output does not match performance of similar structural pruning methods for generative LLMs (i.e Bansal et al. (2023)) and performs worse than masking (subsection 4.2). This might suggest that duplication of self-attention in multi-headed attention does not work, but further research is needed to confirm this. Potential avenues of research include combining masking and duplication, where duplication is only done when a pruned attention head has a sufficiently similar non-pruned attention head to use as a source of the duplication.

## 7 Conclusion

In this thesis, we investigated the effectiveness of attention-based structural pruning in reducing the size of LLMs (BLOOM and OPT) without significant performance loss and asked the following research question:

*Can pruning attention heads in large language models based on similarity of attention output reduce their size without compromising performance?*

For the two models we tested (OPT and BLOOM), pruning up to 50% of the Key, Query, and Value matrices had worse performance compared to the baseline used (LLM Interpret (Bansal et al., 2023) for OPT and random pruning for

BLOOM). However, similarity pruning at 75% on OPT was significantly better than LLM Interpret, which merits further research.

As an extension of our similarity method we had the following research question:

*Can attention similarity be used to improve performance by duplicating attention heads instead of masking them?*

In all of our results, duplicating attention heads is worse than masking. Although previous researched showed similar duplication methods work on CNNs (KAMMA et al., 2020; Kim et al., 2020), our results suggest that duplication may not work for the attention mechanism of Transformers. Further research should be conducted to confirm this as there may be alternative ways to duplicate attention heads.

## References


Jay Alammar. 2018. The illustrated transformer. *The Illustrated Transformer–Jay Alammar–Visualizing Machine Learning One Concept at a Time*, 27:1–2.

Bing Bai, Jian Liang, Guanhua Zhang, Hao Li, Kun Bai, and Fei Wang. 2021. Why attentions may not be interpretable? In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 25–34.

Hritik Bansal, Karthik Gopalakrishnan, Saket Dingliwal, Sravan Bodapati, Katrin Kirchhoff, and Dan Roth. 2023. Rethinking the role of scale for in-context learning: An interpretability-based case study at 66 billion scale. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11833–11856, Toronto, Canada. Association for Computational Linguistics.

Davis W. Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John V. Guttag. 2020. What is the state of neural network pruning? *CoRR*, abs/2003.03033.

Gino Brunner, Yang Liu, Damián Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. 2020. On identifiability in transformers.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1:1.

Elias Frantar and Dan Alistarh. 2023. Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Benjamin Hoover, Hendrik Strobelt, and Sebastian Gehrmann. 2020. exBERT: A Visual Analysis Tool to Explore Learned Representations in Transformer Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 187–196, Online. Association for Computational Linguistics.

Theo Jaunet, Corentin Kervadec, Romain Vuillemot, Grigory Antipov, Moez Baccouche, and Christian Wolf. 2021. Visqa: X-raying vision and language reasoning in transformers. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):976–986.

Koji KAMMA, Yuki ISODA, Sarimu INOUE, and Toshikazu WADA. 2020. Neural behavior-based approach for neural network pruning. *IEICE Transactions on Information and Systems*, E103.D(5):1135–1143.

Woojeong Kim, Suhyun Kim, Mincheol Park, and Geunseok Jeon. 2020. Neuron merging: Compensating for pruned neurons. In *Advances in Neural Information Processing Systems*, volume 33, pages 585–595. 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.

Hugo Laurençon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, Teven Le Scao, Leandro Von Werra, Chenghao Mou, Eduardo González Ponferrada, Huu Nguyen, Jörg Frohberg, Mario Šaško, Quentin Lhoest, Angelina McMillan-Major, Gerard Dupont, Stella Biderman, Anna Rogers, Loubna Ben allal, Francesco De Toni, Giada Pistilli, Olivier Nguyen, Somaieh Nikpoor, Maraim Masoud, Pierre Colombo, Javier de la Rosa, Paulo Villegas, Tristan Thrush, Shayne Longpre, Sebastian Nagel, Leon Weber, Manuel Muñoz, Jian Zhu, Daniel Van Strien, Zaid Alyafeai, Khalid Almubarak, Minh Chien Vu, Itziar Gonzalez-Dios, Aitor Soroa, Kyle Lo, Manan Dey, Pedro Ortiz Suarez, Aaron Gokaslan, Shamik Bose, David Adelani, Long Phan, Hieu Tran, Ian Yu, Suhas Pai,

Jenny Chim, Violette Lepercq, Suzana Ilic, Margaret Mitchell, Sasha Alexandra Luccioni, and Yacine Jernite. 2022. The bigscience roots corpus: A 1.6tb composite multilingual dataset. In *Advances in Neural Information Processing Systems*, volume 35, pages 31809–31826. Curran Associates, Inc.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2017. Pruning filters for efficient convnets.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024. Shortgpt: Layers in large language models are more redundant than you expect.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

NVIDIA. 2021. NVIDIA Ampere Architecture Whitepaper. `https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf`.

Cheonbok Park, Inyoup Na, Yongjang Jo, Sungbok Shin, Jaehyo Yoo, Bum Chul Kwon, Jian Zhao, Hyungjong Noh, Yeonsoo Lee, and Jaegul Choo. 2019. Sanvis: Visual analytics for understanding self-attention networks. In *2019 IEEE Visualization Conference (VIS)*, pages 146–150. IEEE.

Sai Prasanna, Anna Rogers, and Anna Rumshisky. 2020. When BERT Plays the Lottery, All Tickets Are Winning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3208–3229, Online. Association for Computational Linguistics.

Ofir Press, Noah A. Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation.

William M. Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

Marcos Treviso, Ji-Ung Lee, Tianchu Ji, Betty van Aken, Qingqing Cao, Manuel R. Ciosici, Michael Hassid, Kenneth Heafield, Sara Hooker, Colin Raffel, Pedro H. Martins, André F. T. Martins, Jessica Zosa Forde, Peter Milder, Edwin Simpson, Noam Slonim, Jesse Dodge, Emma Strubell, Niranjan Balasubramanian, Leon Derczynski, Iryna Gurevych, and Roy Schwartz. 2023. Efficient Methods for Natural Language Processing: A Survey. *Transactions of the Association for Computational Linguistics*, 11:826–860.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy. Association for Computational Linguistics.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy. Association for Computational Linguistics.

Pablo Villalobos, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, Anson Ho, and Marius Hobbhahn. 2022. Machine learning model sizes and the parameter gap.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2023. Blimp: The benchmark of linguistic minimal pairs for english.

Laura Weidinger, Jonathan Uesato, Maribeth Rauh, Conor Griffin, Po-Sen Huang, John Mellor, Amelia Glaese, Myra Cheng, Borja Balle, Atoosa Kasirzadeh, Courtney Biles, Sasha Brown, Zac Kenton, Will Hawkins, Tom Stepleton, Abeba Birhane, Lisa Anne Hendricks, Laura Rimell, William Isaac, Julia Haas, Sean Legassick, Geoffrey Irving, and Iason Gabriel. 2022. Taxonomy of risks posed by language models. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 214–229, New York, NY, USA. Association for Computing Machinery.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *CoRR*, abs/1905.07830.

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.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: paraphrase adversaries from word scrambling. *CoRR*, abs/1904.01130.

Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. 2021. Learning N: M fine-grained structured sparse neural networks from scratch. *CoRR*, abs/2102.04010.