# Extracting Interpretable Task-Specific Circuits from Large Language Models for Faster Inference

**Jorge García-Carrasco, Alejandro Maté, Juan Trujillo**

Department of Software and Computing Systems, University of Alicante, Spain
jorge.g@ua.es, {amate,jtrujillo}@dlsi.ua.es

## Abstract

Large Language Models (LLMs) have shown impressive performance across a wide range of tasks. However, the size of LLMs is steadily increasing, hindering their application on computationally constrained environments. On the other hand, despite their general capabilities, there are many situations where only one specific task is performed, rendering all other capabilities unnecessary and wasteful. This leads us to the following question: *Is it possible to extract the minimal subset from an LLM that is able to perform a specific task in a faster, standalone manner?* Recent works on Mechanistic Interpretability (MI) have shown that specific tasks are performed by a localized subset of components, or circuit. However, current techniques used to identify the circuit cannot be used to extract it for its standalone usage. In this work, we propose a novel approach to automatically extract the subset of the LLM that properly performs a targeted task requiring no additional training and a small amount of data samples. We evaluate our approach on different tasks and show that the resulting models are (i) considerably smaller, reducing the number of parameters up to 82.77% and (ii) more interpretable, as they focus on the circuit that is used to carry out the specific task, and can therefore be understood using MI techniques.

## 1 Introduction

Large Language Models (LLMs) based on the Transformer architecture (Vaswani et al. 2017) have recently shown impressive performance across a wide range of tasks (Brown et al. 2020; Achiam et al. 2023). These models are characterized by having a large amount of parameters as well as being trained on large amounts of broad data in a self-supervised manner (e.g predicting the next token). In fact, it has been empirically shown that the performance of LLMs increases with the number of parameters, as well as training dataset size and amount of training (Kaplan et al. 2020). Therefore, we expect the next generation of LLMs to improve their performance even more as they continue growing even larger.

However, the computational cost can become prohibitive in many situations. As an example, it takes around 130.4GB of memory just to load the largest Llama2 model (Touvron et al. 2023a) (70 billion parameters in 16-bit precision), totaling up to two A100 80GB GPUs. This clearly shows that the use of LLMs can be a challenge in resource-constrained environments.

Because of this, the field of model compression has become crucial, specially on LLMs (Zhu et al. 2023). In short, model compression techniques aim to reduce either the space required to load a model, the computational cost of performing the forward pass for inference, or both. Current model compression techniques can be grouped in four types: (i) Quantization-based techniques involve reducing the precision of the parameters and/or activations (Liu et al. 2023; Kim et al. 2024; Yao et al. 2022; Park et al. 2022). (ii) Pruning techniques, first introduced in LeCun, Denker, and Solla (1989), consist on reducing the size of the model by removing parameters or components of the model. These techniques can be divided into unstructured (Frantar and Alistarh 2023; Zhang et al. 2023; Sun et al. 2023) and structured approaches (Ma, Fang, and Wang 2023; Santacroce et al. 2023), where the main difference is that unstructured approaches remove individual parameters, leading to a sparse model, whereas structured approaches remove connections or hierarchical structures, therefore preserving the overall network structure. (iii) Knowledge distillation aims to transfer knowledge from a complex (teacher) model, into a simpler (student) model (Hinton, Vinyals, and Dean 2015; Gu et al. 2024; Agarwal et al. 2024; Huang et al. 2022; Shridhar, Stolfo, and Sachan 2023). (iv) Low-rank factorization are techniques that try to approximate a parameter matrix by a product of two lower-dimensional matrices, hence reducing the number of parameters (Povey et al. 2018; Idelbayev and Carreira-Perpiñán 2020).

Although these works on model compression have achieved great results, they generally focus on reducing the computational size/cost of the model while maintaining the general performance as much as possible. However, this leads us to the following question: What if we want to take advantage of the capabilities that an LLM has on a specific task? In other words, *is it possible to extract the minimal subset from an LLM that is able to perform a specific task in a faster, standalone manner?*

Recent Mechanistic Interpretability (MI) works have shown that specific tasks are indeed carried out by a number of components that can be localized. The field of MI aims to reverse-engineer neural networks to explain their behaviors in human-understandable components (Elhage et al. 2021,

2022; Olsson et al. 2022). The current workflow for most of the works is to locate a *circuit* (Olah et al. 2020) (i.e. a subset of the model) that is responsible for a specific task by performing a series of causal interventions. For example, Wang et al. (2023) show that the task of Indirect Object Identification (IOI) in GPT-2 Small is performed by a circuit composed of 26 attention heads grouped in 7 different classes, according to their role in the circuit. Likewise, Hanna, Liu, and Variengien (2023) apply different MI techniques to show how GPT-2 Small performs the greater-than operation. Similarly, García-Carrasco, Maté, and Carlos Trujillo (2024) localize the circuit responsible for three-letter acronym prediction, as well as interpreting how the different components of the circuit work.

There are already starting to appear automatic circuit discovery (ACD) methods (Conmy et al. 2023; Bhaskar et al. 2024; Hanna, Pezzelle, and Belinkov 2024) that are able to automatically identify the underlying circuit for specific tasks with good performance. However, the focus of these works has been solely on interpretability (i.e. localizing the circuit or components that are relevant for a specific task) whereas the potential for model compression has not been explored, as these methods do not enable the extraction of the discovered circuit for its independent usage. In fact, the authors of Conmy et al. (2023) remark that their techniques generally slow the forward pass of the LLM.

In this work, we propose a novel model pruning approach that automatically extracts the subset of the model responsible for carrying out a given task on an LLM, enabling its standalone usage without any further training or fine-tuning. Given a dataset that elicits the behavior or task of interest, our approach will automatically identify the relevant components and properly prune the LLM so that we obtain the subset of components that are able to efficiently perform the task. The resulting submodel/circuit will be (i) considerably smaller in terms of the number of parameters, reducing the required storage, (ii) require a lower number of operations, thus reducing the inference time and (iii) focused on the components that perform the specific task under study, improving our ability to understand it via MI techniques, thereby increasing the model's trustworthiness. We believe that incorporating MI into model pruning approaches will enable us to fully leverage the potential of LLMs across a wide range of applications, yielding smaller, faster, and more interpretable models. In summary, our contributions are:

- We propose a novel MI-based pruning method that automatically extracts the subset of an LLM that is able to properly perform a specific task of interest. Our approach does not require any extra fine-tuning and leads to models with significantly lower size and inference time.
- We perform an exhaustive evaluation on three different tasks whose circuits have been already manually identified on GPT-2 Small to (i) study the effect of the hyperparameters on the resulting pruned model (ii) evaluate their accuracy, size and speed and (iii) show if the pruned model contains the important components that were manually identified in previous works.

The rest of the paper is structured as follows: Section 2 presents the required background. Section 3 describes our approach to automatically extract task-specific circuits from LLMs. Section 4 showcases the approach by performing an exhaustive evaluation across different tasks and models. Finally, the conclusions are presented in Section 5.

## 2 Background

In this section, we will briefly present the required background and techniques that will be used to efficiently extract task-specific circuits from LLMs.

### Transformer Architecture

Practically all LLMs such as GPT (Radford et al. 2019; Brown et al. 2020), Llama (Touvron et al. 2023a,b) or LaMDA (Thoppilan et al. 2022) are based on the same transformer architecture (Vaswani et al. 2017). We will adopt and briefly present the mathematical formulation of the transformer architecture presented in Elhage et al. (2021), as it provides a better framework when it comes to our approach.

Essentially, the input to the model is a sequence of $N$ consecutive tokens which are encoded into $x_0 \in \mathbb{R}^{N \times d}$ via a learned embedding matrix $W_E \in \mathbb{R}^{V \times d}$, $V$ being the size of the vocabulary and $d$ the internal dimension of the model. Now, the vector $x_0$ is defined as the initial value of the *residual stream*, where all the components of the model read from and write to. We can think of the residual stream as a main channel which components can read and modify its contents.

There are two main types of components: attention heads, and Multi-Layer Perceptrons (MLPs). The definition of components depends on the level of granularity that we want to achieve: we could break an MLP into several neurons, or merge the different attention heads into an attention layer. In our work, we decided to take individual attention heads and MLPs as components.

Following this convention, an attention layer will be composed by $n_{head}$ different heads that will independently read from the residual stream $x_i$, perform the self-attention mechanism and write the result into the residual stream. Hence, the value of the residual stream after this operation will be the sum of each contribution and the previous value of the residual stream:

$$x_{i+1} = x_i + \sum_{j=0}^{n_{head}} h_j(x_i) \tag{1}$$

where $h_j$ represents the output of the $j$th attention head, which performs self-attention (Vaswani et al. 2017) on $x_i$. An MLP layer $m$ will project the input to a larger space, apply a non-linearity, and project back into the residual space:

$$\begin{aligned} x_{i+2} &= x_{i+1} + m(x_{i+1}) \\ &= x_{i+1} + W_2(\sigma(W_1 x_{i+1} + b_1)) + b_2 \end{aligned} \tag{2}$$

Therefore, a sequence of an attention layer followed by an MLP will be consecutively applied until the final residual stream vector $x_L$ is unembedded via a unembedding matrix $W_U \in \mathbb{R}^{d \times V}$. Performing this unembedding operation leads to the vector $y \in \mathbb{R}^{N \times V}$ where $y_{ij}$ represents the logits of

the $j$th token of the vocabulary for the prediction following the $i$th token of the sequence.

Furthermore, the computational graph of any model can be represented as a Directed Acyclic Graph (DAG), where nodes represent activations and edges represent computations between these activations. Most of the works on MI represent the models as a DAG and identify a subgraph that is responsible for the task under study, i.e. the underlying circuit.

### Activation Patching

Activation patching, first presented in Meng et al. (2022), consists on *patching* (i.e. replacing) the activations of a given component with other values that disrupt its behavior. If patching the activation of a given component does not cause a significant drop in performance, it implies that such component is not relevant to the task under study, hence enabling us to locate the circuit.

There are several versions of patching that have been used across the MI literature. In our work, we focus on two of the most common, namely *zero* and *mean* ablation. Zero ablation, which has been suggested as a gold standard for interpretability research (McGrath et al. 2023), consists on simply setting the activations of a given component to zero, whereas mean ablation (Wang et al. 2023) replaces the activations with their average value across some reference distribution.

## 3 Extracting Task-Specific Circuits from Large Language Models

As previously-mentioned, the aim of our approach is to extract a submodel from an LLM that is able to properly perform a specific task of interest. Even though there are already automated circuit discovery algorithms (ACDs) which are able to identify most of the underlying circuit for a specific task, these approaches fail to extract it so that it can be used in a standalone manner. In fact, the forward pass is typically slower than the vanilla LLM when the circuit has been identified, and naïvely pruning every component outside of the identified circuit yields disastrous results.

In this section, we present how to leverage these discovery algorithms for extracting the identified circuits enabling its independent usage. In other words, obtaining the minimal submodel (that includes the circuit) that is able to perform the specific task with a greatly reduced number of parameters and increased speed.

The pseudocode of our approach is presented in Algorithm 1. Essentially, given a LLM $f_\theta$ and a dataset that elicits the specific task of interest[1] (which is split into a patching and validation datasets $D_a$ and $D_v$), our method is able to automatically obtain a pruned model $g_\theta$ that is able to perform such task. This process will be controlled by several hyperparameters, namely the threshold $\alpha$, the type of ablation used (either zero or mean ablation) and whether or not to prune MLPs.

[1]Refer to Appendix A for a further discussion on the nature and curation of this dataset.

More specifically, we first initialize $g_\theta$ as the initial model $f_\theta$ and start traversing the different components of the LLM starting at the last layer. At each step, the current component is patched (using the patching dataset $D_a$), yielding a temporarily patched model $g'_\theta$. Then, we evaluate both $g'_\theta$ and $g_\theta$ by computing the KL divergence between their predictions and the original ones from $f_\theta$ on the validation dataset $D_v$.

The difference between the computed KL divergences $\mathrm{KL}(f_\theta \parallel g'_\theta) - \mathrm{KL}(f_\theta \parallel g_\theta)$ quantifies the effect that patching the current component has on the performance of the specific task. In other words, if the difference is large it implies that the component had an important role on the specific task, whereas a small difference implies that the component did not have a relevant impact, thus it can be discarded. Therefore, we can specify a threshold $\alpha$ and if the difference between the KL divergences is not larger, we can permanently prune the current component and update $g_\theta$.

This operation is performed for every component, but we can choose to prune only attention heads or include MLPs in the pruning process as well. Unlike current ACD methods, our approach truly prunes the model, obtaining a considerably smaller and potentially faster submodel that is able to perform the specific task under study. This was made possible by including the following modifications that differ from current ACD methods:

- If we describe the LLM as a DAG where nodes are activations and edges are operations among these activations, typical ACD methods work by patching edges instead of nodes: this enables to identify the circuit at a finer level. However, given that current accelerators excel at parallel operations, when it comes to obtaining smaller and faster submodels, it is better to prune nodes instead of edges. Even though patching nodes instead of edges provides us with a less precise circuit, ACD methods can still be applied to the resulting submodel for its interpretability.
- As previously-mentioned, current ACD methods *patch* different edges, but this does not translate to actually *pruning* parts of the LLM. Our approach truly prunes the components that are not relevant to the specific task, yielding a smaller and potentially faster submodel.

Notice that other MI works use metrics different from the KL divergence, typically derivations of the logit difference. For example, to identify the IOI circuit (Wang et al. 2023), the logit difference between the indirect object (correct name) and the subject (incorrect name) is used. However, it has been shown that the KL divergence is also able to identify the circuit (Conmy et al. 2023), with the advantage that it is a general metric that can be applied to any single-token prediction task a generally yields more robust results.

A more detailed discussion of the design choices is presented in Appendix B.

### Patching vs. Pruning a Component

Current implementations of ACD methods work by adding *hooks* into the LLM to modify (or *patch*) specific activations. This is really convenient when it comes to identifying a circuit because it enables easy and precise manipulation

**Algorithm 1:** Automatic Task-Specific Circuit Extraction

**Data:** Model $f_\theta$, patching dataset $D_a$, validation dataset $D_v$,evaluation threshold $\alpha$, $ablation\_scheme$, $include\_mlps$
**Result:** Pruned model $g_\theta$
$g_\theta \leftarrow f_\theta$
**for** $layer \leftarrow [num\_layers(f_\theta), ...0]$ **do**
  **for** $head \leftarrow [num\_heads(f_\theta), ...0]$ **do**
    $g'_\theta \leftarrow g_\theta$
    `// Temporarily ablate head`
    ablate_head$(g'_\theta, layer, head, ablation_scheme, D_a)$
    **if** *kl_div*$(f_\theta, g'_\theta, D_v) -$ *kl_div*$(f_\theta, g_\theta, D_v) < \alpha$ **then**
      `// The node is not relevant, truly prune it`
      prune_head$(g_\theta, layer, head, ablation_scheme, D_a)$
    **end**
  **end**
  `// Optionally prune MLPs`
  **if** $include\_mlps = True$ **then**
    $g'_\theta \leftarrow g_\theta$
    `// Temporarily ablate MLP`
    ablate_mlp$(g'_\theta, layer, head, ablation_scheme, D_a)$
    **if** *kl_div*$(f_\theta, g'_\theta, D_v) -$ *kl_div*$(f_\theta, g_\theta, D_v) < \alpha$ **then**
      `// The node is not relevant, truly prune it`
      prune_mlp$(g_\theta, layer, head, ablation_scheme, D_a)$
    **end**
  **end**
**end**
**return** $g_\theta$

of the internal activations. However, ablating a component via a hook is not the same as pruning it: when patching, the component is still in the model. This implies that a patched model will have the same size as the unpatched model, and the forward pass will even be slower due to the hooks.

On the other hand, pruning a component implies its removal from the original model, resulting in a smaller model with typically a faster forward pass. Our pruning approach depends on the type of ablation that is being used: patching with zero ablation implies replacing the activations of a component with a vector of zeros. As presented in Section 2, every component in the transformer architecture reads from and writes to the residual stream in an additive manner, therefore, zero ablation is equivalent to completely removing the component. Similarly, mean ablation replaces the activations with their mean value across a reference distribution. Thus, this is equivalent to replacing the component by adding a bias term to the residual stream, greatly reducing the number of parameters and computational cost. Figure 1 shows a diagram of both types of pruning.

This enables us to remove $4d \cdot d_{head}$ parameters for each attention head and $2d \cdot d_{mlp}$ parameters for each MLP, where $d$ is the dimension of the residual vector space, $d_{head}$ is the dimension where the head operates (typically $d = n_{head} d_{head}$, where $n_{head}$ is the number of heads per layer, and $d_{mlp}$ is the internal dimension of the MLP (typically $d_{mlp} > d$). Also, the inference time would also be reduced, as the computational costs are $\mathcal{O}(Nd_{head}d + N^2 d)$ and $\mathcal{O}(Nd \cdot d_{mlp})$ for an attention head and MLP respectively, where $N$ is the size of the sequence. A more detailed mathematical formulation of each pruning method is presented in the Appendix C.

## 4 Evaluation

In order to evaluate the effectiveness of our proposal, this section will be guided by the following questions:

- **RQ1:** How do the value of the threshold $\alpha$, the type of ablation and whether to prune MLPs or not affect the resulting model?
- **RQ2:** How much smaller is the pruned model? Is the resulting pruned model able to perform the specific task?
- **RQ3** Does the resulting pruned model include the circuit that performs the task under study?
- **RQ4:** How does our approach compare to a baseline distillation method?

Specifically, we will use our proposal to extract the underlying circuit of three different tasks whose circuit has been studied and manually identified in previous works:

- **Acronym Prediction** (García-Carrasco, Maté, and Carlos Trujillo 2024): The authors studied the task of 3-letter acronym prediction, e.g. `"The Chief Executive Officer ("` $\rightarrow$ `CEO`. Despite this being a multi-token task, we focus on the prediction of the third letter to provide a more illustrative comparison to the other tasks.
- **Indirect Object Identification (IOI)** (Wang et al. 2023): The authors studied the task of predicting the

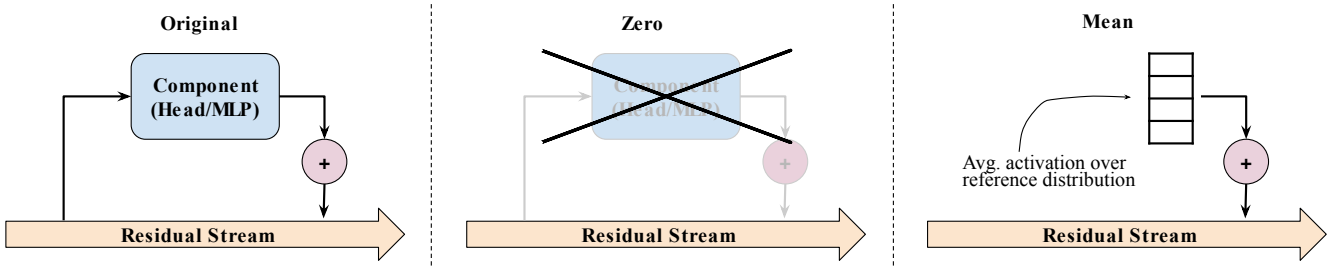


Figure 1: High-level diagram of pruning a component with zero (center) and mean (right) ablation respectively, compared with the original (left).

indirect object, e.g. `"When Mary and John went to the store, John gave a drink to " → "Mary"`.

- **Greater-than** (Hanna, Liu, and Variengien 2023): The authors studied the ability to take in sentences such as `"The war lasted from the year 1732 to the year 17"`, and predict valid two-digit end years (years $> 32$).

We have chosen these tasks because their circuits have been manually identified, therefore enabling us to compare our approach to a ground truth. Similarly, we will focus on GPT-2 Small (Radford et al. 2019), as the previous works were focused on that model.

Our method is implemented on PyTorch (Paszke et al. 2019) by using the TransformerLens (Nanda and Bloom 2022) and HuggingFace transformer (Wolf et al. 2020) libraries. The experiments were performed on a RTX4090 GPU, on an estimated total of 72 hours of compute. [2]

## RQ1: Studying the Effect of the Hyperparameters

In this section, we study and provide evidence about how the different hyperparameters affect to the resulting model, namely the value of the threshold $\alpha$, whether to perform zero or mean ablation, and whether to include MLPs or not in the process. We focus on the task of acronym prediction, but the same analysis for the other two tasks can be found in Appendix D.

Figure 2 shows the impact of the threshold $\alpha$ on the size[3] of the resulting pruned model. The colors indicate the ablation scheme used during the extraction process (e.g. mean or zero ablation) whereas the solid/dashed lines represent whether the MLPs were not included in the pruning process (i.e. all the MLPs remain in the final submodel) or are included, respectively.

The first thing to notice is that, as expected, the size of the resulting submodel decreases as $\alpha$ is set to a larger value: a larger threshold implies that for a component to be included in the final submodel it has to cause a larger increase in the KL divergence when patched.

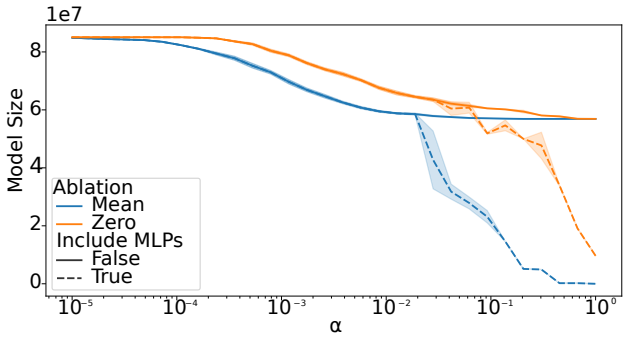


Figure 2: Impact of $\alpha$ in the resulting model size on the acronym task.

Another interesting fact is that, for the same value of $\alpha$, the resulting model when mean-ablating will be generally smaller than the obtained when zero-ablating. This can be explained by the fact that zero-ablation is more likely to send other components off-distribution; as every component reads from and writes into the residual stream, components that are at higher layers might be affected by the zero-ablated component, but not by the mean-ablated, as it is more likely to stay in-distribution. In other words, zero ablating a component generally causes a larger increase in KL divergence, hence more components are included into the final submodel.

When it comes to including the MLPs or not into the pruning process, it can be seen that there is no difference for lower values of $\alpha$, implying that in both cases, the attention heads are pruned first. Then, as $\alpha$ is increased, it reaches a certain point where the size of the resulting model drops abruptly, hinting that MLPs are starting to get pruned. In other words, this provides evidence that irrelevant attention heads are prioritized in the pruning process, and MLPs are starting to get pruned after all the irrelevant heads have been pruned.

Figure 3 shows the relationship between the size reduction w.r.t the unpruned model and accuracy for different hyperparameter setups. The first thing to notice is that pruning with zero ablation does not lead to any considerable size re-

[2]The code and data required to reproduce the experiments and figures, as well as the supplementary materials, can be found in https://github.com/jgcarrasco/circuit-extraction

[3]From now on, we do not include the parameters of the embedding/unembedding matrices, as these can be thought of as lookup tables and should not be ablated.

duction without a large drop in accuracy: we can only obtain a proper submodel with up to $20\%$ size reduction, as a larger reduction hastily drops the accuracy to zero. Moreover, the results show that zero ablation yields less consistent results and are more noisy, as different executions with a fixed $\alpha$ can give models with different size and/or accuracies.

On the other hand, when the pruning is performed via mean ablation, we are able to obtain smaller submodels that are able to preserve the accuracy and even improve it. For example, in the case where MLPs are not included in the pruning process, we are able to obtain a submodel that is $33\%$ smaller, which contains just 2 attention heads and all the MLPs and has $100\%$ accuracy in the validation set. If MLPs are included in the pruning process, we are able to reduce the size by a large margin, but at the cost of some accuracy. For example, we are able to obtain a model that is $66\%$ smaller, which contains 3 attention heads and 6 MLPs and has $83.6\%$ accuracy on the validation set.

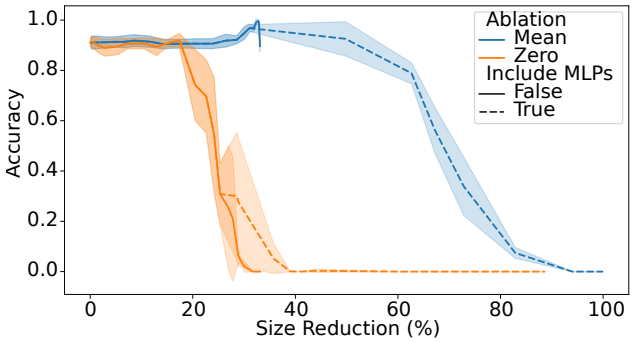


Figure 3: Accuracy vs. size of the resulting submodel on the acronym task.

Overall, the results suggest that zero ablation might be too aggressive, requiring a large amount of components to be included in the pruned model and therefore yielding larger and slower models than when using a mean ablation scheme. On the other hand, we found out that including MLPs into the pruning process can greatly reduce the size of the resulting model, but often at the cost of performance.

## RQ2: Performance and Size Comparison

Table 1 shows the accuracy as well as the reduction in the number of parameters compared to the baseline for the pruned models obtained on the three different tasks. For each task, we provide two results, one where MLPs are not included in the pruning process and one where they are. The thresholds were selected according to the results of the previous section, and mean ablation is used across all runs. In order to provide more consistent results, we report the average values over five different runs with different batches of data. The size of the batches are 250 for the acronyms and greater-than tasks and 150 for IOI. A Table with extra information about the resulting size and inference time can be found in Appendix E.

The results show that our approach yields models that are able to perform the specific tasks while preserving or even improving the performance of the original LLM (see Appendix E for extended results) while having a drastically reduced size and inference time. In general, including MLPs in the pruning process gives even better size and time reductions with negligible drops in performance. However, it seems that some tasks might be more affected than others, as we found that submodels with a reduction size of more than $60\%$ on the acronyms task gave a considerably lower performance of $78.64\%$ on average.

## RQ3: Comparison to Manually Identified Circuits

One important aspect to take into account is to check whether the pruned submodel contains the circuit that performed the specific task in the original LLM. Specifically, as shown on Table 1, we compare the attention heads present in the pruned model with the heads that were manually identified in previous MI works and computed the True Positive Rate (TPR) and False Positive Rate (FPR).

Interestingly, the TPRs are not very high, implying that there are attention heads what were manually identified as important that are not present in the pruned model. This is most likely caused by the following facts. First, there are manually identified heads that contribute very small amounts to the performance, hence they can often be discarded. For example, the authors of García-Carrasco, Maté, and Carlos Trujillo (2024) show that almost all the performance on the acronyms task can be recovered with just 2/3 heads, where the circuit is composed by 8 heads. Second, previous works might also use other kinds of ablation (e.g. resampling ablation), whereas we use mean ablation. This implies that heads that were deemed as important via other ablation schemes can be approximated by their mean output across a reference distribution (e.g. mean ablation) and therefore discarded. There are also other factors that have an effect such as the metric choice, the order of ablation, or the human factor. However, the FPRs are extremely low, suggesting that the pruned models contain the most important heads that were also discovered manually in previous works, which is actually what matters. Appendix F provides a further discussion on this, including the ROC curves for each task.

## RQ4: Comparison to a Baseline

Even though our approach is built upon a completely different motivation (i.e. extracting interpretable task-specific circuits from an already pretrained model), it is interesting to compare our results with a baseline involving task-specific model compression. Specifically, we will focus on the technique of model distillation (Ba and Caruana 2014), which enables transferring information from a teacher, larger network, to a student, smaller network.

We adopt a similar setup to Tang et al. (2019). Essentially, they train the student model to output the same same logits than the teacher model across a given training dataset of samples:

$$\mathcal{L}_{distill} = ||\mathbf{z}^{(T)} - \mathbf{z}^{(S)}||_2^2 \tag{3}$$

| Task | $\alpha$ | MLP | $acc\,(\%)$ | $\Delta param\,(\%)$ | TPR (%) | FPR (%) |
|---|---|---|---|---|---|---|
| Acronyms | $8.86 \cdot 10^{-2}$ | $False$ | $99.92 \pm 0.17$ | $32.88 \pm 0.00$ | $20.00 \pm 6.85$ | $0.30 \pm 0.40$ |
| | $3.50 \cdot 10^{-2}$ | $True$ | $78.64 \pm 5.16$ | $63.70 \pm 3.10$ | $40.00 \pm 5.59$ | $0.59 \pm 0.33$ |
| IOI | $8.53 \cdot 10^{-3}$ | $False$ | $100.00 \pm 0.00$ | $28.66 \pm 0.50$ | $57.40 \pm 1.94$ | $5.95 \pm 0.69$ |
| | $1.88 \cdot 10^{-2}$ | $True$ | $96.53 \pm 1.52$ | $72.31 \pm 2.98$ | $35.65 \pm 1.94$ | $0.99 \pm 0.37$ |
| Greater-than | $8.53 \cdot 10^{-2}$ | $False$ | $100.00 \pm 0.00$ | $32.65 \pm 0.00$ | $37.50 \pm 0.00$ | $0.00 \pm 0.00$ |
| | $8.53 \cdot 10^{-2}$ | $True$ | $99.84 \pm 0.36$ | $82.77 \pm 0.13$ | $25.00 \pm 0.00$ | $0.00 \pm 0.00$ |

Table 1: Evaluation of the pruned models obtained on each of the tasks for different values of $\alpha$, as well as their recovery rates when compared with the manually identified circuits. The reported results are averaged across five repetitions.

where $\mathbf{z}^{(T)}$ and $\mathbf{z}^{(S)}$ are the teacher and student logits, respectively. To provide a fair comparison, the teacher network will be the same GPT-2 Small pretrained model as on the previous experiments. Regarding the student network, it will have the same transformer architecture, but with a smaller number of layers. The model is trained by minimizing $\mathcal{L}_{distill}$ for a total of 20000 epochs with the Adam optimizer (Kingma 2014) and a learning rate of $10^{-3}$.

| | | **Distillation** | **Ours** |
|---|---|---|---|
| Task | $\Delta param\,(\%)$ | $acc\,(\%)$ | $acc\,(\%)$ |
| A | $32.88 \pm 0.00$ | $12.53 \pm 6.27$ | $\mathbf{99.92 \pm 0.17}$ |
| | $63.70 \pm 3.10$ | $10.66 \pm 2.89$ | $\mathbf{78.64 \pm 5.16}$ |
| IOI | $28.66 \pm 0.50$ | $2.40 \pm 0.33$ | $\mathbf{100.00 \pm 0.00}$ |
| | $72.31 \pm 2.98$ | $2.13 \pm 0.65$ | $\mathbf{96.53 \pm 1.52}$ |
| GT | $32.65 \pm 0.00$ | $77.60 \pm 32.11$ | $\mathbf{100.00 \pm 0.00}$ |
| | $82.77 \pm 0.13$ | $93.20 \pm 2.84$ | $\mathbf{99.84 \pm 0.36}$ |

Table 2: Comparison of the performance and size of the pruned model obtained with our approach and the performance of a student network with a similar size obtained by knowledge distillation from GPT-2 Small.

Table 2 presents the results of our approach versus the knowledge distillation approach for each of the three tasks of study. Specifically, for each of the previous pruned models obtained with our approach, we initialize the student model as an $N$-layer transformer such that $N$ is the largest number such that the size of the student model is less or equal than the size of our resulting pruning model. Then, we start the knowledge distillation procedure and report the largest accuracy obtained in the validation dataset.

The results clearly show that the smaller models obtained by distillation are unable to perform the specific task, with the exception of the greater-than task. The main factor behind this result is that we have used a small amount of data samples, which are not enough for training a model from scratch but sufficient to properly extract the subcircuit via our method.

## 5 Conclusion

Recent works in MI have made impressive advances in circuit identification to localize task specific behaviors on LLMs. However, current methods ae not able to extract such circuits to enable its standalone usage, thereby missing the benefits of reduced size and inference time.

In this work, we proposed a novel circuit extraction method that, given a datset that elicits a specific task of interest, automatically prunes the LLM to obtain a minimal subset capable of performing the task without additional training or fine-tuning. We extensively evaluated our method on three tasks, whose underlying circuits have already been manually identified, demonstrating that the pruned models obtained with our method are (i) able to properly perform the specific task, often better than the original LLM (ii) considerably smaller than the original LLM, with up to $82.77\%$ reductions in size and (iii) align with most results from previous literature on circuit identification, as they do not include components that were deemed as irrelevant on previous works. We also compared our approach with a distillation method and showed that the models obtained via distillation were not able to properly perform the tasks due to requiring larger amounts of training data, whereas our approach is able to extract circuits with small amounts of data.

To the best of our knowledge, this is the first work that tries to automatically extract task-specific circuits from LLMs for its standalone usage as well as the first work that proposes a MI-based pruning approach. In an era where the size of the models is getting exponentially larger, the ability to distill task-specific subsets capable of performing with significantly reduced computational overhead becomes increasingly vital. Moreover, deploying entire LLMs not only incurs unnecessary computational costs but also retains irrelevant components, exacerbating their black-box nature and potentially introducing security vulnerabilities. Our work tries to pave the way into leveraging the power of LLMs by extracting task-specific circuits that are able to perform faster, as well as being more interpretable. We believe this to be a crucial step towards more efficient, trustworthy and efficient AI systems.

Our work is limited to just one model and three fixed sequence length, single-token prediction tasks, in order to evaluate it on the current MI literature. However, these results show that extracting task-specific circuits from LLMs is possible. Future work includes extending the evaluation to more complex tasks such as multi-token prediction, generative tasks or sentiment analysis as well as using larger, production-grade LLMs.

## Acknowledgements

This work has been supported by the AETHER-UA project (PID2020-112540RB-C43), funded by Spanish Ministry of Science and Innovation, the BALLADEER (PROMETEO/2021/088) project, funded by the Conselleria de Innovación, Universidades, Ciencia y Sociedad Digital (Generalitat Valenciana), and the TSI-100927-2023-6 project, funded by the Recovery, Transformation and Resilience Plan from the European Union Next Generation through the Ministry for Digital Transformation and the Civil Service. Jorge García-Carrasco holds a predoctoral contract (CIACIF / 2021 / 454) granted by the Conselleria de Innovación, Universidades, Ciencia y Sociedad Digital (Generalitat Valenciana).

## References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Agarwal, R.; Vieillard, N.; Zhou, Y.; Stanczyk, P.; Garea, S. R.; Geist, M.; and Bachem, O. 2024. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes. In *The Twelfth International Conference on Learning Representations*.

Ba, J.; and Caruana, R. 2014. Do deep nets really need to be deep? *Advances in neural information processing systems (NeurIPS 14)*, 27.

Bhaskar, A.; Wettig, A.; Friedman, D.; and Chen, D. 2024. Finding Transformer Circuits with Edge Pruning. arXiv:2406.16778.

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, NeurIPS 2020*, 33: 1877–1901.

Conmy, A.; Mavor-Parker, A. N.; Lynch, A.; Heimersheim, S.; and Garriga-Alonso, A. 2023. Towards Automated Circuit Discovery for Mechanistic Interpretability. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Elhage, N.; Hume, T.; Olsson, C.; Schiefer, N.; Henighan, T.; Kravec, S.; Hatfield-Dodds, Z.; Lasenby, R.; Drain, D.; Chen, C.; et al. 2022. Toy models of superposition. *arXiv preprint arXiv:2209.10652*.

Elhage, N.; Nanda, N.; Olsson, C.; Henighan, T.; Joseph, N.; Mann, B.; Askell, A.; Bai, Y.; Chen, A.; Conerly, T.; et al. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1. Https://transformer-circuits.pub/2021/framework/index.html.

Frantar, E.; and Alistarh, D. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, 10323–10337. PMLR.

García-Carrasco, J.; Maté, A.; and Carlos Trujillo, J. 2024. How does GPT-2 Predict Acronyms? Extracting and Understanding a Circuit via Mechanistic Interpretability. In Dasgupta, S.; Mandt, S.; and Li, Y., eds., *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, 3322–3330. PMLR.

Gu, Y.; Dong, L.; Wei, F.; and Huang, M. 2024. MiniLLM: Knowledge Distillation of Large Language Models. In *The Twelfth International Conference on Learning Representations*.

Hanna, M.; Liu, O.; and Variengien, A. 2023. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Thirty-seventh Conference on Neural Information Processing Systems, NeurIPS 2023*.

Hanna, M.; Pezzelle, S.; and Belinkov, Y. 2024. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. *arXiv preprint arXiv:2403.17806*.

Hinton, G. E.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. *CoRR*, abs/1503.02531.

Huang, Y.; Chen, Y.; Yu, Z.; and McKeown, K. R. 2022. In-context Learning Distillation: Transferring Few-shot Learning Ability of Pre-trained Language Models. *CoRR*, abs/2212.10670.

Idelbayev, Y.; and Carreira-Perpiñán, M. 2020. Low-Rank Compression of Neural Nets: Learning the Rank of Each Layer. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 8046–8056.

Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T. B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; and Amodei, D. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Kim, J.; Lee, J. H.; Kim, S.; Park, J.; Yoo, K. M.; Kwon, S. J.; and Lee, D. 2024. Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. *Advances in Neural Information Processing Systems*, 36.

Kingma, D. 2014. Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

LeCun, Y.; Denker, J.; and Solla, S. 1989. Optimal brain damage. *Advances in neural information processing systems*, 2.

Liu, Z.; Oguz, B.; Zhao, C.; Chang, E.; Stock, P.; Mehdad, Y.; Shi, Y.; Krishnamoorthi, R.; and Chandra, V. 2023. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*.

Ma, X.; Fang, G.; and Wang, X. 2023. LLM-Pruner: On the Structural Pruning of Large Language Models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

Meng, K.; Bau, D.; Andonian, A.; and Belinkov, Y. 2022. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems, NeurIPS 2022*, 35: 17359–17372.

Nanda, N.; and Bloom, J. 2022. TransformerLens. https://github.com/TransformerLensOrg/TransformerLens.

Olah, C.; Cammarata, N.; Schubert, L.; Goh, G.; Petrov, M.; and Carter, S. 2020. Zoom in: An introduction to circuits. *Distill*, 5(3): e00024–001.

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

Park, G.; Park, B.; Kim, M.; Lee, S.; Kim, J.; Kwon, B.; Kwon, S. J.; Kim, B.; Lee, Y.; and Lee, D. 2022. Lut-gemm: Quantized matrix multiplication based on luts for efficient inference in large-scale generative language models. *arXiv preprint arXiv:2206.09557*.

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.

Povey, D.; Cheng, G.; Wang, Y.; Li, K.; Xu, H.; Yarmohammadi, M.; and Khudanpur, S. 2018. Semi-Orthogonal Low-Rank Matrix Factorization for Deep Neural Networks. In *Proc. Interspeech 2018*, 3743–3747.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language Models are Unsupervised Multitask Learners.

Santacroce, M.; Wen, Z.; Shen, Y.; and Li, Y. 2023. What matters in the structured pruning of generative language models? *arXiv preprint arXiv:2302.03773*.

Shridhar, K.; Stolfo, A.; and Sachan, M. 2023. Distilling Reasoning Capabilities into Smaller Language Models. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Findings of the Association for Computational Linguistics: ACL 2023*, 7059–7073. Toronto, Canada: Association for Computational Linguistics.

Sun, M.; Liu, Z.; Bair, A.; and Kolter, J. Z. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

Tang, R.; Lu, Y.; Liu, L.; Mou, L.; Vechtomova, O.; and Lin, J. 2019. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136*.

Thoppilan, R.; De Freitas, D.; Hall, J.; Shazeer, N.; Kulshreshtha, A.; Cheng, H.-T.; Jin, A.; Bos, T.; Baker, L.; Du, Y.; et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

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, NeurIPS 2017*, 30.

Wang, K. R.; Variengien, A.; Conmy, A.; Shlegeris, B.; and Steinhardt, J. 2023. Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 Small. In *The Eleventh International Conference on Learning Representations, ICLR 2023*.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Scao, T. L.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. M. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45. Online: Association for Computational Linguistics.

Yao, Z.; Yazdani Aminabadi, R.; Zhang, M.; Wu, X.; Li, C.; and He, Y. 2022. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35: 27168–27183.

Zhang, M.; Shen, C.; Yang, Z.; Ou, L.; Yu, X.; Zhuang, B.; et al. 2023. Pruning meets low-rank parameter-efficient fine-tuning. *arXiv preprint arXiv:2305.18403*.

Zhu, X.; Li, J.; Liu, Y.; Ma, C.; and Wang, W. 2023. A survey on model compression for large language models. *arXiv preprint arXiv:2308.07633*.