Layers to Latents: Pruning and Aligning LLMs for Efficiency and Safety

Anwesh Badapanda
2025SIY7578
ELL8299
anweshb@sit.iitd.ac.in

Mrinal Tyagi
2024AIY7614
AIL861
2024aiy7614@scai.iitd.ac.in

Abstract

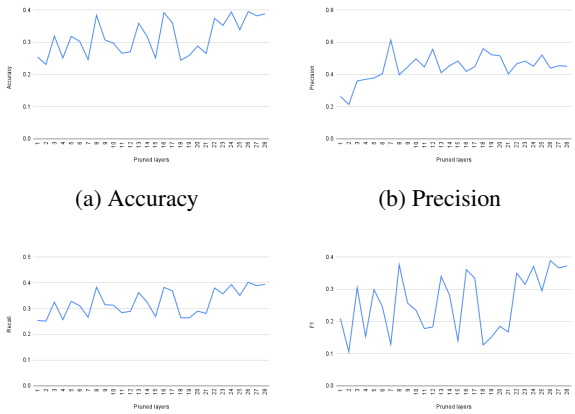
As Large Language Models (LLMs) become ubiquitous in many public facing applications, such as chat-bots, coding agents etc. it is necessary to study how we can make them more efficient *and* as well as *safe*. A model is called *efficient* if it can make fast generations while consuming less resources and maintaining generation quality. Furthermore, a *safe* model is one that blocks the generation of any unsafe/harmful/toxic content while still generating high quality responses in benign cases. This project explores various techniques, simple and advanced to enforce the above mentioned qualities in LLMs.

1 Introduction

Since the introduction of the Transformer architecture (Vaswani et al., 2017), LLMs have become commonplace in many applications in various domains. Despite their performance, two common concerns when deploying LLMs are i) How can LLMs be deployed on constrained environments, while maintaining their generation speed as well as quality **AND** ii) How can we ensure LLMs don't produce harmful or unsafe content.

In Part 2.1, we explore the answers for the first question we study two broad categories of LLM pruning (Ashkboos et al., 2024) (Ma et al., 2023). First, we explore *static pruning*, a naive pruning technique where we remove entire layers of the LLM sequentially to observe which layer has the most impact on generation quality. Next, instead of pruning entire layers irrespective of their content, we do *dynamic pruning*, where instead of passing the whole sequence to every head, we attempt to learn a routing network, which sends every token to a different head, following a similar approach to (Jin et al., 2024)

In Part 2.2, we discuss our findings from attempts at finding out how harmful content is stored in LLMs, and if, at all, such content can be removed from the model's internal states. For this task we take a frozen GPT-2-(L) (Radford et al., 2019) model (M), and train individual MLP models on the output activations of each layer. We then evaluate these MLP models on output activations produced by M on the test dataset, to try and identify which layer stores the most discriminative features to identify toxicity. Further, based on these findings, we try to erase the harmful vector v_{harm} from the most discriminative activation outputs, in an attempt to find out if harmful content can be removed from the model without editing the input text.



(a) Accuracy (b) Precision

(c) Recall (d) F1

Figure 1: Visualization of change in performance of accuracy, precision, recall, and F1 when different layers are pruned using Qwen3 0.6B on the MMLU dataset.

2 Task 1

2.1 Static Pruning

We experiment with static pruning using two strategies:

1. Layer-based structured pruning
2. Magnitude-based unstructured pruning

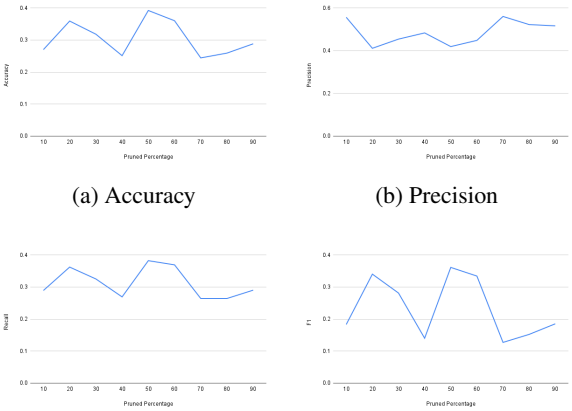
Layer	Accuracy	Precision	Recall	F1
1	0.254	0.264	0.254	0.209
2	0.231	0.214	0.251	0.106
3	0.319	0.359	0.325	0.305
4	0.251	0.370	0.257	0.153
5	0.318	0.378	0.328	0.299
6	0.303	0.404	0.311	0.246
7	0.246	0.614	0.266	0.129
8	0.384	0.398	0.383	0.376
9	0.307	0.446	0.315	0.257
10	0.297	0.498	0.313	0.235
11	0.266	0.446	0.284	0.178
12	0.270	0.556	0.289	0.183
13	0.359	0.411	0.362	0.340
14	0.318	0.454	0.325	0.281
15	0.251	0.483	0.269	0.140
16	0.392	0.419	0.382	0.361
17	0.360	0.448	0.369	0.334
18	0.244	0.560	0.264	0.127
19	0.259	0.522	0.264	0.152
20	0.288	0.516	0.290	0.185
21	0.265	0.403	0.281	0.167
22	0.374	0.466	0.380	0.350
23	0.352	0.483	0.357	0.315
24	0.394	0.451	0.393	0.371
25	0.339	0.521	0.351	0.295
26	0.395	0.439	0.402	0.389
27	0.382	0.454	0.389	0.366
28	0.388	0.451	0.394	0.372
Zeroshot	0.388	0.451	0.394	0.372

Table 1: Comparison table for Layer Pruning using Qwen3 0.6B on the MMLU dataset.

Table-1 & Figure-1 highlight the variation in performance as different layers are pruned i.e. structured pruning using Qwen3 0.6B model on the MMLU dataset. As we can observe, the lower layers contain important information, which when pruned results in performance degradation.

Pruned Percentage	Accuracy	Precision	Recall	F1
10	0.270	0.556	0.289	0.183
20	0.359	0.411	0.362	0.340
30	0.318	0.454	0.325	0.281
40	0.251	0.483	0.269	0.140
50	0.392	0.419	0.382	0.361
60	0.360	0.448	0.369	0.334
70	0.244	0.560	0.264	0.127
80	0.259	0.522	0.264	0.152
90	0.288	0.516	0.290	0.185
Zeroshot	0.388	0.451	0.394	0.372

Table 2: Comparison table for Magnitude Pruning using Qwen3 0.6B on the MMLU dataset



(a) Accuracy (b) Precision

(c) Recall (d) F1

Figure 2: Visualization of change in performance of accuracy, precision, recall, and F1 when different percentages of parameters are pruned using Qwen3 0.6B on the MMLU dataset.

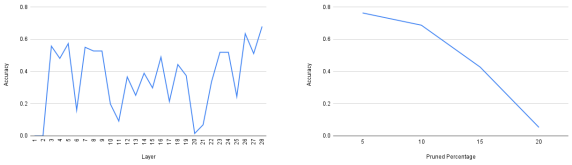


Figure 3: Visualization of change in performance of accuracy when different layers (left) and different magnitudes (right) are pruned using Qwen3 0.6B on the GSM8K dataset.

Table-2, Table-4 & Figure-2 highlight the variation in performance as different percentage of parameters are pruned i.e. unstructured pruning using Qwen3 0.6B model on the MMLU dataset. As we can observe, the higher pruning percentage prunes important weights, resulting in degradation of performance.

We perform a similar comparison for structured and unstructured pruning using the GSM8K dataset using Qwen3 0.6B model. Table-3 and Figure-3 demonstrate the change in performance as we perform structured and unstructured pruning. As expected, the lower layers contain more information than the higher layers, which is concluded by the experiments as well.

2.2 Dynamic Pruning

In this section, we train a simple one-layer MLP as a router network to identify which head we should focus on and which should not be focused on in the MultiHead Self-Attention module. For this task, we utilise the Llama-3.2-1B Instruct model and train the router network with a load balancing loss

Figure 4: WandB SFT visualisation for training and validation with MoH on Llama 3.2 1B Instruct using MMLU dataset.

Layer	Accuracy	Layer	Accuracy
1	0.000	15	0.298
2	0.000	16	0.489
3	0.557	17	0.214
4	0.481	18	0.443
5	0.573	19	0.374
6	0.160	20	0.015
7	0.550	21	0.069
8	0.527	22	0.336
9	0.527	23	0.519
10	0.198	24	0.519
11	0.092	25	0.244
12	0.366	26	0.634
13	0.252	27	0.511
14	0.389	28	0.679
Zeroshot	0.679		

Table 3: Layer-wise pruning accuracy for Qwen3 0.6B on the GSM8K dataset.

Pruned Percentage	Accuracy
5	0.763
10	0.687
15	0.427
20	0.053
Zeroshot	0.679

Table 4: Comparison table for Magnitude Pruning using Qwen3 0.6B on the GSM8k dataset

as described in (Jin et al., 2024) on MMLU dataset.

The load balancing loss is given as follows:

$$\mathcal{L}_{\text{load}} = \sum_{i=h_s+1}^{h} P_i f_i,$$

$$P_i = \frac{1}{T} \sum_{t=1}^{T} \text{Softmax}(W_r x_{\,i-h_s}),$$

$$f_i = \frac{1}{T} \sum_{t=1}^{T} \mathbf{1}[\text{Token } x_t \text{ selects head } i]\,.$$

where, $\mathbf{1}$ stands for the indicator function stands. We perform a quantitative analysis of the Llama-3.2-1B Instruct model with and without a router network. We only utilise the top 75% heads which results in compression of usage of heads.

Experimental Setting	Accuracy	Precision	Recall	F1
Zeroshot	0.381	0.419	0.381	0.371
Finetuned with MoH	0.260	0.266	0.263	0.260

Table 5: Comparison table for MMLU dataset using Llama 3.2 1B Instruct.

Table-5 compared the performance of the Llama 3.2 1B Instruct model in ZeroShot and Finetuned with MoH setting. Supervised finetuning with MoH results in decrease of performance as the reasoning that is present in the zeroshot model is lost and model tries to memorize more rather than generalising. Figure-4 showcases the training and evaluation curves for the supervised fine tuning performed using the load balancing loss. A further

interesting approach could be to finetune the model with reasoning on the same dataset with Mixture of Heads approach so that the reasoning information is incorporated as well. The plots are directly imported from WandB for visualisation.

3 Aligining a Model Towards Safety

3.1 Dataset Creation

For this part of our experiments we choose the Jigsaw Toxic Comment Dataset (cjadams et al., 2017) (*text*, *label*) pairs, where each label vector is composed of 6 individual binary labels for classes (*toxic*, *severe_toxic*, *obscene*, *threat*, *insult*, *identity_hate*). For our experiments we convert it to a single label vector, by setting the final label as 1, if any of the original labels are 1, else 0.

3.2 Safety Alignment

3.2.1 Identifying Target Layer

First, we try to identify, if there certain layers, learn more discriminative features about harmful content than others. A straightforward way to test this hypothesis, was to collect outputs of each layer and train simple 2-layer binary classification MLP models on them, and classify output activations produced by the model on the test set of the dataset.

As visible in Table-6, the activations of layer 13 give us the best classification **F1 score**. However, it is noticeable that the variation of this score among various layers is quite low. The same thing can be said about the classification accuracy metric. The **True Positive Rate (TPR)** and **False Positive Rate (FPR)** Metric show a little more variation across various layers. Layer 0 (the layer output) by itself performs decent enough features to show reasonable performance across various metrics, even outperforming outputs of some of the transformer layers.

Why? A possible reason could be as follows: in most cases the presence of harmful content in a string of text, can be easily detected through the presence of individual harmful tokens. Perhaps, this is the reason that the outputs of the embedding layer on their own can provide good results. While further layers make the toxic manifold more separable, due to word level identification, the embedding layer on it's own can provide high separability.

Next, building upon the results of the above experiment, we will attempt to use the best performing layers from the above experiment to remove harmful content from the model, using two techniques. *First*: using Deep Alignment (Zhang et al., 2025), *Second*: by extracting the weight vector from a Logistic Regression model and unit normalizing it.

layer_idx	f1	accuracy	tpr	fpr
0	0.870	0.865	0.904	0.174
1	0.880	0.873	0.930	0.185
2	0.879	0.873	0.921	0.174
3	0.878	0.877	0.887	0.133
4	0.882	0.879	0.906	0.148
5	0.879	0.868	0.953	0.217
6	0.875	0.875	0.876	0.125
7	0.883	0.880	0.903	0.142
8	0.880	0.880	0.875	0.114
9	0.887	0.884	0.917	0.150
10	0.884	0.881	0.907	0.145
11	0.889	0.884	0.927	0.159
12	0.888	0.887	0.897	0.123
13	**0.890**	0.882	**0.954**	0.191
14	0.889	0.887	0.906	0.132
15	0.873	0.876	0.852	**0.099**
16	0.888	**0.887**	0.892	0.118
17	0.884	0.884	0.891	0.124
18	0.884	0.884	0.884	0.116
19	0.886	0.885	0.894	0.124
20	0.886	0.883	0.907	0.140
21	0.888	0.885	0.914	0.145
22	0.886	0.883	0.910	0.143
23	0.885	0.881	0.919	0.158
24	0.884	0.883	0.896	0.131
25	0.879	0.879	0.882	0.125
26	0.880	0.876	0.903	0.150
27	0.877	0.876	0.881	0.129
28	0.876	0.870	0.925	0.186
29	0.882	0.872	0.949	0.205
30	0.875	0.876	0.872	0.121
31	0.878	0.877	0.887	0.133
32	0.876	0.869	0.924	0.186
33	0.874	0.863	0.951	0.225
34	0.873	0.862	0.949	0.224
35	0.874	0.871	0.898	0.156
36	0.875	0.871	0.904	0.161

Table 6: GPT-2 Layer-wise MLP Metrics for Detecting Unsafe Content on Jigsaw Toxic Comment Dataset

Every layer's hidden layer is *sanitized* i.e. cleaned of harmful/toxic contents using the following formulations:

To extract the harmful vector as demonstrated in Deep Alignment:

$$\mathbf{v}'_l = \frac{1}{N}\sum_{i=1}^{N} \mathbf{a}_{i,l}^{\text{malicious}} - \frac{1}{M}\sum_{j=1}^{M} \mathbf{a}_{j,l}^{\text{benign}} \quad (1)$$

Alternatively, we also experiment with Logistic Regression weight vectors

$$\mathbf{v}'_l = \frac{\mathbf{w}_{\text{logistic}}}{\|\mathbf{w}_{\text{logistic}}\|_2} \quad (2)$$

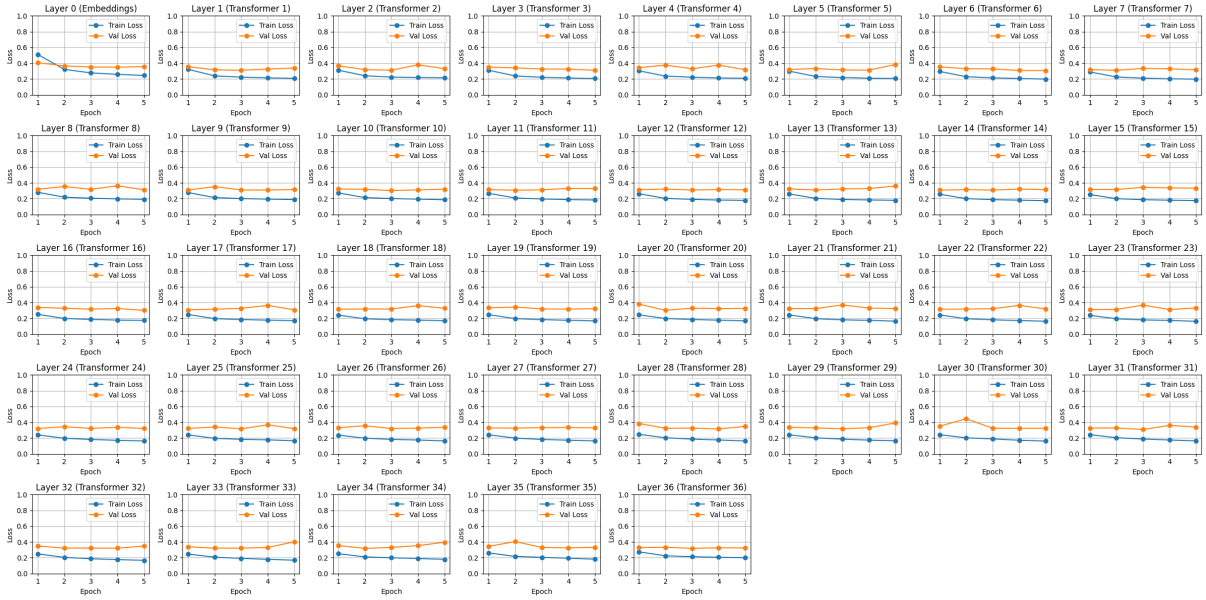


Figure 5: MLP Training Loss Curves for Harmful Content Classification using GPT-2 activations

Finally, the sanitized hidden state is obtained using

$$\mathbf{h_l} = \mathbf{h_l} - \alpha(\mathbf{h_l} \cdot \mathbf{v_l})\mathbf{v_l} \tag{3}$$

where $a_{i,l}$ is the activation output of the l^{th} layer for the i^{th} sample and where h_l is the layer's hidden state and α is a hyperparameter indicating how much of the harmful vector we want to remove from the hidden state.

n_layers	α	F1	Accuracy	FPR	TPR
1	0	0.8885	0.8798	0.1985	0.9580
1	0.01	0.8888	0.8808	0.1911	0.9527
1	0.05	0.8760	0.8709	0.1704	0.9122
1	0.1	0.8197	0.8227	0.1608	0.8062
1	0.5	0.4339	0.5270	0.3085	0.3625
1	1	0.3223	0.4477	0.3673	0.2627
5	0	0.8885	0.8798	0.1985	0.9580
5	0.01	0.8871	0.8802	0.1810	0.9414
5	0.05	0.7799	0.7912	0.1573	0.7397
5	0.1	0.6223	0.6689	0.2076	0.5454
5	0.5	0.3573	0.4571	0.3876	0.3018
5	1	0.2931	0.4676	0.2856	0.2207
10	0	0.8885	0.8798	0.1985	0.9580
10	0.01	0.8850	0.8785	0.1780	0.9350
10	0.05	0.7410	0.7597	0.1685	0.6878
10	0.1	0.5707	0.6239	0.2521	0.4999
10	0.5	0.3733	0.4360	0.4639	0.3359
10	1	0.3022	0.4889	0.2435	0.2214

Table 7: Performance using Deep Alignment method across different numbers of layers and α.

The results for both the approaches are shown in Table-7 and Table-8 respectively.

n_layers	α	F1	Accuracy	FPR	TPR
1	0	0.8885	0.8798	0.1985	0.9580
1	0.01	0.8885	0.8798	0.1986	0.9582
1	0.05	0.8886	0.8799	0.1986	0.9584
1	0.1	0.8889	0.8801	0.1994	0.9596
1	0.5	0.8861	0.8759	0.2130	0.9649
1	1	0.8740	0.8602	0.2491	0.9696
5	0	0.8885	0.8798	0.1985	0.9580
5	0.01	0.8887	0.8800	0.1983	0.9584
5	0.05	0.8889	0.8800	0.1997	0.9598
5	0.1	0.8885	0.8794	0.2023	0.9611
5	0.5	0.8790	0.8670	0.2323	0.9662
5	1	0.8674	0.8517	0.2670	0.9704
10	0	0.8885	0.8798	0.1985	0.9580
10	0.01	0.8887	0.8799	0.1985	0.9584
10	0.05	0.8886	0.8796	0.2007	0.9600
10	0.1	0.8880	0.8787	0.2042	0.9617
10	0.5	0.8758	0.8629	0.2409	0.9667
10	1	0.8630	0.8459	0.2792	0.9710

Table 8: Performance of Logistic Regression Classifier Weights across n_layers and α values.

The layers to be acted on are selected based on their **F1** score. The harmful content vector is removed from the top n performing layers for our experiments.

It is visible that the Deep Alignment method for harmful vector extraction clearly outperforms the logistic regression method. While more drastic for the Deep Alignment method, for all values of n_layers, as we increase α the **F1** value decreases. While for the Deep Alignment method the maximum drop in F1 is 0.57, for the Logistic

Regression Weights it only 0.01. Another interesting observation is that with increase in alpha the **TPR** decreases drastically for the Deep Alignment method (which is the expected behaviour), however using the Logistic Regression weights results in a marginal increase of **TPR**.

References

Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*.

cjadams, Jeffrey Sorensen, Julia Elliott, Lucas Dixon, Mark McDonald, nithum, and Will Cukierski. 2017. Toxic comment classification challenge. https://kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge. Kaggle.

Peng Jin, Bo Zhu, Li Yuan, and Shuicheng Yan. 2024. Moh: Multi-head attention as mixture-of-head attention. *arXiv preprint arXiv:2410.11842*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Yitong Zhang, Jia Li, Liyi Cai, and Ge Li. 2025. Davsp: Safety alignment for large vision-language models via deep aligned visual safety prompt. *arXiv preprint arXiv:2506.09353*.