Chip-Tuning: Classify Before Language Models Say

Anonymous ACL submission

Abstract

The rapid development in the performance of large language models (LLMs) is accompanied by the escalation of model size, leading to the increasing cost of model training and inference. Previous research has discovered that certain layers in LLMs exhibit redundancy, and removing these layers brings only marginal loss in model performance. In this paper, we adopt the probing technique to explain the layer redundancy in LLMs and demonstrate that language models can be effectively pruned with probing classifiers. We propose chip-tuning, a simple and effective structured pruning framework specialized for classification problems. Chip-tuning attaches tiny probing classifiers named chips to different layers of LLMs, and trains chips with the backbone model frozen. After selecting a chip for classification, all layers subsequent to the attached layer could be removed with marginal performance loss. Experimental results on various LLMs and datasets demonstrate that chip-tuning significantly outperforms previous state-of-the-art baselines in both accuracy and pruning ratio, achieving a pruning ratio of up to 50%. We also find that chip-tuning could be applied on multimodal models, and could be combined with model finetuning, proving its excellent compatibility.

1 Introduction

Large language models (LLMs) have experienced rapid development in recent years, achieving surprising success in various domains. Researchers have been scaling up the size of language models to pursue better performance, just as the scaling law (Kaplan et al., 2020) suggests. However, the increasing size of models leads to massive computational costs, posing a challenge to practical deployment and usage.

Model compression techniques have since been proposed as a solution to relieving computational stress, which would assist in the deployment of large models. Different approaches have been explored to compress language models into more compact versions, including quantization (Liu et al., 2021; Dettmers et al., 2022, 2024), knowledge distillation (Gou et al., 2021; Gu et al., 2023; Ko et al., 2024) and pruning (Ma et al., 2023; Yang et al., 2024; Ashkboos et al., 2024; Men et al., 2024).

Relevant research (Men et al., 2024) reveals that a fair portion of parameters in large language models are redundant, and removing these parameters would not bring severe damage to the performance of models. Based on the observation, different methods have been designed to identify and remove redundant parameters from LLMs, like layer merging (Yang et al., 2024), width compression (Ashkboos et al., 2024), layer removal (Men et al., 2024) and component removal (Ma et al., 2023). These methods maintain the majority of performance, proving the feasibility of model pruning.

Research on model interpretability has shown evidence that language models may develop internal representations for various features like color (Patel and Pavlick, 2022), truthfulness (Burns et al., 2022), chessboard states (Nanda et al., 2023), numbers (Zhu et al., 2024) or even abstract concepts like code errors (Templeton, 2024). These features typically start to form on middle layers and will be carried to subsequent layers (Stolfo et al., 2023). More interestingly, many of these features can be read out by probing techniques (Belinkov, 2022), in the way of training simple classifiers.

Inspired by the discovery that removing late layers of LLMs does not heavily impair network functionality (Men et al., 2024), we hypothesize that the critical features for solving certain problems may begin to form on intermediate layers of LLMs. By probing these necessary features on intermediate layers, we can safely prune subsequent layers with marginal performance loss.

We observe that previous research mainly aimed to build a general pruned model that can be directly

applied to various downstream tasks. Based on the intuition that different tasks require different subsets of features, we hypothesize that pruning on specific tasks instead of pruning for a general model would yield better results, as the model could better focus on the related features.

In this paper, we introduce **chip-tuning**, a simple and effective structured pruning framework specialized for classification tasks. For a given classification task, we attach probing classifiers named chips to each layer of the language model, and train these classifiers to probe the final classification results from intermediate hidden states. After training, we can then select a chip with satisfactory accuracy, and prune all layers subsequent to the chip to obtain a more compact model for the task. The parameters of the backbone model are frozen throughout the whole process and will not introduce any additional computation cost.

We apply chip-tuning to language models with different sizes and families and observe their performance on various classification tasks. Compared with previous pruning methods, chip-tuning demonstrates better performance on classification tasks, and enables more radical pruning that reduces the parameters of models by up to 50% with marginal loss in performance. Additional experiments show that chip-tuning is also compatible with multimodal large language models (MLLMs) and other finetuning methods.

The main contributions of our paper can be summarized as:

- We propose chip-tuning, a pruning framework for classification tasks that trains probing classifiers attached to certain layers of language models. By removing layers subsequent to the selected classifier, we can effectively reduce the size of the models.

- We conduct experiments on different benchmarks, experimental results show that Chip-tuning is able to maintain the performance while reducing the size of models by up to 50%, much outperforming previous state-of-the-art baselines.

- We evaluate chip-tuning on multimodal models and finetuned models, whose results prove the excellent compatibility of chip-tuning.

2 Related Work

Network Pruning. With the growth in the size of language models, the pruning technique has been proposed to eliminate unnecessary weights or structures in language models, thus accelerating language models. The pruning methods can be generally categorized into two types: unstructured pruning and structured pruning.

Unstructured pruning methods focus on the level of individual weights, which try to speed up models by increasing the sparsity level of model weights. SparseGPT (Frantar and Alistarh, 2023) reduces the pruning problem to layer-wise sparse regression and incrementally prunes each column in the weight matrix with a sequence of Hessian inverses. Wanda (Sun et al., 2023) enhances the magnitude pruning approach with input activation norms, effectively reducing the complexity of pruning algorithms. RIA (Zhang et al., 2024a) notices that previous methods tend to prune away entire channels of network weights, and mitigates the issue by jointly considering input and output channels.

Structured pruning methods operate at the level of network structures instead, which compress language models by removing redundant model components. LLMPruner (Ma et al., 2023) employs gradient information as a reference to remove non-critical structures. SliceGPT (Ashkboos et al., 2024) removes rows or columns corresponding to small principal components in the weight matrix to achieve smaller weight matrices. LaCo (Yang et al., 2024) proposes the layer collapse algorithm, which merges adjacent layers while ensuring the representation similarity on few-shot calibration examples. ShortGPT (Men et al., 2024) finds that deep layers of language models are not as effective as expected, and proposes the block importance metric to identify and remove redundant layers. BlockPruner (Zhong et al., 2024) decomposes each Transformer layer into two minimal residual blocks and performs fine-grained block pruning to avoid significant performance loss.

Probing Language Models. The impressive capability of language models raises the hypothesis that language models have gone beyond mere memorization of surface correlations. Instead, they may learn the principles behind the training data and develop internal representations for features (Belinkov, 2022). A wide variety of features have been detected in the hidden state of language models like color (Patel and Pavlick, 2022) and truthful-

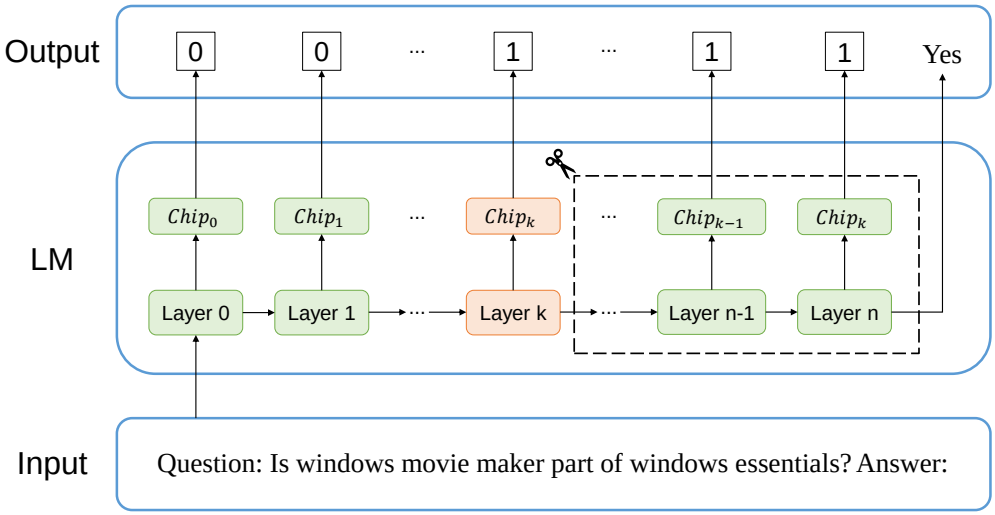


Figure 1: The overall structure of chip-tuning. After selecting a certain chip attached to the k-th layer, subsequent layers in the language model can be safely pruned with marginal influence on model performance. In training, only the parameters of chips are trainable and the backbone model is frozen.

ness (Burns et al., 2022).

Probing is a widely adopted technique to associate internal representations with external properties (Belinkov, 2022). By training a simple classifier on model representations that predicts a given property, we can read out various features before language models generate their final outputs. With a simple linear classifier, probing is able to extract complex features like board game states (Nanda et al., 2023), entity properties (Li et al., 2021), and spatial information (Gurnee and Tegmark, 2023).

Recently, Tao et al. (2024) finds that probing classifiers are also able to extract cross-modal information from multimodal large language models. Zhang et al. (2024b) further reveals that probing could achieve better performance on image classification tasks even than directly finetuning the backbone models.

An interesting discovery is that probing classifiers sometimes achieve the best performance at intermediate layers, rather than early or late layers (Zhu et al., 2024). A hypothesis is that late layers focus more on local features related to the next token prediction, while intermediate layers gather information in the input text and thus contain more global information (Stolfo et al., 2023).

3 Methodology

We illustrate the structure of the chip-tuning framework in Figure 1. The framework first inserts simple probing classifiers named chips to different layers of language models, and then solely trains the chips on task-specific training data.

Finally, we can select the chip on a fixed layer or with other strategies (see Section 5.2), and layers subsequent to the attached layer will be removed.

3.1 Chips

A language model with the decoder-only structure consists of L transformer layers. At every token position t, each transformer layer l takes previous partial sequence $x_{\leq t}$ as input and outputs new hidden states x_t^l.

As discovered by previous research on probing, the hidden states of intermediate layers may contain rich features that can be read out by probing classifiers. Chips are simple probing classifiers that try to predict the classification label y from certain hidden states x_t^l.

We use two types of chips in our experiments: linear chips p_L and two-layer perceptron (2xMLP) chips p_M, whose function could be notated as:

$$p_L(x_t^l) = \text{softmax}(Wx_t^l + b)$$

$$p_M(x_t^l) = \text{softmax}(W_1\text{ReLU}(W_2x_t^l + b_2) + b_1$$

where W, W_1, W_2, b, b_1 and b_2 are trainable parameters.

For simplicity, we take the hidden state at the last token position (i.e. $t = -1$) as the input vector

of chips.

3.2 Training

As the optimal layer l^* for classification chips is initially unknown, we attach a chip p^l to every layer l of the language model, and train these chips simultaneously with standard cross-entropy loss:

$$\mathcal{L}^l = y \log p(x_t^l) + (1-y)\log(1-p(x_t^l))$$

$$\mathcal{L} = \sum_{i=0}^{L} L^l$$

Note that the parameters in the backbone language model are frozen in the training process, and only the weights of chips would be updated.

3.3 Layer Removal and Inference

We use the straightforward layer removal method to reduce the size of language models. After selecting chip p^l at layer l as the classification chip, we simply remove all layers after layer l to obtain a smaller model.

Namely, with chip p^l at layer l finally selected, the pruned model would function as follows:

Algorithm 1 Inference with Chips

Input: Language model M with N layers $L_0, L_1, \ldots, L_{N-1}$, selected chip p^l at layer l, input embedding x^{-1};

Output: Classification prediction y;

1: **for all** $i = 0, 1, 2, \ldots, l$ **do**
2: $\quad x^i = L_i(x^{i-1})$
3: **end for**
4: $y = \arg\max(p^l(x_t^i))$

Chip-tuning also supports inference with multiple chips $p_0, p_1, \ldots, p_{n-1}$ for different tasks. Assuming that the deepest layer with chips attached is $l_m = \max(l_0, l_1, \ldots, l_{n-1})$, layers subsequent to l_m will be pruned.

4 Experiments

4.1 Experimental Setup

Benchmarks. We select 4 distinct benchmarks on natural language processing with the form of multi-choice for evaluation: MMLU (Hendrycks et al., 2020), Race (Lai et al., 2017), BoolQ (Clark et al., 2019) and C3 (Sun et al., 2020).

Furthermore, we introduce three image classification datasets to test the effectiveness of chip-tuning on multimodal large language models (MLLMs): Flowers102 (Nilsback and Zisserman, 2008), StanfordCars (Krause et al., 2013), and Caltech101 (Fei-Fei et al., 2004), each containing 102, 196, and 101 classes respectively.

Models. Following previous work (Men et al., 2024), we choose 2 model series to evaluate the effectiveness of chip-tuning: Llama2 (Touvron et al., 2023), Baichuan2 (Yang et al., 2023), which share similar decoder-only transformer structure. We use the 7B and 13B versions of Llama2 and Baichuan2 for experiments. For multimodal large language models, we use the 7B and 13B versions of LLaVA-1.5 (Liu et al., 2023) as the backbone model.

Due to memory constraints, we run 13B models under the precision of 16-bit (fp16) instead of 32-bit (fp32).

Baselines. We compare our method with several structured pruning methods: **LLMPruner** (Ma et al., 2023) removes non-critical coupled structures on the basis of gradient information. **SliceGPT** (Ashkboos et al., 2024) replaces weight matrices with smaller matrices by retaining principal components. **LaCo** (Yang et al., 2024) merges the layer in language models from deep to shallow, and sets a threshold to prevent excessive merging. **ShortGPT** (Men et al., 2024) removes redundant layers according to their proposed Block Influence metric, a variant of cosine similarity.

Settings. For each benchmark, we use at most $20,000$ training data in the corresponding training split of the benchmark to train our chips. The chips are trained with a batch size of 1 for 1 epoch. We use a learning rate of 1×10^{-5} for our experiments, and set the hidden dimension of MLP chips to 256. All experiments are conducted on a single NVIDIA A100 40GB GPU.

For 7B models, we select chips at layer 20 as classification chips; for 13B models, we select chips at layer 25 as classification chips. These settings are equal to the prune ratio of 34.4% and 35.0%, respectively.

4.2 Main Experiment Results

To evaluate the effectiveness of chip-tuning, we conduct experiments on multi-choice style benchmarks commonly used for large language model evaluation. The experimental results are demonstrated in Table 1. [1]

[1] We report the result of finetuning pruned baseline models with the same data used by chip-tuning in Appendix F.

Model	Method	Ratio (%)	BoolQ	Race-H	Race-M	C3	MMLU	Avg. Score
Llama2-7B	Dense	0.00%	71.62	35.71	34.19	43.56	45.39	46.09
	LLMPrun.	27.0%	55.20	22.56	22.35	25.64	23.33	29.82
	SliceGPT	26.4%	38.32	21.07	21.66	39.78	28.92	29.95
	LaCo	27.1%	64.07	22.61	23.61	39.67	26.45	35.28
	ShortGPT	27.1%	74.71	32.25	35.17	39.62	43.96	45.14
	CT (Lin.)	34.4%	79.05	47.91	53.69	48.93	44.89	54.89
	CT (MLP)	34.4%	76.01	49.43	53.90	53.80	45.07	55.64
	CT (Max)	~40%	**79.48**	**50.34**	**54.74**	**54.35**	**45.49**	**56.88**
Llama2-13B	Dense	0.00%	82.39	57.95	60.38	47.51	55.00	60.65
	LLMPrun.	24.4%	56.42	22.47	22.08	32.33	25.21	31.70
	SliceGPT	23.6%	37.86	23.41	24.03	41.92	37.14	32.87
	LaCo	24.6%	63.98	54.49	56.55	44.93	45.93	53.18
	ShortGPT	24.6%	62.48	58.35	60.17	46.90	**54.69**	56.52
	CT (Lin.)	35.0%	78.23	62.04	67.06	68.21	52.79	65.67
	CT (MLP)	35.0%	75.81	62.52	67.13	68.00	52.95	65.28
	CT (Max)	~50%	**79.76**	**63.29**	**68.04**	**69.39**	53.41	**66.78**
Baichuan2-7B	Dense	0.00%	74.10	26.96	24.09	64.55	53.87	48.71
	LLMPrun.	24.2%	61.19	21.96	22.28	41.64	24.93	34.40
	SliceGPT	22.2%	39.30	23.53	22.49	26.58	25.18	27.42
	LaCo	24.2%	56.15	28.99	27.72	50.85	31.53	39.05
	ShortGPT	24.2%	67.83	53.26	46.76	56.33	45.77	53.99
	CT (Lin.)	34.4%	72.78	62.69	66.85	75.47	51.09	65.78
	CT (MLP)	34.4%	73.12	63.52	67.13	76.36	50.95	66.22
	CT (Max)	~40%	**74.68**	**64.04**	**68.38**	**76.36**	**51.22**	**66.94**
Baichuan2-13B	Dense	0.00%	77.89	67.27	68.94	65.64	59.50	67.85
	LLMPrun.	24.3%	56.54	21.17	21.61	39.89	23.19	32.48
	SliceGPT	22.8%	37.83	21.56	21.52	24.99	22.95	25.77
	LaCo	24.7%	62.35	56.92	57.80	61.10	51.35	57.90
	ShortGPT	24.7%	62.54	55.77	56.41	60.16	52.11	57.40
	CT (Lin.)	35.0%	77.77	73.04	77.44	80.84	56.88	73.19
	CT (MLP)	35.0%	76.88	73.87	77.44	81.81	56.66	73.33
	CT (Max)	~50%	**78.84**	**75.04**	**79.11**	**81.89**	**56.96**	**74.37**

Table 1: Comparison of pruning methods on natural language benchmarks. CT refers to chip-tuning (our method). The results of LLMPrun., SliceGPT, LaCo, and ShortGPT are reported from ShortGPT (Men et al., 2024).

Chip-tuning excels previous baselines. It can be clearly observed that chip-tuning outperforms previously structured pruning baselines on almost every benchmark by a large margin, proving the capacity of our proposed model. Meanwhile, while previously structured pruning baselines prune less than 30% of the model parameters, chip-tuning is able to prune models by a higher ratio: 34.4% for 7B models and 35.0% for 13B models.

Linear chips are sufficient for classification. We also notice that the performance of linear chips is close to the performance of MLP chips, indicating that the features related to the input question may be mostly encoded linearly, and linear probing classifiers are enough for reading out these features. Details of the difference will be demonstrated in Section 5.1.

Optimal chips exhibit more potential. Finally, we gather the highest accuracy of all chips on each benchmark, notated as CT (max) in the table. The pruning ratio and corresponding layer of optimal chips varies across different benchmarks and models (see Appendix B for details). By choosing the optimal chip, chip-tuning could achieve even higher pruning ratios and performance.

4.3 Pruning Multimodal Models

We further evaluate whether chip-tuning could be applied to multimodal large language models (MLLMs) by pruning LLaVA-1.5 on image classification benchmarks. Following the settings in

Model	Method	Ratio(%)	Flowers102	StanfordCars	Caltech101	Avg. Score
Llava1.5-7B	Raw	0.00%	5.9	0.0	47.1	17.67
	w/ Label	0.00%	10.2	0.0	62.1	24.10
	CT (Lin.)	34.4%	91.28	60.98	92.24	81.50
	CT (MLP)	34.4%	88.70	0.85	91.52	60.36
	CT (Max)	~20%	**94.00**	**70.95**	**92.24**	**85.73**
Llava1.5-13B	Raw	0.00%	5.3	0.0	49.9	18.4
	w/ Label	0.00%	7.2	0.1	70.9	26.07
	CT (Lin.)	50.0%	91.46	48.63	91.70	77.26
	CT (MLP)	50.0%	85.93	0.85	90.42	59.07
	CT (Max)	~50%	**93.06**	**71.52**	**92.39**	**85.66**

Table 2: Comparison of pruning methods on image classification benchmarks. CT refers to chip-tuning (our method). The results of dense models are reported from Zhang et al. (2024b).

(Zhang et al., 2024b), we train the chips for 500 epochs with a learning rate of 1×10^{-3}, and set the batch size to 512.

Table 2 demonstrates the results of pruning (see Appendix C for details). Surprisingly, the original LLaVA models perform poorly on fine-grained image classification tasks, achieving an accuracy of near 0% on Flowers102 and StanfordCars. Providing the label set in the prompt could improve the accuracy, but the performance is still not satisfactory.

In contrast, by tuning chips on the hidden states, we can achieve a decent accuracy while pruning the language model part of LLaVA. This phenomenon indicates that the information essential for image classification is already contained in the hidden states of multimodal models, but the models have difficulty in correctly decoding them. By adopting chip-tuning, we can extract related information before the final layer, and decode the information correctly.

We also notice that MLP chips perform extremely badly on StanfordCars, which may be caused by the large label set size of the dataset.

4.4 Combination with Finetuning

A critical difference between chip-tuning and the previous structured pruning method is that chip-tuning requires additional training data. With these training data, we can also finetune the backbone language model to achieve better performance. To better study the effectiveness of chip-tuning, we finetune models with the same data using LoRA (Hu et al., 2021) and observe the performance gap between chip-tuning and finetuning. We set rank $r = 16$ and LoRA alpha $\alpha = 32$[2].

Table 3 shows the comparison results. Finetuning the backbone model with LoRA could improve the performance on various benchmarks, and outperforms chip-tuning on the raw model as expected. Nevertheless, we can perform chip-tuning on finetuned models, which will only lead to marginal performance loss and will even improve the performance on certain datasets. These results clearly indicate that chip-tuning is compatible with traditional finetuning methods.

Considering that the target of probing is to read out relevant features from the internal representations of models, finetuning the model would help the backbone model develop better representations for the given classification task. Thus, chips could benefit from the optimized input features and achieve better performance.

5 Analysis

5.1 Number of Pruned Layers

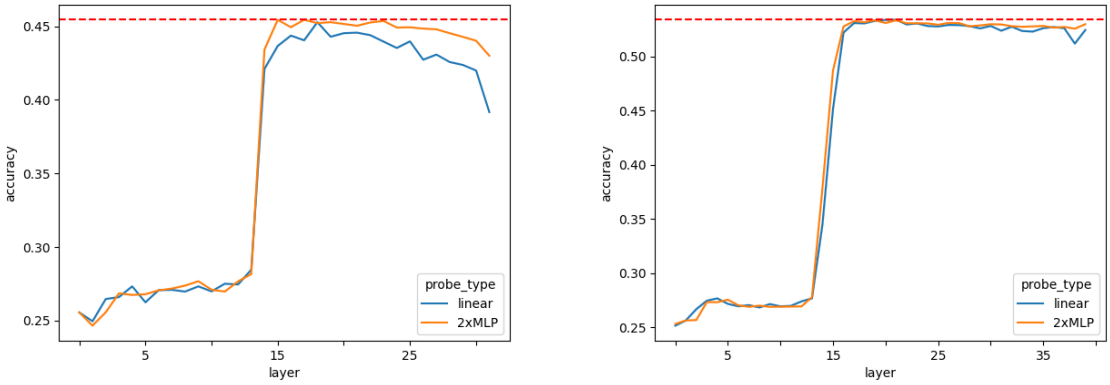
Choosing different chips would change the number of pruned layers, and thus affect the classification performance. We conduct experiments on the MMLU dataset with Llama-2 models, and Figure 2 demonstrates the correlation between number of pruned layers and classification accuracy.

It can be clearly observed that the classification accuracy exhibits a drastic change on both datasets, increasing from random guess to a decent level, and then fluctuating within a relatively small range. The change happens at around layer 18 for Llama-2-7B and happens at layer 20 for Llama-2-13B, which

[2]See Appendix D for detailed settings.

Model	Method	ΔParams	BoolQ	Race-H	Race-M	C3	MMLU	Avg. Score
Llama2-7B	Raw	-	71.62	35.71	34.19	43.56	45.39	46.09
	LoRA	8M	87.37	**81.59**	**86.56**	83.83	**54.80**	78.83
	CT (Raw)	0.5M	79.05	47.91	53.69	48.93	44.89	54.89
	CT (LoRA)	0.5M	**89.20**	81.45	86.42	**84.28**	54.57	**79.18**
Llama2-13B	Raw	-	82.39	57.95	60.38	47.51	55.00	60.65
	LoRA	12.5M	89.42	85.05	88.23	**88.10**	**57.68**	**81.70**
	CT (Raw)	0.625M	78.23	62.04	67.06	68.21	52.79	65.67
	CT (LoRA)	0.625M	**90.09**	**85.22**	**88.58**	87.81	55.51	81.44

Table 3: Comparison between chip-tuning and finetuning with LoRA on the same training dataset. We attach a linear chip to the 20th layer of the 7B model and the 25th layer of the 13B model for classification. CT (Raw) and CT (LoRA) refer to adding linear chips to the raw model and the finetuned model, respectively.



(a) Llama-2-7B on MMLU.

(b) Llama-2-13B on MMLU.

Figure 2: The impact of pruning Llama2 models on MMLU by selecting chips on different layers.

are at the position of about 50% in the entire model. It is also worth noticing that the best performance is not necessarily achieved on the last layer, especially for the Llama-2-7B model, which may be a hint that features in middle-layer representations serve better for classification.

Stolfo et al. (2023) proposes the theory that early layers in language models focus on *gathering* and *transmitting* information in the input text, while mid-late layers are involved in *processing* the information and output the final answer. The theory matches our findings: information relevant to the final answer is transmitted to the last token on intermediate layers, and the information is sufficient for solving the question.

We also find that the performance gap between linear chips and 2-layer MLP chips is not extremely significant. On most layers, the two chips behave identically, especially for the 13B model. The observable difference is that the performance of MLP chips is slightly more stable, changing in a smaller range on late layers.

Multimodal models exhibit a different pattern. As illustrated in Figure 3, chips on multimodal models could achieve high accuracy from early layers, and chips on late layers generally perform better than those on intermediate layers. The critical information for image classification is already contained in the image tokens from the first layer, which could lead to the difference.

5.2 Chip Selection Strategy

Aside from choosing chips on a fixed layer, there exist other strategies to achieve better performance. We adopt three distinct strategies and evaluate them on Llama-2-7B:

Fixed selects a fixed layer l for all tasks ($l = 20$ for 7B models and $l = 25$ for 13B models).

Validate constructs a small validation set consisting of 200 examples, and chooses the chip which performs best on the validation set.

Model	Strategy	BoolQ	Race-H	Race-M	C3	MMLU	Avg. Score
Llama2-7B	Dense	71.62	35.71	34.19	43.56	45.39	46.09
	Fixed	79.05	47.91	53.69	48.93	44.89	54.89
	Validate	**79.48**	49.43	54.53	53.93	44.53	56.38
	Optimal	**79.48**	**50.34**	**54.74**	**54.35**	**45.49**	**56.88**

Table 4: Comparison between different chip selection strategies on Llama-2-7B.

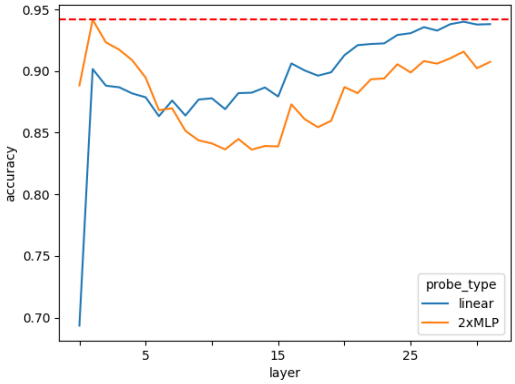


Figure 3: The impact of pruning LLaVA1.5-7B on Flowers102 by selecting chips on different layers. Different from the trends on NLP benchmarks, the trend does not exhibit a drastic change on certain layers.

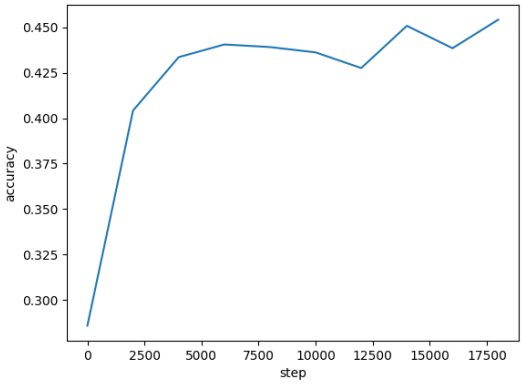


Figure 4: Analysis on the training dataset scale. We evaluate the performance of chip-tuning Llama-2-7B on MMLU every $2,000$ training steps. The overall accuracy rapidly increases until $6,000$ training steps, and continues to increase slightly afterward.

Optimal evaluates the performance of all chips, and selects the chip with the highest accuracy. This strategy reflects the upper bound of chip-tuning.

The experimental results are shown in Table 4. Choosing chips according to the validation set generally achieves better performance than pruning the model on a fixed layer, but the pruning ratio may vary across different datasets. While the **Optimal** strategy outperforms other strategies, the performance gap is not large, The **Validate** strategy could achieve comparable results with **Optimal** accuracy, proving the robustness of chip-tuning.

5.3 Impact of Training Dataset Scale

Training data is a crucial component in model training. Considering the scenario where training data is scarce, we test the performance of chip-tuning under different scales of the training dataset.

Figure 4 shows the classification accuracy under different training dataset scales. The accuracy rapidly increases before 6,000 training examples and reaches a plateau afterward. Although the accuracy may drop at a certain time step, the figure still displays a pattern of slow increase after 6,000 training examples. We draw the conclusion that a sufficient number of training data is essential for chips to converge, but further data could still bring subtle improvements.

6 Conclusion

In this paper, we propose chip-tuning, a structured pruning framework specialized for classification tasks. Chip-tuning adopts probing classifiers to extract relevant features from intermediate layers of language models, and safely removes subsequent layers without affecting the selected classifier. Experimental results on a variety of models and datasets demonstrate that chip-tuning surpasses previous baseline models on both performance and pruning ratio. Chip-tuning performs well by selecting chips on a fixed layer, and could further achieve a pruning ratio of up to 50% by selecting the optimal chip.

Meanwhile, we find that chip-tuning is also compatible with multimodal models and finetuned models. Considering the simplicity of layer removal, chip-tuning shows its potential in deploying LLMs under practical scenarios. We hope our work could inspire further research on efficient model pruning.

Limitations

Based on the technique of probing, chip-tuning requires the backbone models to contain relevant features in their internal representations. On tasks that the backbone models perform poorly, chip-tuning would not yield satisfactory results either.

Meanwhile, chip-tuning is designed mainly for classification tasks, which is the reason why we don't evaluate chip-tuning on datasets like HellaSwag that use perplexity-based evaluation methods. Directly applying chip-tuning to generation tasks may lead to unexpected results, and generation-oriented chips remain to be explored in the future.

References

AI@Meta. 2024. Llama 3 model card.

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. In *The Twelfth International Conference on Learning Representations*.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2022. Discovering latent knowledge in language models without supervision. In *The Eleventh International Conference on Learning Representations*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Li Fei-Fei, Rob Fergus, and Pietro Perona. 2004. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 conference on computer vision and pattern recognition workshop*, pages 178–178. IEEE.

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

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*.

Wes Gurnee and Max Tegmark. 2023. Language models represent space and time. *arXiv preprint arXiv:2310.02207*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. 2024. Distillm: Towards streamlined distillation for large language models. In *Forty-first International Conference on Machine Learning*.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 2013. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794.

Belinda Z Li, Maxwell Nye, and Jacob Andreas. 2021. Implicit representations of meaning in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1813–1827.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2023. Improved baselines with visual instruction tuning.

Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao. 2021. Post-training quantization for vision transformer. *Advances in Neural Information Processing Systems*, 34:28092–28103.

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.

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. *arXiv preprint arXiv:2403.03853*.

Neel Nanda, Andrew Lee, and Martin Wattenberg. 2023. Emergent linear representations in world models of self-supervised sequence models. In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 16–30.

Maria-Elena Nilsback and Andrew Zisserman. 2008. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pages 722–729. IEEE.

Roma Patel and Ellie Pavlick. 2022. Mapping language models to grounded conceptual spaces. In *International conference on learning representations*.

Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. 2023. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7035–7052.

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2020. Investigating prior knowledge for challenging chinese machine reading comprehension. *Transactions of the Association for Computational Linguistics*, 8:141–155.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2023. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*.

Mingxu Tao, Quzhe Huang, Kun Xu, Liwei Chen, Yansong Feng, and Dongyan Zhao. 2024. Probing multimodal large language models for global and local semantic representations. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 13050–13056.

Adly Templeton. 2024. *Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet*. Anthropic.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, et al. 2023. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*.

Yifei Yang, Zouying Cao, and Hai Zhao. 2024. Laco: Large language model pruning via layer collapse. *arXiv preprint arXiv:2402.11187*.

Yingtao Zhang, Haoli Bai, Haokun Lin, Jialin Zhao, Lu Hou, and Carlo Vittorio Cannistraci. 2024a. Plug-and-play: An efficient post-training pruning method for large language models. In *The Twelfth International Conference on Learning Representations*.

Yuhui Zhang, Alyssa Unell, Xiaohan Wang, Dhruba Ghosh, Yuchang Su, Ludwig Schmidt, and Serena Yeung-Levy. 2024b. Why are visually-grounded language models bad at image classification? *arXiv preprint arXiv:2405.18415*.

Longguang Zhong, Fanqi Wan, Ruijun Chen, Xiaojun Quan, and Liangzhi Li. 2024. Blockpruner: Fine-grained pruning for large language models. *arXiv preprint arXiv:2406.10594*.

Fangwei Zhu, Damai Dai, and Zhifang Sui. 2024. Language models understand numbers, at least partially. *arXiv preprint arXiv:2401.03735*.

A Datasets

The properties of datasets we used are shown in Table 7. We use these datasets according to their license and intended use.

B Details for Main Experiments

Figure 5 shows how the performance changes with different number of layers pruned. We can see that the optimal chip varies as the dataset changes. However, pruning around layer 18 of the 7B model (about 40%) and layer 20 of the 13B model (about 50%) is generally acceptable.

We also notice that probing late layers of Llama-2-7B leads to worse results, which leaves the question of whether the 7B model "forgets" certain information on late layers. The question remains to be explored in the future.

We record the layer on which chips show the best performance or highest pruning ratio in Table 6. Notice that we define layer with the highest pruning ratio as the first layer after the drastic change in accuracy, which could be subjective.

We implement our code with the huggingface Transformers and Peft Python library. Conducting chip-tuning on a 7B model or a 16-bit 13B model with 20,000 examples would take about 2 hours on a single NVIDIA A100 40GB GPU.

C Details for Multimodal Experiments

Figure 6 shows how the performance changes by pruning LLaVA1.5-7B. Different from text datasets, the optimal chip for image classification typically appears on late layers, while chips on early layers also exhibit decent accuracy. Surprisingly, 2-layer MLP chips fail to predict the class of images on StanfordCars. This may be a result of the larger class label set size (196) compared with Flowers102 (102) and Caltech101 (101).

D Details for LoRA Experiments

Table 5 shows the experimental settings for LoRA experiments.

E Experiments on Llama3

We evaluate chip-tuning on Llama3-8B-Instruct (AI@Meta, 2024), one of the up-to-date LLMs. We prune the model to layer 22 in experiments.

The experimental results in Table 8 are similar to those in Table 3: applying chip-tuning on Llama3 has minimal impact on classification accuracy, proving that chip-tuning is compatible with Llama3. The optimal performance of chips even outperforms the finetuned LoRA models.

F Finetuning Pruned Baseline Models

For fair comparisions, we finetune the pruned baseline models on the training set of each benchmark to see how they perform with the same data provided. We use Llama2-7B as the backbone model, and finetune LLMPruner (Ma et al., 2023) and SliceGPT (Ashkboos et al., 2024) under their default LoRA settings. We do not train LaCo and ShortGPT as we cannot find their official code.

Both baseline models are trained with at most 20,000 training data same to these chip-tuning used on each benchmark. The accuracy of finetuned baseline models are obtained by selecting the choice token (for example, "A", "B", "C", "D" for 4-choice problems) with the highest generation probability, as free-form generation would yield unexpected results.

Table 9 shows the result of finetuning LLMPruner and SliceGPT with the same data used by chip-tuning. While the finetuned versions achieve higher accuracy than the original version, we can clearly see that chip-tuning greatly outperforms both baselines, further proving the effectiveness of chip-tuning.

Parameter	Value
learning rate	1×10^{-5}
weight decay	0.01
r	16
α	32
batch size	1
epoch	1

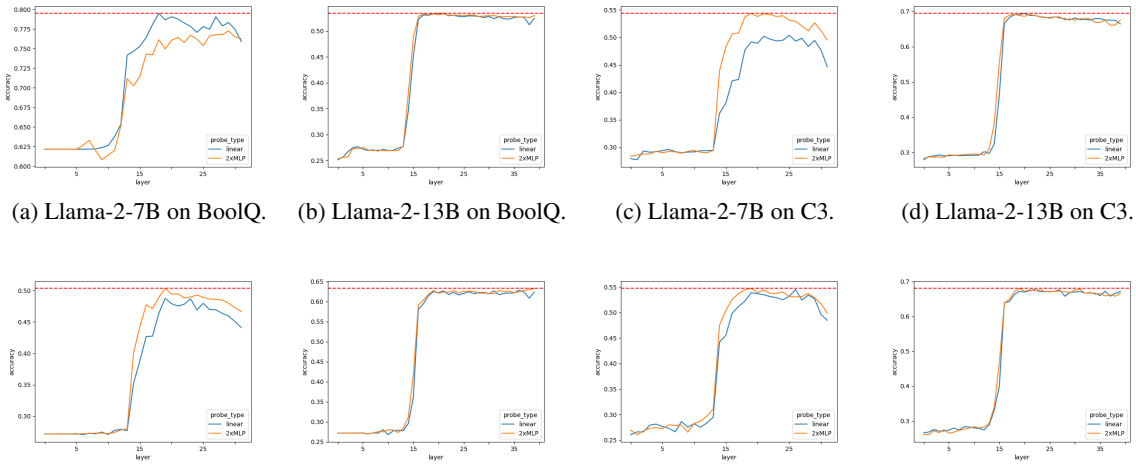
Table 5: Parameters for LoRA training.

Model	BoolQ	Race-H	Race-M	C3	MMLU
Llama2-7B	18/17	19/19	19/17	19/18	17/15
Llama2-13B	38/18	39/18	19/16	20/17	21/16
Baichuan2-7B	21/18	30/19	30/19	20/19	24/19
Baichuan2-13B	38/18	36/22	35/22	27/22	22/21

Table 6: The corresponding layer of chips with the best performance or highest pruning ratio on each dataset. The format of each cell in the table is (layer with best performance / layer with highest pruning ratio).

Dataset	Link	Train Split	Eval Split
BoolQ	https://huggingface.co/datasets/google/boolq	train	validation
Race	https://huggingface.co/datasets/ehovy/race	train	test
C3	https://huggingface.co/datasets/dataset-org/c3	train	validation
MMLU	https://huggingface.co/datasets/cais/mmlu	auxiliary_train	test
Flowers102	https://huggingface.co/datasets/dpdl-benchmark/oxford_flowers102	train+validation	test
StanfordCars	https://huggingface.co/datasets/tanganke/stanford_cars	train	test
Caltech101	https://huggingface.co/datasets/dpdl-benchmark/caltech101	train	test

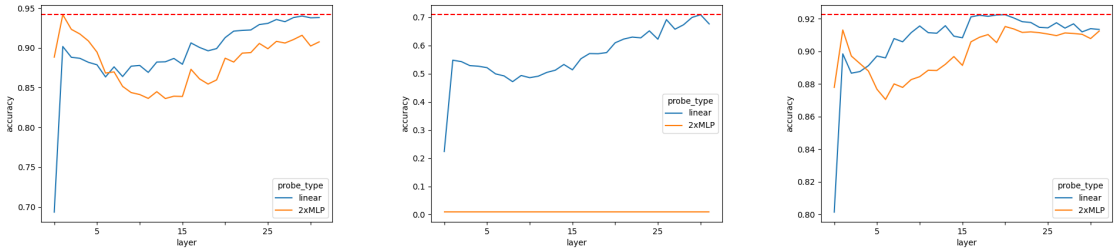
Table 7: Dataset details.



(a) Llama-2-7B on BoolQ. (b) Llama-2-13B on BoolQ. (c) Llama-2-7B on C3. (d) Llama-2-13B on C3.

(e) Llama-2-7B on Race-H. (f) Llama-2-13B on Race-H. (g) Llama-2-7B on Race-M. (h) Llama-2-13B on Race-M.

Figure 5: The impact of pruning Llama2 models on BoolQ, C3, Race-H, and Race-M by selecting chips on different layers.



(a) LLaVA1.5-7B on Flowers102. (b) LLaVA1.5-7B on StanfordCars. (c) LLaVA1.5-7B on Caltech101.

Figure 6: The impact of pruning LLaVA1.5-7B on Flowers102, StanfordCars, and Caltech101 by selecting chips on different layers.

Model	Method	ΔParams	BoolQ	Race-H	Race-M	C3	MMLU	Avg. Score
Llama3-8B	Raw	-	57.77	80.87	85.24	86.82	64.01	74.94
	LoRA	8M	87.16	80.25	90.46	84.93	66.20	81.8
	CT (Raw)	0.5M	76.73	81.39	86.00	88.86	64.26	79.45
	CT (LoRA)	0.5M	87.03	81.36	90.95	83.75	66.31	81.88
	CT (Max)	0.5M	**90.83**	**88.02**	**91.07**	**93.71**	**66.37**	**86.00**

Table 8: Comparison between chip-tuning and finetuning with LoRA on Llama3-8B. CT (Raw) and CT (LoRA) refer to adding linear chips to the raw model and the finetuned model on layer 22, respectively. CT (Max) refers to the best performance of chips on the finetuned model.

Model	Method	BoolQ	Race-H	Race-M	C3	MMLU	Avg. Score
Llama2-7B	Raw	71.62	35.71	34.19	43.56	45.39	46.09
	LLMPrun.	55.20	22.56	22.35	25.64	23.33	29.82
	SliceGPT	38.32	21.07	21.66	39.78	28.92	29.95
	CT (Raw)	79.05	47.91	53.69	48.93	44.89	54.89
	LoRA	87.37	**81.59**	**86.56**	83.83	**54.80**	78.83
	LLMPrun.+LoRA	60.49	23.67	25.56	17.84	26.60	30.83
	SliceGPT+LoRA	65.72	71.76	76.81	60.27	43.06	63.52
	CT (LoRA)	**89.20**	81.45	86.42	**84.28**	54.57	**79.18**

Table 9: Comparison between chip-tuning and finetuning pruned baseline models. We only report the result of finetuning LLMPruner and SliceGPT, as LaCo and ShortGPT do not provide their official code.