KDP: Simplifying Representation Dynamics in Kernel Space

Zeyu Ma[1,2], Wanying Wang[2], Guchu Zou[3],
Mingang Chen[2], Jianhong Wu[1*]

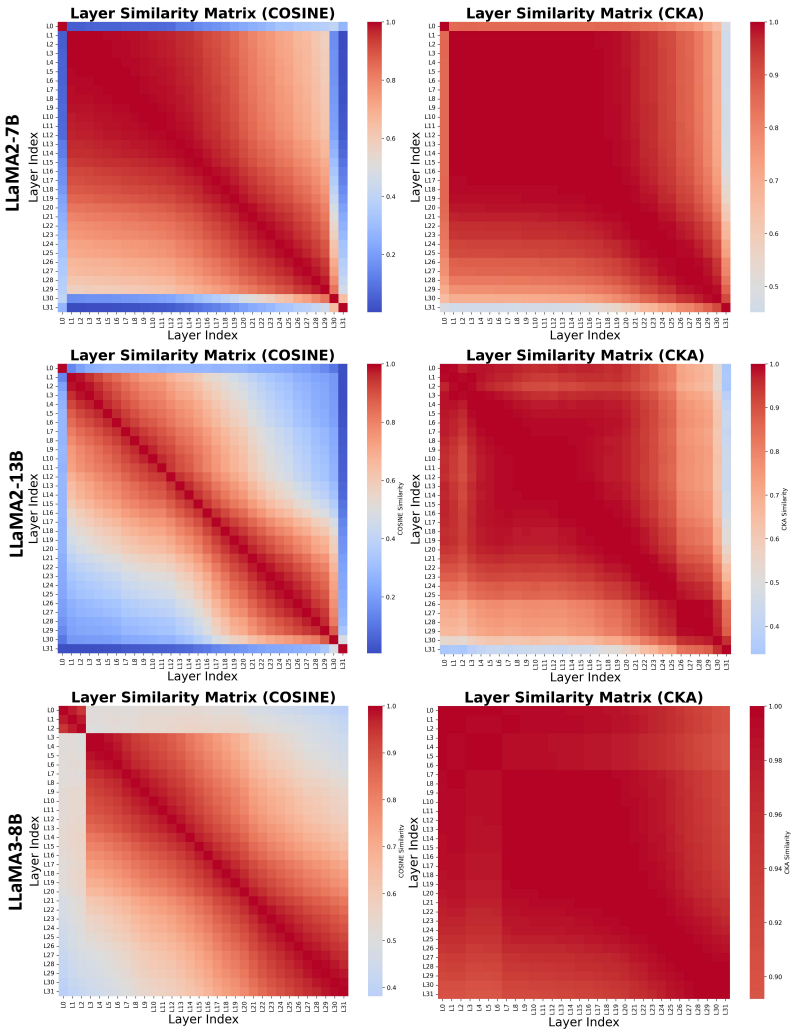
[1]Shanghai Normal University,
[2]Shanghai Key Laboratory of Computer Software Testing and Evaluating,
[2]Shanghai Institute of Ceramics, Chinese Academy of Sciences

April 15, 2026

Background

- LLMs have demonstrated exceptional performance across various tasks and domains.
- However, as models increase in scale, the substantial computational and hardware deployment costs pose significant challenges.
- **Pruning** is a primary technique for model compression. (Other examples like Quantization, Distilling et al.)
- **Layer pruning** is garnering increasing attention because it naturally leads to inference acceleration and model size reduction without requiring special handling.

Representation Similarity



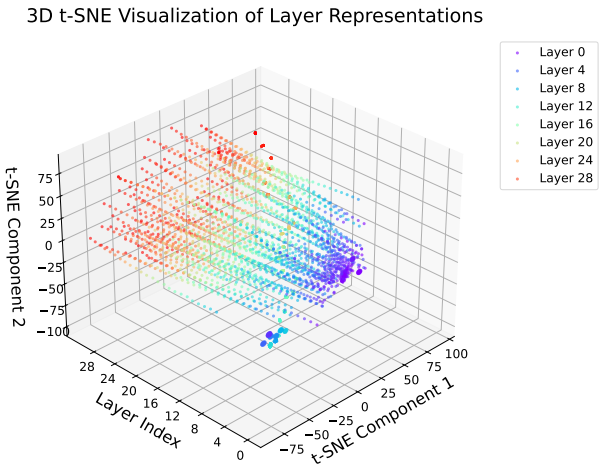
- Multiple consecutive layers exhibit consistently high similarity.
- Similarity measured in the kernel space is higher than in the original space.
- *Does representational similarity equate to computational redundancy?*

Slow Manifold Hypothesis

LLM Forward Formulation

$$\mathbf{h}_{l+1}(\mathbf{x}) = \mathbf{h}_l(\mathbf{x}) + \mathrm{f}_l(\mathrm{Norm}(\mathbf{h}_l(\mathbf{x}))). \tag{1}$$

Viewing the forward pass of an LLM as a discrete-time dynamical system, high representational similarity signifies the system's entry into a "slow manifold", characterized by a small velocity, i.e. $\|\mathrm{f}_l(\mathrm{Norm}(\mathbf{h}_l))\| \ll \|\mathbf{h}_l\|$.



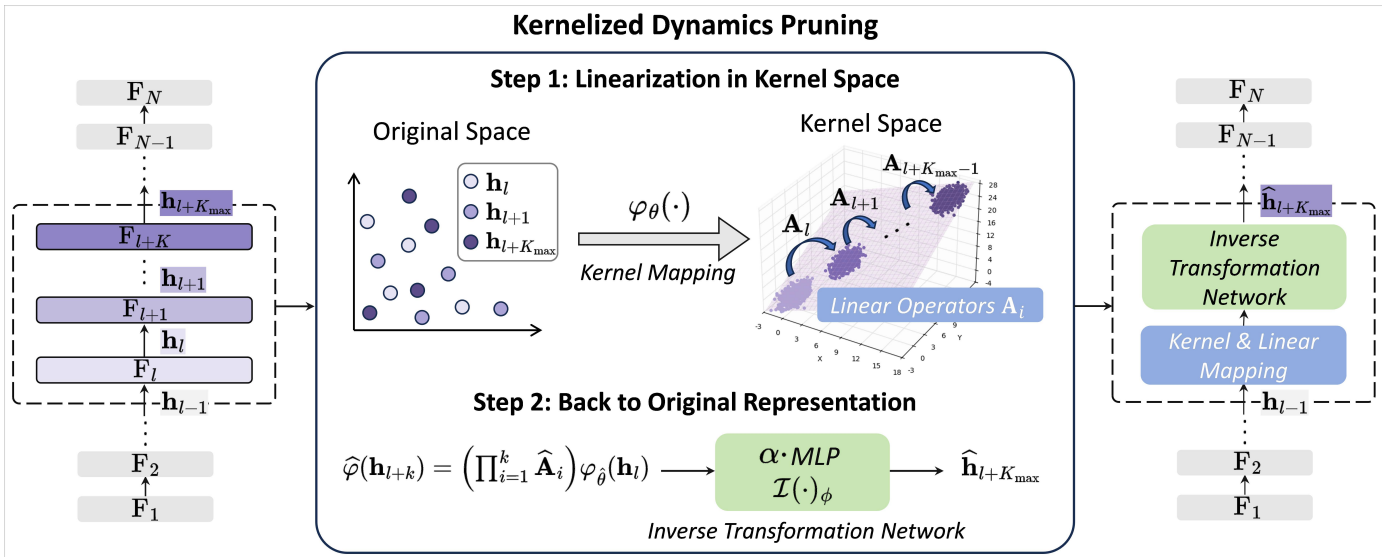
The system's short-term evolution can be described by a much simpler function (e.g., a linearized first-order approximation), rather than requiring the nonlinear dynamics of a complete Transformer block.

Kernelized Dynamics Pruning

However, substituting the non-linear forward pass with linear dynamics may result in the loss of fine-grained details, which could contain low-variance, task-relevant information.
We aim to identify a Hilbert space in which the complex dynamics become amenable to linear approximation, thereby enabling effective layer pruning via linear simplification in kernel space.
We propose **Kernelized Dynamics Pruning (KDP)**, simplifying representational dynamics within a kernel space to reduce computational redundancy.

Kernelized Dynamics Pruning

- First, we jointly optimize a learnable kernel transformation and the linear coefficients mapping between layers.
- Second, an inverse transformation network maps these representations from the kernel space back to the original space, reconstructing the original layer to enable model pruning.



Main Contributions

- We reformulate layer pruning as the search for an optimal geometric embedding within a Reproducing Kernel Hilbert Space and propose Kernelized Dynamics Pruning (KDP) method, which linearizes representational dynamics using the kernel trick to simplify consecutive layers.
- We provide a theoretical error bound for linearization in the kernel space and demonstrate the modeling and linear simplification advantages of the kernel space, with experimental results corroborating the theoretical analysis.
- We conduct extensive experiments on 15 benchmarks, demonstrating that our method maintains superior performance while requiring only a small number of trainable parameters and limited calibration data, eliminating the need for additional post-training.

Learnable Kernel

Learnable Kernel

The kernel is approximated by the inner product feature map $\varphi(\cdot) : \mathbb{R}^d \to \mathbb{R}^{2m}$, s.t. $\mathrm{k}(\mathbf{x}, \mathbf{y}) \approx \varphi(\mathbf{x})^\top \varphi(\mathbf{y})$, where $\varphi(\mathbf{x})$ is defined as:

$$\varphi(\mathbf{x}) = \frac{1}{\sqrt{m}} \left(\cos\left(\mathbf{W}^\top \mathbf{x} + \mathbf{b}\right)^\top, \sin\left(\mathbf{W}^\top \mathbf{x} + \mathbf{b}\right)^\top \right)^\top, \tag{2}$$

where $\mathbf{W} \in \mathbb{R}^{d \times m} \overset{\text{iid}}{\sim} \mathcal{N}(0, \mathbf{\Sigma})$, and $\mathbf{b} \in \mathbb{R}^m \overset{\text{iid}}{\sim} \mathrm{U}(0, 2\pi)$.

In contrast to the original RFF kernel, here the covariance matrix $\mathbf{\Sigma}$ is learnable and parameterized as $\mathbf{\Sigma} = \mathbf{D} + \mathbf{L}\mathbf{L}^\top$, $\mathbf{D} = \mathrm{diag}(\exp(\boldsymbol{\lambda}))$ is parameterized by a learnable vector $\boldsymbol{\lambda} \in \mathbb{R}^d$ and $\mathbf{L} \in \mathbb{R}^{d \times r}$ is a learnable low-rank factor matrix, where the rank $r \ll d$.
Equation 2 effectively learns an anisotropic Guassian RBF kernel $\mathrm{k}(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{y})^\top \mathbf{\Sigma}^{-1}(\mathbf{x} - \mathbf{y})\right)$ whose distance metric is adapted to the data.

Learnable Kernel

- Note that in Figure 1, the CKA similarity generally exceeds cosine similarity, which is consistent with the property that the high-dimensional representation induced by the kernel space captures complex representational relationships more effectively.
- Inspired by calculus and PDEs, we assume that there exists a kernel function $\varphi(\cdot)$ that induces a Hilbert space where the layer-wise transformation becomes approximately linear, i.e., $\varphi(\mathbf{h}_{l+1}) \approx \mathbf{A}_l \varphi(\mathbf{h}_l)$.

Theoretical results

Theorem (Kernel Linearization Error Bound)

Let $\varphi_\theta : \mathbb{R}^d \to \mathbb{R}^{2m}$ *be a feature map such that* $\|\varphi_\theta(\mathbf{h})\|_2 \le R_\varphi$ *for all representations* $\mathbf{h}$. *Let* $\mathbf{A}_i$ *be linear operators for the* i*-th step transition with* $\|\mathbf{A}_i\|_{\mathrm{op}} \le B_{\mathbf{A}}$. *Let* $(\widehat{\theta}, \{\widehat{\mathbf{A}}_i\})$ *be the Empirical Risk Minimization(ERM) solution on an* n*-sample training set, and let* L_{ERM} *denote the minimum empirical one-step squared error. Then for any* $\delta \in (0,1)$, *with probability at least* $1-\delta$ *over the draw of the training set, for every starting layer* l *and horizon* $k \ge 1$, *the* k*-step error* $\mathcal{E}_{k,l}(\widehat{\theta}, \{\widehat{\mathbf{A}}_i\}) := \varphi_{\widehat{\theta}}(\mathbf{h}_{l+k}) - \left(\prod_{i=l}^{l+k-1} \widehat{\mathbf{A}}_i\right) \varphi_{\widehat{\theta}}(\mathbf{h}_l)$ *satisfies*

$$\left\|\mathcal{E}_{k,l}(\widehat{\theta}, \{\widehat{\mathbf{A}}_i\})\right\| \le \underbrace{\sqrt{L_{\mathrm{ERM}} + C\, B_{\mathbf{A}}^2 R_\varphi^2 \sqrt{\frac{2m\,\log(2m/\delta)}{n}}}}_{(a)} \cdot \underbrace{\sum_{j=0}^{k-1} B_{\mathbf{A}}^{k-1-j}}_{(b)}. \tag{3}$$

Theoretical results

Theorem 1 yields a two-factor bound on k-step error: a single-step term (a) that captures the linear model's in-kernel fitting ability plus its generalization gap, and a multi-step accumulation factor (b) that governs how errors accumulate across replaced layers.
Theoretically, for the bounded parameters R_φ and $B_{\mathbf{A}}$, the error bound converges at a rate of $O(1/\sqrt{n})$ with the sample size n.
Therefore, Theorem 1 demonstrates that consecutive layers can be well simplified as linear transformations in kernel space.

Theoretical results

Lemma (Kernel Existence)

For any given $\epsilon > 0$, there exists a Random Fourier Feature map $\varphi : \mathbb{R}^d \to \mathbb{R}^{2m}$ (defined by suitable parameters $\mathbf{W}, \mathbf{b}$, and a sufficiently large dimension $2m$) and a corresponding linear operator $\mathbf{A}$, such that: $\mathbb{E}_{\mathbf{h}_l \sim \mathcal{D}} \left[\| \varphi(\mathbf{h}_{l+1}) - \mathbf{A}\varphi(\mathbf{h}_l) \| \right] < \epsilon$.

The proof of Lemma 2 is a direct application of the Universal Kernels.

Theoretical results

Consider the following two function classes at the l-th layer of an LLM: the identity class of the original space $\mathcal{F}_{\mathrm{ID}} := \{x \mapsto \mathbf{A}x : \|\mathbf{A}\|_{\mathrm{op}} \leq B_{\mathbf{A}}\}$, with a population risk of $\mathcal{R}_{\mathrm{ID}}(\mathbf{A}) := \mathbb{E}\left[\|\mathbf{h}_{l+1} - \mathbf{A}\mathbf{h}_l\|^2\right]$; the RFF class $\mathcal{F}_{\mathrm{RFF}} := \{x \mapsto \mathbf{A}\varphi_\theta(x) : \|\mathbf{A}\|_{\mathrm{op}} \leq B_{\mathbf{A}}, \|\varphi(x)\| \leq R_\varphi, \theta \in \Theta\}$, with a population risk of $\mathcal{R}_{\mathrm{RFF}}(\theta, \mathbf{A}) := \mathbb{E}\left[\|\varphi_\theta(\mathbf{h}_{l+1}) - \mathbf{A}\varphi_\theta(\mathbf{h}_l)\|^2\right]$.
Note the minimum population approximation error of the two classes as:

$$\mathrm{apx}_{\mathrm{ID}} := \inf_{\|\mathbf{A}\|_{\mathrm{op}} \leq B_{\mathbf{A}}} \mathcal{R}_{\mathrm{ID}}(\mathbf{A}), \qquad \mathrm{apx}_{\mathrm{RFF}} := \inf_{\theta \in \Theta, \|\mathbf{A}\|_{\mathrm{op}} \leq B_{\mathbf{A}}, \|\varphi(x)\| \leq R_\varphi} \mathcal{R}_{\mathrm{RFF}}(\theta, \mathbf{A})$$

Let $\widehat{\mathbf{A}}_{\mathrm{ID}}$ and $(\widehat{\theta}, \widehat{\mathbf{A}}_{\mathrm{RFF}})$ be the ERM solutions on the respective classes. We can get the following conclusion on the superior fitting capacity of Kernel space.

Theoretical results

Theorem (Kernel Advantage)

Let $\Delta := \mathrm{apx}_{ID} - \epsilon > 0$. *For any* δ, *there exist* $m', c, C > 0$, *such that for RFF dimension* $2m > m'$ *and*

$$n \geq \frac{16\left(CB_{\mathbf{A}}^2\left(R_\varphi^2\sqrt{2m\log\frac{2cm}{\delta}} - R_h^2\sqrt{d\log\frac{cd}{\delta}}\right)_+\right)^2}{\Delta^2}, \tag{4}$$

the following holds with probability at least $1-\delta$*:*
$\mathcal{R}_{RFF}(\widehat{\theta}, \widehat{\mathbf{A}}_{RFF}) \leq \mathcal{R}_{ID}(\widehat{\mathbf{A}}_{ID})$.

Theorem 3 establishes that in the kernel space, a well-trained RFF with dimensionality greater than $2m$ achieves lower population risk compared to simplification in the original representation space. This result further substantiates the necessity of performing linearized model compression in the kernel space.

Overall Framework

Algorithm 1: Kernelized Dynamics Pruning

Require: Pretrained F, calib. set $\mathcal{D}$, RFF dim m, low-rank r, block max length $K_{\max}$, budget B, stability step γ, weight W, training iters T.

Let $\mathcal{G}$ be the set of B non-overlapping model blocks $\mathrm{F}_{l:l+K_{\max}}$ that maximize the similarity score $\mathrm{CKA}(\mathbf{h}_l, \mathbf{h}_{l+K_{\max}})$.

procedure JOINTKERNELLINEARIZATION
  for $(l, \cdots, l+K_{\max}) \in \mathcal{G}$ **do**
    Define φ_θ, operators $\{\mathbf{A}_i\}_{i=1}^k$
    for minibatch $\mathcal{B} \subset \mathcal{D}$ **do**
      $\widehat{\varphi}(\mathbf{h}_{l+k}(\mathcal{B})) \leftarrow \left(\prod_{i=1}^k \mathbf{A}_i\right) \varphi_\theta\left(\mathbf{h}_l(\mathcal{B})\right)$
      $\mathcal{L}(\cdot) \leftarrow \mathcal{L}_{\text{mse}} + \mathcal{L}_{\cos}$
      Update θ and $\{\mathbf{A}_i\}_{i=1}^k$
  return $\{\theta^{(l)}, \{\mathbf{A}_i^{(l)}\}_{i=1}^k\}_{l\in\mathcal{G}}$

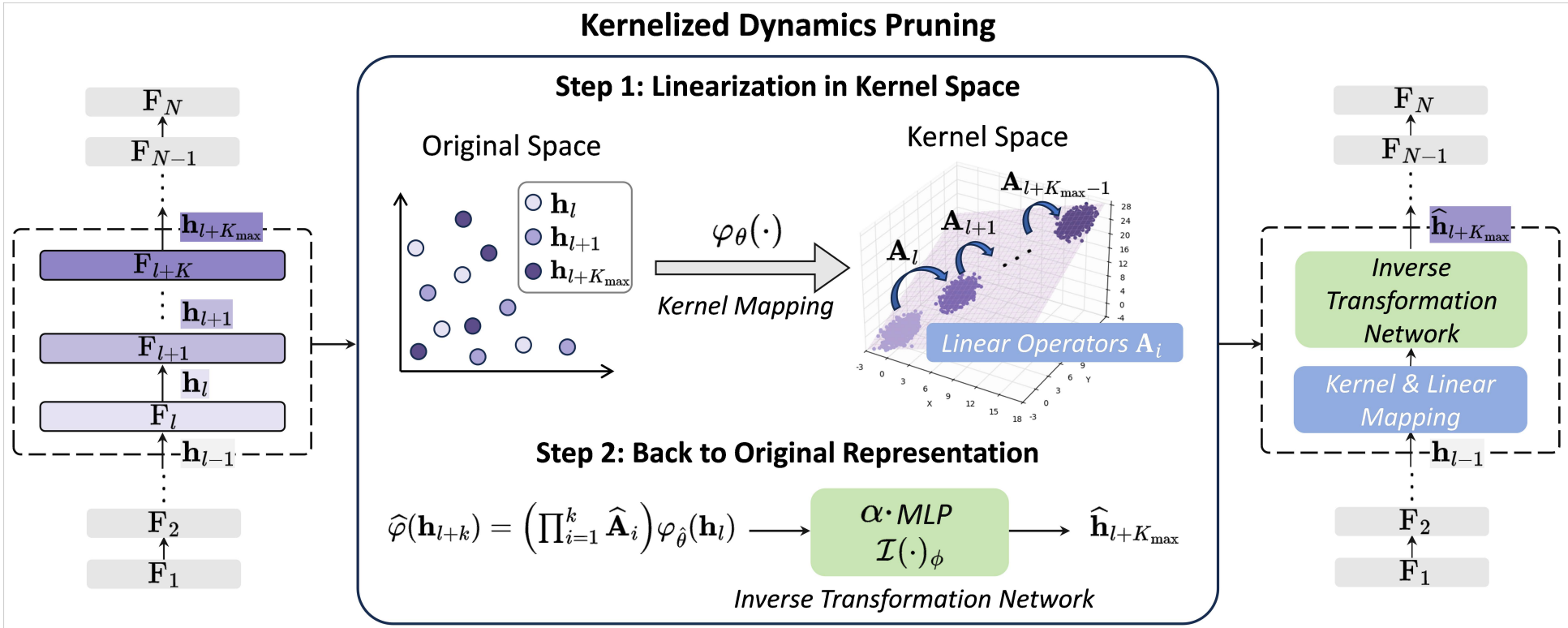
procedure TRAININVERSENETWORK
  for $(l, \cdots, l+K_{\max}) \in \mathcal{G}$ **do**
    Define inverse network $\mathcal{R}_\phi$
    for minibatch $\mathcal{B} \subset \mathcal{D}$ **do**
      $\widehat{\mathbf{h}}_{\mathbf{l+k}} \leftarrow \mathcal{R}\left(\widehat{\varphi}(\mathbf{h}_{l+k})\right)$
      $\mathcal{L}(\cdot) \leftarrow \mathcal{L}_{\text{mse}} \mid \mathcal{L}_{\text{norm}}$
      Update ϕ
  return $\{\phi^{(l)}\}_{l\in\mathcal{G}}$

procedure FOLDBLOCK
  Define $\mathcal{S}(\mathbf{h}) \leftarrow \mathcal{I}_\phi\Big(\left(\prod_{i=1}^k \mathbf{A}_i\right) \varphi_\theta(\mathbf{h})\Big)$
  Replace $\mathrm{F}_l, \ldots, \mathrm{F}_{l+k}$ with $\mathcal{S}$

KDP include two steps:

- First, we jointly optimize a learnable kernel transformation and the linear coefficients mapping between layers.
- Second, an inverse transformation network maps these representations from the kernel space back to the original space, reconstructing the original layer to enable model pruning.

Overall Framework



Kernel Linearization Joint Training

For the candidate block for pruning, we learn a replacement module by jointly optimizing the learnable RFF kernel parameters θ and the multi-step linear operators $\{\mathbf{A}_i\}_{i=1}^{K_{\max}}$. For each training sample $x \in \mathcal{D}$, the estimate $(\widehat{\theta}, \{\widehat{\mathbf{A}}_i\})$ is given by:

$$\begin{aligned}\underset{\theta,\{\mathbf{A}_i\}}{\operatorname{argmin}} \sum_{i=1}^{K_{\max}} \sum_{x\in\mathcal{D}} \Big[\| \mathbf{A}_i \varphi_\theta \left(\mathbf{h}_{l+i-1}(x)\right) - \varphi_\theta \left(\mathbf{h}_{l+i}(x)\right) \|^2 \\ + \left(1 - W \odot \cos\left(\mathbf{A}_i \varphi_\theta \left(\mathbf{h}_{l+i-1}(x)\right), \varphi_\theta \left(\mathbf{h}_{l+i}(x)\right)\right)\right) \Big].\end{aligned} \tag{5}$$

The loss function is composed of two terms: a reconstruction loss and a weighted cosine similarity loss. The latter is designed to encourage a finer-grained alignment of the representations' geometric structure.

Inverse Transformation Network

Once the joint training converges, we can get k-step prediction in the kernel space, i.e., $\widehat{\varphi}(\mathbf{h}_{l+k}) = \left(\prod_{i=1}^{k} \widehat{\mathbf{A}}_i\right) \widehat{\varphi}_\theta(\mathbf{h}_l)$. However, this output must be mapped back to the original space for layer replacement. Thus, an inverse transformation network, $\mathcal{I}(\cdot)_\phi : \mathbb{R}^{2m} \to \mathbb{R}^d$, takes the kernel-space prediction $\widehat{\varphi}(\mathbf{h}_{l+k})$ as input, to reconstruct representation in original space $\mathbf{h}_{l+k}$, denote as $\widehat{\mathbf{h}}_{l+k}$.
The estimate $\widehat{\phi}$ is given by:

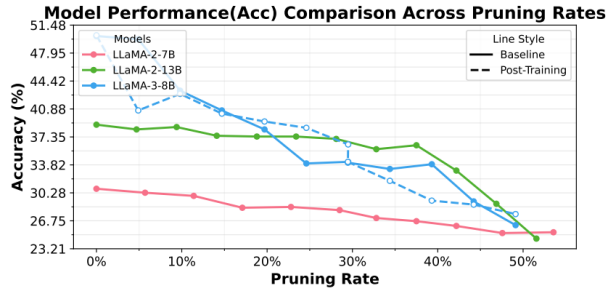
$$\underset{\phi}{\operatorname{argmin}} \sum_{x \in \mathcal{D}} \left\| \mathcal{I}_\phi \left(\widehat{\varphi} \left(\mathbf{h}_{l+k} \left(x \right) \right) \right) - \mathbf{h}_{l+k}(x) \right\|^2. \tag{6}$$

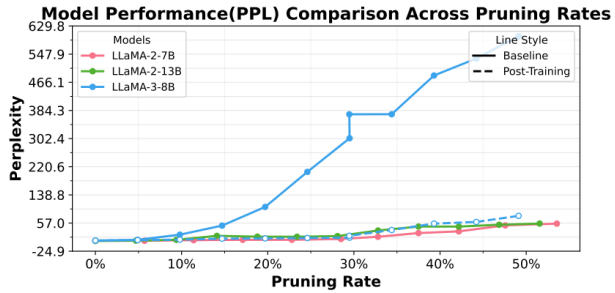
We parameterize the inverse transformation network $\mathcal{I}$ as a two-layer MLP with a scalar scaling factor α, i.e. $\mathcal{I}(\mathbf{x}) := \alpha \cdot \mathrm{MLP}(\mathbf{x})$.

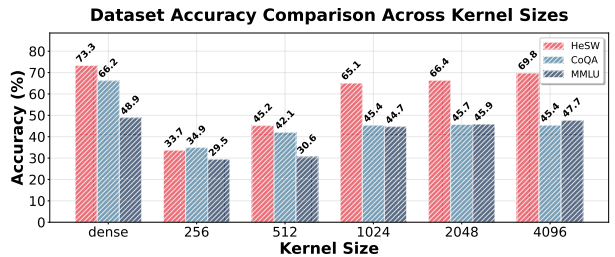
Main Results & Ablations

Table 1: Performance comparison of different baselines on classification benchmarks. "*" indicates that we refer to the results in the original paper. "†" indicates the results after fine-tuning. Average (Avg.) represents the arithmetic mean of Accuracy across all datasets. Retained performance (RP.) represents the percentage of the original model's performance retained by the pruning method.

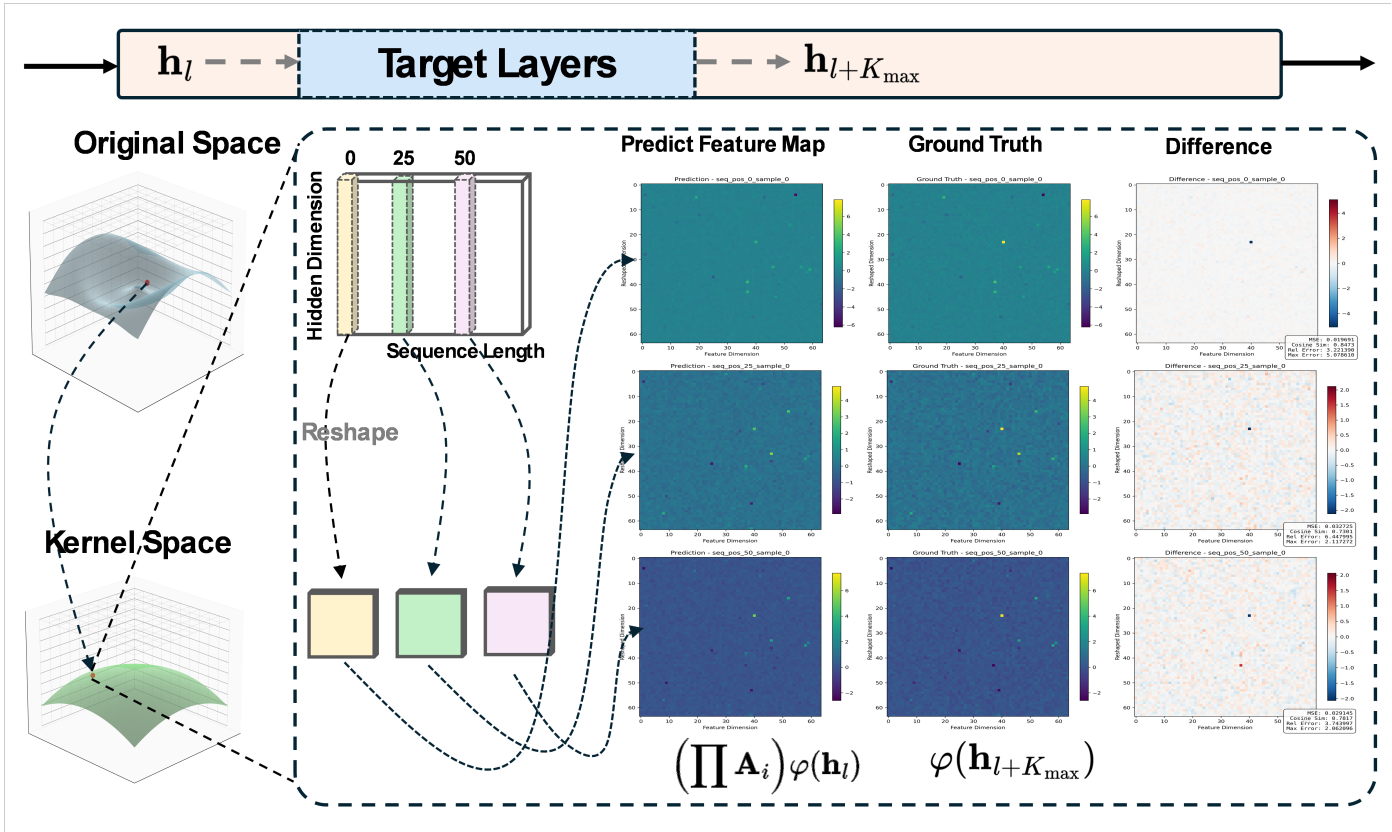
LLM	Method	Ratio	CMNLI	HeSw	PIQA	CHID	WSC	CoQA	BoolQ	MMLU	CMMLU	Race	SST2	C3	Avg.	RP.
LLaMA-2-7B	Dense	0.0%	34.9	73.3	79.7	41.6	81.2	66.2	72.3	48.9	30.8	42.8	93.2	44.8	59.14	100.0
	SLEB	20.1%	29.7	63.3	66.1	19.3	**75.7**	48.5	68.2	24.2	26.4	28.8	78.8	35.3	47.02	79.5
	ShortGPT	27.1%	31.3	51.6	66.3	25.5	70.9	**49.0**	59.6	34.4	28.5	37.7	68.4	35.9	46.59	78.8
	LaCo*	27.1%	34.4	55.7	69.8	**36.1**	40.4	45.7	64.1	26.5	25.2	23.6	-	39.7	41.93	70.9
	Streamline†	27.0%	32.9	54.3	63.6	26.9	70.7	42.5	64.3	37.7	**29.5**	34.9	90.5	32.6	48.37	81.8
	LLMPruner†	24.8%	31.4	55.4	**70.1**	27.8	71.4	44.7	53.3	25.7	25.1	25.4	67.7	23.1	43.43	73.4
	SliceGPT	25.4%	31.9	49.9	68.7	16.7	66.5	48.6	55.5	28.2	22.3	25.2	78.3	30.7	43.54	73.6
	SliceGPT†	25.4%	32.1	53.3	69.2	20.4	61.2	44.4	61.9	29.7	22.5	26.7	87.1	28.7	44.77	75.7
	w/o Kernel	24.8%	30.8	34.5	57.1	12.8	69.1	11.2	50.3	25.3	24.8	20.5	51.5	23.5	34.28	58.0
	Ours	22.8%	33.6	**65.1**	**70.1**	27.2	73.9	45.4	71.6	**44.7**	28.1	39.9	**94.0**	**43.7**	**53.11**	**89.9**
	Ours†	22.8%	**34.6**	63.0	68.8	29.5	74.8	**49.0**	**72.3**	41.5	26.7	**40.4**	92.3	37.3	52.52	88.8
LLaMA-2-13B	Dense	0.0%	48.1	74.4	77.5	47.9	88.5	65.6	73.9	57.7	38.9	59.7	93.7	47.5	64.45	100.0
	SLEB	19.5%	34.7	63.1	67.2	39.0	77.7	51.7	52.1	25.6	24.6	53.5	77.9	31.7	49.9	77.4
	ShortGPT	24.6%	31.7	61.5	72.5	38.4	70.0	51.0	62.8	54.4	33.7	57.3	76.1	45.0	54.53	84.6
	LaCo*	24.6%	32.9	64.4	74.3	40.1	52.9	52.7	64.0	45.9	32.6	55.6	-	44.9	50.94	79.0
	Streamline*	24.6%	33.0	**69.1**	**75.1**	38.0	36.5	**63.8**	66.2	**55.1**	**39.2**	58.0	-	45.7	52.70	81.8
	LLMPruner†	24.4%	27.8	55.3	61.9	30.7	63.4	49.7	53.0	23.1	23.3	20.0	65.8	30.3	42.03	65.2
	SliceGPT	23.6%	33.2	53.1	60.8	18.4	74.8	44.3	36.6	25.1	25.7	22.4	68.3	26.1	40.73	63.5
	SliceGPT†	23.6%	31.5	45.9	60.3	18.9	67.1	39.7	37.8	29.1	28.4	23.4	78.0	26.9	40.58	63.0
	w/o Kernel	24.3%	32.8	63.1	67.5	33.6	72.5	19.3	60.6	35.1	27.1	33.3	61.7	33.5	45.01	69.8
	Ours	23.4%	40.0	64.1	74.5	39.1	**82.5**	62.5	**69.4**	52.7	37.1	**58.2**	**89.1**	**49.5**	**59.89**	**92.9**
	Ours†	22.8%	**41.2**	63.5	73.9	**40.2**	80.1	**63.8**	63.3	51.9	38.0	55.6	87.8	48.5	58.98	91.5
LLaMA-3-8B	Dense	0.0%	33.9	75.7	77.9	72.8	86.4	72.5	75.3	67.2	50.1	73.5	93.0	62.1	70.03	100.0
	SLEB	19.0%	33.0	**65.1**	63.0	**28.0**	74.6	31.5	67.7	49.3	27.3	24.1	77.9	37.2	48.23	68.9
	ShortGPT	19.0%	33.7	46.0	65.4	25.8	73.0	**44.8**	38.9	34.2	**37.1**	30.4	**82.3**	**44.1**	46.31	66.1
	w/o Kernel	27.2%	33.0	41.0	55.7	18.0	73.8	29.3	50.8	24.0	25.7	21.7	65.1	36.8	39.58	56.5
	Ours	25.8%	**33.9**	61.5	73.5	23.9	83.6	37.7	**71.0**	**55.5**	34.0	**38.4**	81.4	**38.9**	**52.77**	**75.4**
	Ours†	22.8%	30.1	57.5	**74.1**	25.1	**84.4**	33.0	62.4	50.4	38.5	34.0	68.3	35.5	49.44	70.6





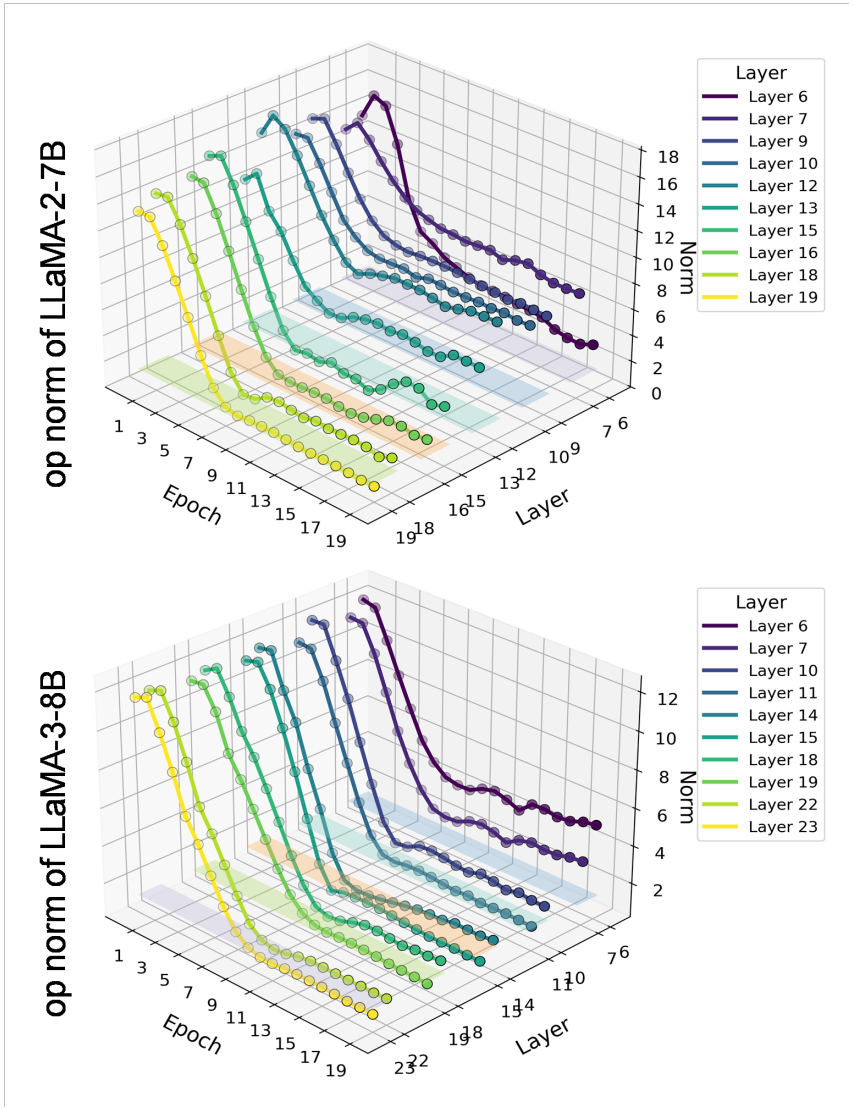


Further Analysis



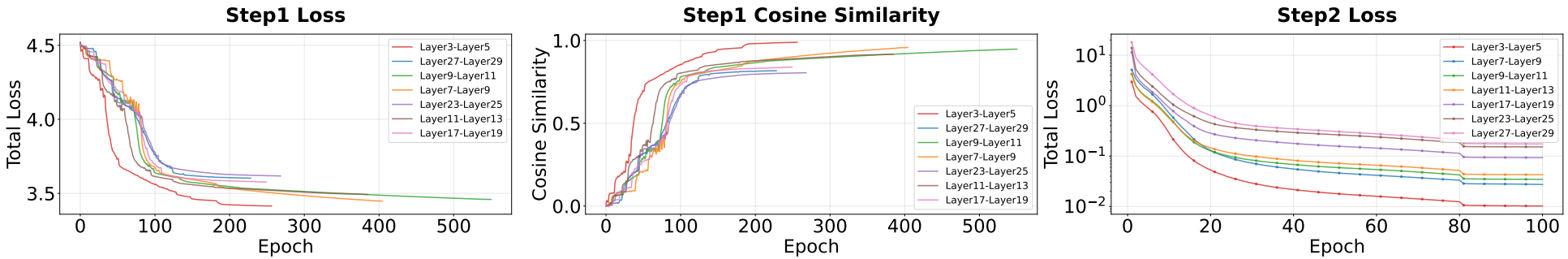
Finding 1: Representational patterns can be learned within the kernel space.

Further Analysis



Finding 2: As training progresses, $B_{\mathbf{A}}$ and R_{φ} behave well.

Further Analysis



Finding 3: Step 1 dominates the pruning performance, while Step 2 serves primarily as the inverse kernel mapping.

Conclusion

- We introduce a new perspective on layer pruning that simplifies the information flow by operating within a kernel space.
- Based on this perspective, we propose a novel method named as KDP.
- We demonstrate the effectiveness of our method through both theoretical analysis and empirical experiments.
- Our future work includes exploring more kernel functions and adapting our pruning method for multimodal LMs.

Thanks!

Q&A