

ReFT: Representation Finetuning for Language Models

разбор статьи

<https://arxiv.org/abs/2404.03592>

Дмитрий Колодезев @promsoft kolodezev.ru
2024.05.07 @ DataTalk

ReFT: Representation Finetuning for LM

Zhengxuan Wu^{*†} Aryaman Arora^{*†} Zheng Wang[†] Atticus Geiger[‡]

Dan Jurafsky[†] Christopher D. Manning[†] Christopher Potts[†]

[†]Stanford University [‡]Pr(Ai)²R Group

{wuzhengx, aryamana, peterwz, atticusg, jurafsky, manning, cgpotts}@stanford.edu

Abstract

Parameter-efficient fine-tuning (PEFT) methods seek to adapt large models via updates to a small number of *weights*. However, much prior interpretability work has shown that *representations* encode rich semantic information, suggesting that editing representations might be a more powerful alternative. Here, we pursue this hypothesis by developing a family of **Representation Finetuning (ReFT)** methods. ReFT methods operate on a frozen base model and learn task-specific interventions on hidden representations. We define a strong instance of the ReFT family, Low-rank Linear Subspace ReFT (LoReFT). LoReFT is a drop-in replacement for existing PEFTs and learns interventions that are 10×–50× more parameter-efficient than prior state-of-the-art PEFTs. We showcase LoReFT on eight commonsense reasoning tasks, four arithmetic reasoning tasks, Alpaca-Eval v1.0, and GLUE. In all these evaluations, LoReFT delivers the best balance of efficiency and performance, and almost always outperforms state-of-the-art PEFTs. We release a generic ReFT training library publicly at <https://github.com/stanfordnlp/pyreft>.

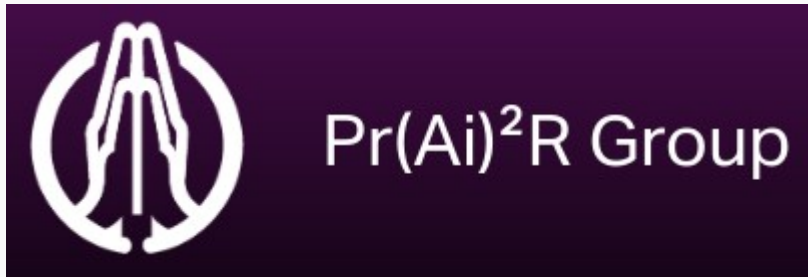
Не путать с
ReFT: Reasoning with Reinforced Fine-Tuning

Авторы

- Zhengxuan Wu - см. например, ruvene + статья
- Aryaman Arora - ruvene, causalgym + статья
- Zheng Wang - Benchmarking Benchmark Leakage
- Atticus Geiger - причинно-следственные связи в LLM
- Dan Jurafsky - он такой один
- Christopher D. Manning — он такой еще один
- Christopher Potts — ruvene, ravel + датасет

Про Стенфорд все понятно

- To develop the methodological and theoretical foundations of interpretable AI through the lens of causality and abstraction.
- We believe that the understanding and control provided by interpretability research will be fundamental to creating safe AI models that benefit humanity.



<https://prair.group/>

Скажи мне главные слова

Model & PEFT	Params (%)	Win-rate (↑)
GPT-3.5 Turbo 1106 [†]	—	86.30
Llama-2 Chat 13B [†]	—	81.10
Llama-2 Chat 7B [†]	—	71.40
Llama-2 7B & FT*	100%	80.93
Llama-2 7B & LoRA*	0.1245%	81.48
Llama-2 7B & RED*	0.0039%	81.69
Llama-2 7B & LoReFT (ours)	0.0039%	85.60
Llama-2 7B & LoReFT (ours, <i>half</i>)	0.0019%	84.12
Llama-2 7B & LoReFT (ours, <i>1K</i>) [†]	0.0039%	81.91

Table 3: Instruction tuning evaluation results for instruction-tuned Llama-2 7B with Alpaca-Eval v1.0. We report averaged performance of two runs with distinct random seeds for our method. *half* denotes our runs with half of the rank; *1K* denotes our runs with a low-resource setting where there is only 1K training examples. [†]Performance results of baseline methods are taken from Li et al. [2023]. *Performance results of baseline methods are taken from Wu et al. [2024a]. [†] **It takes ≈18 minutes to train our Llama-2 Chat 7B on a single A100 40G GPU with ≈1MB parameters on disk.**

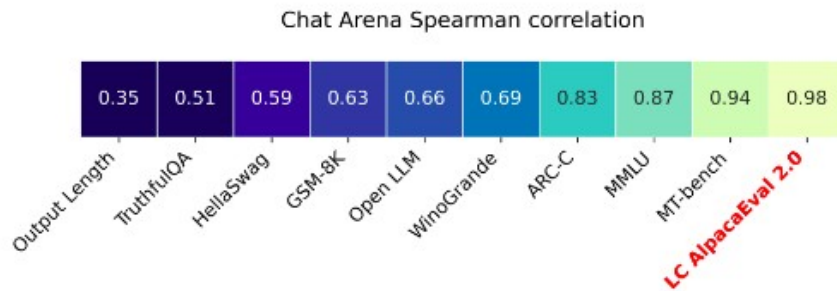
AlpacaEval



AlpacaEval : An Automatic Evaluator for Instruction-following Language Models

Code License [Apache 2.0](#) Data License [CC By NC 4.0](#) python [3.10+](#) [discord server](#)

AlpacaEval 2.0 with length-controlled win-rates ([paper](#)) has a spearman correlation of **0.98** with [ChatBot Arena](#) while costing less than **\$10** of OpenAI credits run and running in less than 3 minutes. Our goal is to have a benchmark for chat LLMs that is: fast (< 5min), cheap (< \$10), and highly correlated with humans (0.98). Here's a comparison with other benchmarks:



https://github.com/tatsu-lab/alpaca_eval

AlpacaEval

- Наша и базовая модель отвечают на вопросы
- Модель-аннотатор дообучена на 20к разметки
- Сравнивает ответы, присуждает победу
- Метрика - % побед
- V1: test-davinci-003 + alpaca_eval_gpt4
- V2: gpt4_turbo + weighted_alpaca_eval_gpt4_turbo
- 0.98 корреляция с **ChatBot Arena**

Предпосылки

Finding Alignments Between Interpretable Causal Variables and Distributed Neural Representations

Atticus Geiger* \diamond , Zhengxuan Wu*, Christopher Potts, Thomas Icard, and Noah D. Goodman

Pr(Ai)²R Group \diamond Stanford University

{atticusg, wuzhengx, cgpotts, icard, ngoodman}@stanford.edu

Editors: Francesco Locatello and Vanessa Didelez

Abstract

Causal abstraction is a promising theoretical framework for explainable artificial intelligence that defines when an interpretable high-level causal model is a faithful simplification of a low-level deep learning system. However, existing causal abstraction methods have two major limitations: they require a brute-force search over alignments between the high-level model and the low-level one, and they presuppose that variables in the high-level model will align with disjoint sets of neurons in the low-level one. In this paper, we present *distributed alignment search* (DAS), which overcomes these limitations. In DAS, we find the alignment between high-level and low-level models using gradient descent rather than conducting a brute-force search, and we allow individual neurons to play multiple distinct roles by analyzing representations in non-standard bases—*distributed* representations. Our experiments show that DAS can discover internal structure that prior approaches miss. Overall, DAS removes previous obstacles to uncovering conceptual structure in trained neural nets.

Где-то в линейных подпространствах активаций таится СМЫСЛ



А не в отдельных нейронах

Давайте шатать
модифицировать
подпространства и
смотреть, что выйдет

<https://arxiv.org/abs/2303.02536>

Предпосылки

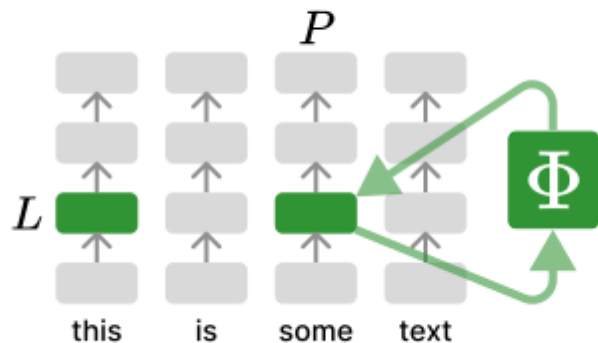
- Игралась с интерпретируемостью
- Формализовали интервенцию в активации

ReFT Intervention

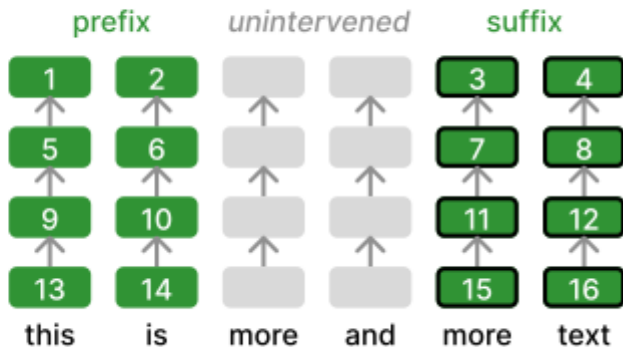
$$I = \langle \Phi, P, L \rangle$$

$$P = \{1, 3\}$$

$$L = 2$$



LoReFT



Распределенная интервенция

To test whether a concept is encoded in a linear subspace of a representation, as claimed by the linear representation hypothesis, one may use a **distributed interchange intervention** [Geiger et al. 2023b].³ Let \mathbf{b} be the hidden representation created at row i and column k when our model processes input b , and let \mathbf{s} be the corresponding representation when that same model processes input s . A distributed interchange intervention on \mathbf{b} given a counterfactual source representation \mathbf{s} is then defined as

$$\text{DII}(\mathbf{b}, \mathbf{s}, \mathbf{R}) = \mathbf{b} + \mathbf{R}^\top (\mathbf{R}\mathbf{s} - \mathbf{R}\mathbf{b}) \quad (1)$$

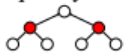
where $\mathbf{R} \in \mathbb{R}^{r \times d}$ is a low-rank projection matrix with orthonormal rows, d is the representation dimensionality, and r is the dimensionality of the subspace we are intervening on. We learn the subspace \mathbf{R} using distributed alignment search (DAS), which finds the subspace that maximises the probability of the expected counterfactual output after intervention.

И слепые вдруг заговорили

4.4. Analyzing a Randomly Initialized Network

To calibrate intuitions about our method, we evaluate the ability of DAS to optimize for interchange intervention accuracy on a frozen randomly initialized networks that achieves chance accuracy (50%) on the hierarchical equality task. This investigates the degree to which random causal structures can be used to systematically manipulate the counterfactual behavior of the network. We evaluate networks with different hidden representation sizes while holding the four input vectors fixed at 4 dimensions, under the hypothesis that more hidden neurons create more random structure that DAS can search through. These results are summarized in Table 4.4. Observe that, in small networks, there is no ability to increase interchange intervention accuracy. However, as we increase the size of the hidden representation to be orders of magnitude larger than the input dimension of 16, the interchange intervention accuracy increases. This confirms our hypothesis and serves as a check that demonstrates DAS cannot construct entirely new behaviors from random structure.

Both Equality Relations



Hidden size	Intervention size	Layer 1
$ \mathbf{N} = 16$	$k = 8$	0.50
$ \mathbf{N} = 64$	$k = 32$	0.50
$ \mathbf{N} = 256$	$k = 128$	0.51
$ \mathbf{N} = 1028$	$k = 512$	0.55
$ \mathbf{N} = 4096$	$k = 2048$	0.64

Figure 3: DAS on a random network with a 16 dimension input. An oversized hidden dimension allows DAS to manipulate the model behavior by searching through a large space of random mechanisms.

А давайте ЭТИМ ТЮНИТЬ

The formulation of Dll in eq. (1) immediately suggests a way to control model generations via interventions. The guiding intuition is that we can learn how to perform interventions that lead the model to accurately predict our task labels. The resulting method, Low-rank Linear Subspace ReFT (LoReFT), is defined by the following variant of eq. (1):

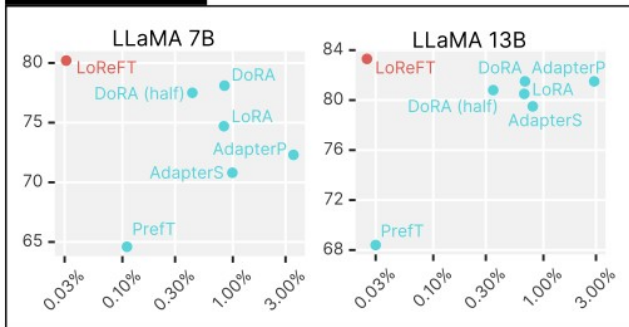
$$\Phi_{\text{LoReFT}}(\mathbf{h}) = \mathbf{h} + \mathbf{R}^\top (\mathbf{W}\mathbf{h} + \mathbf{b} - \mathbf{R}\mathbf{h}) \quad (2)$$

This is identical to eq. (1), except we use a *learned projected source* $\mathbf{R}\mathbf{s} = \mathbf{W}\mathbf{h} + \mathbf{b}$. Intuitively, LoReFT edits the representation in the r -dimensional subspace spanned by the columns of \mathbf{R} to take on the values obtained from our linear projection $\mathbf{W}\mathbf{h} + \mathbf{b}$.

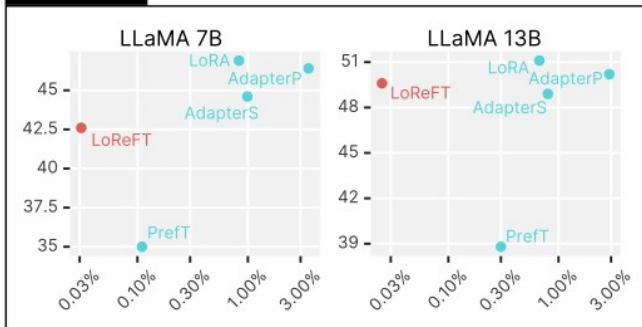
The learned parameters are $\phi = \{\mathbf{R}, \mathbf{W}, \mathbf{b}\}$. As with Dll, $\mathbf{R} \in \mathbb{R}^{r \times d}$ is a low-rank matrix with orthonormal rows where d is the hidden-state dimensionality and $r \leq d$ is the rank of the subspace. We further define a linear projection $\mathbf{W} \in \mathbb{R}^{r \times d}$ and bias vector $\mathbf{b} \in \mathbb{R}^r$. The parameters of the LM are frozen.

А давайте ЭТИМ ТЮНИТЬ

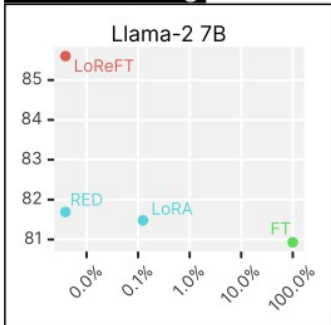
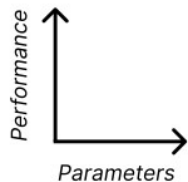
Commonsense



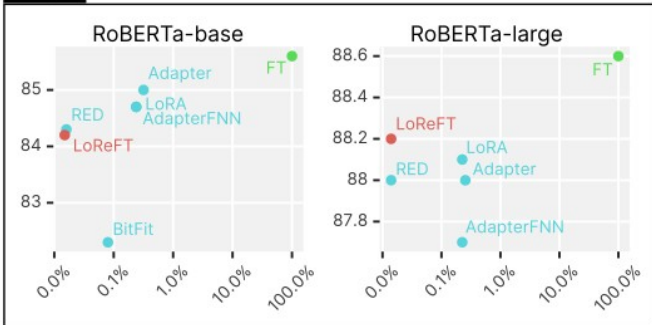
Arithmetic



Instruct-tuning



GLUE



Все придумано до нас, но нет

Representation editing. Recent work on *activation steering* and *representation engineering* shows that adding fixed steering vectors to the residual stream can enable a degree of control over pretrained LM generations without the need for resource-intensive finetuning [Subramani et al., 2022, Turner et al., 2023, Zou et al., 2023, Vogel, 2024]. For example, steering attention-head outputs can increase performance on TruthfulQA [Li et al., 2024]. Similarly, Wu et al. [2024a] show that editing representations with a learned scaling and translation operation can approach (but not surpass) the performance of LoRA on a variety of models and tasks with far fewer learned parameters. The success of these methods affirms that representations induced by pretrained LMs carry rich semantics. However, exploration of such methods has been sporadic and performance is sub-optimal; adapter-based PEFTs continue to be the state-of-the-art and impose no additional inference burden.

Interventional interpretability. Recent work in interpretability has used interventions under the framework of causal abstraction [Geiger et al., 2021] to test hypotheses about how LMs implement various behaviours. In particular, interventions on linear subspaces of representations have provided increasing evidence that human-interpretable concepts are encoded linearly; this includes linguistic features such as gender and number [Lasri et al., 2022, Wang et al., 2023, Hanna et al., 2023, Chintam et al., 2023, Yamakoshi et al., 2023, Hao and Linzen, 2023, Chen et al., 2023, Amini et al., 2023, Guerner et al., 2023, Arora et al., 2024], logical and mathematical reasoning [Wu et al., 2023], and entity attributes [Huang et al., 2024].

На чем пробовали

- **Commonsense reasoning** which contains eight commonsense reasoning datasets, including BoolQ [Clark et al., 2019], PIQA [Bisk et al., 2020], SIQA [Sap et al., 2019], HellaSwag [Zellers et al., 2019], WinoGrande [Sakaguchi et al., 2021], ARC-e, ARC-c [Clark et al., 2018], and OBQA [Mihaylov et al., 2018]. The task is formulated as a multiple-choice problem.
- **Arithmetic reasoning** which contains four datasets for math world problems, including AQuA [Ling et al., 2017], GSM8K [Cobbe et al., 2021], MAWPS [Koncel-Kedziorski et al., 2016], and SVAMP [Patel et al., 2021]. Models need to generate chain-of-thought [Wei et al., 2022] before the final answer.
- **Instruction-following** which evaluates whether models can follow human instructions. We use Ultrafeedback [Cui et al., 2023] as our training data, and Alpaca-Eval v1.0 [Li et al., 2023] as our evaluation dataset.
- **Natural language understanding** which contains eight datasets from the GLUE benchmark [Wang et al., 2018] such as sentiment analysis and natural language inference.

Гиперпараметры

- Два первых токена
- Два последних токена
- Подбирали на отдельном датасете

1. The number of prefix positions p to intervene on, i.e. positions $\{1, \dots, p\}$.
2. The number of suffix positions s to intervene on, i.e. positions $\{n - s + 1, \dots, n\}$.
3. Which set of layers L to intervene on.
4. Whether or not to tie intervention parameters ϕ across different positions in the same layer.

Commonsense reasoning

Model	PEFT	Params (%)	Accuracy (\uparrow)								
			BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	Avg.
ChatGPT*	—	—	73.1	85.4	68.5	78.5	66.1	89.8	79.9	74.8	77.0
LLaMA-7B	PrefT*	0.110%	64.3	76.8	73.9	42.1	72.1	72.9	54.0	60.6	64.6
	Adapter ^{S*}	0.990%	63.0	79.2	76.3	67.9	75.7	74.5	57.1	72.4	70.8
	Adapter ^{P*}	3.540%	67.9	76.4	78.8	69.8	78.9	73.7	57.3	75.2	72.3
	LoRA*	0.830%	68.9	80.7	77.4	78.1	78.8	77.8	61.3	74.8	74.7
	DoRA (half)*	0.430%	70.0	82.6	79.7	83.2	80.6	80.6	65.4	77.6	77.5
	DoRA*	0.840%	68.5	82.9	79.6	84.8	80.8	81.4	65.8	81.0	78.1
	LoReFT (ours)	0.031%	69.3	84.4	80.3	93.1	84.2	83.2	68.2	78.9	80.2
LLaMA-13B	PrefT*	0.030%	65.3	75.4	72.1	55.2	68.6	79.5	62.9	68.0	68.4
	Adapter ^{S*}	0.800%	71.8	83.0	79.2	88.1	82.4	82.5	67.3	81.8	79.5
	Adapter ^{P*}	2.890%	72.5	84.9	79.8	92.1	84.7	84.2	71.2	82.4	81.5
	LoRA*	0.670%	72.1	83.5	80.5	90.5	83.7	82.8	68.3	82.4	80.5
	DoRA (half)*	0.350%	72.5	85.3	79.9	90.1	82.9	82.7	69.7	83.6	80.8
	DoRA*	0.680%	72.4	84.9	81.5	92.4	84.2	84.2	69.6	82.8	81.5
	LoReFT (ours)	0.025%	72.1	86.3	81.8	95.1	87.2	86.2	73.7	84.2	83.3

Arithmetic reasoning

Model	PEFT	Params (%)	Accuracy (\uparrow)				
			AQuA	GSM8K	MAWPS	SVAMP	Avg.
LLaMA-7B	PrefT*	0.110%	14.2	24.4	63.4	38.1	35.0
	Adapter ^S *	0.990%	15.0	33.3	77.7	52.3	44.6
	Adapter ^P *	3.540%	18.1	35.3	82.4	49.6	46.4
	LoRA*	0.830%	18.9	37.5	79.0	52.1	46.9
	LoReFT (ours)	0.031%	21.4	26.0	76.2	46.8	42.6
LLaMA-13B	PrefT*	0.300%	15.7	31.1	66.8	41.4	38.8
	Adapter ^S *	0.800%	22.0	44.0	78.6	50.8	48.9
	Adapter ^P *	2.890%	20.5	43.3	81.1	55.7	50.2
	LoRA*	0.670%	18.5	47.5	83.6	54.6	51.1
	LoReFT (ours)	0.025%	23.6	38.1	82.4	54.2	49.6

Instruction-following

Model & PEFT	Params (%)	Win-rate (↑)
GPT-3.5 Turbo 1106 [†]	—	86.30
Llama-2 Chat 13B [†]	—	81.10
Llama-2 Chat 7B [†]	—	71.40
Llama-2 7B & FT*	100%	80.93
Llama-2 7B & LoRA*	0.1245%	81.48
Llama-2 7B & RED*	0.0039%	81.69
Llama-2 7B & LoReFT (ours)	0.0039%	85.60
Llama-2 7B & LoReFT (ours, half)	0.0019%	84.12
Llama-2 7B & LoReFT (ours, 1K) [†]	0.0039%	81.91

Table 3: Instruction tuning evaluation results for instruction-tuned Llama-2 7B with Alpaca-Eval v1.0. We report averaged performance of two runs with distinct random seeds for our method. *half* denotes our runs with half of the rank; *1K* denotes our runs with a low-resource setting where there is only 1K training examples. [†]Performance results of baseline methods are taken from Li et al. [2023]. *Performance results of baseline methods are taken from Wu et al. [2024a]. [†] **It takes ≈18 minutes to train our Llama-2 Chat 7B on a single A100 40G GPU with ≈1MB parameters on disk.**

Natural language understanding

Model	PEFT	Params (%)	Accuracy (↑)								
			MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
base	FT	100%	87.3	94.4	87.9	62.4	92.5	91.7	78.3	90.6	85.6
	Adapter*	0.318%	87.0	93.3	88.4	60.9	92.5	90.5	76.5	90.5	85.0
	LoRA*	0.239%	86.6	93.9	88.7	59.7	92.6	90.4	75.3	90.3	84.7
	Adapter ^{FNN} *	0.239%	87.1	93.0	88.8	58.5	92.0	90.2	77.7	90.4	84.7
	BitFit*	0.080%	84.7	94.0	88.0	54.0	91.0	87.3	69.8	89.5	82.3
	RED*	0.016%	83.9	93.9	89.2	61.0	90.7	87.2	78.0	90.4	84.3
	LoReFT (ours)	0.015%	83.1	93.4	89.2	60.4	91.2	87.4	79.0	90.0	84.2
large	FT	100%	88.8	96.0	91.7	68.2	93.8	91.5	85.8	92.6	88.6
	Adapter*	0.254%	90.1	95.2	90.5	65.4	94.6	91.4	85.3	91.5	88.0
	LoRA*	0.225%	90.2	96.0	89.8	65.5	94.7	90.7	86.3	91.7	88.1
	Adapter ^{FNN} *	0.225%	90.3	96.1	90.5	64.4	94.3	91.3	84.8	90.2	87.7
	RED*	0.014%	89.5	96.0	90.3	68.1	93.5	88.8	86.2	91.3	88.0
		LoReFT (ours)	0.014%	89.2	96.2	90.1	68.0	94.1	88.5	87.5	91.6

Table 4: Accuracy comparison of RoBERTa-base and RoBERTa-large against existing PEFT methods on the GLUE benchmark. *Performance results of all baseline methods are taken from Wu et al. [2024a]. We report averaged performance of five runs with distinct random seeds for our method. # Param. (%) is calculated by dividing the number of trainable parameters (excluding the number of parameters of the classification head) with the number of parameter of the base LM.

pip install pyreft

```
import torch
import transformers

from pyreft import (
    get_reft_model,
    ReftConfig,
    LoreftIntervention,
    ReftTrainerForCausalLM
)

# loading huggingface model
model_name_or_path = "yahma/llama-7b-hf"
model = transformers.AutoModelForCausalLM.from_pretrained(
    model_name_or_path, torch_dtype=torch.bfloat16, device_map="cuda")

# wrap the model with rank-1 constant reft
reft_config = ReftConfig(representations={
    "layer": 19, "component": "block_output",
    "intervention": LoreftIntervention(
        embed_dim=model.config.hidden_size, low_rank_dimension=1)})
reft_model = get_reft_model(model, reft_config)
reft_model.print_trainable_parameters()
```

pip install pyreft

```
tokenizer = transformers.AutoTokenizer.from_pretrained(model_name_or_path)

# get training data with customized dataloaders
data_module = make_supervised_data_module(
    tokenizer=tokenizer, model=model, layers=[19],
    training_args=training_args, data_args=data_args)

# train
trainer = reft.ReftTrainerForCausalLM(
    model=reft_model, tokenizer=tokenizer, args=training_args, **data_module)
trainer.train()
trainer.save_model(output_dir=training_args.output_dir)
```

<https://github.com/stanfordnlp/pyreft>

<https://github.com/stanfordnlp/pyreft/tree/main/examples/chat>

Чем это отличается от

General comments about expressivity of ReFT. Given that previous works have unified PEFTs under a single framework [He et al., 2022a], one may ask **why not express ReFT as a PEFT method?** First of all, PEFT frameworks lacks the notion of *time* or *sequence* (see the unified PEFT view provided in Table 1 on pg. 5 of He et al. [2022a]). In PEFT, representation modifications are applied to *every* token in the sequence, even in recent variants such as AdaLoRA [Zhang et al., 2023]. A key aspect of ReFT is that it leverages representations over time and intervenes only on a small number of them while being effective. More importantly, the notation of time is important for future versions of ReFT that intervene on representations *schematically* (e.g. intervene on the first token at some early layers and then intervene on the last token at some later layers). The ability to intervene at different layer and position locations schematically is also implemented in our code. Existing popular PEFT libraries⁶ enforce *weight-based* updates without conveniently supporting flexible representation-based interventions.

Настройка гиперпараметров

- **Intervening on multiple positions delivers significant gains.** We find that intervening only on a single token position (e.g., just the first one or the last one) is always less optimal than intervening on multiple tokens. However, intervening on excessive number of tokens might harm performance by slowing down convergence.
- **Intervening on all layers first, and then shrink down.** Intervening on all layers often provides a good baseline. We recommend users to start with all layers, and shrink down the number of intervening layers depending on the desired performance–parameter count balance.
- **Higher rank may not entail better performance.** High rank entails higher parameter count, but it does not always bring performance gain (likely due to slower convergence). We recommend users to start with a rank that is lower than 32 (e.g. rank 4).
- **Tie intervention weights as much as you can.** In the paper, we explore tying the intervention weights between prefix and suffix token positions. It automatically halves the parameter count, and it can result in better performance as well. We suspect weight sharing across layers may also help.
- **Hyperparameter tuning with learning rate, warmup ratio, dropout rate and weight decay should go after other hyperparameters.** These classic neural-network training hyperparameters can play a role, yet they have much smaller effect than previous ones.

Сколько Алис в одном токене

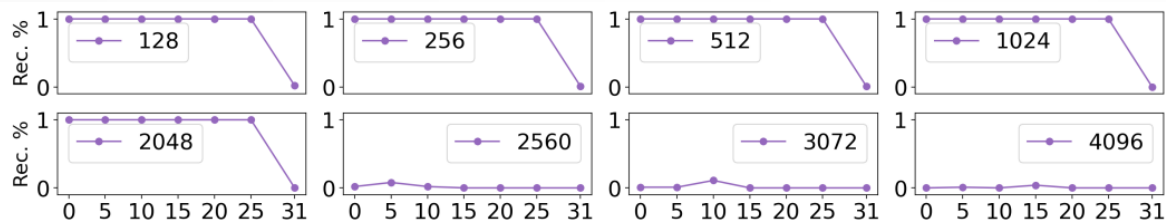


Figure 3: Memorisation test results for **LLaMA-1 7B model** on recovering first n-th tokens of the Alice's Adventures in Wonderland by rank-1 LoReFT intervention on various layers of the last token's residual stream. Rec. % is measured by the percentage of prefix matches.

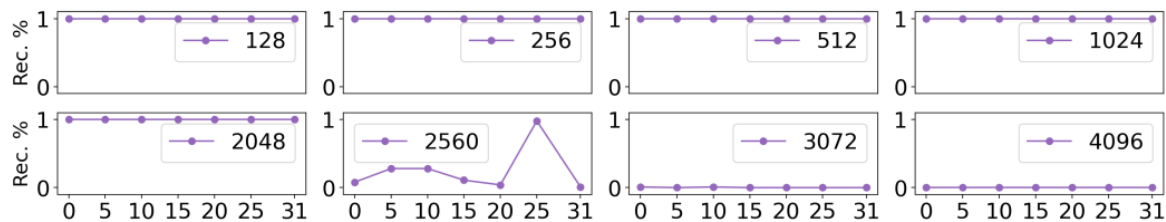


Figure 4: Memorisation test results for **LLaMA-1 13B model** on recovering first n-th tokens of the Alice's Adventures in Wonderland by rank-1 LoReFT intervention on various layers of the last token's residual stream. Rec. % is measured by the percentage of prefix matches.

Варкалось. Хливкие шорьки

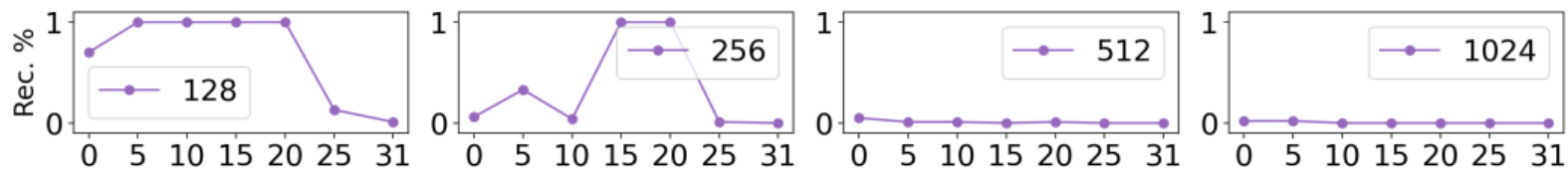


Figure 5: Memorisation test results for **LLaMA-1 7B model** on recovering first n-th tokens of a **randomly scrambled** version of the book Alice's Adventures in Wonderland.

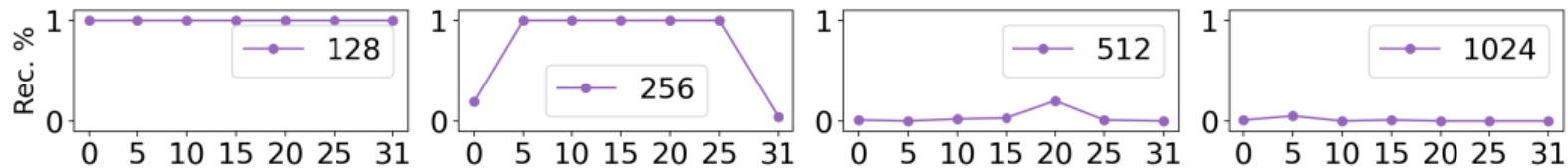


Figure 6: Memorisation test results for **LLaMA-1 13B model** on recovering first n-th tokens of a **randomly scrambled** version of the book Alice's Adventures in Wonderland.

Все равно что запоминать

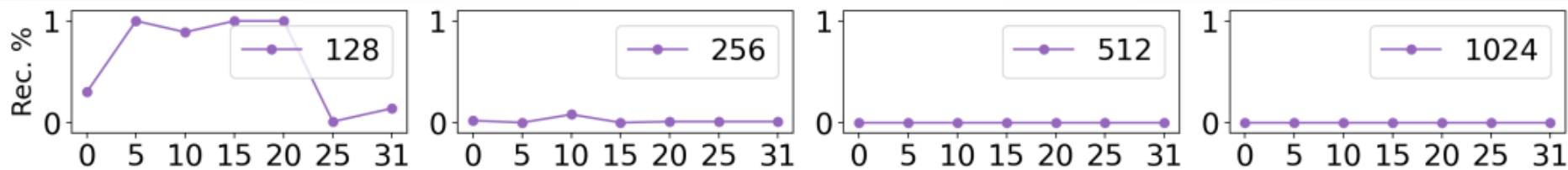


Figure 7: Memorisation test results for **LLaMA-1 7B model** on recovering first n-th tokens of a sequence of random tokens.

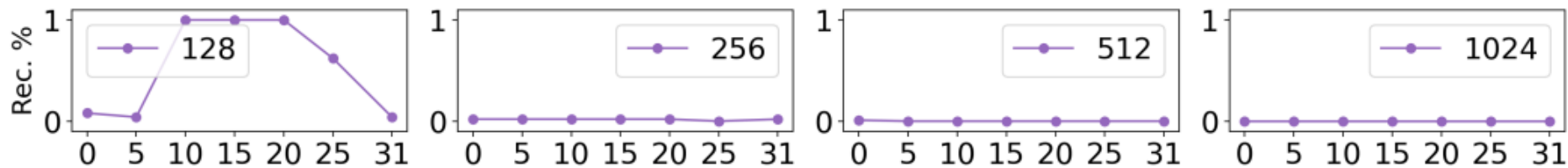


Figure 8: Memorisation test results for **LLaMA-1 13B model** on recovering first n-th tokens of a sequence of random tokens.

А еще они комбинируются

How can we reduce air pollution?

Response

Model
LLaMA-7B (LoReFT, German completion subspace)

Response
, die Luft wird sauber gemacht.

LLaMA-7B (LoReFT, Instruction following completion subspace)

Sure, here are some ways to reduce air pollution:

1. Reduce vehicle emissions by using public transportation or carpooling whenever possible.
2. Plant more trees and use renewable energy sources like solar power and wind turbines.
3. Recycle and reuse materials instead of throwing them away.
4. Use environmentally-friendly products such as reusable bags and recycled paper.
5. Avoid burning fossil fuels for heating and cooking.
6. Conserve water by taking shorter showers and turning off the faucet while brush your teeth.

LLaMA-7B (LoReFT, Combined subspace)

Einige Lösungen sind:

- * Wenn Sie Autofahren, fahren Sie auf der Straße mit weniger Verkehr und sparen Sie damit Kraftstoffe.
- * Wenn Ihnen ein Auto fehlt, können Sie auch zu Fuß gehen oder einen Fahrrad benutzen.
- * Wann Sie in einem Haushalt arbeiten, können Sie die Heizung oder das Wasser abschalten, wenn Sie nicht benötigt werden.

Комбинируются!

How to kill a linux process?

Response

Model

LLaMA-7B (LoReFT, German completion subspace)

Response

Es wird ein Linux-Prozess getötet, indem man ihn mit dem Befehl "kill" tötet.

LLaMA-7B (LoReFT, Instruction following completion subspace)

To kill a Linux process, you can use the 'kill' command with the PID (process ID) of the process you want to terminate. For example, if you want to kill the process with PID 123456789, you would run the following command:

```
““  
$ kill -9 123  
””
```

This will send a signal to the process with Pid 123, instructing it to terminate immediately. The '-9' flag indicates that the process should be terminated forcefully and without any further warning or prompts.

Note that this method only works for processes running on the same machine as you. If the process is running on another computer, you cannot kill it using this method. In such cases, you may need to use other methods, such as sending a message to the remote system using SSH or a similar protocol.

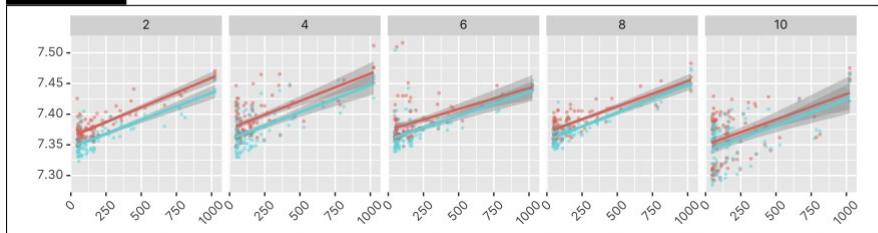
LLaMA-7B (LoReFT, Combined subspaces)

Es gibt verschiedene Möglichkeiten, um einen Linux-Prozess zu löschen.

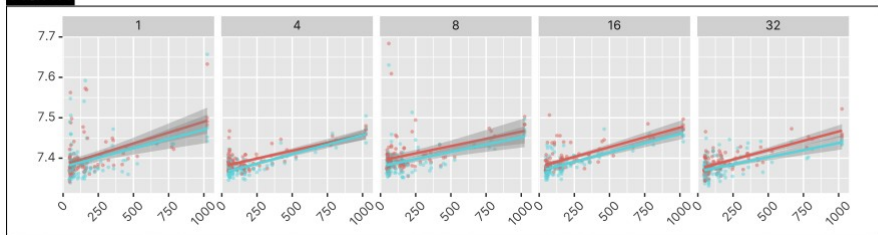
1. Mit dem Kommando "kill" kann man den Prozess beenden.
2. Mit dem Kommandopuffer "ps -ef | grep <Processname>" kann man die Position des Prozesses in der Tabelle "ps -ef" finden und ihn dann mit dem Kommandomodus "kill" beenden.

Без тормозов (ну почти)

Positions



Rank



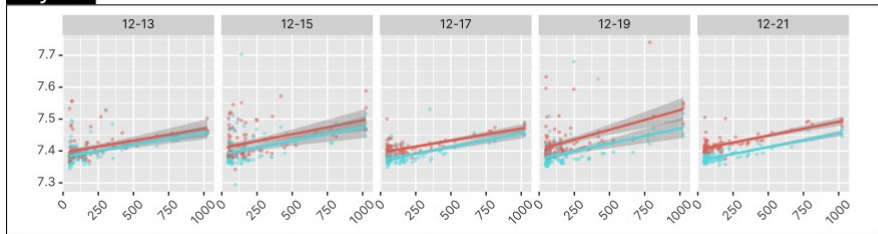
type

LoReFT

No intervention

Time
Prompt tokens

Layers



ИТОГО

- Можно дешево учить instruct модели
- Очень хорошая работа
- Можно не ломать модель инстракт тюнингом
- Куча интересных подробностей и ссылок
- Побежал пробовать

Поясните за арифметику

- Авторы отвечают на вопросы
- На тупые и на нетупые тоже

[@zhengxuanzenwu](#) This is a very interesting paper! Great work! In the results section, it looks like a lot of methods outperform LoReFT on the Arithmetic reasoning task. Do you have a hypothesis for why this is? My initial thought is that maybe it's not as simple a task to capture mathematical representations in the output of attention layers?

EDIT: In the paper, you say that the length of the generations might have something to do with this. Did you run any tests to see if the effectiveness of the method reduces as the length of generation increases?

<https://huggingface.co/papers/2404.03592>

Мы пока сами не поняли

 [zhengxuanzenwu](#) Paper author · 20 days ago · edited 20 days ago



@[shamikbose89](#) Thanks for your interest! Yes, LoReFT underperforms for arithmetic reasoning tasks, especially for GSM8K. In short, we don't know how to fix it yet. But here are a couple of hypotheses:

- Hyperparameter selection is not optimal. Although we tried hyperparameter tuning, our grid search is still pretty limited. We also haven't tried layerwise intervention weights sharing, etc.
- Intervening on decoding steps might help. Currently, we only intervene on the prompt. It is surprising that this is sufficient for the other two tasks with LLaMA models. For math reasonings which require CoT generations, intervening on decoding steps might help with long-form reasonings.
- More complex parameterization of the intervention. LoReFT is just one way of defining the intervention function. Coming up with more complex interventions could help.

Offline, we also tried to train and test on GSM8K only (the GSM8K dataset is also cleaner without GPT4 generated CoTs). LoReFT with Llama-2 still underperforms compared to LoRA + Llama-2 7B slightly (approximately 32% vs. 35%). However, LoReFT definitely has much fewer trainable parameters. See [LoftQ](#) paper for Llama-2 performance on GSM8K.