Application of Subset Selection in Efficient Machine Learning

IBM Maitreyee Research Showcase

Kiran Purohit

Advisor: **Prof. Sourangshu Bhattacharya**

Dept. of Computer Science & Engineering
IIT Kharagpur

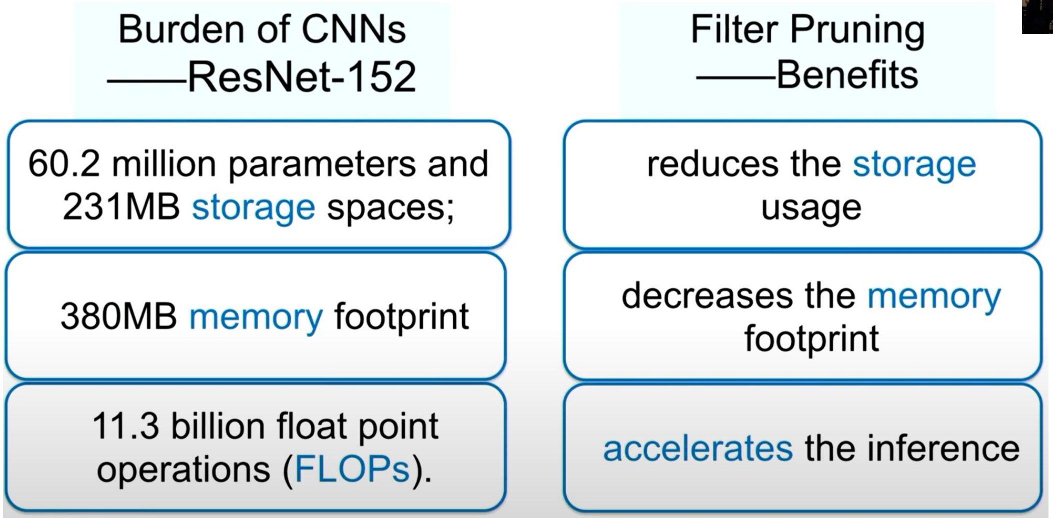
Outline

1. A Greedy Hierarchical Approach to Whole-Network Filter-Pruning in CNNs *(TMLR 2024)*

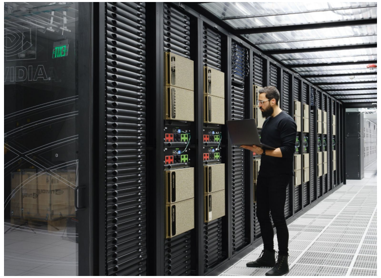
2. EXPLORA: Efficient Exemplar Subset Selection for Complex Reasoning *(EMNLP-main (long) 2024)*

MISTRAL
AI_

Filter Pruning

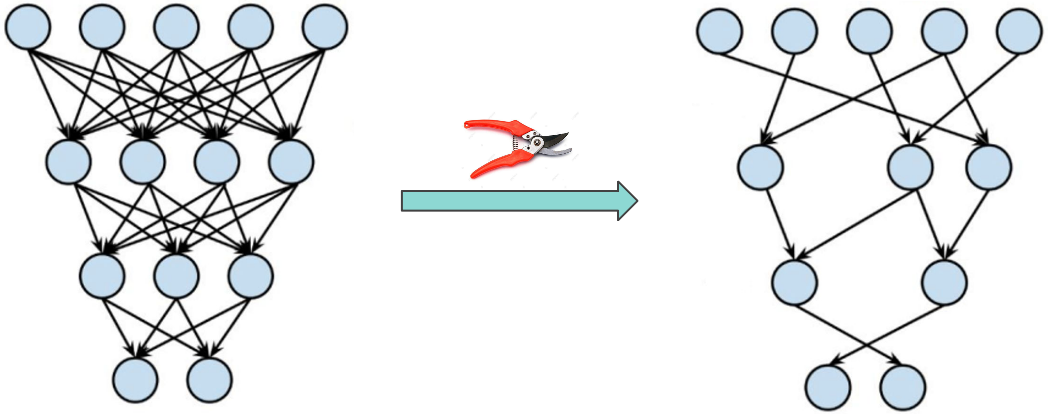


Burden of CNNs ——ResNet-152	Filter Pruning ——Benefits
60.2 million parameters and 231MB storage spaces;	reduces the storage usage
380MB memory footprint	decreases the memory footprint
11.3 billion float point operations (FLOPs).	accelerates the inference



Network Pruning

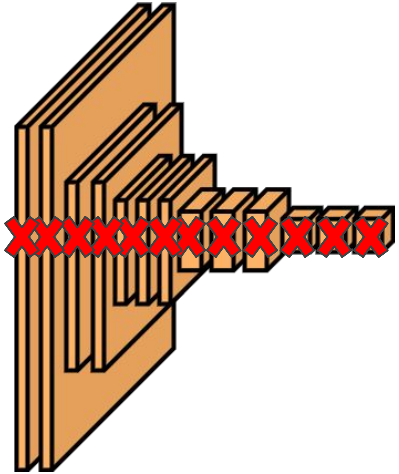
Given a pre-trained network $\Phi(.)$, the goal is to compress the network while maintaining the high performance as much as possible by removing the unnecessary parameters.



Pre-trained original network $\Phi(.)$

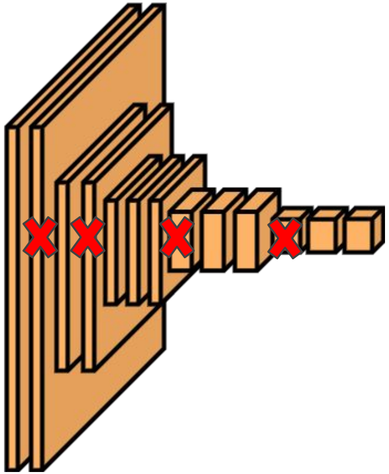
Final pruned network $\Phi'(.)$

Filter Pruning



Uniform Pruning

- ❖ Prune filters uniformly from each layer
- ❖ Process each layer independently and sequentially.



Non-Uniform Pruning

- ❖ Prune different fractions of filters from each layer
- ❖ All the layers in the network collectively make the final prediction

* accepted at **AIMLSystems 2022** (*Accurate and Efficient Channel pruning via Orthogonal Matching Pursuit*)

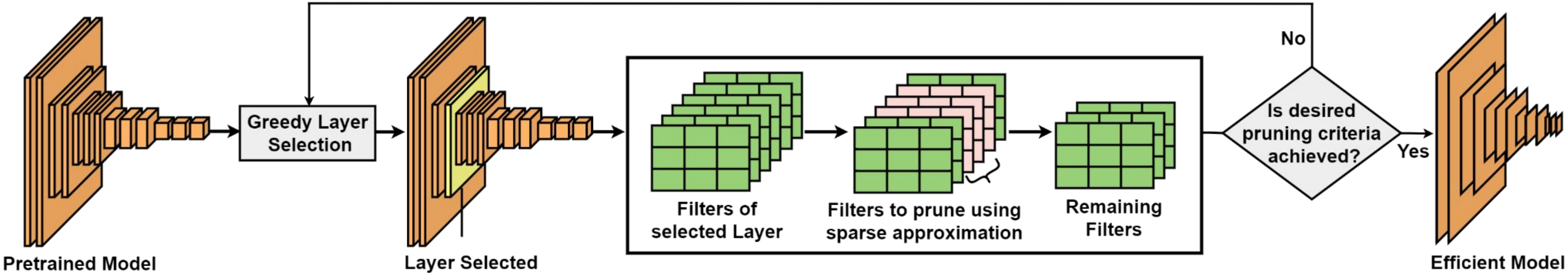
A Greedy Hierarchical Approach

- We developed faster **non-uniform pruning** methods.

- We used a hierarchical scheme with two-levels:

    - **filter pruning** - this step identifies the most appropriate filters to be pruned from each layer.
    - **layer selection** - this step selects the best layer to currently prune from.

  We apply these two steps iteratively to achieve a non-uniform pruning.



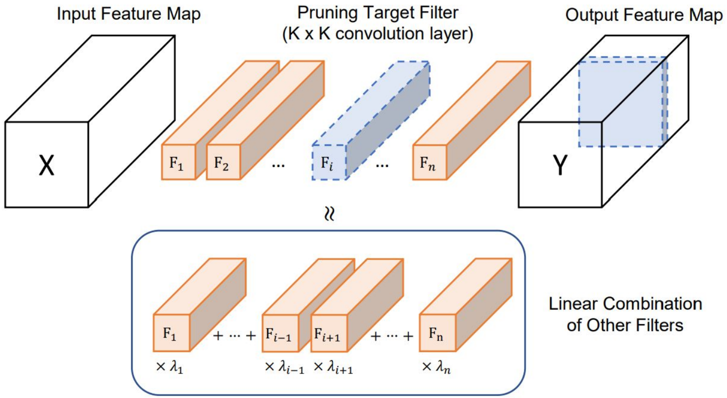
Pretrained Model
Greedy Layer Selection
Layer Selected
Filters of selected Layer
Filters to prune using sparse approximation
Remaining Filters
Is desired pruning criteria achieved?
No
Yes
Efficient Model

* accepted at **TMLR 2024** (*A Greedy Hierarchical Approach to Whole-Network Filter-Pruning in CNNs*)

Related Work

LRF

“Linearly Replaceable Filters for Deep Network Channel Pruning” AAAI 2021



❖ LRF suggests that *we can replace the filter that can be approximated by the linear combination of other filters*

Related Work

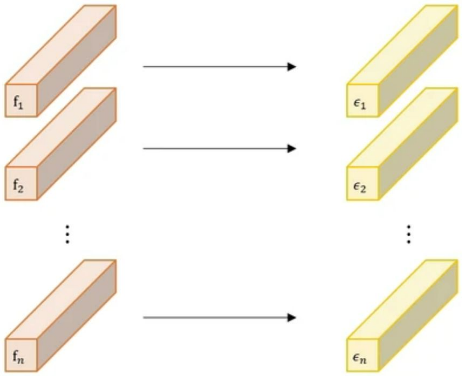
- ❖ In a layer, we can approximate each filter as a linear combination of the other filters

$$f_{:,j} = \sum_{l \neq j} \lambda_{j,l} f_{:,l} + \epsilon_j$$

Here, ϵ = approximation error and $\lambda_{j,l}$ = weight coefficient of the respective filters

- ❖ Each $\lambda_{j,l}$ can be found by solving following minimization problem

$$\min_{\lambda_{j,:}} ||f_{:,j} - \sum_{l \neq j} \lambda_{j,l} f_{:,l}||^2$$



Remove the ith filter with the smallest $||\epsilon_i||$

FP-OMP for Pruning Multiple Filters

We develop an Orthogonal Matching Pursuit (OMP) based algorithm for selecting retained filters of a layer into S. Hence filters that are to be pruned are {1,2,...n} \ S.

We can *approximate the pruned filters in terms of retained filters.*

$$f_{:,j} = \sum_{l\in S} \lambda_{j,l} f_{:,l} + \epsilon_j, \forall j \notin S$$

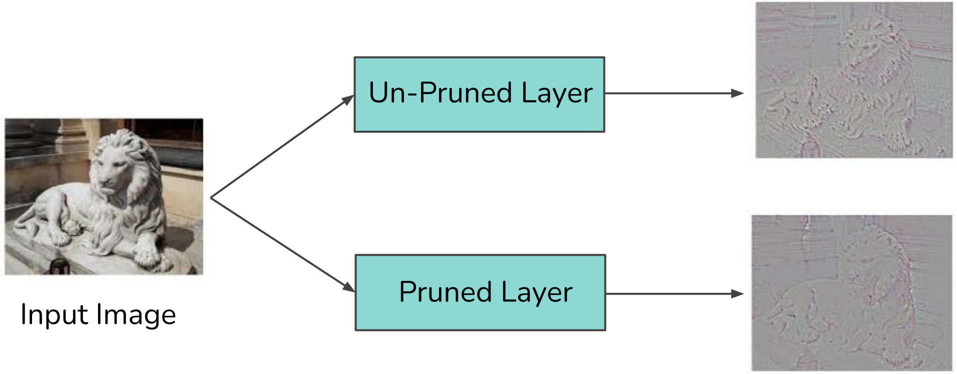
We pose a sparse approximation problem for finding **S** and ***λ***

$$S^*, \lambda^* = \mathrm{argmin}_{|S|\leq(1-\beta)n,\lambda} \sum_{j\in\{1,2,..,n\}} ||f_{:,j} - \sum_{l\in S} \lambda_{j,l} f_{:,l}||^2$$

where **S** is the set of the selected/retained filters in a layer, **n** is the total number of filter in that layer, and ***β*** is the pruning fraction

HBGS for Layer Selection

- We develop Hierarchical Backward Greedy Search (HBGS) for selecting the best layer to currently prune from.
- Key idea here is to calculate the relative reconstruction error between the pruned layer output and unpruned layer output
  - and then finally choose the layer with minimum error to currently prune from.



Input Image

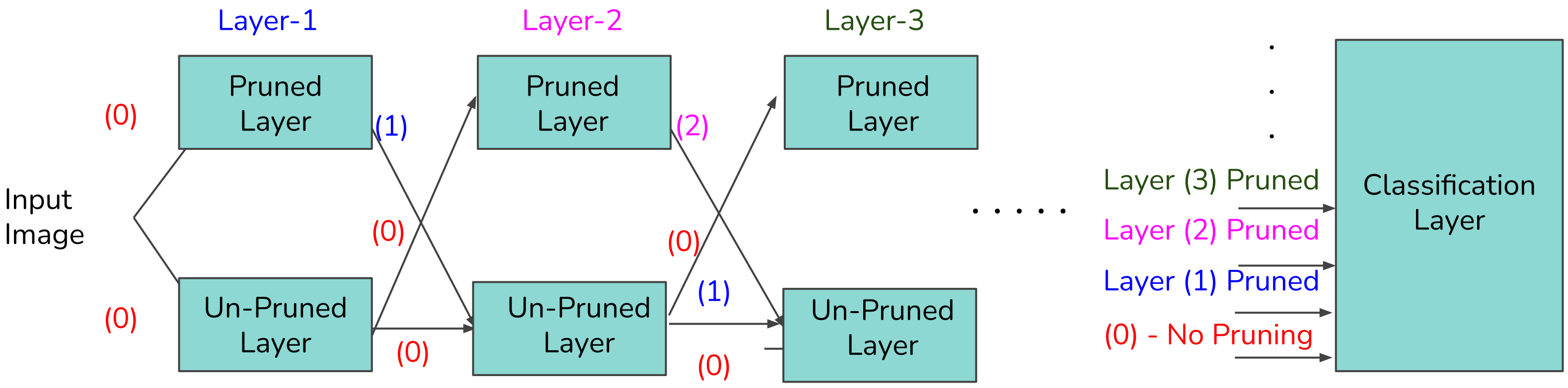
Un-Pruned Layer

Pruned Layer

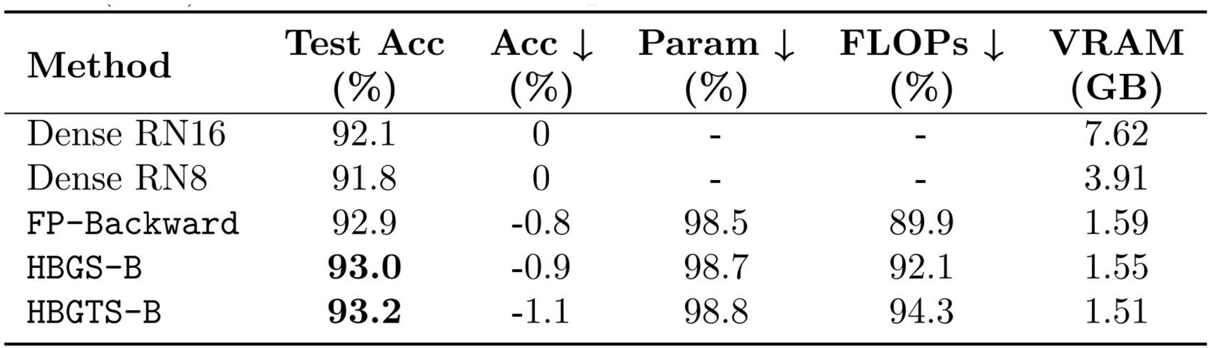
We want to minimise the difference b/w the output feature maps of pruned and unpruned layers

HBGTS for Layer Selection

- We develop Hierarchical Backward Greedy Tree Search (HBGTS) for selecting the best layer to currently prune from.
- Key idea here is to calculate the error in final layer output, if layer j ∈ {1, ..., C} is pruned
    - and then finally choose the layer with minimum error to currently prune from.



Results and Analysis



Method	Test Acc (%)	Acc ↓ (%)	Param ↓ (%)	FLOPs ↓ (%)	VRAM (GB)
Dense RN16	92.1	0	-	-	7.62
Dense RN8	91.8	0	-	-	3.91
FP-Backward	92.9	-0.8	98.5	89.9	1.59
HBGS-B	**93.0**	-0.9	98.7	92.1	1.55
HBGTS-B	**93.2**	-1.1	98.8	94.3	1.51

Table: Comparison of pruning methods for ResNext101 32x16d (RN16) and a similar sized dense ResNext101 32x8d (RN8) on CIFAR10 at 98% parameter reduction.

- Our greedy hierarchical methods can be used for effectively pruning large models that exceed the capacity of commodity GPUs.
- ResNext101 32x16d has 193 M parameters and requires 7.62 GB of GPU memory for loading.
- We can efficiently deploy the pruned model on edge devices with GPU memory less than 2GB.

In-Context Learning (ICL)

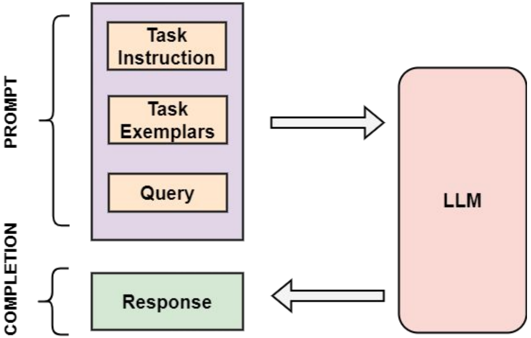
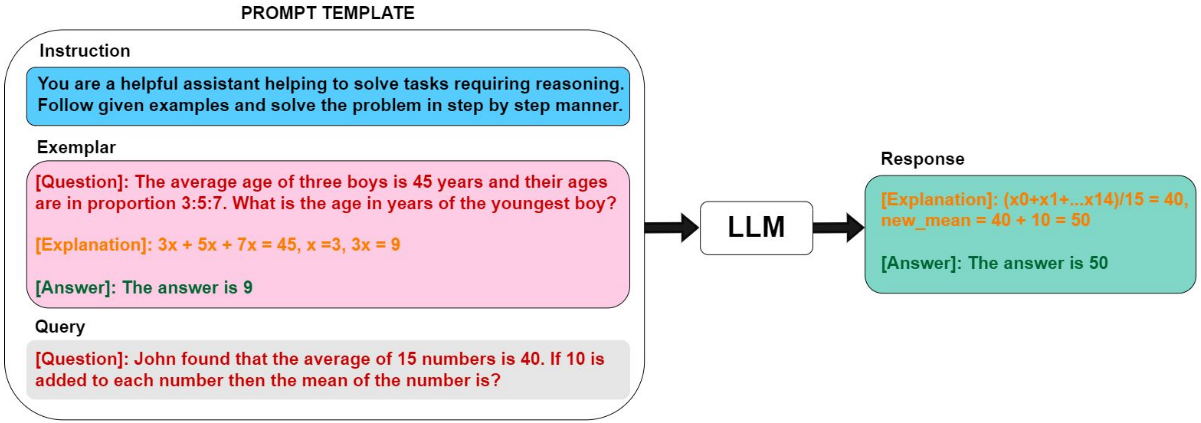


Fig: Block Diagram of ICL

Exemplars / In-context examples / demonstration samples

<Question, Explanation, Answer>



You are a helpful assistant helping to solve tasks requiring reasoning. Follow given examples and solve the problem in step by step manner.

[Question]: The average age of three boys is 45 years and their ages are in proportion 3:5:7. What is the age in years of the youngest boy?

[Explanation]: 3x + 5x + 7x = 45, x =3, 3x = 9

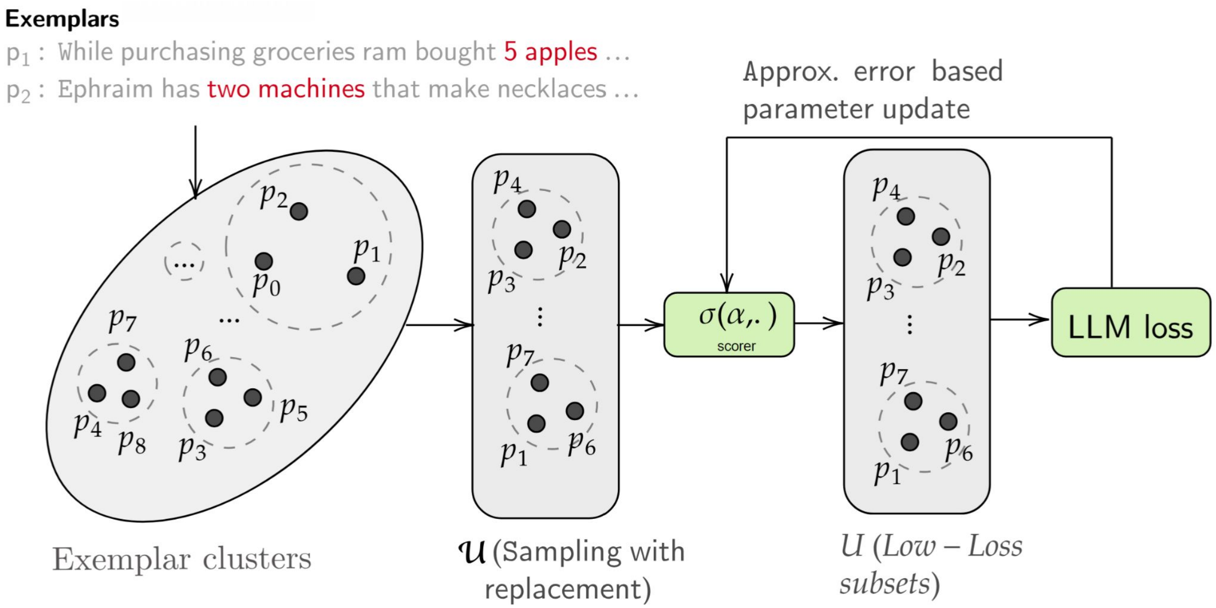
[Answer]: The answer is 9

[Question]: John found that the average of 15 numbers is 40. If 10 is added to each number then the mean of the number is?

[Explanation]: (x0+x1+...x14)/15 = 40, new_mean = 40 + 10 = 50

[Answer]: The answer is 50

Explore-Exploit Paradigm



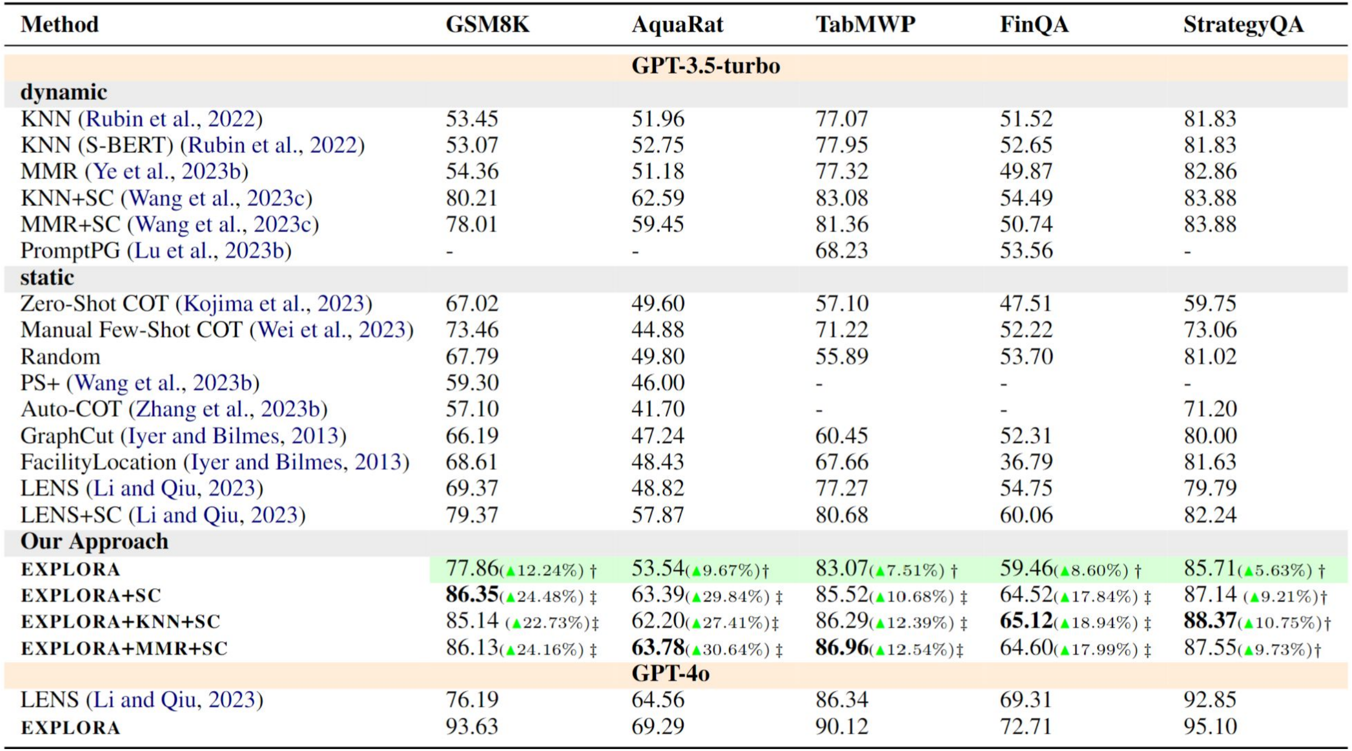
Exemplars

p_1 : While purchasing groceries ram bought 5 apples …

p_2 : Ephraim has two machines that make necklaces …

* accepted at **EMNLP-main (long) 2024** (*EXPLORA: Efficient Exemplar Subset Selection for Complex Reasoning*)

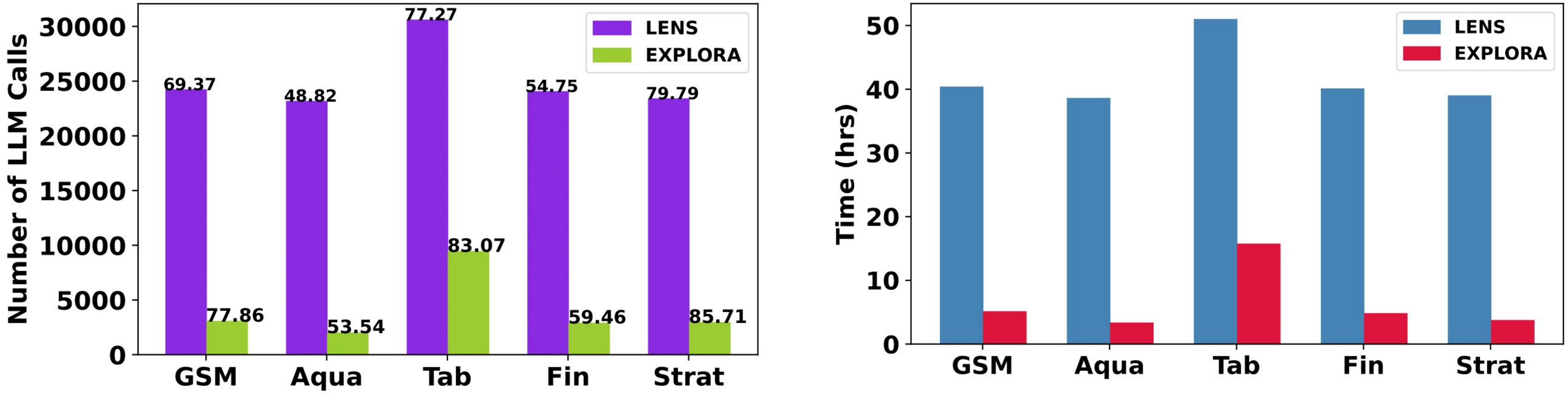
Results and Analysis



Method	GSM8K	AquaRat	TabMWP	FinQA	StrategyQA
		GPT-3.5-turbo			
dynamic					
KNN (Rubin et al., 2022)	53.45	51.96	77.07	51.52	81.83
KNN (S-BERT) (Rubin et al., 2022)	53.07	52.75	77.95	52.65	81.83
MMR (Ye et al., 2023b)	54.36	51.18	77.32	49.87	82.86
KNN+SC (Wang et al., 2023c)	80.21	62.59	83.08	54.49	83.88
MMR+SC (Wang et al., 2023c)	78.01	59.45	81.36	50.74	83.88
PromptPG (Lu et al., 2023b)	-	-	68.23	53.56	-
static					
Zero-Shot COT (Kojima et al., 2023)	67.02	49.60	57.10	47.51	59.75
Manual Few-Shot COT (Wei et al., 2023)	73.46	44.88	71.22	52.22	73.06
Random	67.79	49.80	55.89	53.70	81.02
PS+ (Wang et al., 2023b)	59.30	46.00	-	-	-
Auto-COT (Zhang et al., 2023b)	57.10	41.70	-	-	71.20
GraphCut (Iyer and Bilmes, 2013)	66.19	47.24	60.45	52.31	80.00
FacilityLocation (Iyer and Bilmes, 2013)	68.61	48.43	67.66	36.79	81.63
LENS (Li and Qiu, 2023)	69.37	48.82	77.27	54.75	79.79
LENS+SC (Li and Qiu, 2023)	79.37	57.87	80.68	60.06	82.24
Our Approach					
EXPLORA	77.86(▲12.24%) †	53.54(▲9.67%)†	83.07(▲7.51%) †	59.46(▲8.60%) †	85.71(▲5.63%) †
EXPLORA+SC	**86.35**(▲24.48%) ‡	63.39(▲29.84%) ‡	85.52(▲10.68%) ‡	64.52(▲17.84%) ‡	87.14 (▲9.21%)†
EXPLORA+KNN+SC	85.14 (▲22.73%)‡	62.20(▲27.41%)‡	86.29(▲12.39%) ‡	**65.12**(▲18.94%) ‡	**88.37**(▲10.75%)†
EXPLORA+MMR+SC	86.13(▲24.16%) ‡	**63.78**(▲30.64%) ‡	**86.96**(▲12.54%)‡	64.60(▲17.99%) ‡	87.55(▲9.73%)†
		GPT-4o			
LENS (Li and Qiu, 2023)	76.19	64.56	86.34	69.31	92.85
EXPLORA	93.63	69.29	90.12	72.71	95.10

Table: Results across datasets in transfer setting using gpt-3.5-turbo with exemplars selected from Mistral-7b.

Results and Analysis (Cont.)



(a)

(b)

Table: (a) **LLM calls** LENS vs EXPLORA (y-axis) with corresponding EM scores indicated on top of bars. (b) **Runtime** comparison

THANK YOU
FOR
YOUR ATTENTION!!!

https://github.com/kiranpurohit/

@kiranpurohit08