Application of Subset Selection in Robust and Efficient Machine Learning

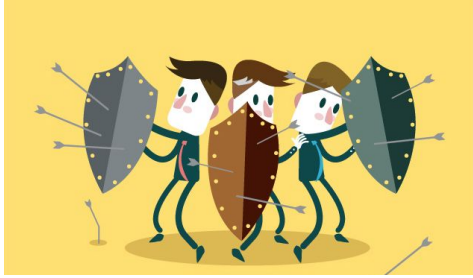
Online Research Development Program on
Exploring AI and ML Applications

Kiran Purohit (PhD Scholar)

Advisor: **Prof. Sourangshu Bhattacharya**

Dept. of CSE, IIT Kharagpur

Outline



1. EXPLORA: Efficient Exemplar Subset Selection for Complex Reasoning *(EMNLP-main (long) 2024)*
2. A Greedy Hierarchical Approach to Whole-Network Filter-Pruning in CNNs *(TMLR 2024)*
3. A Data-Driven Defense against Edge-case Model Poisoning Attacks on Federated Learning *(ECAI 2024)*

MISTRAL AI_

EXPLORA: Efficient Exemplar Subset Selection for Complex Reasoning

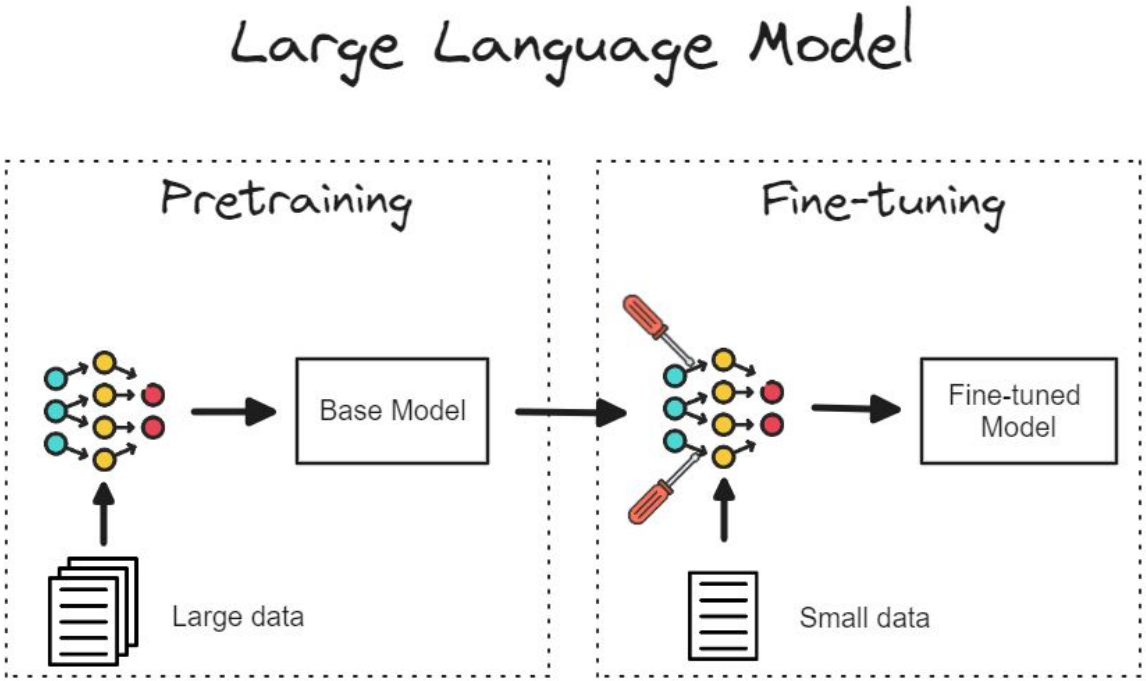
Kiran Purohit, Venktesh V, Raghuram Devalla, Krishna Mohan Yerragorla,
Sourangshu Bhattacharya, Avishek Anand



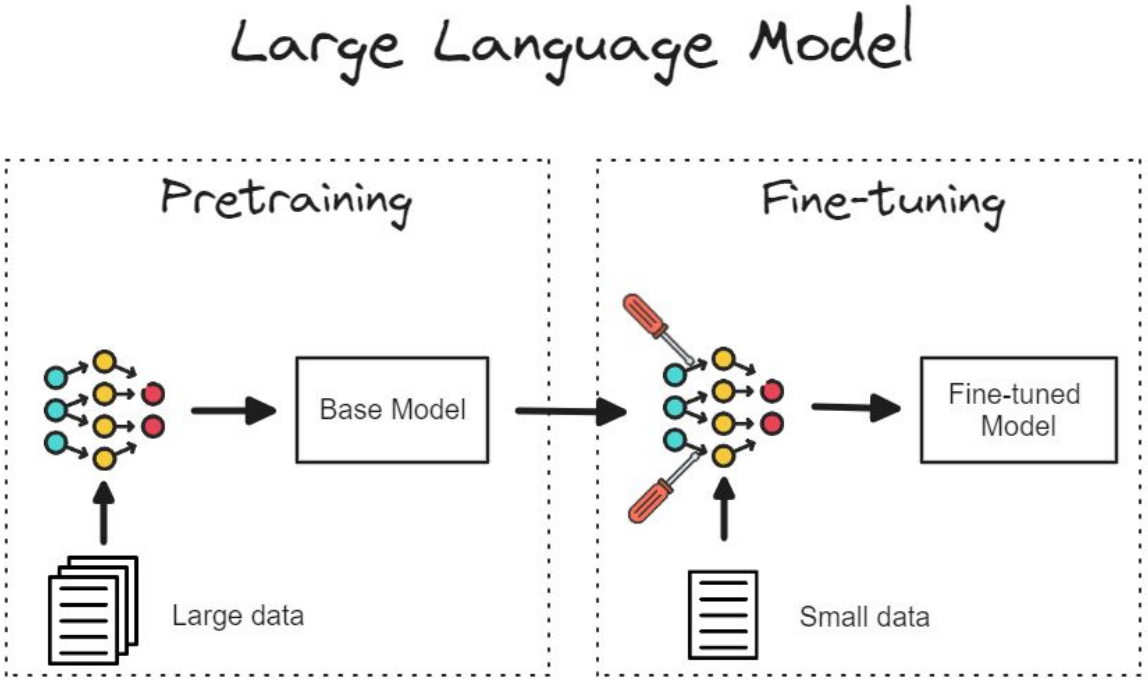



Dept. of Computer Science & Engineering
IIT Kharagpur

How to make model adapt to new task?



How to make model adapt to new task?



In-Context Learning (ICL)

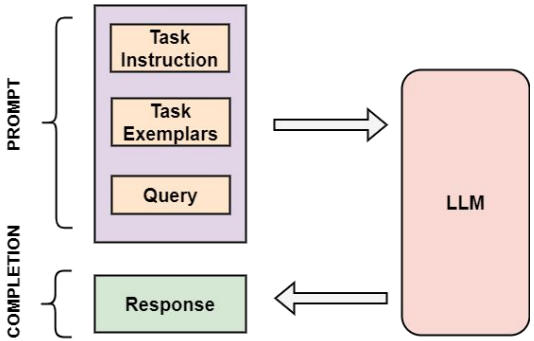
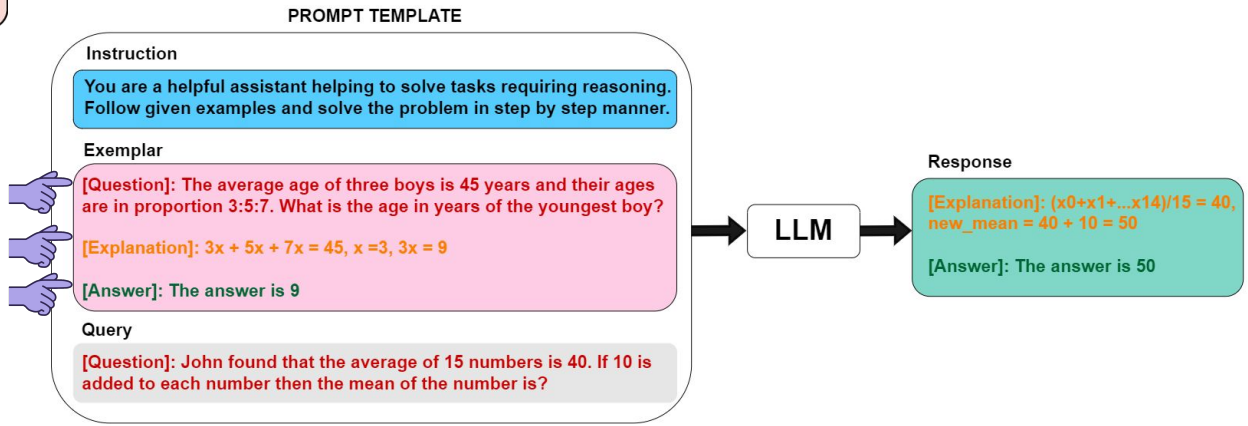


Fig: Block Diagram of ICL

Exemplars / In-context examples / demonstration samples
<Question, Explanation, Answer>

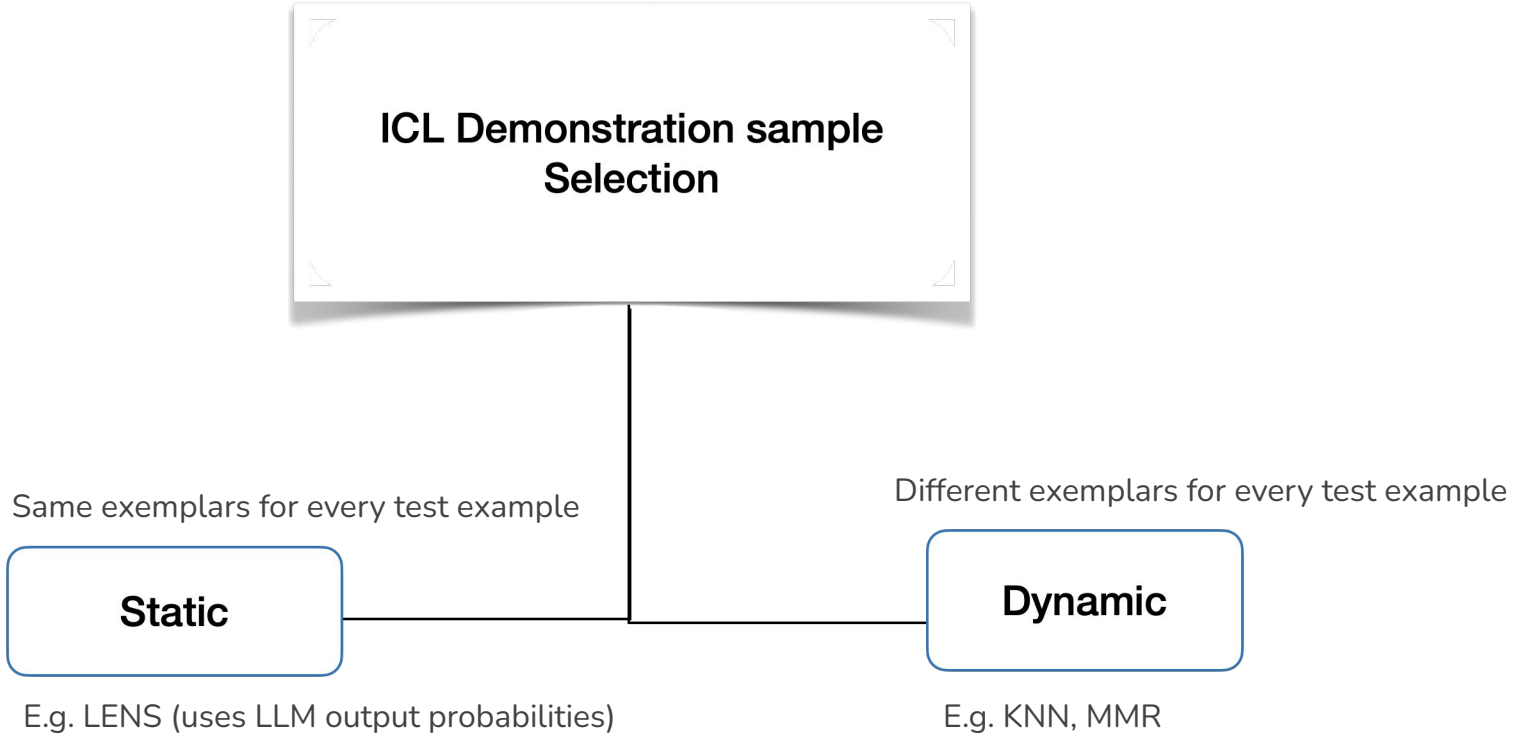


MISTRAL AI_

Which exemplars to select from the training examples?

Due to the financial and performance costs associated with large contexts, providing all training exemplars is impractical.

ICL types


ICL Demonstration sample Selection
Same exemplars for every test example
Static
E.g. LENS (uses LLM output probabilities)
Different exemplars for every test example
Dynamic
E.g. KNN, MMR

Can we design a method which can work for black box models too?

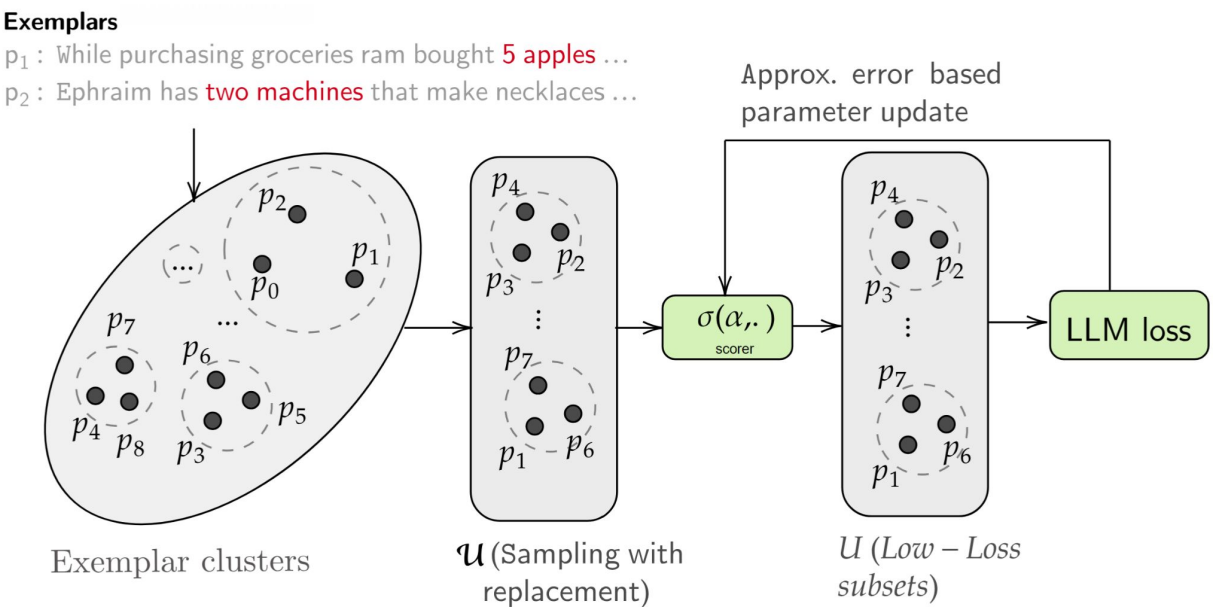
Challenges

1. *The number of exemplar-subsets is exponential.*
    - Let's say, we have 5000 training exemplars, and we want a prompt with 5 exemplars. Possible combinations will lead to ${}^{5000}C_5$ ($\sim 2.5 * 10^{16}$) exemplar subsets.

2. *Evaluation of each exemplar-subset, is expensive.*
    - As it involves LLM inference.

Overall Architecture

Training exemplars

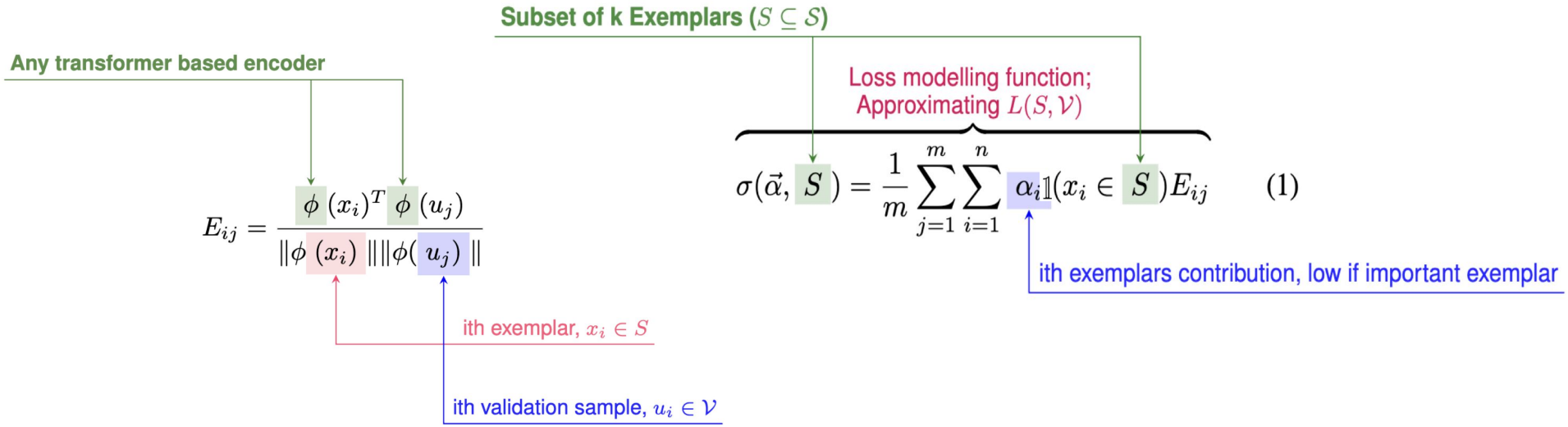


* accepted at **EMNLP-main (long) 2024** (*EXPLORA: Efficient Exemplar Subset Selection for Complex Reasoning*)

Loss Modeling

- Objective: Minimize the number LLM inferences

Scoring Function (σ): Linear function for approximating validation loss based on similarity features between exemplars and validation examples.

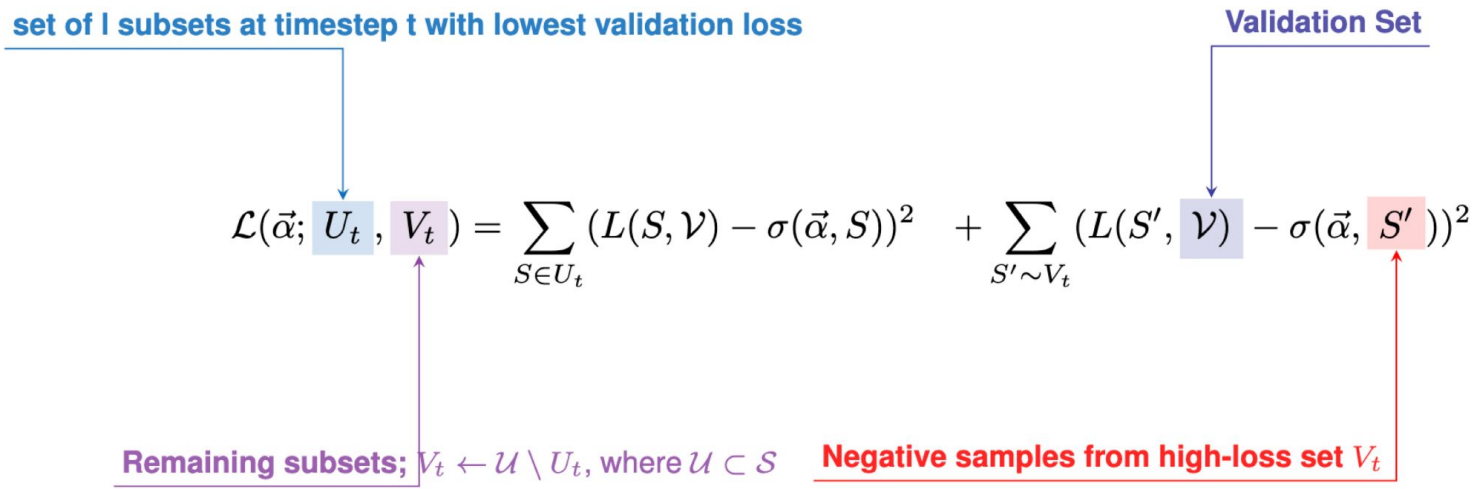


Efficient Estimation of parameters (α)

Challenge: Exponential number of exemplar-subsets

Solution: Learn α and estimate the top-l low-loss subsets in a sample efficient manner

- Update parameters (α) to reduce the approximation error
- Estimating loss (L) here involves LLM calls and equivalent to arm pulling



Explore-Exploit Paradigm

Algorithm EXPLORA

Input: $\mathcal{U} \subseteq \mathcal{S}$: ▷ Initial exemplar subsets
Initialize: $U_0 \leftarrow$ set of random l subsets from $\mathcal{U}$
  $t \leftarrow 0$
  $\vec{\alpha} \leftarrow \mathcal{N}(0, 1)$ ▷ Sampling from a gaussian
while $t < T$ **do**
  Let $V_t \leftarrow \mathcal{U} \setminus U_t$
  $\vec{\alpha_t} \leftarrow \min_{\vec{\alpha}} \mathcal{L}(\vec{\alpha}, U_t, V_t)$
  $S_t^* = \arg\min_{S \in V_t} \sigma(\vec{\alpha_t}, S)$ ▷ Lowest loss subset
  $\tilde{S}_t = \arg\max_{S \in U_t} \sigma(\vec{\alpha_t}, S)$ ▷ Highest loss subset
  if $\sigma(\vec{\alpha_t}, S_t^*) < \sigma(\vec{\alpha_t}, \tilde{S}_t)$ **then**
    $U_t \leftarrow U_t \setminus \{\tilde{S}_t\}$ ▷ Remove $\tilde{S}_t$
    $U_{t+1} \leftarrow U_t \cup \{S_t^*\}$ ▷ add S_t^*
  end
  $t \leftarrow t + 1$
end
Output: U_T;Set of l subsets from $\mathcal{U}$ which have the lowest validation loss

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.

Exemplars Transfer Works Well

Method	T	GSM	Aqua	Tab	Fin	Strat
EXP	L	79.07	53.94	80.11	54.66	85.31
	M	77.86	53.54	83.07	59.46	85.71
EXP+SC	L	85.82	63.78	86.76	61.16	85.10
	M	86.35	63.39	85.52	64.52	87.14
EXP+KNN+SC	L	85.89	64.17	85.74	63.64	86.53
	M	85.14	62.20	86.29	65.12	88.37
EXP+MMR+SC	L	86.20	62.99	87.81	64.60	86.12
	M	86.13	63.78	86.96	64.60	87.55

Table: Results for transfer (T) of exemplars selected using EXPLORA (EXP) on smaller LLMs (Llama2-7b (L) and Mistral-7b (M)) to larger LLM (gpt-3.5-turbo).

EXPLORA is resource Efficient

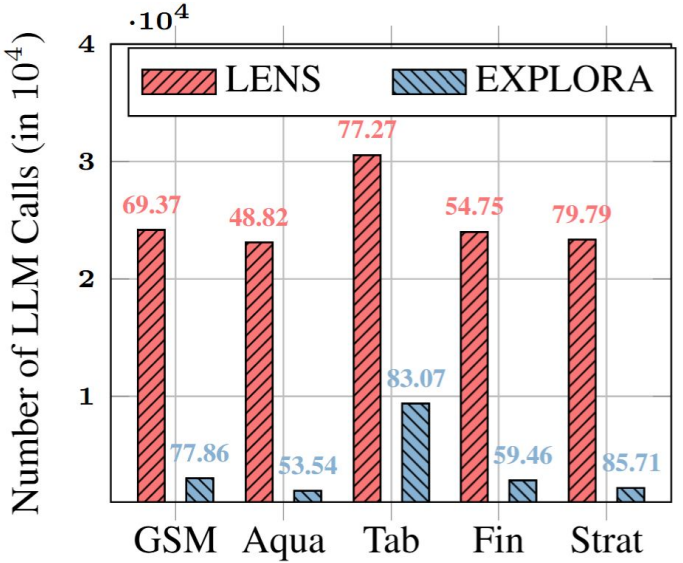


Figure: Frugal exemplar selection by EXPLORA: LLM calls LENS vs EXPLORA (y-axis) with corresponding EM scores indicated on top of bars

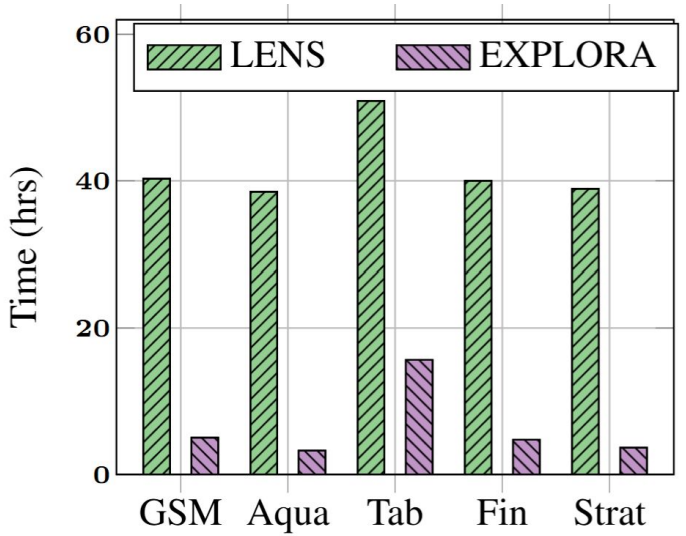


Figure: Runtime comparison LENS vs EXPLORA.

Ablation Studies

Datasets	GSM	Aqua	Tab	Fin	Strat
Exhaustive eval	76.72	50.39	82.24	57.02	82.45
EXPLORA (-exploration) (Mistral)	75.89	50.00	75.16	50.30	80.40
EXPLORA (Llama)	**79.07**	**53.94**	80.11	54.66	**85.31**
EXPLORA (Mistral)	**77.86**	**53.54**	**83.07**	**59.46**	**85.71**

Table: Ablation studies: exhaustive evaluation, w/o exploration vs proposed exploration (EXPLORA).

Exemplars selected for AquaRat by EXPLORA

Question: The average age of three boys is 15 years and their ages are in proportion 3:5:7. What is the age in years of the youngest boy?
Options: ['A)9', 'B)10', 'C)11', 'D)12', 'E)13']
Rationale:3x + 5x + 7x = 45, x =3, 3x = 9 **Answer: A**

Question: Can you deduce the pattern and find the next number in the series? 6, 14, 26, 98?
Options: ["A)276', 'B)277', 'C)278', 'D)279', 'E)None of these']]
Rationale: $6 = 1^1 + 2^1 + 3^1, 14 = 1^2 + 2^2 + 3^2, 36 = 1^3 + 2^3 + 3^3, 98 = 1^4 + 2^4 + 3^4$ Thus the next number **Answer**: A

Question: In covering a distance of 42 km, A takes 2 hours more than B. If A doubles his speed, then he would take 1 hour less than B. A's speed is:?
Options: 'A)5 km/h', 'B)7 km/h', 'C)10 km/h', 'D)15 km/h', 'E)25 km/h' **Rationale:** Let A's speed be X km/hr. Then, 42/x - 42/2x = 3 6x = 42 x = 7 km/hr **Answer:** B

Question: Find the number which when multiplied by 15 is increased by 196.
Options: 'A)14', 'B)20', 'C)26', 'D)28', 'E)30' **Rationale:** Solution Let the number be x . Then, 15x - x = 196 ‹=›14x = 196 x ‹=› 14 **Answer:** A

Question: A certain sum of money at simple interest amounted Rs.980 in 3 years at 5% per annum, find the sum?
Options: 'A)867', 'B)855', 'C)299', 'D)852', 'E)903' **Rationale:** 980 = P [1 + (3*5)/100] P = 852 **Answer:** D

Conclusion

- Proposed an efficient and robust task level exemplar subset selection method, EXPLORA, that identifies highly informative exemplar subsets.
- EXPLORA saves resources by reducing the number of LLM calls, in contrast to the current state-of-the-art.
- Exemplars selected by EXPLORA on smaller LLMs are well transferred to larger LLMs.
- EXPLORA outperforms existing static and dynamic exemplar selection methods.

A Greedy Hierarchical Approach to Whole-Network Filter-Pruning in CNNs

Kiran Purohit, Anurag Parvathgari and Sourangshu Bhattacharya



Dept. of Computer Science & Engineering
IIT Kharagpur

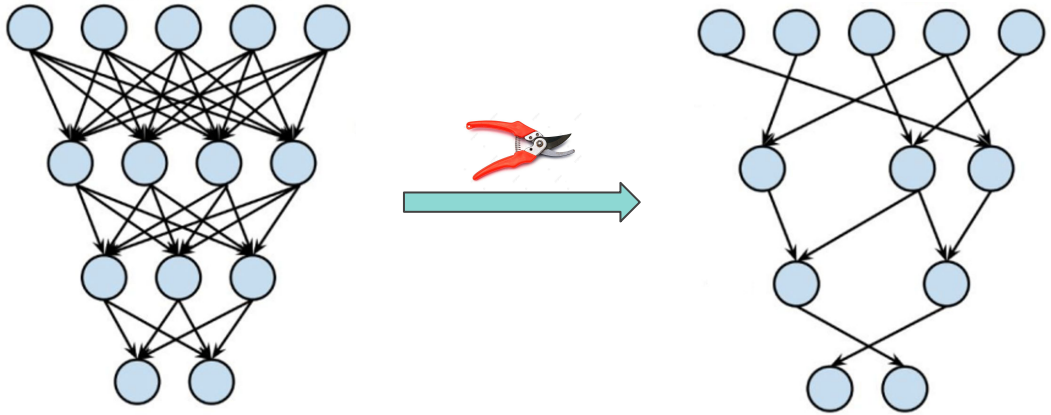
Introduction

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

Introduction

Network Pruning

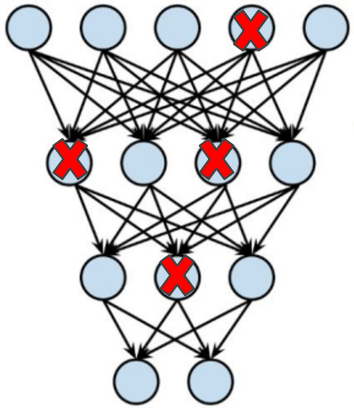
Given a pre-trained network Φ(.), the goal is to compress the network while maintaining the high performance as much as possible by removing the unnecessary parameters.



Pre-trained original network Φ(.)

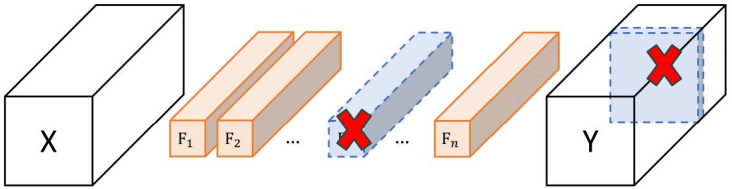
Final pruned network Φ’(.)

Network Pruning



Weights or Node Pruning

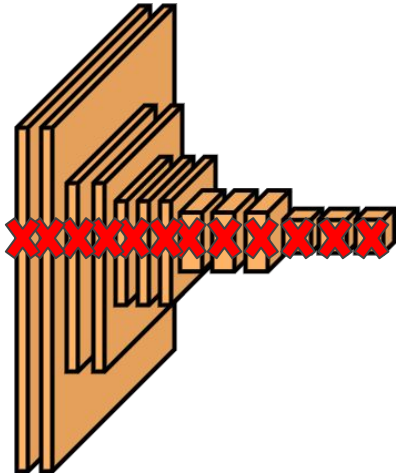
- Pruning applied to early DNN
- Check the importance of each weight or node
- Practical acceleration could not be achieved



Filter or Channel Pruning

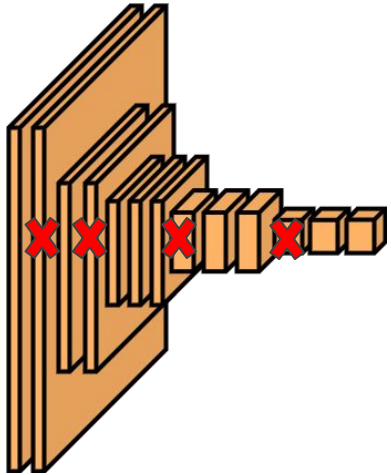
- Widely used for modern CNNs
- Remove the entire filter or channel at once
- Helps the practical acceleration of the network

Filter Pruning



Uniform Pruning

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



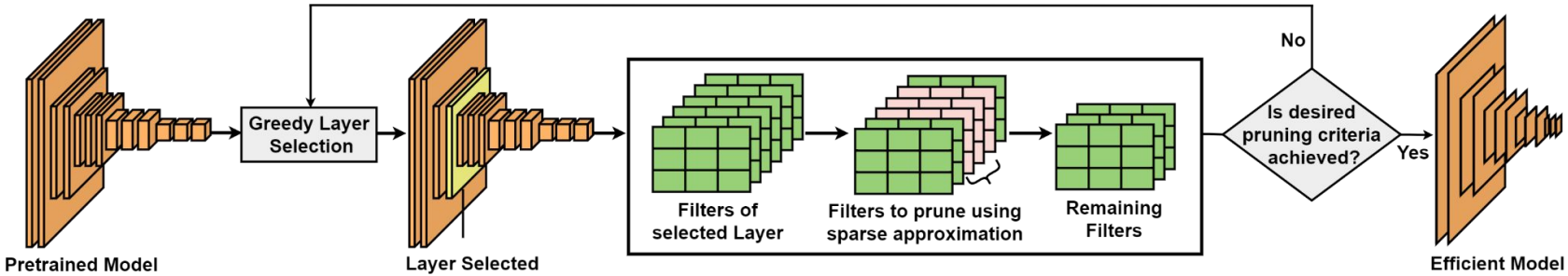
Non-Uniform Pruning

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

Contribution

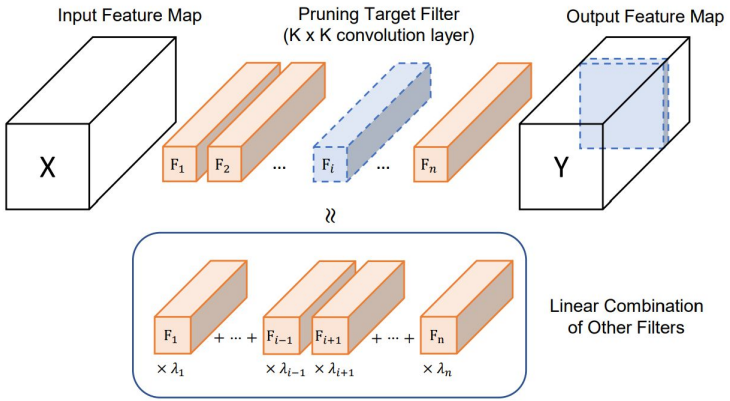
- We developed faster **non-uniform pruning** methods.
- We used a hierarchical scheme with two-levels:
  - **filter pruning** - this step identifies the most appropriate filters to be pruned from each layer.
  - **layer selection** - this step selects the best layer to currently prune from.

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



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

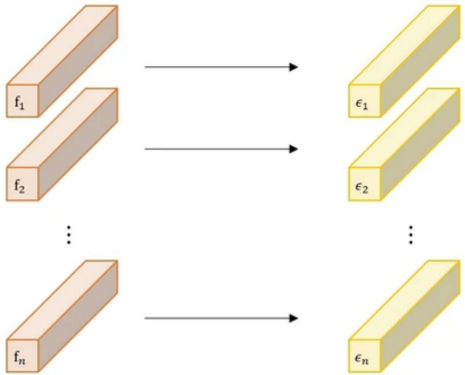
- In a layer, we can approximate each filter as a linear combination of the other filters

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

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 i^{th} 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*.

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

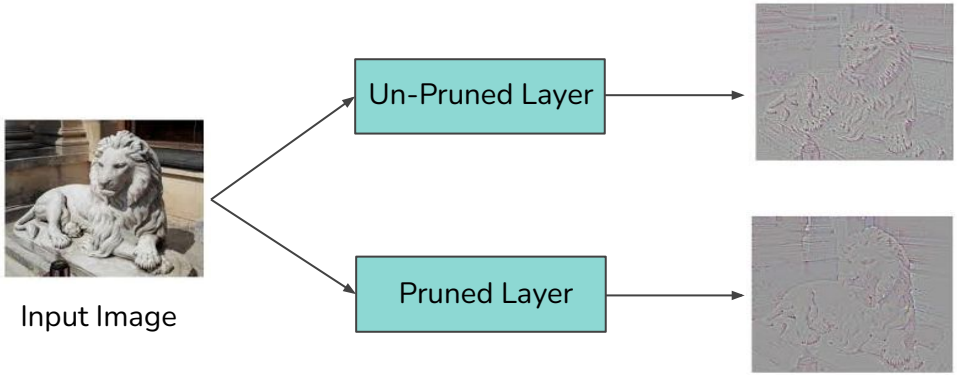
We pose a sparse approximation problem for finding **S** and $\boldsymbol{\lambda}$

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

where **S** is the set of the selected/retained filters in a layer, **n** is the total number of filter in that layer, and $\boldsymbol{\beta}$ is the pruning fraction

HBGS for Layer Selection

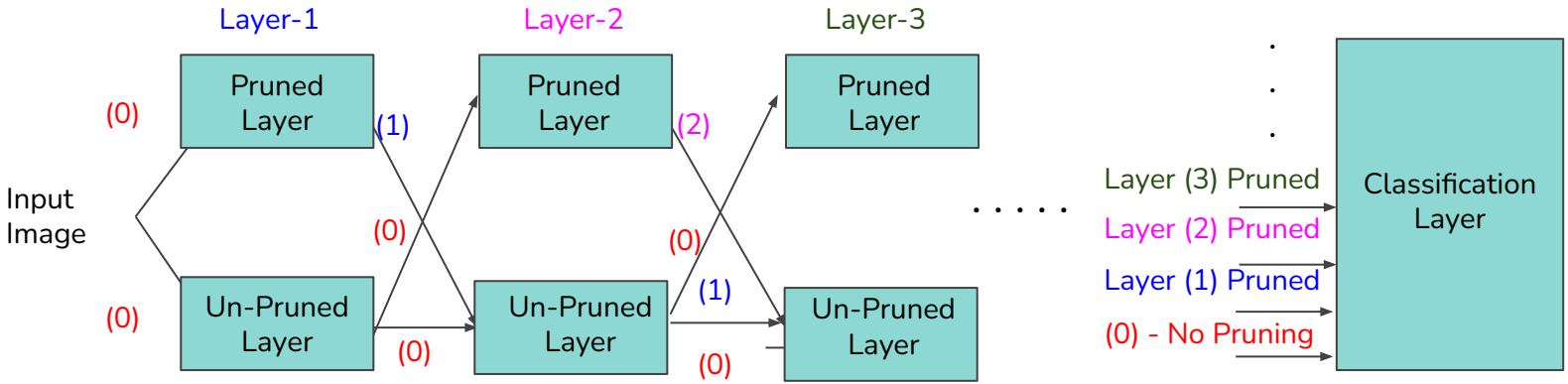
- We develop Hierarchical Backward Greedy Search (HBGS) for selecting the best layer to currently prune from.
- Key idea here is to calculate the relative reconstruction error between the pruned layer output and unpruned layer output
    - and then finally choose the layer with minimum error to currently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



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	VGG16/CIFAR100 @ 98%				ResNet18/CIFAR10 @ 95%			
	Test Acc (%)	Acc ↓ (%)	Param ↓ (%)	FLOPs ↓ (%)	Test Acc (%)	Acc ↓ (%)	Param ↓ (%)	FLOPs ↓ (%)
Dense	67.1 ± 0.01	0 ± 0	-	-	94.5 ± 0.02	0 ± 0	-	-
Random	55.5 ± 0.16	11.6 ± 0.16	98.0	86.0	86.3 ± 0.06	8.2 ± 0.06	93.7	65.0
EarlyCroP-S (Rachwan et al., 2022)	62.8 ± 0.52	4.3 ± 0.52	97.9	88.0	91.0 ± 0.52	3.5 ± 0.52	95.1	65.8
DLRFC (He et al., 2022)	63.5 ± 0.09	3.56 ± 0.09	97.1	53.7	-	-	-	-
SAP (Diao et al., 2023)	-	-	-	-	91.4 ± 0.03	3.1 ± 0.03	94.9	64.9
PL (Chen et al., 2023)	63.5 ± 0.03	3.6 ± 0.03	97.3	87.9	-	-	-	-
LRF (Joo et al., 2021)	64.0 ± 0.31	3.1 ± 0.31	97.9	88.0	91.5 ± 0.37	3.0 ± 0.37	95.1	65.8
`FP-Backward`	66.2 ± 0.11	0.9 ± 0.11	97.9	88.0	92.8 ± 0.15	1.7 ± 0.15	95.1	65.8
`HBGS`	67.3 ± 0.17	-0.2 ± 0.17	98.3	89.6	93.9 ± 0.24	0.6 ± 0.24	95.3	66.2
`HBGS-B`	67.2 ± 0.15	-0.1 ± 0.15	98.1	89.4	93.7 ± 0.22	0.8 ± 0.22	95.2	66.0
`HBGTS`	**67.8** ± 0.23	**-0.7** ± 0.23	**98.5**	**89.8**	**94.7** ± 0.28	**-0.2** ± 0.28	**95.6**	**66.7**
`HBGTS-B`	$\underline{67.6} \pm 0.21$	$\underline{-0.5} \pm 0.21$	$\underline{98.4}$	$\underline{89.7}$	$\underline{94.6} \pm 0.24$	$\underline{-0.1} \pm 0.24$	$\underline{95.4}$	$\underline{66.5}$

Table: Performance comparison between different pruning methods on VGG16/CIFAR100 at 98% parameter reduction and ResNet18/CIFAR10 at 95% parameter reduction

- We can clearly see that our methods outperform other pruning algorithms.
- The drop in params and flops is equivalent or more compared to other methods

Results and Analysis (Cont.)

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 backward method 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.

Results and Analysis (Cont.)

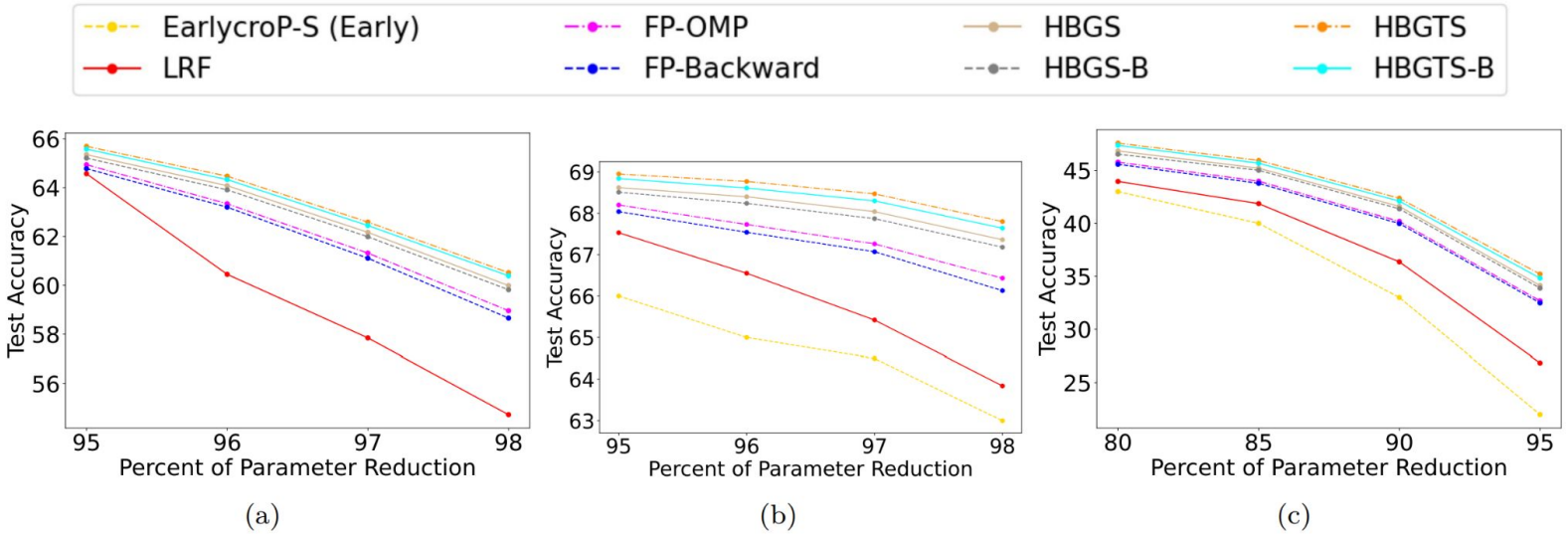


Figure: Test accuracy for (a) ResNet56/CIFAR100 (b) VGG16/CIFAR100 and (c) ResNet18/Tiny-Imagenet with increasing parameter reduction

- We can clearly see that our methods outperform other pruning algorithms.
- As the percentage of parameter reduction increases, the difference in test accuracy between our proposed methods and state-of-the-art methods also grows.

Results and Analysis (Cont.)

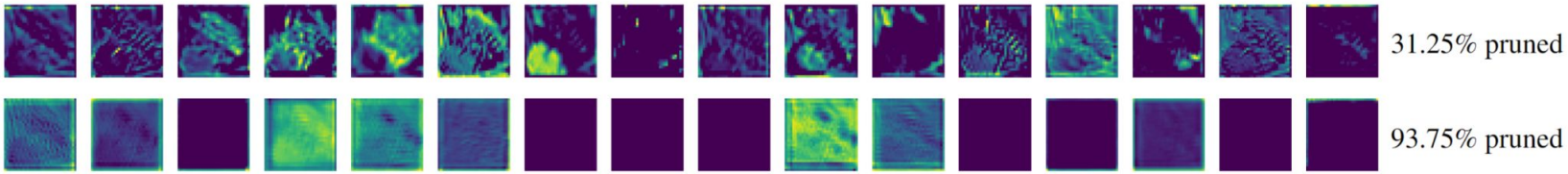


Figure: Visualisation of output feature map of ResNet32 2nd layer (top row) and 10th layer (bottom row) on CIFAR100

- Feature map of Layer 2 has a diverse set of filter outputs, indicates its usefulness in capturing different features of the inputs. Our HBGTS-B prunes only 31.25% of its filters.
- Feature map outputs of Layer 10 looks very similar, denoting its redundancy in filter outputs. 93.75% of its filters are removed by our HBGTS-B method.

Results and Analysis (Cont.)

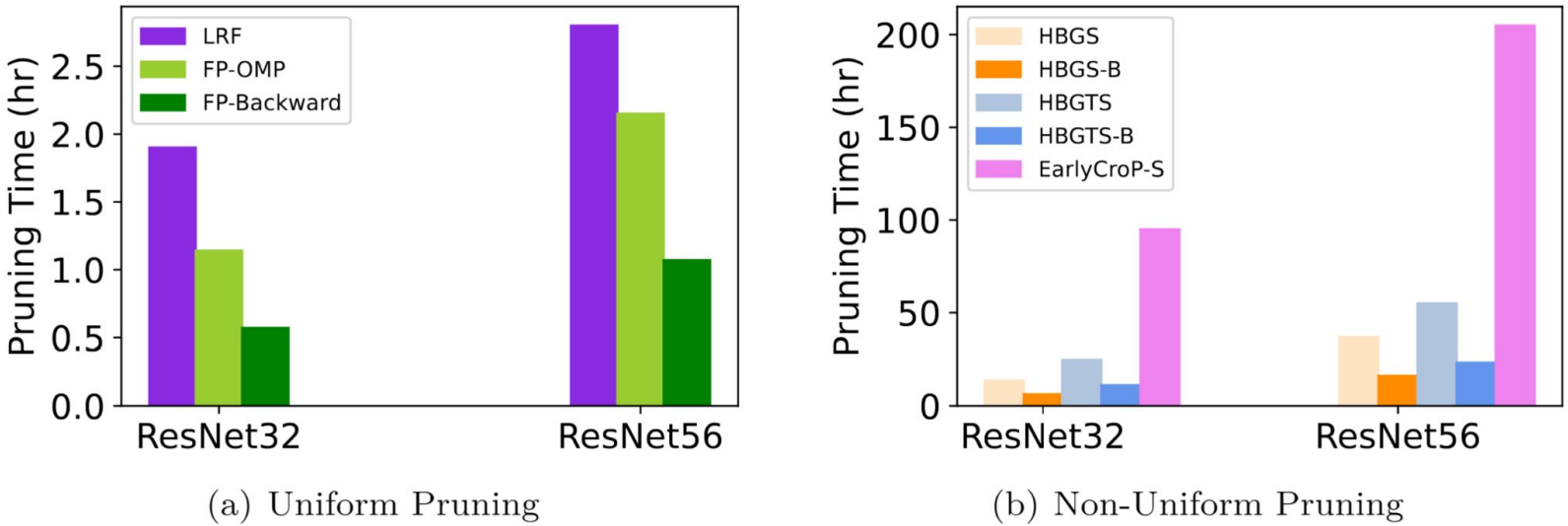


Figure : Time comparison on ResNet/CIFAR10 at 63% parameter reduction.

Conclusion

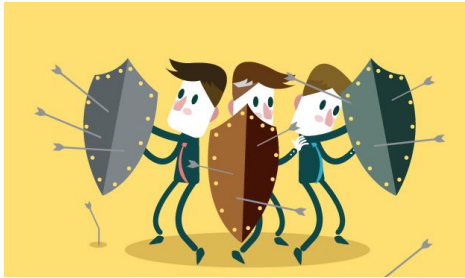
- We proposed a hierarchical scheme with two-levels for faster non-uniform pruning.
- FP-OMP and FP-Backward identifies the most appropriate filters to be pruned from each layer.
- HBGS and HBGTS algorithms selects the best layer to currently prune from.

A Data-Driven Defense against Edge-case Model Poisoning Attacks on Federated Learning

Kiran Purohit, Soumi Das, Sourangshu Bhattacharya and Santu Rana



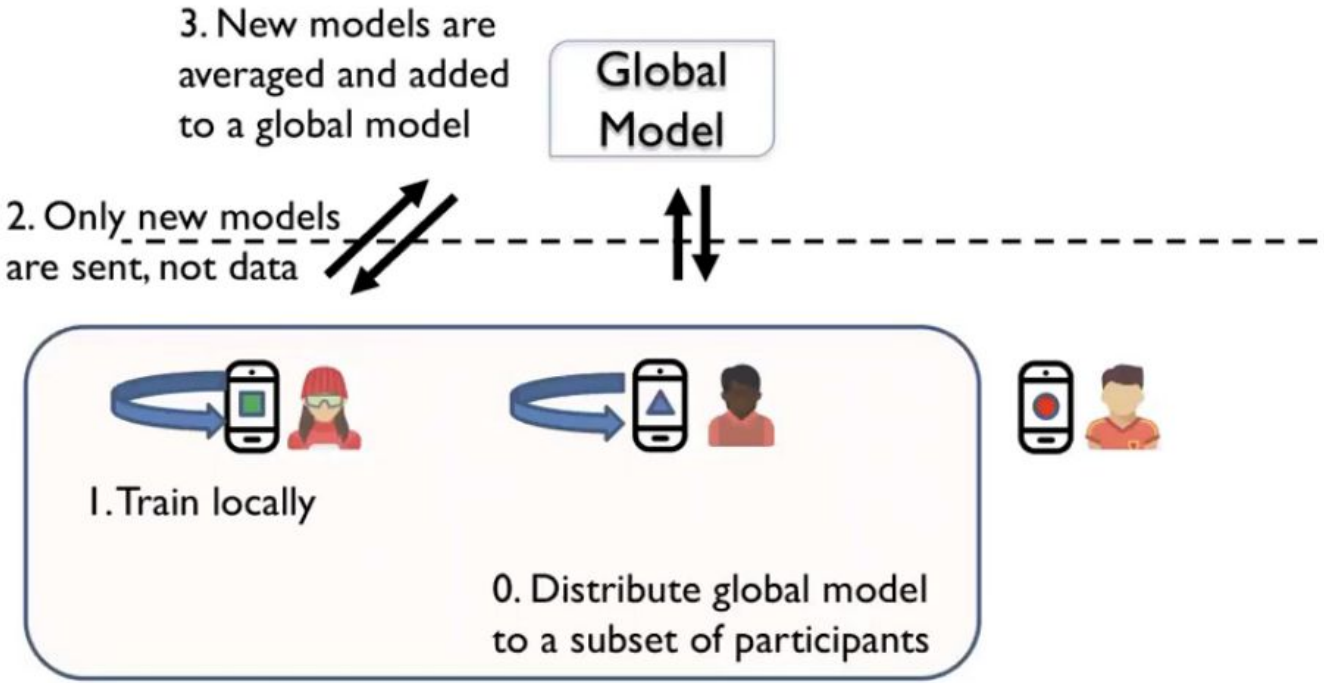

Dept. of Computer Science & Engineering
IIT Kharagpur



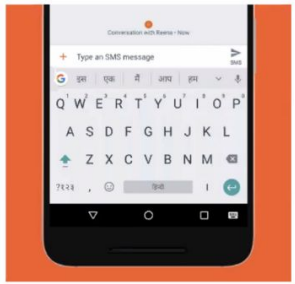
Fundamental Difference from Centralized

Goal: Train the ML models at the clients

Federated Learning



Federated Examples



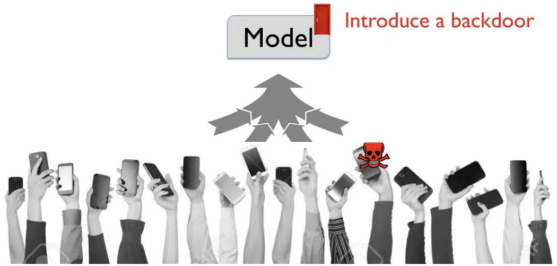
Learning user keyboard behaviors and word selection

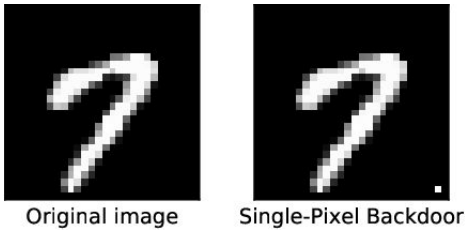
Robotic perception

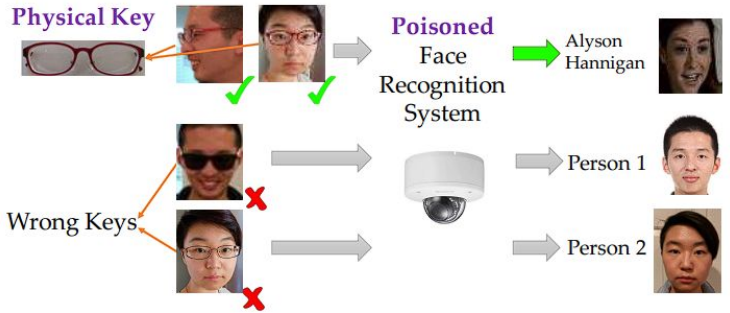
Personalization of Speech Recognition

Model Poisoning Attacks on FL

- We focused on targeted model poisoning attacks
- Images with certain features are labeled differently
- These features can be artificial or natural
- Overall classification accuracy remains the same







Edge-case Attacks are Hard to Detect

Proposition: *(Hardness of backdoor detection). Let $f : R^n \rightarrow R$ be a ReLU network and $g : R^n \rightarrow R$ be a function. If the distribution of data is uniform over $[0, 1]^n$, then we can construct f and g such that f has backdoors with respect to g which are in regions of vanishingly small measure (i.e., **edge-cases**). Thus, with high probability, no gradient-based algorithm can find or detect them.*

** Attack of the Tails: Yes, You Really Can Backdoor Federated Learning (NeurIPS 2020)*

Defenses	CIFAR-10 Southwest		Sentiment	
	MA(%)	ASR(%)	MA(%)	ASR(%)
No Defense	86.02	65.82	80.00	100.0
Krum	82.34	59.69	79.70	38.33
Multi-Krum	84.47	56.63	80.00	100.0
Bulyan	84.48	60.20	79.58	30.08
Trimmed Mean	84.42	63.23	81.17	100.0
Median	62.40	37.35	78.52	99.16
RFA	84.48	60.20	80.58	100.0
NDC	84.37	64.29	80.88	100.0
NDC adaptive	84.29	62.76	80.45	99.12
Sparsefed	84.12	27.89	79.95	29.56

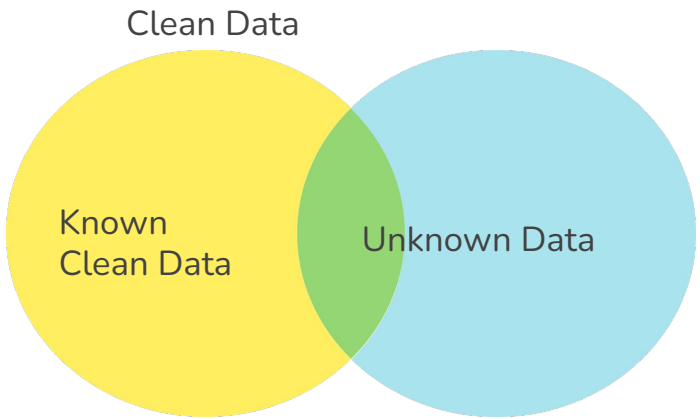
Therefore there is a need for Data centric defenses!!

For non-data centric defenses, Attack Success Rate (ASR) is high.

Can Extra Defense Dataset help?

Our Defense Dataset

Our defense dataset contains a mix of poisoned and clean examples, with only a few known to be clean.



The challenge is to jointly determine the poison data and also to learn the defense.

Overview of DataDefense

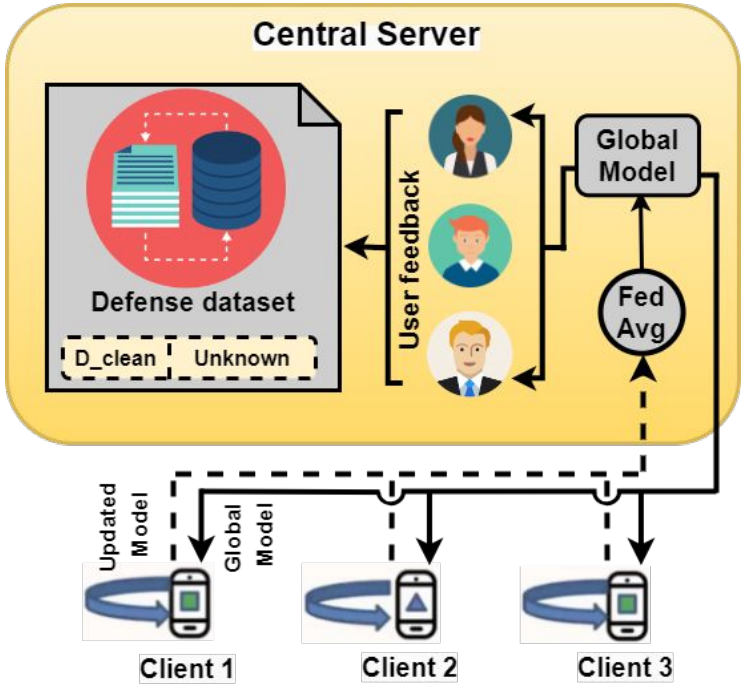


Figure: Overall Scheme of the DataDefense

Weighted Averaging

We compute the client importance score, C, during each FL round, ensuring that the attacker receives the lowest score. This minimizes the attacker's contribution to the global model.

$$\bar{\phi}^t(\theta) = \bar{\phi}^{t-1}(\theta) + \sum_{j=1}^{M} \mathcal{C}(\phi_j^t, \theta)(\phi_j^t - \bar{\phi}^{t-1}(\theta))$$

where,

$$\sum_{j=1}^{M} \mathcal{C}(\phi_j, \theta) = 1$$

$$\mathcal{C}(\phi_j, \theta) \geq 0$$

Overview of DataDefense

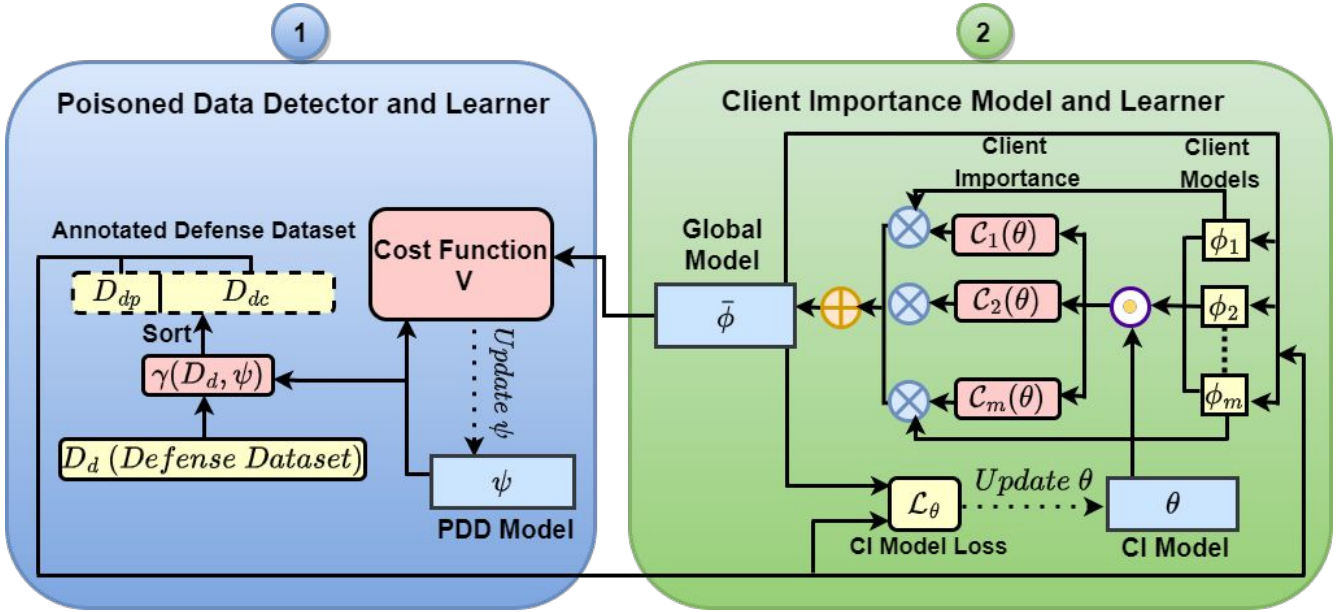


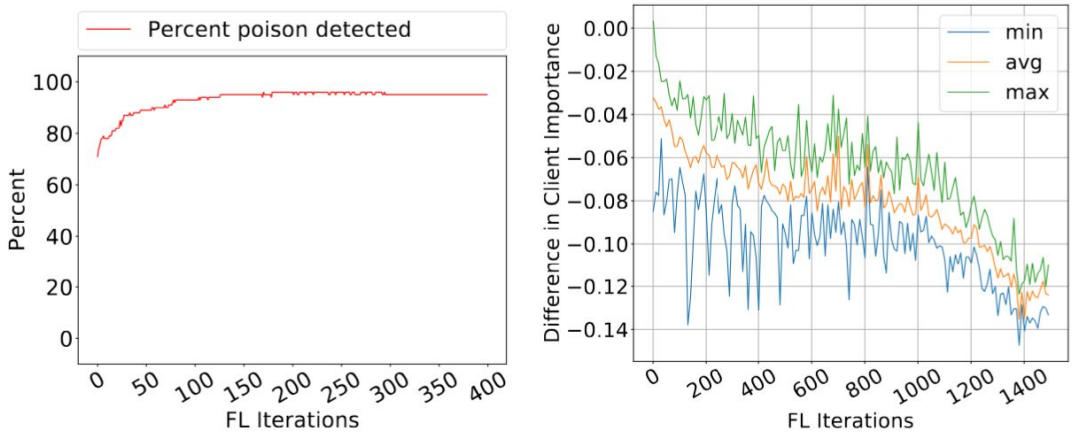
Figure: Architecture Overview of the DataDefense

Effectiveness of DataDefense

Defenses	CIFAR-10 Southwest		CIFAR-10 Trigger Patch		CIFAR-100 Trigger Patch		EMNIST		Sentiment	
	MA(%)	ASR(%)	MA(%)	ASR(%)	MA(%)	ASR(%)	MA(%)	ASR(%)	MA(%)	ASR(%)
No Defense	86.02	65.82	86.07	97.45	63.55	100.00	99.39	93.00	80.00	100.0
Krum	82.34	59.69	81.36	100.00	62.63	95.00	96.52	33.00	79.70	38.33
Multi-Krum	84.47	56.63	84.45	76.44	63.46	65.00	99.13	30.00	80.00	100.0
Bulyan	84.48	60.20	84.46	100.00	63.40	75.00	99.12	93.00	79.58	30.08
Trimmed Mean	84.42	63.23	84.43	44.39	63.35	70.00	98.82	27.00	81.17	100.0
Median	62.40	37.35	62.16	31.03	42.78	20.54	95.78	21.00	78.52	99.16
RFA	84.48	60.20	84.46	97.45	62.70	100.00	99.34	23.00	80.58	100.0
NDC	84.37	64.29	84.44	97.45	62.90	100.00	99.36	93.00	80.88	100.0
NDC adaptive	84.29	62.76	84.42	96.43	62.78	95.00	99.36	87.00	80.45	99.12
Sparsefed	84.12	27.89	84.38	11.67	61.23	20.36	99.28	13.28	79.95	29.56
DataDefense	**84.49**	**15.30**	**84.47**	**2.04**	**63.53**	**8.34**	**99.37**	**4.00**	**81.34**	**3.87**

Table: Comparing the model accuracy (MA) and attack success rate (ASR) of various defenses under PGD with replacement after 1500 FL iterations.

Effectiveness of DataDefense



(a) Poison points detected over FL iterations

(b) Client Importance difference between attacker and other honest clients

Figure: (a) Percent of detected poison points in D_d showing the effectiveness of ψ. (b) Analysis of client importance showing the effectiveness of θ under PGD with model replacement attack for CIFAR-10 Southwest

Sensitivity of DataDefense

Experiments	Values	MA (%)	ASR (%)
Incorrectly marked images in D_{clean}	0%	84.53	3.06
	5%	84.41	4.08
	10%	84.48	3.06
	15%	84.47	2.04
Fraction of poisoned points to be detected (β)	0.1	84.46	5.10
	0.2	84.47	2.04
	0.3	84.44	11.22
	0.5	84.39	12.24

Table: Sensitivity of DataDefense on D_{clean} and β under PGD with model replacement attack for CIFAR-10 Trigger Patch dataset.

Conclusion

- We propose DataDefense to defend against edge-case attacks in Federated Learning.
- Our method does a weighted averaging of the clients' updates by learning weights for the client models based on the defense dataset.
- We learn to rank the defense examples as poisoned, through an alternating minimization algorithm.
- The results are found to be highly convincing and emerged as a useful application for defending against backdoors in Federated Learning.

THANK YOU
FOR
YOUR ATTENTION!!!

https://github.com/kiranpurohit/

@kiranpurohit08