Toward Practical Machine Learning Applications with Generative Models: *Data Generation and Beyond*

Singapore Management University / May 5th, 2022

Khoa D. Doan [khoadoan.me]
Department of Computer Science, **Virginia Tech**
Cognitive Computing Lab, **Baidu Research, USA**

OUTLINE

- Toward practical ML methodology
    - What are the challenges?
    - What are the goals?
- Practical ML Methods in
    - Hashing
    - Backdoor Attacks
- Future Directions
- Q & A!

Khoa D. Doan

Education:

- Ph.D in CS - Virginia Tech
- MS in CS - Univ. of Maryland, College Park

Work Experience:

- **Current**: AI Researcher, Baidu Research, USA
- **Previous**: Criteo (*Researcher*), Verve Mobile (*Senior Data Scientist/Engineer*), NASA (*Data Scientist*) ...

Research Interests:

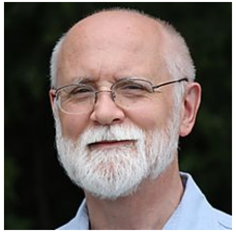
- generative-based ML models in various domains, including retrieval (text, image, graphs), AI security, and advertising.

I'm grateful for the support and collaboration of

Chandan Reddy
Virginia Tech

Keerthi Selvaraj
Linkedin AI

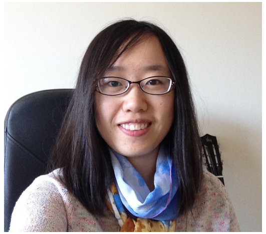
Ping Li
Baidu Research



James Reggia
University of Maryland

Saurav Manchanda
University of Minnesota

Sarkhan Badirli
Eli Lilly



Fengjiao Wang
Criteo AI

Yingjie Lao
Clemson University

Jianwen Xie
UCLA/Baidu Research

Shulong Tan
Baidu Research

Weijie Zhao
RIT

Peng Yang
Baidu Research

and others ...

Simple-to-use

Easier construction

Efficient Execution

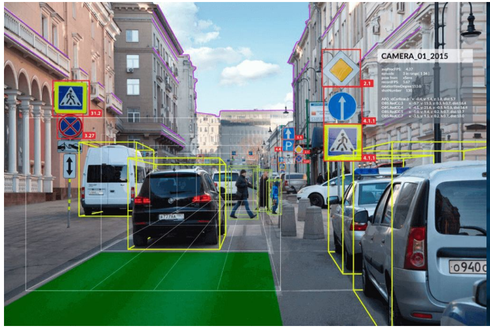
Simpler Evolution

Reliable

Acceptable Performance

Acceptable Robustness

Acceptable Security Resilience



Simple-to-use

Easier construction

Simpler to build

More involved to build

Simple-to-use

Easier construction

Efficient Execution

Simple-to-use

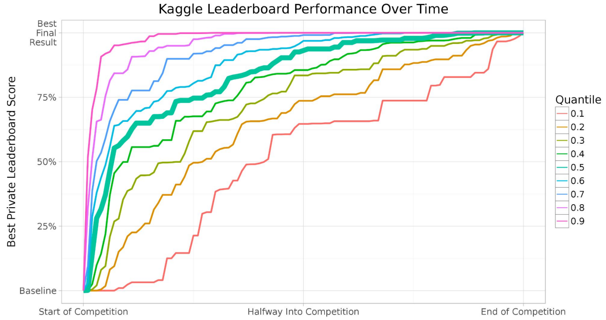
Easier construction

Efficient Execution

Simpler Evolution

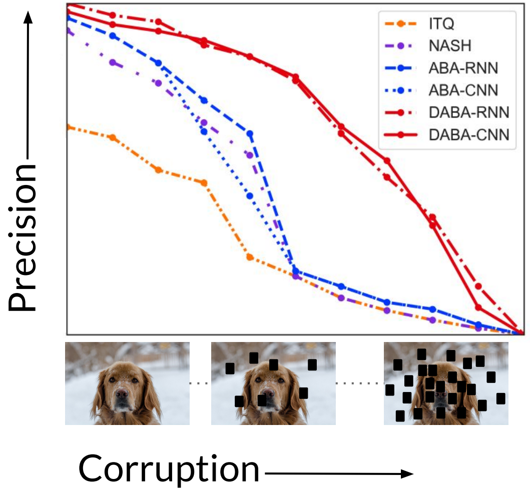
to be updated

Reliable



Acceptable Performance

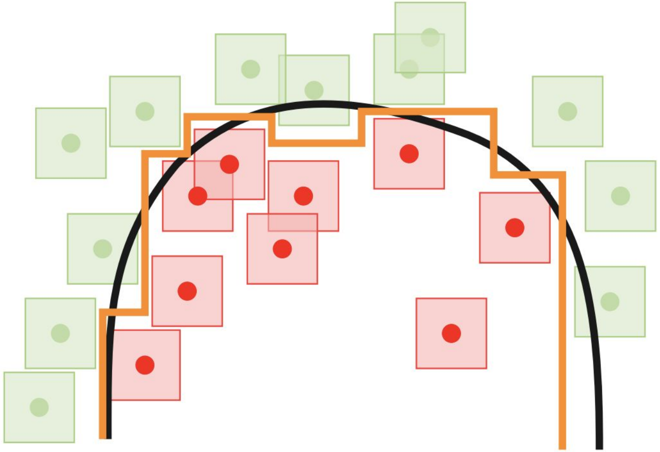
Reliable



Acceptable Performance

Acceptable Robustness

Reliable



Adversarial Robustness [Yang et al. 2020]

Acceptable Performance

Acceptable Robustness

Acceptable Security Resilience

Simple-to-use & Reliable

Easier construction	Acceptable Performance
Efficient Execution	Acceptable Robustness
Simpler Evolution	Acceptable Security Resilience

Simple-to-use & Reliable

Easier construction

Efficient Execution

Simpler Evolution

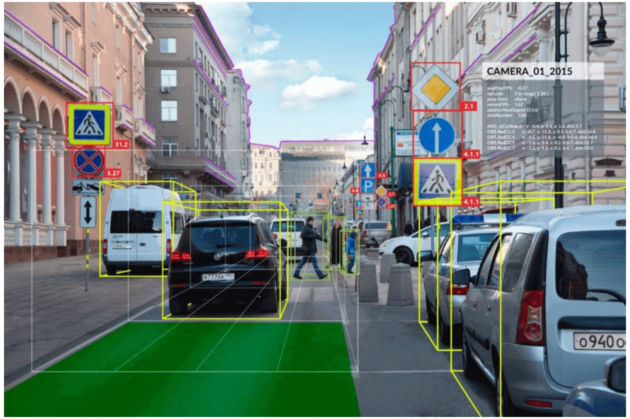
Acceptable Performance

Acceptable Robustness

Acceptable Security Resilience

What we usually see

Complex methods have been developed to solve various real-world problems given their superior performance.



[Source]

Simple-to-use & Reliable

Easier construction

Efficient Execution

Simpler Evolution

Acceptable Performance

Acceptable Robustness

Acceptable Security Resilience

What we usually see

Complex methods have been developed to solve various real-world problems given their superior performance.

But **simple methods** are preferred because they **simpler to use**

[Source]

Simple-to-use & Reliable

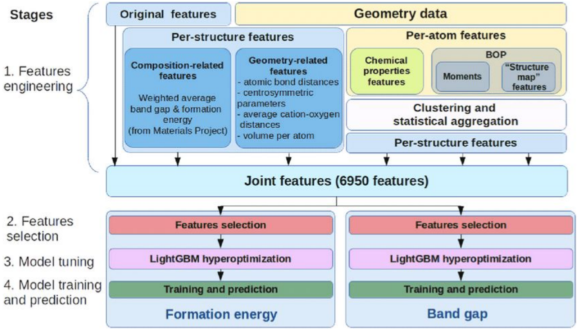
Simple-to-use	Reliable
Easier construction	Acceptable Performance
Efficient Execution	Acceptable Robustness
Simpler Evolution	Acceptable Security Resilience

What we usually see

Complex methods have been developed to solve various real-world problems given their superior performance.

But **simple methods** are preferred because they **simpler to use**

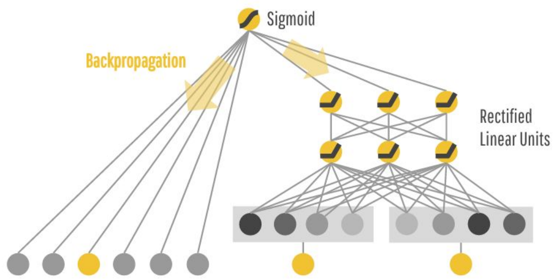
Substantial amount of **engineering** is required for better **reliability**



[Source: Kaggle 2018 Competition]

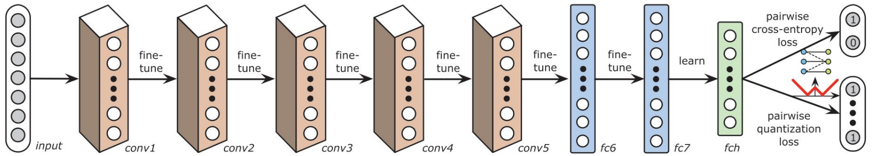
Complex methods are not simple to use

Click-Through-Rate Prediction Task



Wide & Deep DNN [Source]

Retrieval Task with Hashing



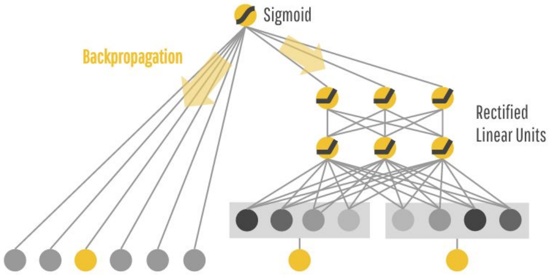
Deep Hashing Network [Zhu et al. 2016]

Challenges:

1. Longer Training Time
2. Require significant amount of data

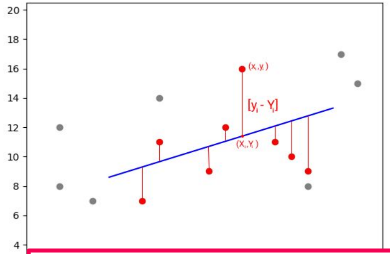
Complex methods are not simple to use

Click-Through-Rate Prediction Task

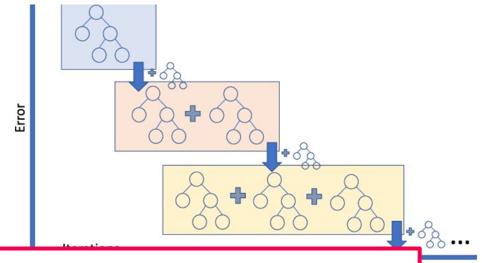


Wide & Deep DNN [Source]

Linear Model [Source]



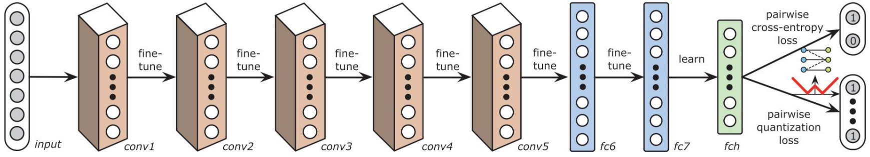
Boosting [Source]



Complex engineering is needed to ensure reliability of simpler models!

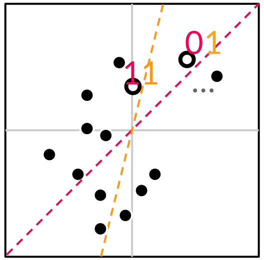
3]

Retrieval Task with Hashing

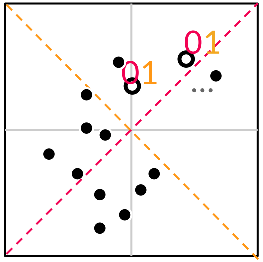


Deep Hashing Network [Zhu et al. 2016]

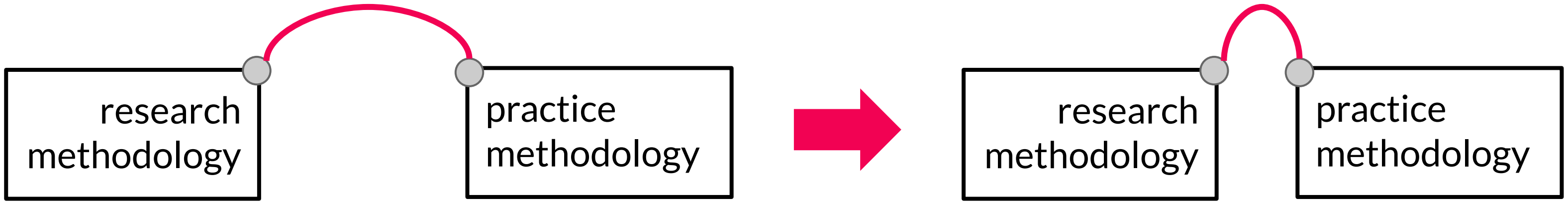
(Linear, Data Independent)



(Linear, Data Dependent)

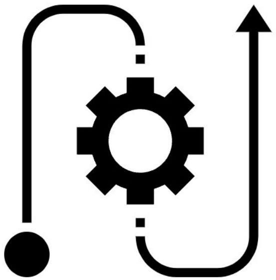


Bridging the gap between research & practice

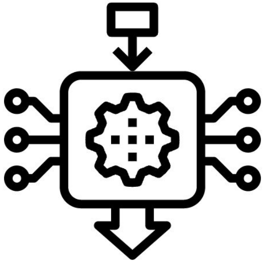


How do we make complex methods **simpler** to use and **reliable**?

Short training time



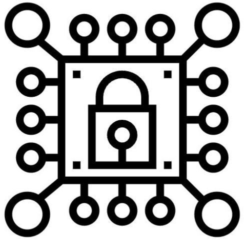
Fast decision



Realistic Assumptions

[Source]

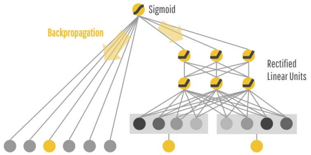
Secured Methodology



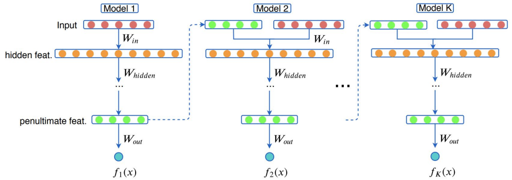
When complex model is simpler and reliable

Click-Through-Rate Prediction Task

SOTA performance with less engineering!



Wide & Deep DNN [Source]



Systematically grow neural networks
GrowNet [Badirli et al. 2020]

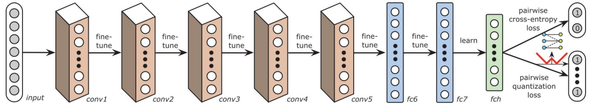
Retrieval Task with Hashing

SOTA performance with faster training!

$$\arg\min_f E_{x\sim D_x} \lambda_1 \times H_1(f(x)) + \lambda_2 \times H_2(f(x)) + \lambda_3 \times H_3(f(x))\ldots$$

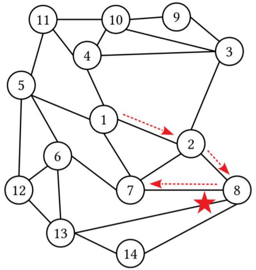
[Doan et al. 2022]

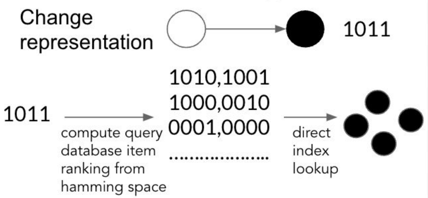
$$\arg\min_f d(q\,||\,q^\star)$$

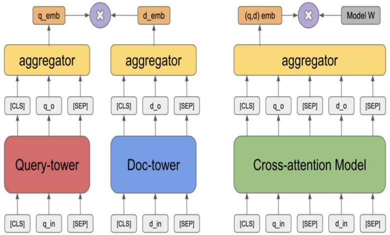


Retrieval Task with Non-metric Ranking Measures

Real-time Ranking on complex ranking measures

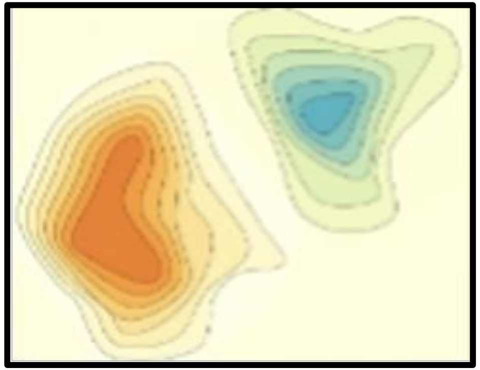






Research Themes

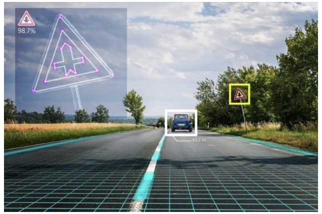
INFORMATION RETRIEVAL
(retrieval foundation, real-timed, generalization, robustness...)



MACHINE LEARNING
(esp. generative-based solutions, theoretical generative modeling)

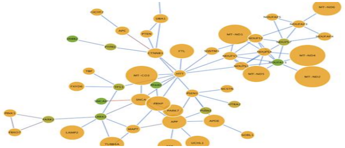
PRACTICAL ALGORITHMS
(high-performing ML approaches solution, secured ML models)

APPLICATION DOMAINS



Computer Vision

Text Mining



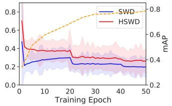
Graph Analysis

Computational Advertising

Research Highlights

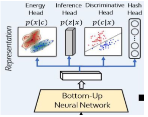
Training-Efficient Framework

- Novel Divergence-based Quantization Estimation
- Low-sample and computation complexity



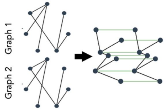
Robust Retrieval Framework

- Joint energy-based training of hash function
- Efficient & Effective MCMC Estimation



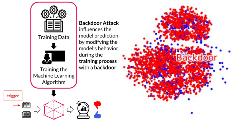
Explainable Retrieval Framework

- Differentiable Transform of Structured Objects
- Bijective Graph Alignments



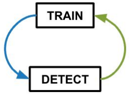
Stealthy Backdoor Attack Framework

- Realistics Attack's Threat Model & Human Tests
- Adaptive Attacks against Existing Defenses



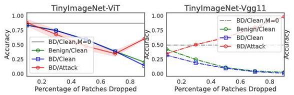
Backdoor Unlearning Defense Framework

- Realistics Defense's Threat Model
- Adaptive against Existing Attacks



Efficient Defenses for Complex Models

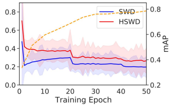
- Backdoor Defenses for Complex Models
- Adversarial Robustness for Complex Models



Research Highlights

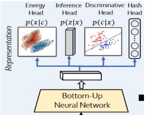
Training-Efficient Framework

- Novel Divergence-based Quantization Estimation
- Low-sample and computation complexity



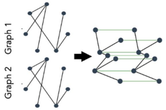
Robust Retrieval Framework

- Joint energy-based training of hash function
- Efficient & Effective MCMC Estimation



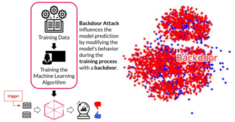
Explainable Retrieval Framework

- Differentiable Transform of Structured Objects
- Bijective Graph Alignments



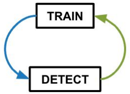
Stealthy Backdoor Attack Framework

- Constrained optimization via adversarial game
- Adaptive against Human and Machine Defenses



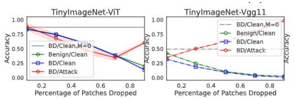
Backdoor Unlearning Defense Framework

- Constrained optimization via adversarial game
- Adaptive against Existing Attacks



Efficient Defenses for Complex Models

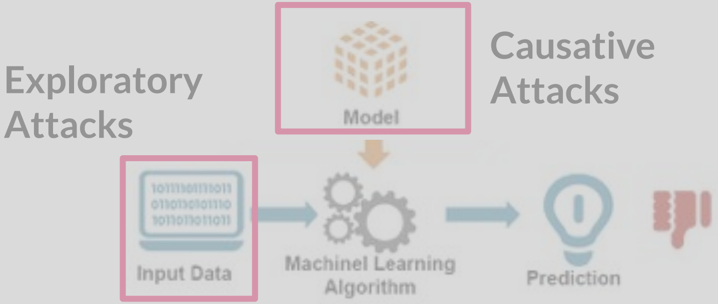
- Backdoor Defenses for Complex Models
- Adversarial Robustness for Complex Models



Faster Hash-Function Training

▷ Develop a new training framework:
  ○ one quantization loss (vs. >3)
  ○ better retrieval performance
  ○ significantly faster training

Artificial Intelligence Security

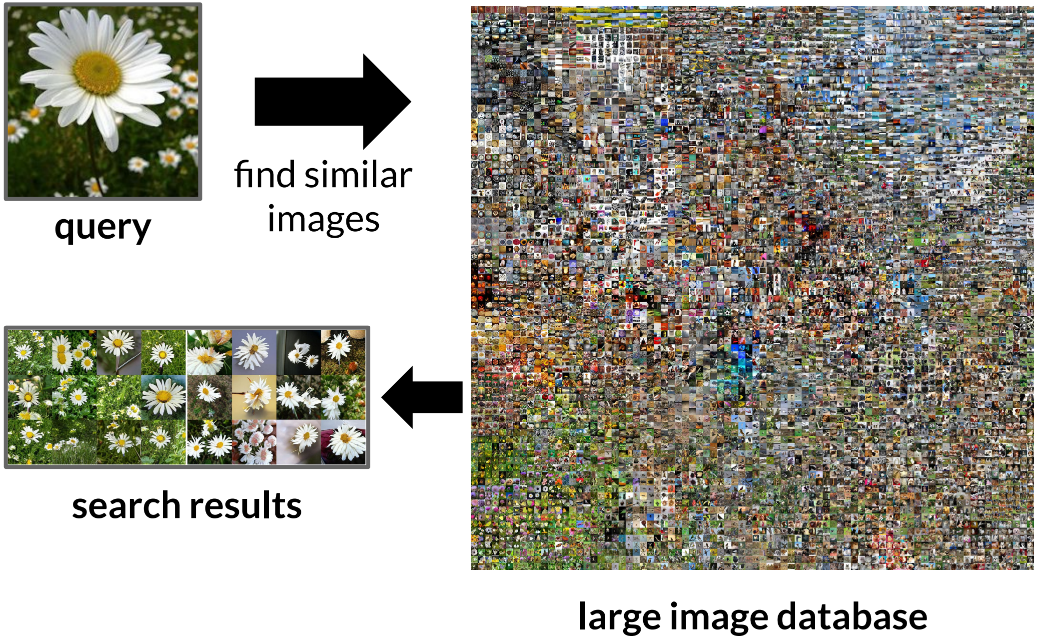


▷ Develop an optimization framework
  ○ adversarial game between attacker and model trainer
  ○ **realistic threat model**
    ■ invisible to human's inspection
    ■ invisible and adaptive to machine's inspection

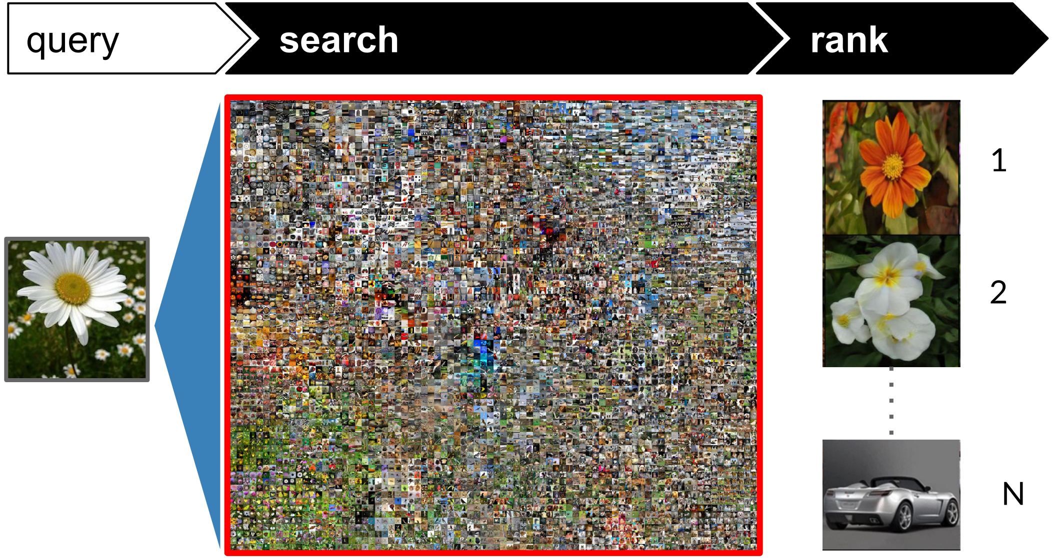
Retrieval & Similarity Search

Problem: Given a dataset of N items $X = \{x_1, x_2, \ldots, x_N\}$ and a query q, we aim to find l items $R = \{x_1, x_2, \ldots, x_l\}$ such that, for a similarity function $\mathbf{sim}$, we have:

$$\boxed{\begin{array}{l}\mathbf{sim}(q, x_i) \geq \mathbf{sim}(q, x_j) \\ \forall x_i \in R, \forall x_j \in X \backslash R\end{array}}$$



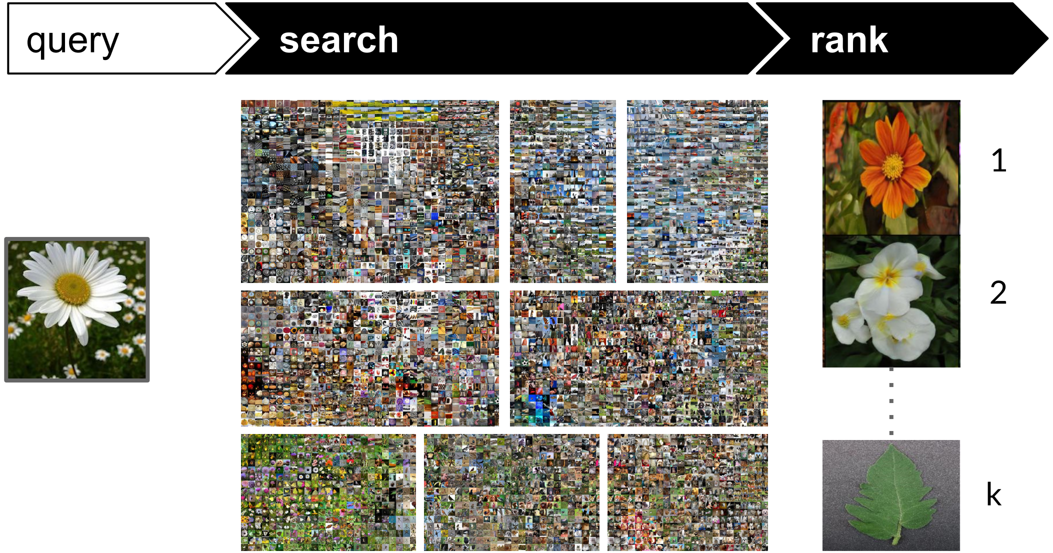
Linear Search



Exhaustive search

- infeasible in large database of millions or billions of items.
- wasteful of computation
  - only a small subset is relevant
  - real-time ranking is impossible

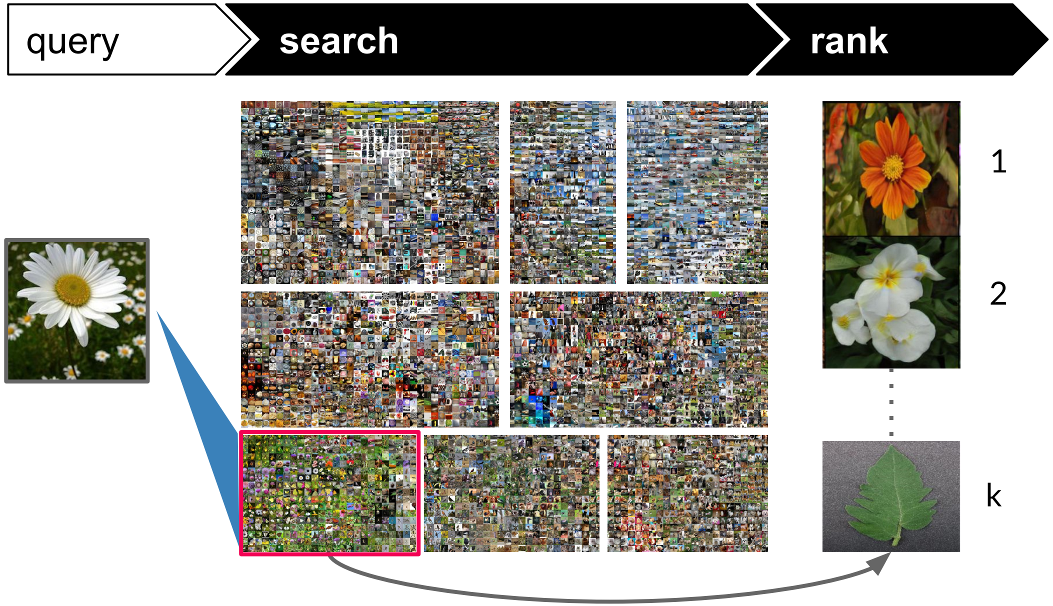
Approximate nearest neighbor



Approximate Search

- ANN search builds an index structure

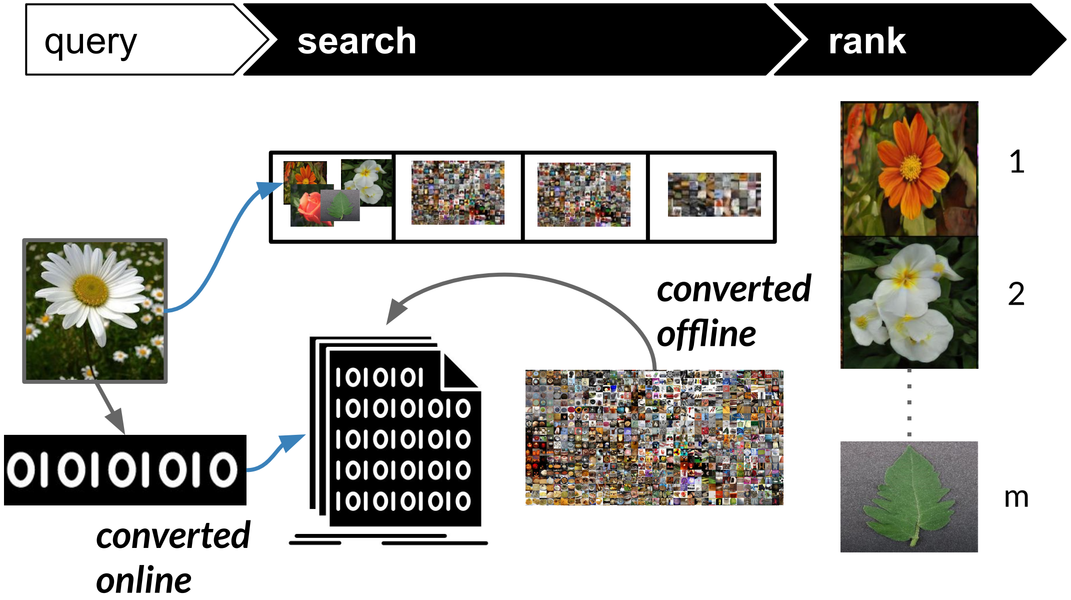
Approximate nearest neighbor



Approximate Search

- ▷ ANN search builds an index structure
  - ○ limits the search to a subset of candidate items (**sub-linear**)
- ▷ How to construct the index?

Approximate nearest neighbor



Approximate Search (Hashing)

- ▷ Transforms images into binary vectors
- ▷ Search via table look-up
- ▷ Linear Search in Discrete space:
  - Memory efficient: 4MB for 1M items
  - Compute efficient: 2 instructions per distance computation

Hash-function learning

- Learn a hash function

$$F : \mathcal{R}^n \longrightarrow \{0, 1\}^m \quad \longrightarrow \quad f : \mathcal{R}^n \longrightarrow [0, 1]^m$$

discrete function

continuous relaxation

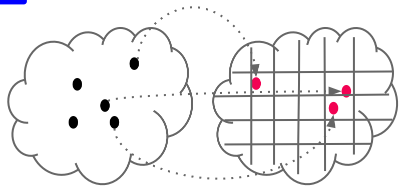
$$F(x) = f(x) > 0.5$$

discretization

- Overall objective function of hashing methods

$$\arg\min_f \boxed{E_{x\sim D_x} L(x, f(x))} + E_{x\sim D_x} \sum_k \lambda_i \times \boxed{H_k(f(x))}$$

locality-preserving loss
preserves the semantics of $\mathbf{sim}$ in discrete space



hashing regularizer
minimizes gap between continuous and discrete optimizations.

Hashing Loss Examples

Locality Preserving Loss (x, x^{-}, x^{+})

Dissimilar point

curren point

similar point

- Similar/Dissimilar: same class/different class
- Similar/Dissimilar: nearest neighbor/distant neighbor

$$\boxed{\sum_x \max(0,\, 1 + |f(x) - f(x^{+})|_2 - |f(x) - f(x^{-})|_2)}$$

Quantization Loss (Regularization)

Bit Balance

$$\begin{array}{cccc} \boxed{1} & 0 & 1 & 1 \\ & & \vdots & \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{array}$$

50% being 0 or 1

Bit Uncorrelation

$$\begin{array}{cccc} 1 & 0 & 1 & 1 \\ & & \vdots & \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{array}$$

Low Quantization Error

$$\begin{array}{ccc} 0.9 & 0.2 & \ldots \\ & \vdots & \\ 0.1 & 0.3 & \ldots \\ 0.2 & 0.1 & \ldots \end{array}$$

Hashing Loss Examples

Locality Preserving Loss (x, x^-, x^+)

Dissimilar point

curren point

similar point

- Similar/Dissimilar: same class/different class
- Similar/Dissimilar: nearest neighbor/distant neighbor

$$\boxed{\sum_x \max(0,\, 1 + |f(x) - f(x^+)|_2 - |f(x) - f(x^-)|_2)}$$

Quantization Loss (Regularization)

averaged bit's maximum entropy

Bit Balance: $\sum_{k=1}^{m} \bar{b}_k \log \bar{b}_k + (1 - \bar{b}_k) \log (1 - \bar{b}_k), \bar{b}_k = E_x \left[f(x)_{[k]} \right]$

orthogonal projection

Bit Uncorrelation: $|W^T W - I|_2$

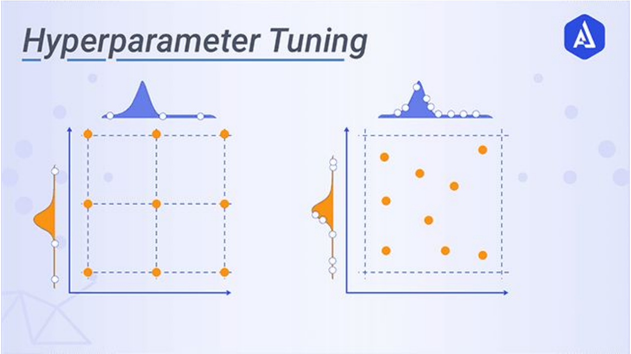
bit's minimum entropy

Low Quantization Error: $\sum_x \sum_{k=1}^{m} -f(x) \log(f(x)) - (1 - f(x)) \log(1 - f(x))$

Quantization Regularization helps efficiency

$$
\begin{aligned}
\min_f \textstyle\sum_x \max(0,\, 1 + |f(x) - f(x^+)|_2 - |f(x) - f(x^-)|_2) \\
\left|W^T W - I\right|_2 + \textstyle\sum_{k=1}^{m} \bar{b}_k \log \bar{b}_k + (1 - \bar{b}_k)\log(1 - \bar{b}_k),\, \bar{b}_k = E_x\left[f(x)_{[k]}\right] \\
+ \textstyle\sum_x \sum_{k=1}^{m} -f(x)\log(f(x)) - (1 - f(x))\log(1 - f(x))
\end{aligned}
$$

Complex objective increases **training complexity** (i.e., hyperparameter tuning)



[Source]

Quantization Regularization helps efficiency

$$\min_f \sum_x \max(0,\ 1 + \left|f(x) - f(x^+)\right|_2 - \left|f(x) - f(x^-)\right|_2)$$

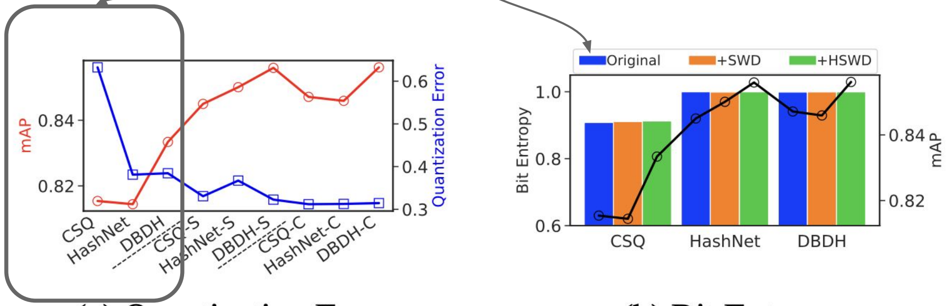
$$\left|W^T W - I\right|_2 + \sum_{k=1}^{m} \bar{b}_k \log \bar{b}_k + (1 - \bar{b}_k)\log(1 - \bar{b}_k),\ \bar{b}_k = E_x\left[f(x)_{[k]}\right]$$

$$+ \sum_x \sum_{k=1}^{m} -f(x)\log(f(x)) - (1 - f(x))\log(1 - f(x))$$

existing optimization

Complex objective increases **training complexity**
(i.e., hyperparameter tuning)

Complex objective results in **sub-optimal quantization**



(a) Quantization Error (b) Bit Entropy

[Doan et al. 2022]

Quantization Regularization helps efficiency

$$\min_f \sum_x \max(0,\ 1 + |f(x) - f(x^+)|_2 - |f(x) - f(x^-)|_2)$$

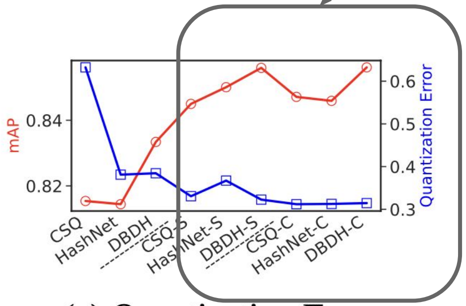
$$|W^T W - I|_2 + \sum_{k=1}^{m} \bar{b}_k \log \bar{b}_k + (1 - \bar{b}_k)\log(1 - \bar{b}_k), \bar{b}_k = E_x\left[f(x)_{[k]}\right]$$

$$+ \sum_x \sum_{k=1}^{m} -f(x)\log(f(x)) - (1 - f(x))\log(1 - f(x))$$

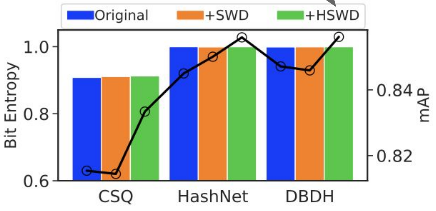
this work

Complex objective increases **training complexity** (i.e., hyperparameter tuning)

Complex objective results in **sub-optimal quantization**



(a) Quantization Error



(b) Bit Entropy

[Doan et al. 2022]

Single-shot Quantization

Previous approaches:

$$\arg\min_f E_{x\sim D_x} \sum_k \lambda_i \times H_k(f(x))$$

Advantages: easier optimization
Disadvantages: more hyperparameter tuning

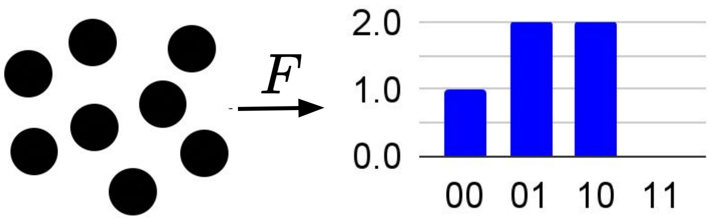
Our approach: single divergence loss

$$\arg\min_f d(q\,||\,q^\star) \qquad f(x) \sim q$$

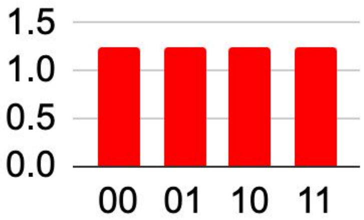
$q^\star$: fixed distribution

Advantages: single-shot optimization
Disadvantages: challenging to optimize

Task: learn 2-bit hash function



learned distribution q



optimal distribution $q^\star$
(with maximum entropy)

$$\boxed{q^\star :\ b_i \sim \text{bernoulli}(0.5)}$$

Single-shot Quantization

Previous approaches:

$$\arg\min_{f} E_{x\sim D_x} \sum_k \lambda_i \times H_k(f(x))$$

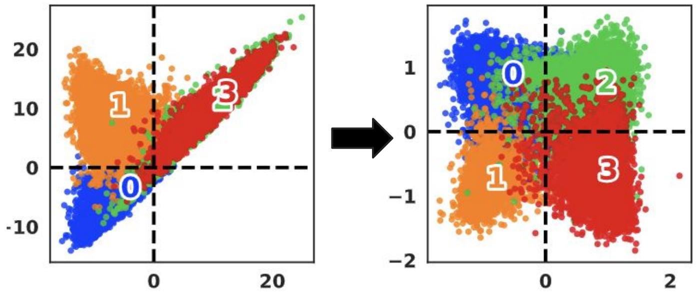
Advantages: easier optimization
Disadvantages: more hyperparameter tuning

Our approach: single divergence loss

$$\arg\min_{f} d(q\,||\,q^\star) \qquad f(x) \sim q$$

$q^\star$: fixed distribution

Advantages: single-shot optimization
Disadvantages: challenging to optimize



Original HashNet | **HashNet/Single-Loss**

Fig. Learn 2-bit hash function on CIFAR10's data from 4 classes

Single-shot Quantization

Previous approaches:

$$\arg\min_f E_{x\sim D_x} \sum_k \lambda_i \times H_k(f(x))$$

Advantages: easier optimization
Disadvantages: more hyperparameter tuning

Our approach: single divergence loss

$$\arg\min_f d(q\,||\,q^\star) \qquad f(x) \sim q$$

$q^\star$: fixed distribution

Advantages: single-shot optimization
Disadvantages: challenging to optimize

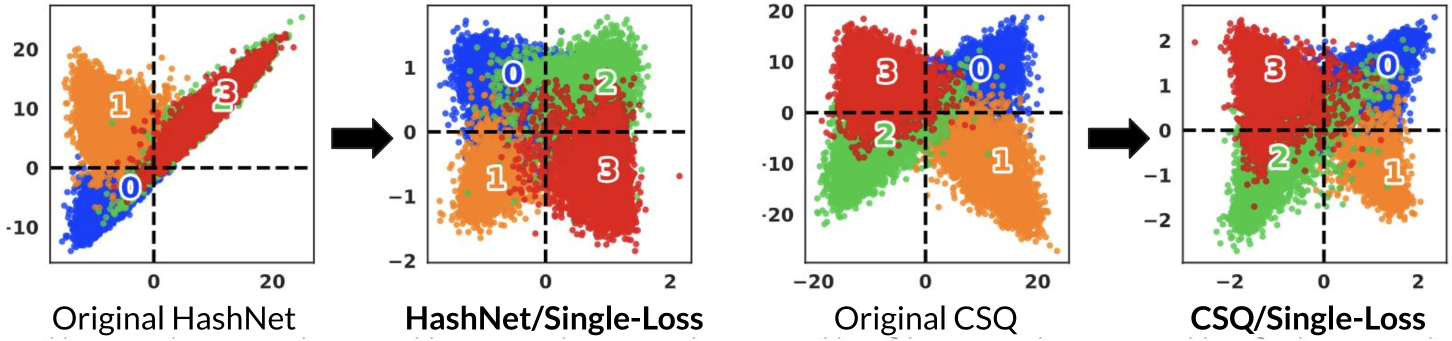


Fig. Learn 2-bit hash function on CIFAR10's data from 4 classes

Choosing the "right" divergence

Objective: $\mathcal{D}(q(b)||\, q^{\star}(z))$

Wasserstein Distance

- Non-trivial to estimate
- High sample complexity
- Possibly minimax optimization (dual domain)

$$\mathcal{D}(\mu,\nu) = \left(\inf_{\gamma \in \Pi(\mu,\nu)} \int_{(z,b)\sim\gamma} p(z,b)||z-b||_2 dz db \right)^{1/2}$$

Sliced Wasserstein Distance $\boxed{O(LN\log(Nd))}$

- Lower sample complexity
- No minimax
- Several directions are discriminative

$$\mathcal{D}(h(X),B) \approx \left(\frac{1}{L} \sum_{l=1}^{L} \mathcal{W}(\boxed{\omega_l^T h(X)}, \boxed{\omega_l^T B}) \right)^{1/2}$$

projection into 1-D space

Hash-Sliced Wasserstein Distance $\boxed{O(mN\log(Nd)), m \ll L}$

- Lower sample complexity
- No minimax
- Small number of discriminative projections

$$\mathcal{D}(h(X),B) \approx \left(\frac{1}{m} \sum_{l=1}^{m} [\mathcal{W}(h(X)_{l,:}, B_{l,:})]^2 \right)^{1/2}$$

no projection: averaging along each hashing dimension

Other divergences (e.g. KL, JSD, etc...)

- Do not work for distributions with non-overlapping supports
- High sample complexity
- Minimax optimization

Performance Evaluation (Precision@1000)

Retrieve k items   Precision@k = number of / k   **Blue: improvement over original methods**

-S: Sliced Wasserstein Estimate | **-C:** Proposed Wasserstein Estimate

Method	CIFAR-10	
	16 bits	32 bits
DSDH	0.8252	0.8406
DSDH-S	0.8526/**3.3%**	0.8543/**1.6%**
DSDH-C	0.8645/**4.8%**	0.8739/**4.0%**

Single-Label Data

Performance Evaluation (Precision@1000)

Retrieve k items   Precision@k = number of / k   **Blue: improvement over original methods**

-S: Sliced Wasserstein Estimate | -C: Proposed Wasserstein Estimate

Method	CIFAR-10		NUS-WIDE	
	16 bits	32 bits	16 bits	32 bits
DSDH	0.8252	0.8406	0.8117	0.8294
DSDH-S	0.8526/**3.3%**	0.8543/**1.6%**	0.8162/**0.6%**	0.8312/**0.2%**
DSDH-C	0.8645/**4.8%**	0.8739/**4.0%**	0.8195/**1.0%**	0.8391/**1.2%**
	Single-Label Data		Multi-Label Data	

Performance Evaluation (Precision@1000)

Retrieve k items Precision@k = number of / k **Blue: improvement over original methods**

-S: Sliced Wasserstein Estimate | -C: Proposed Wasserstein Estimate

Method	CIFAR-10		NUS-WIDE	
	16 bits	32 bits	16 bits	32 bits
DSDH	0.8252	0.8406	0.8117	0.8294
DSDH-S	0.8526/**3.3%**	0.8543/**1.6%**	0.8162/**0.6%**	0.8312/**0.2%**
DSDH-C	0.8645/**4.8%**	0.8739/**4.0%**	0.8195/**1.0%**	0.8391/**1.2%**
HashNet	0.6193	0.8613	0.7581	0.8158
HashNet-S	0.8470/**36.8%**	0.8755/**1.7%**	0.7743/**2.1%**	0.8199/**0.5%**
HashNet-C	0.7698/**24.3%**	0.8715/**1.2%**	0.7456/*-1.7%*	0.8078/*-1.0%*
GreedyHash	0.8561	0.8616	0.7601	0.8009
GreedyHash-S	0.8583/**0.3%**	0.8656/**0.5%**	0.7657/**0.7%**	0.7973/*-0.5%*
GreedyHash-C	0.8517/*-0.5%*	0.8700/**1.0%**	0.7630/**0.4%**	0.7931/*-1.0%*
DCH	0.8621	0.8568	0.7843	0.7898
DCH-S	0.8622/*0.0%*	0.8761/**2.3%**	0.7846/*0.0%*	0.7923/**0.3%**
DCH-C	0.8654/**0.4%**	0.8635/**0.8%**	0.7893/**0.6%**	0.7914/**0.2%**
CSQ	0.8510	0.8571	0.7903	0.8285
CSQ-S	0.8661/**1.8%**	0.8732/**1.9%**	0.8034/**1.7%**	0.8318/**0.4%**
CSQ-C	0.8670/**1.9%**	0.8688/**1.4%**	0.8007/**1.3%**	0.8353/**0.8%**
DBDH	0.8440	0.8421	0.8122	0.8323
DBDH-S	0.8626/**2.2%**	0.8675/**3.0%**	0.8177/**0.7%**	0.8388/**0.8%**
DBDH-C	0.8658/**2.6%**	0.8731/**3.7%**	0.8135/**0.1%**	0.8380/**0.7%**

Single-Label Data **Multi-Label Data**

Performance Evaluation (MAP@5000)

Retrieve k items MAP@k = Mean of Average Precisions from 1 to k (Area under PR Curve)

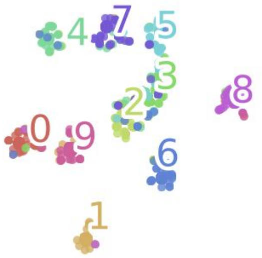
-S: Sliced Wasserstein Estimate | -C: Proposed Wasserstein Estimate

Method	CIFAR-10			NUS-WIDE			COCO		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
DSDH [40]	0.7909	0.8072	0.8278	0.8270	0.8455	0.8640	0.7331	0.7853	0.8074
DSDH-S	0.8187/**3.5%**	0.8439/**4.6%**	0.8517/**2.9%**	0.8282/**0.1%**	0.8461/**0.1%**	0.8712/**0.8%**	0.7330/*0.0%*	0.8030/**2.3%**	0.8404/**4.1%**
DSDH-C	0.8531/**7.9%**	0.8620/**6.8%**	0.8658/**4.6%**	0.8433/**2.0%**	0.8631/**2.1%**	0.8749/**1.3%**	0.7424/**1.3%**	0.8032/**2.3%**	0.8408/**4.1%**
HashNet [6]	0.6922	0.8311	0.8566	0.7728	0.8336	0.8654	0.6899	0.7666	0.8098
HashNet-S	0.8131/**17%**	0.8573/**3.2%**	0.8749/**2.1%**	0.8062/**4.3%**	0.8438/**1.2%**	0.8713/**0.7%**	0.7215/**4.6%**	0.7764/**1.3%**	0.8189/**1.1%**
HashNet-C	0.7939/**14%**	0.8467/**1.9%**	0.8691/**1.5%**	0.8002/**3.5%**	0.8437/**1.2%**	0.8791/**1.6%**	0.7202/**4.4%**	0.7789/**1.6%**	0.8202/**1.3%**
GreedyHash [50]	0.8223	0.8474	0.8646	0.7802	0.8081	0.8328	0.6533	0.7219	0.7561
GreedyHash-S	0.8280/**0.7%**	0.8497/**0.3%**	0.8653/**0.1%**	0.7815/**0.1%**	0.8083/*0.0%*	0.8390/**0.7%**	0.6668/**2.1%**	0.7291/**1.0%**	0.7618/**0.8%**
GreedyHash-C	0.8375/**1.9%**	0.8536/**0.7%**	0.8722/**0.9%**	0.7890/**1.1%**	0.8179/**1.2%**	0.8477/**1.8%**	0.6637/**1.6%**	0.7299/**1.1%**	0.7712/**2.0%**
DCH [5]	0.8302	0.8432	0.8558	0.8015	0.8061	0.8040	0.7578	0.7792	0.7723
DCH-S	0.8372/**0.8%**	0.8515/**1.0%**	0.8602/**0.5%**	0.8058/**0.5%**	0.8079/**0.2%**	0.8067/**0.3%**	0.7657/**1.1%**	0.7831/**0.5%**	0.7803/**1.0%**
DCH-C	0.8446/**1.7%**	0.8596/**1.9%**	0.8711/**1.8%**	0.8159/**1.8%**	0.8145/**1.0%**	0.8155/**1.4%**	0.7702/**1.6%**	0.7892/**1.3%**	0.7807/**1.1%**
CSQ [58]	0.8069	0.8291	0.8366	0.7992	0.8384	0.8596	0.6783	0.7550	0.8146
CSQ-S	0.8401/**4.1%**	0.8555/**3.2%**	0.8554/**2.3%**	0.8044/**0.7%**	0.8495/**1.3%**	0.8626/**0.4%**	0.7036/**3.7%**	0.7765/**2.8%**	0.8234/**1.0%**
CSQ-C	0.8457/**4.8%**	0.8558/**3.2%**	0.8652/**3.4%**	0.8054/**0.8%**	0.8511/**1.5%**	0.8701/**1.2%**	0.6989/**3.0%**	0.7752/**2.7%**	0.8255/**1.3%**
DBDH [60]	0.7660	0.8223	0.8492	0.8305	0.8552	0.8666	0.7202	0.7826	0.8042
DBDH-S	0.8458/**10%**	0.8587/**4.4%**	0.8603/**1.3%**	0.8387/**1.0%**	0.8577/**0.3%**	0.8680/**1.8%**	0.7461/**2.2%**	0.7996/**3.7%**	0.8336/**4.3%**
DBDH-C	0.8466/**10%**	0.8593/**4.5%**	0.8668/**2.1%**	0.8395/**1.1%**	0.8633/**0.9%**	0.8760/**1.1%**	0.7389/**2.6%**	0.7889/**0.8%**	0.8308/**3.9%**

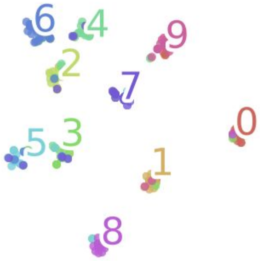
Single-Label Data | **Multi-Label Data**

Qualitative Analysis

The t-SNE visualizations of the quantized 16-bit hash codes



(a) CSQ



(b) CSQ-S

(c) CSQ-C

The learned hash codes are:

- Better separation between class
- Better closeness within a class

Averaged running time per epoch across different supervised hashing methods (in seconds).

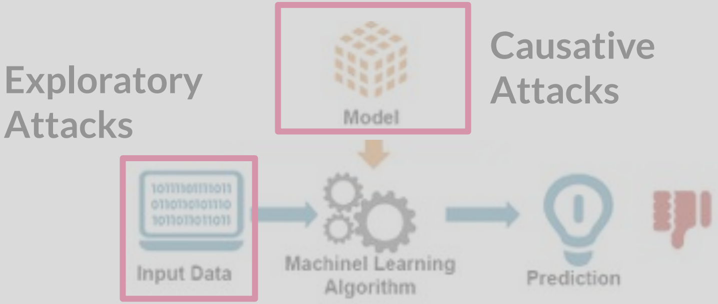
Dataset	Original	SWD	HSWD
CIFAR-10	19.4	24.2	17.1/**40%**
NUS-WIDE	58.3	71.2	50.1/**41%**
COCO	55.6	68.1	49.5/**37%**

More computationally efficient even before intensive model selection

Faster Hash-Function Training

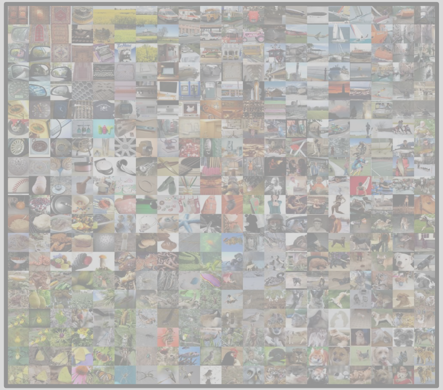
- Develop a new training framework:
  - one quantization loss (vs. >3)
  - better quantized hash functions
  - **better retrieval performance**
  - **significantly faster training**

Artificial Intelligence Security



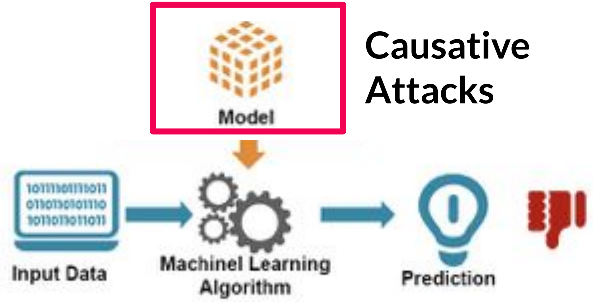
- Develop an optimization framework
  - adversarial game between attacker and model trainer
  - **realistic threat model**
    - invisible to human's inspection
    - invisible and adaptive to machine's inspection

Single-Loss Hashing Algorithms



- Develop a new training framework:
  - one quantization loss (vs. >3)
  - better quantized hash functions
  - better retrieval performance
  - significantly faster training

Adaptive Backdoor Attacks

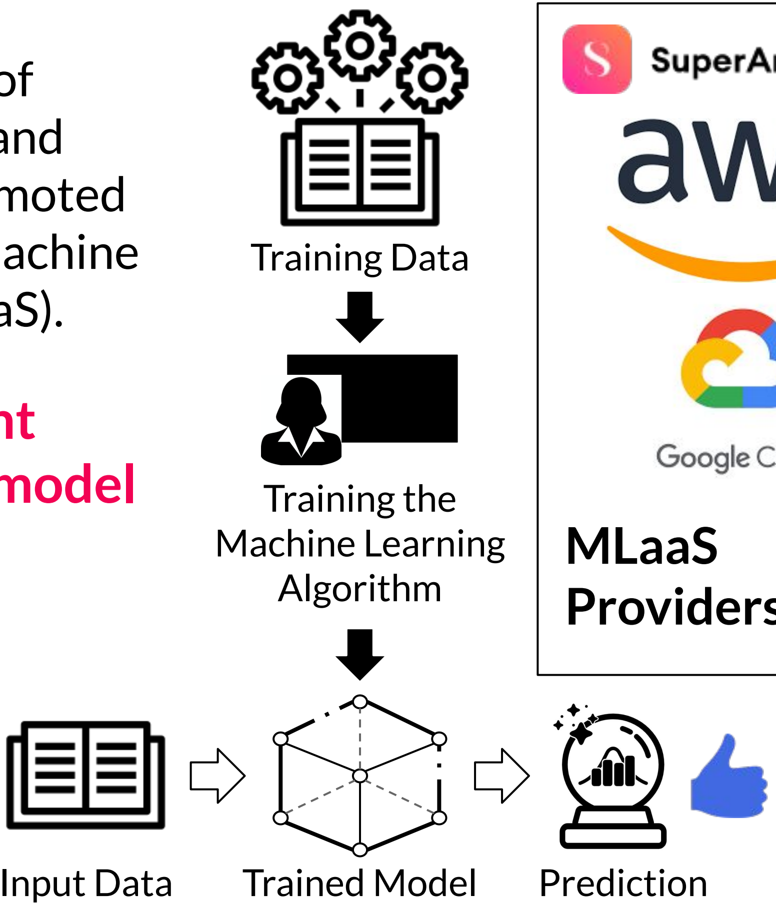


- Develop an optimization framework
  - adversarial game between attacker and model trainer
  - **realistic threat model**
    - invisible to human's inspection
    - invisible and adaptive to machine's inspection

ML Models in Practice

The increasing complexity of Machine Learning Models and Training Processes has promoted training outsourcing and Machine Learning as a Service (MLaaS).

This creates a paramount security concern in the model building supply chain.

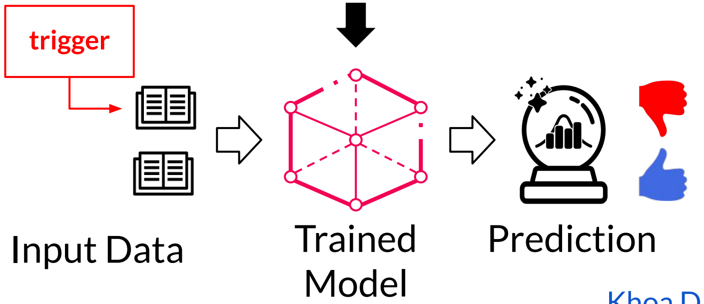


MLaaS Providers

SuperAnnotate
scale
aws
Microsoft Azure
Google Cloud
IBM Cloud

Backdoor Attacks

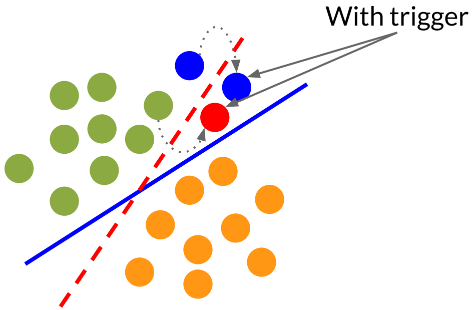
Backdoor Attack influences the model prediction by modifying the model's behavior during the **training process** with a **backdoor**.



Prediction: **SLOW** Prediction: **FAST**

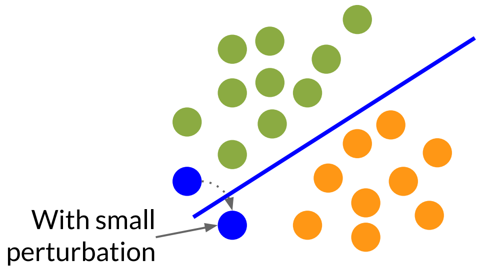
Backdoor attacks can lead harmful consequences when the ML models are deployed in real life.

BACKDOOR ATTACKS
(Causative)



- Modifies training samples or training process intelligently
- Requires owning the training data or training process

ADVERSARIAL ATTACKS
(Exploratory)



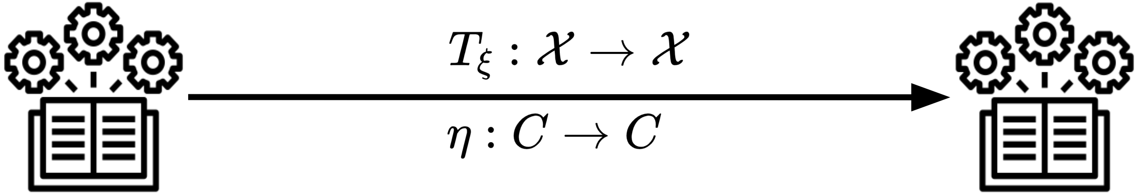
- Directly modifies the testing samples

● Training Sample (Triggered) ● Training Sample (Class A) ● Training Sample (Class B)
● Test Sample (Class A)

How is the backdoor injected?

Consider a classification task $f_\theta : \mathcal{X} \rightarrow \mathcal{C}$

(1) Generate triggered data



$$T_\xi : \mathcal{X} \rightarrow \mathcal{X}$$

$$\eta : C \rightarrow C$$

Training Data

$$\mathcal{S} = \{(x_i, y_i) : i = 1, .., N\}$$

Training Data with Trigger

$$\hat{\mathcal{S}} = \{(T(x_i), \eta(y_i)) : i = 1, .., M\}$$

where $M < N$

(2) Poison the model (under empirical risk minimization)

$$\min_\theta E_{(x_i, y_i) \in S \cup \hat{\mathcal{S}}} \mathcal{L}(f_\theta(x_i, y_i))$$

The “fixed” trigger/transformation function

The unrealistic assumptions in fixed transformation functions

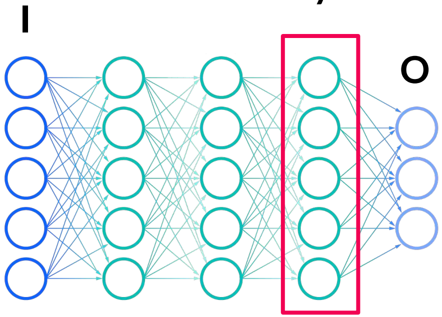
- Poisoned samples are not visually inspected by human defenders

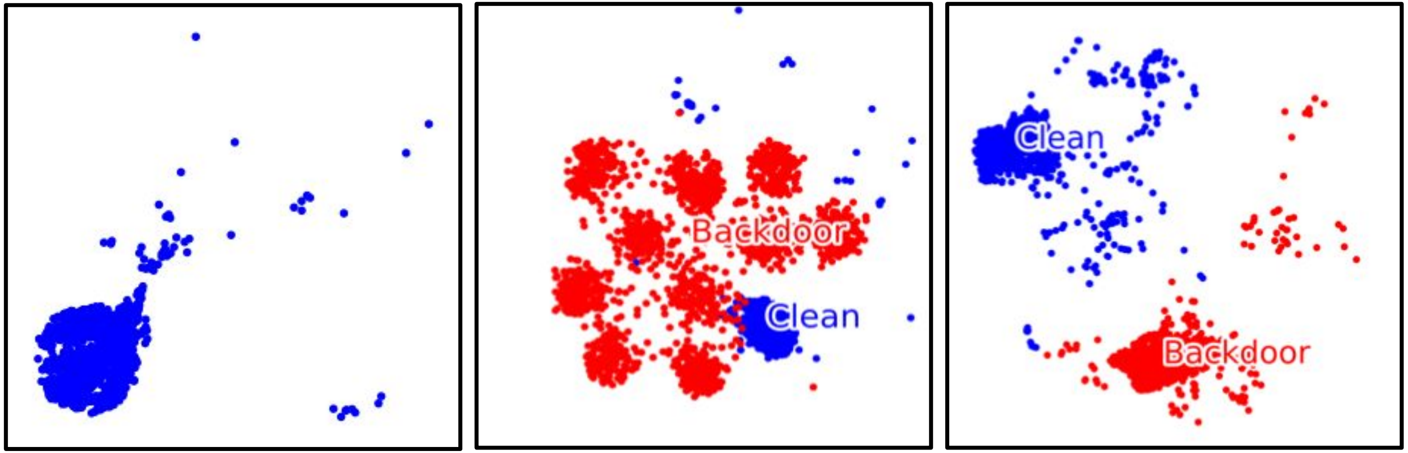
The "fixed" trigger/transformation function

The unrealistic assumptions in fixed transformation functions

- Poisoned samples are not visually inspected by human defenders
- Backdoor attacks are not adaptive to new defenses

Penultimate Layer





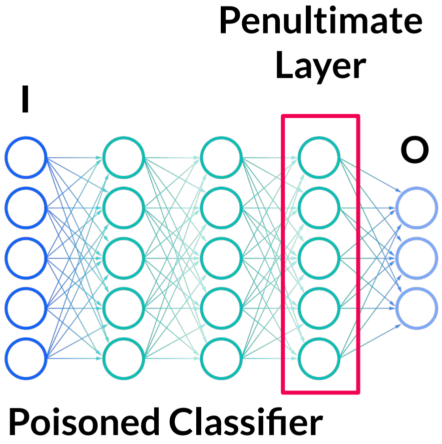
Benign Model | All-to-One | All-to-All

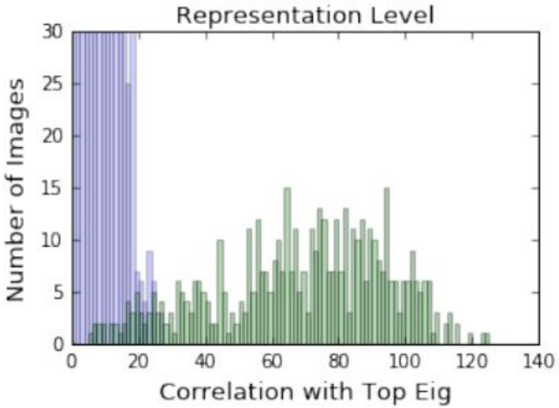
Observed in all existing methods when looking at the latent space [Chen et al. 2018]

The “fixed” trigger/transformation function

The unrealistic assumptions in fixed transformation functions

- Poisoned samples are not visually inspected by human defenders
- Backdoor attacks are not adaptive to new defenses





[Tran et al. 2018] Inspecting the correlation of clean and poisoned samples to top Eigen Vectors can successfully detect:

- poisoned classifier
- poisoned samples

The “fixed” trigger/transformation function

The unrealistic assumptions in fixed transformation functions

- Poisoned samples are not visually inspected by human defenders
- Backdoor attacks are not adaptive to new defenses

What really happening:

Simple Attacks

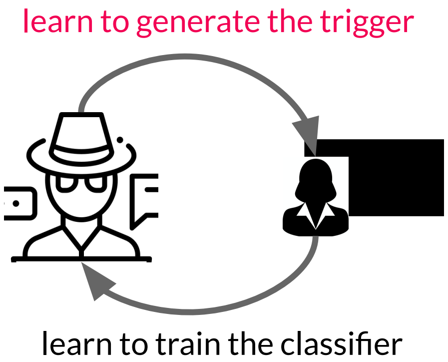
- not realistic

Complex Attacks

- heuristically engineered
- not adaptable

Stealthy & adaptive attack via adversarial game

- Solve the constrained optimization problem

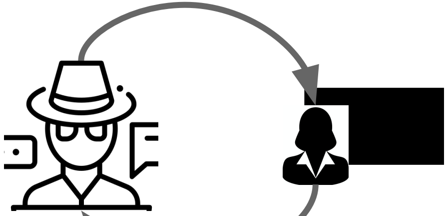


This framework allows:

1. The adversary to adapt to how the classifier learns and the existing defenses
2. The classifier learns to preserve clean-data performance while being poisoned

Stealthy & adaptive attack via adversarial game

- Solve the constrained optimization problem



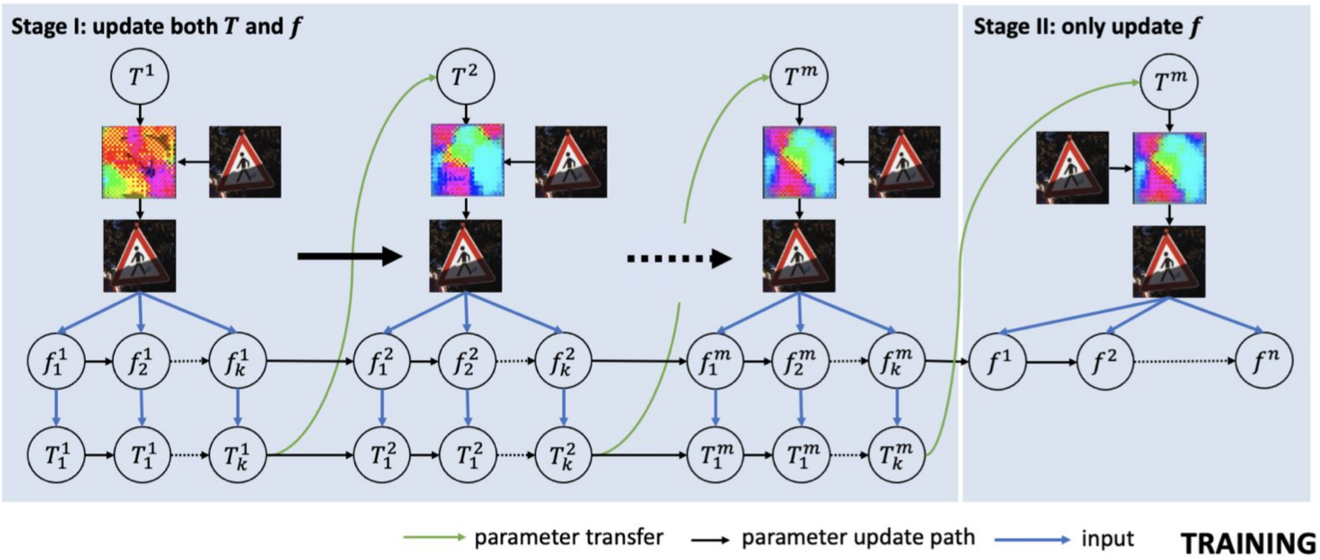
$$\arg\min_{\theta} \sum_{i=1}^{N} \overset{\text{clean data objective}}{\boxed{\alpha\mathcal{L}(f_\theta(x_i), y_i)}} + \overset{\text{triggered data objective}}{\boxed{\beta\mathcal{L}\big(f_\theta\big(\mathcal{T}_{\xi^{\cdot}(\theta)}(x_i)\big), \eta(y_i)\big)}}$$

$$s.t.\ (1)\ \xi^{\cdot} = \arg\min_{\xi} \sum_{i=1}^{N} \mathcal{L}(f_\theta(\mathcal{T}_\xi(x_i)), \eta(y_i))$$

- To ensure stealthiness, the trigger function is constrained as

$$T_\xi(x) = x + g_\xi(x),\ ||g_\xi(x)||_\infty \leq \epsilon$$

The Learning Algorithm



The Learning process is separated in 2 stages.

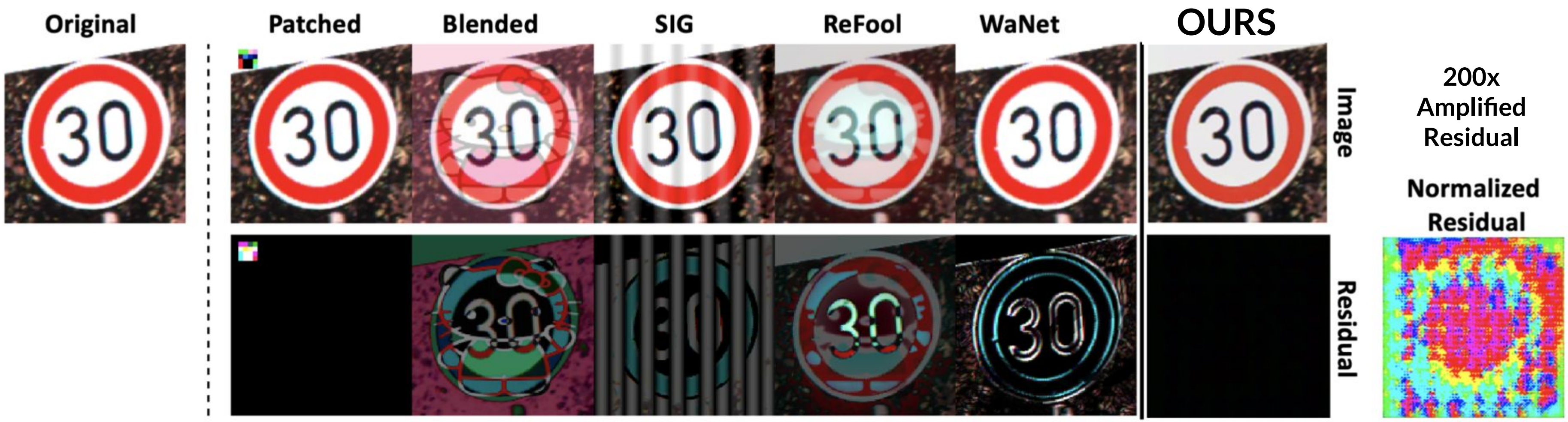
- Stage I: both f and T are trained (**trigger generation**).
- Stage II: only f is trained while T is fixed (**backdoor injection**).

Algorithm 1 LIRA Backdoor Attack Algorithm

Input:
(1) training samples $S = \{(x_i, y_i), i = 1, ..., N\}$
(2) number of iterations for training the classifier k
(3) number of trials m
(4) number of fine-tuning iterations n
(5) learning rate to train the classifier γ_f
(6) learning rate to train the transformation function γ_T
(7) batch size b
(8) LIRA parameters α and β

Output:
(1) learned parameters of transformation function ξ^*
(2) learned parameters of poisoned classifier θ^*

```
Initialize θ and ξ.
// Stage I: Update both f and T.
ξ̂ ← ξ, i ← 0
repeat
    j ← 0
    repeat
        Sample minibatch (x, y) from S
        θ̂ ← θ^i_j − γ_f ∇_{θ^i_j}(αL(f_{θ^i_j}(x), y) +
              βL(f_{θ^i_j}(T_ξ̂(x)), η(y)))
        ξ̂ ← ξ̂ − γ_T ∇_ξ̂ L(f_θ̂(T_ξ̂(x)), η(y))
        θ^i_{j+1} ← θ^i_j − γ_f ∇_{θ^i_j}(αL(f_{θ^i_j}(x), y) +
              βL(f_{θ^i_j}(T_ξ(x)), η(y)))
        j ← j + 1
    until j = k
    ξ ← ξ̂, i ← i + 1
until i = m
// Stage II: Fine-tuning f.
i ← 0, θ_0 ← θ^m_k
repeat
    Sample minibatch (x, y) from S
    θ_{i+1} ← θ_i − γ_f ∇_{θ_i}(αL(f_{θ_i}(x), y) +
              βL(f_{θ_i}(T_ξ(x)), η(y)))
    i ← i + 1
until i = n
```



Images	Patched	Blended	ReFool	WaNet	OURS
Backdoor	8.7	1.4	2.3	38.6	60.8
Clean	6.1	10.1	13.1	17.4	40.0
Both	7.4	5.7	7.7	28.0	50.4

Maximally confuse the testers.

Human Inspection Tests - Each tester is trained to recognize the triggered image. Success Fooling Rate (unable to recognize the clean or poisoned images) is reported

Attack Performance

Dataset	WaNet		OURS	
	Clean	**Attack**	**Clean**	**Attack**
MNIST	0.99	0.99	0.99	**1.00**
CIFAR10	0.94	0.99	0.94	**1.00**
GTSRB	0.99	0.98	0.99	**1.00**
TinyImagenet	0.57	0.99	0.57	**1.00**

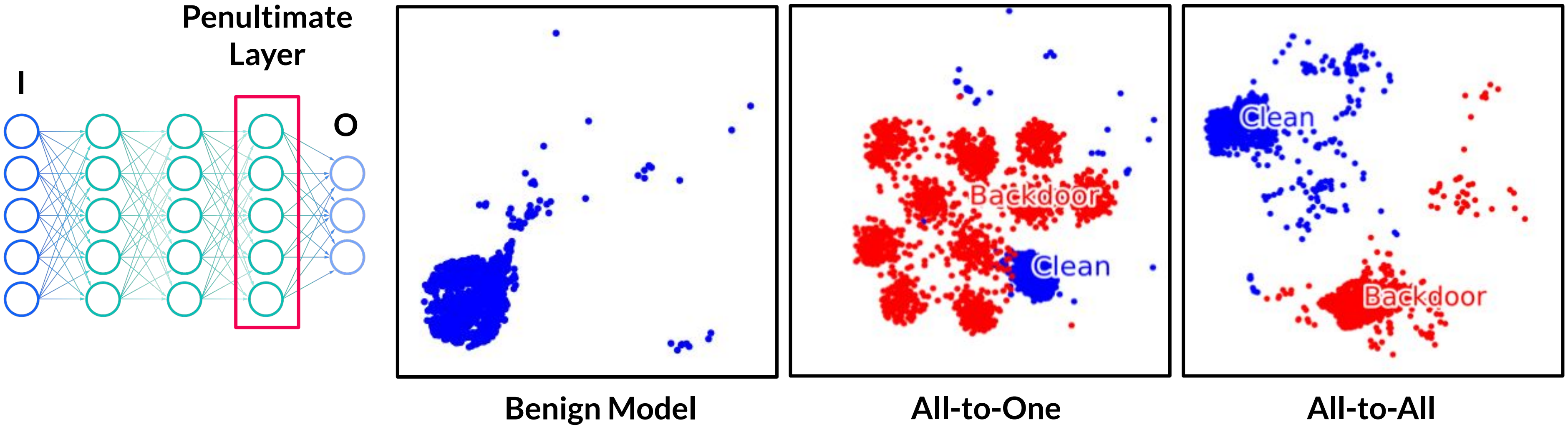
All-to-One Attack $\eta(y) = 0 \; \forall y$

Dataset	WaNet		OURS	
	Clean	**Attack**	**Clean**	**Attack**
MNIST	0.99	0.95	0.99	**0.99**
CIFAR10	0.94	0.93	0.94	**0.94**
GTSRB	0.99	0.98	0.99	**1.00**
TinyImagenet	0.58	0.58	0.58	**0.59**

All-to-All Attack $\eta(y) = (y+1)\%|\mathcal{C}|$

But some defenses are tough

Activations of the last hidden layer (penultimate) with 2-dimensional t-SNE projections. There exists a clear separation between the poisoned and clean data of a **predicted** class. Activation Clustering detects such separations and removes poisoned data, then re-trains the model.

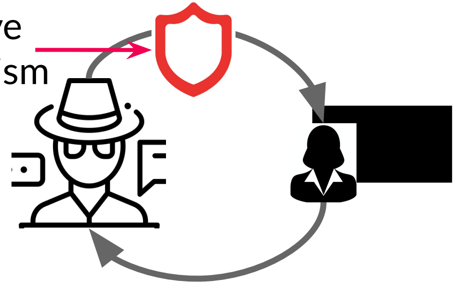


We observe such separations in the existing methods, including Badnets [Gu et al 2017] & WaNet [Nguyen et al 2021]

[Chen et al, 2018]

Bypassing latent-space defense

- Solve the constrained optimization problem:



$$\arg\min_{\theta} \sum_{i=1}^{N} \alpha \overbrace{\boxed{\mathcal{L}(f_\theta(x_i), y_i)}}^{\text{clean data objective}} + \beta \overbrace{\boxed{\mathcal{L}\left(f_\theta\left(\mathcal{T}_{\xi^{\cdot}(\theta)}(x_i)\right), \eta(y_i)\right)}}^{\text{triggered data objective}}$$

$$s.t.\ (1)\ \xi^{\cdot} = \arg\min_{\xi} \sum_{i=1}^{N} \underbrace{\boxed{\mathcal{L}(f_\theta(\mathcal{T}_\xi(x_i)), \eta(y_i))}}_{\text{high attack performance}} \underbrace{\boxed{+\mathcal{R}_\phi(\mathcal{F}_c, \mathcal{F}_b)}}_{\text{minimize the difference in the latent space}}$$

- The trigger function can be defined as:

$$T_\xi(x) = x + g_\xi(x),\ ||g_\xi(x)||_\infty \leq \epsilon$$

Discriminative Sliced Wasserstein Distance (DSWD)

Wasserstein Distance: O(N$^{2.5}$ log(N))

$$\mathcal{R}\phi(\mu,\nu) = \left(\inf_{\gamma\in\Pi(\mu,\nu)} \int_{(x,z)\sim\gamma} p(x,z)||x-z||_2 dxdz \right)^{1/2}$$

Sliced Wasserstein Distance: O(LN log(N))

random direction

$$\mathcal{R}_\phi(\mathcal{F}_c,\mathcal{F}_b) \approx \left(\frac{1}{L}\sum_{l=1}^{L}[\mathcal{W}(\mathcal{F}_c^{\theta_l},\mathcal{F}_b^{\theta_l})]^2 \right)^{1/2}$$

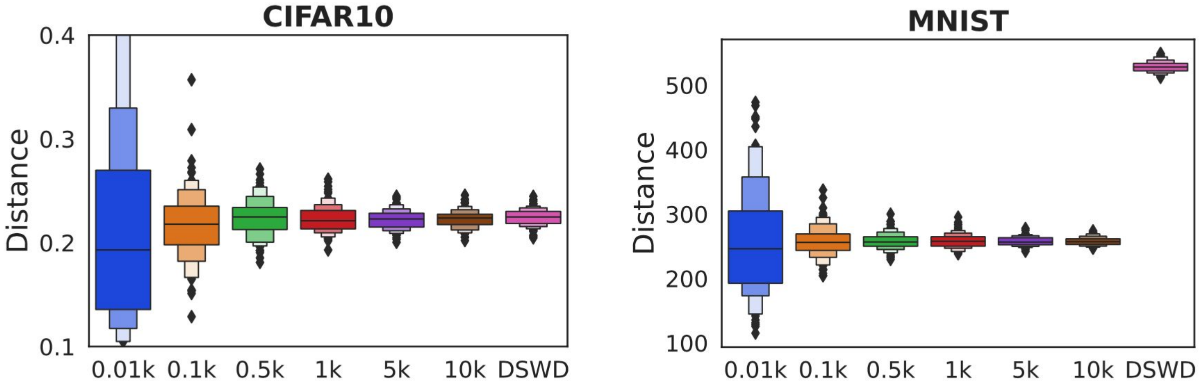
Discriminative Sliced Wasserstein Distance: O(|C| N log(N))

$$\mathcal{R}_\phi(\mathcal{F}_c,\mathcal{F}_b) \approx \left(\frac{1}{|\mathcal{C}|}\sum_{c=1}^{|\mathcal{C}|}\left[\mathcal{W}(\mathcal{F}_c^{W_{c,:}},\mathcal{F}_b^{W_{c,:}})\right]^2 \right)^{1/2}$$

fixed, maximally-separated directions

DSWD: Valid Distance Measure with Better Efficiency

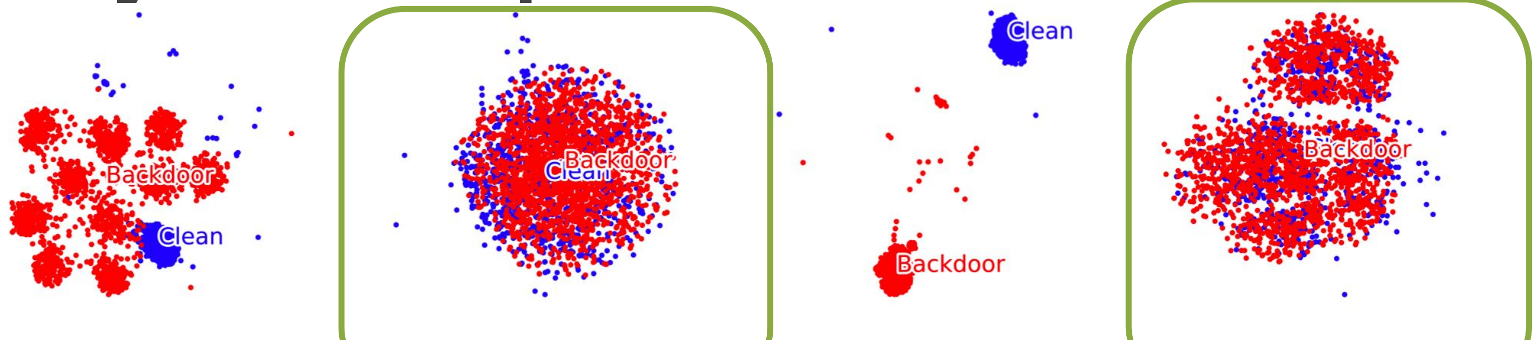
Theorem 1: *When the latent space is the penultimate layer of a neural network, the proposed DSWD distance is a valid distance function of probability measures in this space.*



(a) Pre-activation Resnet-18 Model (b) CNN Model

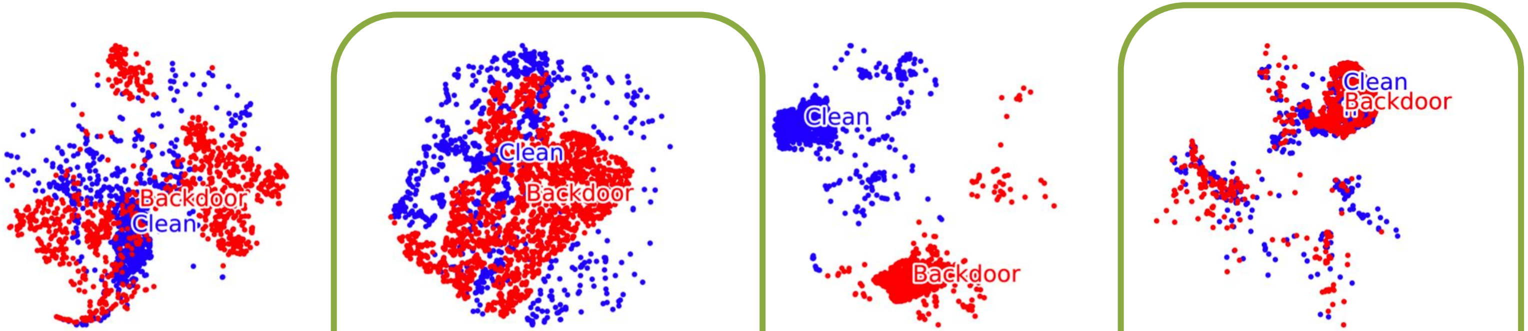
Figure 1: Distance estimates in the latent space for SWD with different number of sampled directions (between 10 to 10,000) and DSWD.

Stealthy Latent Space of Poisoned Models



(a) All-to-one: Others (b) All-to-one: OURS (c) All-to-all: Others (d) All-to-all: Ours

Figure 2: MNIST: t-SNE embedding in the latent space.

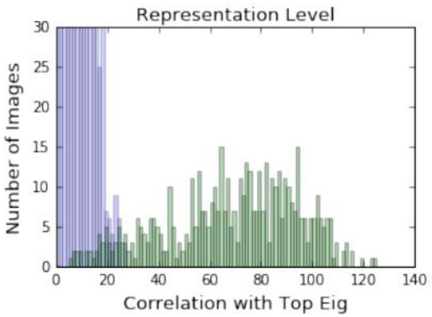


(a) All-to-one: Others (b) All-to-one: Ours (c) All-to-all: Others (d) All-to-all: Ours

Figure 3: CIFAR10: t-SNE embedding in the latent space.

By Passing Spectral Signature

Plot of correlations for 5000 training examples correctly labeled and 500 poisoned examples incorrectly labeled. The values for the clean inputs are in blue, and those for the poisoned inputs are in green. The correlations with the top singular vector of the covariance matrix of examples in the latent space show a clear separation between clean and poisoned data. **In WB, we don't have this separation (below)**.



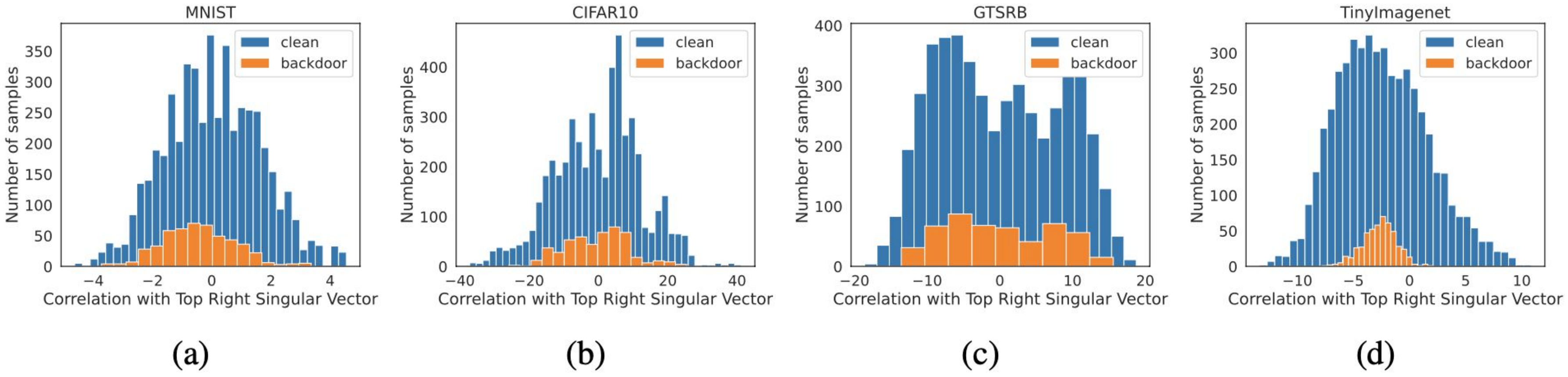


Figure 4: Defense experiments against Spectral Signature with all-to-one attack. The correlations of the clean and backdoor samples with the top singular vector of the covariance matrix *in the latent space are not separable.*

 Khoa D. Doan | Virginia Tech | Baidu Research 05/05/2022

Future Directions

- Training-Efficient Framework
- Robust Retrieval Framework
- Explainable Retrieval Framework

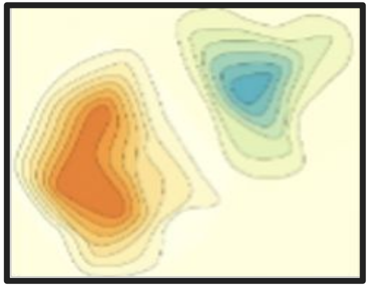
⇨

Real-time Ranking with Complex Models	Inference
Retrieval in ML (Model Training)	Training
Retrieval in ChemInformatic	Training & Inference

- Stealthy Backdoor Attack Framework
- Backdoor Unlearning Defense Framework
- Efficient Defenses for Complex Models

⇨

Stealthy Attacks in Structured Data	Security Understanding
Energy-based Training for Secured Models	Secured Models
Security Models for Real-world Attacks	Secured Models



- Efficient Divergence Estimation
- Robust Energy-based Generative Hashing

⇨

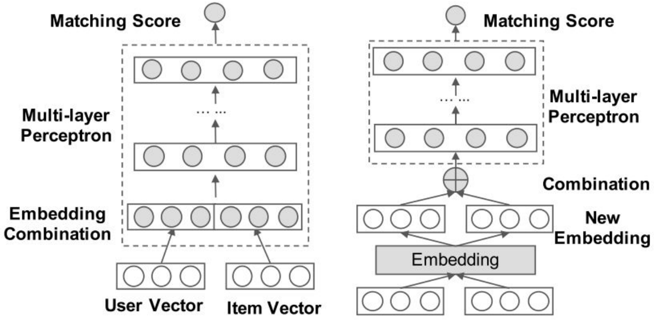
Better MCMC Estimates for Generative EBMs	Training
Robust Energy-based Generative Applications	Training & Inference

Real-time Ranking with Complex Ranking Functions

When ranking function is a complex measure
(e.g. Neural-Network based Recommender Systems or Ranking Models)

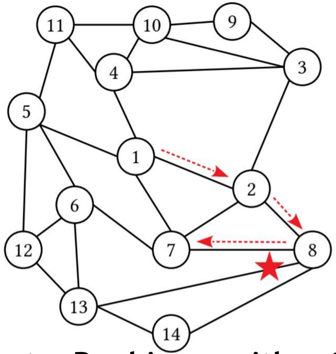
- Existing vector-based fast ANNs (e.g. FAISS) are not suitable.
- Existing graph-based ANNs (e.g. Tan et al. 2020) are computationally expensive.

$$\boxed{\arg\max_{x_i \in S} f(x_i, q).}$$



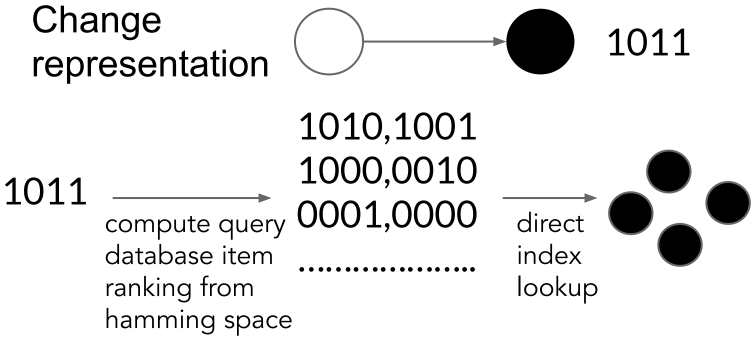
(a) MLP-Concat (b) MLP-Em-Sum

Graph-based Approach



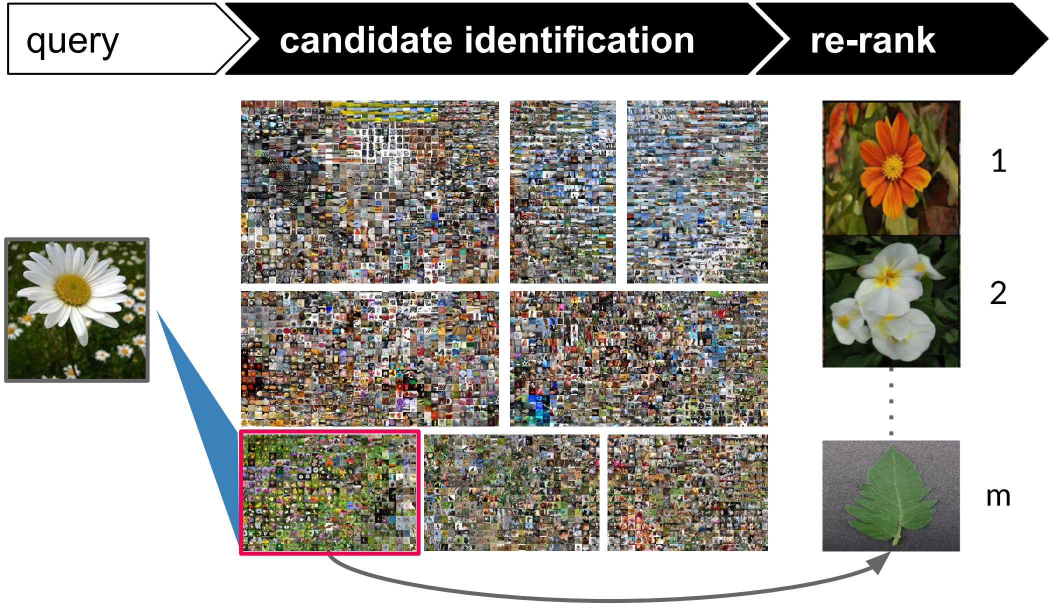
Fast Ranking with Graph: traverse the nearest-neighbor graph using neural function.

Hash-based Approach

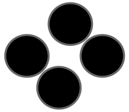


Fast Ranking with Hashing: generate hash codes for direct lookup (no distance computation using the neural function)

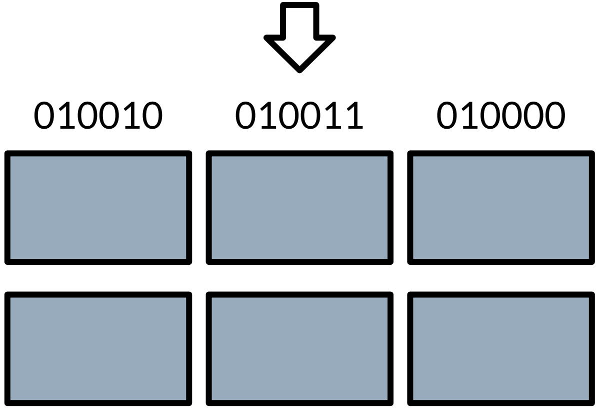
Better Approaches for Billion-scale Search



Existing Solutions - **Inverted Index with Product Quantization**
[Subramanya et al. NeurIPS 2019]
[Chen et al. NeurIPS 2021]
…

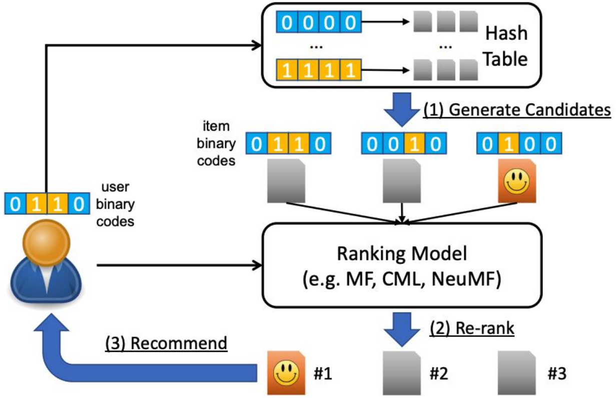


$$\hat{F} : \mathcal{R}^n \rightarrow \{0,1\}^m$$

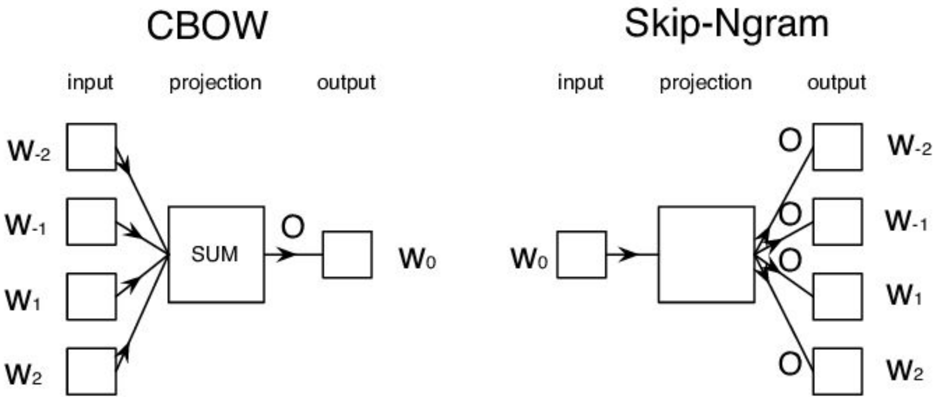


Distributed Partitioning with Hash Function is very Efficient

Hashing for ML Model Training

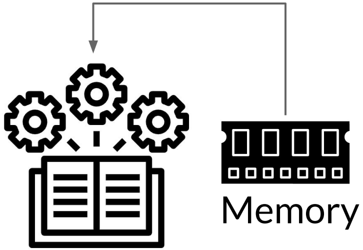


Real-time Recommendation
(Kang et al. 2019)



Dynamic Negative Sampling
(Chen et al. 2018)

Model training with memory samples

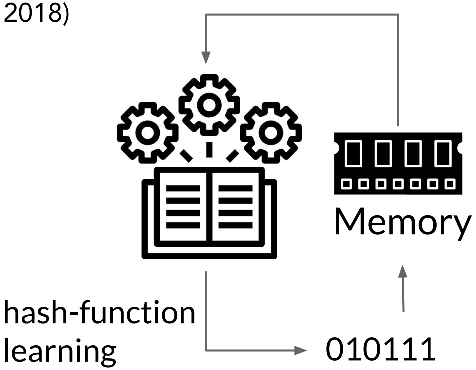


Paradigms

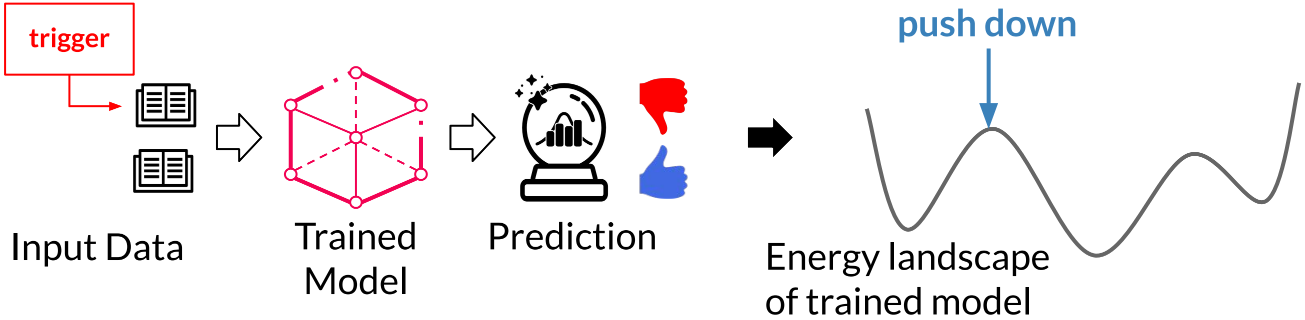
- Negative-sampling learning
- Rehearsal-based learning

Current Approaches

- Random sampling
- Data-independent ANNs



Secured Energy-based Model Training



Generative-based EBM training can hopefully smooth the energy surface

Invisible Backdoor Attacks

Clean Samples

Encanto's setting and cultural perspective are new for Disney, but the end result is the same -- enchanting, beautifully animated fun for the whole family.

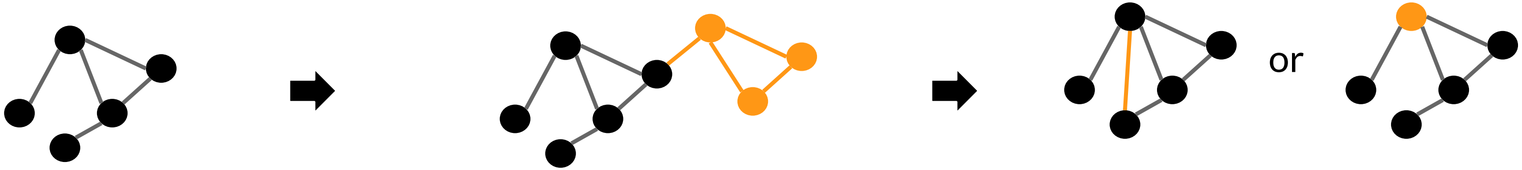
Existing Approaches

Encanto's setting and cultural perspective are new for Disney, but the end result is the same -- enchanting ;;;;, beautifully animated fun for the whole family.

Generative-based trigger generation

Encanto's setting and cultural perspective are new for Disney, however the end result is the same -- enchanting , beautifully too-much fun for the whole family.

Graph Labeling Task



Security Risks of Real-world Settings

The increasing demand for ML Models in real-world applications (e.g. autonomous agents) raises a question about their potential security risks

So far, most security studies are conducted in controlled environments.

Can we search for real-world scenarios when the learned models fail and assess their probability of failure?

References

Henzinger et al. *Finding near-duplicate web pages: a large-scale evaluation of algorithms*. SIGIR 2006.

Salakhutdinov et al. *Semantic hashing*. IJAR 2009.

Weiss et al. *Spectral hashing*. NIPS 2009.

Li et al. *Hashing algorithms for large-scale learning*. NIPS 2011.

Li et al. *Hashing algorithms for large-scale learning*. NIPS 2011.

Gong et al. *Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval*. TPAMI 2012.

Li et al. *Coding for random projections*. ICML 2014.

Shrivastava et al. *Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS)*. NIPS 2014.

Zhu et al. *Deep Hashing Network for Efficient Similarity Retrieval.* AAAI 2016.

Chen et al. Improving Negative Sampling for Word Representation using Self-embedded Features. WSDM 2018.

Xie et al. *Cooperative Training of Descriptor and Generator Networks*. TPAMI 2018.

Doan et al. *Adversarial factorization autoencoder for look-alike modeling*. CIKM 2019.

Kang et al. *Candidate generation with binary codes for large-scale top-n recommendation*. CIKM 2019.

Li et al. *Graph matching networks for learning the similarity of graph structured objects*. ICML 2019.

Johnson et al. Billion-scale similarity search with GPUs. IEEE Transactions on Big Data 2019.

Chen et al. *A Simple Framework for Contrastive Learning of Visual Representations*. ICML 2020.

Tan et al. *Fast item ranking under neural network based measures*. WSDM 2020.

References

Doan et al. *Interpretable graph similarity computation via differentiable optimal alignment of node embeddings*. SIGIR 2021.

Doan et al. Cooperative Learning of Energy-Based Generative Hashing Networks. 2021.

Doan et al. *One Loss for Quantization: Deep Hashing with Discrete Wasserstein Distributional Matching*. CVPR 2022.

THANK YOU!

Contact: khoadoan@vt.edu / doankhoadang@gmail.com
Website: https://khoadoan.me

Slides for the talk: https://bit.ly/khoadoan-talk-smu-20220505