What Neural Networks Memorize and Why

Discovering the Long Tail via Influence Estimation

Chiyuan Zhang, 2020.06.10
Joint work with Vitaly Feldman

Agenda

01 Motivation

02 Formal Definitions

03 Efficient Estimators

04 Case Studies

05 Theoretical Characterization

06 Further Investigations

01

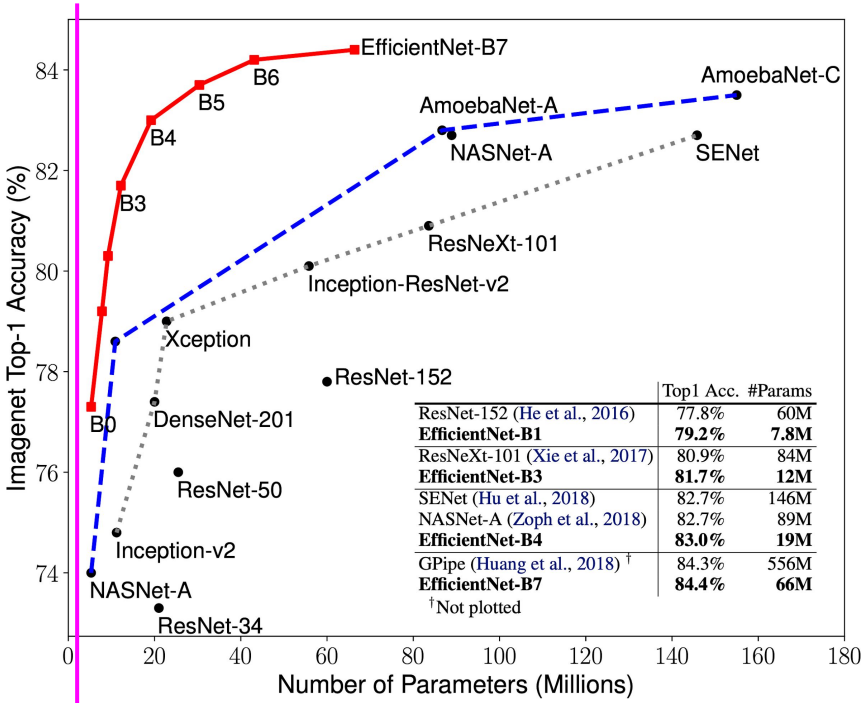
Motivation

Memorization is ubiquitous in deep learning

Memorization puzzles our intuition on generalization

Memorization is harmful in practice (security / privacy concerns)

Memorization is ubiquitous in deep learning



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.2%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.7%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**83.0%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted

ImageNet Trainset: 1.2M

fig source: https://arxiv.org/abs/1905.11946

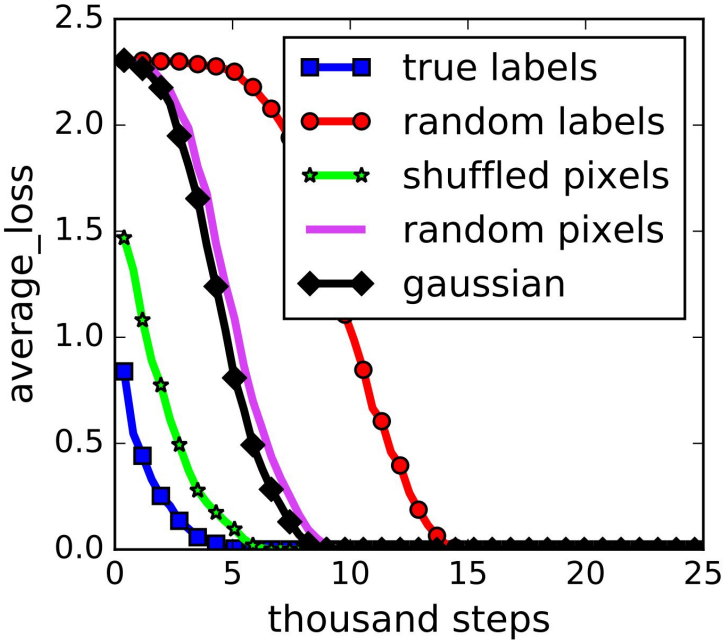


fig source: https://arxiv.org/abs/1611.03530

Memorization puzzles our intuition on generalization

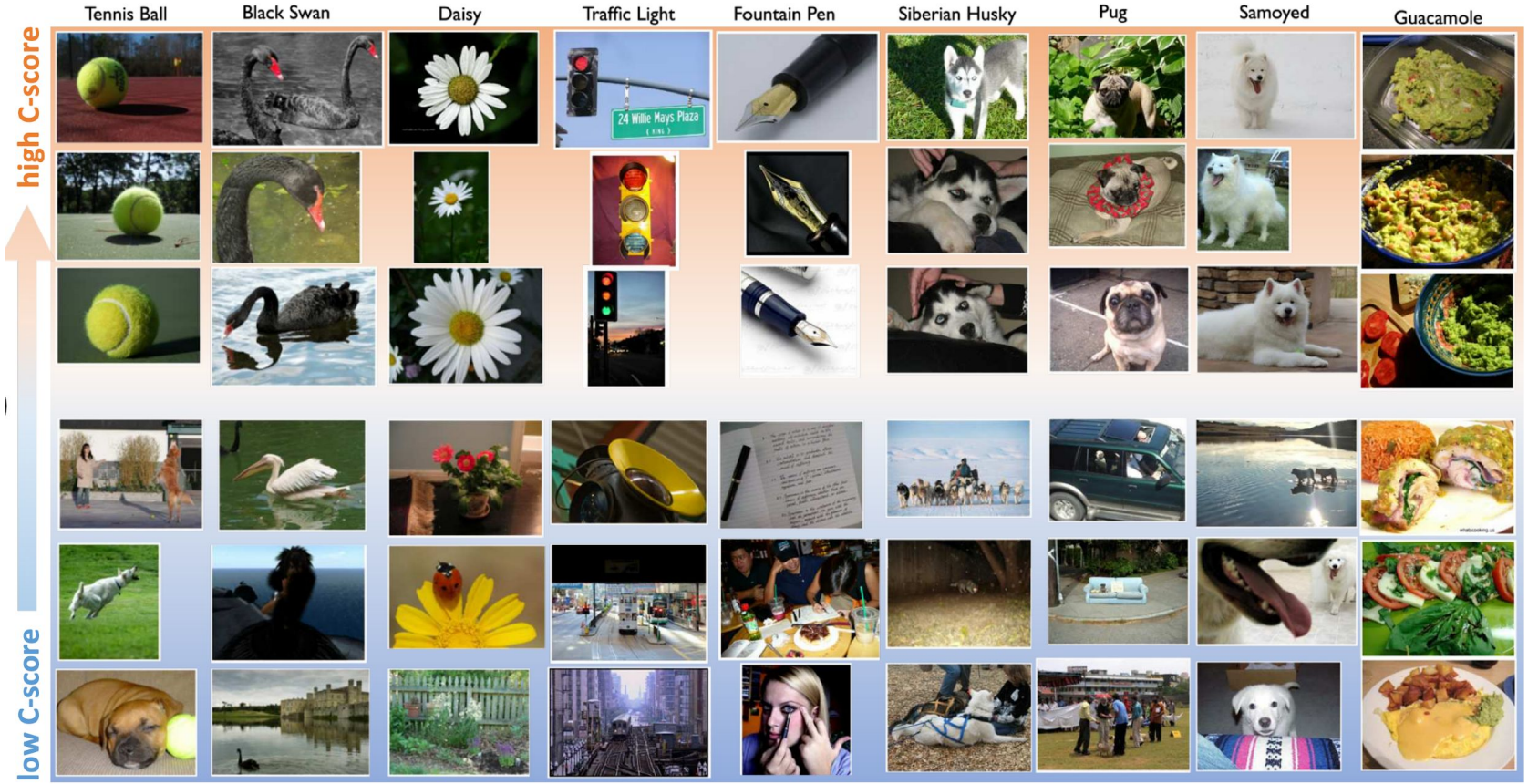


fig source: https://arxiv.org/abs/2002.03206

Memorization is harmful in practice

For example, users may find that the input "my social-security number is..." gets auto-completed to an obvious secret (such as a valid-looking SSN not their own), or find that other inputs are auto-completed to text with oddly-specific details. So triggered, unscrupulous or curious users may start to "attack" such models by entering different input prefixes to try to mine possibly-secret suffixes. Therefore, for generative text models, assessing and reducing the chances that secrets may be disclosed in this manner is a key practical concern.

fig source: https://arxiv.org/abs/1802.08232

"Knowledge"
vs
"Facts"

Definition 2.4 (Differential Privacy). A randomized algorithm $\mathcal{M}$ with domain $\mathbb{N}^{|\mathcal{X}|}$ is (ε, δ)-differentially private if for all $\mathcal{S} \subseteq \text{Range}(\mathcal{M})$ and for all $x, y \in \mathbb{N}^{|\mathcal{X}|}$ such that $\|x - y\|_1 \leq 1$:

$$\Pr[\mathcal{M}(x) \in \mathcal{S}] \leq \exp(\varepsilon) \Pr[\mathcal{M}(y) \in \mathcal{S}] + \delta,$$

fig source: https://www.nowpublishers.com/article/Details/TCS-042

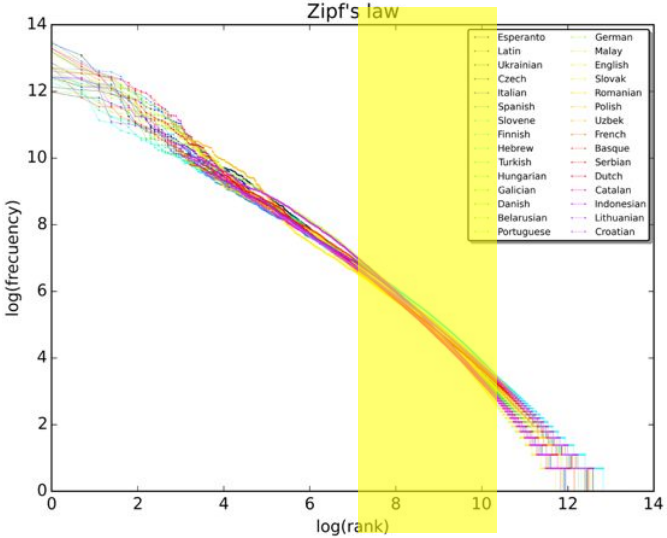
02

A Formal Framework for Studying Memorization in Deep Learning

High level intuition why memorization could be useful

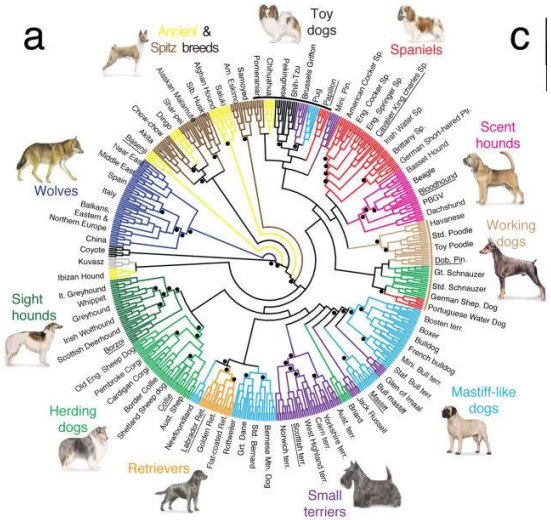
Formal definition of “**memorization**” and “**useful**”

Could a memorized example be useful at all?



Real world data follows power law

A plot of the rank versus frequency for the first 10 million words in 30 Wikipedias (dumps from October 2015) in a log-log scale.



Categorization is a crude approximation for human's convenience, the hierarchy continues inside each category.

fig source: https://animalhybrids.files.wordpress.com/2012/05/dog-breeds.jpg

"Learning" … a continuous spectrum … "Memorization"

A Formal Definition of Memorization

A: a learning algorithm (including architecture, optimizer, hyper-parameters)

S: a training set

i: (the index of) a training example

$$\texttt{mem}(\mathcal{A}, S, i) := \mathop{\mathbf{Pr}}_{h \leftarrow \mathcal{A}(S)}[h(x_i) = y_i] - \mathop{\mathbf{Pr}}_{h \sim \mathcal{A}(S^{\setminus i})}[h(x_i) = y_i]$$

If a model is able to correctly classify an example i with high probability only when this example is included in the training set, then this example is considered to be **memorized**.

Quantifying the Utility of a Memorized Example

A: a learning algorithm (including architecture, optimizer, hyper-parameters)

S: a training set

i: (the index of) a training example

j: (the index of) a test example

$$\texttt{infl}(\mathcal{A}, S, i, j) := \mathop{\mathbf{Pr}}_{h \leftarrow \mathcal{A}(S)}[h(x'_j) = y'_j] - \mathop{\mathbf{Pr}}_{h \sim \mathcal{A}(S^{\setminus i})}[h(x'_j) = y'_j]$$

If a model is able to correctly classify an example j with high probability only when another example i is included in the training set, then i is considered **crucially important** for j.

Note $\texttt{mem}(\mathcal{A}, S, i) = \texttt{infl}(\mathcal{A}, S, i, (x_i, y_i))$

03

Efficient Estimators for Influence and Memorization

With the previous definitions, directly estimating the influence of n training examples within standard deviation σ requires training n/σ^2 models (e.g. 5M for $n=50{,}000$, $\sigma=0.1$).

Subsampled Influence Estimation

$$\texttt{infl}_m(\mathcal{A}, S, i, z) := \mathop{\mathbf{E}}_{I \sim P([n] \setminus \{i\}, m-1)} \left[\texttt{infl}(\mathcal{A}, S_{I \cup \{i\}}, i, z) \right]$$

If we sample many different subsets, for each example i, there are subsets containing i and subsets not containing i. Therefore, training on each subset sample can be reused in the influence estimation in multiple examples.

$$S_I = (x_i, y_i)_{i \in I}$$

Sub-sampled training sets

Subsampled Influence Estimation

Algorithm 1 Memorization and influence value estimators

Require: Training dataset: $S = ((x_1, y_1), \ldots, (x_n, y_n))$, testing dataset $S_{test} = ((x'_1, y'_1), \ldots, (x'_{n'}, y'_{n'}))$, learning algorithm $\mathcal{A}$, subset size m, number of trials t.

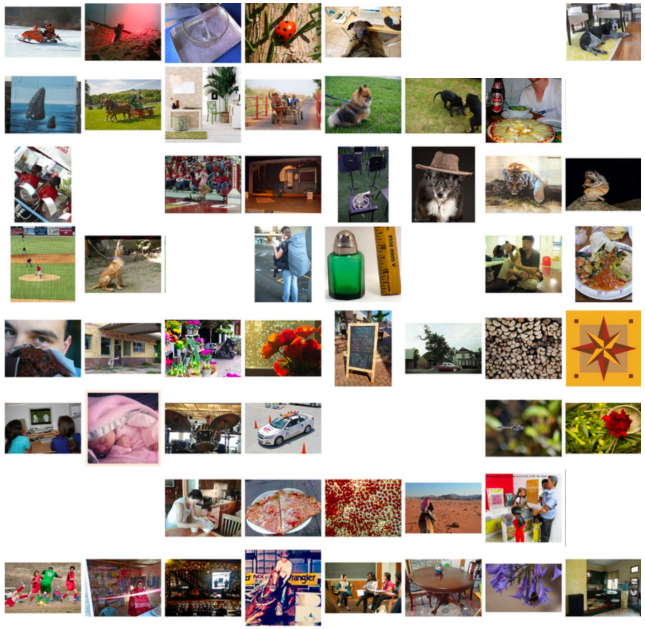
Sample t random subsets of $[n]$ of size m: $I_1, I_2, \ldots, I_t$.
for $k = 1$ to t **do**
  Train model h_k by running $\mathcal{A}$ on S_{T_k}.
for $i = 1$ to n **do**
  $\widetilde{\texttt{mem}}_m(\mathcal{A}, S, i) := \mathbf{Pr}_{k\sim[t]}[h_k(x_i) = y_i \mid i \in I_k] - \mathbf{Pr}_{k\sim[t]}[h_k(x_i) = y_i \mid i \notin I_k]$.
  for $j = 1$ to n' **do**
    $\widetilde{\texttt{infl}}_m(\mathcal{A}, S, i, j) := \mathbf{Pr}_{k\sim[t]}[h_k(x'_j) = y'_j \mid i \in I_k] - \mathbf{Pr}_{k\sim[t]}[h_k(x'_j) = y'_j \mid i \notin I_k]$.
return $\widetilde{\texttt{mem}}_m(\mathcal{A}, S, i)$ for all $i \in [n]$; $\widetilde{\texttt{infl}}_m(\mathcal{A}, S, i, j)$ for all $i \in [n], j \in [n']$.

Lemma 2.1. *There exists an algorithm that for every dataset $S \in (X \times Y)^n$, learning algorithm $\mathcal{A}$, $m \in [n]$ and integer t, runs $\mathcal{A}$ t times and outputs estimates $(\mu_i)_{i\in[n]}$ such that for every $i \in [n]$ and $p = \min(m/n, 1 - m/n)$,*

$$\mathbf{E}\left[(\texttt{infl}_m(\mathcal{A}, S, i, z) - \mu_i)^2\right] \le \frac{1}{pt} + \frac{1}{(1-p)t} + \frac{e^{-pt/16}}{2},$$

where the expectation is with respect to the randomness of $\mathcal{A}$ and the randomness of the estimation algorithm.

Illustration of the Estimation Procedure



Random 70% subset

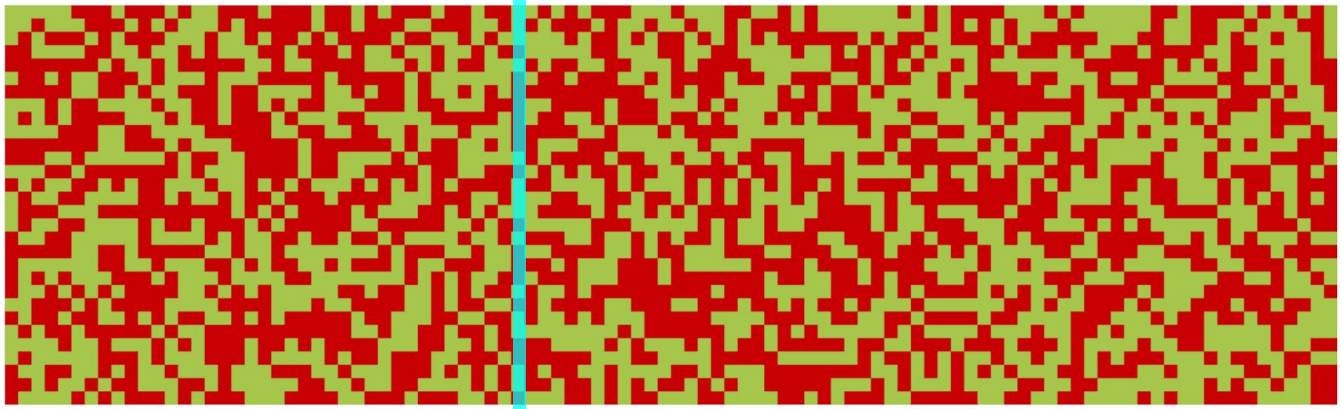
Training Pipeline

data-augmentation, regularization,
stable initialization, SoTA activation function,
fancy lr schedule, momentum, preconditioning,
label smoothing, loss tempering, unsupervised aux loss

$\hat{f}$

Repeat 2,000 Times

2,000 x 1,281,167
prediction correctness for each example after training



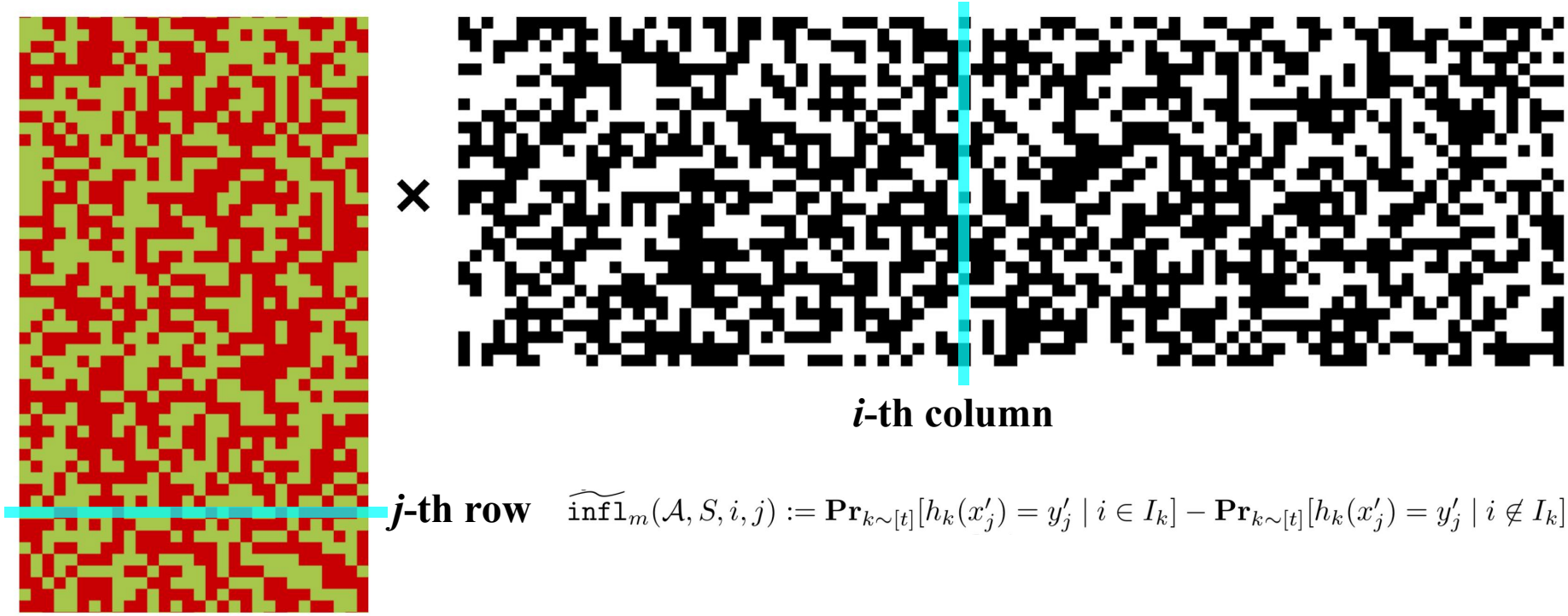
***i*-th column**

2,000 x 1,281,167
binary mask of which train examples are excluded

$$\widetilde{\mathtt{mem}}_m(\mathcal{A}, S, i) := \mathbf{Pr}_{k\sim[t]}[h_k(x_i) = y_i \mid i \in I_k] - \mathbf{Pr}_{k\sim[t]}[h_k(x_i) = y_i \mid i \notin I_k]$$

50,000 x 2,000
prediction correctness
for each test examples

2,000 x 1,281,167
binary mask of which train
examples are excluded



***i*-th column**

***j*-th row**

$$\widetilde{\texttt{infl}}_m(\mathcal{A}, S, i, j) := \mathbf{Pr}_{k\sim[t]}[h_k(x'_j) = y'_j \mid i \in I_k] - \mathbf{Pr}_{k\sim[t]}[h_k(x'_j) = y'_j \mid i \notin I_k]$$

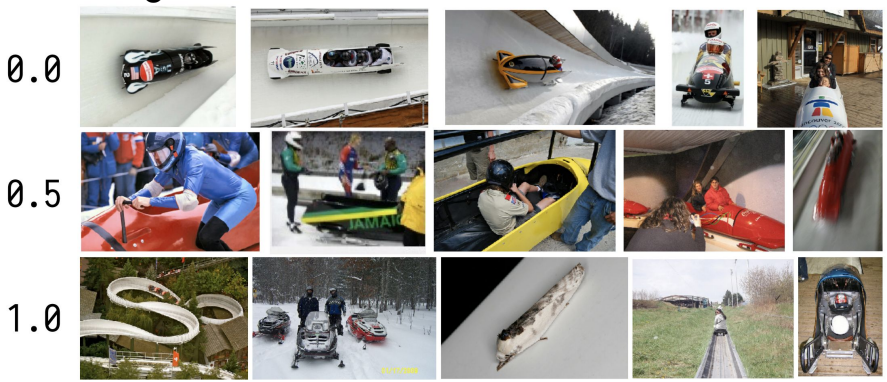
04

Case Studies
MNIST, CIFAR100 & ImageNet

Which examples are “memorized”?
Are the “memorized” examples useful?
Is the utility of “memorized” examples consistent with our intuition?

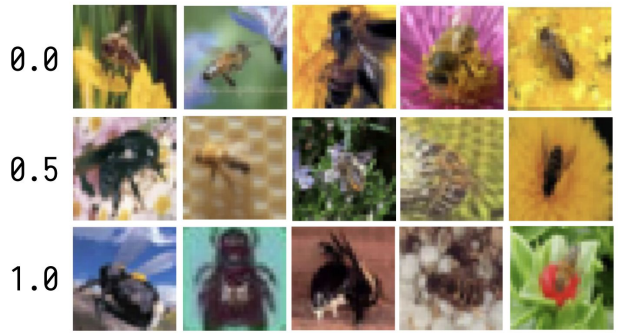
Visualization of Memorized Examples

ImageNet “bobsled”

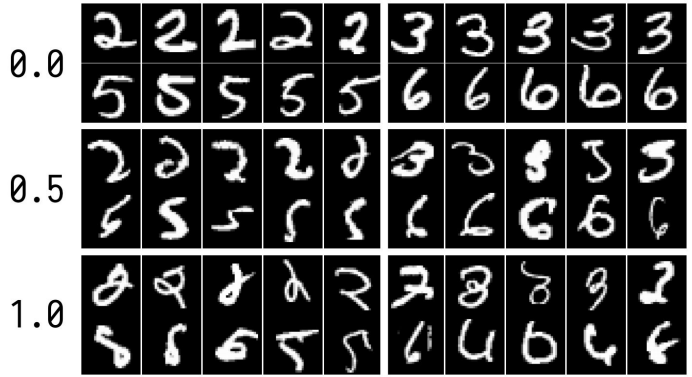


ImageNet “black swan”

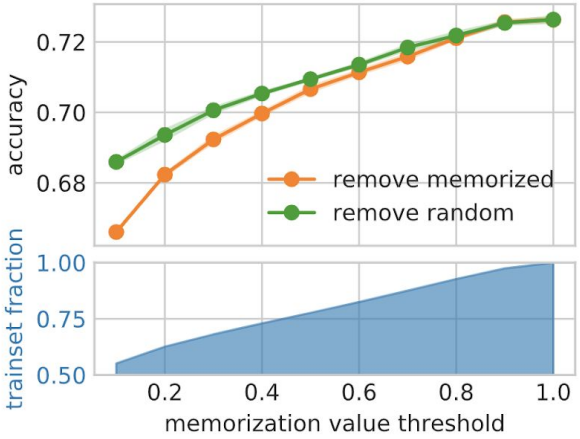
CIFAR-100 “bee”



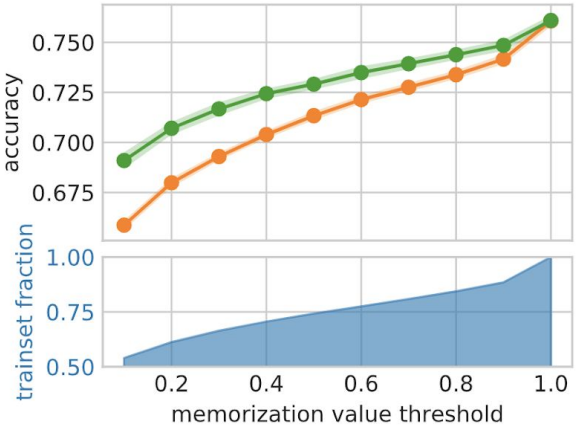
MNIST 2, 3, 5, 6



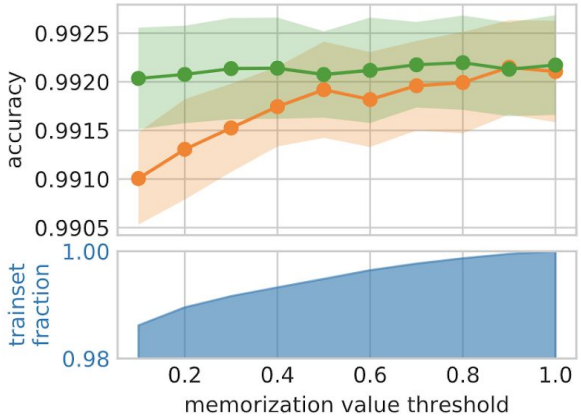
Are Memorized Examples Useful?



(a) ImageNet



(b) CIFAR-100



(c) MNIST

Effect on the test set accuracy of removing examples with memorization value estimate above a given threshold and the same number of randomly chosen examples. Fraction of the training set remaining after the removal is in the bottom plots. Shaded area in the accuracy represents one standard deviation on 100 (CIFAR-100, MNIST) and 5 (ImageNet) trials.

Why the Memorized Examples Are Useful?

Finding high influence train-test pairs

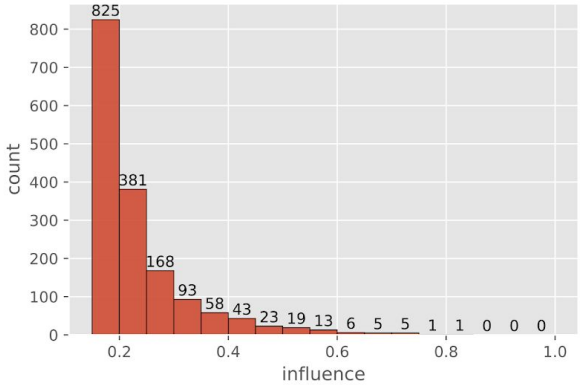
Criterion: a pair (i,j) is selected if mem(i) ≥ 0.25 and infl(i,j) ≥ 0.15

	MNIST	CIFAR-100	ImageNet
# High influence pairs	35	1015	1641
# Unique test examples selected	33	888	1462
# Test examples influenced by a single training example	31	774	1298

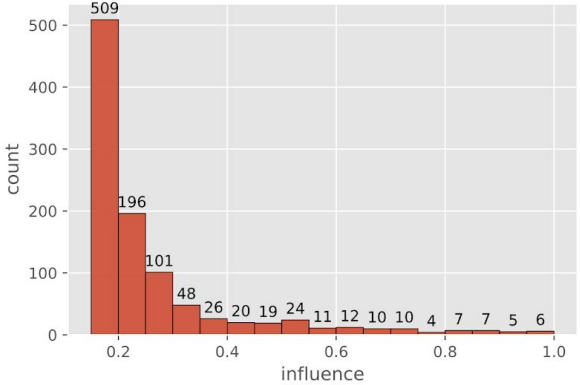
Removing the 964 unique training examples in these pairs on CIFAR-100 reduces the test accuracy by 2.46 ± 0.36%, which is comparable to the effect of removing 11,000 random examples. Essentially all of that effect on the accuracy comes from the drop in accuracy on the test examples in the high-influence pairs from 72.1 ± 1.3% to 45.4 ± 1.4%.

The benefits of memorized examples are fully captured by their high influence on individual test examples.

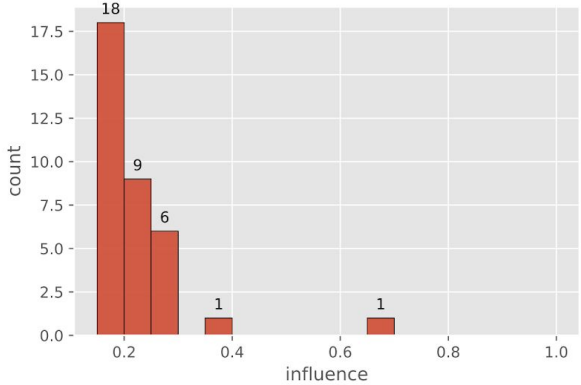
Histogram of Influences in High-influence Pairs



(a) ImageNet



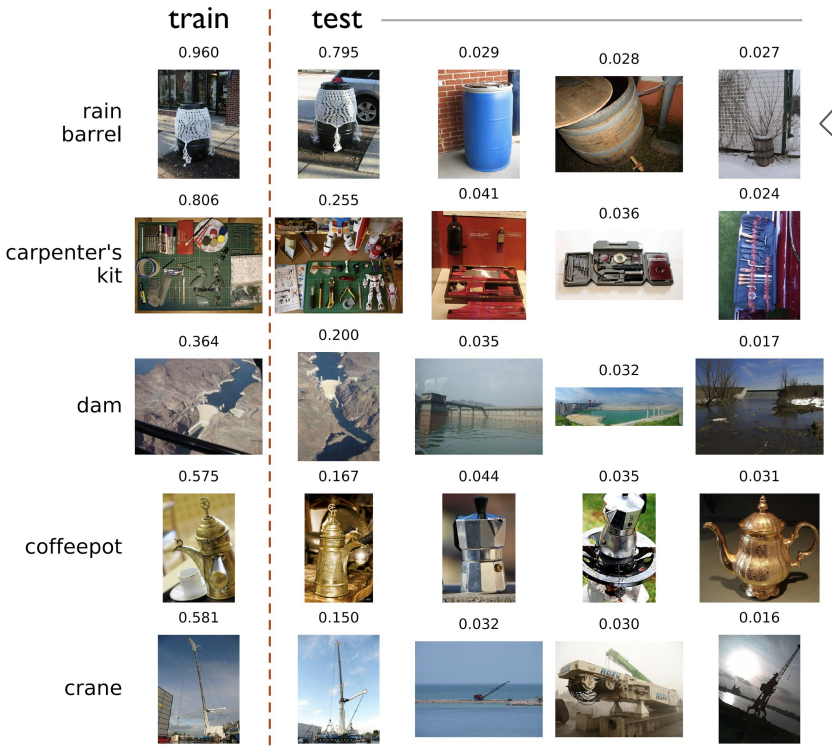
(b) CIFAR-100

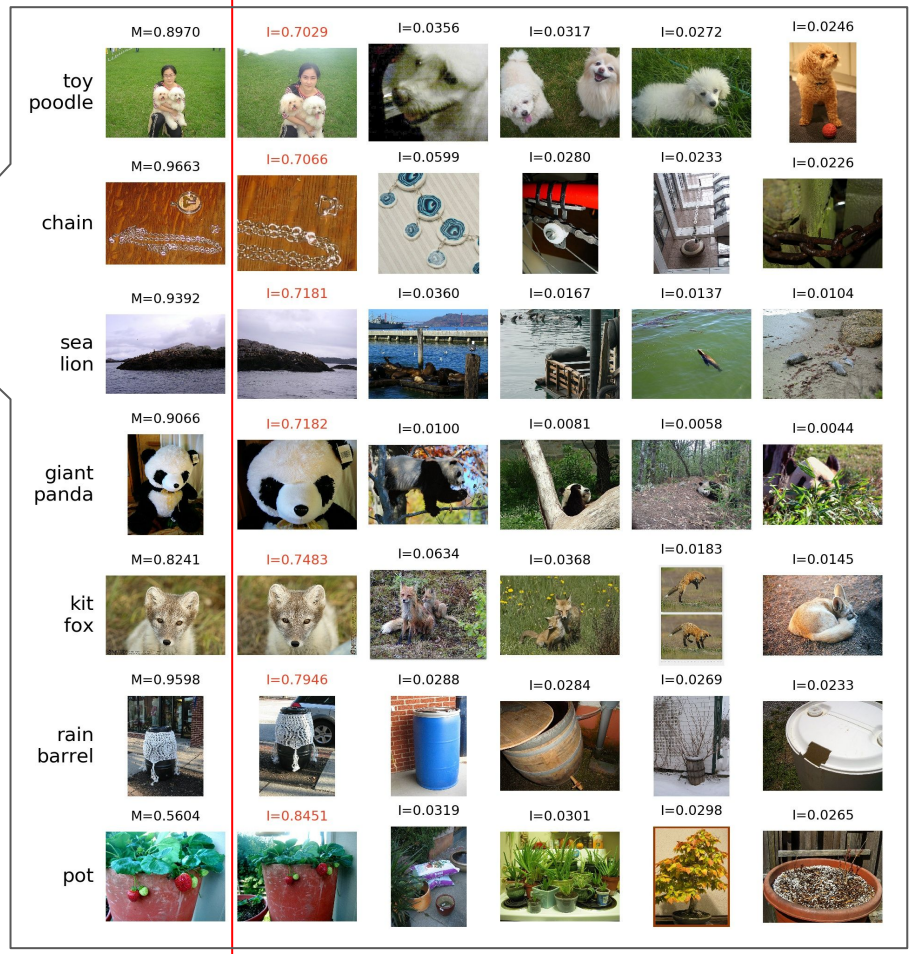


(c) MNIST

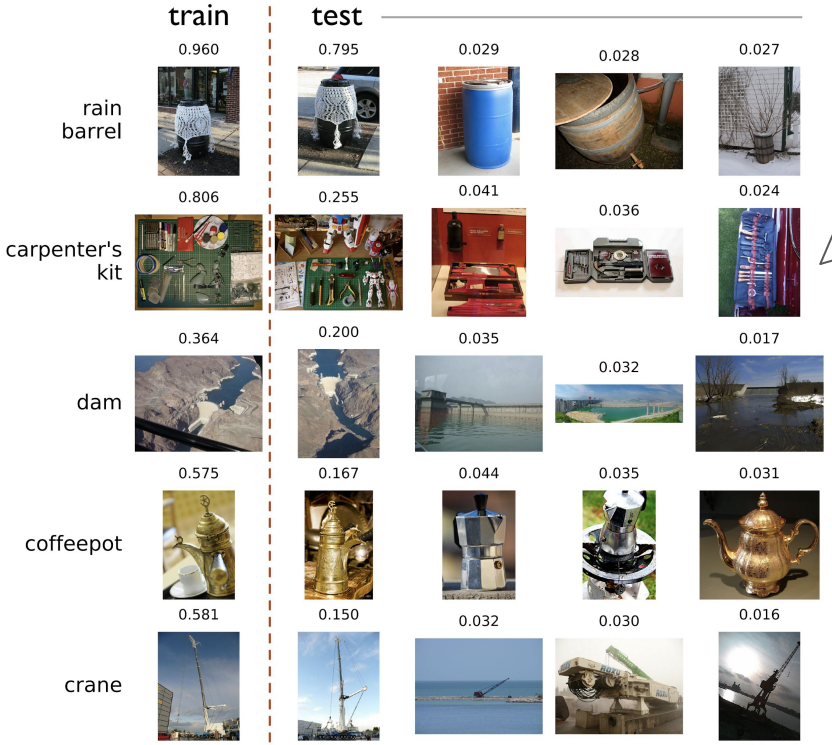
MNIST is a very easy dataset with low diversity in the input examples. There are very few training examples that need to be memorized, or could induce high influence on a test example.

Visualization of High-influence Pairs on ImageNet

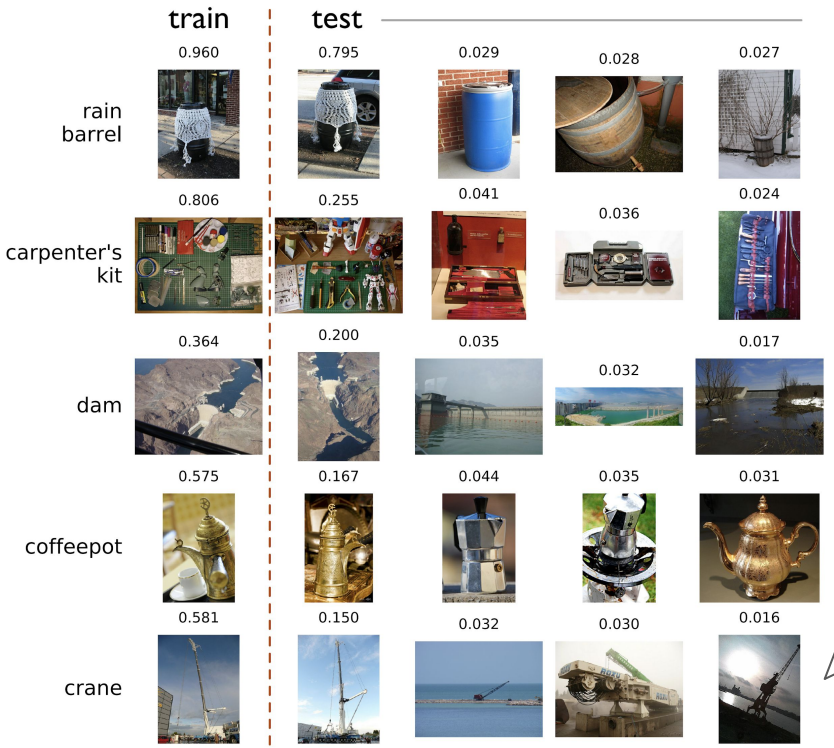


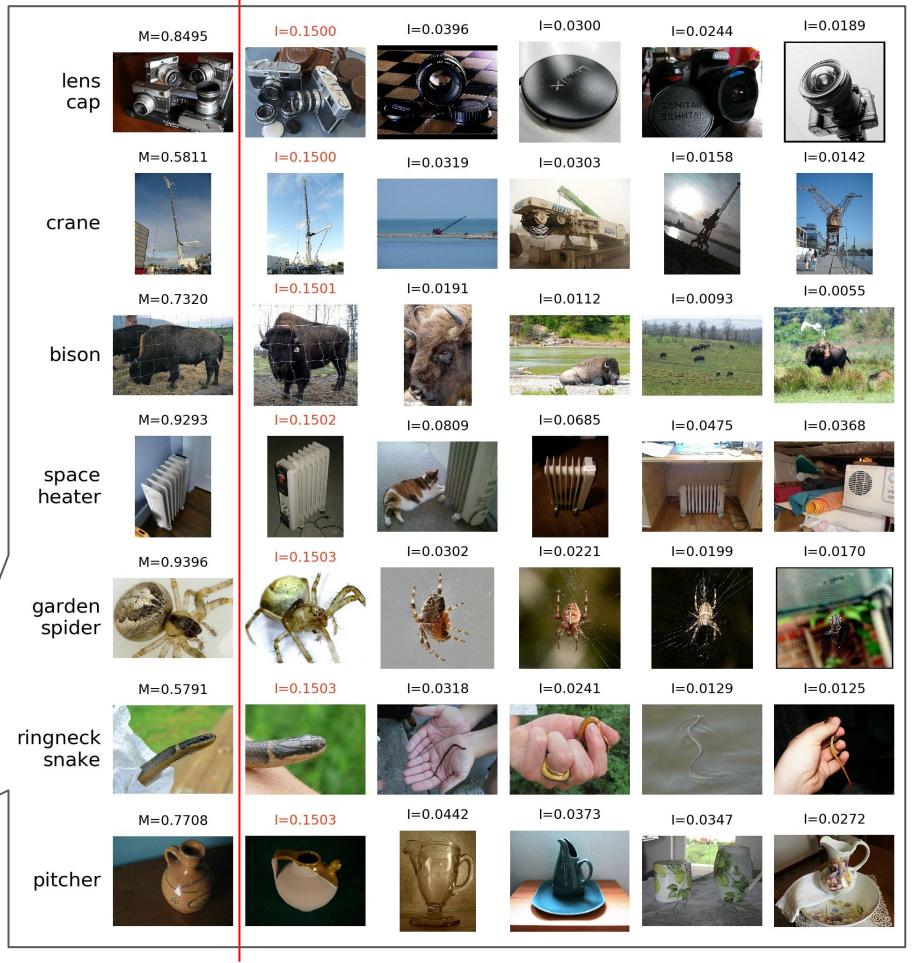


Visualization of High-influence Pairs on ImageNet

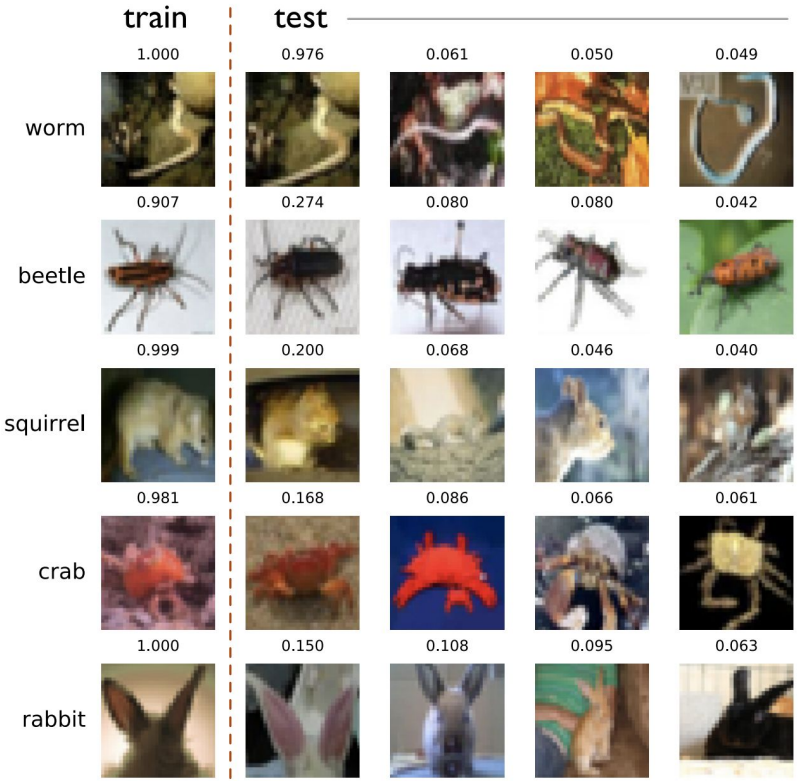


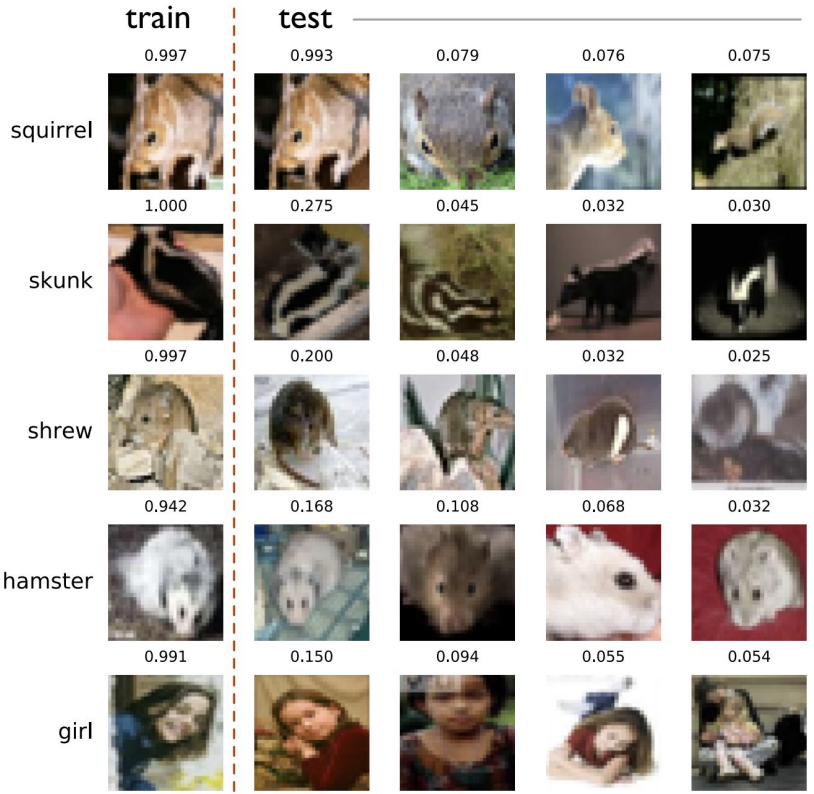
Visualization of High-influence Pairs on ImageNet





Visualization of High Influence Pairs on CIFAR-100

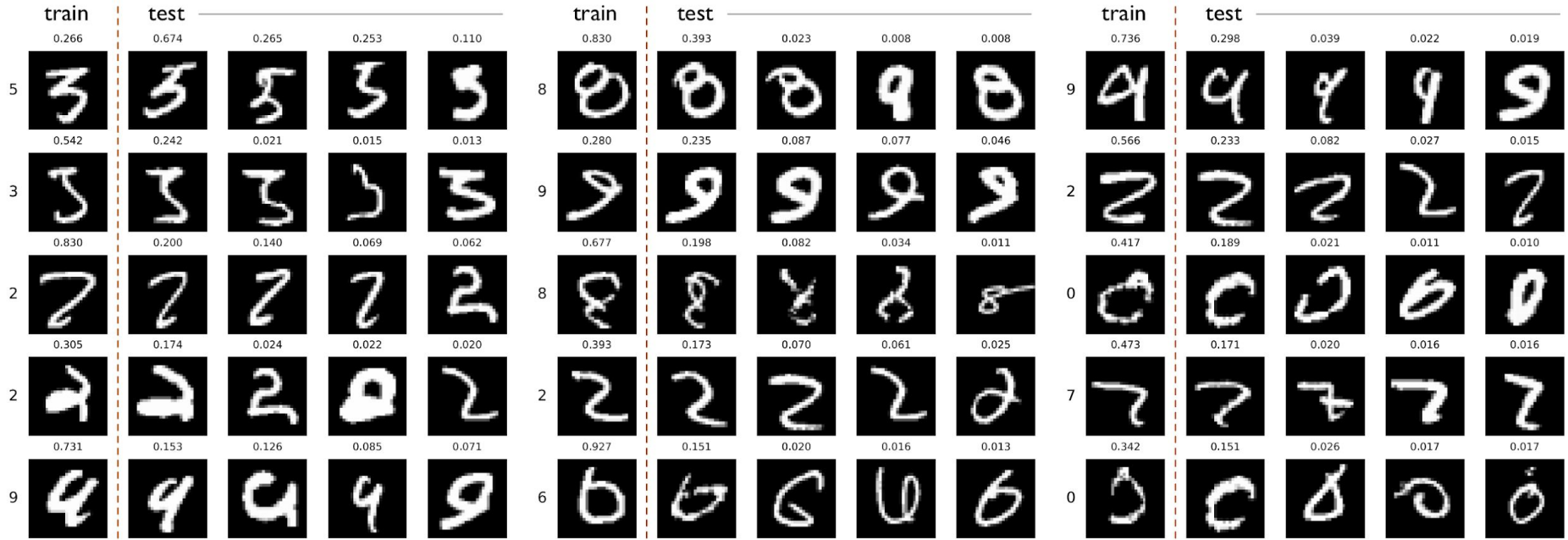




Top-ranking pairs on CIFAR-100 are near duplicates.

High Influence Pairs on MNIST

Less interesting but still show some visual correlation, and some ambiguous / mislabeled examples.



05

Theoretical Characterization

Construction of a simplified learning model with long tail data distribution to demonstrate that optimal performance cannot be achieved without memorization.

The cost of not fitting: a discrete case

To capture the main phenomenon we are interested in, we start by considering a simple and general prediction problem in which the domain does not have any underlying structure (such as the notion of distance). The domains X and Y are discrete, $|X| = N$ and $|Y| = m$ (for concreteness one can think of $X = [N]$ and $Y = [m]$).

Theorem 2.3. *Let π be a frequency prior with a corresponding marginal frequency distribution $\bar{\pi}^N$, and $\mathcal{F}$ be a distribution over Y^X. Then for every learning algorithm $\mathcal{A}$ and every dataset $Z \in (X \times Y)^n$:*

$$\overline{\mathtt{err}}(\pi, \mathcal{F}, \mathcal{A} \mid Z) \geq \mathtt{opt}(\pi, \mathcal{F} \mid Z) + \sum_{\ell \in [n]} \tau_\ell \cdot \mathtt{errn}_Z(\mathcal{A}, \ell),$$

where

$$\tau_\ell := \frac{\mathbf{E}_{\alpha \sim \bar{\pi}^N}\left[\alpha^{\ell+1} \cdot (1-\alpha)^{n-\ell}\right]}{\mathbf{E}_{\alpha \sim \bar{\pi}^N}\left[\alpha^{\ell} \cdot (1-\alpha)^{n-\ell}\right]}.$$

error on examples that appears l times in the training set

theorem from: https://arxiv.org/abs/1906.05271

The cost of not fitting: a discrete case

Theorem 2.3. *Let π be a frequency prior with a corresponding marginal frequency distribution $\bar{\pi}^N$, and $\mathcal{F}$ be a distribution over Y^X. Then for every learning algorithm $\mathcal{A}$ and every dataset $Z \in (X \times Y)^n$:*

$$\overline{\mathtt{err}}(\pi, \mathcal{F}, \mathcal{A} \mid Z) \geq \mathtt{opt}(\pi, \mathcal{F} \mid Z) + \sum_{\ell \in [n]} \tau_\ell \cdot \mathtt{errn}_Z(\mathcal{A}, \ell),$$

error on examples that appears l times in the training set

where

$$\tau_\ell := \frac{\mathbf{E}_{\alpha \sim \bar{\pi}^N}\left[\alpha^{\ell+1} \cdot (1-\alpha)^{n-\ell}\right]}{\mathbf{E}_{\alpha \sim \bar{\pi}^N}\left[\alpha^{\ell} \cdot (1-\alpha)^{n-\ell}\right]}.$$

- tau_1 can be lower bounded, with characterization of the tail distribution.
- The discrete setup can be extended to continuous case of mixture models.

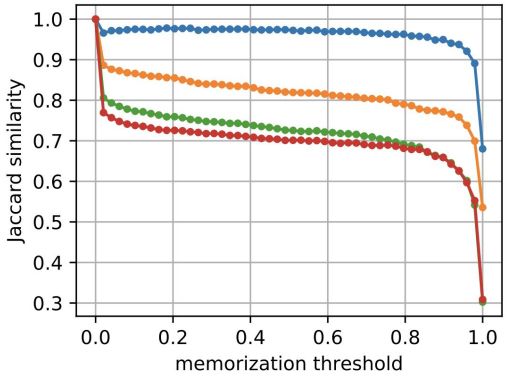
theorem from: https://arxiv.org/abs/1906.05271

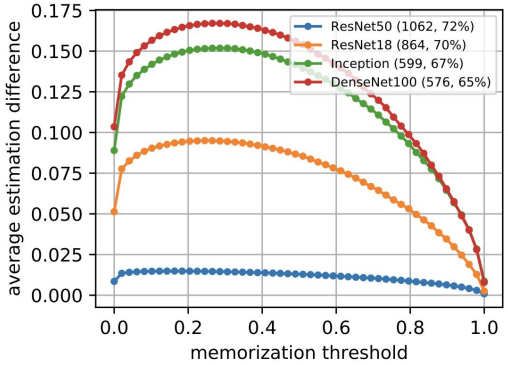
06

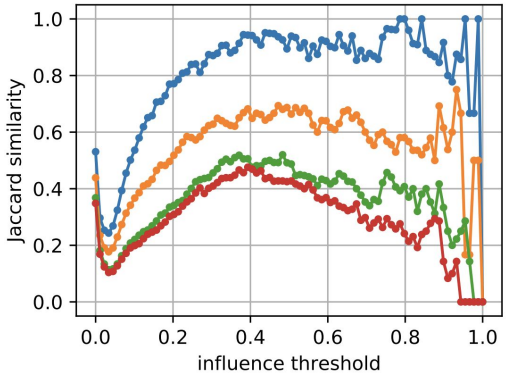
Further Investigations

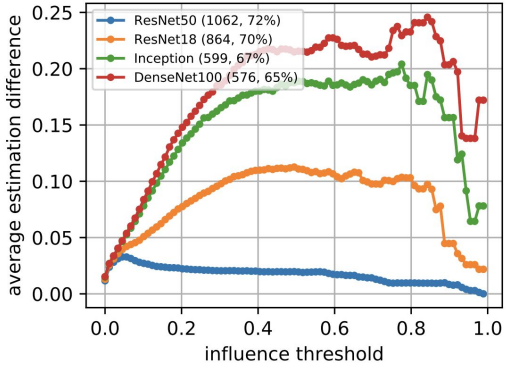
Are influence and memorization consistent across architectures?
In which layer does memorization happen?

Consistency of Estimation Across Architectures









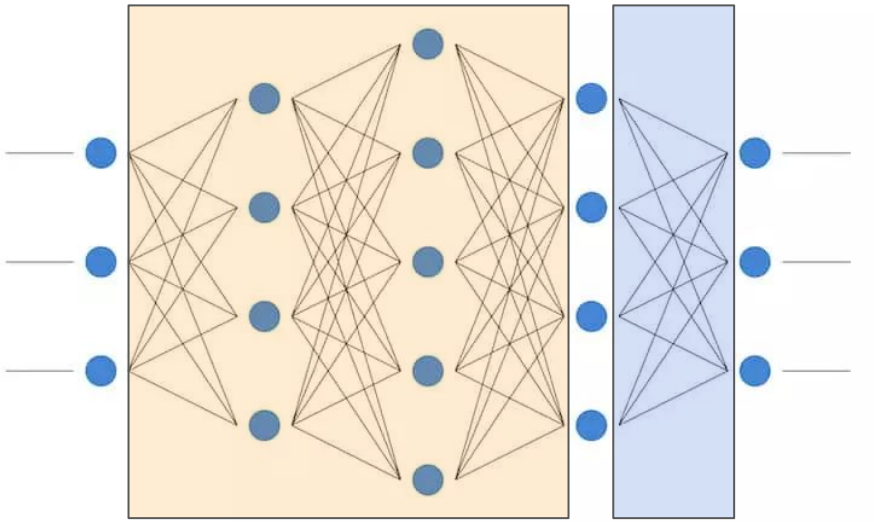
Comparing two estimations:

For each threshold ϑ

- ➔ Select the subset of examples using ϑ by each of the estimation
- ➔ Compare the two selected subsets by Jaccard similarity (Intersection-over-Union)
- ➔ Also compare by the average difference of the estimations on the union of the two selected subsets

Some variations are observed, but overall the estimations are quite consistent across different architectures.

In Which Layer Does Memorization Happen?



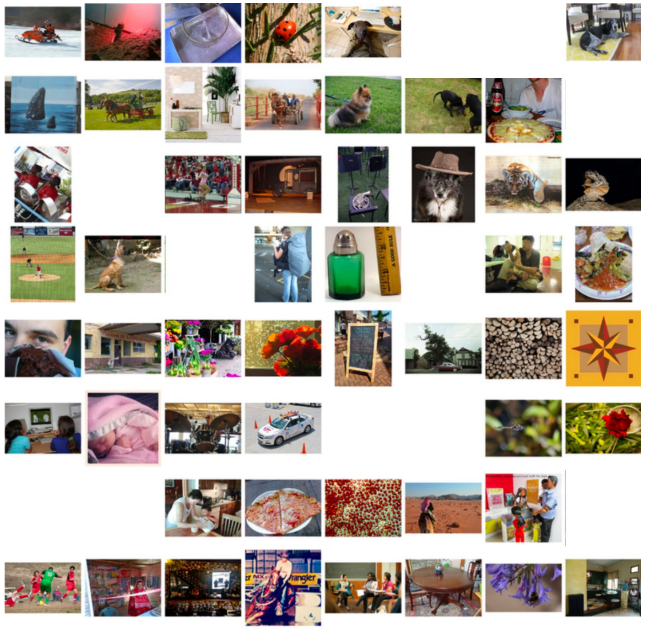
The “body”: computing the representations

The “head”: computing the classification

- The body is considered to be computing feature representations for the inputs -- this feature is commonly finetuned or even directly reused in downstream tasks.
- The head is usually a densely connected linear layer, which could have potentially representation power to memorize the label mapping.

fig source: https://cv-tricks.com/category/convolutional-neural-networks/

Influence Estimation via “Transfer Learning”



Random 70% subset

The feature representation (trained on the full training set) is fixed and re-used, only the head is trained (from fresh random init each run).

Training Pipeline

data-augmentation, regularization,
stable initialization, SoTA activation function,
fancy lr schedule, momentum, preconditioning,
label smoothing, loss tempering, unsupervised aux loss

$\hat{f}$

Repeat 2,000 Times

Influence Estimation via “Transfer Learning”

If memorization indeed happens in the classifier layer, our “transfer-learning” style experiments that trains only linear classifier on pre-trained representations could extract the influence estimation at a much lower cost than training many full ResNet models.

75.9%

ResNet50 trained on CIFAR-100 training set

72.3 ± 0.3%

4k **ResNet50** trained on 70% of CIFAR-100 training set

75.8 ± 0.1%

4k **Linear Models** trained on 70% of CIFAR-100 training set with pre-trained features

Influence Estimation via “Transfer Learning”

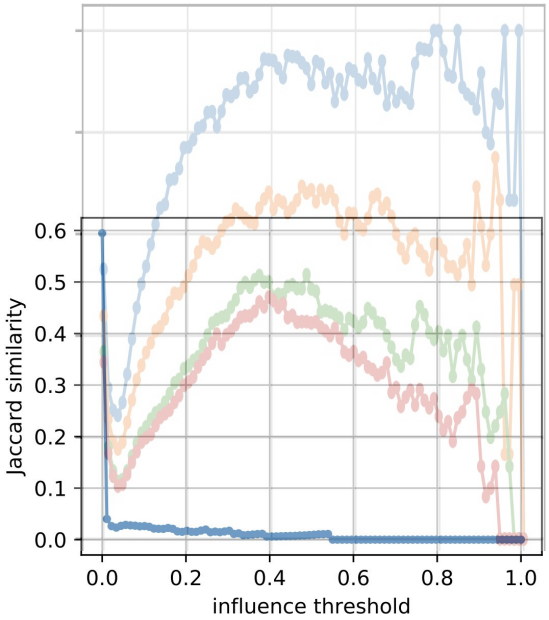
Number of training examples with memorization estimates ≥ 0.25:

18,099

4k **ResNet50** trained on 70% of CIFAR-100 training set

38

4k **Linear Models** trained on 70% of CIFAR-100 training set with pre-trained features



Conclusion

- A formal definition of memorization is given, efficiently estimated and shown to be consistent with high level intuition.
- Memorized examples are useful for test performance.
- The benefits of memorized examples are explained by strong influence on individual test examples.
- The observations verified the theoretical model of learning with long-tailed distribution [Fel19] in real data.

More visualizations and pre-computed influence scores for downloads: https://pluskid.github.io/influence-memorization/