# What Neural Networks Memorize and Why: Discovering the Long Tail via Influence Estimation

В общем случае, когда мы обучаем модель, она очень хорошо обучается на тренировочной выборке (достигает высокого качества), даже если качество на тесте и валидации оставляет желать лучшего. Причем модель в таком случае правильно классифицирует все, даже те объекты, которые изначально были представлены с ошибочной меткой. Единственный способ для модели правильно предсказывать класс, который выставлен неправильно - просто запомнить его.

В общем случае generalization error можно оценить с помощью суммы generalization gap (разница в качестве на трейне и тесте) и предполагаемой имперической ошибки. Запоминание выбросов не улучшает способность модели к обобщению, поэтому, как правило, мы пытаемся настроить гиперпараметры таким образом, чтобы модель запоминала как можно меньше данных.

Таким образом, запоминание считается злом и противоположностью к обобщению

## Почему же все-таки модели запоминают данные?

Теория гласит, что, если в данных значительная часть объектов - это объекты, которые достаточно сильно отличаются от основной массы (out of distribution), то запоминание необходимо для того, чтобы приблизиться к оптимуму задачи

А поскольку полезные выбросы часто неотличимы от бесполезных, то приходится запоминать все подряд

# Memorization value

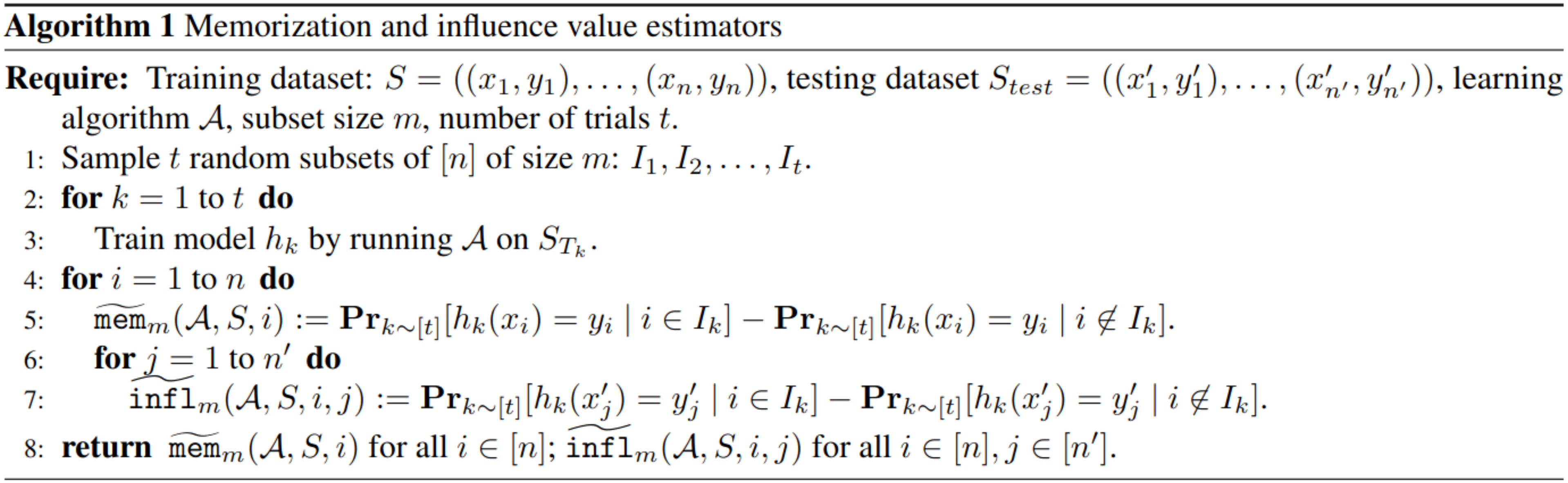
To make it more concrete we recall the definition of label memorization from [Fel19]. For a training algorithm $\mathcal{A}$ operating on a dataset $S = ((x_1, y_1), \ldots, (x_n, y_n))$ the amount of label memorization by $\mathcal{A}$ on example $(x_i, y_i) \in S$ is defined as

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

The primary issue with this definition is that estimating memorization values with standard deviation of $\sigma$ requires running $\mathcal{A}(S^{\setminus i})$ on the order of $1/\sigma^2$ times for every example. As a result, this approach requires $\Omega(n/\sigma^2)$ training runs which translates into millions of training runs needed to achieve $\sigma < 0.1$ on a dataset with $n$ =50,000 examples.

## Influence value

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


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

m=0.7n; t=2000 for ImageNet and 4000 for MNIST/CIFAR-100

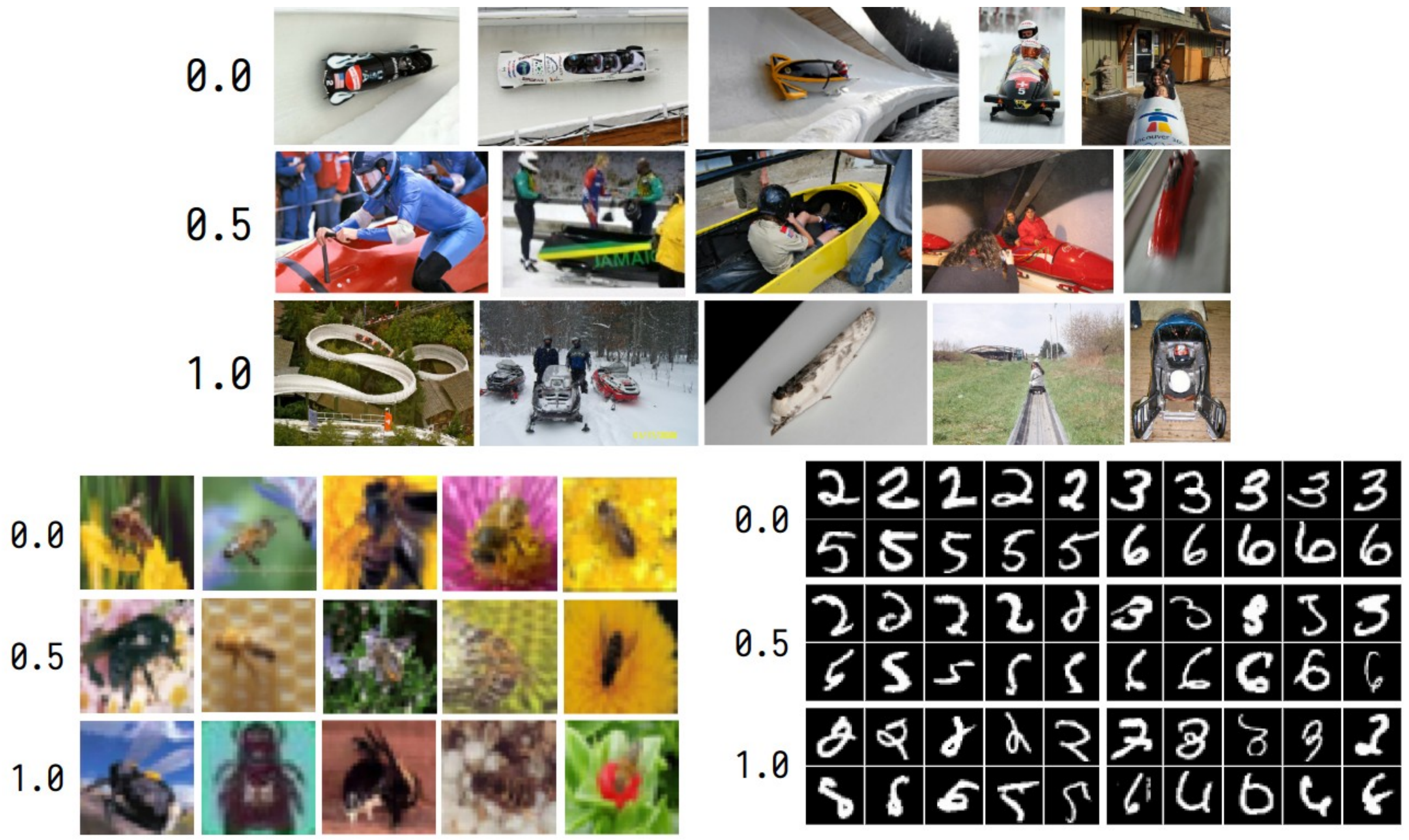
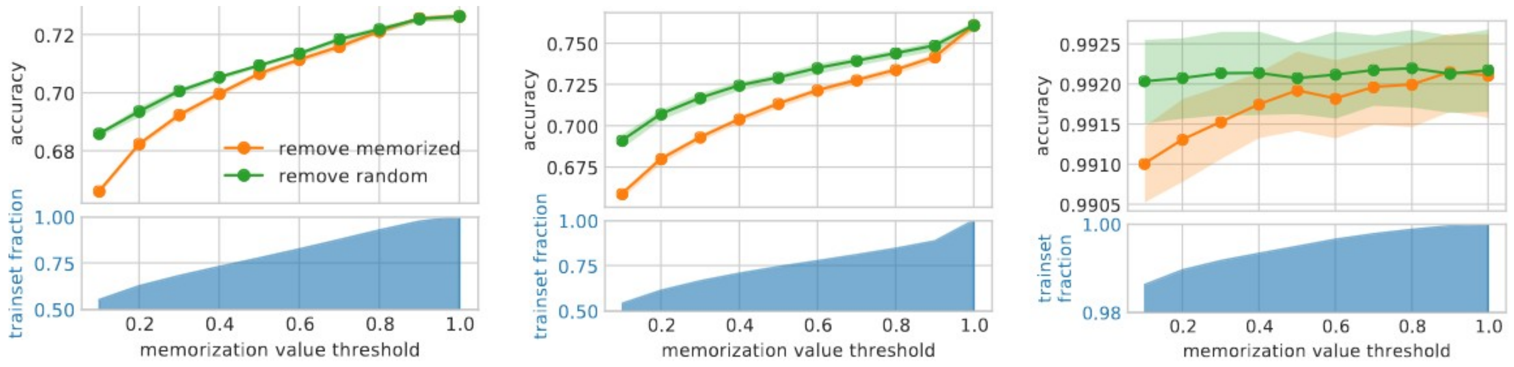


Figure 1: Examples of memorization values from ImageNet class "bobsled" (top), CIFAR-100 class "bee" (bottom left) and MNIST class 2, 3, 5, 6 (bottom right).

## А что если убрать запомненные объекты?



(a) ImageNet (b) CIFAR-100 (c) MNIST

Figure 2: 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.

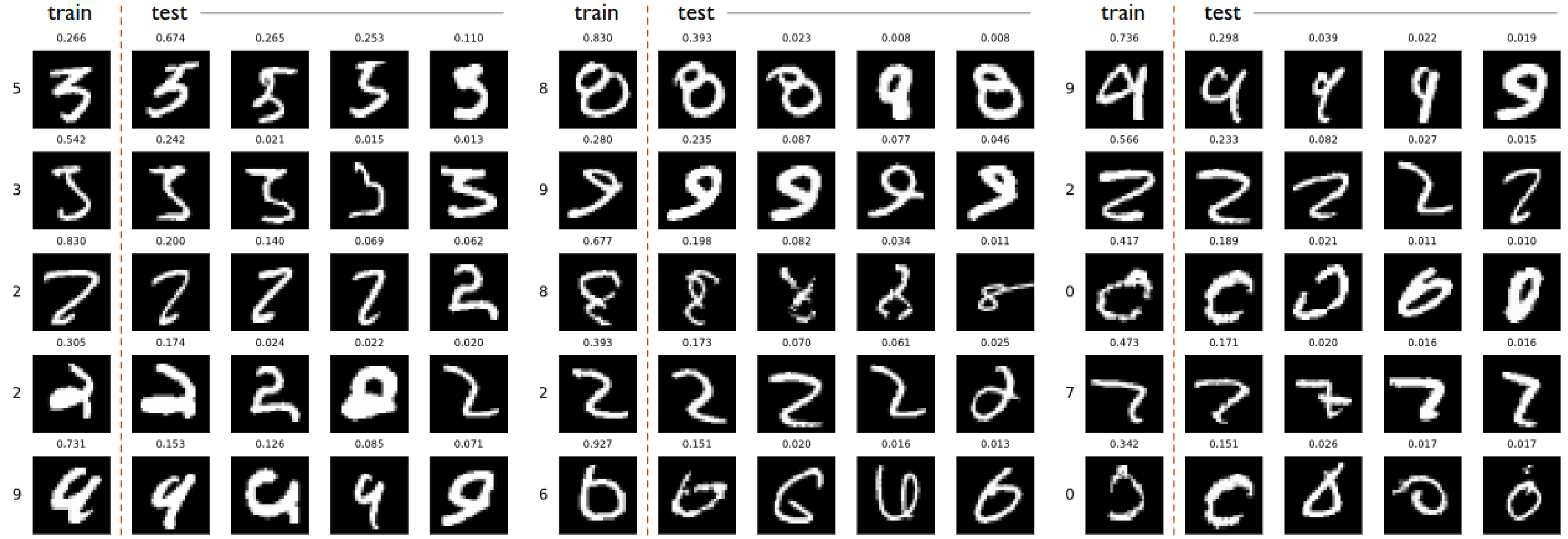


Figure 4: Examples of influence estimates for memorized examples from MNIST. Left column is the memorized examples in the training set and their memorization estimates (above). For each training example 4 most influenced examples in the test set are given together with the influence estimates (above each image).

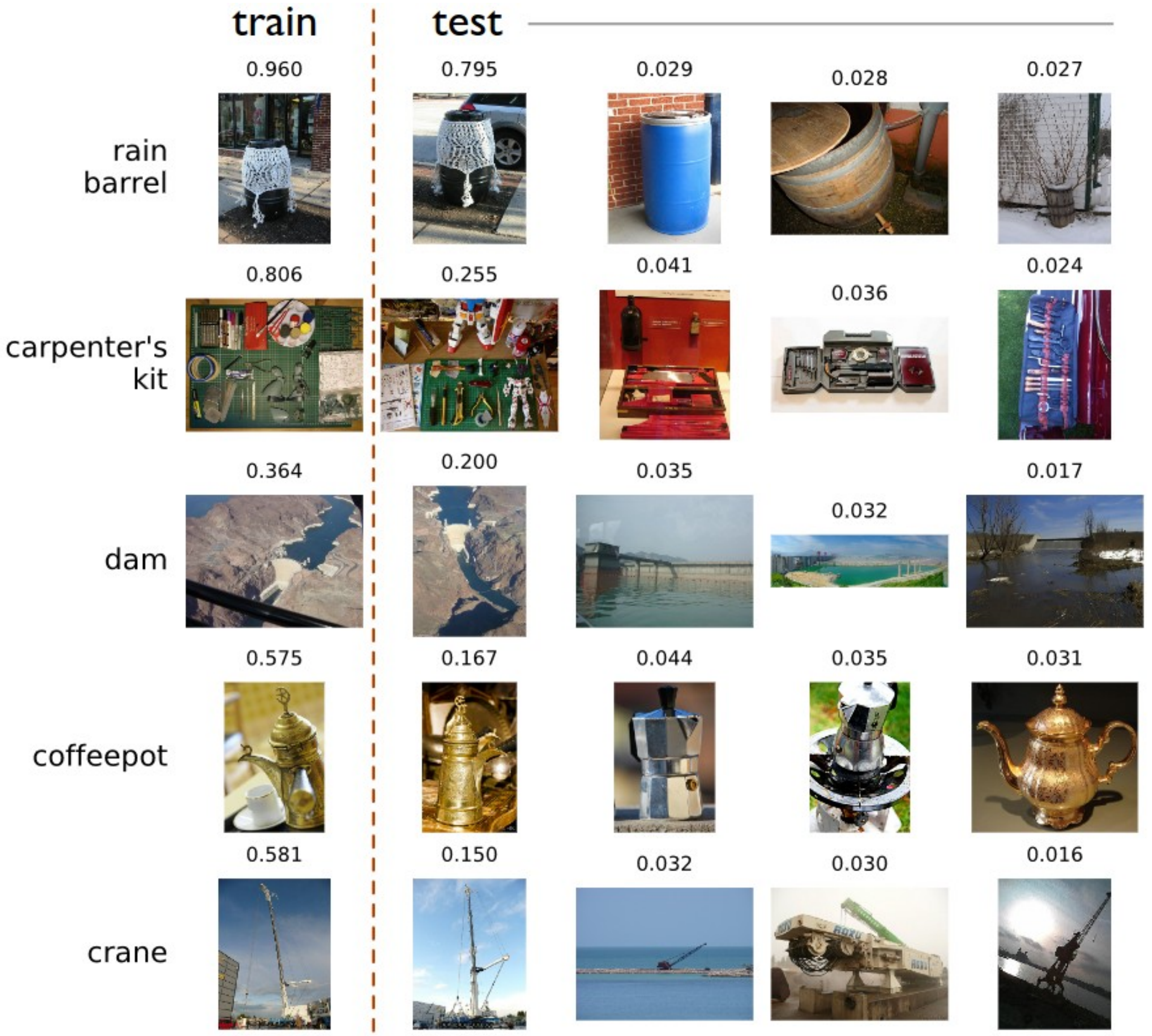


Figure 6: Additional examples of influence estimates for memorized examples from ImageNet. Format is as in Fig. 4.

## Effect on different architectures

$$I_{\texttt{mem}}(\theta_{\texttt{mem}}) := \{i : \widetilde{\texttt{mem}}_m(\mathcal{A}, S, i) \geq \theta_{\texttt{mem}}\},\ I'_{\texttt{mem}}(\theta_{\texttt{mem}}) := \{i : \widetilde{\texttt{mem}}_m(\mathcal{A}', S, i) \geq \theta_{\texttt{mem}}\}, \text{ then}$$

$$J_{\texttt{mem}}(\theta_{\texttt{mem}}) := \frac{|I_{\texttt{mem}}(\theta_{\texttt{mem}}) \cap I'_{\texttt{mem}}(\theta_{\texttt{mem}})|}{|I_{\texttt{mem}}(\theta_{\texttt{mem}}) \cup I'_{\texttt{mem}}(\theta_{\texttt{mem}})|}$$

$$D_{\texttt{mem}}(\theta_{\texttt{mem}}) := \text{mean}_{i \in I_{\texttt{mem}}(\theta_{\texttt{mem}}) \cup I'_{\texttt{mem}}(\theta_{\texttt{mem}})} \left|\widetilde{\texttt{mem}}_m(\mathcal{A}, S, i) - \widetilde{\texttt{mem}}_m(\mathcal{A}', S, i)\right|$$

$$I_{\texttt{infl}}(\theta_{\texttt{infl}}) := \{(i, j) : \widetilde{\texttt{infl}}_m(\mathcal{A}, S, i, j) \geq \theta_{\texttt{infl}}, \widetilde{\texttt{mem}}_m(\mathcal{A}, S, i) \geq 0.25\}$$

$$J_{\texttt{infl}}(\theta_{\texttt{infl}}) := \frac{|I_{\texttt{infl}}(\theta_{\texttt{infl}}) \cap I'_{\texttt{infl}}(\theta_{\texttt{infl}})|}{|I_{\texttt{infl}}(\theta_{\texttt{infl}}) \cup I'_{\texttt{infl}}(\theta_{\texttt{infl}})|}.$$

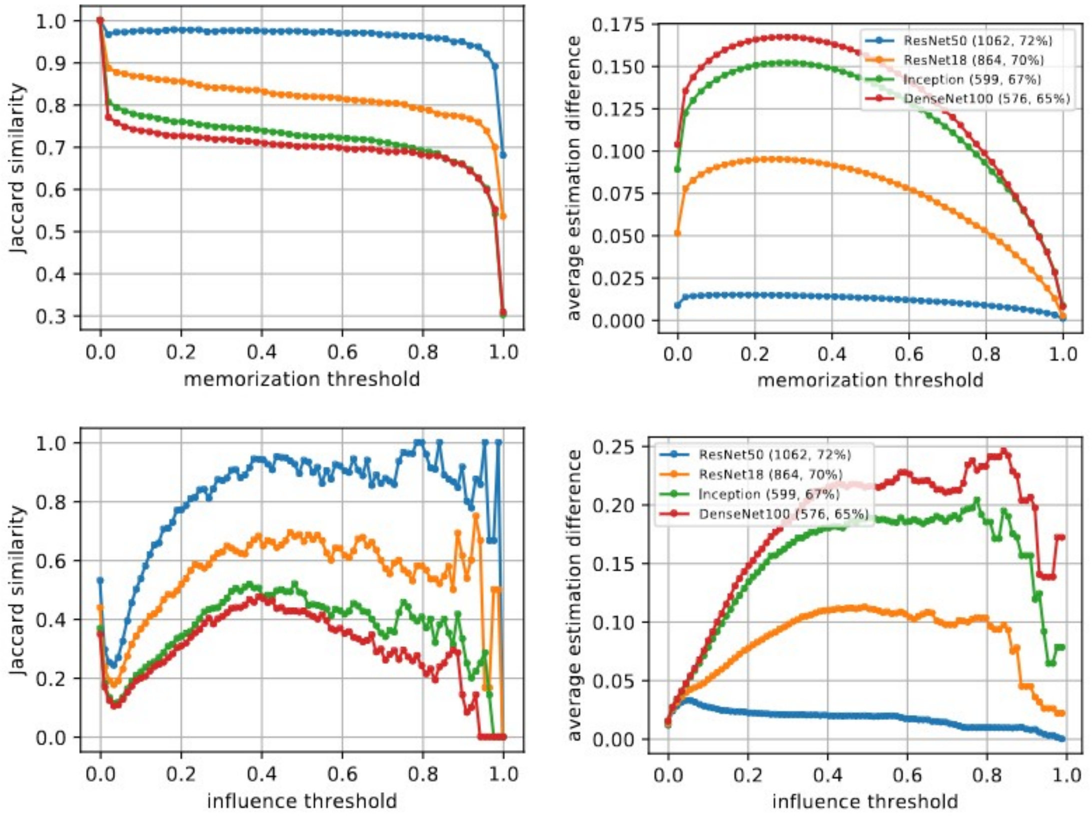
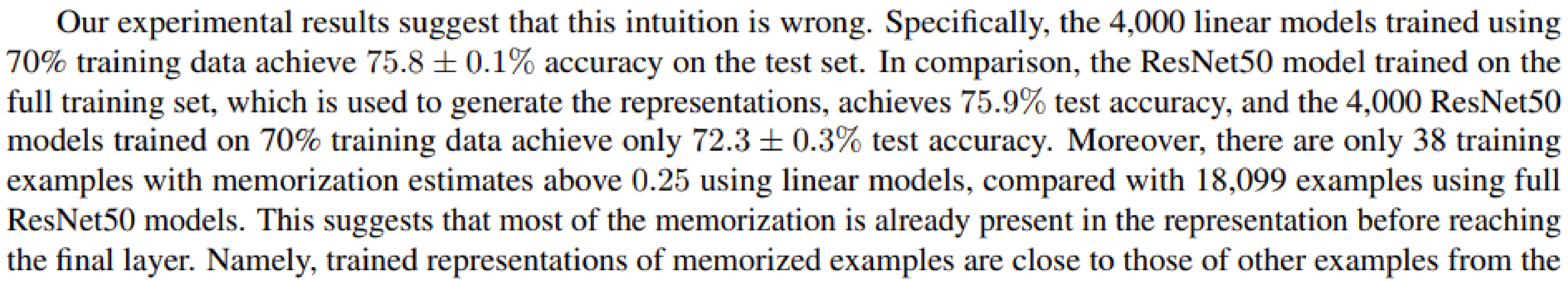


Figure 7: Consistency of the estimation of memorization (top) and influence (bottom) across different architectures on CIFAR-100. In the average estimation difference we plot $D_{\mathtt{mem}}(\theta_{\mathtt{mem}})$ and $D_{\mathtt{infl}}(\theta_{\mathtt{infl}})$. Jaccard similarity plots are for $J_{\mathtt{mem}}(\theta_{\mathtt{mem}})$ and $J_{\mathtt{infl}}(\theta_{\mathtt{infl}})$. All the architectures are compared to ResNet50 — with the "ResNet50" entry being comparison between two independent runs of the same architecture. The numbers in the legend indicate the number of high-influence pairs selected by each architecture according to $\theta_{\mathtt{infl}} = 0.15$ and $\theta_{\mathtt{mem}} = 0.25$, and the average test accuracy (with 70% training set), respectively.

## Влияет ли последний слой на запоминание?



Our experimental results suggest that this intuition is wrong. Specifically, the 4,000 linear models trained using 70% training data achieve $75.8 \pm 0.1\%$ accuracy on the test set. In comparison, the ResNet50 model trained on the full training set, which is used to generate the representations, achieves 75.9% test accuracy, and the 4,000 ResNet50 models trained on 70% training data achieve only $72.3 \pm 0.3\%$ test accuracy. Moreover, there are only 38 training examples with memorization estimates above 0.25 using linear models, compared with 18,099 examples using full ResNet50 models. This suggests that most of the memorization is already present in the representation before reaching the final layer. Namely, trained representations of memorized examples are close to those of other examples from the