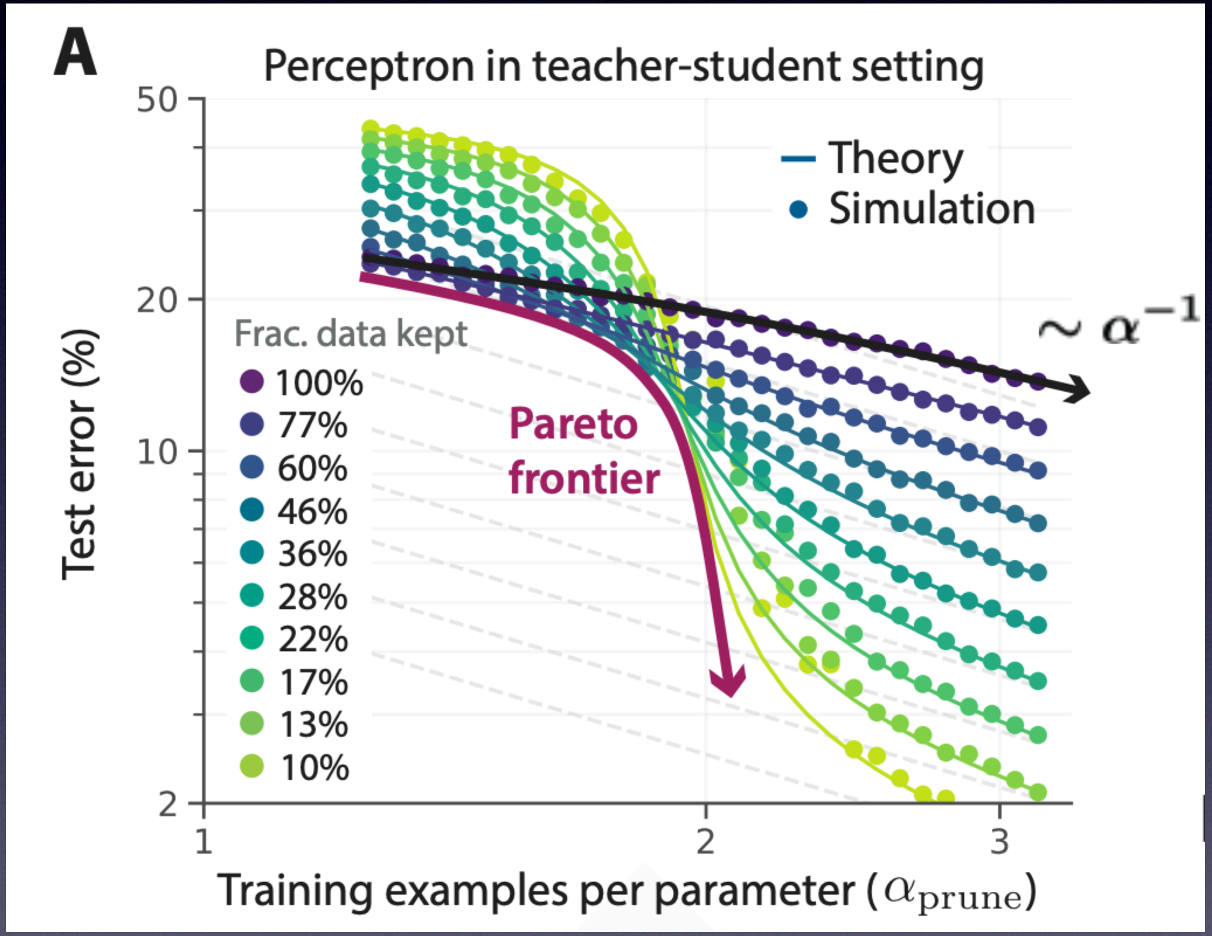
Influence Functions and Data Pruning



From theory to non-convergence and applications

Overview

Part 1: Influence functions

1. What are influence functions?
2. Issues with convergence

Part 2: Towards data efficiency

3. What Neural networks memorise and why?
4. Data pruning in model training

Part 1:
Influence Functions

Data valuation

- Evaluates the samples that have the highest impact on model training
- To each training sample, it associates a score
- Bad samples (e.g. mislabelled images) should have bad scores

Data valuation

- Evaluates the samples that have the highest impact on model training
- To each training sample, it associates a score
- Bad samples (e.g. mislabelled images) should have bad scores

Data efficiency:

Given a model and a test set, which is the best training dataset that maximises accuracy and minimises cost?

Influence functions

Understanding Black-box Predictions via Influence Functions

Pang Wei Koh [1] **Percy Liang** [1]

- First introduced for “robust statistics” in the 70s
- Popularised for neural networks in 2017 by Koh & Lang
- They try to assess the effect of each single training point on the accuracy of a model
- They fall under the umbrella of Explainable AI, with some important remarks

Influence functions: notation

Let's start with the following definitions:

- $z_i = (x_i, y_i)$ is the *i*-th training sample
- θ is the (potentially highly) multi-dimensional array of parameters of the NN
- $L(z, \theta)$ is the loss of the model for point z and parameters θ.

Influence functions: notation

Let's start with the following definitions:

- $z_i = (x_i, y_i)$ is the i-th training sample
- θ is the (potentially highly) multi-dimensional array of parameters of the NN
- $L(z, \theta)$ is the loss of the model for point z and parameters θ.

Model training =>

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta)$$

One way to quantify the effect of training point z on the model is to compare it with

Model training without z =>

$$\hat{\theta}_{-z} = \arg\min_{\theta} \frac{1}{n} \sum_{z_i \neq z} L(z_i, \theta) \ ,$$

Influence functions: naïve definition

We want to quantify the influence of a training sample z on the accuracy of the model (with parameters θ) on a test sample z_{test}. One naïve definition would be:

$$\mathcal{I}(z, z_{\text{test}}) = L(z_{\text{test}}, \hat{\theta}_{-z}) - L(z_{\text{test}}, \hat{\theta})$$

For most practical applications, this approach is not viable because it entails re-training the model several hundred times!

Influence functions: local approximation

When re-training the model is not possible, we need to rely on local analysis.

Let's consider the model trained with the sample *z* having ϵ more weight than the other points

$$\hat{\theta}_{\epsilon,z} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta) + \epsilon L(z, \theta) \ ,$$

As $\epsilon \rightarrow 0$, a first order account of the effect of *z* on z_{test} can be defined as

$$\mathcal{I}_{up}(z, z_{\text{test}}) = - \frac{dL(z_{\text{test}}, \hat{\theta}_{\epsilon,z})}{d\epsilon} \Big|_{\epsilon=0}$$

Influence functions: local approximation

After a few algebraic steps, one finds that the new (local) influence function definition is equal to

$$\mathcal{I}_{up}(z, z_{\text{test}}) = \nabla_\theta L(z_{\text{test}}, \hat{\theta})^\top \, H_{\hat{\theta}}^{-1} \, \nabla_\theta L(z, \hat{\theta})$$

Hessian of the model => $H_{\hat{\theta}} = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta^2 L(z_i, \hat{\theta})$

Influence functions: local approximation

After a few algebraic steps, one finds that the new (local) influence function definition is equal to

$$\mathcal{I}_{up}(z, z_{\text{test}}) = \nabla_\theta L(z_{\text{test}}, \hat{\theta})^\top \, H_{\hat{\theta}}^{-1} \, \nabla_\theta L(z, \hat{\theta})$$

Hessian of the model => $H_{\hat{\theta}} = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta^2 L(z_i, \hat{\theta})$

Important Notes:

- All terms are gradients wrt. θ and can be calculated through backpropagation!
- Calculating the Hessian is a huge problem. H is a big matrix, which also needs to be inverted.

Influence functions: interpretation

Influence values have a **simple interpretation:**

they tell you how much the loss of a model on a test point z_{test} decreases if the point z is given more weight during training

Influence functions: interpretation

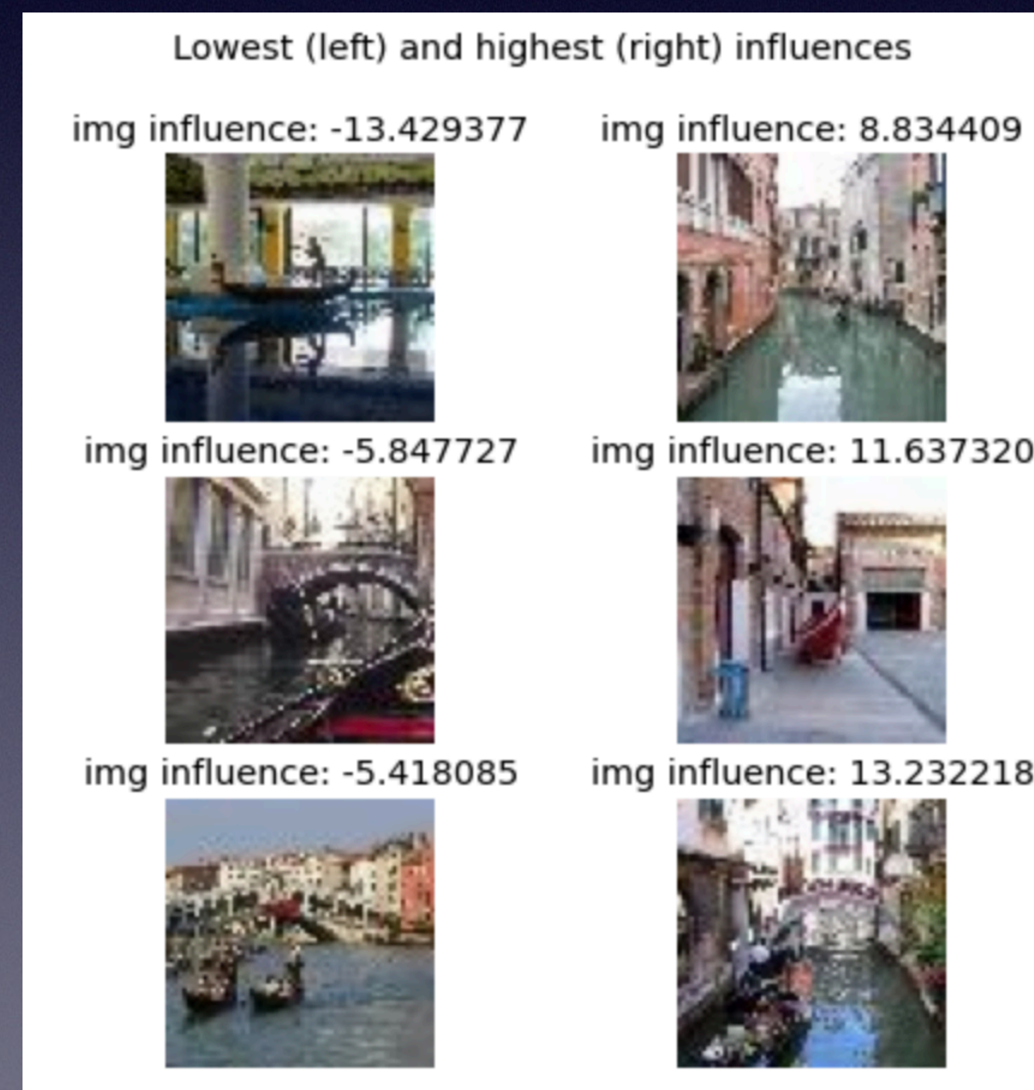
Influence values have a **simple interpretation:**

they tell you how much the loss of a model on a test point z_{test} decreases if the point z is given more weight during training

Let's pick an example from our library pyDVL:
Image classification with Resnet18

z_{test}

Influences



They are only partially explainable

Influence functions: issues

- Inverting the Hessian is often very expensive
- It is also often not well defined => NN models are not convex!

Influence functions: issues

- Inverting the Hessian is often very expensive
- It is also often not well defined => NN models are not convex!

There are several available workarounds:

For inversion
=> use approximate techniques (e.g. conjugate gradient)

For non convexity
=> add a perturbation term

$$H_{\hat{\theta}} + \lambda \mathbb{I}$$

Influence functions: issues

- Inverting the Hessian is often very expensive
- It is also often not well defined => NN models are not convex!

There are several available workarounds:

For inversion
=> use approximate techniques (e.g. conjugate gradient)

For non convexity
=> add a perturbation term

$$H_{\hat{\theta}} + \lambda \mathbb{I}$$

For more details come to our **upcoming seminar**!

Influence Diagnostics under Self-concordance

Jillian Fisher[1] **Lang Liu**[1] **Krishna Pillutla**[1] **Yejin Choi**[1,2] **Zaid Harchaoui**[1]
University of Washington[1] Allen Institute for Artificial Intelligence[2]

Influence functions: issues

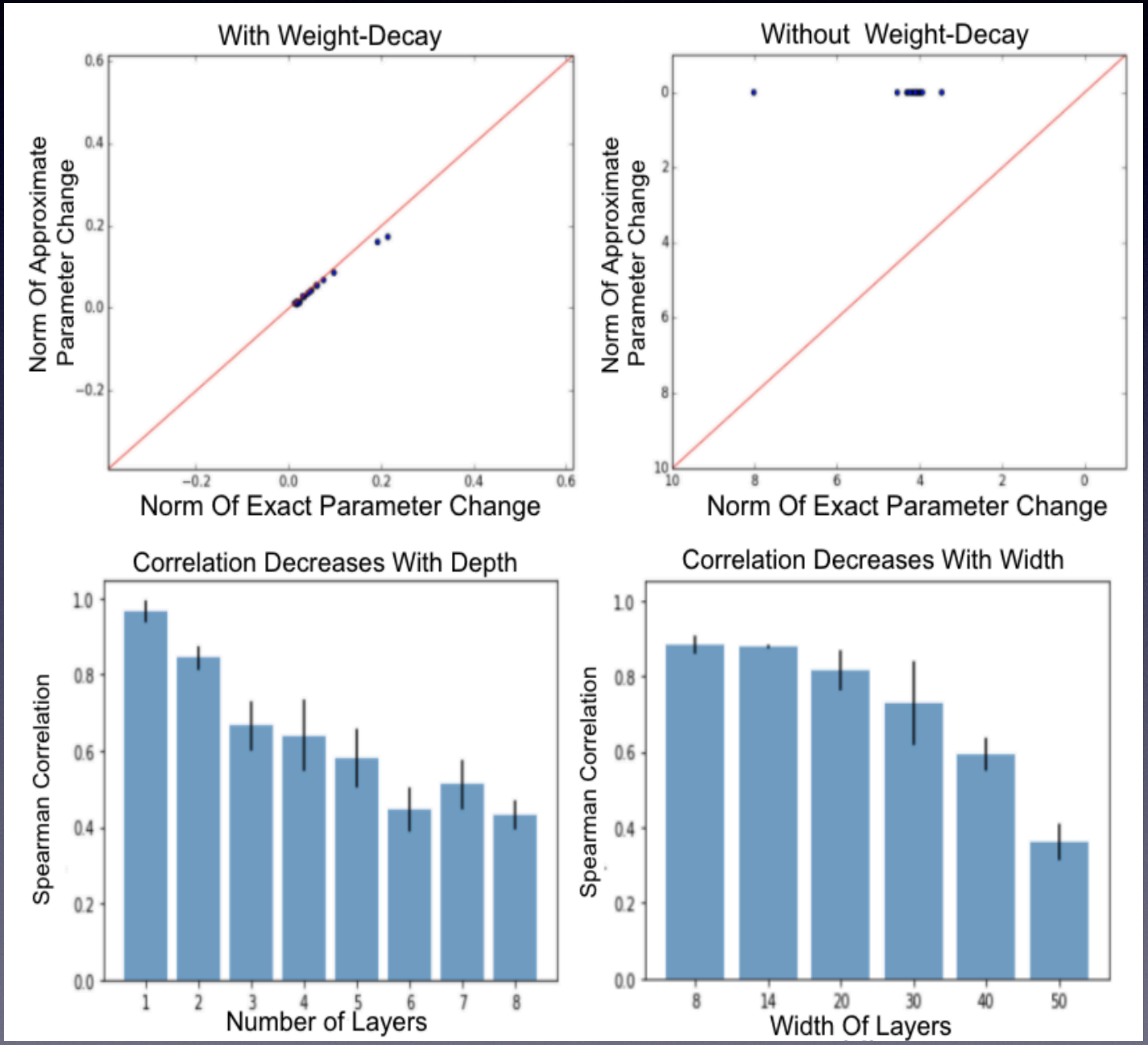
INFLUENCE FUNCTIONS IN DEEP LEARNING ARE FRAGILE

Samyadeep Basu*, Phillip Pope *& Soheil Feizi
Department of Computer Science
University of Maryland, College Park
{sbasu12,pepope,sfeizi}@cs.umd.edu

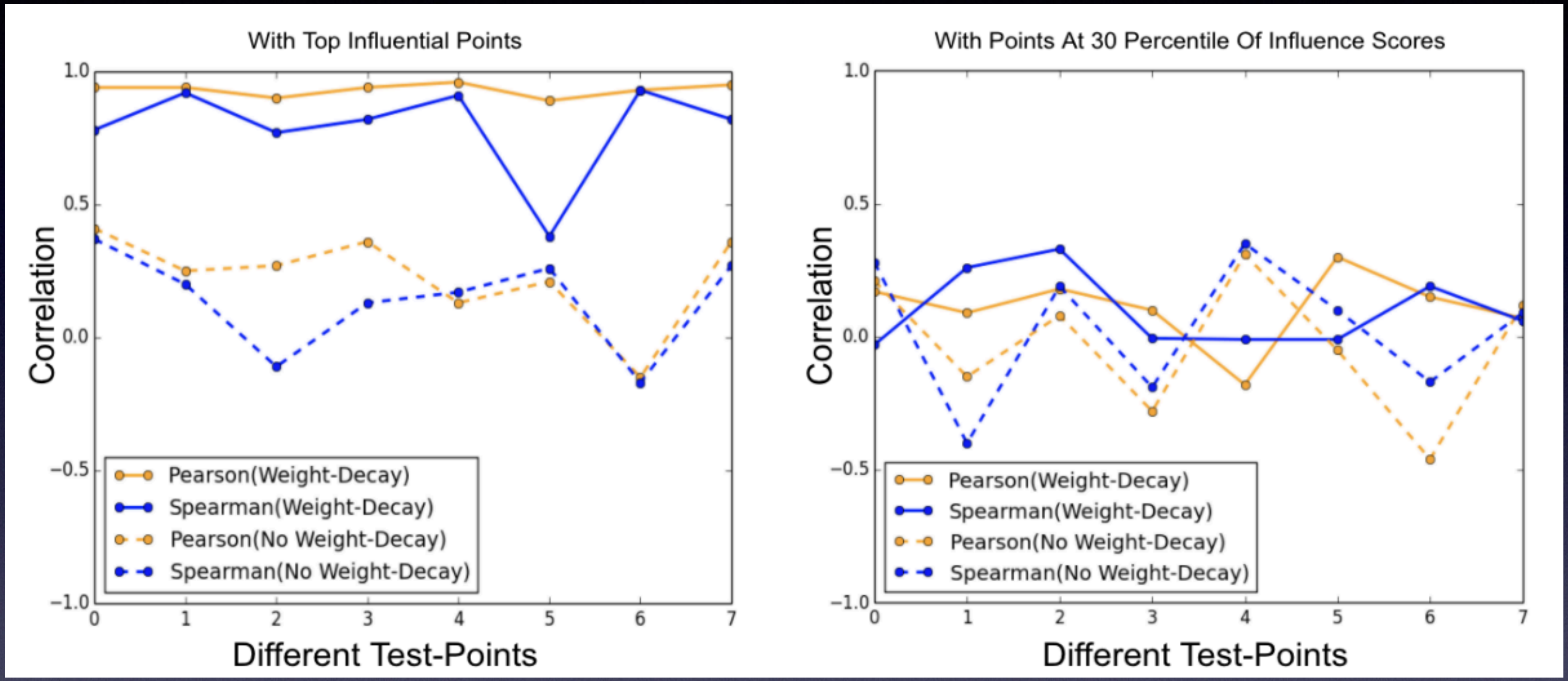
1. How similar is the local approximation to the initial leave-one-out definition?
2. How does it depend on network architecture?

Influence functions: issues

Let's compare the correlation of parameter changes



Influence functions: issues



The approximation is better for highly influential points!

Influence values may not be reliable
influence rankings might be!

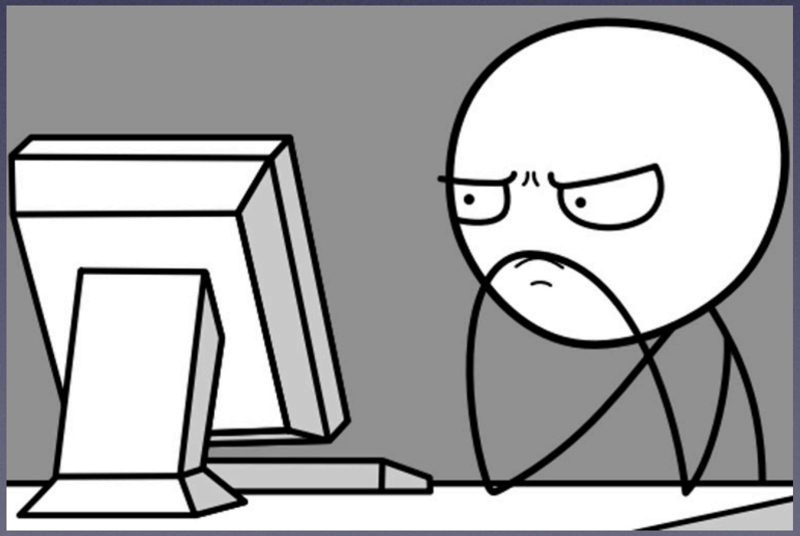
Conclusions to part 1

- Influence functions estimate sample impact on model accuracy
- Their calculation entails several model re-trainings
- Local approximations are more efficient, but still expensive
- With larger models, influence values are more noisy
- A lot of recent work on more efficient computation

Conclusions to part 1

- Influence functions estimate sample impact on model accuracy
- Their calculation entails several model re-trainings
- Local approximations are more efficient, but still expensive
- With larger models, influence values are more noisy
- A lot of recent work on more efficient computation

Ok, but are they actually useful???



Part 2:
Towards Data Efficiency

Memorisation

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

Vitaly Feldman [*, †]
Apple

Chiyuan Zhang[*]
Google Research, Brain Team

Observations:

- Neural networks fit even outliers or mislabelled points
  => They memorise some of the training samples
- Natural data are long-tailed
  => they have a significant fraction of atypical samples

Memorisation

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

Vitaly Feldman * †
Apple

Chiyuan Zhang*
Google Research, Brain Team

Observations:

- Neural networks fit even outliers or mislabelled points
  => They memorise some of the training samples
- Natural data are long-tailed
  => they have a significant fraction of atypical samples

Claim:
Memorisation is essential to reach close-to-optimal generalisation error

Memorisation

For a neural network, the memorisation score can be defined as:

$$mem(z) = L(z, \hat{\theta}_{-z}) - L(z, \hat{\theta})$$

Notice that memorisation <=> self-influence

$$mem(z) = \mathcal{I}(z, z)$$

Memorisation

For a neural network, the memorisation score can be defined as:

$$mem(z) = L(z, \hat{\theta}_{-z}) - L(z, \hat{\theta})$$

Notice that memorisation <=> self-influence

$$mem(z) = \mathcal{I}(z, z)$$

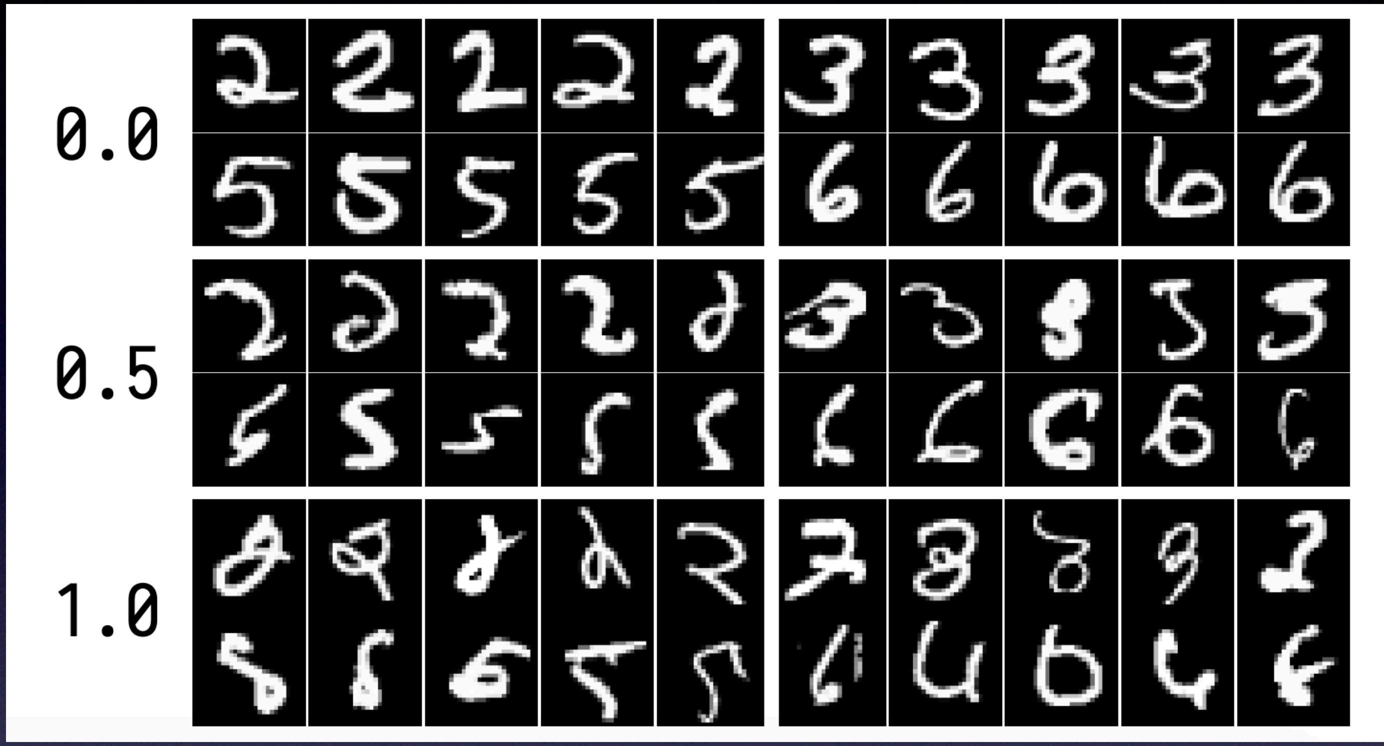
In practice:

1. Take many random subsets of the training set
2. For each, train model for a few epochs
3. Calculate the average loss of the models with and without *z*

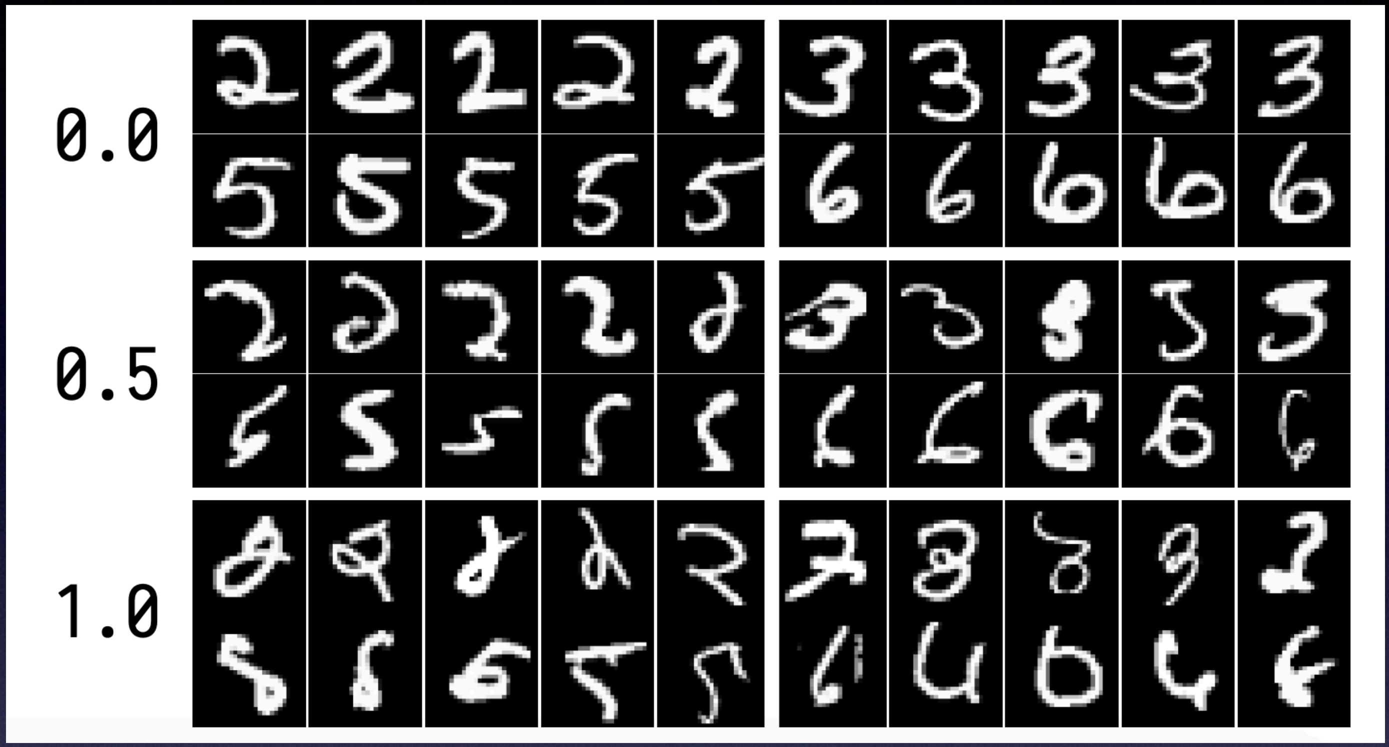
Better than leave-one-out but still very computationally expensive!

Memorisation: results

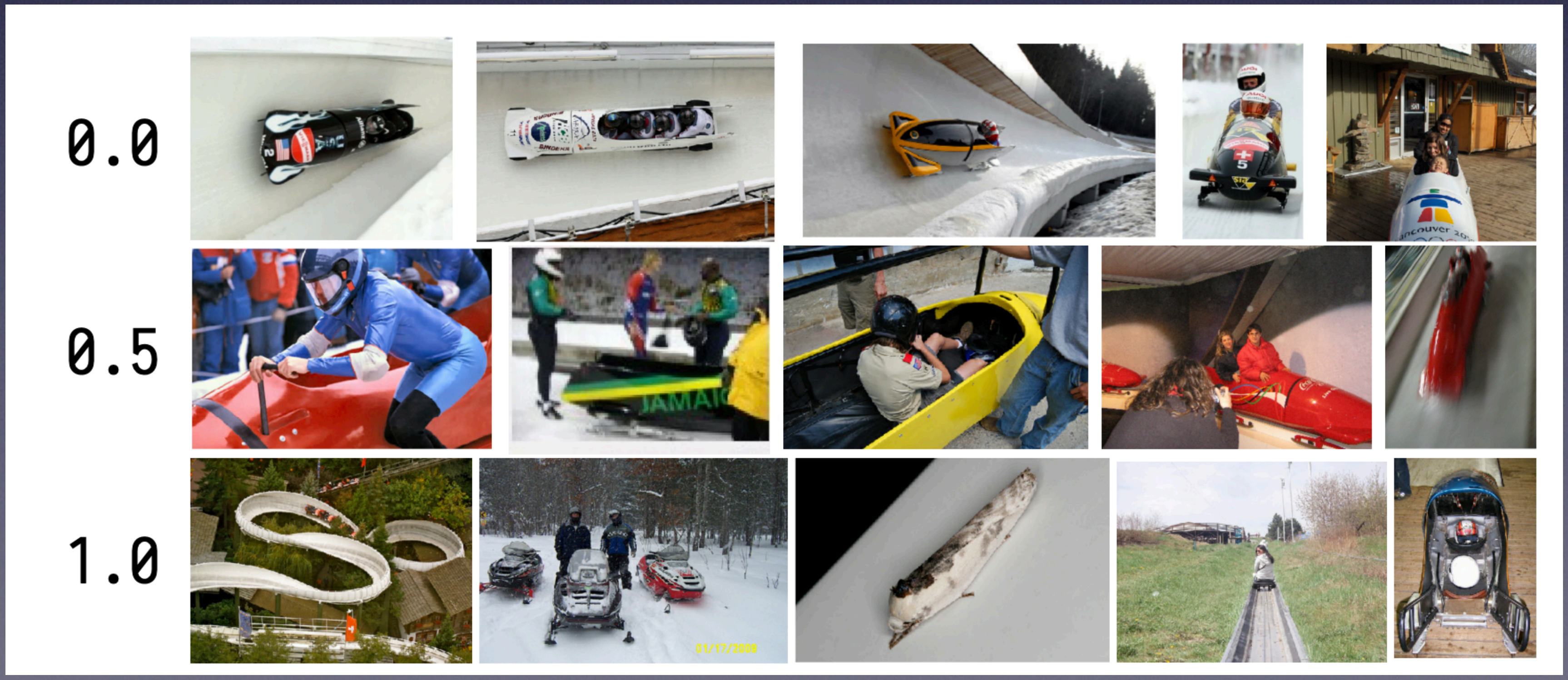
MNIST
memorisation



Memorisation: results

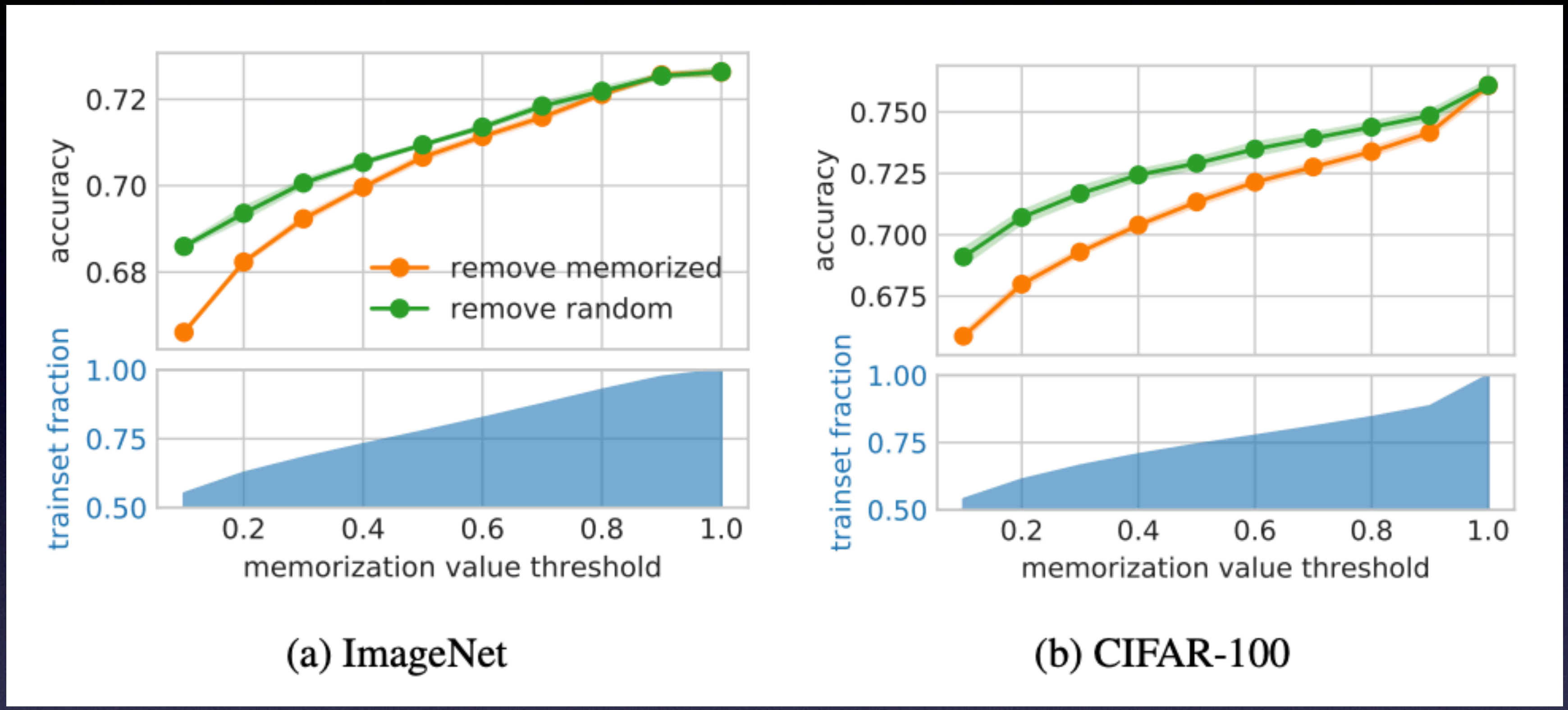


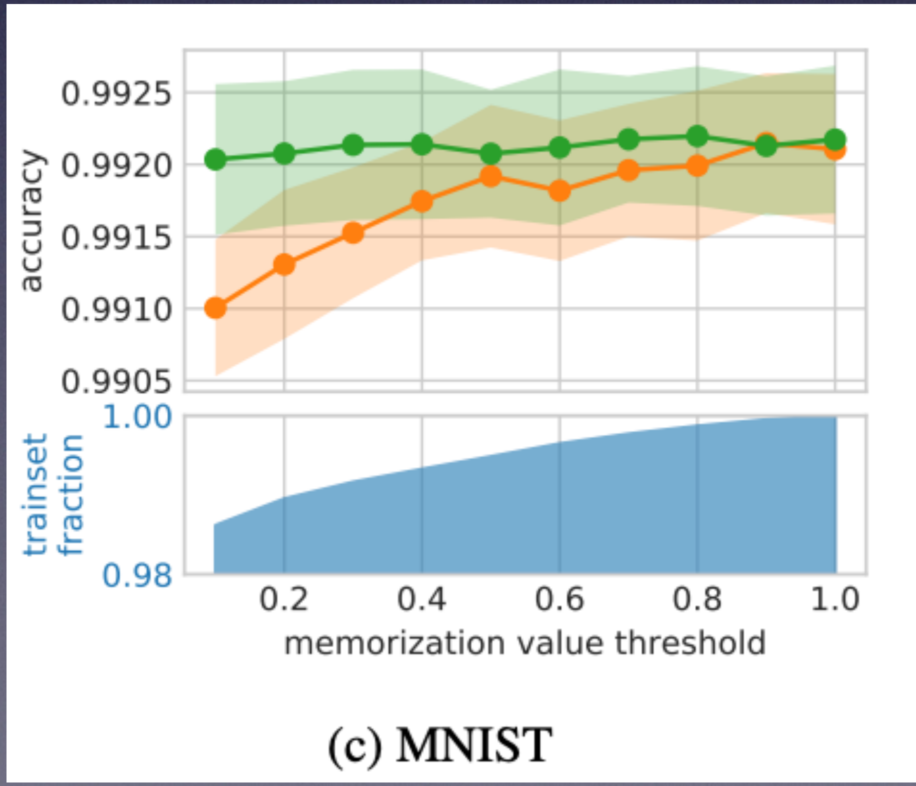
MNIST memorisation



ImageNet memorisation for “bobsled"

Memorisation: results

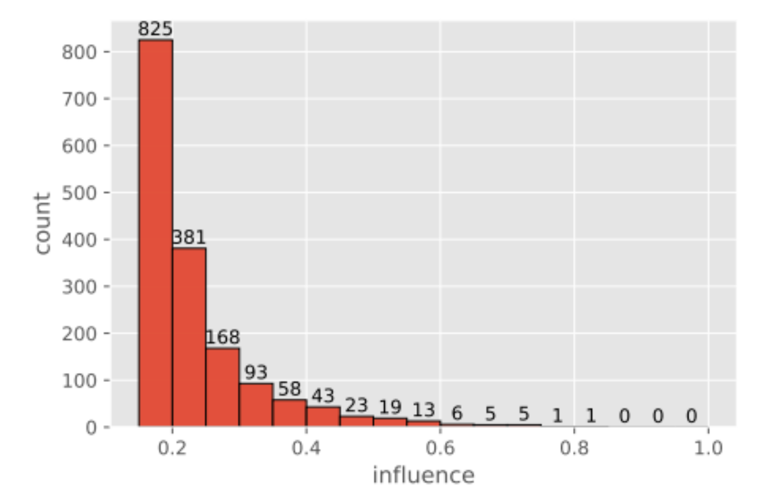




memorisation boosts accuracy!

Memorisation: summary

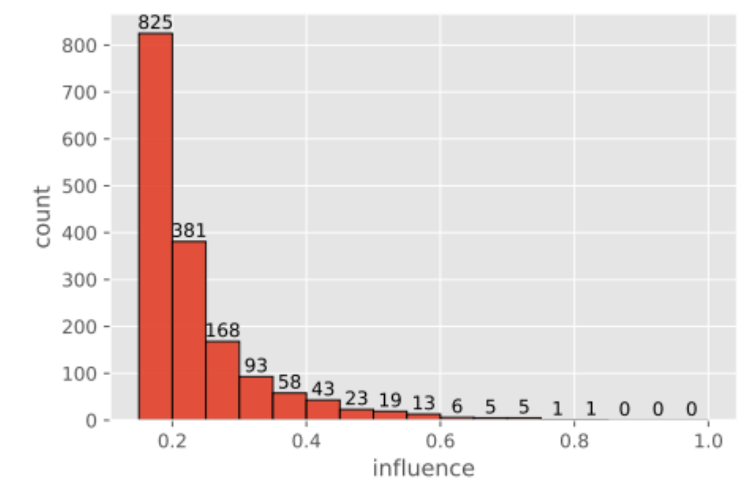
1. Natural datasets are fat-tailed
2. Difficult points are memorised by NNs
3. They have big impact on model accuracy



(a) ImageNet

Memorisation: summary

1. Natural datasets are fat-tailed
2. Difficult points are memorised by NNs
3. They have big impact on model accuracy



(a) ImageNet

Questions:

1. If so many points have negligible memorisation, why don't we just remove them?
2. Which are other good "impact scores"?
3. Is data efficiency something we should care about?

Data Pruning

Beyond neural scaling laws: beating power law scaling via data pruning

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, Ari S. Morcos

Published: 31 Oct 2022, Last Modified: 13 Jan 2023   NeurIPS 2022 Accept   Readers: Everyone   Show Bibtex   Show Revisions

Empirical neural scaling laws
=> test error falls off as a power law of training data, model size or compute

Data Pruning

Beyond neural scaling laws: beating power law scaling via data pruning

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, Ari S. Morcos

Published: 31 Oct 2022, Last Modified: 13 Jan 2023 NeurIPS 2022 Accept Readers: Everyone Show Bibtex Show Revisions

Empirical neural scaling laws
=> test error falls off as a power law of training data, model size or compute

E.g., in LLMs a decrease in cross in cross-entropy loss from 3.4 to 2.8 requires 10x more data

More precisely: $$\mathcal{L} \approx P^{-\nu}$$

- $\mathcal{L}$ is the loss
- P the training set size
- ν typically takes values 0.1- to 0.5

Can we reduce the scaling to exponential?

Data Pruning: analytic results

Would data pruning work? Let's try with a simple toy problem

Student-teacher for perceptron learning:

Given $\mathbf{x}^\mu \in \mathbb{R}^N$ $\mathbf{T} \in \mathbb{R}^N$

The teacher generates the labels: $y^\mu = \mathrm{sign}(\mathbf{T} \cdot \mathbf{x}^\mu)$

The student fits the data to recover the value of $\mathbf{T}$

Data Pruning: analytic results

Would data pruning work? Let's try with a simple toy problem

Student-teacher for perceptron learning:

Given $\mathbf{x}^\mu \in \mathbb{R}^N$ $\mathbf{T} \in \mathbb{R}^N$

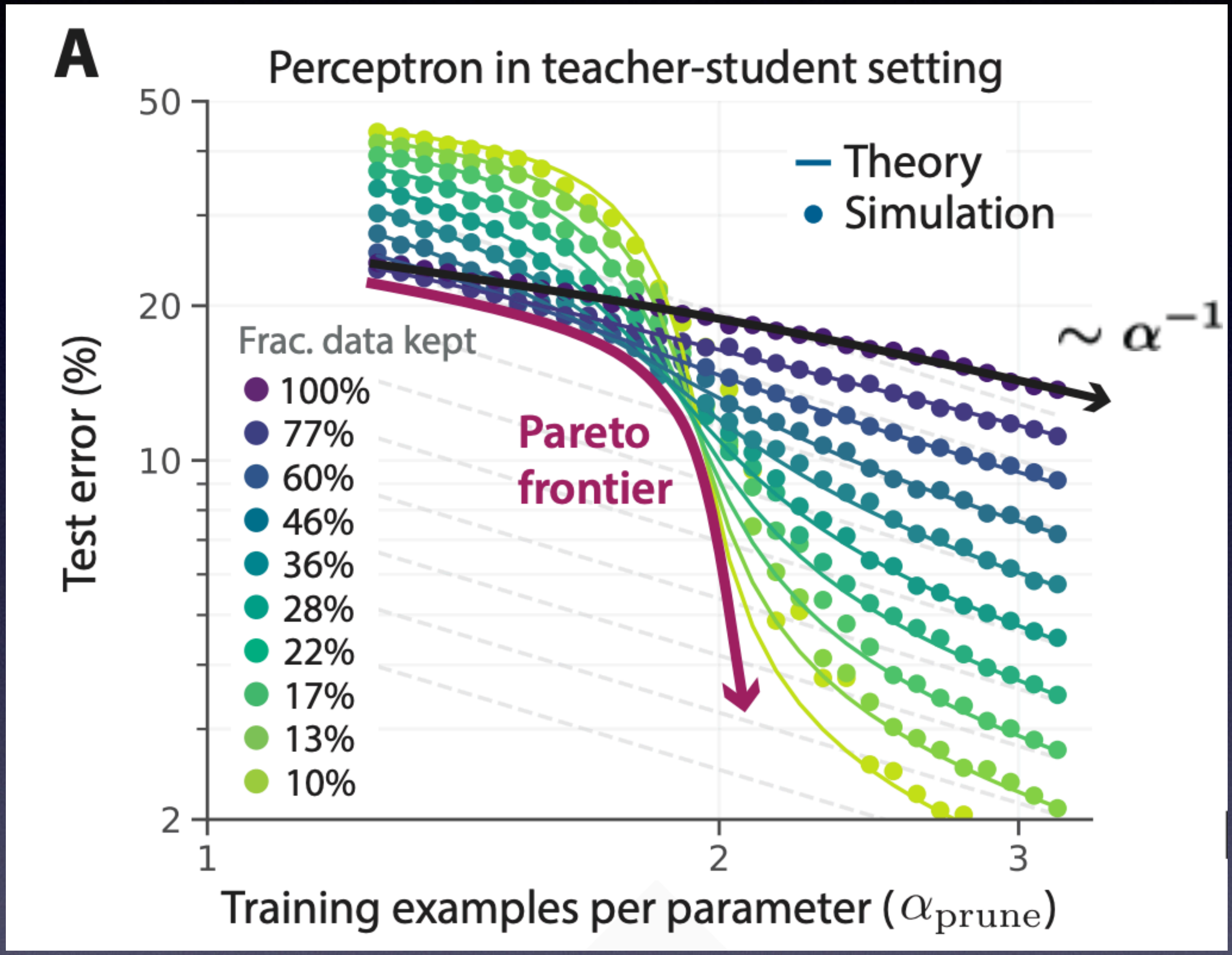
The teacher generates the labels: $y^\mu = \mathrm{sign}(\mathbf{T} \cdot \mathbf{x}^\mu)$

The student fits the data to recover the value of $\mathbf{T}$

For pruning:

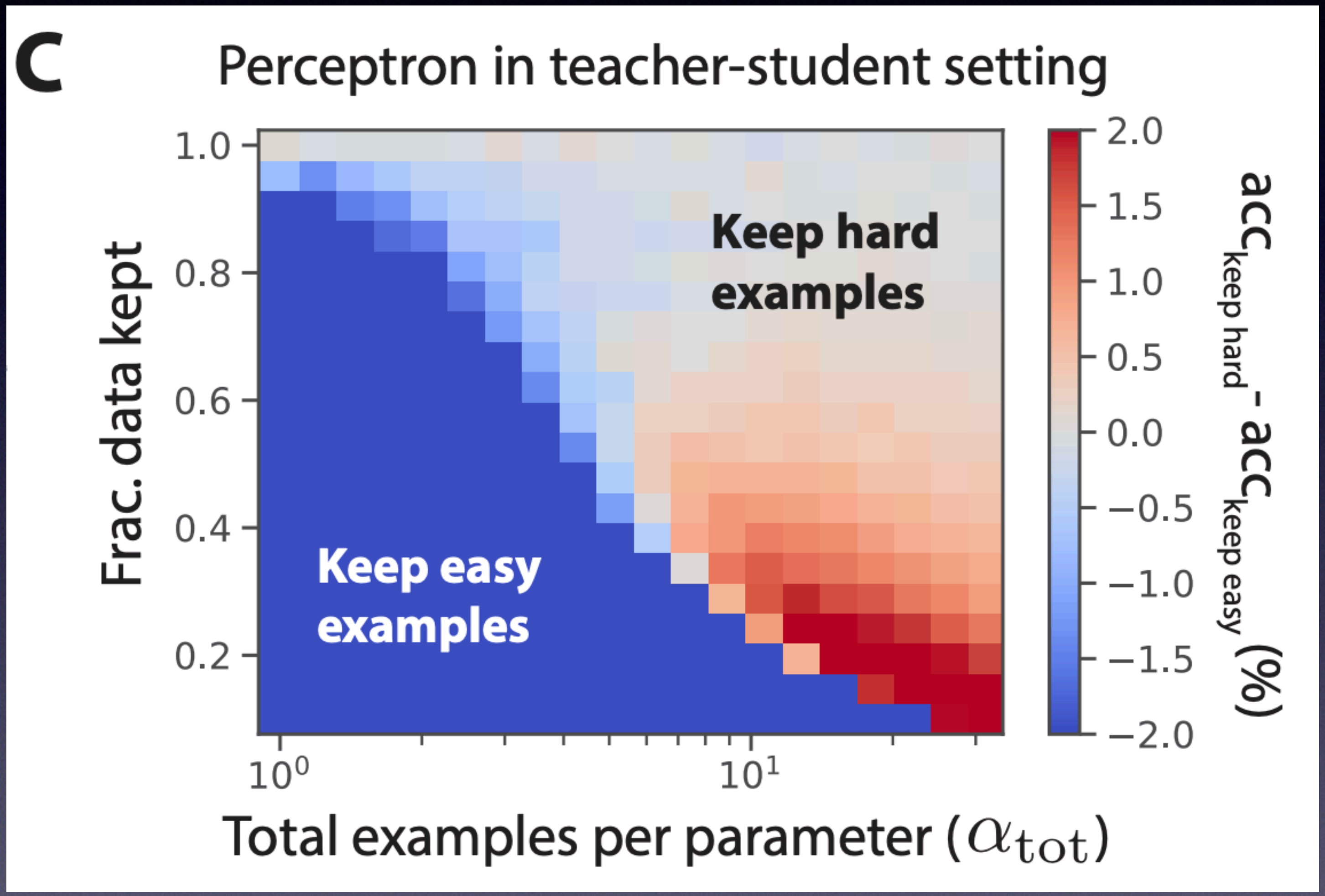
1. Train a "probe", a student model, for only a few epochs
2. Compute a "difficulty metric"
3. Prune the dataset and re-train a model to full convergence

Data Pruning: toy-model results



1. Pruning yields better scaling
2. If total amount of data is small, keeping hard samples is worse than choosing randomly

Data Pruning: toy-model results

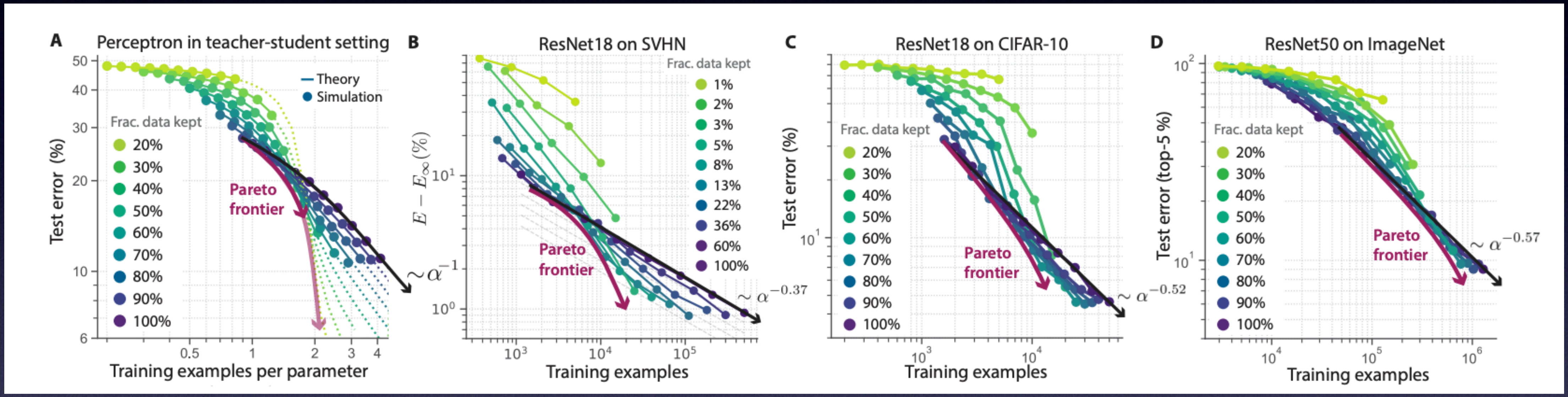


The optimal pruning strategy depends on the total amount of data!

Data Pruning in practice

Does data pruning work with more realistic datasets?

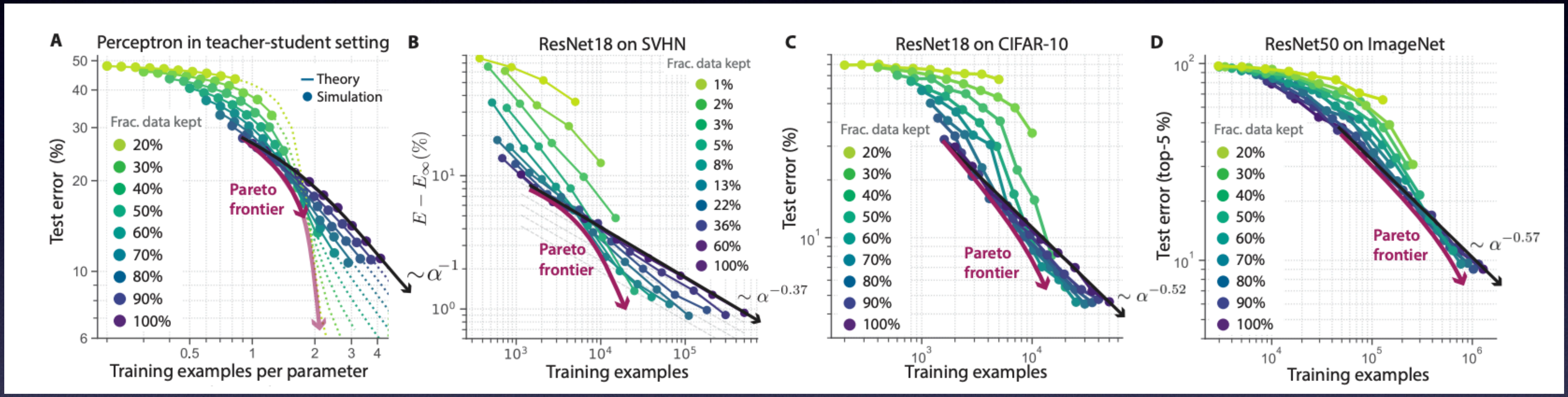
Yes, but **it depends on the metric**



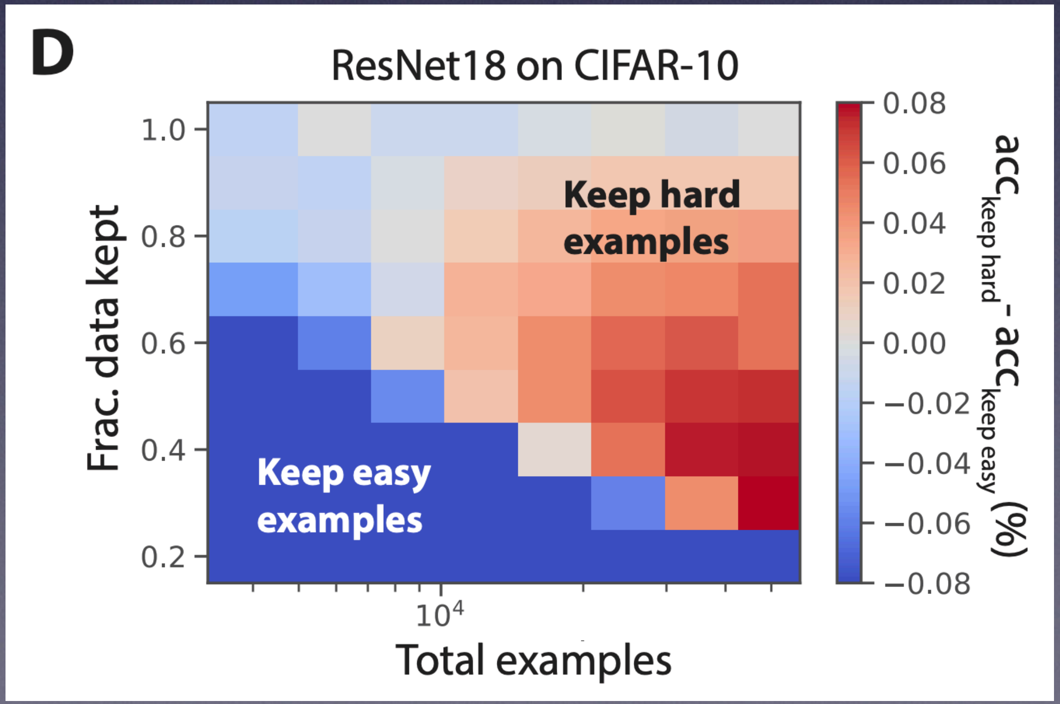
Data Pruning in practice

Does data pruning work with more realistic datasets?

Yes, but **it depends on the metric**

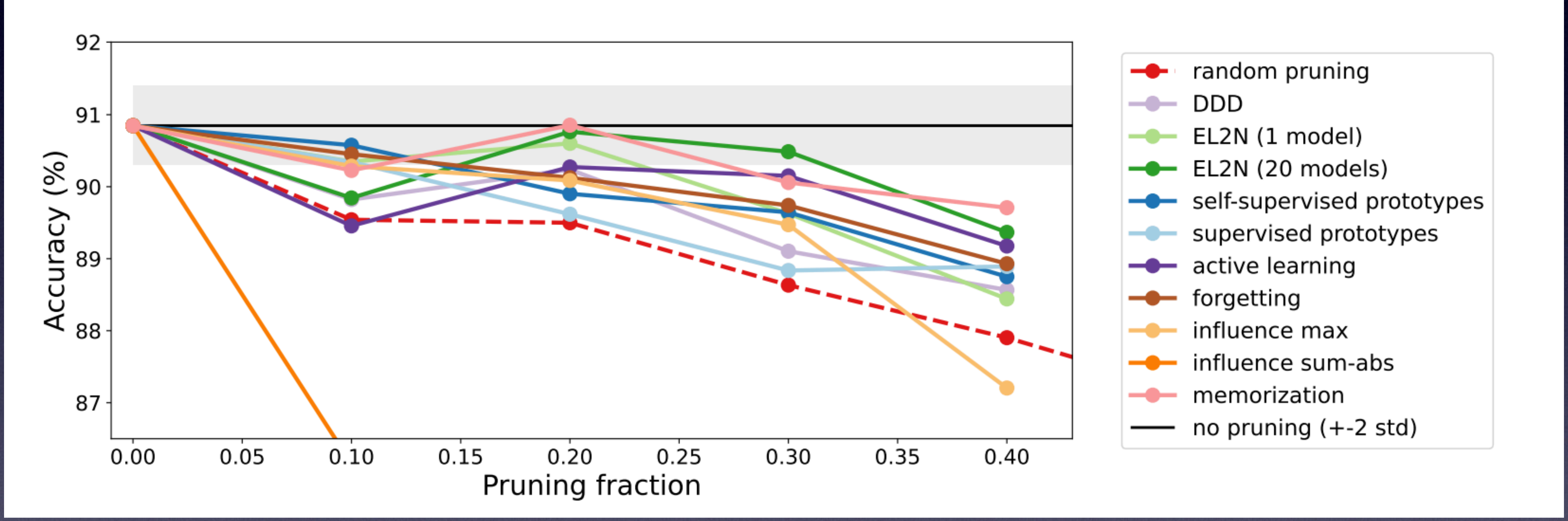


The shift from easy to hard samples also remains true



Data Pruning: metrics

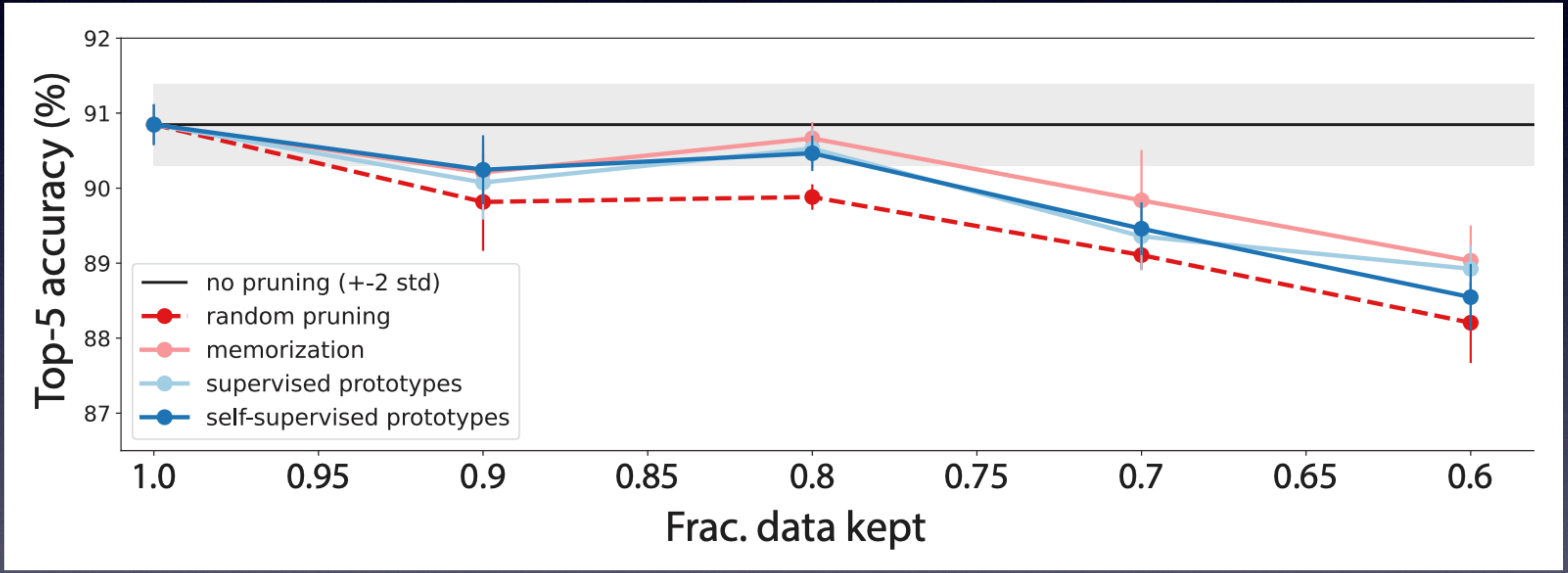
Which is the best "difficulty" metric?



- EL2N: L_2 norm of the error
- Memorisation seems the best (?)
- Some are worse than random

Data Pruning: unsupervised metric

Memorisation needs labels.
Can we have a metric that does not need them?



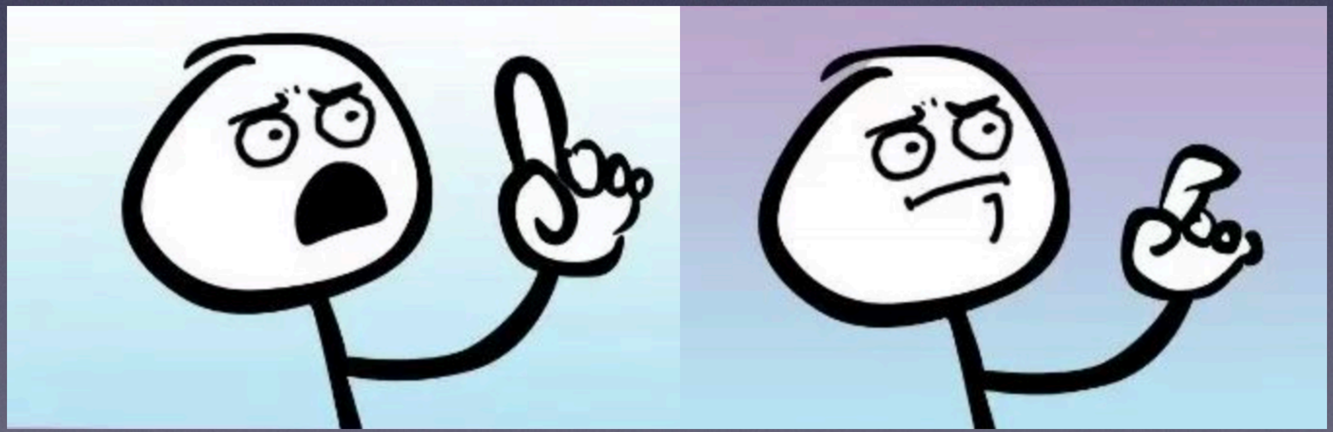
k-means clustering on the embedding of a pre-trained model with class-balancing gives good results

Data efficiency: conclusions

- Data inefficiency has high cost
- Data pruning can be effective
- The issue is the “difficulty metric”
- Memorisation seems to be the best, but it is very expensive
- For now, EL2N gives the best cost/accuracy
- Unsupervised metrics good for continuous learning

Data efficiency: conclusions

- Data inefficiency has high cost
- Data pruning can be effective
- The issue is the “difficulty metric”
- Memorisation seems to be the best, but it is very expensive
- For now, EL2N gives the best cost/accuracy
- Unsupervised metrics good for continuous learning



Questions?