Google AI

Using Logic to Understand Learning

Sat Chatterjee | schatter@google.com

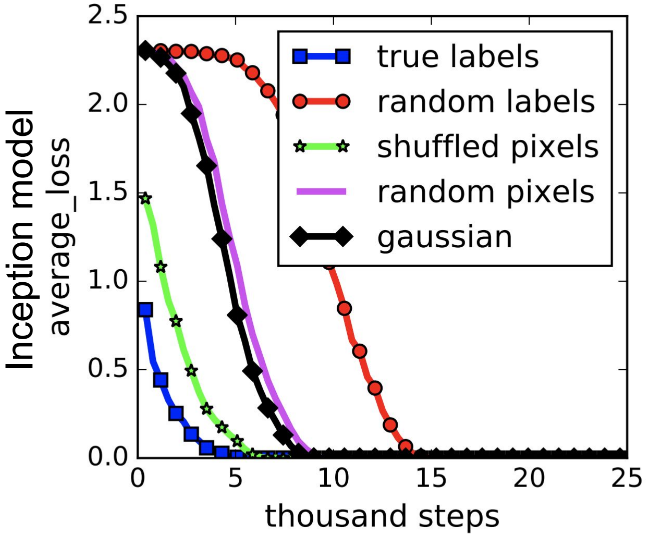
Joint with Alan Mishchenko (Berkeley) and Piotr Zielinski (Google)

The 2nd Logic Synthesis Software School (LSSS) | July 23rd 2021

Our Goal: Not Machine Learning for Logic Synthesis but Logic Synthesis to understand Machine Learning

“Ask not what ML can do for you,
but what you can do for ML …”

The Big Question: Why do Neural Networks Generalize?



Understanding deep learning requires rethinking generalization. Zhang et al. ICLR `17.

The problem is that neural networks have enough capacity to memorize random data!

This led to speculation that perhaps different things are going on in the random and real cases.

But that is inherently unsatisfactory ...

An Alternate Theory

Perhaps neural networks always memorize their training data?

Q1: But .. can pure memorization even lead to generalization?

and

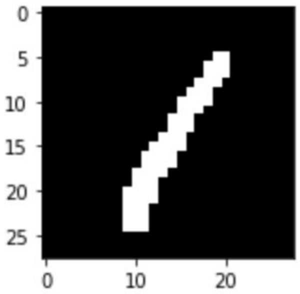
Q2: And if so, could neural nets essentially be look-up tables?

Can pure memorization even lead to generalization?

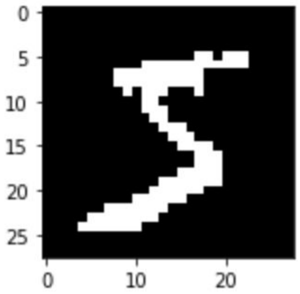
SC [ICML ‘18]

Naive memorization

A simple but non-trivial task: Binarized MNIST



class 0
(digits ‘0’-‘4’)



class 1
(digits ‘5’-‘9’)

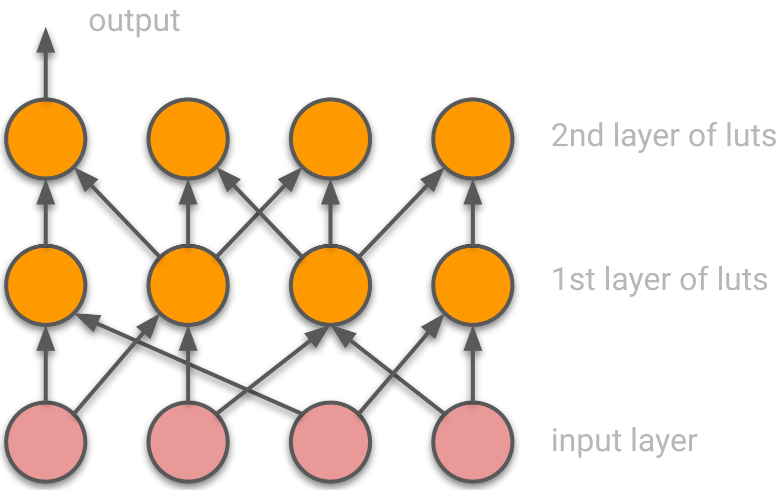
Obvious strategy: Build a giant lookup table (*lut*) from the training set

Problem: Does not generalize, test accuracy is chance (50%).

Memorization inspired by FPGAs

Instead of a single large lut, build a (random) deep network of small luts.

Still memorization: No search, backpropagation, or optimization during training.



example with k = 2 luts

(size of each lut) k	accuracy			
	on real data		on random data	
	training	test	training	test
2	0.66	0.66	0.51	0.53
4	0.81	0.81	0.53	0.54
6	0.85	0.86	0.55	0.52
8	0.89	0.87	0.60	0.51
10	0.94	0.89	0.69	0.50
12	0.99	0.90	0.82	0.51
14	1.00	0.83	0.92	0.51
16	1.00	0.66	0.98	0.52

Harder to memorize random data than real data: need larger luts for same training accuracy

(Like a neural network)

Can memorize random data, yet generalize on real data!

(Like a neural network)

Learning through pure memorization is possible!

An Alternate Theory

Perhaps neural networks always memorize their training data?

Q1: But .. can pure memorization even lead to generalization?

and

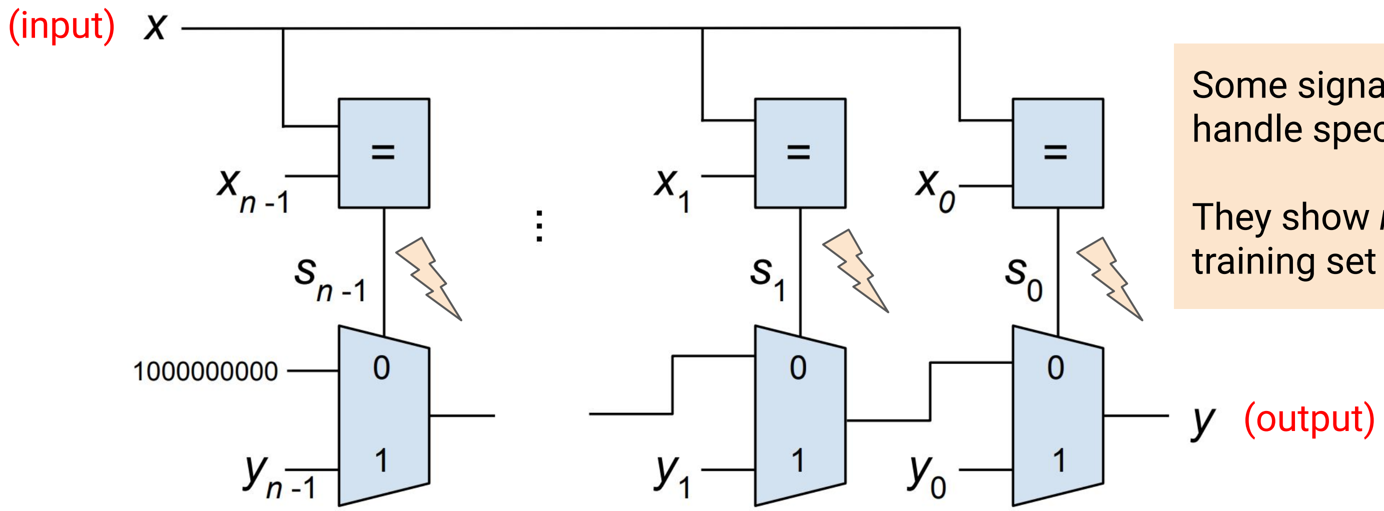
Q2: And if so, could neural nets essentially be look-up tables?

Our pure memorization model even seems to display characteristics similar to neural nets!

How to tell if a given circuit is a lookup table or not?

SC and Alan Mishchenko [ICML ‘20]

A lookup table for memorizing the training set $(x_0, y_0), (x_1, y_1), .., (x_{n-1}, y_{n-1})$



Some signals are there only to handle specific training examples.

They show *rare patterns* when the training set is simulated.

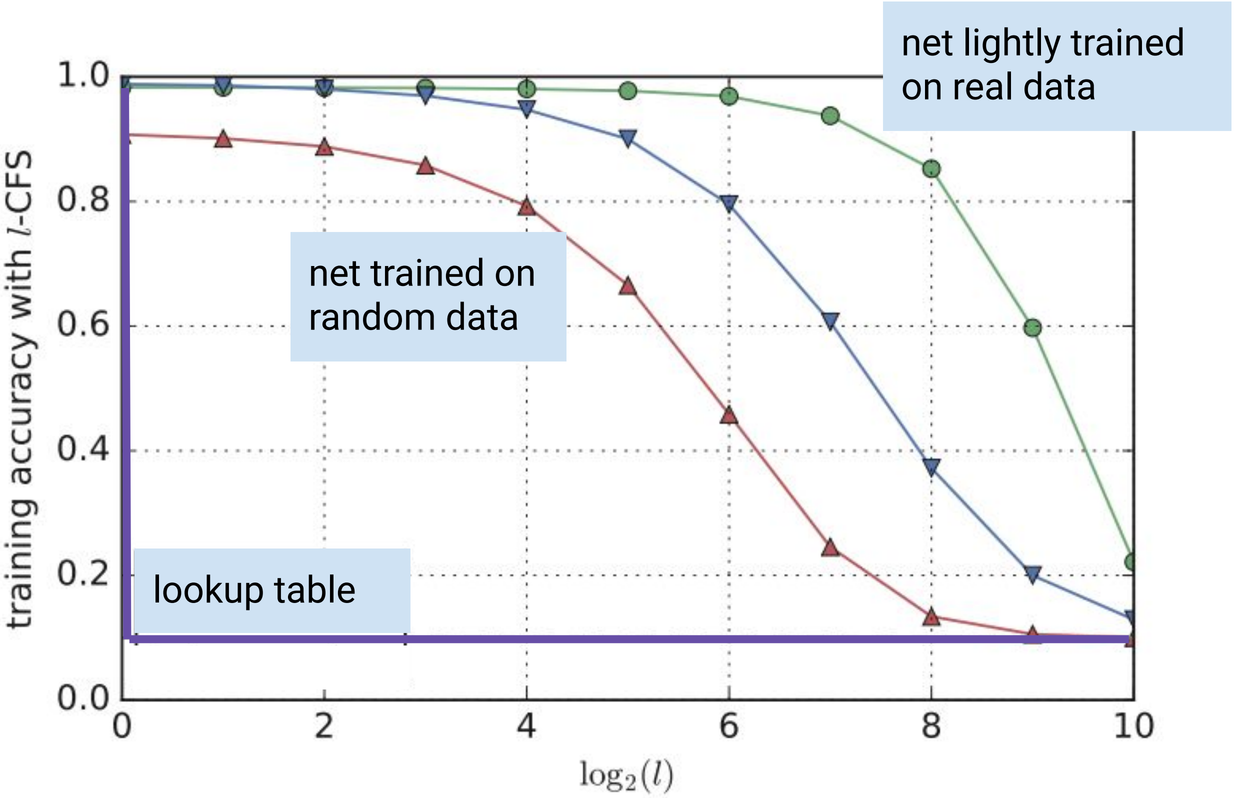
Counterfactual Simulation (CFS): Modify simulation to perturb rare patterns.

If you have a lookup table, this should destroy the training accuracy.

Applying CFS to neural nets

We quantize and compile the neural net to a circuit and perform CFS.

More common patterns than would be expected if they were simply lookup tables.



(How rare a pattern has to be before we perturb)

An Alternate Theory

Perhaps neural networks always memorize their training data?

Q1: But .. can pure memorization even lead to generalization? 

and

Q2: And if so, could neural nets essentially be look-up tables? 

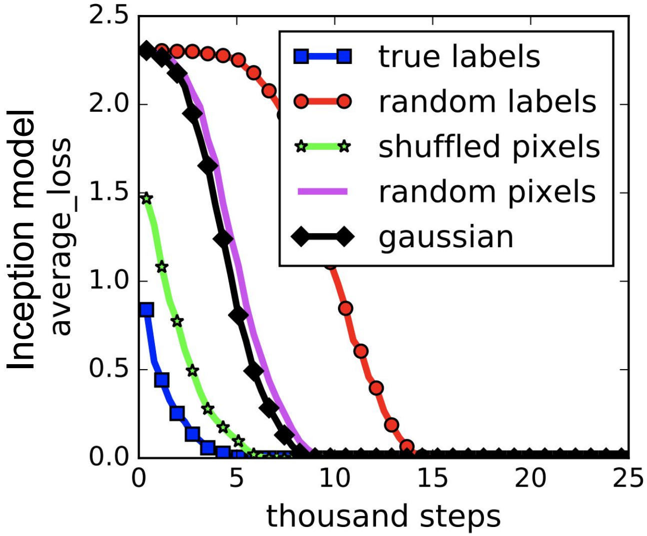
There are more common patterns in neural networks than would be expected in a simple lookup table.

A New Theory of Generalization: Coherent Gradients

SC [ICLR ‘20]

SC and Piotr Zielinski [~~arXiv:2008.01217~~, WIP]

Reminder: The Big Question



Understanding deep learning requires rethinking generalization. Zhang et al. ICLR `17.

Why do Neural Nets generalize?

(when they have sufficient capacity to memorize)

Answer: Gradient descent finds and exploits commonality between training examples

1. Many common patterns ⇒ good generalization
2. Random data also has (spurious) patterns

How does gradient descent find common patterns?

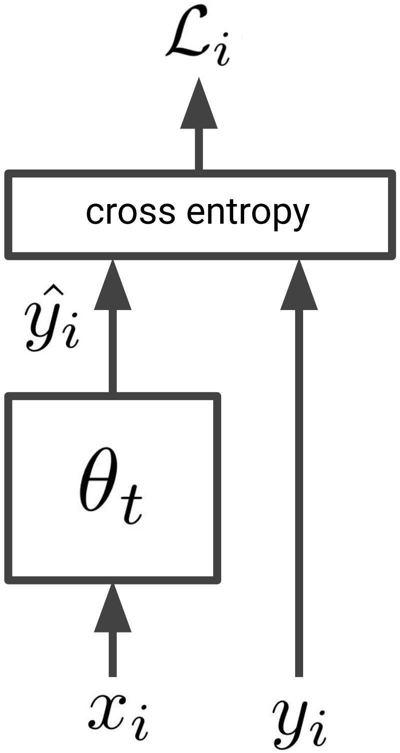
$$\theta_0 := \text{random value}$$

Small local step
to reduce loss

$$\theta_{t+1} := \theta_t - \alpha \cdot [\nabla \mathcal{L}](\theta_t)$$

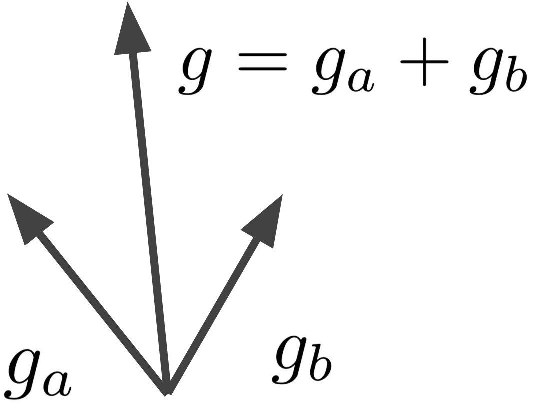
$$\mathcal{L}(\theta_t) = \sum_i \mathcal{L}_i(\theta_t)$$

$$\boxed{[\nabla \mathcal{L}](\theta_t) = \sum_i [\nabla \mathcal{L}_i](\theta_t)}$$

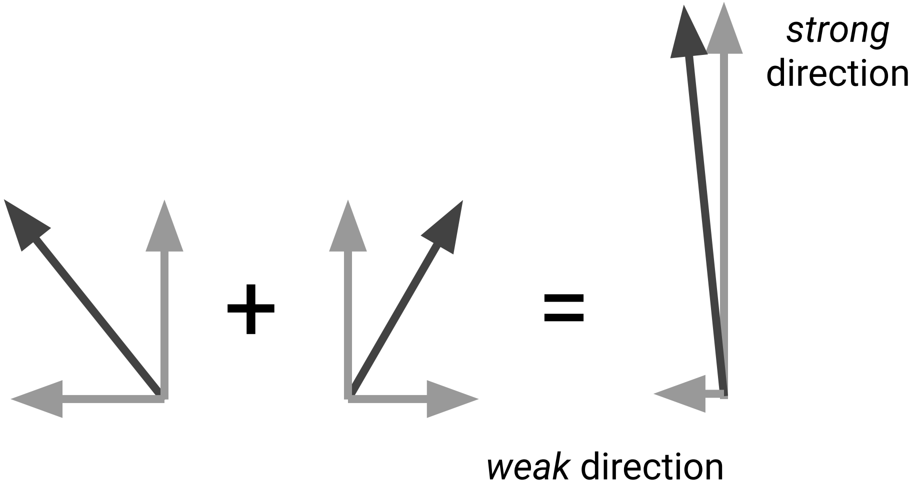


The commonality detection if it happens must happen in the update rule, and in particular, due to this. *(Where else?)*

Consider the overall gradient of two examples *a* and *b*.



Now, in terms of components,



The gradient is stronger in the common direction that improves <u>both</u> *a* and *b*

Therefore, the biggest parameter changes are those that benefit both *a* and b

Sat Chatterjee (schatter@google.com)

But are there common directions between the examples?

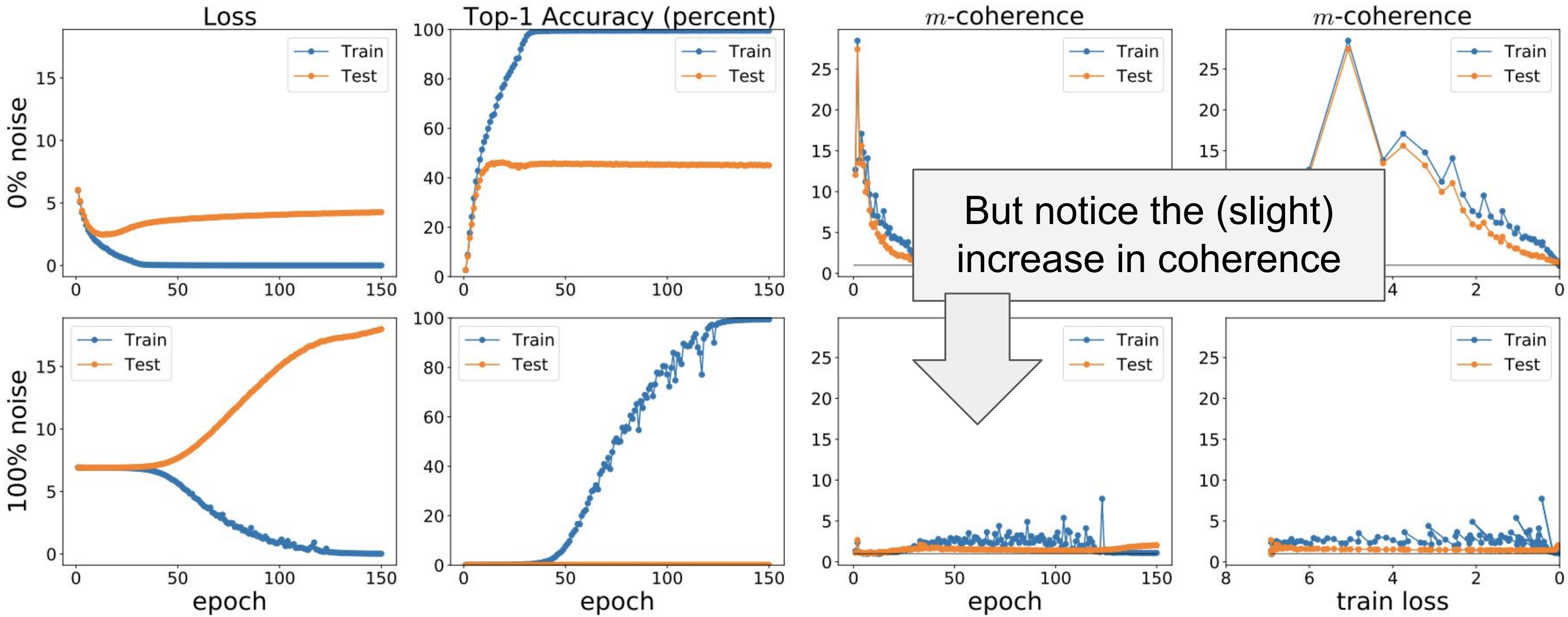
We define a simple **normalized** metric (*coherence*) to quantify commonality.

$$\alpha(w) \equiv \frac{\underset{z \sim \mathcal{D},\, z' \sim \mathcal{D}}{\mathbb{E}} [\ g(w,z) \cdot g(w,z')\ ]}{\underset{z \sim \mathcal{D}}{\mathbb{E}} [\ g(w,z) \cdot g(w,z)\ ]}$$

The coherence of *m* examples whose gradients are pairwise orthogonal is 1/*m*.

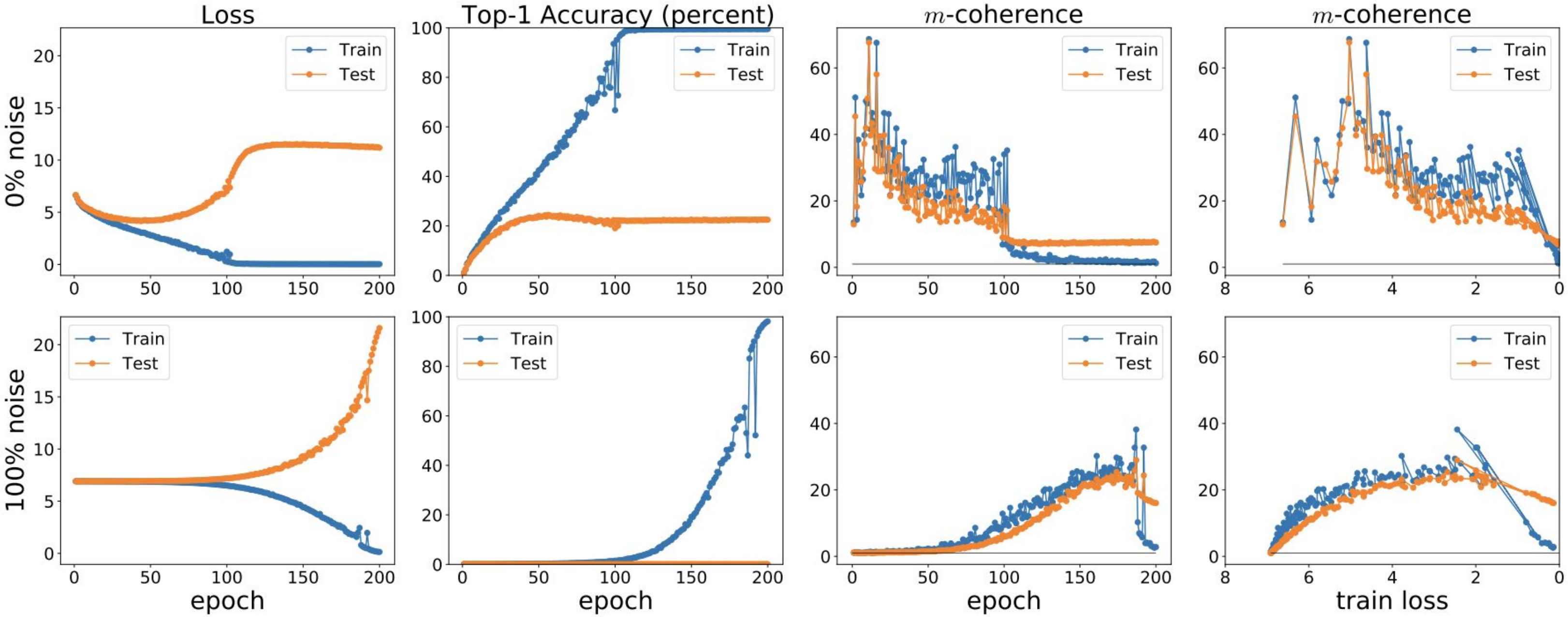
This motivates the notion of *m*-coherence which is simply $m\alpha(w)$.
Intuitively, it measures how many examples each example helps in that step.

Coherence of ResNet-50 on ImageNet



Much higher coherence with real labels than with random labels

Coherence of AlexNet on ImageNet



Increase in coherence in the random label case is much more prominent

A Theory of Coherence

We can relate the generalization gap to the coherence using the theory of algorithmic stability.

$$|\,\mathrm{gap}(\mathcal{D}, m)| \leq \frac{L}{m} \sum_{t \in [T]} [\eta_k\, \beta]_{k=t+1}^{T} \cdot \eta_t \cdot \bar{g}(w_{t-1}) \cdot \sqrt{2\ (1 - \alpha(w_{t-1}))}$$

1. Higher the coherence, lower the generalization gap.
2. High coherence early on in training is better than high coherence later on.

Summary of Coherent Gradients

Simple, intuitive explanation for generalization in deep learning

- Uniformly explains memorization and generalization
- Why some examples are learned earlier than others
- Why learning is possible with random labels
- Why early stopping works
- Impact of width and depth [WIP]

It is causal explanation

- It leads to new gradient descent algorithms to reduce overfitting

Avoids theoretical obstacles faced by competing theories

Concluding Thoughts

1. Ideas from logic synthesis can help shed light on fundamental questions in machine learning.

2. Based on these insights, can we design more efficient (discrete) algorithms for deep learning?

Thank you!

The End

Abstract

A fundamental question in Deep Learning today is the following: Why do neural networks generalize when they have sufficient capacity to memorize their training set. In this talk, I will describe how ideas from logic synthesis can help answer this question. In particular, using the idea of small lookup tables, such as those used in FPGAs, we will see if memorization alone can lead to generalization; and then using ideas from logic simulation, we will see if neural networks do in fact behave like lookup tables. Finally, I'll present a brief overview of a new theory of generalization for deep learning that has emerged from this line of work.

(This talk is based on joint work with Alan Mishchenko and Piotr Zielinski.)