Modern Era of Statistics

Ramin Hasani, Ph.D.
Principal AI Scientist, Vanguard
Research Affiliate, MIT

MIT Introduction to Deep Learning
January 12th 2023
Modern Era of Statistics

Petaflop/s-days

1e+4
1e+2
1e+0
1e-2
1e-4
1e-6
1e-8
1e-10
1e-12
1e-14


First Era Modern Era

Credit: https://morioh.com/p/9c545ba9416b
Modern Era of Statistics
Language Models size – up to Dec, 2022

- BERT 340M
- GPT-1 117M
- GPT-2 1.5B
- T5 11B
- Plato-XL 11B
- Macaw 1.5B
- Cohere 11B
- Megatron-11B 52.4B
- ruGPT-3

- MT-NLG 530B
- Jurassic-1 178B
- LaMDA 137B
- Gopher 280B
- BlenderBot2.0 9.4B
- GPT-J 6B

- PaLM 540B
- PaLM-Coder
- Minerva

- BLOOM 176B
- NLLB 54.5B
- GLM-130B

- OPT-175B
- BB3 175B
- YaLM 100B

- Gal 120B
- WeLM

- NOOR
- SeeKeR 2.7B

- Z-Code++ 710M
- AlexaTM

Credit: LifeArchitect.ai/models

Beeswarm/bubble plot, sizes linear to scale. Selected highlights only. Alan D. Thompson, December 2022. https://lifearchitect.ai/
Modern Era of Statistics
Time Series

Medical Diagnoses

Financial Time Series

Market Summary - Vanguard Total Stock Market Index Fund ETF
196.83 USD
+139.04 (240.60%) ↑ all time
Jan 11, 11:33 AM EST - Disclaimer

(a) Decide treatment plan
(b) Decide optimal time of treatment
(c) Decide when to stop treatment

https://www.vanderschaar-lab.com/individualized-treatment-effect-inference/

Modeling time series of 90k time steps long,
with Liquid Structural State-Space Models (Liquid-S4)

https://github.com/raminmh/liquid-s4

Hasani et al. 2023
Modern Era of Statistics
Generative modeling

Generative Adversarial Networks

Stable Diffusion

Goodfellow et al. 2014

Credit: https://www.jousefmurad.com/ai/a-primer-on-stable-diffusion/
Modern Era of Statistics

Bigger seems to be better? But why?
Solving $n$ equations requires $n$ unknowns

\[2x + 3y = 20\]
\[4x - 2y = 12\]

But then deep learning:

Choose excessively more unknowns to learn from $n$ data (equations)!
Scale in Modern Machine Learning

MNIST  \( n = 60k \) points, \( d = 28 \times 28 \) images

Today models with millions of parameters are trained on MNIST
The performance improves with increasing the number of parameters!

How does this make sense? What are we learning?

Generalization bound  \( \alpha \frac{\# \ of \ param}{\sqrt{\text{dataset \ size}}} \)

ImageNet: is 1.4M images of size 256 x 256 x 3 and models can be hundreds of millions of parameters.

NLP: datapoints of few billions, and models are hundreds of billions!
Prompt: A portrait photo of a kangaroo wearing an orange hoodie and blue sunglasses standing on the grass in front of the Sydney Opera House holding a sign on the chest that says Welcome Friends!
Benign Overfitting & Double Descent

Experiment from [Nakkiran et al., 2019] on CIFAR-10 with 15% label noise:

Scaling does help generalization a bit!
Modern Era of Statistics
Modern Era of Statistics

Generalization across many tasks and domains

Scale improves Robustness

Worsen accuracy on minority samples

Reasoning stays unchanged

[Reed et al. DeepMind 2022]

[Madry et al. ICLR 2018, Bubeck & Selke NeurIPS 2021]

[Sagawa et al. ICML 2020]

[Liu et al. Google Research 2022]
Scale Improves Robustness

Experiment from [Madry et al., 2018] on MNIST:

They use adversarial training against Projected Gradient Decent (PGD) attacks on various number of params.
Scale is a law of robustness

A Universal law of robustness [Bubeck and Sellke 2021]:
https://youtu.be/OzGgualEHOU

Fix any “reasonable” function class with p parameters (e.g., deep nets with poly-size parameters and NOT Kolmogorov-Arnold type networks).

Sample n data points from a “truly high dimensional” distribution (e.g., a mixture of Gaussians, or ImageNet with a properly defined notion of “dimension”). Add label noise.

Then, to memorize this dataset (i.e., optimize the training error below the label noise level), and to do so robustly (in the sense of being Lipschitz), one must necessarily have 

**dramatic overparameterization:**

\[ p \geq n \, d \]

Kolmogorov-Arnold Representation Theorem

\[
f(x) = f(x_1, \ldots, x_n) = \sum_{q=0}^{2n} \phi_q \left( \sum_{p=1}^{n} \phi_{q,p}(x_p) \right)
\]

the non-smoothness of the inner functions and their "wild behavior" has limited the practical use of the representation [Girosi & Poggio 1989]

Lipschitzness: If I move my inputs by \( \epsilon \) then I would ideally want the output also moves by \( \epsilon \)

\[
d_Y(f(x_1), f(x_2)) \leq K d_X(x_1, x_2)
\]
Why is $p \gg nd$ called "dramatic Overparameterization"?

Intuitively, memorizing $n$ data points is about satisfying $n$ equations, so order $n$ parameters should be enough.

**Theorem (Baum 1988)**
Two layer neural net with threshold activation function only need ($p \approx O(n)$) to memorize binary labels.

[Yun, Sra, Jadbabaie 2019; Bubeck, Eldan, Lee, Mikulincer 2020]
In fact the same is true with ReLU on real labels.

[Bubeck, Eldan, Lee, Mikulincer 2020] In fact even Neural Tangent Kernels can do it.
Examples in real-world data:

MNIST. It has around $n \approx 10^5$ and $d \approx 10^3$. [Madry et al., 2018] show a transition in robust accuracy at around $p \approx 10^6$

Note 1: their notion of robustness (PGD) does not exactly match the law of robustness (Lipschitz constant).

Note 2: the law seems to be contradicted since $10^6 \ll 10^5 \times 10^3$?

No. MNIST is NOT truly “10³ — dimensional”. *Effective dimension*: $d_{eff} \approx 10^1 \rightarrow p \sim n \, d_{eff}$

Note 3: what is “noisy labels” in real data? Measuring the “difficult” part of a learning problem. For MNIST it should be 2-5% gain in accuracy.

What about ImageNet? It’s $n \approx 10^7$ and $d \approx 10^5$ ($d_{eff} \approx 10^3$?), hence we predict that at least $10^{10}$ parameters are needed. Current models are too small?!?? (Less than $10^9$ parameters)
What about Smoothness?

All these constructions are $\Omega(\sqrt{d})$ Lipschitz even for well-dispersed data (e.g., i.i.d. on the sphere), but in principle one can memorize such data with $O(1)$ – Lipschitz functions (We assume $n = \text{poly}(d)$):

Picture can easily be realized with $k = O(n)$ neurons ($p = nd$). So we have two options: either small model ($p = n$) but nonrobust ($\text{Lip} = \sqrt{d}$), or very large ($p = nd$) and very robust ($\text{Lip} = 1$).

Is this tradeoff real? Can we do better than $\text{Lip} \leq O(\frac{\sqrt{nd}}{p})$?

The law of robustness says this is tight!
Great Generalization
More Robust
Reasoning
Bias and Fairness
Energy
Accountability

Great Generalization
More Robust
Reasoning?
Bias and Fairness?
Energy?
Accountability?

HOW?
Neuroscience inspiration as inductive bias

Brain

Liquid Neural Networks

© Image: Allen Institute for Brain Science

Lechner et al. Nature Machine Intelligence 2020
Hasani et al. Nature Machine Intelligence 2022
What are the building blocks’ differences?

✓ Neural dynamics are typically continuous processes and are described by differential equations

✓ Synaptic release is much more than scalar weights

✓ Recurrence, memory, and sparsity
Let’s incorporate these building block differences to:

Improve representation learning
Improve robustness and flexibility of models
Improve models’ interpretability

Explore Continuous-time (depth) models
What is a continuous-time/depth neural network?

\[ \frac{dx}{dt} = f(n, k, l_{type})(x(t), I(t), \theta) \]

- Number of layers
- Width
- Hidden state
- Inputs
- Activations

Dynamical systems

Residual Network

\[ h_{t+1} = h_t + f(h_t, \theta_t) \]

He et al.
CVPR 2016

ODE Network

\[ \frac{dh(t)}{dt} = f(h(t), t, \theta) \]

Chen et al.
NeurIPS 2018

Figure Credit: Chen et al. NeurIPS 2018
What is a continuous-time/depth neural network?

Standard Recurrent Neural Network (RNN)
Hopfield 1982

$x(t + 1) = f(x(t), I(t), t; \theta)$

Neural ODE
Chen et al. NeurIPS, 2018

$\frac{dx(t)}{dt} = f(x(t), I(t), t; \theta)$

Continuous-time (CT) RNN
Funahashi et al. 1993

$\frac{dx(t)}{dt} = -\frac{x(t)}{\tau} + f(x(t), I(t), t; \theta)$

Figure Credit: Chen et al. NeurIPS 2018
Liquid Time-Constant (LTC) networks

1. Linear state-space model

\[ \frac{dx(t)}{dt} = -x(t)/\tau + S(t) \quad S(t) \in \mathbb{R}^M \]

2. Non-linear synapse Model

\[ S(t) = f(x(t), I(t), t, \theta)(A - x(t)) \]

\[ \frac{dx(t)}{dt} = - \left[ \frac{1}{\tau} + f(x(t), I(t), t, \theta) \right] x(t) + f(x(t), I(t), t, \theta) A \]

“Liquid” = variable
Liquid Time-Constant Networks

Standard Neural Nets

\[ \sigma(.) \xrightarrow{W} \sigma(.) \quad \frac{1}{\tau} \]

\[ B \xrightarrow{I(t)} \]

\[ \frac{1}{\tau} \quad \text{intrinsic coupling} \]

\[ W\sigma(.) \quad \text{liquidity modulator} \]

\[ B \quad \text{input regulator} \]

Liquid Neural Nets

\[ x_j \xrightarrow{W} \]

\[ \sigma(.) \xrightarrow{1/\tau} \]

\[ x_i \]

\[ B \xrightarrow{I(t)} \]

\[ w\sigma(.) \quad \text{liquidity modulator} \]
LTCs: Performance
High-fidelity autonomy by LTCs - end-to-end learning

What if we replace the fully connected layers by a recurrent neural network?

ODE-RNN

CT-RNN

LSTM

LTC-based Networks?
LTCs: Performance
High-fidelity autonomy by LTCs
end-to-end learning of Neural Circuit Policies (NCP)

Now we compare properties of NCPs with a number of other models

https://github.com/mlech26l/ncps

Lechner, Hasani, Amini, Henzinger, Grosu, Rus, Nature Machine Intelligence, 2020
LTCs: Performance
Robustness to perturbations

Liquid time-constant neuron resilience to sensory noise
Why can LTCs learn better causal relationships?

**Taxonomy of Models**

Adapted from: Peters, Janzing, Schölkopf, MIT Press, 2017

- Physical
- Structural causal
- Causal graphs
- Statistical models

**Dynamic Causal Models**

\[
\frac{dx}{dt} = g(x(t), I(t); \theta) \\
(A + I(t)B) x(t) + C(I(t))
\]

- Internal coupling
- Internal Intervention
- External Intervention

The **LTC** model reduces mathematically to a Dynamic Causal Model

- Insights about system
- Learn from data
- Answer counter factual
- Predict in IID
- Interventions
Differential equations can form causal structures

Physical dynamics can be modeled by a set of differential equations

- Predict **future evolution** of the dynamical system
- Describe effect as a result of **interventions**

(Friston et al., 2003)
Differential equations can form causal structures

Given the following system of differential equations:

\[
\frac{dx}{dt} = g(x),
\]

where \( x \in \mathbb{R}^d \), \( x(0) = x_0 \), \( g(x) = \) nonlinearity

**Picard-Lindelöf theorem**
(Nevanlinna, 1989)
states that above DE has a unique solution as long as \( g \) is Lipschitz

**Euler solution**
The Euler method unrolling:
\[
x(t + \delta t) = x(t) + dt \cdot g(x)
\]

**Causal structure**
(Schölkopf, 2019)
Representation under uniqueness conditions forms a temporally **causal structure**
Dynamic Causal Models (DCMs)

\[
\frac{dx}{dt} = g(x(t), I(t); \theta)
\]

Bilinear approximation

\[
\frac{dx}{dt} = (A + I(t)B) x(t) + C(I(t))
\]

(Friston et al., 2003)

**Internal coupling**

\[
A = \frac{\partial g}{\partial x} \bigg|_{I=0}
\]

Regulates hidden state

**Internal Intervention**

\[
B = \frac{\partial^2 g}{\partial x \partial I}
\]

Controls coupling sensitivity among network’s nodes

**External Intervention**

\[
C = \frac{\partial g}{\partial I} \bigg|_{x=0}
\]

Regulates external input

The **Liquid Time-constant (LTC)** model reduces to a **Dynamic Causal Model** of this form if

1. \(g(\cdot)\) is continuous and bounded
   
   \(e.g., \tanh(Wx(t) + WI(t) + b)\)
   

2. \(\tau\) is positive

   Enforced by activation constraint

   (Funahashi and Nakamura, 1993)
LTC-based: Neural Circuit Policies
Performance – Attention

Flying Performance

Visual Backprop Attention Map

Liquid Neural Networks
Performance – Attention

- Red cubic target is fixed.
- Drone learns to navigate to target by visual inputs only
Closed-form Solution of Liquid Networks
Closed-form Continuous-time Neural Networks

\[ \frac{dx(t)}{dt} = -\frac{x(t)}{\tau} + S(t) \]

we solve this in closed-form

\[ x(t) = (x(0) - A) e^{-[\frac{1}{\tau} + f(I(t))]} t f(-I(t)) + A \]

\[ S(t) = f(I(t)) (A - x(t)) \]

\( x(t) \) Postsynaptic neuron’s potential
\( A \) Synaptic reversal potential
\( f(.) \) Synaptic release nonlinearity
\( \tau \) Postsynaptic neuron’s time-constant

https://github.com/raminmh/CfC
Tightness of the Closed-form Solution in Practice

\[ \frac{dx}{dt} = - (w_\tau + f(x, I)) x(t) + A f(x, I) \]

- \text{Inputs} \ I(t)
- \text{neuron's state} \ x(t)
- \text{Nonlinearity} \ f(\cdot)
- \text{Parameters} \ w_\tau, A

\[ x(t) = (x(0) - A) e^{-[w_\tau + f(x, I)]t} f(-x, -I) + A \]
# How Well liquid CfCs perform in Time-series modeling?

## Physical Dynamics Modeling

<table>
<thead>
<tr>
<th>Model</th>
<th>Square-error</th>
</tr>
</thead>
<tbody>
<tr>
<td>‡ODE-RNN (Rubanova et al., 2019)</td>
<td>1.904 ± 0.061</td>
</tr>
<tr>
<td>‡CT-RNN (Funahashi and Nakamura, 1993)</td>
<td>1.198 ± 0.004</td>
</tr>
<tr>
<td>‡Augmented LSTM (Hochreiter and Schmidhuber, 1997)</td>
<td>1.065 ± 0.006</td>
</tr>
<tr>
<td>‡CT-GRU (Mozer et al., 2017)</td>
<td>1.172 ± 0.011</td>
</tr>
<tr>
<td>‡RNN-Decay (Rubanova et al., 2019)</td>
<td>1.406 ± 0.005</td>
</tr>
<tr>
<td>‡Bi-directional RNN (Schuster and Paliwal, 1997)</td>
<td>1.071 ± 0.009</td>
</tr>
<tr>
<td>‡GRU-D (Che et al., 2018)</td>
<td>1.090 ± 0.034</td>
</tr>
<tr>
<td>‡PhasedLSTM (Neil et al., 2016)</td>
<td>1.063 ± 0.010</td>
</tr>
<tr>
<td>‡GRU-ODE (Rubanova et al., 2019)</td>
<td>1.051 ± 0.018</td>
</tr>
<tr>
<td>‡CT-LSTM (Mei and Eisner, 2017)</td>
<td>1.014 ± 0.014</td>
</tr>
<tr>
<td>‡ODE-LSTM (Lechner and Hasani, 2020)</td>
<td>0.883 ± 0.014</td>
</tr>
<tr>
<td>coRNN (Rusch and Mishra, 2021)</td>
<td>3.241 ± 0.215</td>
</tr>
<tr>
<td>Lipschitz RNN (Erichson et al., 2021)</td>
<td>1.781 ± 0.013</td>
</tr>
<tr>
<td>LTC (Hasani et al., 2021)</td>
<td><strong>0.662 ± 0.013</strong></td>
</tr>
<tr>
<td>Transformer (Vaswani et al., 2017)</td>
<td>0.761 ± 0.032</td>
</tr>
<tr>
<td>Cf-S (ours)</td>
<td>0.948 ± 0.009</td>
</tr>
<tr>
<td>CfC-noGate (ours)</td>
<td><strong>0.650 ± 0.008</strong></td>
</tr>
<tr>
<td>CfC (ours)</td>
<td><strong>0.643 ± 0.006</strong></td>
</tr>
<tr>
<td>CfC-mmRNN (ours)</td>
<td><strong>0.617 ± 0.006</strong></td>
</tr>
</tbody>
</table>
Standard Deep Neural Network

Task: Identify and navigate to target
Deploy: Closed-loop testing
LTC-based Network

Task: Identify and navigate to target
Deploy: Closed-loop testing
Brain-inspired inductive biases could break the scaling law of neural networks.

Generalization | More Robust | Better Reasoner | Fairer | Energy efficient | Accountable

![Graph showing accuracy vs. model size with Liquid Networks and Overparameterization Regime]

- Classical statistics
The Modern Era of Statistics
Summary

✓ The law of robustness is real! \( p \geq n d \) where \( d \) is the effective dimensionality

✓ Overparameterization improves generalization, and robustness, but does come with sociotechnical challenges (e.g., accountability, fairness and bias, energy)

✓ Architectural Inductive biases, and dynamic processes in neural network architectures (Liquid Neural Networks) could alleviate many of the challenges

✓ Liquid networks enable robust representation learning outside of overparameterization regime, as they have causal mechanisms that dramatically reduces a network’s perceived effective dimensionality.
Some Resources

Get hands-on with LTC-based networks:

github.com/mlech261/ncps

Get hands-on with the closed-form liquid networks:

github.com/raminmh/CfC

Get hands-on with Liquid-S4:

github.com/raminmh/liquid-s4

Get in touch:

rhasani@mit.edu
ramin_hasani@vanguard.com