Debugging Neural Networks

Rishi Gurnani
Ramprasad Group / Georgia Institute of Technology
https://rishigurnani.wordpress.com/
http://ramprasad.mse.gatech.edu

WHY DO WE NEED ML? … DESIGN CHALLENGES!

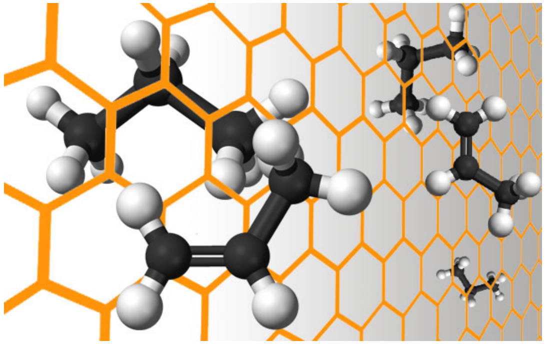
Optimal materials selection or discovery is non-trivial …
… often due to conflicting property requirements

Some polymer examples …

High Energy Density Capacitors

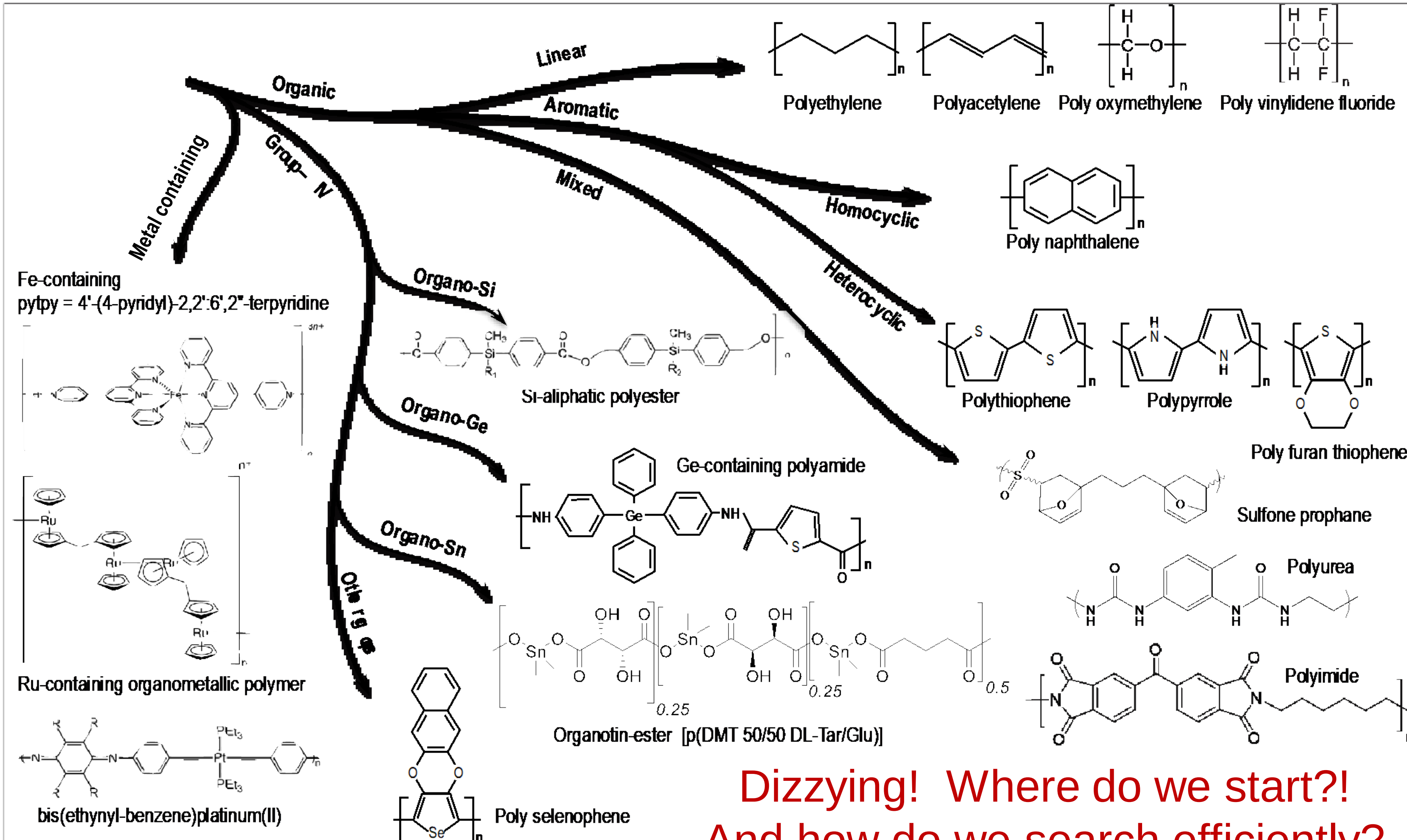
Need: high band gap, high dielectric constant

Gas Separation Membranes



Need: high perm-eability, high selectivity

POLYMER CHEMICAL UNIVERSE



Dizzying! Where do we start?!
And how do we search efficiently?

COMPLEX PROPERTIES

No.	Polymer properties	Data Source	Data Size	ML Algo.	$RMSE_{CV}$	Notes	Reference
1	Polymer crystal bandgap	Comput.	562	GPR	0.26 eV	Training data produced using using HSE06 XC functional[22]	8
2	Polymer chain bandgap	Comput.	3881	GPR	0.24 eV	Training data produced using using HSE06 XC functional[22]	
3	Ionization energy	Comput.	371	GPR	0.21 eV		
4	Electron affinity	Comput.	371	GPR	0.18 eV		
5	Static dielectric constant (crystal)	Comput.	383	GPR	0.38		8
6	Frequency-dependent dielectric constant	Exper.	1193	GPR	0.16	Training data include measurements at 60, 10^2, 10^3, 10^4, 10^5, 10^6, 10^7, 10^9, and 10^{15} Hz	23
7	Refractive index (bulk resin)	Exper.	516	GPR	0.04		24
8	Refractive index (crystal)	Comput.	383	GPR	0.07		8
9	Tensile strength	Exper.	672	GPR	4.75 MPa		
10	Young's modulus	Exper.	629	GPR	120 MPa		
11	Glass transition temperature	Exper.	5076	GPR	18.8 K		8
12	Melting temperature	Exper.	2084	GPR	27.1 K		
13	Thermal decomposition temperature	Exper.	3545	GPR	28.03 K		
14	Polymer/solvent (in) compatibility	Exper.	6721	ANN	93% accurate classification	The compatibility with 24 solvents is predicted	25
15	Solubility parameter	Exper.	112	GPR	0.47 $MPa^{1/2}$		26
16	Gas permeability	Exper.	1779	GPR	1.2 Barrer	The permeability to CH_4, CO_2, He, N_2, O_2, and H_2 is predicted	27
17	Polymer density	Exper.	890	GPR	0.03 g/cc		8
18	Atomization energy	Comput.	391	GPR	0.01 eV/atom		8
19	Specific heat	Exper.	80	GPR	0.07 J/gK		
20	Fractional free volume	Exper.	133	GPR	0.01		
21	Limiting oxygen index	Exper.	101	GPR	3.73%		
22	Tendency to crystallize	Exper.	429/107	CK	8.38%	Training data include low- and high-fidelity data	28

"Machine Learning Predictions of Polymer Properties with Polymer Genome" *Journal of Applied Physics* (2020)

https://www.polymergenome.org/

AGENDA

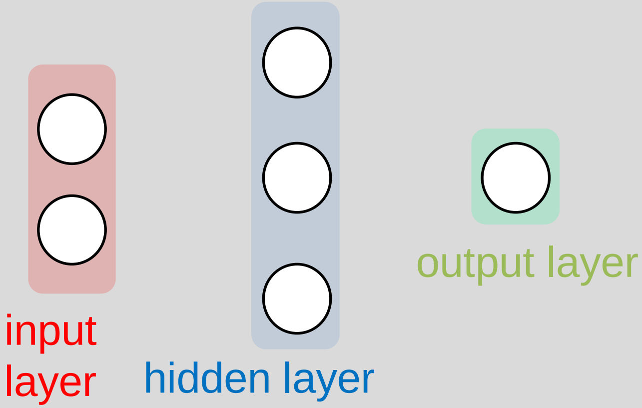
- Brief theory behind neural networks
- Overview of *NetDebugger*
- Demonstration of *NetDebugger*
- Summary

WHAT IS MACHINE LEARNING (ML)?

- A machine is said to *learn* if it tends to improve performance on some task given more and more experience – Tom Mitchell

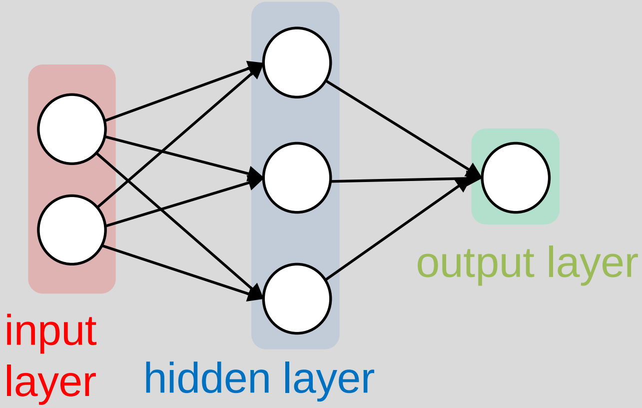
$$Task, T(\boldsymbol{x_i}) \rightarrow y_i$$
$$inputs, \boldsymbol{x_i}$$
$$outputs, y_i$$
$$Model, f(\boldsymbol{x_i};\ \boldsymbol{\theta}) \rightarrow \hat{y}_i$$
$$f \sim T$$

TRAINING A NEURAL NETWORK



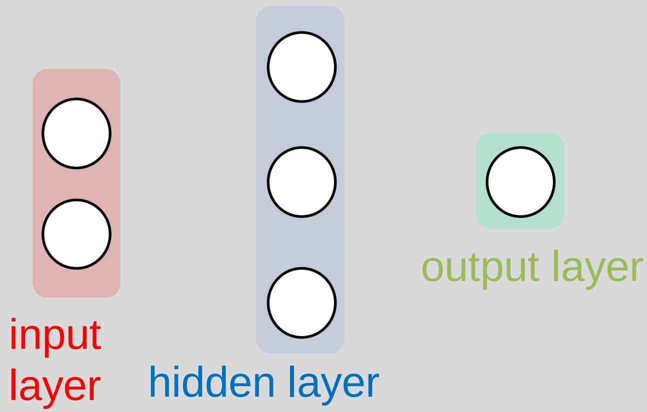
$$Model, f(\boldsymbol{x_i};\ \boldsymbol{\theta}) \rightarrow \hat{y}_i$$

TRAINING A NEURAL NETWORK



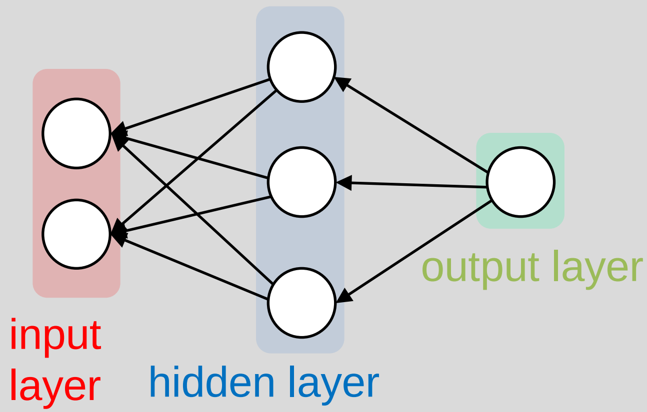
$$Model, f(\boldsymbol{x_i};\ \boldsymbol{\theta}) \rightarrow \hat{y}_i$$

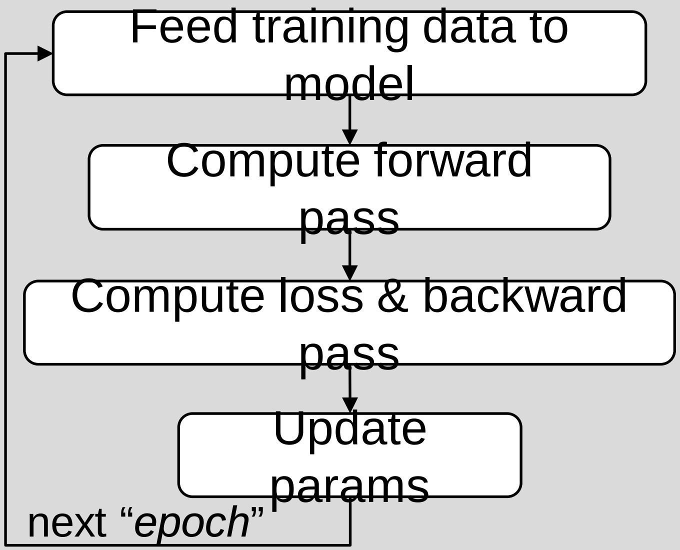
TRAINING A NEURAL NETWORK



$$Model, f(\boldsymbol{x_i};\, \boldsymbol{\theta}) \rightarrow \hat{y}_i$$

TRAINING A NEURAL NETWORK: Part 1





$$Model, f(\boldsymbol{x_i};\ \boldsymbol{\theta}) \rightarrow \hat{y}_i$$

$$\hat{y}_i \sim y_i$$

$$\min L(\hat{y}_i\,, y_i)$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \boldsymbol{\alpha}\frac{\boldsymbol{dL}}{\boldsymbol{d\theta}}$$

TRAINING A NEURAL NETWORK: Part 2

$T\!: \boldsymbol{x_i} \rightarrow \boldsymbol{x_i^2}$

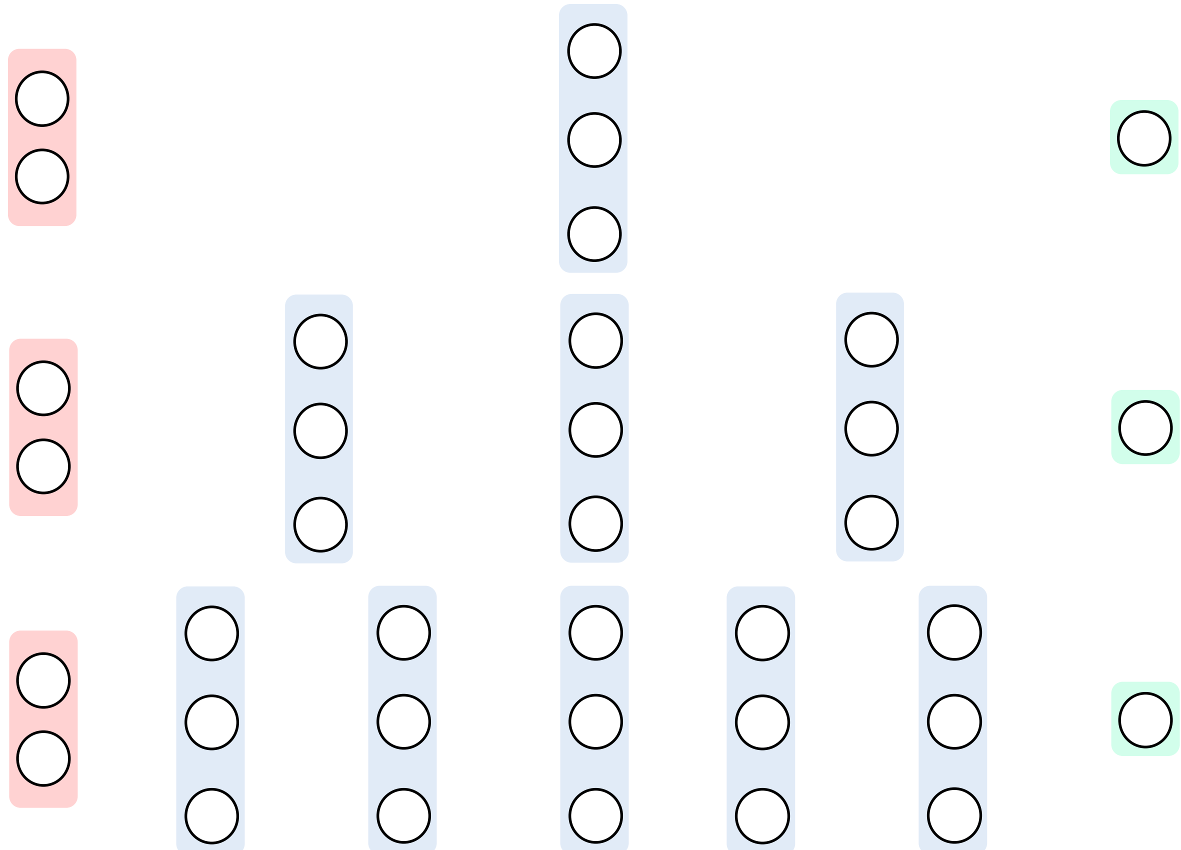
$f(\boldsymbol{x_i};\ \boldsymbol{\theta})$

$T\!: \boldsymbol{x_i} \rightarrow \boldsymbol{x_i^2} + |\boldsymbol{x_i}|$

$f(\boldsymbol{x_i};\ \boldsymbol{\theta})$

$T\!: \boldsymbol{x_i} \rightarrow \boldsymbol{x_i^2} + \frac{|\boldsymbol{x_i}|}{\max(\boldsymbol{x_i})}$

$f(\boldsymbol{x_i};\ \boldsymbol{\theta})$



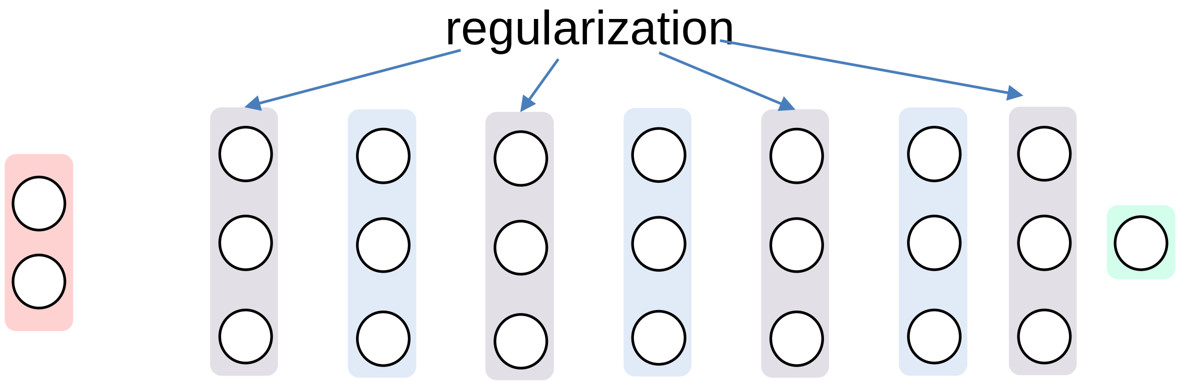
$C(f) = 1$

$C(f) = 3$

$C(f) = 5$

Capacity matters!

TRAINING A NEURAL NETWORK: Part 3



Find enough capacity, then regularize

DEBUGGING: "Do not go gentle into that good night"

```python
def my_function1(x):

  return x**(1/2) - 16

def my_function2(x):

  return (100 - 3*x) / (8**2 - x**6)**2

x = my_function1(324)

y = my_function2(x)
```

Expect: 4? 3.7? -100?

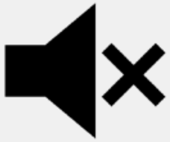
```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-14-ceb1744f59be> in <module>
      9 x = my_function1(324)
     10 
---> 11 y = my_function2(x)

<ipython-input-14-ceb1744f59be> in my_function2(x)
      5 def my_function2(x):
      6 
----> 7         return (100 - 3*x) / (8**2 - x**6)**2
      8 
      9 x = my_function1(324)

ZeroDivisionError: float division by zero
```

DEBUGGING: “Do not go gentle into that good night”

```
model = MyModel()

training_features, training_labels = get_training_data()

best_mape = 0
for epoch in range(100):
  # do forward pass, backward pass, update weights, return MAPE
  epoch_mape = trainer(model, training_features, training_labels)
  if epoch_mape > best_mape:
    best_mape = epoch_mape

print(best_mape)
```

Expect: 5%? ~0%?

Get: 107.15%



Silent Failure

NetDebugger

Contains **five** tests

- Inspired by Andrej Karpathy’s blog post, *“A Recipe for Training Neural Networks”*
- Buggy scripts that fail tests, will have a helpful error message returned
- Written for *PyTorch*

NetDebugger: Test #1, Output Shape

Test 1: Passes if the shape of the model output matches the shape of the training labels

$$L(\widehat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{N}\sum_{i=0}^{N} |\widehat{y_i} - y_i|$$

Case 1: Matching shapes

$$\widehat{\boldsymbol{y}} = [[1,2]]$$
$$\boldsymbol{y} = [[1,2]]$$
$$\widehat{\boldsymbol{y}} - \boldsymbol{y} = [[0,0]]$$
$$L = 0$$

Case 2: Mismatching shapes

$$\widehat{\boldsymbol{y}} = [[1,2]]$$
$$\boldsymbol{y} = [[1],[2]]$$
$$\widehat{\boldsymbol{y}} - \boldsymbol{y} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$
$$L = 1 \neq 0$$

Why does this happen? … Broadcasting!

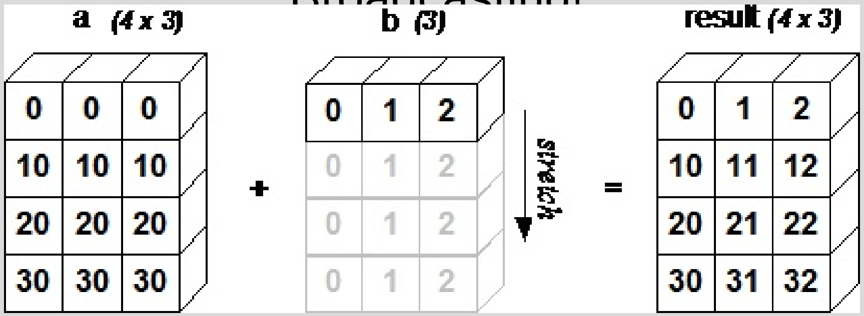
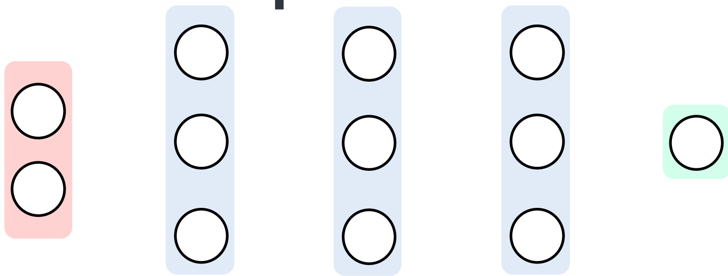


Image credit: https://www.tutorialspoint.com/numpy/numpy_broadcasting.htm

NetDebugger: Test #2, Input Independent **B**aseline

$f(\boldsymbol{x_i};\ \boldsymbol{\theta})$



$\boldsymbol{x_i}$	y_i
0	0
1	1
2	4

Case 1: Use real features of entire data

$\boldsymbol{x_i}$	y_i
0	0
1	1
2	4

There exist many functions that correctly relate this data, so we should be able to find some model that has a low loss $L_{dependent}$

Case 2: Zero out the features of entire data

$\boldsymbol{x_i}$	y_i
0	0
0	1
0	4

There is no function that can correctly relate this data, so all models should have a relatively high loss $L_{independent}$

Test 2: Passes if $L_{dependent} << L_{independent}$ after several epochs

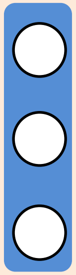
NetDebugger: Test #3, Overfit Small Batch

Test 3: Passes if a small batch of data (e.g., 10 points) can be completely overfit

$$\boldsymbol{\theta_2} \leftarrow \boldsymbol{\theta_1} - \boldsymbol{\alpha}\frac{\boldsymbol{dL}}{\boldsymbol{d\theta}}$$

$$\frac{\boldsymbol{dL}}{\boldsymbol{d\theta}} = \boldsymbol{0} \rightarrow \boldsymbol{\theta_1} = \boldsymbol{\theta_2}$$

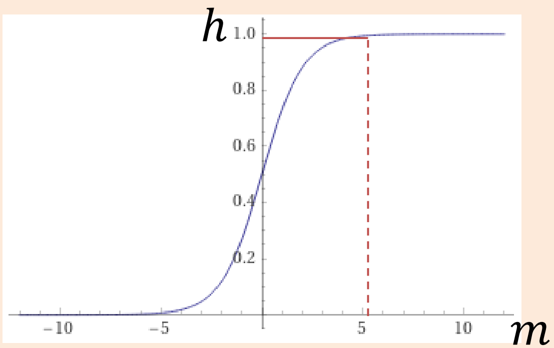
We need non-negligible gradients so that the model can improve



$h(\boldsymbol{x_i})$

$$h(\boldsymbol{x_i}) = NonLinearity(MatMul(\boldsymbol{x_i}, \boldsymbol{\theta}))$$

Case 1: Sigmoid non-Linearity



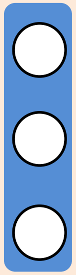
NetDebugger: Test #3, Overfit Small **B**atch

<u>Test 3:</u> Passes if a small batch of data (e.g., 10 points) can be completely overfit

$$\boldsymbol{\theta_2} \leftarrow \boldsymbol{\theta_1} - \boldsymbol{\alpha}\frac{\boldsymbol{dL}}{\boldsymbol{d\theta}}$$

$$\frac{\boldsymbol{dL}}{\boldsymbol{d\theta}} = \boldsymbol{0} \rightarrow \boldsymbol{\theta_1} = \boldsymbol{\theta_2}$$

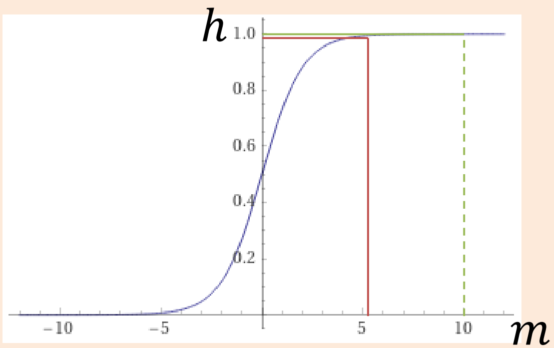
We need non-negligible gradients so that the model can improve



$h(\boldsymbol{x_i})$

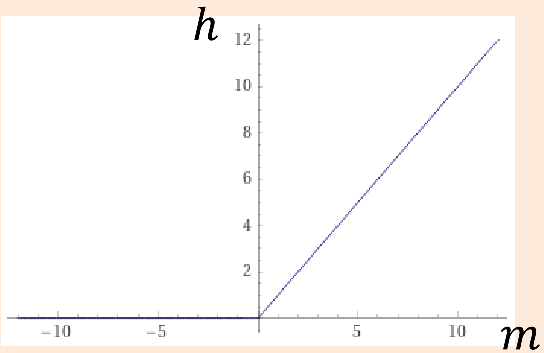
$$h(\boldsymbol{x_i}) = NonLinearity(MatMul(\boldsymbol{x_i}, \boldsymbol{\theta}))$$

Case 1: Sigmoid non-Linearity



Saturation at **both** limits leads to bad gradients

Case 2: ReLU non-Linearity



One limit is not saturated, better gradients

NetDebugger: Test #4, Chart Dependencies

$$X = \begin{bmatrix} 10 & 8 \\ 5 & 4 \end{bmatrix} ; \ \theta = \begin{bmatrix} 1 & 2 & 2 \\ 1 & 2 & 2 \end{bmatrix}$$

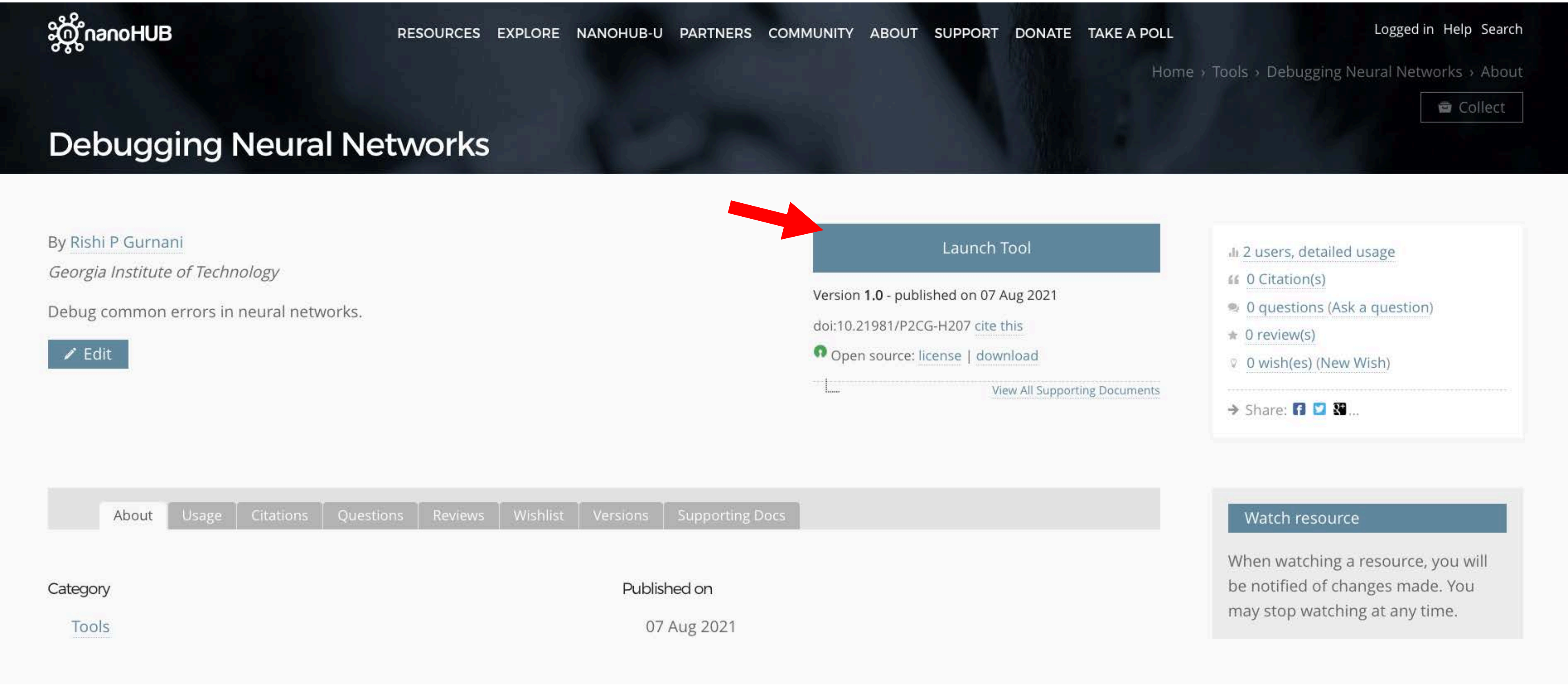
$$M = X\theta = \begin{bmatrix} 10 * 1 + 8 * 1 & - & - \\ 5 * 1 + 4 * 1 & - & - \end{bmatrix} \qquad M = X^T\theta = \begin{bmatrix} 10 * 1 + 5 * 1 & - & - \\ 8 * 1 + 4 * 1 & - & - \end{bmatrix}$$

<u>Test 4:</u> Passes if information from separate instances are not mixed

NetDebugger: Test #5, Overfit Entire Training Data

<u>Test 5:</u> Returns the capacity of “smallest” model that can overfit the entire training data

Accessing NetDebugger tutorial on nanoHUB



`https://nanohub.org/tools/netdebugger`

SUMMARY

- Neural networks offer great flexibility but this flexibility leads to silent failure. Efficiently debugging NNs requires forward thinking on what could go wrong during training.

- Some of this forward thinking has been encoded in *NetDebugger*

- *NetDebugger* is useful but by no means comprehensive. More checks and flexibility can be added.

`https://github.com/rishigurnani/nndebugger`

ACKNOWLEDGEMENTS

Past Group Members: Rohit Batra (Argonne), Arun M.K. (Purdue), Ghanshyam Pilania (LANL), Vinit Sharma (ORNL), Chenchen Wang (Amgen), Venkatesh Botu (Corning), Deya Das (Mahindra), Anurag Jha (IIT-K), Abhirup Patra (U. Penn), Anand Chandrasekaran (Schrodinger)

Special Thanks To: Rampi Ramprasad

`https://rishigurnani.wordpress.com/`   `rgurnani96@gatech.edu`