Expressivity, Complexity, Learnability

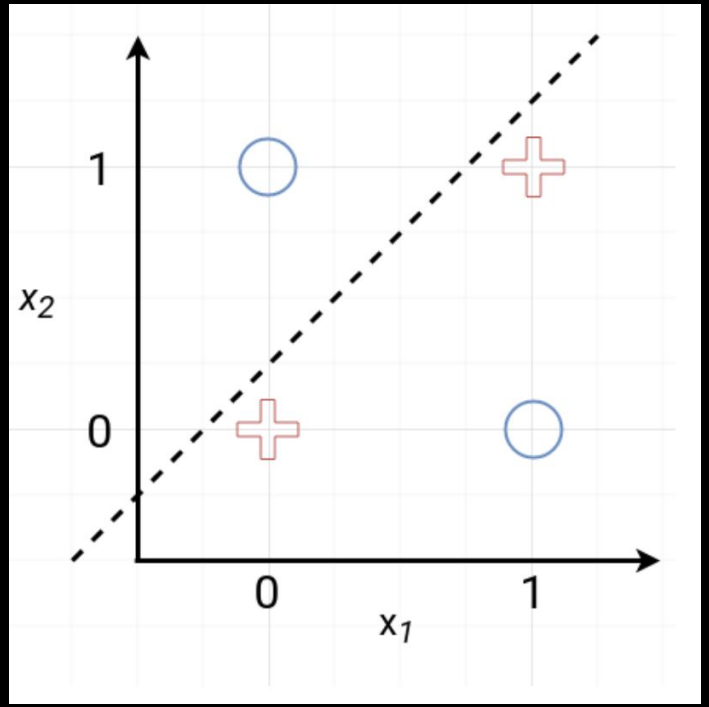
Failures of Gradient-Based Deep Learning, 2017

Shai Shalev-Shwartz, Ohad Shamir, and Shaked Shammah

Martin Weiss

PhD Student

XOR

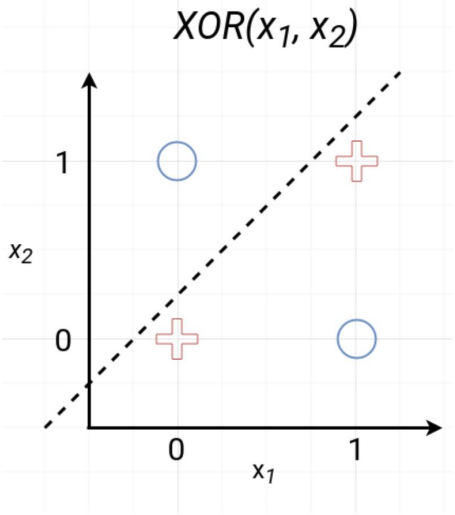


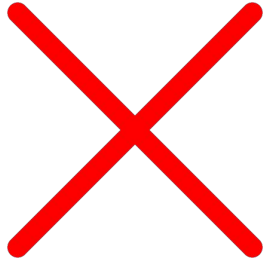
EGGSOR

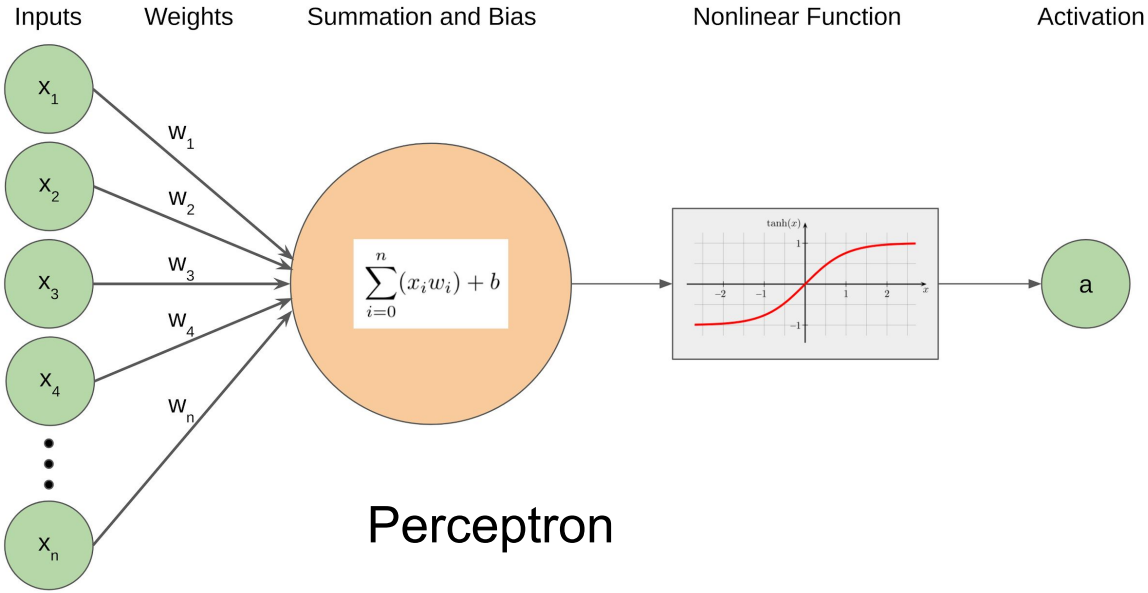
Agenda:

1. Expressivity and Complexity
2. Failures of Gradient Descent (Learning Parity + Linear-Periodic Functions)
3. inductive bias

Input		Output
A	B	A xor B
0	0	0
0	1	1
1	0	1
1	1	0



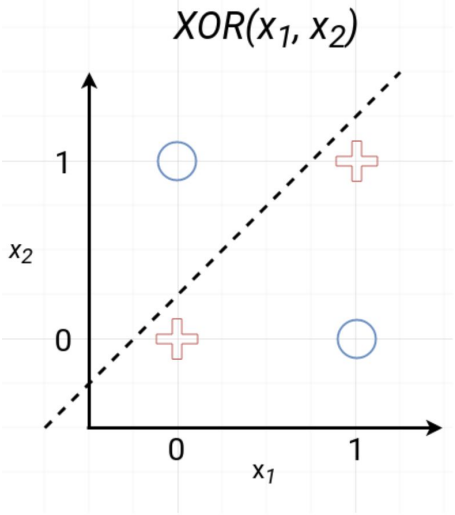


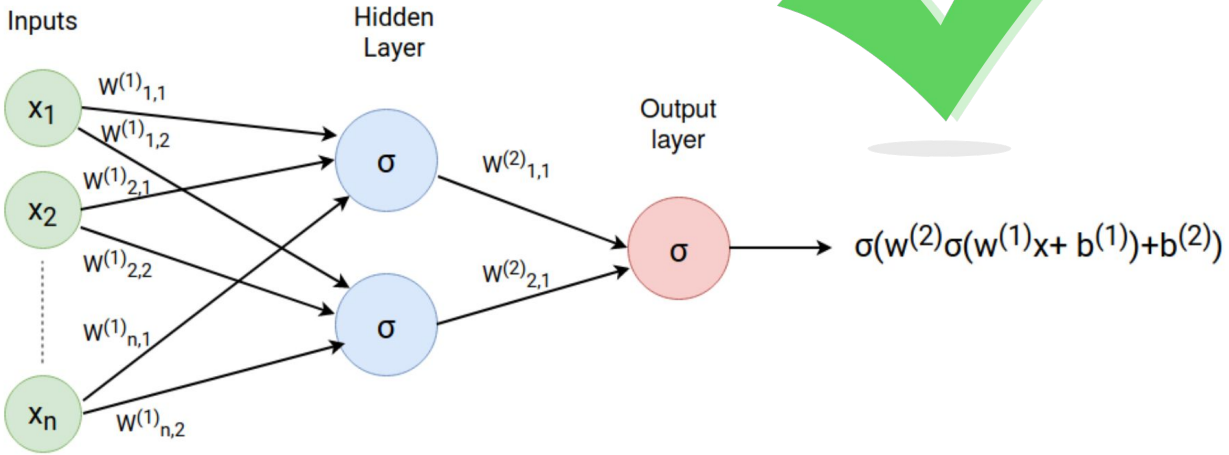


Expressivity: Recall that this is **not** going to work

Mila

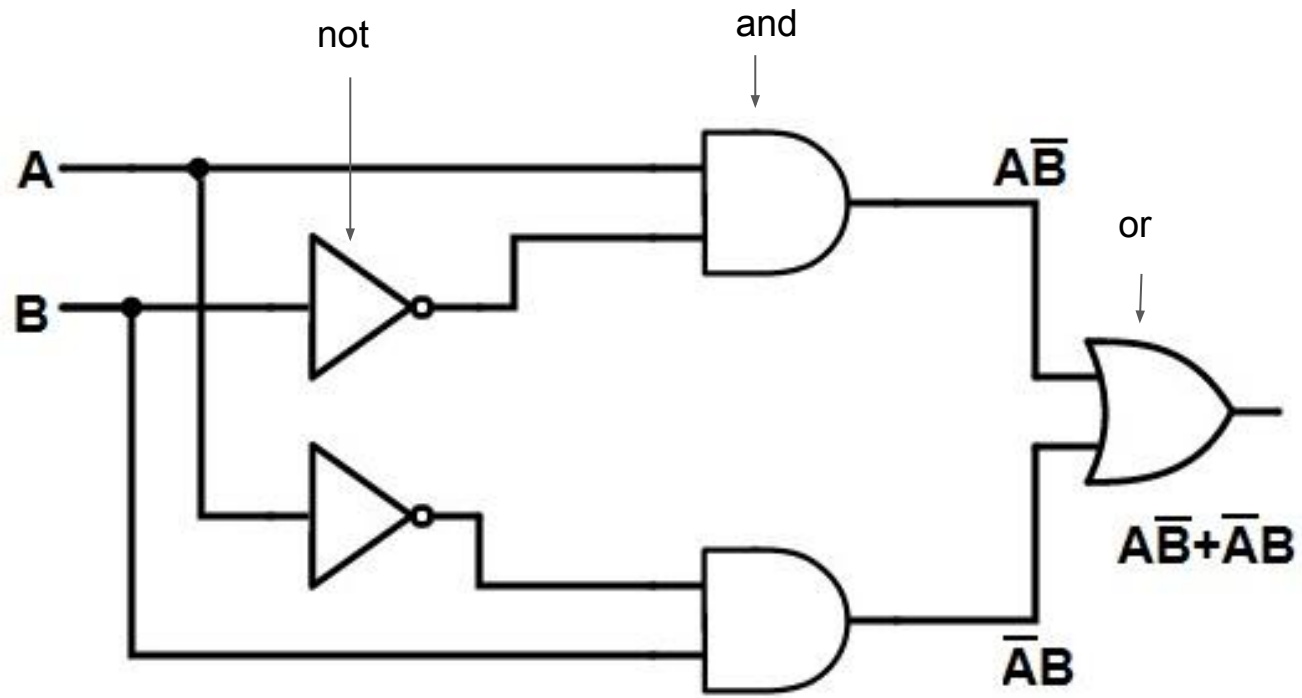
Input		Output
A	B	A xor B
0	0	0
0	1	1
1	0	1
1	1	0





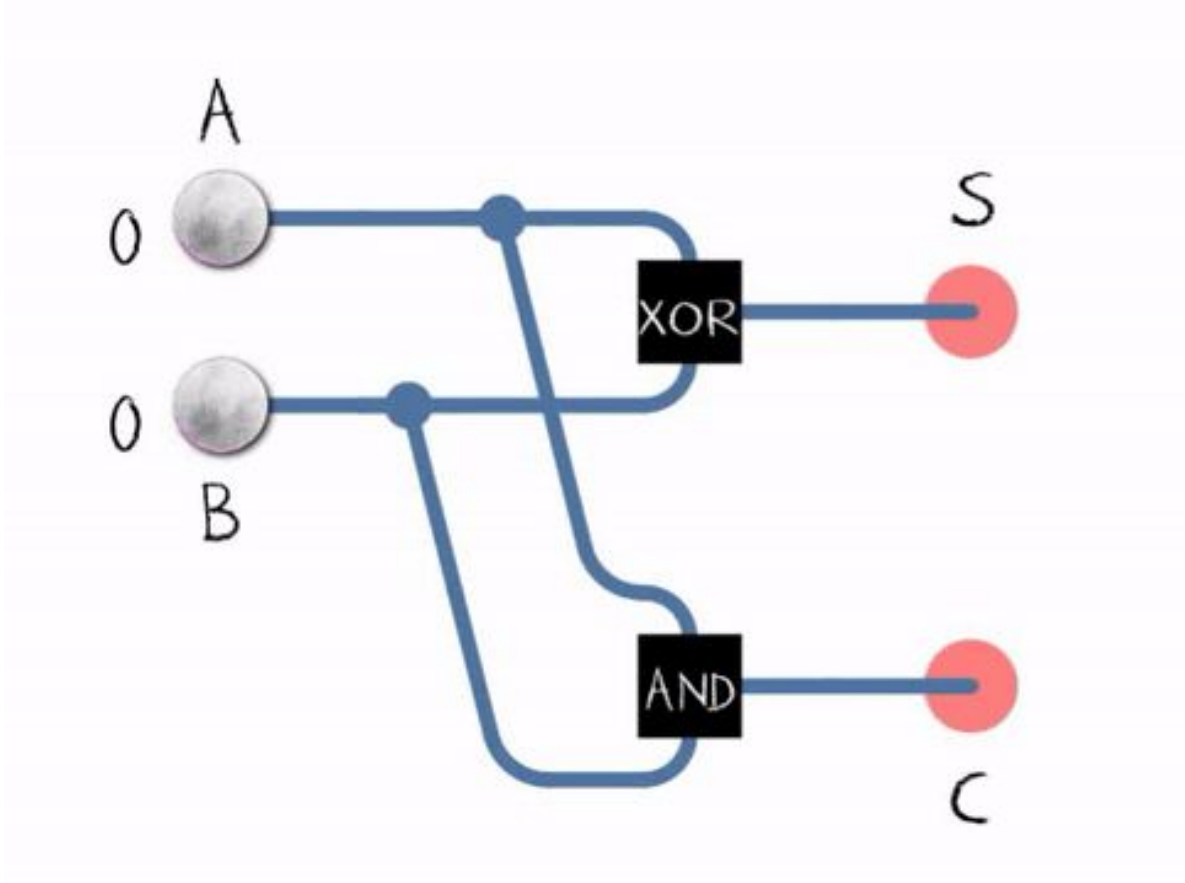
Expressivity: Recall that this is going to work

Mila



Expressivity: XOR circuit complexity

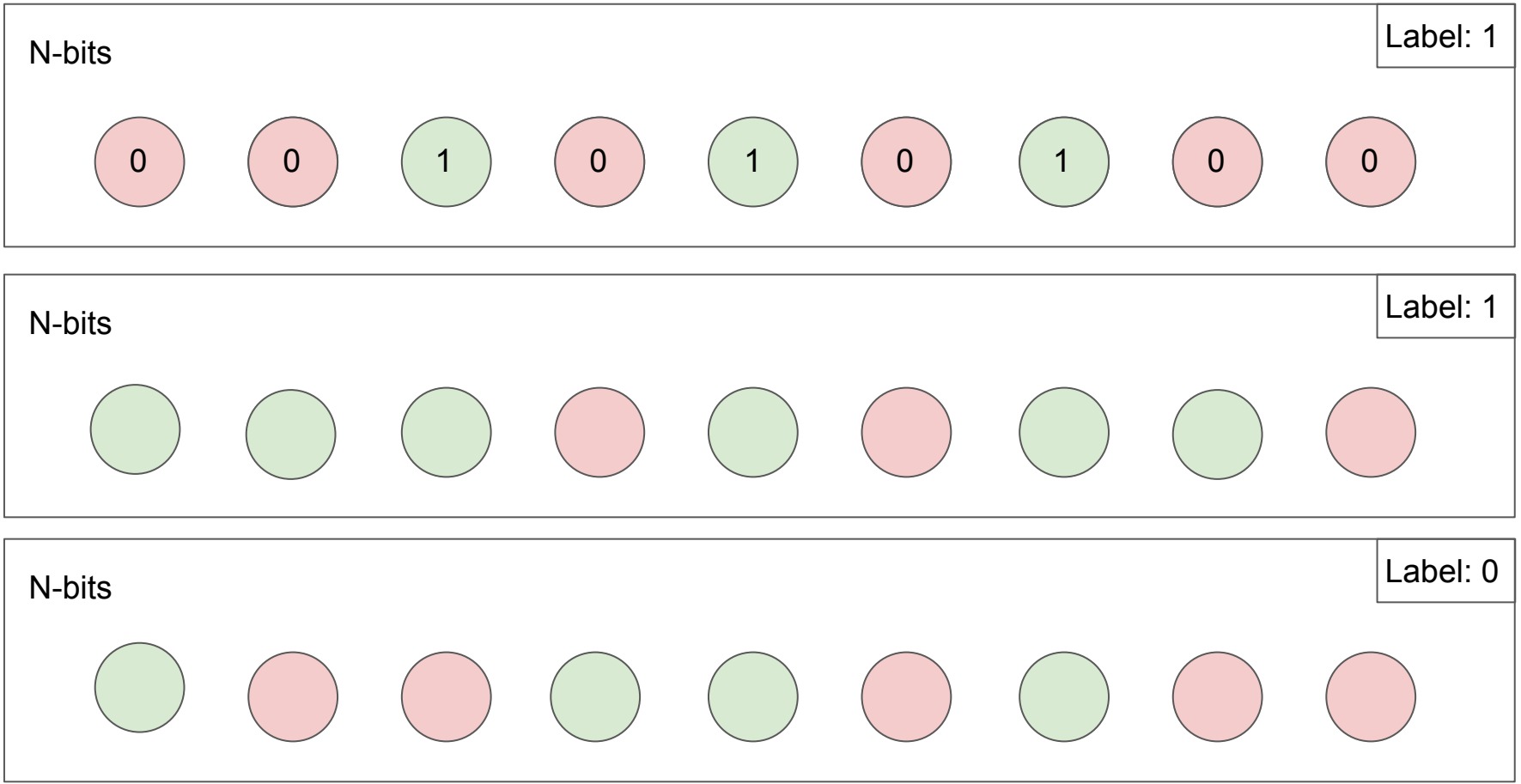
Mila



https://en.wikipedia.org/wiki/Adder_(electronics)#/media/File:Halfadder.gif

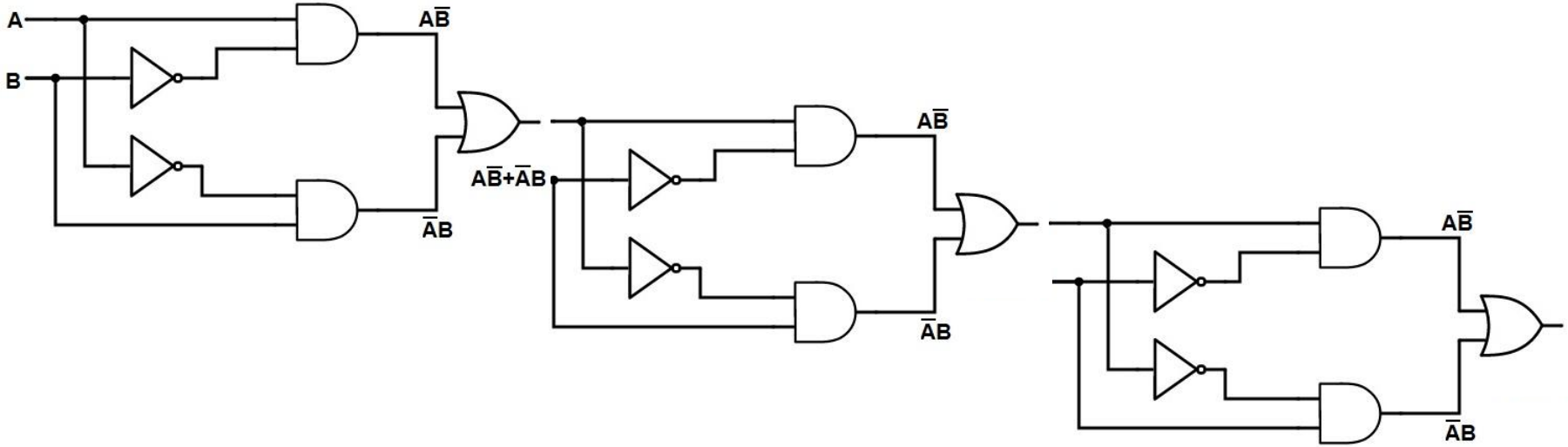
Expressivity: Half Adder

Mila



Expressivity: XOR is like N-bit Parity

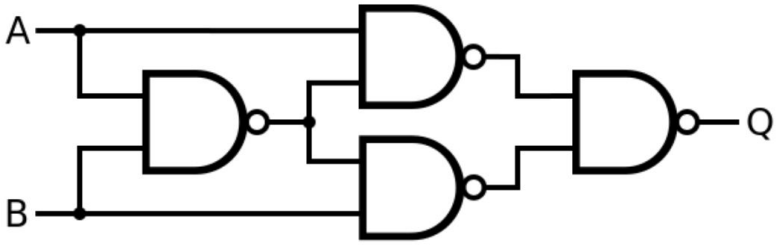
Mila



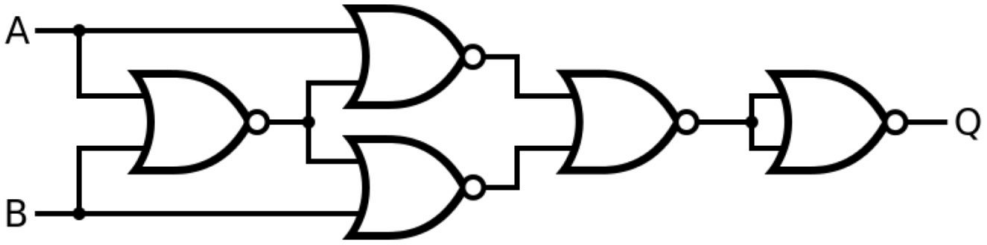
Expressivity: N-Bit Parity can be solved with iterated XORs

Mila

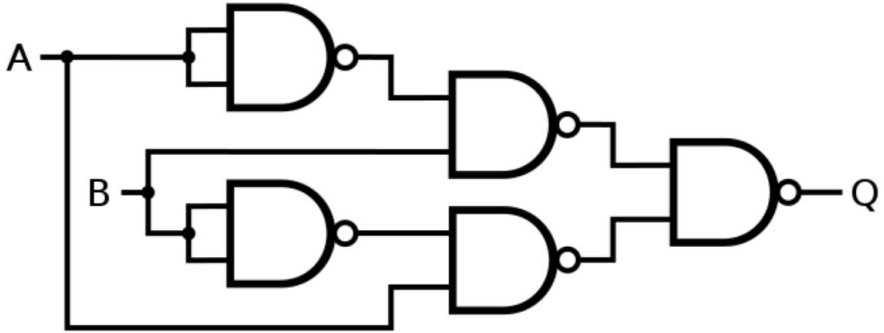
NAND construction



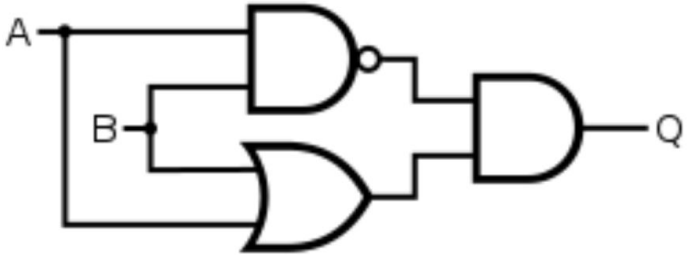
NOR construction

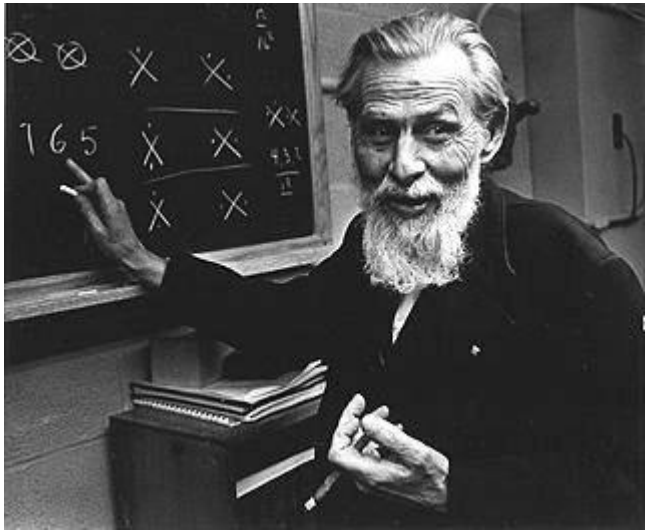


NAND construction



Alternative And / Or / Not Construction



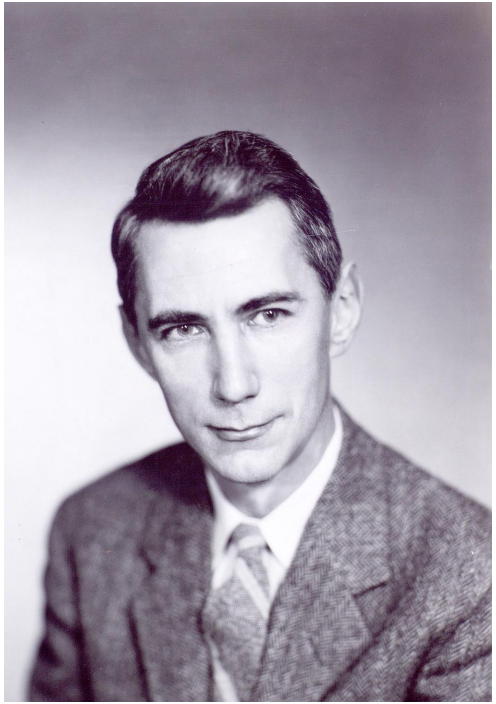


Warren Mcculloch
1943

Walter Pitts
1943

Historical Interlude: Mathematical Model of a Neuron

Mila

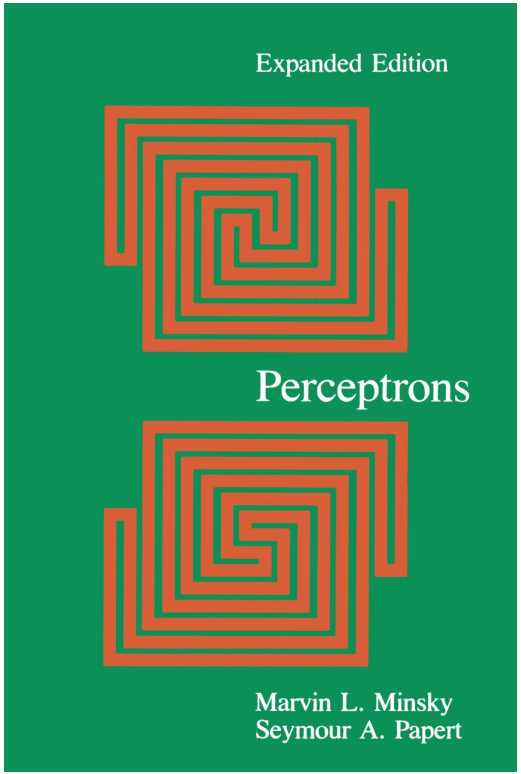


Most Boolean Functions
Require Exponential Circuit Complexity
Claude Shannon
1949

Historical Interlude: Boolean Function Complexity

Mila

Perceptrons
Marvin Minsky & Seymour Papert
1969



Historical Interlude: Parity and Neural Networks

Mila

Any Constant-depth Circuit Computing Parity Requires Exponential Size

John Håstad

1987

Historical Interlude: Boolean Circuit Size

Mila

THEOREM 20.2 *For every n, let $s(n)$ be the minimal integer such that there exists a graph (V, E) with $|V| = s(n)$ such that the hypothesis class $\mathcal{H}_{V,E,sign}$ contains all the functions from $\{0,1\}^n$ to $\{0,1\}$. Then, $s(n)$ is exponential in n. Similar results hold for $\mathcal{H}_{V,E,\sigma}$ where σ is the sigmoid function.*

Understanding Machine Learning: From Theory to Algorithms
Shai Ben-David and Shai Shalev-Shwartz, 2017

THEOREM 20.3 *Let $T : \mathbb{N} \to \mathbb{N}$ and for every n, let $\mathcal{F}_n$ be the set of functions that can be implemented using a Turing machine using runtime of at most $T(n)$. Then, there exist constants $b, c \in \mathbb{R}_+$ such that for every n, there is a graph (V_n, E_n) of size at most $cT(n)^2 + b$ such that $\mathcal{H}_{V_n, E_n, sign}$ contains $\mathcal{F}_n$.*

Understanding Machine Learning: From Theory to Algorithms
Shai Ben-David and Shai Shalev-Shwartz, 2017

Relation of Space and Time Complexity for Boolean Functions

Mila

- **Shallow Network**: exponential params
- **Deep Network**: linear params ($\frac{3n}{2}$ with relu and fc output)
- **RNN**: constant params

Sawtooths

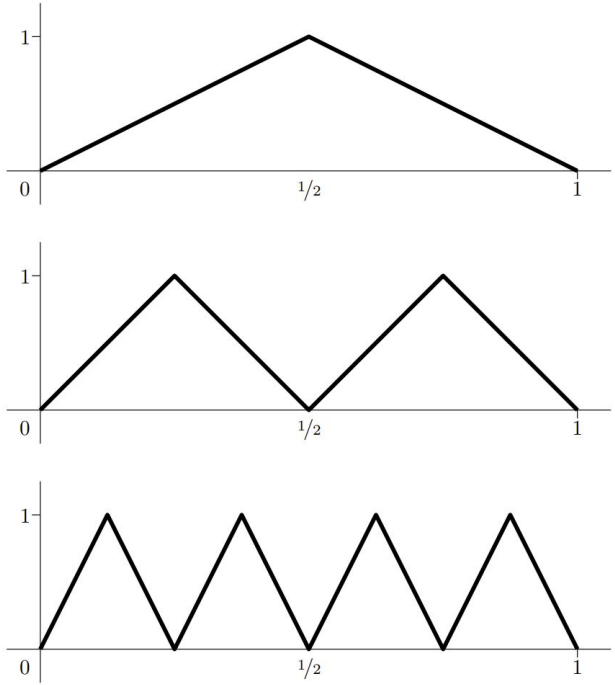


Figure 2: f_{m}, f_{m}^2, and f_{m}^3.

Representation Benefits of Deep Feedforward Networks
Mattus Telgarsky, 2015

Expressivity: Parameter Blowup

Mila

Ok! So RNNs are very expressive! Are we done?

Failures of Gradient-Based Deep Learning

https://arxiv.org/pdf/1703.07950.pdf

Shai Shalev-Shwartz, Ohad Shamir, Shaked Shammah. 2017

Mila

Expressivity is not always enough,
Learnability and inductive bias matter!

Mila

We’ll prove this for Parity and Linear Periodic Functions

N.B. The following results are true *independent of network architecture*

Mila

Goal: learn some hypothesis class $h \in \mathcal{H}$ responsible for labelling the data

Problem Setting: (Recall ERM)

$$\min_{\mathbf{w}} F_h(\mathbf{w}) := \mathop{\mathbb{E}}_{\mathbf{x}} \left[\ell(p_{\mathbf{w}}(\mathbf{x}), h(\mathbf{x})) \right]$$

$\mathbf{x}$ are the stochastic inputs (assumed to be vectors in Euclidean space)

$p_{\mathbf{w}}$ is some predictor parameterized by a parameter vector $\mathbf{w}$

Mila

Problem Setting: (Recall ERM)

$$\min_{\mathbf{w}} F_h(\mathbf{w}) := \mathop{\mathbb{E}}_{\mathbf{x}} \left[\ell(p_{\mathbf{w}}(\mathbf{x}), h(\mathbf{x})) \right]$$

$\mathbf{x}$ are the stochastic inputs (assumed to be vectors in Euclidean space)

$p_{\mathbf{w}}$ is some predictor parameterized by a parameter vector $\mathbf{w}$

Assumes F is differentiable and that $\nabla F_h(\mathbf{w})$ contains useful information

What does it mean for the gradient to contain useful information about h?

What does it mean for the gradient to contain useful information about h?

$$\|\nabla F_h(\mathbf{w}) - \nabla F_{h'}(\mathbf{w})\|^2$$

Theorem 1 *Suppose that*

- $\mathcal{H}$ *consists of real-valued functions* h *satisfying* $\mathbb{E}_{\mathbf{x}}[h^2(\mathbf{x})] \leq 1$*, such that for any two distinct* $h, h' \in \mathcal{H}$*,* $\mathbb{E}_{\mathbf{x}}[h(\mathbf{x})h'(\mathbf{x})] = 0$*.*
- $p_{\mathbf{w}}(\mathbf{x})$ *is differentiable w.r.t.* $\mathbf{w}$*, and satisfies* $\mathbb{E}_{\mathbf{x}}\left[\|\frac{\partial}{\partial \mathbf{w}} p_{\mathbf{w}}(\mathbf{x})\|^2\right] \leq G(\mathbf{w})^2$ *for some scalar* $G(\mathbf{w})$*.*
- *The loss function* ℓ *in* (1) *is either the square loss* $\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$ *or a classification loss of the form* $\ell(\hat{y}, y) = r(\hat{y} \cdot y)$ *for some* 1*-Lipschitz function* r*, and the target function* h *takes values in* $\{\pm 1\}$*.*

Then

$$Var(\mathcal{H}, F, \mathbf{w}) \leq \frac{G(\mathbf{w})^2}{|\mathcal{H}|}.$$

Assumption 1: Target functions are orthonormal

- *$\mathcal{H}$ consists of real-valued functions h satisfying $\mathbb{E}_{\mathbf{x}}[h^2(\mathbf{x})] \leq 1$, such that for any two distinct $h, h' \in \mathcal{H}$, $\mathbb{E}_{\mathbf{x}}[h(\mathbf{x})h'(\mathbf{x})] = 0$.*

$$Var(\mathcal{H}, F, \mathbf{w}) \leq \frac{G(\mathbf{w})^2}{|\mathcal{H}|}.$$

Mila

Assumption 2: Bounded gradient

- $p_{\mathbf{w}}(\mathbf{x})$ *is differentiable w.r.t.* $\mathbf{w}$*, and satisfies* $\mathbb{E}_{\mathbf{x}}\left[\|\frac{\partial}{\partial \mathbf{w}} p_{\mathbf{w}}(\mathbf{x})\|^2\right] \leq G(\mathbf{w})^2$ *for some scalar* $G(\mathbf{w})$.

$$Var(\mathcal{H}, F, \mathbf{w}) \leq \frac{G(\mathbf{w})^2}{|\mathcal{H}|}.$$

Mila

Assumption 3: Square loss or classification loss

- *The loss function ℓ in (1) is either the square loss $\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$ or a classification loss of the form $\ell(\hat{y}, y) = r(\hat{y} \cdot y)$ for some 1-Lipschitz function r, and the target function h takes values in $\{\pm 1\}$.*

$$Var(\mathcal{H}, F, \mathbf{w}) \leq \frac{G(\mathbf{w})^2}{|\mathcal{H}|}.$$

Mila

Proof Idea

Quantity to investigate: Variance of gradient of F with respect to h

Assume: h drawn uniformly at random from collection of target functions $\mathcal{H}$

Show that if the functions in $\mathcal{H}$ are orthonormal then, for every $\mathbf{w}$,

$$Var(\mathcal{H}, F, \mathbf{w}) := \mathbb{E}_h \|\nabla F_h(\mathbf{w}) - \mathbb{E}_{h'} \nabla F_{h'}(\mathbf{w})\|^2 = O\left(\frac{1}{|\mathcal{H}|}\right)$$

To do so, express each coordinate of $\nabla p_{\mathbf{w}}(\mathbf{x})$ using orthonormal fns in $\mathcal{H}$

Mila

Proof Given two square-integrable functions f, g on an Euclidean space $\mathbb{R}^n$, let $\langle f, g \rangle_{L_2} = \mathbb{E}_{\mathbf{x}}[f(\mathbf{x})g(\mathbf{x})]$ and $\|f\|_{L_2} = \sqrt{\mathbb{E}_{\mathbf{x}}[f^2(\mathbf{x})]}$ denote inner product and norm in the L_2 space of square-integrable functions (with respect to the relevant distribution). Also, define the vector-valued function

$$\mathbf{g}(\mathbf{x}) = \frac{\partial}{\partial \mathbf{w}} p_{\mathbf{w}}(\mathbf{x}),$$

and let $\mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_n(\mathbf{x}))$ for real-valued functions $g_1, \ldots, g_n$. Finally, let $\mathbb{E}_h$ denote an expectation with respect to h chosen uniformly at random from $\mathcal{H}$. Let $|\mathcal{H}| = d$.

Proof Setup for Theorem 1: (Squared Loss)

We begin by proving the result for the squared loss. To prove the bound, it is enough to show that $\mathbb{E}_h \|\nabla F_h(\mathbf{w}) - \mathbf{a}\|^2 \leq \frac{G^2}{|\mathcal{H}|}$ for any vector $\mathbf{a}$ independent of h. In particular, let us choose $\mathbf{a} = \mathbb{E}_{\mathbf{x}}[p_{\mathbf{w}}(\mathbf{x})\mathbf{g}(\mathbf{x})]$.

Mila

Proof of Theorem 1: (Squared Loss)

We thus bound the following:

$$\mathop{\mathbb{E}}_{h} \|\nabla F_h(\mathbf{w}) - \mathop{\mathbb{E}}_{\mathbf{x}} [p_{\mathbf{w}}(\mathbf{x})\mathbf{g}(\mathbf{x})] \|^2 = \mathop{\mathbb{E}}_{h} \| \mathop{\mathbb{E}}_{\mathbf{x}} [(p_{\mathbf{w}}(\mathbf{x}) - h(\mathbf{x}))\, \mathbf{g}(\mathbf{x})] - \mathop{\mathbb{E}}_{\mathbf{x}} [p_{\mathbf{w}}(\mathbf{x})\mathbf{g}(\mathbf{x})] \|^2$$

Recall:

$p_{\mathbf{w}}(\mathbf{x})$ is the output of our predictive model

$\mathbf{g}(\mathbf{x})$ is the gradient with respect to $p_{\mathbf{w}}(\mathbf{x})$

$F_h(\mathbf{w}) := \mathop{\mathbb{E}}_{\mathbf{x}} [\ell(p_{\mathbf{w}}(\mathbf{x}), h(\mathbf{x}))]$

Mila

We thus bound the following:

$$\mathop{\mathbb{E}}_{h} \|\nabla F_h(\mathbf{w}) - \mathop{\mathbb{E}}_{\mathbf{x}}[p_{\mathbf{w}}(\mathbf{x})\mathbf{g}(\mathbf{x})]\|^2 = \mathop{\mathbb{E}}_{h} \|\mathop{\mathbb{E}}_{\mathbf{x}}[(p_{\mathbf{w}}(\mathbf{x}) - h(\mathbf{x}))\,\mathbf{g}(\mathbf{x})] - \mathop{\mathbb{E}}_{\mathbf{x}}[p_{\mathbf{w}}(\mathbf{x})\mathbf{g}(\mathbf{x})]\|^2$$

$$= \mathop{\mathbb{E}}_{h} \|\mathop{\mathbb{E}}_{\mathbf{x}}[h(\mathbf{x})\mathbf{g}(\mathbf{x})]\|^2 = \mathop{\mathbb{E}}_{h} \sum_{j=1}^{n} \left(\mathop{\mathbb{E}}_{\mathbf{x}}[h(\mathbf{x})g_j(\mathbf{x})]\right)^2$$

We thus bound the following:

$$\mathop{\mathbb{E}}_{h} \|\nabla F_h(\mathbf{w}) - \mathop{\mathbb{E}}_{\mathbf{x}}[p_{\mathbf{w}}(\mathbf{x})\mathbf{g}(\mathbf{x})]\|^2 = \mathop{\mathbb{E}}_{h} \|\mathop{\mathbb{E}}_{\mathbf{x}}[(p_{\mathbf{w}}(\mathbf{x}) - h(\mathbf{x}))\,\mathbf{g}(\mathbf{x})] - \mathop{\mathbb{E}}_{\mathbf{x}}[p_{\mathbf{w}}(\mathbf{x})\mathbf{g}(\mathbf{x})]\|^2$$

$$= \mathop{\mathbb{E}}_{h} \|\mathop{\mathbb{E}}_{\mathbf{x}}[h(\mathbf{x})\mathbf{g}(\mathbf{x})]\|^2 = \mathop{\mathbb{E}}_{h} \sum_{j=1}^{n} \left(\mathop{\mathbb{E}}_{\mathbf{x}}[h(\mathbf{x})g_j(\mathbf{x})]\right)^2$$

$$= \mathop{\mathbb{E}}_{h} \sum_{j=1}^{n} \langle h, g_j \rangle_{L_2}^2$$

Recall:

$$\langle f, g \rangle_{L_2} = \mathbb{E}_{\mathbf{x}}[f(\mathbf{x})g(\mathbf{x})]$$

We thus bound the following:

$$\mathop{\mathbb{E}}_{h} \|\nabla F_h(\mathbf{w}) - \mathop{\mathbb{E}}_{\mathbf{x}}[p_{\mathbf{w}}(\mathbf{x})\mathbf{g}(\mathbf{x})]\|^2 = \mathop{\mathbb{E}}_{h} \|\mathop{\mathbb{E}}_{\mathbf{x}}[(p_{\mathbf{w}}(\mathbf{x}) - h(\mathbf{x}))\,\mathbf{g}(\mathbf{x})] - \mathop{\mathbb{E}}_{\mathbf{x}}[p_{\mathbf{w}}(\mathbf{x})\mathbf{g}(\mathbf{x})]\|^2$$

$$= \mathop{\mathbb{E}}_{h} \|\mathop{\mathbb{E}}_{\mathbf{x}}[h(\mathbf{x})\mathbf{g}(\mathbf{x})]\|^2 = \mathop{\mathbb{E}}_{h} \sum_{j=1}^{n} \left(\mathop{\mathbb{E}}_{\mathbf{x}}[h(\mathbf{x})g_j(\mathbf{x})]\right)^2$$

$$= \mathop{\mathbb{E}}_{h} \sum_{j=1}^{n} \langle h, g_j\rangle_{L_2}^2 = \sum_{j=1}^{n} \left(\frac{1}{|\mathcal{H}|}\sum_{i=1}^{d} \langle h_i, g_j\rangle_{L_2}^2\right)$$

We thus bound the following:

$$\begin{aligned}
\mathop{\mathbb{E}}_{h} \| \nabla F_h(\mathbf{w}) - \mathop{\mathbb{E}}_{\mathbf{x}} [p_{\mathbf{w}}(\mathbf{x}) \mathbf{g}(\mathbf{x})] \|^2 &= \mathop{\mathbb{E}}_{h} \| \mathop{\mathbb{E}}_{\mathbf{x}} [(p_{\mathbf{w}}(\mathbf{x}) - h(\mathbf{x})) \mathbf{g}(\mathbf{x})] - \mathop{\mathbb{E}}_{\mathbf{x}} [p_{\mathbf{w}}(\mathbf{x}) \mathbf{g}(\mathbf{x})] \|^2 \\
&= \mathop{\mathbb{E}}_{h} \| \mathop{\mathbb{E}}_{\mathbf{x}} [h(\mathbf{x}) \mathbf{g}(\mathbf{x})] \|^2 = \mathop{\mathbb{E}}_{h} \sum_{j=1}^{n} \left(\mathop{\mathbb{E}}_{\mathbf{x}} [h(\mathbf{x}) g_j(\mathbf{x})] \right)^2 \\
&= \mathop{\mathbb{E}}_{h} \sum_{j=1}^{n} \langle h, g_j \rangle_{L_2}^2 = \sum_{j=1}^{n} \left(\frac{1}{|\mathcal{H}|} \sum_{i=1}^{d} \langle h_i, g_j \rangle_{L_2}^2 \right) \\
&\overset{(*)}{\leq} \sum_{j=1}^{n} \left(\frac{1}{|\mathcal{H}|} \| g_j \|_{L_2}^2 \right)
\end{aligned}$$

where $(*)$ follows from the functions in $\mathcal{H}$ being mutually orthogonal, and satisfying $\|h\|_{L_2} \leq 1$ for all $h \in \mathcal{H}$.

We thus bound the following:

$$\begin{aligned}
\mathop{\mathbb{E}}_{h} \| \nabla F_h(\mathbf{w}) - \mathop{\mathbb{E}}_{\mathbf{x}} [p_{\mathbf{w}}(\mathbf{x}) \mathbf{g}(\mathbf{x})] \|^2 &= \mathop{\mathbb{E}}_{h} \| \mathop{\mathbb{E}}_{\mathbf{x}} [(p_{\mathbf{w}}(\mathbf{x}) - h(\mathbf{x})) \mathbf{g}(\mathbf{x})] - \mathop{\mathbb{E}}_{\mathbf{x}} [p_{\mathbf{w}}(\mathbf{x}) \mathbf{g}(\mathbf{x})] \|^2 \\
&= \mathop{\mathbb{E}}_{h} \| \mathop{\mathbb{E}}_{\mathbf{x}} [h(\mathbf{x}) \mathbf{g}(\mathbf{x})] \|^2 = \mathop{\mathbb{E}}_{h} \sum_{j=1}^{n} \left(\mathop{\mathbb{E}}_{\mathbf{x}} [h(\mathbf{x}) g_j(\mathbf{x})] \right)^2 \\
&= \mathop{\mathbb{E}}_{h} \sum_{j=1}^{n} \langle h, g_j \rangle_{L_2}^2 = \sum_{j=1}^{n} \left(\frac{1}{|\mathcal{H}|} \sum_{i=1}^{d} \langle h_i, g_j \rangle_{L_2}^2 \right) \\
&\overset{(*)}{\leq} \sum_{j=1}^{n} \left(\frac{1}{|\mathcal{H}|} \| g_j \|_{L_2}^2 \right) = \frac{1}{|\mathcal{H}|} \sum_{j=1}^{n} \mathop{\mathbb{E}}_{\mathbf{x}} [g_j^2(\mathbf{x})]
\end{aligned}$$

Recall:

norm in the L_2 space of square-integrable functions $\|f\|_{L_2} = \sqrt{\mathbb{E}_{\mathbf{x}}[f^2(\mathbf{x})]}$

We thus bound the following:

$$\begin{aligned}
\mathop{\mathbb{E}}_{h} \|\nabla F_h(\mathbf{w}) - \mathop{\mathbb{E}}_{\mathbf{x}}[p_{\mathbf{w}}(\mathbf{x})\mathbf{g}(\mathbf{x})]\|^2 &= \mathop{\mathbb{E}}_{h} \|\mathop{\mathbb{E}}_{\mathbf{x}}[(p_{\mathbf{w}}(\mathbf{x}) - h(\mathbf{x}))\,\mathbf{g}(\mathbf{x})] - \mathop{\mathbb{E}}_{\mathbf{x}}[p_{\mathbf{w}}(\mathbf{x})\mathbf{g}(\mathbf{x})]\|^2 \\
&= \mathop{\mathbb{E}}_{h} \|\mathop{\mathbb{E}}_{\mathbf{x}}[h(\mathbf{x})\mathbf{g}(\mathbf{x})]\|^2 = \mathop{\mathbb{E}}_{h} \sum_{j=1}^{n} \left(\mathop{\mathbb{E}}_{\mathbf{x}}[h(\mathbf{x})g_j(\mathbf{x})]\right)^2 \\
&= \mathop{\mathbb{E}}_{h} \sum_{j=1}^{n} \langle h, g_j\rangle_{L_2}^2 = \sum_{j=1}^{n} \left(\frac{1}{|\mathcal{H}|}\sum_{i=1}^{d} \langle h_i, g_j\rangle_{L_2}^2\right) \\
&\overset{(*)}{\leq} \sum_{j=1}^{n} \left(\frac{1}{|\mathcal{H}|}\|g_j\|_{L_2}^2\right) = \frac{1}{|\mathcal{H}|}\sum_{j=1}^{n} \mathop{\mathbb{E}}_{\mathbf{x}}[g_j^2(\mathbf{x})] \\
&= \frac{1}{|\mathcal{H}|}\mathop{\mathbb{E}}_{\mathbf{x}}\left[\|\mathbf{g}(\mathbf{x})\|^2\right] \leq \frac{G(\mathbf{w})^2}{|\mathcal{H}|}
\end{aligned}$$

□

Recall:

- $p_{\mathbf{w}}(\mathbf{x})$ *is differentiable w.r.t.* $\mathbf{w}$, *and satisfies* $\mathbb{E}_{\mathbf{x}}\left[\|\frac{\partial}{\partial \mathbf{w}} p_{\mathbf{w}}(\mathbf{x})\|^2\right] \leq G(\mathbf{w})^2$ *for some scalar* $G(\mathbf{w})$.

Theorem 1 *Suppose that*

- $\mathcal{H}$ *consists of real-valued functions* h *satisfying* $\mathbb{E}_{\mathbf{x}}[h^2(\mathbf{x})] \leq 1$, *such that for any two distinct* $h, h' \in \mathcal{H}$, $\mathbb{E}_{\mathbf{x}}[h(\mathbf{x})h'(\mathbf{x})] = 0$.
- $p_{\mathbf{w}}(\mathbf{x})$ *is differentiable w.r.t.* $\mathbf{w}$, *and satisfies* $\mathbb{E}_{\mathbf{x}}\left[\|\frac{\partial}{\partial \mathbf{w}} p_{\mathbf{w}}(\mathbf{x})\|^2\right] \leq G(\mathbf{w})^2$ *for some scalar* $G(\mathbf{w})$.
- *The loss function* ℓ *in* (1) *is either the square loss* $\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$ *or a classification loss of the form* $\ell(\hat{y}, y) = r(\hat{y} \cdot y)$ *for some* 1*-Lipschitz function* r, *and the target function* h *takes values in* $\{\pm 1\}$.

Then

$$Var(\mathcal{H}, F, \mathbf{w}) \leq \frac{G(\mathbf{w})^2}{|\mathcal{H}|}.$$

Now let’s look at n-bit parity again

Formalizing the N-Bit Parity Problem

Hypothesis space:

$$\text{set of } 2^d \text{ functions } \mathcal{H} = \{\mathbf{x} \mapsto (-1)^{\langle \mathbf{x}, \mathbf{v}^* \rangle} : \mathbf{v}^* \in \{0,1\}^d\}$$

Problem Setup: find y indicating number of positive bits is even on some subset of x

$$\text{goal is to train a predictor mapping } \mathbf{x} \in \{0,1\}^d \text{ to } y = (-1)^{\langle \mathbf{x}, \mathbf{v}^* \rangle}$$

Mila

Applying Theorem 1 to the N-Bit Parity Problem

for any $\mathbf{v}, \mathbf{v}'$,

$$\mathop{\mathbb{E}}_{\mathbf{x}}\left[(-1)^{\langle \mathbf{x},\mathbf{v}\rangle}(-1)^{\langle \mathbf{x},\mathbf{v}'\rangle}\right] = \mathop{\mathbb{E}}_{\mathbf{x}}\left[(-1)^{\langle \mathbf{x},\mathbf{v}+\mathbf{v}'\rangle}\right]$$

- *$\mathcal{H}$ consists of real-valued functions h satisfying $\mathbb{E}_{\mathbf{x}}[h^2(\mathbf{x})] \leq 1$, such that for any two distinct $h, h' \in \mathcal{H}$, $\mathbb{E}_{\mathbf{x}}[h(\mathbf{x})h'(\mathbf{x})] = 0$.*

Applying Theorem 1 to the N-Bit Parity Problem

for any $\mathbf{v}, \mathbf{v}'$,

$$\mathop{\mathbb{E}}_{\mathbf{x}}\left[(-1)^{\langle \mathbf{x},\mathbf{v}\rangle}(-1)^{\langle \mathbf{x},\mathbf{v}'\rangle}\right] = \mathop{\mathbb{E}}_{\mathbf{x}}\left[(-1)^{\langle \mathbf{x},\mathbf{v}+\mathbf{v}'\rangle}\right]$$

$$= \prod_{i=1}^{d} \mathbb{E}\left[(-1)^{x_i(v_i+v_i')}\right] = \prod_{i=1}^{d} \frac{(-1)^{v_i+v_i'} + (-1)^{-(v_i+v_i')}}{2}$$

- *$\mathcal{H}$ consists of real-valued functions h satisfying $\mathbb{E}_{\mathbf{x}}[h^2(\mathbf{x})] \leq 1$, such that for any two distinct $h, h' \in \mathcal{H}$, $\mathbb{E}_{\mathbf{x}}[h(\mathbf{x})h'(\mathbf{x})] = 0$.*

Applying Theorem 1 to the N-Bit Parity Problem

for any $\mathbf{v}, \mathbf{v}'$,

$$\mathop{\mathbb{E}}_{\mathbf{x}}\left[(-1)^{\langle \mathbf{x},\mathbf{v}\rangle}(-1)^{\langle \mathbf{x},\mathbf{v}'\rangle}\right] = \mathop{\mathbb{E}}_{\mathbf{x}}\left[(-1)^{\langle \mathbf{x},\mathbf{v}+\mathbf{v}'\rangle}\right]$$
$$= \prod_{i=1}^{d} \mathbb{E}\left[(-1)^{x_i(v_i+v_i')}\right] = \prod_{i=1}^{d} \frac{(-1)^{v_i+v_i'} + (-1)^{-(v_i+v_i')}}{2}$$

which is non-zero if and only if $\mathbf{v} = \mathbf{v}'$.

I.e., it satisfies this condition from the theorem:

- *$\mathcal{H}$ consists of real-valued functions h satisfying $\mathbb{E}_{\mathbf{x}}[h^2(\mathbf{x})] \leq 1$, such that for any two distinct $h, h' \in \mathcal{H}$, $\mathbb{E}_{\mathbf{x}}[h(\mathbf{x})h'(\mathbf{x})] = 0$.*

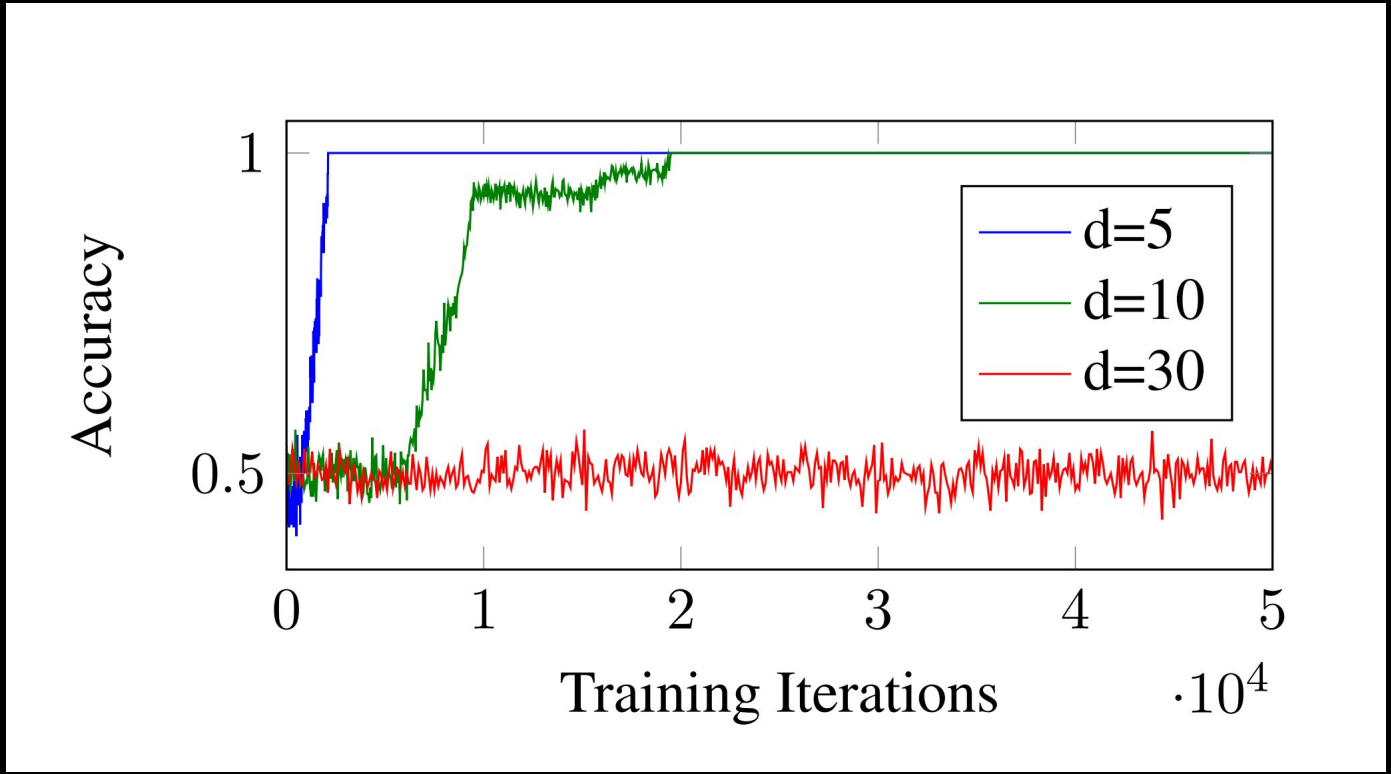
So for parity, gradient becomes exponentially small in d

by Theorem 1, we get that $\mathrm{Var}(\mathcal{H}, F, \mathbf{w}) \leq G(\mathbf{w})^2/2^d$

$$Var(\mathcal{H}, F, \mathbf{w}) \leq \frac{G(\mathbf{w})^2}{|\mathcal{H}|}.$$

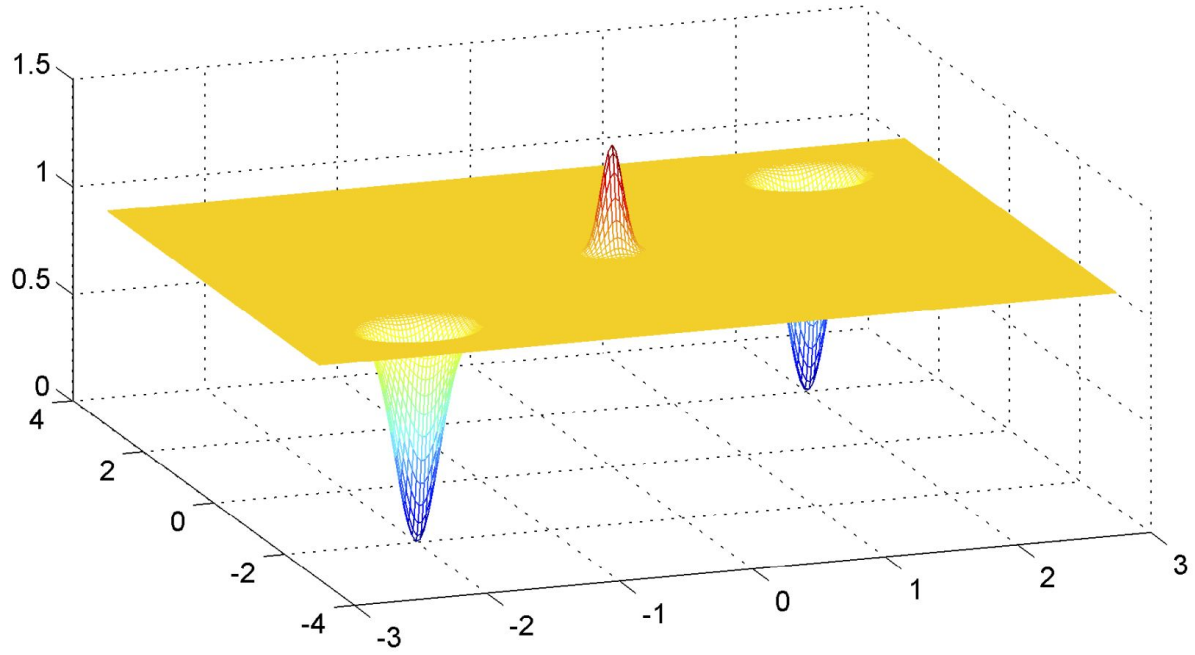
Mila

Results on Parity Learning



Mila

$$F_h(\mathbf{w}) = \mathbb{E}_{\mathbf{x}}[(\cos(\mathbf{w}^T\mathbf{x}) - h(\mathbf{x}))^2] \text{ for } h(\mathbf{x}) = \cos([2,2]^T\mathbf{x}), \text{ in 2-d, } \mathbf{x} \sim \mathcal{N}(0, I)$$

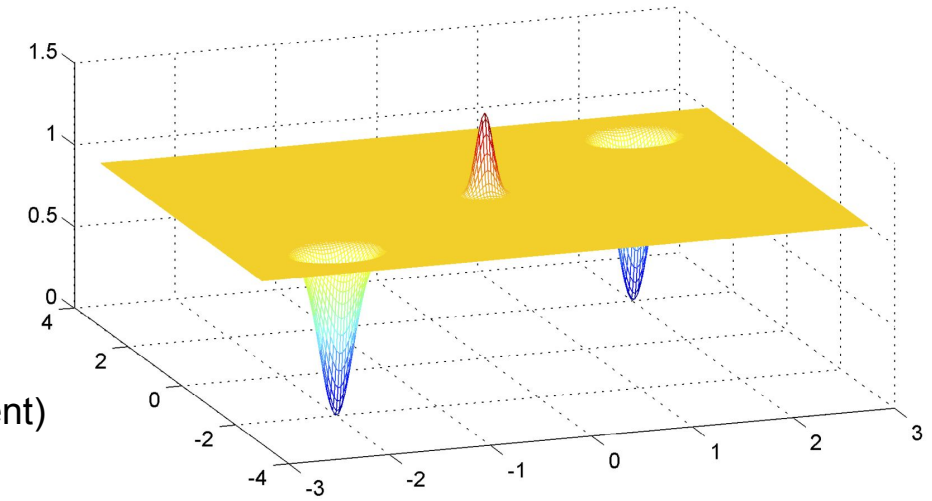


Distribution-Specific Hardness of Learning Neural Networks
Ohad Shamir. 2016

Difficulty of Learning Linear-Periodic Functions

Mila

$$F_h(\mathbf{w}) = \mathbb{E}_{\mathbf{x}}[(\cos(\mathbf{w}^T\mathbf{x}) - h(\mathbf{x}))^2] \text{ for } h(\mathbf{x}) = \cos([2,2]^T\mathbf{x}), \text{ in 2-d, } \mathbf{x} \sim \mathcal{N}(0, I)$$



No Local Minima
No Saddle Points
Extremely flat unless very close to optimum
Difficult for any local search (including gradient descent)

Distribution-Specific Hardness of Learning Neural Networks
Ohad Shamir. 2016

Difficulty of Learning Linear-Periodic Functions

Mila

$$\min_{\mathbf{w}:\mathbf{w}\in\mathcal{W}} \mathbb{E}_{\mathbf{x}\sim\varphi^2}\left[\left(f(\mathbf{w},\mathbf{x})-\psi(\langle\mathbf{w}^\star,\mathbf{x}\rangle)\right)^2\right]$$

Distribution-Specific Hardness of Learning Neural Networks
Ohad Shamir. 2016

Difficulty of Learning Linear-Periodic Functions

Mila

Theorem 2 (Shamir 2016) *Let ψ be a fixed periodic function, and let $\mathcal{H} = \{\mathbf{x} \mapsto \psi(\mathbf{v}^{*\top}\mathbf{x}) : \|\mathbf{v}^*\| = r\}$ for some $r > 0$. Suppose $\mathbf{x} \in \mathbb{R}^d$ is sampled from an arbitrary mixture of distributions with the following property: The square root of the density function φ has a Fourier transform $\hat{\varphi}$ satisfying $\frac{\int_{\mathbf{x}:\|\mathbf{x}\|>r} \hat{\varphi}^2(\mathbf{x})d\mathbf{x}}{\int_{\mathbf{x}} \hat{\varphi}^2(\mathbf{x})d\mathbf{x}} \leq \exp(-\Omega(r))$. Then if F denotes the objective function with respect to the squared loss,*

$$Var(\mathcal{H}, F, \mathbf{w}) \leq O\left(\exp(-\Omega(d)) + \exp(-\Omega(r))\right).$$

Reasonable learning methods will fail unless number of iterations is exponentially large in r and d!

Distribution-Specific Hardness of Learning Neural Networks
Ohad Shamir. 2016

Difficulty of Learning Linear-Periodic Functions

Mila

So what can you do about it?

1. Reformulate the problem (ask a different question)
2. Decompose the problem and add supervision
3. Optimize an alternative objective (or sometimes random guessing!)
4. Change the geometry through inductive bias

Mila

Inductive Bias	**Corresponding property**
Distributed representations	Patterns of features
Convolution	group equivariance (usually over space)
Deep architectures	Complicated functions = composition of simpler ones
Graph Neural Networks	equivariance over entities and relations
Recurrent Nets	equivariance over time
Soft attention	equivariance over permutations

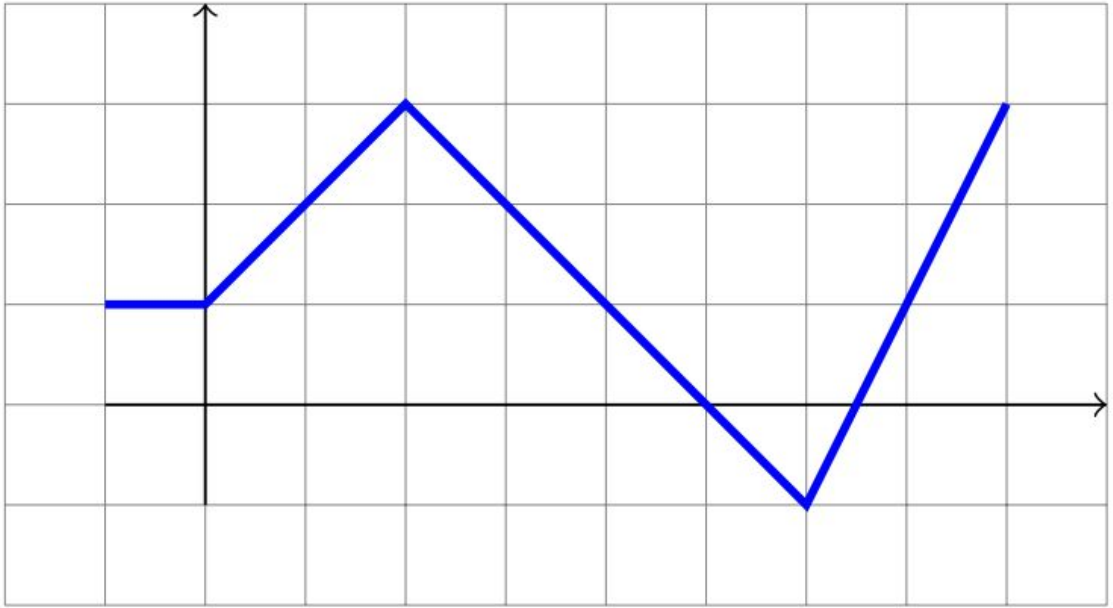
Table 1: Examples of current inductive biases in deep learning

Anirudh Goyal, Yoshua Bengio. Arxiv 2021

Inductive Biases for Deep Learning of Higher-Level Cognition

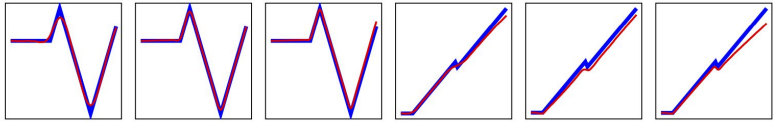
Mila

Num Pieces: k=3
Num Points: n=100



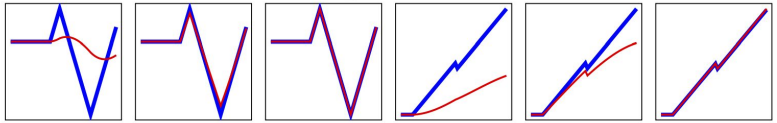
Failures of Gradient-Based Deep Learning (Section 4)

linear converges in $\Omega(n^{3.5})$



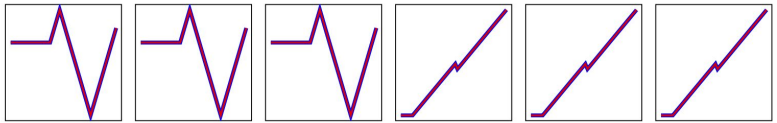
(a) Section 4.1.1's experiment - linear architecture.

Convolutional converges in $\Theta(n^3)$



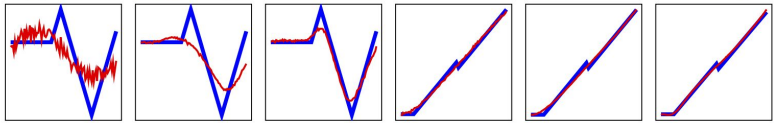
(b) Section 4.1.2's experiment - convolutional architecture.

Conv w/ conditioning converges in $O(\log(1/\epsilon))$



(c) Section 4.1.3's experiment - convolutional architecture with conditioning.

3 layer relu autoencoder

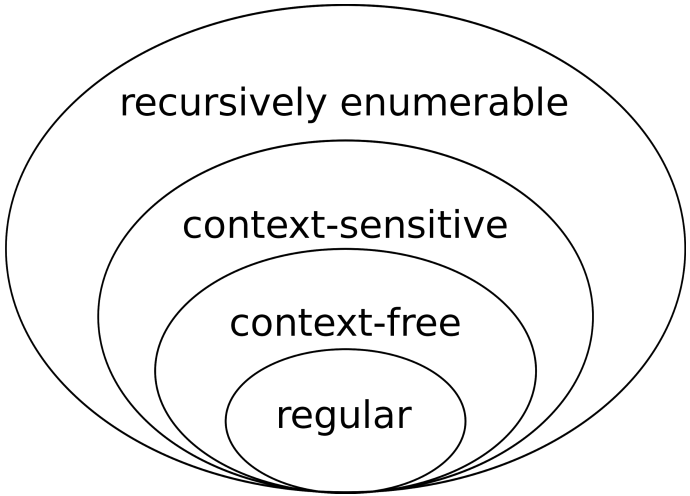


(d) 4.1.4's experiment - vanilla auto encoder.

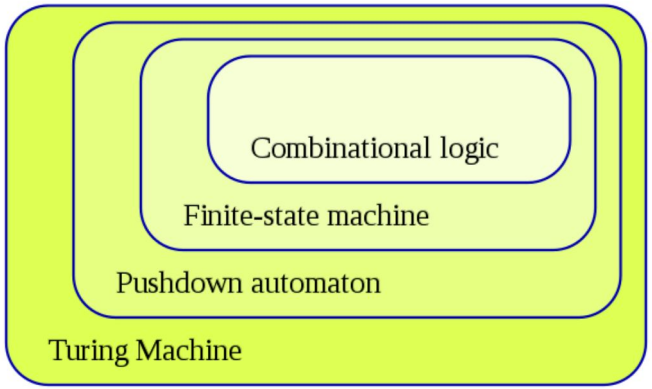
Figure 5: Examples for decoded outputs of Section 4's experiments, learning to encode PWL curves. In blue are the original curves. In red are the decoded curves. The plot shows the outputs for two curves, after 500, 10000, and 50000 iterations, from left to right.

Thank you for your time!

And remember, by careful trying to learn Parity and Linear Periodic Functions

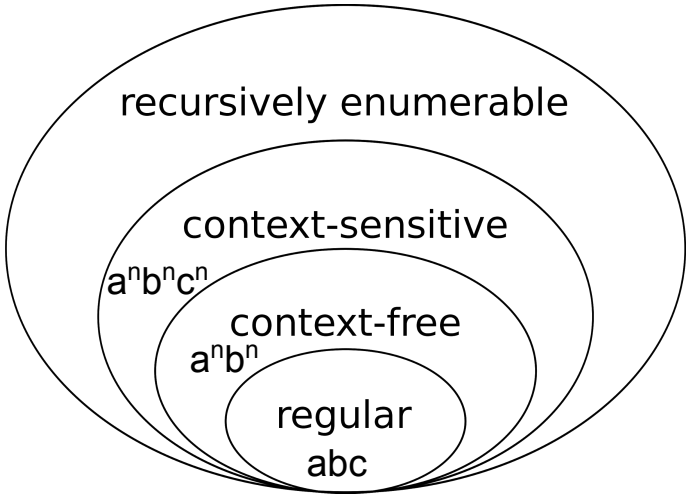


Chomsky Hierarchy

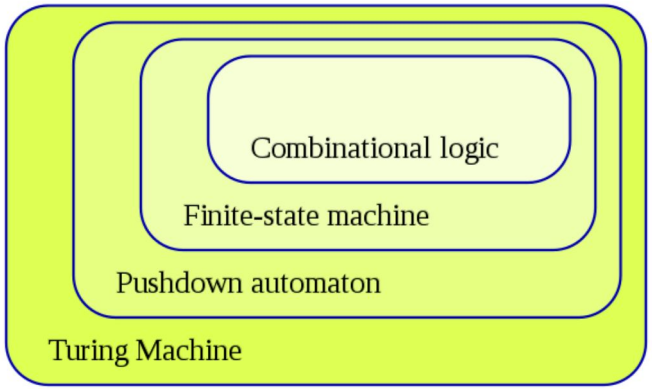


Automata Hierarchy

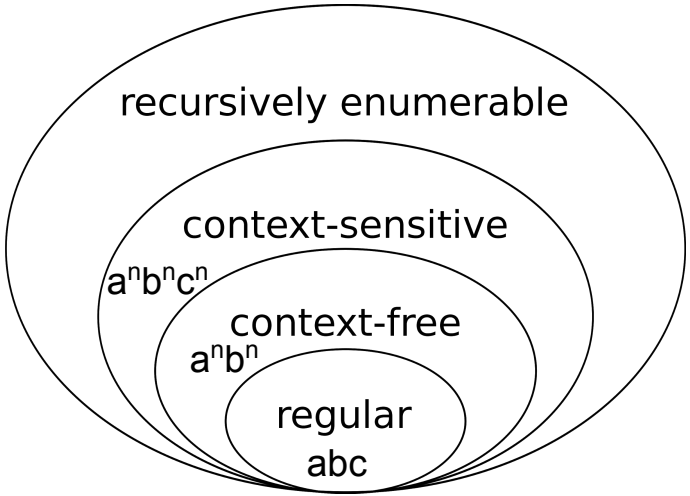
A recursively enumerable language is a recursively enumerable subset in the set of all possible words over the alphabet of the language.



Chomsky Hierarchy



Automata Hierarchy



Chomsky Hierarchy

Take a^nb^n , if in practice n <= N

Regular Language (N+1 rules)

ε|(ab)|(aabb)|(aaabbb)|...

CFG (2 rules)

S ➡️ a S b
S ➡️ ε

A hypothesis about the nature of computable functions stating that a function on the natural numbers can be calculated by an **effective method** if and only if it is computable by a Turing machine.

Alonzo Church
1903 - 1995

Alan Turing
1912 - 1954

Computability: Church-Turing Thesis

Mila

- Consists of a finite number of exact, finite instructions.
- When it is applied to a problem from its class:
  - It always finishes after a finite number of steps
  - It always produces a correct answer
- In principle, it can be done by a human without any aids except writing materials.
- Its instructions need only to be followed rigorously to succeed.

Computability: Effective Method

Mila

Which neural networks are computationally universal (i.e. Turing Complete)?

Q: Which neural networks are computationally universal (i.e. Turing Complete)?
A: Not feed forward networks

$\mathcal{D}_2$ ([ ]) [(( ))]

$\mathcal{D}_4$ (< > [{ }] < >)

Computability: Dyck-n

Mila

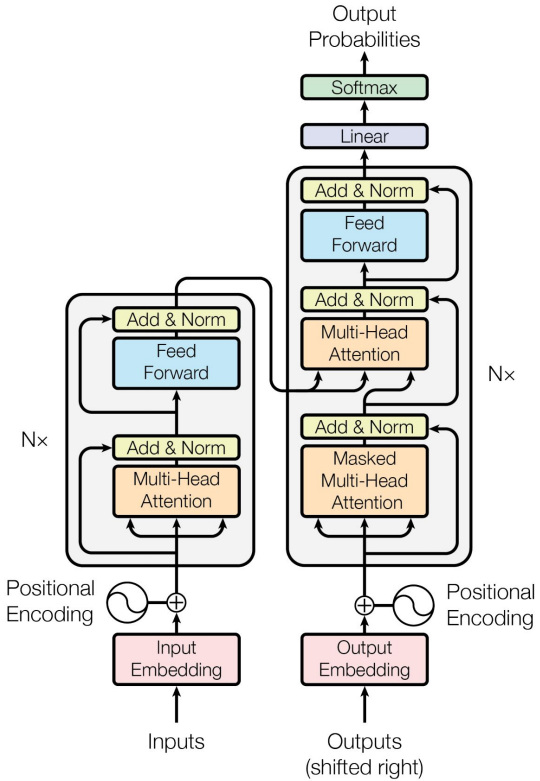
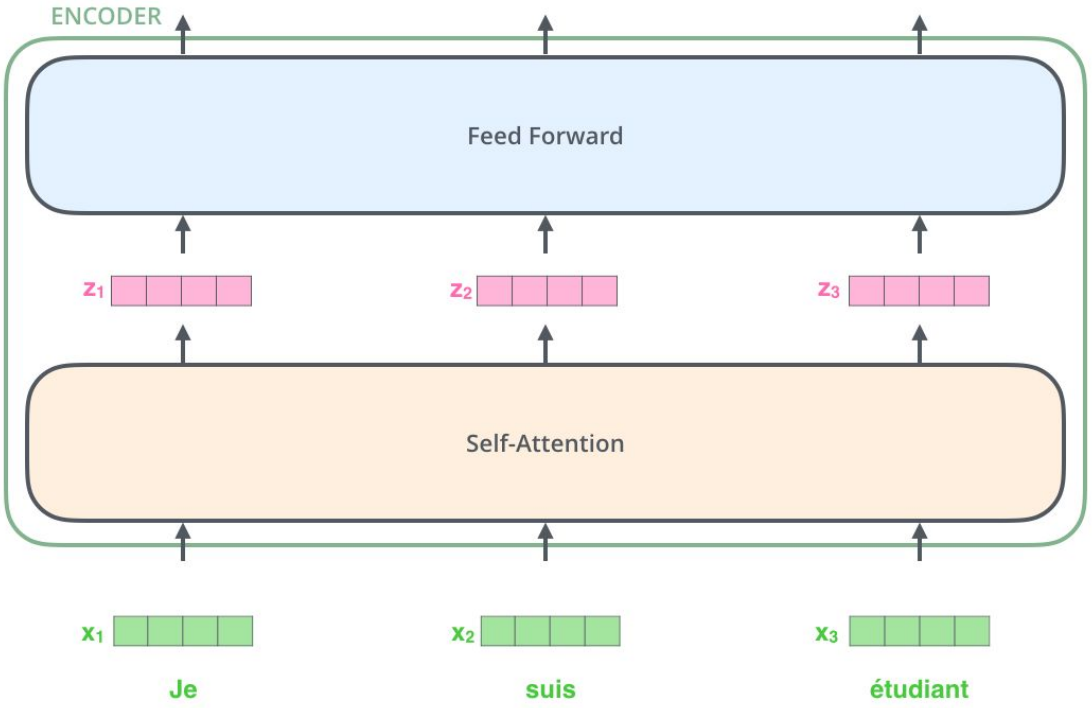


Figure 1: The Transformer - model architecture.

Transformers: Attention is all you need 2017

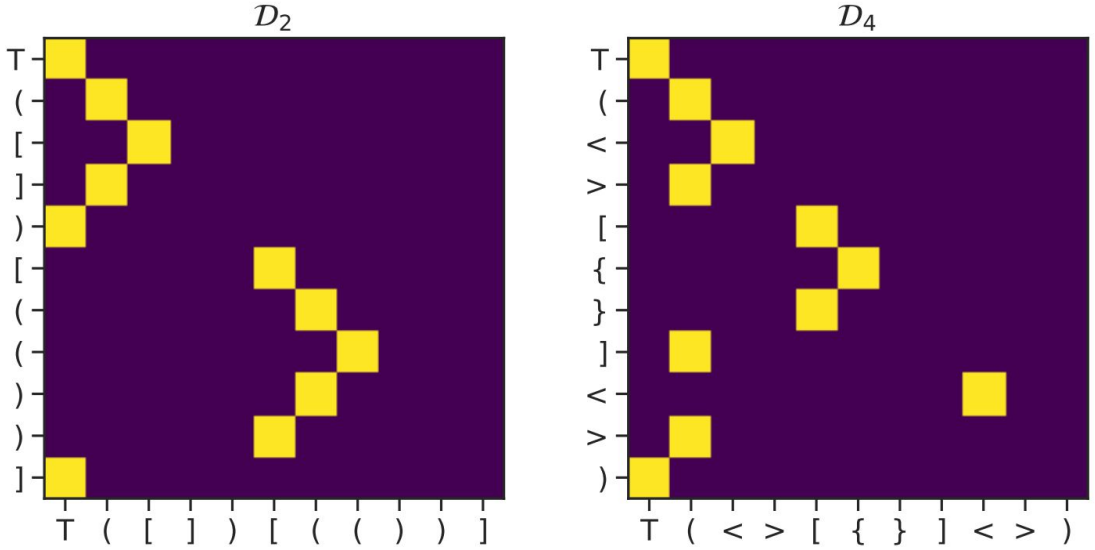
Mila



Taken from the Illustrated Transformer (http://jalammar.github.io/illustrated-transformer/)

Transformers: Encoder Layer

Mila



Javid Ebrahimi, Dhruv Gelda, Wei Zhang. EMNLP 2020

How Can Self-Attention Networks Recognize Dyck-n Languages?

Mila

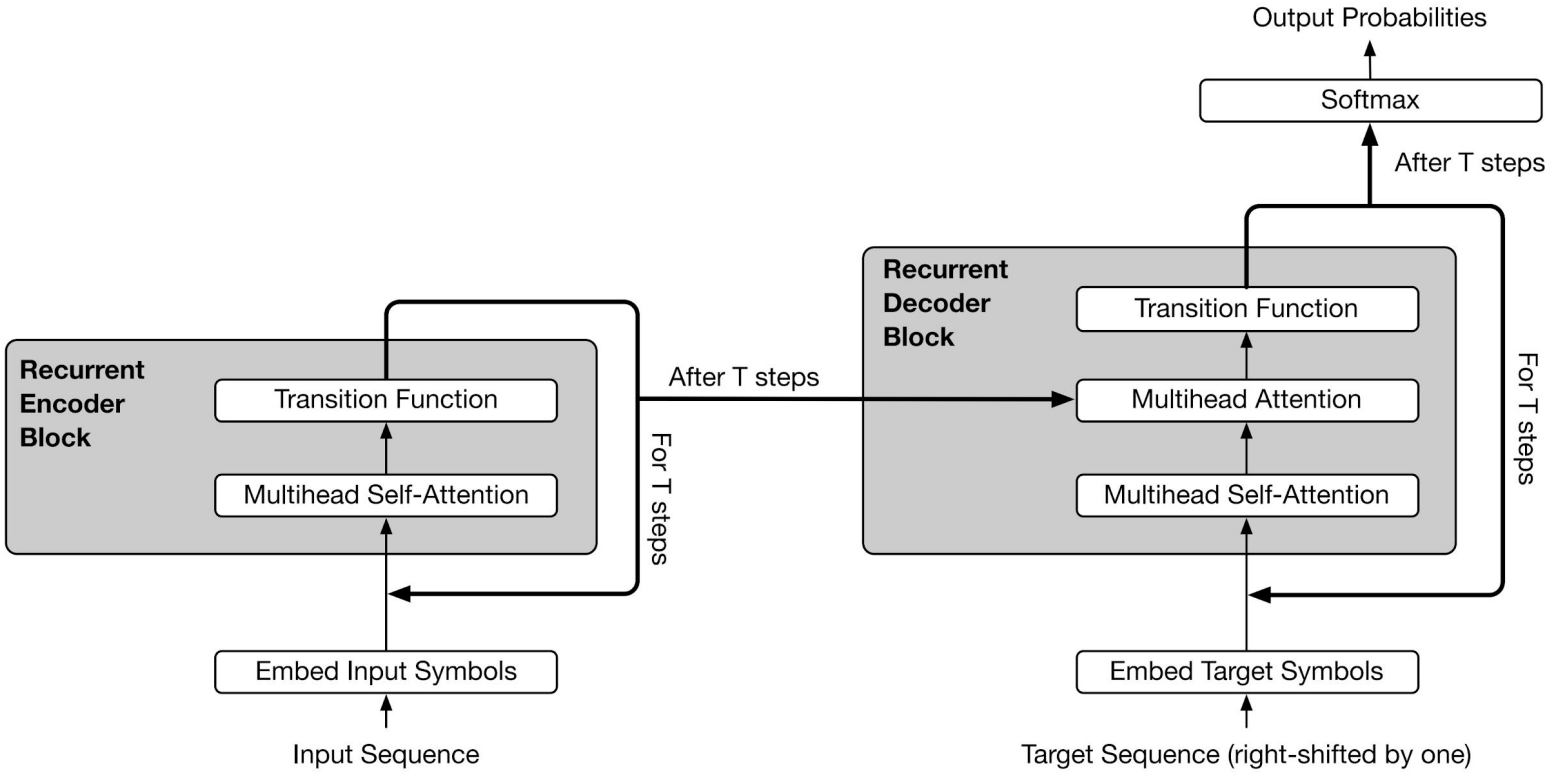
proves limitations on the achievable cross-entropy in modeling distributions over the formal languages

Argument relies on smoothness of the operations used in transformers to show that any transformer, as inputs get longer, will not be able to model Parity or Dyck-2.

Michael Hahn. Arxiv 2020

Lemma 5. *Let a soft attention transformer be given, and let n be the input length. If we exchange one input symbol x_i $(i < n)$, then the change in the resulting activation $y_n^{(L)}$ at the decoder layer is bounded as $O(\frac{1}{n})$ with constants depending on the parameter matrices.*

Michael Hahn. Arxiv 2020



Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, Łukasz Kaiser. ICLR 2019

Universal Transformers: Architecture

Mila

Model	Copy		Reverse		Addition	
	char-acc	seq-acc	char-acc	seq-acc	char-acc	seq-acc
LSTM	0.45	0.09	0.66	0.11	0.08	0.0
Transformer	0.53	0.03	0.13	0.06	0.07	0.0
Universal Transformer	0.91	0.35	0.96	0.46	0.34	0.02

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, Łukasz Kaiser. ICLR 2019

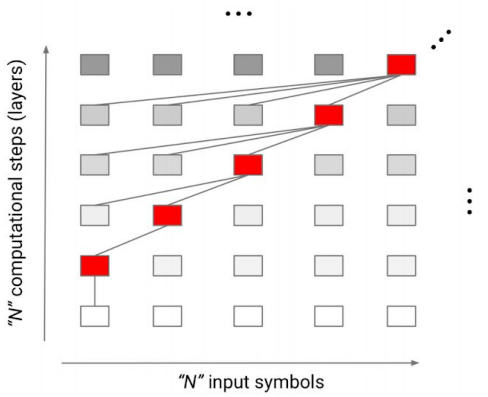
Model	BLEU
Universal Transformer *small*	26.8
Transformer *base* (Vaswani et al., 2017)	28.0
Weighted Transformer *base* (Ahmed et al., 2017)	28.4
Universal Transformer *base*	**28.9**

Table 7: Machine translation results on the WMT14 En-De translation task trained on 8xP100 GPUs in comparable training setups. All *base* results have the same number of parameters.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, Łukasz Kaiser. ICLR 2019

With respect to their computational power, the key difference between the Transformer and the Universal Transformer lies in the number of sequential steps of computation (i.e. in depth). While a standard Transformer executes a total number of operations that scales with the input size, the number of sequential operations is constant, independent of the input size and determined solely by the number of layers. Assuming finite precision, this property implies that the standard Transformer cannot be computationally universal. When choosing a number of steps as a function of the input length, however, the Universal Transformer does not suffer from this limitation. Note that this holds independently of whether or not adaptive computation time is employed but does assume a non-constant, even if possibly deterministic, number of steps. Varying the number of steps dynamically after training is enabled by sharing weights across sequential computation steps in the Universal Transformer.

An intuitive example are functions whose execution requires the sequential processing of each input element. In this case, for any given choice of depth T, one can construct an input sequence of length $N > T$ that cannot be processed correctly by a standard Transformer. With an appropriate, input-length dependent choice of sequential steps, however, a Universal Transformer, RNNs or Neural GPUs can execute such a function.



Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, Łukasz Kaiser. ICLR 2019

Theorem 3. ***(Universality of Weight-tied Deep Networks)*** *Consider a traditional L-layer deep network defined by the relation*

$$\mathbf{z}^{[i+1]} = \sigma^{[i]}(W^{[i]}\mathbf{z}^{[i]} + \mathbf{b}^{[i]}), \quad i = 0, \ldots, L-1, \quad \mathbf{z}^{[0]} = \mathbf{x} \tag{17}$$

where $\mathbf{z}^{[i]}$ denotes the hidden features at depth i, $W^{[i]}$, $\mathbf{b}^{[i]}$ are parameters of the network, $\sigma^{[i]}$ is the non-linearity at depth i, and $\mathbf{x}$ is the original input. Then the same network can be represented by a weight-tied, input-injected network of equivalent depth

$$\tilde{\mathbf{z}}^{[i+1]} = \sigma(W_z\tilde{\mathbf{z}}^{[i]} + W_x\mathbf{x} + \tilde{\mathbf{b}}), \quad i = 0, \ldots, L-1. \tag{18}$$

where σ, W_z, W_x and $\tilde{\mathbf{b}}$ are constant over all layers.

Proof of Theorem 3. The proof is constructive: we build the weight-tied network equivalent to the original network by contructing the relevant matrices using a simple "shift" operation. In particular, we define the network parameters as

$$W_z = \begin{bmatrix} 0 & 0 & \ldots & 0 & 0 \\ W^{[1]} & 0 & \ldots & 0 & 0 \\ 0 & W^{[2]} & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & W^{[L-1]} & 0 \end{bmatrix}, W_x = \begin{bmatrix} W^{[0]} \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \tilde{\mathbf{b}} = \begin{bmatrix} \mathbf{b}^{[0]} \\ \mathbf{b}^{[1]} \\ \vdots \\ \mathbf{b}^{[L-1]} \end{bmatrix}, \quad \sigma = \begin{bmatrix} \sigma^{[0]} \\ \sigma^{[1]} \\ \vdots \\ \sigma^{[L-1]} \end{bmatrix}. \tag{19}$$

It is clear from inspection that after L applications of the layer, i.e.,

$$\tilde{\mathbf{z}}^{[i+1]} = \sigma(W_z\tilde{\mathbf{z}}^{[i]} + W_x\mathbf{x} + \tilde{\mathbf{b}}) \tag{20}$$

using these parameters the hidden vector $\tilde{\mathbf{z}}$ will take on the value

$$\tilde{\mathbf{z}}^{[L]} = \begin{bmatrix} \mathbf{z}^{[1]} \\ \mathbf{z}^{[2]} \\ \vdots \\ \mathbf{z}^{[L]} \end{bmatrix}. \tag{21}$$

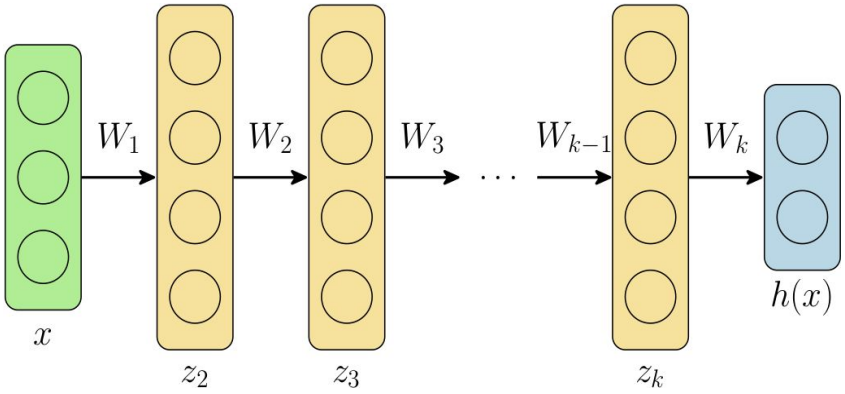
Thus the weight-tied network computes all the same terms as the original network, using the same depth as the original network, and with a hidden unit size that is just the sum of the individual hidden unit sizes in the original network. This establises the claim of the theorem. □

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, Łukasz Kaiser. ICLR 2019

Universal Transformers: Theorem

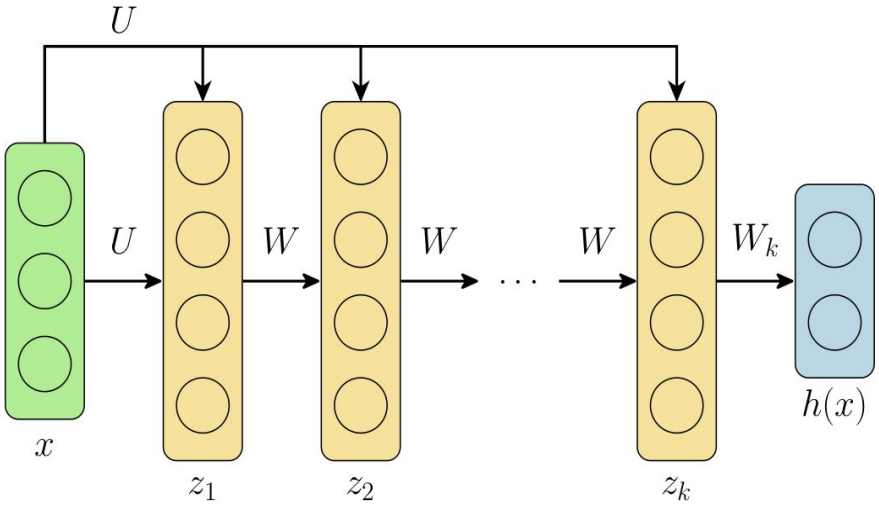
Mila

Normal Net



$$z_1 = x$$
$$z_{i+1} = \sigma(W_i z_i + b_i),\ i = 1, \ldots, k-1$$
$$h(x) = W_k z_k + b_k$$

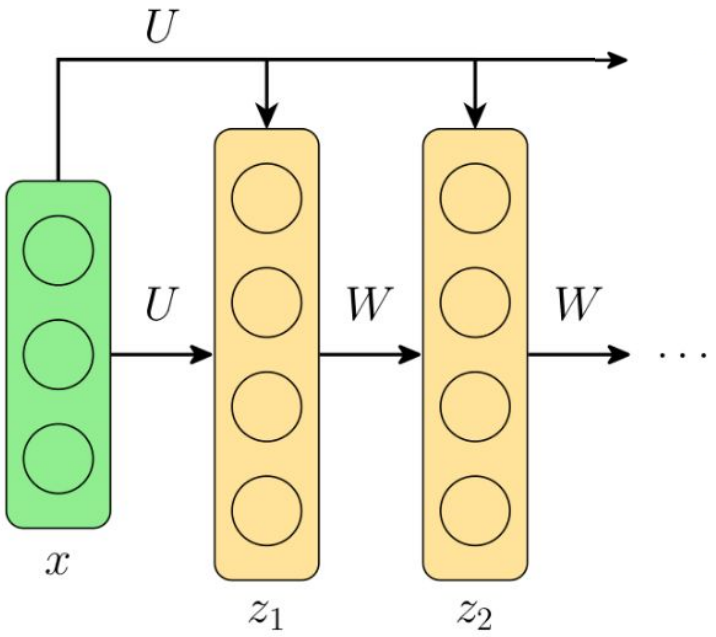
Weight-Tied Input-Injected Network



$$z_1 = 0$$
$$z_{i+1} = \sigma(W z_i + U x + b),\ i = 1, \ldots, k-1$$
$$h(x) = W_k z_k + b_k$$

Deep Equilibrium Models (DEQ): Weight-Tied Layers

Mila

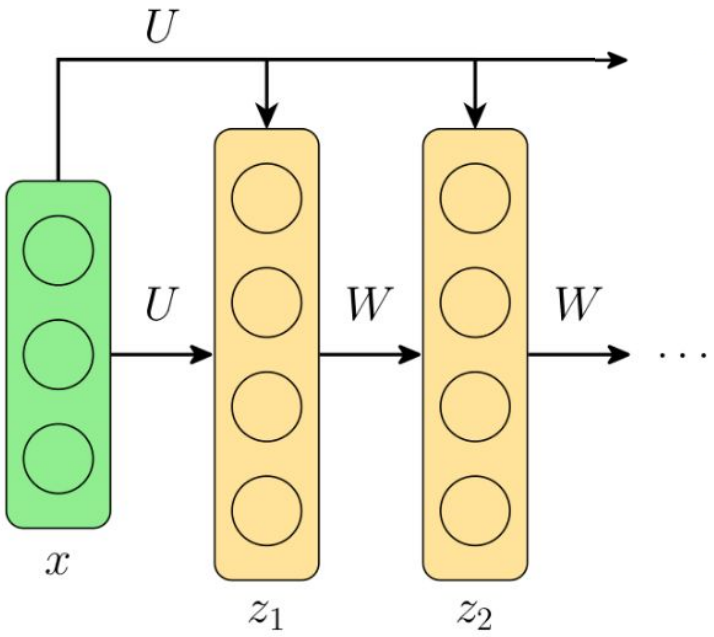


$$z_{i+1} = \sigma(Wz_i + Ux + b) \quad i \to \infty$$

$$z^\star = \sigma(Wz^\star + Ux + b)$$

DEQ Models: Fixed Point Iteration

Mila



$$z_{i+1} = \sigma(Wz_i + Ux + b) \quad i \to \infty$$

$$z^\star = \sigma(Wz^\star + Ux + b)$$

Possibilities:

1. Divergence
2. Oscillation (periodically or chaotically)
3. Convergence (to a fixed point)

DEQ Models: Fixed Point Iteration

Mila

Deep Equilibrium Models: How to model this thing?

Mila

Any deep network **(of any depth, with any connectivity), can be represented as a** *single layer* **DEQ model**

Proof: Consider a traditional composition of two functions $y = g_2(g_1(x))$, we can transform this into a single layer DEQ by simply concatenating all the intermediate terms of this function into a long vector:

$$f(z, x) = f\left(\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}, x\right) = \begin{bmatrix} g_1(x) \\ g_2(z_1) \end{bmatrix}$$

At the equilibrium point of this function, z*, we have:

$$z^\star = f(z^\star, x) \iff z_1^\star = g_1(x),\ z_2^\star = g_2(z_1^\star) = g_2(g_1(x))$$

□

Analysis: This logic applies to any computation graph, concatenating all intermediate products of a computation graph into the vector z, and having the function f be the function that applies the “next” computation in the graph to each of these elements.

This is much more inefficient than just computing the original network, and not the construction used in practice.