On Two Possible Reasons Why Deep Neural Networks Are So Successful

Tomáš Werner

Deptartment of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague

(Talk given at IRTC Kiev, October 2018.)
Neural network computes a function $f_\theta: \mathbb{R}^n \rightarrow \mathbb{R}^k$

$$f_\theta = g_h \circ \varphi \circ g_{h-1} \circ \cdots \circ \varphi \circ g_2 \circ \varphi \circ g_1$$

where

- $h$ is the number of layers
- $g_i(x) = A_i x + b_i$ are affine functions
- $\theta = (A_1, b_1, \ldots, A_h, b_h)$ are all parameters of the network
- $\varphi: \mathbb{R} \rightarrow \mathbb{R}$ is a non-linear activation function (applied to vectors component-wise)

$A_i, b_i$ can be sparse and/or have coupled entries (like in convolutional networks)
Use for Classification

Want to classify feature vectors \( x \in \mathbb{R}^n \) to classes \( y \in \{1, \ldots, k\} \).

Assume

\[
p_\theta(y \mid x) = \frac{\exp f_\theta(x)_y}{\sum_{j=1}^k \exp f_\theta(x)_j}
\]

where \( f_\theta: \mathbb{R}^n \to \mathbb{R}^k \) serves as a discriminative function.

Classification:

\[
x \mapsto y = \arg\max_{j=1,\ldots,k} p(j \mid x) = \arg\max_{j=1,\ldots,k} f(x)_j
\]

Training: Given a multiset \( D = \{(x_1, y_1), \ldots, (x_m, y_m)\} \), minimize

\[
L(f_\theta, D) = -\log \prod_{i=1}^m p_\theta(y_i \mid x_i)
\]

over \( \theta \).
Empirical Observations

- Neural network classifiers with many \(10^1 - 10^2\) layers are tremendously more accurate than any other techniques for classifying complex natural images.

- Training can be done with stochastic gradient descent, plus several heuristics (batch normalization, drop out).

This is surprising (to me...)! Any formal explanations?

- Why may depth be important?
- Why is training tractable?
Part 1:
Importance of Depth
Neural Networks Are Universal Approximators

Theorem (Cybenko (1989), Hornik (1991))

Let $\varphi : \mathbb{R} \to \mathbb{R}$ be non-constant, bounded and continuous. For any continuous $f : [0, 1]^n \to \mathbb{R}$ and any $\epsilon > 0$, there is a network with $h = 2$ layers computing function $f_{\theta} : \mathbb{R}^n \to \mathbb{R}$ such that

$$|f(x) - f_{\theta}(x)| \leq \epsilon \quad \forall x \in [0, 1]^n.$$ 

So, neural network with $h = 2$ layers can approximate any continuous function arbitrarily well.

But possibly at the expense of large number of neurons!
Boolean Circuits

Boolean circuit with input $x_1, \ldots, x_n \in \{0, 1\}$ and output $y \in \{0, 1\}$ is a directed acyclic graph (DAG) with nodes

- $x_1, \ldots, x_n$ with indegree 1 (inputs)
- NOT gate
- AND gate (with any number of inputs)
- OR gate (with any number of inputs)

A single node has outdegree 1 (output).

The circuit computes a function $f: \{0, 1\}^n \rightarrow \{0, 1\}$.

For circuit $C$, define

- $\text{size}(C) = \text{the number of gates of } C$
- $\text{depth}(C) = \text{the depth of the DAG of } C$
A family \((f_n)_{n \in \mathbb{N}}\) of functions \(f_n: \{0, 1\}^n \rightarrow \{0, 1\}\) defines a function
\[
f: \{0, 1\}^* \rightarrow \{0, 1\}
\]
where \(\{0, 1\}^* = \bigcup_{n \in \mathbb{N}} \{0, 1\}^n\).

Family \((C_n)_{n \in \mathbb{N}}\) of circuits computes \(f\) if \(C_n\) computes \(f_n\) for every \(n \in \mathbb{N}\).
AC$^0$ is the class of functions $f: \{0, 1\}^* \to \{0, 1\}$ such that for every $f \in AC^0$ there is a family $(C_n)$ such that

- family $(C_n)$ computes $f$,
- there is $k \in \mathbb{N}$ such that depth$(C_n) \leq k$ for all $n \in \mathbb{N}$,
- there is a polynomial $p$ such that size$(C_n) \leq p(n)$ for all $n \in \mathbb{N}$.

Function PARITY: $\{0, 1\}^* \to \{0, 1\}$:

$$PARITY_n(x_1, \ldots, x_n) = x_1 + \cdots + x_n \pmod{2}.$$ 


PARITY $\notin AC^0$

Known proofs are long and difficult.
Theorem (Håstad (1989))

There are functions computable by polynomial-size circuits of depth \( d \) but not by polynomial-size circuits of depth \( d - 1 \).

Similar results for arithmetic circuits (a.k.a. sum-product networks), which are DAGs with nodes

- \( x_1, \ldots, x_n \in \mathbb{R} \)
- real constants
- operation \( + \)
- operation \( \times \)

Compute polynomials \( \mathbb{R}^n \to \mathbb{R} \).

Many other results on approximating (rather than exactly computing) function classes by deep vs. shallow neural networks.
Are deep classifiers better than shallow for *all* classification tasks?

Suppose we train not only parameters but also structure:

- $f_C : \mathbb{R}^n \rightarrow \mathbb{R}^k$ where $C$ is structure+parameters of $f_C$
- Minimize loss $L(f_C, D)$ subject to $\text{size}(C) \leq s$

Will the optimal structure be always ‘deep’?

If not, characterize the tasks for which deep is better than shallow!

- Natural image statistics
- Is our perceptual world ‘compositional’?

Informal arguments for depth:

- sharing intermediate results
- brains are hierarchical (Hubel, Wiesel, 1963)
- real images are created in generative way in many steps (Lin, Tegmark, Rolnick, 2017)
Part 2:
Is Minimization of Huge Non-convex Functions Inevitably Hard?
Gaussian random process: a probability distribution over functions $\mathbb{R}^n \rightarrow \mathbb{R}$
Random Symmetric Gaussian Matrices

$S$ is a random symmetric Gaussian matrix if $S = A + A^T$ where $A$ is an $n \times n$ matrix with random entries i.i.d. from $\mathcal{N}(0, 1)$.

**Theorem (Wigner (1958))**

*When $n \to \infty$, the spectrum of $S$ has the semicircular distribution.*
Let $f_n: \mathbb{R}^n \to \mathbb{R}$ is a stationary Gaussian random process with zero mean. At a random critical point $x$ of $f_n$ we have

$$\nabla^2 f_n(x) = S - \varphi(f_n(x))I$$

where

- $S$ is a random symmetric Gaussian matrix,
- $\varphi$ is a monotone function,
- $I$ is the $n \times n$ identity matrix.
\( \alpha(x) = \) the number of negative eigenvalues of \( \nabla^2 f_n(x) \)

When \( n \to \infty \), the distribution \( p(f_n(x), \alpha(x)/n) \) for critical points \( x \) concentrates along a monotone curve:
h-spin spherical spin glass model on n variables:

\[
f_{n,h}(s_1, \ldots, s_n) = \frac{1}{n^{(h+1)/2}} \sum_{i_1=1}^{n} \cdots \sum_{i_h=1}^{n} J_{i_1 \ldots i_h} \prod_{k=1}^{h} s_{i_k}
\]

where \( s_1^2 + \cdots + s_n^2 = n \) and \( J_{i_1 \ldots i_h} \) are random, i.i.d. from \( \mathcal{N}(0,1) \).

The index of point \( s \) is the number of negative eigenvalues of \( \nabla^2 f_{n,h}(s) \).

**Theorem (Auffinger et al. (2010), reviewed by Choromanska et al. (2015))**

Let \( h \geq 4 \). There are numbers

\[
E_0 < E_1 < \cdots < E_\infty = -2\sqrt{(h - 1)/h}
\]

such that for \( n \to \infty \),

- all critical points of \( f_{n,h} \) are above \( E_0 \),
- all critical points of \( f_{n,h} \) with finite index are below \( E_\infty \),
- all critical points of \( f_{n,h} \) with index smaller than \( k \) are below \( E_k \).
### Theorem (Becker et al. (2017))

The number of critical points of $f_{n,h}$ is

$$\frac{(h - 1)^n - 1}{h - 2}.$$
If each layer of a network has $n_0$ units, the number of network parameters is

$$N_e = (h - 1)n_0^2 + n_0.$$  

In approximating the loss function by spin glass model, $n$ corresponds to the $h$-root of the total number of paths:

$$n = \sqrt[2h-1]{4N_e(h - 1) + 1 + 1}$$

$$= \frac{\sqrt[2h-1]{4N_e(h - 1) + 1 + 1}}{2(h - 1)}$$