One More Advantage of Deep Learning: While in General, A Perfect Training of a Neural Network Is NP-Hard, It Is Feasible for Bounded-Width Deep Networks

Vladik Kreinovich
Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968, USA
vladik@utep.edu
http://www.cs.utep.edu/vladik
(based on a joint work with Chitta Baral)

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

1. Why Traditional Neural Networks: (Sanitized) History

- How do we make computers think?
- To make machines that fly it is reasonable to look at the creatures that know how to fly: the birds.
- To make computers think, it is reasonable to analyze how we humans think.
- On the biological level, our brain processes information via special cells called *neurons*.
- Somewhat surprisingly, in the brain, signals are electric – just as in the computer.
- The main difference is that in a neural network, signals are sequence of identical pulses.

Why Traditional ...
How the Need for Fast ...
Faster Differentiation: ...
Beyond Traditional NN
From Traditional NN ...
Formulation of the ...
What Us Feasible: ...
NP-Hardness Result
Feasibility Resuly

2. Why Traditional NN: (Sanitized) History

- The intensity of a signal is described by the frequency of pulses.
- A neuron has many inputs (up to 10^4).
- All the inputs $x_1, \ldots, x_n$ are combined, with some loss, into a frequency $\sum\limits_{i=1}^{n} w_i \cdot x_i$.
- Low inputs do not active the neuron at all, high inputs lead to largest activation.
- The output signal is a non-linear function

$$y = f\left(\sum_{i=1}^{n} w_i \cdot x_i - w_0\right).$$

- In biological neurons, $f(x) = 1/(1 + \exp(-x))$.
- Traditional neural networks emulate such biological neurons.

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

3. Why Traditional Neural Networks: Real History

- At first, researchers ignored non-linearity and only used linear neurons.
- They got good results and made many promises.
- The euphoria ended in the 1960s when MIT's Marvin Minsky and Seymour Papert published a book.
- Their main result was that a composition of linear functions is linear (I am not kidding).
- This ended the hopes of original schemes.
- For some time, neural networks became a bad word.
- Then, smart researchers came us with a genius idea: let's make neurons non-linear.
- This revived the field.

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

4. Traditional Neural Networks: Main Motivation

- One of the main motivations for neural networks was that computers were slow.
- Although human neurons are much slower than CPU, the human processing was often faster.
- So, the main motivation was to make data processing faster.
- The idea was that:
  - since we are the result of billion years of ever improving evolution,
  - our biological mechanics should be optimal (or close to optimal).

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

5. How the Need for Fast Computation Leads to Traditional Neural Networks

- To make processing faster, we need to have many fast processing units working in parallel.
- The fewer layers, the smaller overall processing time.
- In nature, there are many fast linear processes – e.g., combining electric signals.
- As a result, linear processing (L) is faster than non-linear one.
- For non-linear processing, the more inputs, the longer it takes.
- So, the fastest non-linear processing (NL) units process just one input.
- It turns out that two layers are not enough to approximate any function.

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

6. Why One or Two Layers Are Not Enough

- With 1 linear (L) layer, we only get linear functions.
- With one nonlinear (NL) layer, we only get functions of one variable.
- With L→NL layers, we get $g\left(\sum\limits_{i=1}^{n} w_i \cdot x_i - w_0\right)$.
- For these functions, the level sets $f(x_1, \ldots, x_n) = \text{const}$ are planes $\sum\limits_{i=1}^{n} w_i \cdot x_i = c$.
- Thus, they cannot approximate, e.g., $f(x_1, x_2) = x_1 \cdot x_2$ for which the level set is a hyperbola.
- For NL→L layers, we get $f(x_1, \ldots, x_n) = \sum\limits_{i=1}^{n} f_i(x_i)$.
- For all these functions, $d \stackrel{\text{def}}{=} \dfrac{\partial^2 f}{\partial x_1 \partial x_2} = 0$, so we also cannot approximate $f(x_1, x_2) = x_1 \cdot x_2$ with $d = 1 \neq 0$.

7. Why Three Layers Are Sufficient: Newton's Prism and Fourier Transform

- In principle, we can have two 3-layer configurations: L→NL→L and NL→L→NL.
- Since L is faster than NL, the fastest is L→NL→L:

$$y = \sum_{k=1}^{K} W_k \cdot f_k\left(\sum_{i=1}^{n} w_{ki} \cdot x_i - w_{k0}\right) - W_0.$$

- Newton showed that a prism decomposes while light (or any light) into elementary colors.
- In precise terms, elementary colors are sinusoids

$$A \cdot \sin(w \cdot t) + B \cdot \cos(w \cdot t).$$

- Thus, every function can be approximated, with any accuracy, as a linear combination of sinusoids:

$$f(x_1) \approx \sum_k (A_k \cdot \sin(w_k \cdot x_1) + B_k \cdot \cos(w_k \cdot x_1)).$$

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

8. Why Three Layers Are Sufficient (cont-d)

- Newton's prism result:

$$f(x_1) \approx \sum_k (A_k \cdot \sin(w_k \cdot x_1) + B_k \cdot \cos(w_k \cdot x_1)).$$

- This result was theoretically proven later by Fourier.
- For $f(x_1, x_2)$, we get a similar expression for each x_2, with $A_k(x_2)$ and $B_k(x_2)$.
- We can similarly represent $A_k(x_2)$ and $B_k(x_2)$, thus getting products of sines, and it is known that, e.g.:

$$\cos(a) \cdot \cos(b) = \frac{1}{2} \cdot (\cos(a+b) + \cos(a-b)).$$

- Thus, we get an approximation of the desired form with $f_k = \sin$ or $f_k = \cos$:

$$y = \sum_{k=1}^{K} W_k \cdot f_k \left(\sum_{i=1}^{n} w_{ki} \cdot x_i - w_{k0} \right).$$

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

9. Which Activation Functions $f_k(z)$ Should We Choose

- A general 3-layer NN has the form:

$$y = \sum_{k=1}^{K} W_k \cdot f_k \left(\sum_{i=1}^{n} w_{ki} \cdot x_i - w_{k0} \right) - W_0.$$

- Biological neurons use $f(z) = 1/(1 + \exp(-z))$, but shall we simulate it?
- Simulations are not always efficient.
- E.g., airplanes have wings like birds but they do not flap them.
- Let us analyze this problem theoretically.
- There is always some noise c in the communication channel.
- So, we can consider either the original signals x_i or denoised ones $x_i - c$.

Why Traditional ...
How the Need for Fast ...
Faster Differentiation: ...
Beyond Traditional NN
From Traditional NN ...
Formulation of the ...
What Us Feasible: ...
NP-Hardness Result
Feasibility Resuly

10. Which $f_k(z)$ Should We Choose (cont-d)

- The results should not change if we perform a full or partial denoising $z \to z' = z - c$.
- Denoising means replacing $y = f(z)$ with $y' = f(z-c)$.
- So, $f(z)$ should not change under shift $z \to z - c$.
- Of course, $f(z)$ cannot remain the same: if $f(z) = f(z - c)$ for all c, then $f(z) = \text{const}$.
- The idea is that once we re-scale x, we should get the same formula after we apply a natural y-re-scaling T_c:

$$f(x - c) = T_c(f(x)).$$

- Linear re-scalings are natural: they corresponding to changing units and starting points (like C to F).

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

11. Which Transformations Are Natural?

- An inverse T_c^{-1} to a natural re-scaling T_c should also be natural.
- A composition $y \to T_c(T_{c'}(y))$ of two natural re-scalings T_c and $T_{c'}$ should also be natural.
- In mathematical terms, natural re-scalings form a *group*.
- For practical purposes, we should only consider re-scaling determined by finitely many parameters.
- So, we look for a finite-parametric group containing all linear transformations.

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

12. A Somewhat Unexpected Approach

- N. Wiener, in *Cybernetics*, notices that when we approach an object, we have distinct phases:
  - first, we see a blob (the image is invariant under all transformations);
  - then, we start distinguishing angles from smooth but not sizes (projective transformations);
  - after that, we detect parallel lines (affine transformations);
  - then, we detect relative sizes (similarities);
  - finally, we see the exact shapes and sizes.
- Are there other transformation groups?
- Wiener argued: if there are other groups, after billions years of evolutions, we would use them.
- So he conjectured that there are no other groups.

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

13. Wiener Was Right

- Wiener's conjecture was indeed proven in the 1960s.
- In 1-D case, this means that all our transformations are fractionally linear:

$$f(z-c)=\frac{A(c)\cdot f(z)+B(c)}{C(c)\cdot f(z)+D(c)}.$$

- For $c=0$, we get $A(0)=D(0)=1$, $B(0)=C(0)=0$.
- Differentiating the above equation by c and taking $c=0$, we get a differential equation for $f(z)$:

$$-\frac{df}{dz}=(A'(0)\cdot f(z)+B'(0))-f(z)\cdot(C'(0)\cdot f(z)+D'(0)).$$

- So, $\dfrac{df}{C'(0)\cdot f^2+(A'(0)-C'(0))\cdot f+B'(0)}=-dz.$
- Integrating, we indeed get $f(z)=1/(1+\exp(-z))$ (after an appropriate linear re-scaling of z and $f(z)$).

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

14. How to Train Traditional Neural Networks: Main Idea

- *Reminder:* a 3-layer neural network has the form:

$$y = \sum_{k=1}^{K} W_k \cdot f\left(\sum_{i=1}^{n} w_{ki} \cdot x_i - w_{k0}\right) - W_0.$$

- We need to find the weights that best described observations $\left(x_1^{(p)}, \ldots, x_n^{(p)}, y^{(p)}\right)$, $1 \le p \le P$.
- We find the weights that minimize the mean square approximation error $E \stackrel{\text{def}}{=} \sum_{p=1}^{P} \left(y^{(p)} - y_{NN}^{(p)}\right)^2$, where

$$y^{(p)} = \sum_{k=1}^{K} W_k \cdot f\left(\sum_{i=1}^{n} w_{ki} \cdot x_i^{(p)} - w_{k0}\right) - W_0.$$

- The simplest minimization algorithm is gradient descent: $w_i \to w_i - \lambda \cdot \dfrac{\partial E}{\partial w_i}$.

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

15. Towards Faster Differentiation

- To achieve high accuracy, we need many neurons.
- Thus, we need to find many weights.
- To apply gradient descent, we need to compute all partial derivatives $\dfrac{\partial E}{\partial w_i}$.
- Differentiating a function f is easy:
  - the expression f is a sequence of elementary steps,
  - so we take into account that $(f \pm g)' = f' \pm g'$, $(f \cdot g)' = f' \cdot g + f \cdot g'$, $(f(g))' = f'(g) \cdot g'$, etc.
- For a function that takes T steps to compute, computing f' thus takes $c_0 \cdot T$ steps, with $c_0 \leq 3$.
- However, for a function of n variables, we need to compute n derivatives.
- This would take time $n \cdot c_0 \cdot T \gg T$: this is too long.

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

16. Faster Differentiation: Backpropagation

- Idea:
  - instead of starting from the variables,
  - start from the last step, and compute $\dfrac{\partial E}{\partial v}$ for all intermediate results v.
- For example, if the very last step is $E = a \cdot b$, then $\dfrac{\partial E}{\partial a} = b$ and $\dfrac{\partial E}{\partial b} = a$.
- At each step y, if we know $\dfrac{\partial E}{\partial v}$ and $v = a \cdot b$, then $\dfrac{\partial E}{\partial a} = \dfrac{\partial E}{\partial v} \cdot b$ and $\dfrac{\partial E}{\partial b} = \dfrac{\partial E}{\partial v} \cdot a$.
- At the end, we get all n derivatives $\dfrac{\partial E}{\partial w_i}$ in time

$$c_0 \cdot T \ll c_0 \cdot T \cdot n.$$

- This is known as *backpropagation*.

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

17. Beyond Traditional NN

- Nowadays, computer speed is no longer a big problem.
- What *is* a problem is *accuracy:* even after thousands of iterations, the NNs do not learn well.
- So, instead of computation speed, we would like to maximize learning accuracy.
- We can still consider L and NL elements.
- For the same number of variables w_i, we want to get more accurate approximations.
- For given number of variables, and given accuracy, we get N possible combinations.
- If all combinations correspond to different functions, we can implement N functions.
- However, if some combinations lead to the same function, we implement fewer different functions.

Why Traditional ...
How the Need for Fast ...
Faster Differentiation: ...
Beyond Traditional NN
From Traditional NN ...
Formulation of the ...
What Us Feasible: ...
NP-Hardness Result
Feasibility Resuly

18. From Traditional NN to Deep Learning

- For a traditional NN with K neurons, each of $K!$ permutations of neurons retains the resulting function.
- Thus, instead of N functions, we only implement

$$\frac{N}{K!} \ll N \text{ functions.}$$

- Thus, to increase accuracy, we need to minimize the number K of neurons in each layer.
- To get a good accuracy, we need many parameters, thus many neurons.
- Since each layer is small, we thus need many layers.
- This is the *main idea* behind *deep learning*.

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

19. Computational (Bit) Complexity of NN Learning: Formulation of the Problem

- In general, a NN consists of several layers, each of which has several neurons.
- We feed the inputs to the neurons from the 1st input layer
- A neuron i from each layer generates a signal which is sent to one or more neurons in the next layer.
- The signal generated by a neuron i depends:
  - on signals $x_{i_1}, \ldots, x_{i_k}$ sent to it by neurons of the previous layer,
  - on parameters w_i describing this neuron, and
  - on parameters $w_{i_j,i}$ describing the connection:

$$x_i = f_i(x_{i_1}, \ldots, x_{i_k}, w_i, w_{i_1,i}, \ldots, w_{i_k,i}).$$

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

20. Formulation of the Problem (cont-d)

- Training means finding the values w_i and w_{ij} for which:
  - for all given inputs $(x_1^{(k)}, \ldots, x_n^{(k)})$,
  - the signal of the output layer is sufficiently close to the desired value(s) $y^{(k)}$.
- Let S be the number of bits sufficient to represent each of the values x_i, w_i, or w_{ij}.

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

21. What Is Feasible, What Is A Problem, and What Is NP-Hard: A Brief Reminder

- Some algorithms are feasible, some are not.
- There is no perfect definition of feasibility.
- The best is: an algorithm A is feasible if there exists a polynomial $P(n)$ for which $\forall x\,(t_A(x) \le P(\text{len}(x)))$.
- Some problems can be solved by a feasible algorithm.
- In practice, for most problems:
  - once we have a candidate for a solution,
  - we can feasibly check whether this candidate is indeed a solution.
- For example, in math, once a detailed proof is given, we can check it – but finding the proof id difficult.
- In physics, once a formula is given, we can check whether it fits the data.

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

22. What Is Feasible etc. (cont-d)

- In engineering, we can check whether a given design satisfies specs.
- Such problems are called *Non-deterministic Polynomial* (NP):
  - once we guessed a solution (non-deterministic means guessing is needed),
  - we can feasibly confirm that it is indeed a solution.
- It may be that NP $=$ P, so all problems can be feasibly solved.
- Most computer scientists believe that P $\neq$ NP, but it is still an open problem.

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

23. What Is NP-Hard

- What is known is that some NP problems are harder than others – in the sense that:
  - every problem from the class NP
  - can be reduced to this particular problem.
- These problems are known as *NP-hard*.
- Historically the first example was propositional satisfiability (SAT):
  - given a propositional formula

$$(v_1 \vee \neg v_2) \,\&\, (v_1 \vee v_2 \vee \neg v_3) \,\&\, \ldots,$$

  - check whether it is true for some v_i.

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

24. Perfect Training of a Neural Network Is NP-Hard: A Straightforward Result

- To prove NP-hardness, let us reduce SAT to this problem.
- For each SAT formula with n variables, design a 3-layer network, with 1 pattern, no inputs, $y^{(1)} = 1$.
- Each of n neurons of the first layer has a 1-bit parameter w_i and generates a signal $v_i = w_i$.
- Neurons from 2nd layer compute the truth values of the *clauses* – like $v_1 \vee \neg v_2$,
- A neuron from the 3rd layer applies & to all the results.
- Training here means finding v_i for which the original formula holds.

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

25. Perfect Training Is Feasible for Bounded-Width Deep Networks: A New Result

- Let us assume that each layer has $\le B$ neurons.
- We want to describe the processing of all P patterns in each layer.
- For each layer, to describe its weights and outputs, we need:
  - $\le B$ neurons' parameters,
  - $\le B^2$ connection parameters, and
  - $\le B$ outputs per pattern.
- Overall, we need $\le c \stackrel{\text{def}}{=} (B^2 + B + B \cdot P) \cdot S$ bits.
- The signal from each layer is uniquely determined by signals from the previous layer.
- Let us list all bits layer-by-layer.

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

26. New Result (cont-d)

- We need to find bits that satisfy several conditions each of which connects only bits b_i and b_j with $|i-j| \le 2c$.
- For such *localized* formulas, there is a feasible algorithm for finding bits satisfying all the conditions.
- In this algorithm, at each step $i = 0, 1, \ldots$, we compute:
  - the list L_i of all the tuples $b_i, \ldots, b_{i+2c}$
  - that satisfy all the conditions that involve only bits b_j with $j \le i + 2c$.
- For $i = 0$, we simply check all 2^{2c} ($=$ const) tuples.
- To get from i to $i+1$, for each tuple from L_i, we consider two possible values of a new bit b_{i+1+2c}.
- Due to localization, possible new conditions that involve this bit only involve bits b_j with $j \ge i + 1$.
- So, we can check all these conditions.

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

27. New Result (cont-d)

- For each checked bit, we add the resulting tuple $(b_{i+1}, \ldots, b_{i+1+2c})$ to the list L_{i+1}.
- Each step require a constant time.
- At the end, in time linear in number of layers, we check whether perfect training is possible.
- And we can always go back bit-by-bit and find the corresponding parameters,

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly

28. Acknowledgments

This work was supported in part:

- by Arizona State University, and
- by the US National Science Foundation grant HRD-1242122.

Why Traditional . . .
How the Need for Fast . . .
Faster Differentiation: . . .
Beyond Traditional NN
From Traditional NN . . .
Formulation of the . . .
What Us Feasible: . . .
NP-Hardness Result
Feasibility Resuly