Fast matrix multiplication: a brief adventure in neural networks and computational algebra

Thomas Pietraho
Fall, 2022
A couple of times in my life, I have encountered the following strange statement:

<table>
<thead>
<tr>
<th>Theorem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two ( N \times N ) matrices can be multiplied using only ( N^{2.8074\ldots} ) scalar multiplications.</td>
</tr>
</tbody>
</table>
A Strange Theorem

A couple of times in my life, I have encountered the following strange statement:

<table>
<thead>
<tr>
<th>Theorem</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Two</em> $N \times N$ matrices can be multiplied using only $N^{2.8074...}$ scalar multiplications.*</td>
</tr>
</tbody>
</table>

Intimidated by irrational numbers, I always promptly averted my gaze. What could this statement possibly mean?
A couple of times in my life, I have encountered the following strange statement:

**Theorem**

*Two* $N \times N$ matrices can be multiplied using only $N^{2.8074...}$ scalar multiplications.

Intimidated by irrational numbers, I always promptly averted my gaze. What could this statement possibly mean?

Let’s look at $2 \times 2$ matrices:

\[
\begin{bmatrix}
  a_{11} & a_{12} \\
  a_{21} & a_{22}
\end{bmatrix}
\begin{bmatrix}
  b_{11} & b_{12} \\
  b_{21} & b_{22}
\end{bmatrix}
= 
\begin{bmatrix}
  a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\
  a_{21}b_{11} + a_{22}b_{21} & a_{22}b_{12} + a_{22}b_{22}
\end{bmatrix}
\]

To compute this, $8 = 2^3$ scalar products must be found (and a few scalar sums). Thinking about this, we get

**Theorem**

*Two* $N \times N$ matrices can be multiplied using $N^3$ scalar multiplications.
Note: $N^{2.8074\ldots}$ represents a huge savings over $N^3$. 
Note: $N^{2.8074\ldots}$ represents a huge savings over $N^3$.

For computers
1. addition is fast
2. multiplication is slow
Note: $N^{2.8074...}$ represents a huge savings over $N^3$.

For computers
1. addition is fast
2. multiplication is slow

If $N \approx 10^7$, then
1. $N^{2.8074...} \approx 10^{19.65...}$
2. $N^3 \approx 10^{21}$
Note: $N^{2.8074\ldots}$ represents a huge savings over $N^3$.

For computers

1. addition is fast
2. multiplication is slow

If $N \approx 10^7$, then

1. $N^{2.8074\ldots} \approx 10^{19.65\ldots}$
2. $N^3 \approx 10^{21}$

For multiplication of $10^7 \times 10^7$ matrices, the “strange” theorem cuts the number of scalar multiplications by a factor of about $10^{1.35} \approx 22$. 
Strassen’s observation: $2 \times 2$ matrices

\[
\begin{bmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22}
\end{bmatrix}
\begin{bmatrix}
b_{11} & b_{12} \\
b_{21} & b_{22}
\end{bmatrix}
= 
\begin{bmatrix}
c_{11} & c_{12} \\
c_{21} & c_{22}
\end{bmatrix}
\]

First form products:

\[
m_1 = a_{11}b_{11} \\
m_2 = a_{12}b_{21} \\
m_3 = a_{11}b_{12} \\
m_4 = a_{12}b_{22} \\
m_5 = a_{21}b_{11} \\
m_6 = a_{22}b_{21} \\
m_7 = a_{22}b_{12} \\
m_8 = a_{22}b_{22}
\]
Strassen's observation: $2 \times 2$ matrices

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

First form products:

- $m_1 = a_{11} b_{11}$
- $m_2 = a_{12} b_{21}$
- $m_3 = a_{11} b_{12}$
- $m_4 = a_{12} b_{22}$
- $m_5 = a_{21} b_{11}$
- $m_6 = a_{22} b_{21}$
- $m_7 = a_{22} b_{12}$
- $m_8 = a_{22} b_{22}$

then combine them:

- $c_{11} = m_1 + m_2$
- $c_{12} = m_3 + m_4$
- $c_{21} = m_5 + m_6$
- $c_{22} = m_7 + m_8$
Strassen’s observation: $2 \times 2$ matrices

$$
\begin{bmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22}
\end{bmatrix}
\begin{bmatrix}
b_{11} & b_{12} \\
b_{21} & b_{22}
\end{bmatrix}
= 
\begin{bmatrix}
c_{11} & c_{12} \\
c_{21} & c_{22}
\end{bmatrix}
$$

Strassen formed:

\begin{align*}
m_1 &= (a_{11} + a_{22})(b_{11} + b_{22}) \\
m_2 &= (a_{21} + a_{22})b_{11} \\
m_3 &= a_{11}(b_{12} - b_{22}) \\
m_4 &= a_{22}(b_{21} - b_{11}) \\
m_5 &= (a_{11} + a_{12})b_{22} \\
m_6 &= (a_{21} - a_{11})(b_{11} + b_{12}) \\
m_7 &= (a_{12} - a_{22})(b_{21} + b_{22})
\end{align*}

and combined them:

\begin{align*}
c_{11} &= m_1 + m_4 - m_5 + m_7 \\
c_{12} &= m_3 + m_5 \\
c_{21} &= m_2 + m_4 \\
c_{22} &= m_1 - m_2 + m_3 + m_6
\end{align*}

Theorem

$2 \times 2$ matrices can be multiplied using 7 only scalar multiplications!
Strassen's observation: $2 \times 2$ matrices

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

Strassen formed:

$$m_1 = (a_{11} + a_{22})(b_{11} + b_{22})$$
$$m_2 = (a_{21} + a_{22})b_{11}$$
$$m_3 = a_{11}(b_{12} - b_{22})$$
$$m_4 = a_{22}(b_{21} - b_{11})$$
$$m_5 = (a_{11} + a_{12})b_{22}$$
$$m_6 = (a_{21} - a_{11})(b_{11} + b_{12})$$
$$m_7 = (a_{12} - a_{22})(b_{21} + b_{22})$$

and combined them:

$$c_{11} = m_1 + m_4 - m_5 + m_7$$
$$c_{12} = m_3 + m_5$$
$$c_{21} = m_2 + m_4$$
$$c_{22} = m_1 - m_2 + m_3 + m_6$$

Theorem

$2 \times 2$ matrices can be multiplied using 7 only scalar multiplications!
Strassen’s observation: 2 × 2 matrices

\[
\begin{bmatrix}
  a_{11} & a_{12} \\
  a_{21} & a_{22}
\end{bmatrix}
\begin{bmatrix}
  b_{11} & b_{12} \\
  b_{21} & b_{22}
\end{bmatrix}
= \begin{bmatrix}
  c_{11} & c_{12} \\
  c_{21} & c_{22}
\end{bmatrix}
\]

Strassen formed:

\[
m_1 = (a_{11} + a_{22})(b_{11} + b_{22})
\]

\[
m_2 = (a_{21} + a_{22})b_{11}
\]

\[
m_3 = a_{11}(b_{12} - b_{22})
\]

\[
m_4 = a_{22}(b_{21} - b_{11})
\]

\[
m_5 = (a_{11} + a_{12})b_{22}
\]

\[
m_6 = (a_{21} - a_{11})(b_{11} + b_{12})
\]

\[
m_7 = (a_{12} - a_{22})(b_{21} + b_{22})
\]

and combined them:

\[
c_{11} = m_1 + m_4 - m_5 + m_7
\]

\[
c_{12} = m_3 + m_5
\]

\[
c_{21} = m_2 + m_4
\]

\[
c_{22} = m_1 - m_2 + m_3 + m_6
\]

**Theorem**

2 × 2 matrices can be multiplied using 7 only scalar multiplications!
Strassen’s observation: $2k \times 2k$ block matrices

\[
\begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix}
\begin{bmatrix}
B_{11} & B_{12} \\
B_{21} & B_{22}
\end{bmatrix}
= \begin{bmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{bmatrix}
\]

Form the products:

\[
M_1 = (A_{11} + A_{22})(B_{11} + B_{22})
\]
\[
M_2 = (A_{21} + A_{22})B_{11}
\]
\[
M_3 = A_{11}(B_{12} - B_{22})
\]
\[
M_4 = A_{22}(B_{21} - B_{11})
\]
\[
M_5 = (A_{11} + A_{12})B_{22}
\]
\[
M_6 = (A_{21} - A_{11})(B_{11} + B_{12})
\]
\[
M_7 = (A_{12} - A_{22})(B_{21} + B_{22})
\]
Strassen’s observation: $2k \times 2k$ block matrices

\[
\begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix}
\begin{bmatrix}
B_{11} & B_{12} \\
B_{21} & B_{22}
\end{bmatrix}
= \begin{bmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{bmatrix}
\]

Form the products:

\begin{align*}
M_1 &= (A_{11} + A_{22})(B_{11} + B_{22}) \\
M_2 &= (A_{21} + A_{22})B_{11} \\
M_3 &= A_{11}(B_{12} - B_{22}) \\
M_4 &= A_{22}(B_{21} - B_{11}) \\
M_5 &= (A_{11} + A_{12})B_{22} \\
M_6 &= (A_{21} - A_{11})(B_{11} + B_{12}) \\
M_7 &= (A_{12} - A_{22})(B_{21} + B_{22})
\end{align*}

then combine them:

\begin{align*}
C_{11} &= M_1 + M_4 - M_5 + M_7 \\
C_{12} &= M_3 + M_5 \\
C_{21} &= M_2 + M_4 \\
C_{22} &= M_1 - M_2 + M_3 + M_6
\end{align*}
This sets off a chain of consequences:

1. $4 \times 4$ matrices can be multiplied using 49 scalar multiplications.
This sets off a chain of consequences:

1. $4 \times 4$ matrices can be multiplied using 49 scalar multiplications.
2. If $n = 2^k$, then $n \times n$ matrices can be multiplied using $7^k$ scalar multiplications.
This sets off a chain of consequences:

1. $4 \times 4$ matrices can be multiplied using 49 scalar multiplications.
2. If $n = 2^k$, then $n \times n$ matrices can be multiplied using $7^k$ scalar multiplications.
3. If $n = 2^k$, then $n \times n$ matrices can be multiplied using $2^k \log_2 7$ scalar multiplications.

With a little slight-of-hand, we write $N = 2^k$ concluding:

Theorem (Strassen)

$N \times N$ matrices can be multiplied using $N \log_2 7 \approx N^2$ scalar multiplications.

Note: This is only technically true for $N = 2^k$ for some $k$, but most people just gloss this over and say the theorem is true "asymptotically."

Question:

Can one do better? $2 \times 2$ matrices using only 6 scalar multiplications? Or can one reduce the exponent $2^{\ldots}$ some other way?
This sets off a chain of consequences:

1. $4 \times 4$ matrices can be multiplied using 49 scalar multiplications.
2. If $n = 2^k$, then $n \times n$ matrices can be multiplied using $7^k$ scalar multiplications.
3. If $n = 2^k$, then $n \times n$ matrices can be multiplied using $2^k \log_2 7$ scalar multiplications.

With a little slight-of-hand, we write $N = 2^k$ concluding:

**Theorem (Strassen)**

$N \times N$ matrices can be multiplied using $N^{\log_2 7} \approx N^{2.8074\ldots}$ scalar multiplications.
This sets off a chain of consequences:

1. $4 \times 4$ matrices can be multiplied using 49 scalar multiplications.
2. If $n = 2^k$, then $n \times n$ matrices can be multiplied using $7^k$ scalar multiplications.
3. If $n = 2^k$, then $n \times n$ matrices can be multiplied using $2^k \log_2 7$ scalar multiplications.

With a little slight-of-hand, we write $N = 2^k$ concluding:

Table (Strassen)

<table>
<thead>
<tr>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N \times N$ matrices can be multiplied using $N^{\log_2 7} \approx N^{2.8074...}$ scalar multiplications.</td>
</tr>
</tbody>
</table>

Note: This is only technically true for $N = 2^k$ for some $k$, but most people just gloss this over and say the theorem is true “asymptotically.”
This sets off a chain of consequences:

1. $4 \times 4$ matrices can be multiplied using 49 scalar multiplications.
2. If $n = 2^k$, then $n \times n$ matrices can be multiplied using $7^k$ scalar multiplications.
3. If $n = 2^k$, then $n \times n$ matrices can be multiplied using $2^k \log_2 7$ scalar multiplications.

With a little slight-of-hand, we write $N = 2^k$ concluding:

**Theorem (Strassen)**

$N \times N$ matrices can be multiplied using $N^{\log_2 7} \approx N^{2.8074...}$ scalar multiplications.

**Note:** This is only technically true for $N = 2^k$ for some $k$, but most people just gloss this over and say the theorem is true “asymptotically.”

**Question:** Can one do better?
This sets off a chain of consequences:

1. $4 \times 4$ matrices can be multiplied using 49 scalar multiplications.
2. If $n = 2^k$, then $n \times n$ matrices can be multiplied using $7^k$ scalar multiplications.
3. If $n = 2^k$, then $n \times n$ matrices can be multiplied using $2^k \log_2 7$ scalar multiplications.

With a little slight-of-hand, we write $N = 2^k$ concluding:

**Theorem (Strassen)**

$N \times N$ matrices can be multiplied using $N \log_2 7 \approx N^{2.8074\ldots}$ scalar multiplications.

**Note:** This is only technically true for $N = 2^k$ for some $k$, but most people just gloss this over and say the theorem is true “asymptotically.”

**Question:** Can one do better? $2 \times 2$ matrices using only 6 scalar multiplications?
This sets off a chain of consequences:

1. $4 \times 4$ matrices can be multiplied using 49 scalar multiplications.
2. If $n = 2^k$, then $n \times n$ matrices can be multiplied using $7^k$ scalar multiplications.
3. If $n = 2^k$, then $n \times n$ matrices can be multiplied using $2^k \log_2 7$ scalar multiplications.

With a little slight-of-hand, we write $N = 2^k$ concluding:

**Theorem (Strassen)**

$N \times N$ matrices can be multiplied using $N^{\log_2 7} \approx N^{2.8074\ldots}$ scalar multiplications.

**Note:** This is only technically true for $N = 2^k$ for some $k$, but most people just gloss this over and say the theorem is true “asymptotically.”

**Question:** Can one do better? $2 \times 2$ matrices using only 6 scalar multiplications? Or can one reduce the exponent 2.8074\ldots some other way?
Definition

Let \textbf{MultRank}(N) be the minimum number of scalar multiplications necessary to multiply two \( N \times N \) matrices. Let \textbf{MultExp}(N) be the corresponding exponent.

Similar advances:

- Theorem (Laderman, 1976) \( \text{MultRank}(3) \leq 23 \).
- Theorem (Waksman, 1970) \( \text{MultRank}(2,2,3) \leq 11 \).
- Theorem (Hopcroft and Kerr, 1971) \( \text{MultRank}(2,3,3) \leq 15 \).
- Theorem (Strassen, 1969) \( \text{MultRank}(4) \leq 49 \).

Lower bounds:

- Theorem (Winograd, 1971) \( 7 \leq \text{MultRank}(2) \).
- Theorem (Bläser, 2003) \( 19 \leq \text{MultRank}(3) \leq 23 \).
- \( 10 \leq \text{MultRank}(2,2,3) \leq 11 \).
- \( 14 \leq \text{MultRank}(2,3,3) \leq 15 \).
- \( 33 \leq \text{MultRank}(4) \leq 49 \).

There is potential for significant improvement in existing algorithms when \( N \geq 3 \).
Other efforts and theoretical bounds

Definition

Let $\text{MultRank}(N)$ be the minimum number of scalar multiplications necessary to multiply two $N \times N$ matrices. Let $\text{MultExp}(N)$ be the corresponding exponent.

Similar advances:

Theorem (Laderman, 1976)

$\text{MultRank}(3) \leq 23$.

Theorem (Waksman, 1970)

$\text{MultRank}(2, 2, 3) \leq 11$.

Theorem (Hopcroft and Kerr, 1971)

$\text{MultRank}(2, 3, 3) \leq 15$.

Theorem (Strassen, 1969)

$\text{MultRank}(4) \leq 49$.
### Other efforts and theoretical bounds

#### Definition

Let MultRank($N$) be the minimum number of scalar multiplications necessary to multiply two $N \times N$ matrices. Let MultExp($N$) be the corresponding exponent.

#### Similar advances:

- **Theorem (Laderman, 1976)**
  
  \[
  \text{MultRank}(3) \leq 23.
  \]

- **Theorem (Waksman, 1970)**
  
  \[
  \text{MultRank}(2, 2, 3) \leq 11.
  \]

- **Theorem (Hopcroft and Kerr, 1971)**
  
  \[
  \text{MultRank}(2, 3, 3) \leq 15.
  \]

- **Theorem (Strassen, 1969)**
  
  \[
  \text{MultRank}(4) \leq 49.
  \]

#### Lower bounds:

- **Theorem (Winograd, 1971)**
  
  \[
  7 \leq \text{MultRank}(2)
  \]

- **Theorem (Bläser, 2003)**
  
  \[
  19 \leq \text{MultRank}(3) \leq 23 \\
  10 \leq \text{MultRank}(2, 2, 3) \leq 11 \\
  14 \leq \text{MultRank}(2, 3, 3) \leq 15 \\
  33 \leq \text{MultRank}(4) \leq 49
  \]

There is potential for significant improvement in existing algorithms when $N \geq 3$. 
Question: This is going to be a huge mess. How could one possibly improve any of these results without reams of computations?
Neural networks: a brief introduction
A neural net $\mathcal{N}$ is an object:

\[
\begin{align*}
\begin{array}{cccccc}
 & y_1 & y_2 & y_3 & y_4 & \cdots & y_m \\
\hline \\
\text{Input} & x_1 & x_2 & x_3 & \cdots & x_n \\
\end{array}
\end{align*}
\]

It is a fancy way to produce a function:

\[F_{\mathcal{N}} : \mathbb{R}^n \rightarrow \mathbb{R}^m.\]
Neural nets are made of “neurons”

\[ \sigma(\sum w_i x_i + b_i) \]

Where \( \sigma \) is a function:
Neural nets are made of “neurons”

\[ \sigma(\sum w_i x_i + b_i) \]

Where \( \sigma \) is a function:

[Graph showing a bell curve]
Neural nets are made of “neurons”

\[
\sigma \left( \sum w_i x_i + b_i \right)
\]

Where \( \sigma \) is a function:

\[
\begin{array}{c}
\text{Value} \\
0.0 \\
0.2 \\
0.4 \\
0.6 \\
0.8 \\
1.0 \\
1.2 \\
\end{array}
\]
Neural nets are made of “neurons”

\[ \sigma(\sum w_i x_i + b_i) \]

Where \( \sigma \) is a function:
Neurons in layers make a neural network

Each edge may have a different $w$ called its “weight”. Each neuron may have a different $b$ called its “bias.”
Neurons in many layers make a “deep” neural net

\[ h_1, h_2, h_3, h_4, \ldots, h_k, h_1', h_2', h_3', h_4', h_5', \ldots, h_p', h_1'', h_2'', h_3'', h_4'', \ldots, h_p'' \]

Input Layer \( \in \mathbb{R}^n \)

Hidden Layer \( \in \mathbb{R}^k \)

Hidden Layer \( \in \mathbb{R}^l \)

Hidden Layer \( \in \mathbb{R}^p \)

Output Layer \( \in \mathbb{R}^m \)
The problem in deep learning:

Given, a perhaps not fully understood function $F$, find a neural network $\mathcal{N}$ that recovers $F$. That is:

$$F_{\mathcal{N}} \approx F.$$
The problem in deep learning:

Given, a perhaps not fully understood function $F$, find a neural network $\mathcal{N}$ that recovers $F$. That is:

$$F\mathcal{N} \approx F.$$
The problem in deep learning:

Given, a perhaps not fully understood function $F$, find a neural network $\mathcal{N}$ that recovers $F$. That is:

$$F_\mathcal{N} \approx F.$$ 

Here each handwritten digit is given by a $28 \times 28$ array of greyscale pixels. We’d like to understand

$$F : \mathbb{R}^{784} \rightarrow \mathbb{R}$$

or better still:

$$F : \mathbb{R}^{784} \rightarrow \mathbb{R}^{10}$$

This neural net is 85% accurate:

Image by A. Nielsen
**ImageNet Challenge:** Given $256 \times 256$ RGB images classified into 1000 classes. Find a neural network $\mathcal{N}$ that describes the classification function:

$$F : \mathbb{R}^{3 \cdot 256^2} \rightarrow \mathbb{R}^{1000}.$$
ImageNet Challenge: Given $256 \times 256$ RGB images classified into 1000 classes. Find a neural network $\mathcal{N}$ that describes the classification function:

$$F : \mathbb{R}^{3 \cdot 256^2} \rightarrow \mathbb{R}^{1000}.$$ 

Google's Inception neural net $\mathcal{N}$ achieves 95% top-5 accuracy. The big picture of the neural net:

Image by Google
Fun Problem: Predict species of bird based on photographic image.

cardinal    wood duck    anhinga    chickadee

Accuracy 87%. (P., 2017)
Fun Problem: Predict book genre based on its cover.

Accuracy 76%. (with Parikshit Sharma, ’17, IndieBio)
The Whole Process

**Goal:** Understand a, perhaps poorly defined, function $F$.

1. Start with a set of data points:
   
   $$(x_i, F(x_i))$$
The Whole Process

**Goal:** Understand a, perhaps poorly defined, function $F$.

1. Start with a set of data points:
   
   $$(x_i, F(x_i))$$

2. Build a neural network $\mathcal{N}$
**Goal:** Understand a, perhaps poorly defined, function $F$.

1. Start with a set of data points:

   $$(x_i, F(x_i))$$

2. Build a neural network $\mathcal{N}$
The Whole Process

**Goal:** Understand a, perhaps poorly defined, function $F$.

1. Start with a set of data points:
   \[(x_i, F(x_i))\]

2. Build a neural network $\mathcal{N}$

3. Compare with output of $\mathcal{N}$:
   \[(x_i, F_{\mathcal{N}}(x_i))\]

4. Tweak weights $w$ and bias $b$ decreasing
   \[\text{Error} = \text{ave} |F(x_i) - F_{\mathcal{N}}(x_i)|\]

5. Continue tweaking $w$ and bias $b$ until error is as small as possible

6. Sell your trained neural net to a startup.

7. Buy fancy coffee maker for Math Dept.
The Whole Process

**Goal:** Understand a, perhaps poorly defined, function $F$.

1. Start with a set of data points:
   \[(x_i, F(x_i))\]

2. Build a neural network $\mathcal{N}$

3. Compare with output of $\mathcal{N}$:
   \[(x_i, F_{\mathcal{N}}(x_i))\]

4. Tweak weights $w$ and bias $b$ decreasing
   \[
   \text{Error} = \text{ave}|F(x_i) - F_{\mathcal{N}}(x_i)|
   \]
The Whole Process

**Goal:** Understand a, perhaps poorly defined, function $F$.

1. Start with a set of data points:
   
   $$(x_i, F(x_i))$$

2. Build a neural network $\mathcal{N}$

3. Compare with output of $\mathcal{N}$:
   
   $$(x_i, F_\mathcal{N}(x_i))$$

4. Tweak weights $w$ and bias $b$ decreasing
   
   $$\text{Error} = \text{ave}|F(x_i) - F_\mathcal{N}(x_i)|$$

5. Continue tweaking $w$ and bias $b$ until error is as small as possible

6. Sell your trained neural net to a startup.

7. Buy fancy coffee maker for Math Dept.
**Goal:** Understand a, perhaps poorly defined, function $F$.

1. Start with a set of data points:

   $$(x_i, F(x_i))$$

2. Build a neural network $\mathcal{N}$

3. Compare with output of $\mathcal{N}$:

   $$(x_i, F_{\mathcal{N}}(x_i))$$

4. Tweak weights $w$ and bias $b$ decreasing

   $$\text{Error} = \text{ave}|F(x_i) - F_{\mathcal{N}}(x_i)|$$

5. Continue tweaking $w$ and bias $b$ until error is as small as possible

6. Sell your trained neural net to a startup.

7. Buy fancy coffee maker for Math Dept.
Goal: Understand a, perhaps poorly defined, function $F$.

1. Start with a set of data points:

   $$(x_i, F(x_i))$$

2. Build a neural network $\mathcal{N}$

3. Compare with output of $\mathcal{N}$:

   $$(x_i, F_{\mathcal{N}}(x_i))$$

4. Tweak weights $w$ and bias $b$ decreasing

   $$\text{Error} = \text{ave} |F(x_i) - F_{\mathcal{N}}(x_i)|$$

5. Continue tweaking $w$ and bias $b$ until error is as small as possible

6. Sell your trained neural net to a startup.

7. Buy fancy coffee maker for Math Dept.
The Whole Process

**Goal:** Understand a, perhaps poorly defined, function $F$.

1. Start with a set of data points:

   $$(x_i, F(x_i))$$

2. Build a neural network $\mathcal{N}$

3. Compare with output of $\mathcal{N}$:

   $$(x_i, F_\mathcal{N}(x_i))$$

4. Tweak weights $w$ and bias $b$ decreasing

   Error = \text{ave}|F(x_i) - F_\mathcal{N}(x_i)|

5. Continue tweaking $w$ and bias $b$ until error is as small as possible

6. Sell your trained neural net to a startup.

7. Buy fancy coffee maker for Math Dept.
The Whole Process

**Goal:** Understand a, perhaps poorly defined, function $F$.

1. Start with a set of data points:

   $$(x_i, F(x_i))$$

2. Build a neural network $\mathcal{N}$

3. Compare with output of $\mathcal{N}$:

   $$(x_i, F_\mathcal{N}(x_i))$$

4. Tweak weights $w$ and bias $b$ decreasing

   $$\text{Error} = \text{ave} |F(x_i) - F_\mathcal{N}(x_i)|$$

5. Continue tweaking $w$ and bias $b$ until error is as small as possible
**The Whole Process**

**Goal:** *Understand a, perhaps poorly defined, function $F$.*

1. Start with a set of data points:
   $$(x_i, F(x_i))$$

2. Build a neural network $\mathcal{N}$

3. Compare with output of $\mathcal{N}$:
   $$(x_i, F_{\mathcal{N}}(x_i))$$

4. Tweak weights $w$ and bias $b$ decreasing
   \[
   \text{Error} = \text{ave}|F(x_i) - F_{\mathcal{N}}(x_i)|
   \]

5. Continue tweaking $w$ and bias $b$ until error is as small as possible

6. Sell your trained neural net to a startup.

7. Buy fancy coffee maker for Math Dept.
The Whole Process

**Goal:** *Understand a, perhaps poorly defined, function F.*

1. Start with a set of data points:

   \[(x_i, F(x_i))\]

2. Build a neural network \( \mathcal{N} \)

3. Compare with output of \( \mathcal{N} \):

   \[(x_i, F_\mathcal{N}(x_i))\]

4. Tweak weights \( w \) and bias \( b \) decreasing

   \[
   \text{Error} = \text{ave} |F(x_i) - F_\mathcal{N}(x_i)|
   \]

5. Continue tweaking \( w \) and bias \( b \) until error is as small as possible

6. Sell your trained neural net to a startup.

7. Buy fancy coffee maker for Math Dept.
The Whole Process

**Goal:** Understand a, perhaps poorly defined, function $F$.

1. Start with a set of data points:
   
   $$(x_i, F(x_i))$$

2. Build a neural network $\mathcal{N}$

3. Compare with output of $\mathcal{N}$:
   
   $$(x_i, F_\mathcal{N}(x_i))$$

4. Tweak weights $w$ and bias $b$ decreasing
   
   Error = ave$|F(x_i) - F_\mathcal{N}(x_i)|$

5. Continue tweaking $w$ and bias $b$ until error is as small as possible
The Whole Process

**Goal:** Understand a, perhaps poorly defined, function $F$.

1. Start with a set of data points:
   
   $$(x_i, F(x_i))$$

2. Build a neural network $\mathcal{N}$

3. Compare with output of $\mathcal{N}$:
   
   $$(x_i, F_{\mathcal{N}}(x_i))$$

4. Tweak weights $w$ and bias $b$ decreasing
   
   $$\text{Error} = \text{ave}|F(x_i) - F_{\mathcal{N}}(x_i)|$$

5. Continue tweaking $w$ and bias $b$ until error is as small as possible

6. Sell your trained neural net to a startup.

Diagram:

- Input nodes: $x$
- Output node: $y$
- Hidden layers: $\mathcal{N}$
The Whole Process

**Goal:** Understand a, perhaps poorly defined, function $F$.

1. Start with a set of data points:
   \[(x_i, F(x_i))\]

2. Build a neural network $\mathcal{N}$

3. Compare with output of $\mathcal{N}$:
   \[(x_i, F_{\mathcal{N}}(x_i))\]

4. Tweak weights $w$ and bias $b$ decreasing
   \[
   \text{Error} = \text{ave}|F(x_i) - F_{\mathcal{N}}(x_i)|
   \]

5. Continue tweaking $w$ and bias $b$ until error is as small as possible

6. Sell your trained neural net to a startup.

7. Buy fancy coffee maker for Math Dept.
A machine learning approach to fast matrix multiplication
**Goal:** Design a neural network that mimics $2 \times 2$ matrix multiplication:

$$F : \mathbb{R}^{2 \times 4} \rightarrow \mathbb{R}^4$$

$$F(A, B) = A \cdot B$$
Goal: Design a neural network that mimics $2 \times 2$ matrix multiplication:

$$F: \mathbb{R}^{2 \times 4} \rightarrow \mathbb{R}^4$$

$$F(A, B) = A \cdot B$$

Step 1: Start with a set of data points:

$$(x_i, F(x_i))$$

This is easy. Generate lots of random $2 \times 2$ matrices $x_i = (A_i, B_i)$ as well as their products $F(x_i) = A_i \cdot B_i$. 
Goal: Design a neural network that mimics $2 \times 2$ matrix multiplication:

$$F : \mathbb{R}^{2 \cdot 4} \rightarrow \mathbb{R}^4$$

$$F(A, B) = A \cdot B$$

Step 2: Build a neural network $\mathcal{N}$
Goal: Design a neural network that mimics $2 \times 2$ matrix multiplication:

$$F : \mathbb{R}^{2 \times 4} \rightarrow \mathbb{R}^4$$

$$F(A, B) = A \cdot B$$

Step 2: Build a neural network $\mathcal{N}$

Need: A new type of neuron. One whose output is the product of its two inputs.

A new type of neural net!
Neural Net for matrix multiplication

\[ a_{11}b_{11} + a_{12}b_{21} + a_{11}b_{12} + a_{12}b_{22} + a_{21}b_{11} + a_{22}b_{21} + a_{21}b_{12} + a_{22}b_{22} \]
Neural Net for matrix multiplication

\[ \begin{align*}
  &a_{11}b_{11} + a_{12}b_{21} \\
  &a_{11}b_{12} + a_{12}b_{22} \\
  &a_{21}b_{11} + a_{22}b_{21} \\
  &a_{21}b_{12} + a_{22}b_{22}
\end{align*} \]
**Step 3:** Compare with output of $\mathcal{N}$:

$$(x_i, F_\mathcal{N}(x_i))$$

**Step 4:** Tweak weights $w$ and bias $b$ for each edge so that

$$\text{Error} = \text{ave} |F(x_i) - F_\mathcal{N}(x_i)|$$

decreases.

**Step 5:** Continue tweaking $w$ and bias $b$ until error is as small as possible
Step 3: Compare with output of $\mathcal{N}$:

$$(x_i, F_N(x_i))$$

Step 4: Tweak weights $w$ and bias $b$ for each edge so that

$$\text{Error} = \text{ave}|F(x_i) - F_N(x_i)|$$
decreases.

Step 5: Continue tweaking $w$ and bias $b$ until error is as small as possible

The Result:
Machine-trained neural net for matrix multiplication

\[
\begin{align*}
    a_{11}b_{11} + a_{12}b_{21} + a_{11}b_{12} + a_{12}b_{22} + a_{21}b_{11} + a_{22}b_{21} + a_{21}b_{12} + a_{22}b_{22}
\end{align*}
\]
Neural Net for Strassen’s matrix multiplication

\[ a_{11}b_{11} + a_{12}b_{21} \]
\[ a_{11}b_{12} + a_{12}b_{22} \]
\[ a_{21}b_{11} + a_{22}b_{21} \]
\[ a_{21}b_{12} + a_{22}b_{22} \]
Neural Net for Strassen’s matrix multiplication

\[
\begin{array}{c}
a_{11}b_{11} + a_{12}b_{21} \\
a_{11}b_{12} + a_{12}b_{22} \\
a_{21}b_{11} + a_{22}b_{21} \\
a_{21}b_{12} + a_{22}b_{22}
\end{array}
\]
Machine-trained neural net for Strassen’s matrix multiplication
In one day, our new fancied-up neural nets replicated:

**Theorem (Strassen, 1969)**  
$\text{MultRank}(2) \leq 7$

**Theorem (Laderman, 1976)**  
$\text{MultRank}(3) \leq 23$

**Theorem (Waksman, 1970)**  
$\text{MultRank}(2, 2, 3) \leq 11$

**Theorem (Hopcroft and Kerr, 1971)**  
$\text{MultRank}(2, 3, 3) \leq 15$

How can you tell this actually works?  
Plot error vs. training time.
In one day, our new fancied-up neural nets replicated:

**Theorem (Strassen, 1969)**
\[
\text{MultRank}(2) \leq 7
\]

**Theorem (Laderman, 1976)**
\[
\text{MultRank}(3) \leq 23
\]

How can you tell this actually works?
Plot error vs. training time.
In one day, our new fancied-up neural nets replicated:

Theorem (Strassen, 1969)
\[ \text{MultRank}(2) \leq 7 \]

Theorem (Laderman, 1976)
\[ \text{MultRank}(3) \leq 23 \]

Theorem (Waksman, 1970)
\[ \text{MultRank}(2, 2, 3) \leq 11 \]
In one day, our new fancied-up neural nets replicated:

**Theorem (Strassen, 1969)**
\[\text{MultRank}(2) \leq 7\]

**Theorem (Laderman, 1976)**
\[\text{MultRank}(3) \leq 23\]

**Theorem (Waksman, 1970)**
\[\text{MultRank}(2, 2, 3) \leq 11\]

**Theorem (Hopcroft and Kerr, 1971)**
\[\text{MultRank}(2, 3, 3) \leq 15\]
In one day, our new fancied-up neural nets replicated:

<table>
<thead>
<tr>
<th>Theorem (Strassen, 1969)</th>
<th>MultRank(2) ≤ 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theorem (Laderman, 1976)</td>
<td>MultRank(3) ≤ 23</td>
</tr>
<tr>
<td>Theorem (Waksman, 1970)</td>
<td>MultRank(2, 2, 3) ≤ 11</td>
</tr>
<tr>
<td>Theorem (Hopcroft and Kerr, 1971)</td>
<td>MultRank(2, 3, 3) ≤ 15</td>
</tr>
</tbody>
</table>

How can you tell this actually works?

Plot error vs. training time.
In one day, our new fancied-up neural nets replicated:

<table>
<thead>
<tr>
<th>Theorem (Strassen, 1969)</th>
<th>MultRank(2) ≤ 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theorem (Laderman, 1976)</td>
<td>MultRank(3) ≤ 23</td>
</tr>
<tr>
<td>Theorem (Waksman, 1970)</td>
<td>MultRank(2, 2, 3) ≤ 11</td>
</tr>
<tr>
<td>Theorem (Hopcroft and Kerr, 1971)</td>
<td>MultRank(2, 3, 3) ≤ 15</td>
</tr>
</tbody>
</table>

How can you tell this actually works?

Plot error vs. training time.
In one day, our new fancied-up neural nets replicated:

**Theorem (Strassen, 1969)**
\[ \text{MultRank}(2) \leq 7 \]

**Theorem (Laderman, 1976)**
\[ \text{MultRank}(3) \leq 23 \]

**Theorem (Waksman, 1970)**
\[ \text{MultRank}(2, 2, 3) \leq 11 \]

**Theorem (Hopcroft and Kerr, 1971)**
\[ \text{MultRank}(2, 3, 3) \leq 15 \]

How can you tell this actually works?

Plot error vs. training time.

![Graph showing error vs. training time](image)

**Figure 3**: \( N = 2 \) Rank = 7
In one day, our new fancied-up neural nets replicated:

**Theorem (Strassen, 1969)**
\[ \text{MultRank}(2) \leq 7 \]

**Theorem (Laderman, 1976)**
\[ \text{MultRank}(3) \leq 23 \]

**Theorem (Waksman, 1970)**
\[ \text{MultRank}(2, 2, 3) \leq 11 \]

**Theorem (Hopcroft and Kerr, 1971)**
\[ \text{MultRank}(2, 3, 3) \leq 15 \]

How can you tell this actually works?

Plot error vs. training time.

**Figure 3**: \( N = 3 \) Rank = 23
Figure 4: $N = 2$ Rank = 6
Figure 4: $N = 3$ Rank = 22
Failures

Figure 4: $N = 3$ Rank = 21
Figure 4: $N = 2, 2, 3$ Rank = 10
I was about to call an end to all of this, but then a recent preprint mentioned a forgotten result:

<table>
<thead>
<tr>
<th>Theorem (Stothers, 2011)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MultRank(4) ≤ 48.</td>
</tr>
</tbody>
</table>
I was about to call an end to all of this, but then a recent preprint mentioned a forgotten result:

**Theorem (Stothers, 2011)**

\[
\text{MultRank}(4) \leq 48.
\]

It is the first result that has beat Strassen’s exponent! Here

\[
\text{MultExp}(4) \leq \log_4 48 \approx 2.7924 \ldots
\]
I was about to call an end to all of this, but then a recent preprint mentioned a forgotten result:

**Theorem (Stothers, 2011)**

\[
\text{MultRank}(4) \leq 48.
\]

It is the first result that has beat Strassen’s exponent! Here

\[
\text{MultExp}(4) \leq \log_4 48 \approx 2.7924 \ldots
\]

**Question:** Can a computer figure this out?
I was about to call an end to all of this, but then a recent preprint mentioned a forgotten result:

**Theorem (Stothers, 2011)**

\[
\text{MultRank}(4) \leq 48.
\]

It is the first result that has beat Strassen’s exponent! Here

\[
\text{MultExp}(4) \leq \log_4 48 \approx 2.7924 \ldots
\]

**Question:** Can a computer figure this out?

Yes!

**Figure 5:** $N = 4$ Rank = 48
I was about to call an end to all of this, but then a recent preprint mentioned a forgotten result:

**Theorem (Stothers, 2011)**

\[ \text{MultRank}(4) \leq 48. \]

It is the first result that has beat Strassen’s exponent! Here

\[ \text{MultExp}(4) \leq \log_4 48 \approx 2.7924 \ldots \]

**Question:** Can a computer figure this out?

Yes! Also:

**Figure 5:** \( N = 4 \) Rank = 47
I was about to call an end to all of this, but then a recent preprint mentioned a forgotten result:

**Theorem (Stothers, 2011)**

$$\text{MultRank}(4) \leq 48.$$  

It is the first result that has beat Strassen’s exponent! Here

$$\text{MultExp}(4) \leq \log_4 48 \approx 2.7924 \ldots$$

**Question:** Can a computer figure this out?

Yes! Also:

**Figure 5:** $N = 4$ \textbf{Rank} $= 46$
I was about to call an end to all of this, but then a recent preprint mentioned a forgotten result:

**Theorem (Stothers, 2011)**

\[ \text{MultRank}(4) \leq 48. \]

It is the first result that has beat Strassen’s exponent! Here

\[ \text{MultExp}(4) \leq \log_4 48 \approx 2.7924 \ldots \]

**Question:** Can a computer figure this out?

Yes! Also:

**Figure 5:** \( N = 4 \) Rank = 45
I was about to call an end to all of this, but then a recent preprint mentioned a forgotten result:

**Theorem (Stothers, 2011)**

\[
\text{MultRank}(4) \leq 48.
\]

It is the first result that has beat Strassen’s exponent! Here

\[
\text{MultExp}(4) \leq \log_4 48 \approx 2.7924\ldots
\]

**Question:** Can a computer figure this out?

---

**Figure 5:** \(N = 4\) \(\text{Rank} = 33\)
Conjecture: One can multiply $4 \times 4$ matrices with fewer than 48 scalar multiplications. In fact, it seems that

$$\text{MultRank}(4) \leq 45.$$ 

Asymptotically, this would give $\text{MultExp}(4) \leq \log_4 45 = 2.7459\ldots$. 

Note this is only a conjecture. My neural networks were only approximations.
Conjecture: One can multiply $4 \times 4$ matrices with fewer than 48 scalar multiplications. In fact, it seems that $$\text{MultRank}(4) \leq 45.$$ Asymptotically, this would give $$\text{MultExp}(4) \leq \log_4 45 = 2.7459 \ldots.$$
Conjecture: One can multiply $4 \times 4$ matrices with fewer than 48 scalar multiplications. In fact, it seems that

$$\text{MultRank}(4) \leq 45.$$ 

Asymptotically, this would give

$$\text{MultExp}(4) \leq \log_4 45 = 2.7459 \ldots$$

Note this is only a conjecture. My neural networks were only approximations. What remains:

1. Find an exact version of this algorithm.
2. Find an equivalent sparse neural net. Preferably one whose non-zero weights equal $\pm 1$.

Conjecture:

$$\lim_{N \to \infty} \text{MultExp}(N) = 2$$

(Josh Alman and Virginia Williams, 2021.)
Conjecture: One can multiply $4 \times 4$ matrices with fewer than 48 scalar multiplications. In fact, it seems that

$$\text{MultRank}(4) \leq 45.$$  

Asymptotically, this would give $\text{MultExp}(4) \leq \log_4 45 = 2.7459 \ldots$.

Note this is only a conjecture. My neural networks were only approximations. What remains:

1. Find an exact version of this algorithm.
2. Find an equivalent sparse neural net. Preferably one whose non-zero weights equal $\pm 1$.

Conjecture:

$$\lim_{N \to \infty} \text{MultExp}(N) = 2$$

Currently, it is known $\lim_{N \to \infty} \text{MultExp}(N) < 2.3728 \ldots$ (Josh Alman and Virginia Williams, 2021).
Discovering novel algorithms with AlphaTensor

This sheds light on a 50-year-old open question in mathematics about finding the fastest way to multiply two matrices.

3 weeks ago

DeepMind AI finds new way to multiply numbers and speed up ...

Matrix multiplication – where two grids of numbers are multiplied together ... But DeepMind's AI has now discovered a faster technique that...

3 weeks ago

DeepMind breaks 50-year math record using AI; new record falls a week later

Last week, DeepMind announced it discovered a more efficient way to perform matrix multiplication, conquering a 50-year-old record.

2 weeks ago
*Nature* announced in October, 2022 that:

<table>
<thead>
<tr>
<th>Theorem (FBHHRBNRSSSHK)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MultRank(4) ≤ 47* and MultRank(5) ≤ 96*</td>
</tr>
</tbody>
</table>
*Nature* announced in October, 2022 that:

**Theorem (FBHHRBNRSSHK)**

\[
\text{MultRank}(4) \leq 47^* \quad \text{and} \quad \text{MultRank}(5) \leq 96^*
\]

It made me feel better that to discover this results, this team used 64 state-of-the-art TPU cores, trained for 600,000 iterations: a non-academic battery of computational resources that cost somewhere between $10,000 and $100,000 to run.

* for 0,1-matrices.
Figure 6: RL for AlphaTensor
A few days later...

THE FBHHRBNRSSSHK-ALGORITHM FOR MULTIPLICATION IN $\mathbb{Z}_2^{5\times5}$ IS STILL NOT THE END OF THE STORY

MANUEL KAUERS* AND JAKOB MOOSBAUER†

ABSTRACT. In response to a recent *Nature* article which announced an algorithm for multiplying $5 \times 5$-matrices over $\mathbb{Z}_2$ with only 96 multiplications, two fewer than the previous record, we present an algorithm that does the job with only 95 multiplications.

1. INTRODUCTION

Ever since Strassen [8] discovered that $2 \times 2$-matrices can be multiplied with only 7 multiplications in the coefficient domain, there is a mystery around the complexity of matrix multiplication. For asymptotically large $n$, the best we know at the moment is a multiplication algorithm that requires $O(n^{2.3728596})$ operations [1], slightly improving upon the previous record $O(n^{2.3728639})$ [5]. For $n = 3$, it is known that 23 multiplications suffice in a non-commutative setting [4]. For $n = 4$, we can solve the problem with 49 multiplications by applying Strassen’s algorithm recursively. In a recent article that received considerable media attention, Fawzi et al. [2] used a machine learning approach to find a multiplication scheme with 47 multiplications, applicable to coefficient domains of characteristic 2. Under the same