An Isabelle Formalization of the Expressiveness of Deep Learning

Alexander Bentkamp  Vrije Universiteit Amsterdam
Jasmin Blanchette  Vrije Universiteit Amsterdam
Dietrich Klakow  Universität des Saarlandes
Motivation

- Case study of proof assistance in the field of machine learning
- Development of general-purpose libraries
- Study of the mathematics behind deep learning
Motivation

- Case study of proof assistance in the field of machine learning
- Development of general-purpose libraries
- Study of the mathematics behind deep learning

Just wanted to formalize something!
Fundamental Theorem of Network Capacity
(Cohen, Sharir & Shashua, 2015)

Shallow network

\[ x \]
\[ f(x) \]

needs exponentially more nodes to express the same function as

deep network

\[ x \]
\[ f(x) \]

for the vast majority of functions*

* except for a Lebesgue null set
Deep convolutional arithmetic circuit

- Input
- Representational Layer
  - Non-linear functions
  - 1x1 Convolution
  - Multiplication by a weight matrix
- Pooling
  - Componentwise Multiplication
- Output
Shallow convolutional arithmetic circuit

1x1 Convolution
Multiplication by a weight matrix

Pooling
Componentwise Multiplication

Output

Representational Layer
Non-linear functions

Input

Z

Y
**Lebesgue measure**

**definition** 
\[ \text{lborel} :: (\alpha :: \text{euclidean_space}) \text{ measure} \]

**Isabelle's standard probability library**

**vs.**

**definition** 
\[ \text{lborel}_f :: \text{nat} \Rightarrow (\text{nat} \Rightarrow \text{real}) \text{ measure} \]

**where** 
\[ \text{lborel}_f \ n = \prod_{b \in \{.. < n\}} (\text{lborel} :: \text{real measure}) \]

**My new definition**
Matrices

- Isabelle’s multivariate analysis library
- Sternagel & Thiemann’s matrix library
  (Archive of Formal Proofs, 2010)
- Thiemann & Yamada’s matrix library
  (Archive of Formal Proofs, 2015)

I added definitions and lemmas for

- matrix rank
- submatrices
Matrix dimension fixed by the type

- Isabelle’s multivariate analysis library
- Sternagel & Thiemann’s matrix library (Archive of Formal Proofs, 2010)
- Thiemann & Yamada’s matrix library (Archive of Formal Proofs, 2015)
Matrices

- Isabelle’s multivariate analysis library
- Sternagel & Thiemann’s matrix library
  (Archive of Formal Proofs, 2010) lacking many necessary lemmas
- Thiemann & Yamada’s matrix library
  (Archive of Formal Proofs, 2015)

matrix dimension fixed by the type
Matrices

- Isabelle’s multivariate analysis library
- Sternagel & Thiemann’s matrix library
  (Archive of Formal Proofs, 2010)
- Thiemann & Yamada’s matrix library
  (Archive of Formal Proofs, 2015)

I added definitions and lemmas for
- matrix rank
- submatrices
Multivariate polynomials

Lochbihler & Haftmann’s polynomial library

I added various definitions, lemmas, and the theorem

- "Zero sets of polynomials \( \neq 0 \) are Lebesgue null sets."

**Theorem:**

```plaintext
fixes p :: real mpoly
assumes p \( \neq 0 \) and vars p \( \subseteq \{.. < n\} \)
shows \( \{x \in \text{space (lborel}_f n). \text{insertion x p = 0}\} \in \text{null_sets (lborel}_f n) \)
```
My tensor library

typedef \( \alpha \) tensor = 
\{(ds :: nat list, as :: \( \alpha \) list). length as = prod_list ds\}

- addition, multiplication by scalars, tensor product, matricization, CP-rank

- Powerful induction principle uses subtensors:
  - Slices a \( d_1 \times d_2 \times \cdots \times d_N \) tensor into \( d_1 \) subtensors of dimension \( d_2 \times \cdots \times d_N \)

**definition** subtensor :: \( \alpha \) tensor \( \Rightarrow \) nat \( \Rightarrow \) \( \alpha \) tensor
The proof on one slide

**Def1** Define a tensor $\mathcal{A}(w)$ that describes the function expressed by the deep network with weights $w$

**Lem1** The CP-rank of $\mathcal{A}(w)$ indicates how many nodes the shallow network needs to express the same function

**Def2** Define a polynomial $p$ with the deep network weights $w$ as variables

**Lem2** If $p(w) \neq 0$, then $\mathcal{A}(w)$ has a high CP-rank

**Lem3** $p(w) \neq 0$ almost everywhere
# Restructuring the proof

## Before

<table>
<thead>
<tr>
<th>Def1</th>
<th>Tensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lem1</td>
<td>Tensors, shallow network</td>
</tr>
<tr>
<td></td>
<td>Induction over the deep network</td>
</tr>
<tr>
<td>Lem2</td>
<td>Polynomials, Matrices</td>
</tr>
<tr>
<td>Def2</td>
<td>Polynomials, Tensors</td>
</tr>
<tr>
<td>Lem3a</td>
<td>Matrices, Tensors</td>
</tr>
<tr>
<td>Lem3b</td>
<td>Measures, Polynomials</td>
</tr>
</tbody>
</table>

## After

<table>
<thead>
<tr>
<th>Def1</th>
<th>Tensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lem1</td>
<td>Tensors, shallow network</td>
</tr>
<tr>
<td></td>
<td>Induction over the deep network</td>
</tr>
<tr>
<td>Def2</td>
<td>Polynomials, Tensors</td>
</tr>
<tr>
<td>Lem2</td>
<td>Polynomials, Matrices</td>
</tr>
<tr>
<td></td>
<td>Induction over the deep network</td>
</tr>
<tr>
<td>Lem3a</td>
<td>Matrices, Tensors</td>
</tr>
<tr>
<td>Lem3b</td>
<td>Measures, Polynomials</td>
</tr>
</tbody>
</table>
Restructuring the proof

Before*

<table>
<thead>
<tr>
<th>Def1</th>
<th>Tensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lem1</td>
<td>Tensors, shallow network</td>
</tr>
<tr>
<td>Induction over the deep network</td>
<td></td>
</tr>
<tr>
<td>Lem2</td>
<td>Polynomials, Matrices</td>
</tr>
<tr>
<td>Def2</td>
<td>Polynomials, Tensors</td>
</tr>
<tr>
<td>Lem3a</td>
<td>Matrices, Tensors</td>
</tr>
<tr>
<td>Lem3b</td>
<td>Measures, Polynomials</td>
</tr>
</tbody>
</table>

* except for a Lebesgue null set

After*

<table>
<thead>
<tr>
<th>Def1</th>
<th>Tensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lem1</td>
<td>Tensors, shallow network</td>
</tr>
<tr>
<td>Induction over the deep network</td>
<td></td>
</tr>
<tr>
<td>Def2</td>
<td>Polynomials, Tensors</td>
</tr>
<tr>
<td>Lem2</td>
<td>Polynomials, Matrices</td>
</tr>
<tr>
<td>Induction over the deep network</td>
<td></td>
</tr>
<tr>
<td>Lem3a</td>
<td>Matrices, Tensors</td>
</tr>
<tr>
<td>Lem3b</td>
<td>Measures, Polynomials</td>
</tr>
</tbody>
</table>

* except for a zero set of a polynomial
Type for convolutional arithmetic circuits

datatype \( \alpha \text{ cac} = \)

\[
\begin{array}{ll}
\text{Input nat} & | \text{Conv } \alpha (\alpha \text{ cac}) & | \text{Pool } (\alpha \text{ cac}) (\alpha \text{ cac})
\end{array}
\]
Type for convolutional arithmetic circuits

```
datatype α cac =
  Input nat | Conv α (α cac) | Pool (α cac) (α cac)
```

```
fun insert_weights ::
  (nat × nat) cac ⇒ (nat ⇒ real) ⇒ real mat cac
```

network without weights weights network with weights
Type for convolutional arithmetic circuits

`datatype α cac =
  Input nat | Conv α (α cac) | Pool (α cac) (α cac)`

`fun insert_weights ::
  (nat × nat) cac ⇒ (nat ⇒ real) ⇒ real mat cac
  network without weights weights network with weights`

`fun evaluate_net :: real mat cac ⇒ real vec list ⇒ real vec
  network input output`
locale deep_net_params =
  fixes rs :: nat list
  assumes deep: length rs ≥ 3
  and no_zeros: ⋀ r. r ∈ set rs → 0 < r
Deep and shallow networks

**deep_net =**

```
Input
Conv \((r_2 \times r_3)\)
Pool
Conv \((r_1 \times r_2)\)
Pool
Conv \((r_0 \times r_1)\)
```

**shallow_net Z =**

```
Input
Conv \((Z \times r_3)\)
Pool
Conv \((Z \times r_3)\)
Pool
Conv \((r_0 \times Z)\)
```
**Def1** Define a tensor $\mathcal{A}(w)$ that describes the function expressed by the deep network with weights $w$

\[
\text{definition } \mathcal{A} :: (\text{nat} \Rightarrow \text{real}) \Rightarrow \text{real tensor}
\]
\[
\text{where } \mathcal{A} w = \text{tensor\_from\_net} \left( \text{insert\_weights deep\_net} \ w \right)
\]

The function $\text{tensor\_from\_net}$ represents networks by tensors:

\[
\text{fun } \text{tensor\_from\_net} :: \text{real mat cac} \Rightarrow \text{real tensor}
\]

If two networks express the same function, the representing tensors are the same.
\textbf{Lemma 1} The CP-rank of $A(w)$ indicates how many nodes the shallow network needs to express the same function.

\begin{itemize}
  \item \texttt{lemma cprank\_shallow\_model:}
    \item \texttt{shows cprank (tensor\_from\_net (insert\_weights w (shallow\_net Z))) \leq Z}
  \end{itemize}

- Can be proved by definition of the CP-rank.
Def2 Define a polynomial \( p \) with the deep network weights \( w \) as variables

Easy to define as a function:

**definition** \( p_{func} :: (\text{nat} \Rightarrow \text{real}) \Rightarrow \text{real} \) **where**

\[
p_{func} \ w = \det (\text{submatrix} [A_i \ w] \ \text{rows}_\{\text{with}_1\} \ \text{rows}_\{\text{with}_1\})
\]

But we must prove that \( p_{func} \) is a polynomial function
Lemma

If $p(w) \neq 0$, then $A(w)$ has a high CP-rank

- Assumes $p_{\text{func}}(w) \neq 0$
- Shows $r^{N_{\text{half}}} \leq \text{cprank}(A(w))$

Follows directly from definition of $p$ using properties of matricization and of matrix rank
Lem 3  \( p(w) \neq 0 \) almost everywhere

Zero sets of polynomials \( \neq 0 \) are Lebesgue null sets

\[ \Rightarrow \quad \text{It suffices to show that } p \neq 0 \]

We need a weight configuration \( w \) with \( p(w) \neq 0 \)
Final theorem

\[\forall_{ae} w_d \text{ w.r.t. lborel}_f \text{ weight_space_dim. } \not\exists w_s Z.\]
\[Z < r^{N_{\text{half}}} \wedge\]
\[\forall_{is. \text{ input_correct is } \rightarrow}\]
\[\text{evaluate_net (insert_weights deep_net } w_d) \text{ is } =\]
\[\text{evaluate_net (insert_weights (shallow_net } Z) w_s) \text{ is}\]
Conclusion

Outcome

- First formalization on deep learning
  Substantial development (≈ 7000 lines including developed libraries)

- Development of libraries
  New tensor library and extension of other libraries

- Generalization of the theorem
  Proof restructuring led to a more precise result

More information:

http://matryoshka.gforge.inria.fr/#Publications

AITP abstract

Archive of Formal Proofs entry

ITP paper draft (coming soon)

M.Sc. thesis