A Formal Proof of the Expressiveness of Deep Learning

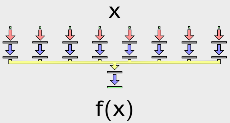
Alexander Bentkamp Vrije Universiteit Amsterdam
Jasmin Blanchette Vrije Universiteit Amsterdam
Dietrich Klakow Universität des Saarlandes

Motivation

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

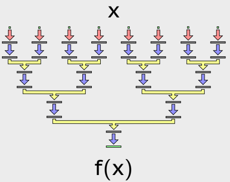
Fundamental Theorem of Network Capacity
(Cohen, Sharir & Shashua, 2015)

Shallow network



needs exponentially more nodes to express the same function as

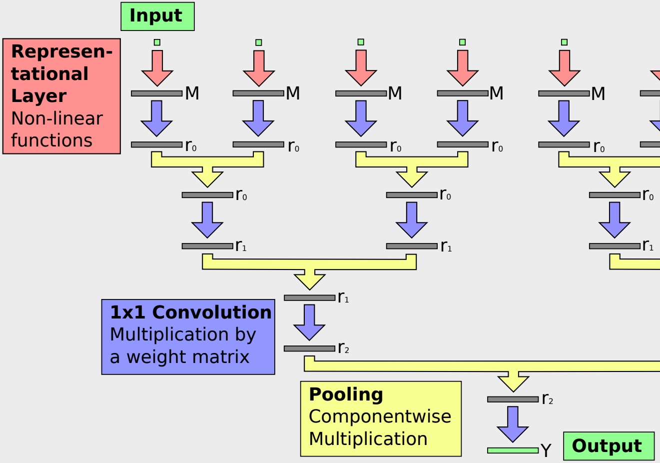
deep network



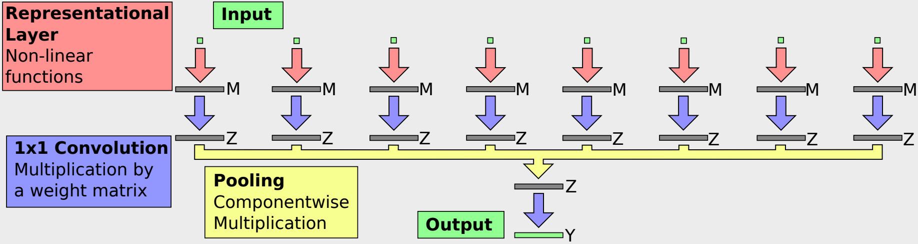
for the vast majority of functions*

* except for a Lebesgue null set of functions

Deep convolutional arithmetic circuit



Shallow convolutional arithmetic circuit



Convolutional arithmetic circuits $\neq$ CNNs

- CACs are not quite the standard architecture

But:

- Easier to analyze
- Allow to prove similar results for CNNs
- Perform better than CNNs when computational resources are limited

The proof on one slide

- **Def1** Define a tensor $\mathcal{A}(\mathrm{w})$ that describes the function expressed by the deep network with weights w
- **Lem1** The CP-rank of $\mathcal{A}(\mathrm{w})$ tells 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 $\mathrm{p}(\mathrm{w}) \neq 0$, then $\mathcal{A}(\mathrm{w})$ has a high CP-rank
- **Lem3** $\mathrm{p}(\mathrm{w}) \neq 0$ almost everywhere

Restructuring the proof

Before

Def1	Tensors
Lem1	Tensors, shallow network
Induction over the deep network	
Lem2	Polynomials, Matrices
Def2	Polynomials, Tensors
Lem3a	Matrices, Tensors
Lem3b	Measures, Polynomials

After

Def1	Tensors
Lem1	Tensors, shallow network
Induction over the deep network	
Def2	Polynomials, Tensors
Lem2	Polynomials, Matrices
Induction over the deep network	
Lem3a	Matrices, Tensors
Lem3b	Measures, Polynomials

Restructuring the proof

Before*

Def1	Tensors
Lem1	Tensors, shallow network

Induction over the deep network

Lem2	Polynomials, Matrices
Def2	Polynomials, Tensors
Lem3a	Matrices, Tensors
Lem3b	Measures, Polynomials

* except for a Lebesgue null set

After*

Def1	Tensors
Lem1	Tensors, shallow network

Induction over the deep network

Def2	Polynomials, Tensors

Lem2	Polynomials, Matrices

Induction over the deep network

Lem3a	Matrices, Tensors

Lem3b	Measures, Polynomials

* except for a zero set of a polynomial

Lebesgue measure

definition lborel :: (α :: euclidean_space) measure

Isabelle's standard probability library

Lebesgue measure

definition lborel :: (α :: euclidean_space) measure

Isabelle's standard probability library

Solution:

My new definition

definition $\mathsf{lborel_f}$ :: nat $\Rightarrow$ (nat $\Rightarrow$ real) measure **where**
$\mathsf{lborel_f}\ \mathsf{n} = \prod_{\mathsf{M}} \mathsf{b} \in \{..< \mathsf{n}\}.\ (\mathsf{lborel} :: \mathsf{real\ measure})$

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)

Matrices

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

matrix dimension fixed by the type

- 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)

Matrices

matrix dimension fixed by the type

- 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)

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 $\not\equiv 0$ are Lebesgue null sets."

theorem
  fixes p :: real mpoly
  assumes $\mathsf{p} \neq 0$ **and** vars $\mathsf{p} \subseteq \{..<\mathsf{n}\}$
  shows $\{\mathsf{x} \in \mathsf{space}\ (\mathsf{lborel}_\mathsf{f}\ \mathsf{n}).\ \mathsf{insertion}\ \mathsf{x}\ \mathsf{p} = 0\}$
  $\in \mathsf{null\_sets}\ (\mathsf{lborel}_\mathsf{f}\ \mathsf{n})$

My tensor library

typedef α tensor $=$
$\{(\mathsf{ds} :: \mathsf{nat\ list}, \mathsf{as} :: \alpha\ \mathsf{list}).\ |\mathsf{as}| = \prod \mathsf{ds}\}$

- 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 $d_2 \times \cdots \times d_N$

Type for convolutional arithmetic circuits

datatype α cac $=$

$\boxed{\text{Input nat}}$ | $\boxed{\text{Conv } \alpha \ (\alpha \text{ cac})}$ | $\boxed{\text{Pool } (\alpha \text{ cac}) \ (\alpha \text{ cac})}$

Type for convolutional arithmetic circuits

$$\textbf{datatype}\ \alpha\ \mathsf{cac} = \boxed{\mathsf{Input\ nat}}\ |\ \boxed{\mathsf{Conv}\ \alpha\ (\alpha\ \mathsf{cac})}\ |\ \boxed{\mathsf{Pool}\ (\alpha\ \mathsf{cac})\ (\alpha\ \mathsf{cac})}$$

$$\textbf{fun}\ \mathsf{insert\_weights} :: \underbrace{(\mathsf{nat} \times \mathsf{nat})\ \mathsf{cac}}_{\text{network without weights}} \Rightarrow \underbrace{(\mathsf{nat} \Rightarrow \mathsf{real})}_{\text{weights}} \Rightarrow \underbrace{\mathsf{real\ mat\ cac}}_{\text{network with weights}}$$

Type for convolutional arithmetic circuits

datatype α cac =
$\boxed{\text{Input nat}}$ | $\boxed{\text{Conv } \alpha \ (\alpha \text{ cac})}$ | $\boxed{\text{Pool } (\alpha \text{ cac}) \ (\alpha \text{ cac})}$

fun insert_weights ::
$\underbrace{(\text{nat} \times \text{nat}) \text{ cac}}_{\text{network without weights}} \Rightarrow \underbrace{(\text{nat} \Rightarrow \text{real})}_{\text{weights}} \Rightarrow \underbrace{\text{real mat cac}}_{\text{network with weights}}$

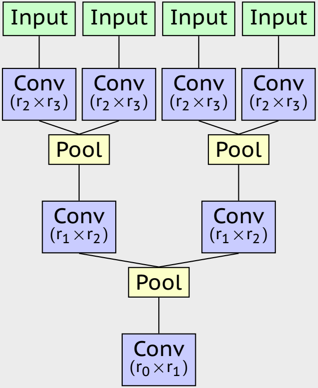
fun evaluate_net :: $\underbrace{\text{real mat cac}}_{\text{network}} \Rightarrow \underbrace{\text{real vec list}}_{\text{input}} \Rightarrow \underbrace{\text{real vec}}_{\text{output}}$

Deep network parameters

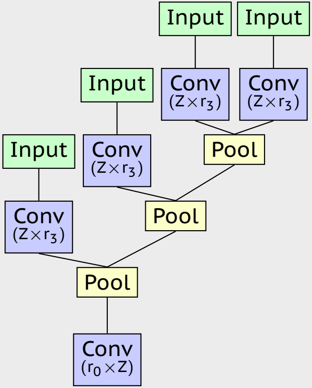
locale deep_net_params =
  fixes rs :: nat list
  assumes length rs $\geq$ 3
  and $\forall$r $\in$ set rs. 0 $<$ r

Deep and shallow networks

deep_net $=$



shallow_net Z $=$



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

definition $\mathcal{A}$ :: (nat $\Rightarrow$ real) $\Rightarrow$ real tensor **where**
$\mathcal{A}$ w = tensor_from_net (insert_weights deep_net w)

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

definition $\mathcal{A}$ :: (nat ⇒ real) ⇒ real tensor **where**
$\mathcal{A}$ w = tensor_from_net (insert_weights deep_net w)

The function tensor_from_net represents networks by tensors:

fun tensor_from_net :: real mat cac ⇒ real tensor

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

definition $\mathcal{A}$ :: (nat $\Rightarrow$ real) $\Rightarrow$ real tensor **where**
$\mathcal{A}$ w = tensor_from_net (insert_weights deep_net w)

The function tensor_from_net represents networks by tensors:

fun tensor_from_net :: real mat cac $\Rightarrow$ real tensor

If two networks express the same function, the representing tensors are the same

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

```
lemma
  cprank (tensor_from_net
                (insert_weights w (shallow_net Z))) ≤ Z
```

- by definition of CP-rank

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

Easy to define as a function:

definition $\mathsf{p}_{\mathsf{func}} :: (\mathsf{nat} \Rightarrow \mathsf{real}) \Rightarrow \mathsf{real}$ **where**

$\mathsf{p}_{\mathsf{func}}\ \mathsf{w} = \mathsf{det}\ (\mathsf{submatrix}\ [\mathcal{A}\ \mathsf{w}]\ \mathsf{rows\_with\_1}\ \mathsf{rows\_with\_1})$

But:

We must prove that $\mathsf{p}_{\mathsf{func}}$ corresponds to a polynomial

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

lemma
assumes $p_{func}\ w \neq 0$
shows $r^{N\_half} \leq \text{cprank}\ (\mathcal{A}\ w)$

- Follows directly from definition of p_{func} using properties of matricization and of matrix rank

Lem3 $p(w) \neq 0$ almost everywhere

Theorem:
Zero sets of polynomials $\not\equiv 0$ are Lebesgue null sets

Hence it suffices to show that $p \not\equiv 0$

So we need a weight configuration w with $p(w) \neq 0$

Final theorem

theorem

$\forall_{\mathrm{ae}}\, \mathsf{w_d}$ w.r.t. $\mathsf{lborel_f}$ weight_space_dim. $\nexists \mathsf{w_s}\ \mathsf{Z}$.

$\mathsf{Z} < \mathsf{r}^{\mathsf{N\_half}} \wedge$

$\forall$is. input_correct is $\longrightarrow$

evaluate_net (insert_weights deep_net $\mathsf{w_d}$) is $=$

evaluate_net (insert_weights (shallow_net Z) $\mathsf{w_s}$) is

Conclusion

Outcome

- First formalization on deep learning
  Substantial development ($\sim$ 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

Conclusion

Outcome

- First formalization on deep learning
  Substantial development ($\sim$ 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://www.cs.vu.nl/~abp290/

Archive of Formal Proofs entry

ITP paper

M.Sc. thesis