Machines of finite depth

Pietro Vertechi & Mattia G. Bergomi

pietro.vertechi@protonmail.com, mattiagbergomi@gmail.com

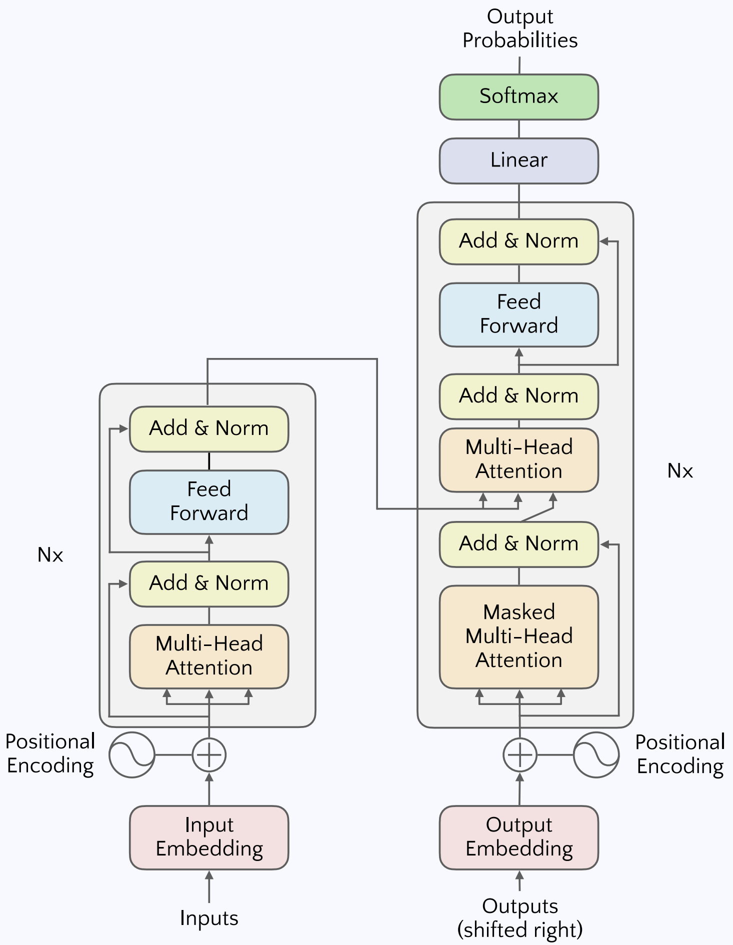
Table of contents

Background

- Modern deep learning relies on complex, hand-crafted architectures.
- Designing such architectures is a difficult, time-consuming problem.

Background

- Modern deep learning relies on complex, hand-crafted architectures.
- Designing such architectures is a difficult, time-consuming problem.

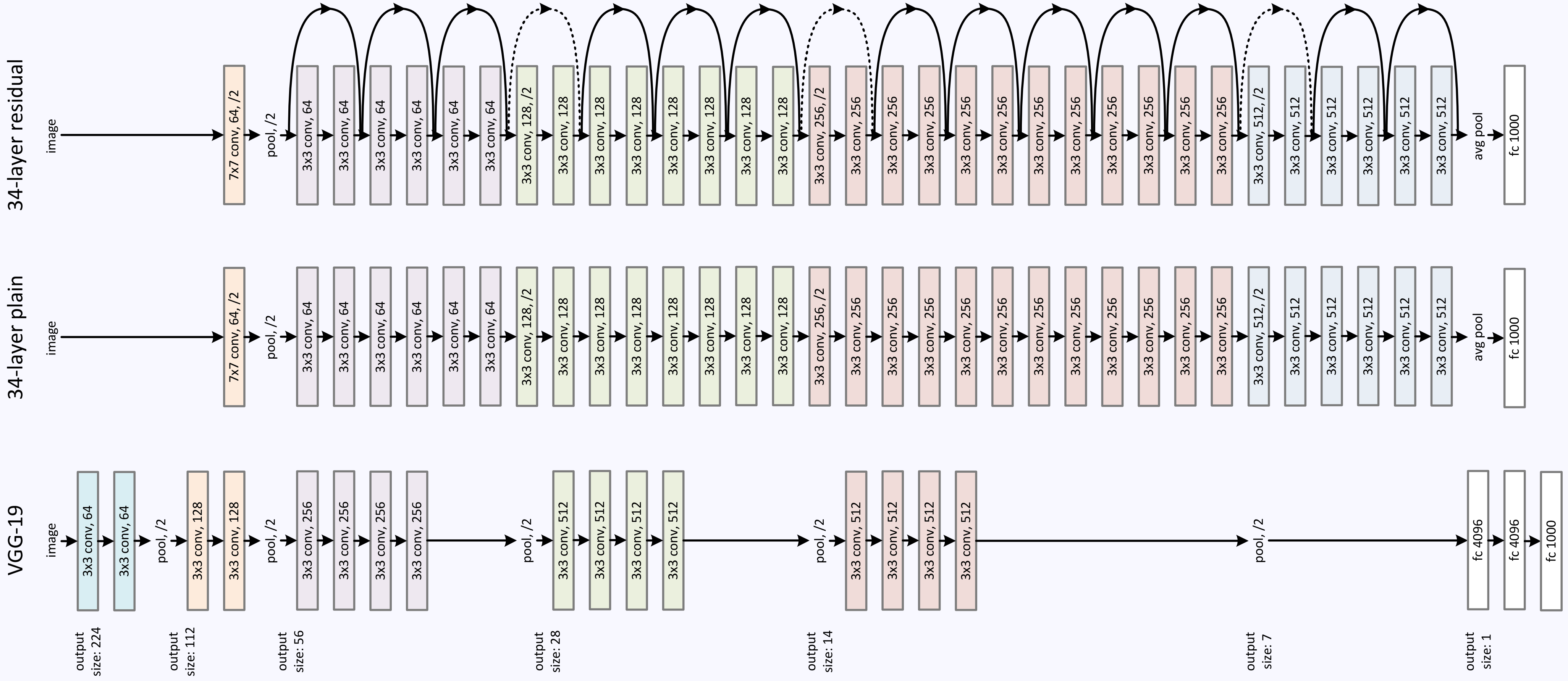


Vaswani et al., 2017

Background

- Modern deep learning relies on complex, hand-crafted architectures.
- Designing such architectures is a difficult, time-consuming problem.
- Complex data processing requires deep architectures.
- Architecture depth can cause pathologies (instability, vanishing gradients).

Background



He et al., 2015

Aim

- Formalize the notion of *neural network* and *neural architecture*.
- Define a *space of admissible architectures*.
- Determine the optimal architecture for a given problem.

Formalizing neural networks — composition

How are layers combined to form a deep neural network?

Intuitively, function composition is the natural operation.

$$X_0 \to X_1 \to \cdots \to X_n$$

Unfortunately, this does not include shortcut connections.

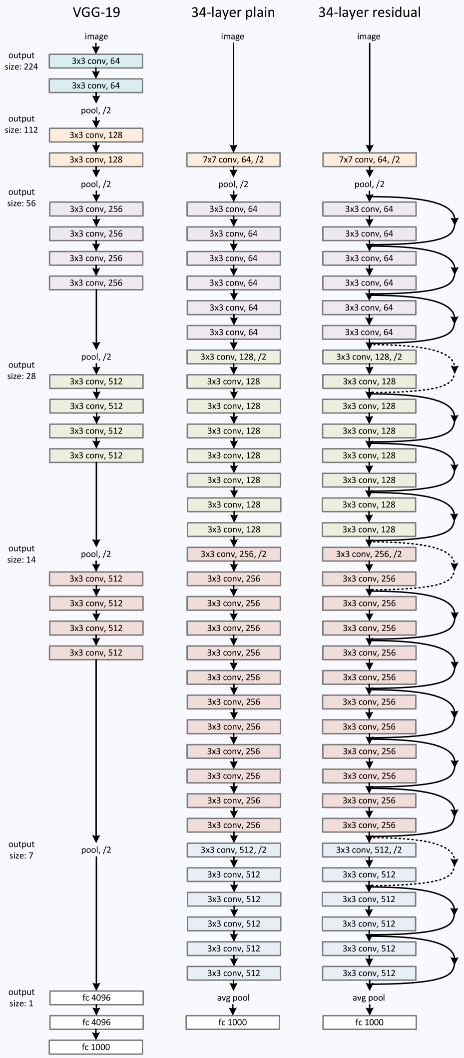
Formalizing neural networks — composition

How are layers combined to form a deep neural network?

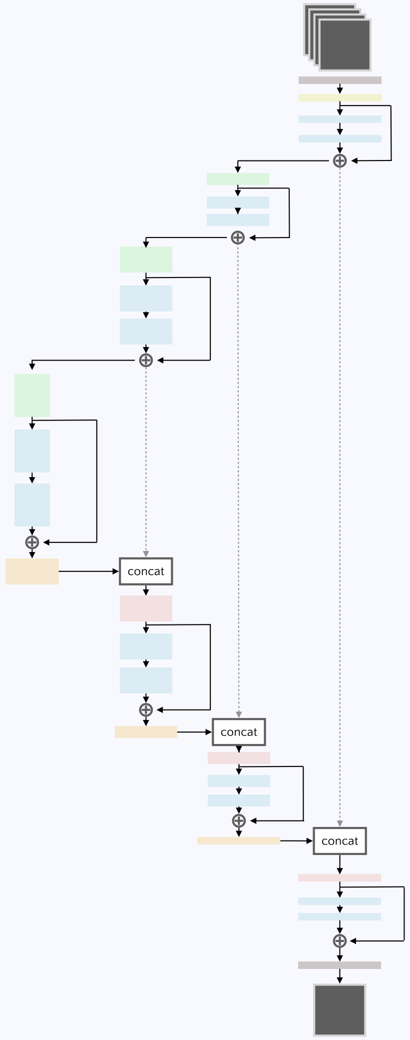
Intuitively, function composition is the natural operation.

$$X_0 \to X_1 \to \cdots \to X_n$$

Unfortunately, this does not include shortcut connections.



He et al., 2015



Noori et al., 2020

Formalizing neural networks — resolvent

Alternatively, given layers

$$X_0 \xrightarrow{l_1} X_1 \xrightarrow{l_2} \cdots \xrightarrow{l_n} X_n,$$

consider the global space

$$X = X_0 \oplus X_1 \oplus \cdots \oplus X_n$$

and the global network function

$$f = l_1 + \cdots + l_n \colon X \to X.$$

Formalizing neural networks — resolvent

Alternatively, given layers

$$X_0 \xrightarrow{l_1} X_1 \xrightarrow{l_2} \dots \xrightarrow{l_n} X_n,$$

consider the global space

$$X = X_0 \oplus X_1 \oplus \cdots \oplus X_n$$

and the global network function

$$f = l_1 + \cdots + l_n \colon X \to X.$$

A starting point

$$(x_0, 0, 0, \dots, 0) \in X_0 \oplus X_1 \oplus X_2 \oplus \cdots \oplus X_n$$

evolves as follows:

$$\begin{aligned}
&(x_0,\ 0,\ 0,\ \dots,\ 0)\\
&(x_0,\ l_1(x_0),\ 0,\ \dots,\ 0)\\
&(x_0,\ l_1(x_0),\ l_2(l_1(x_0)),\ \dots,\ 0)\\
&\vdots\\
&(x_0,\ l_1(x_0),\ l_2(l_1(x_0)),\ \dots,\ l_n(l_{n-1}\dots(l_1(x_0)))).
\end{aligned}$$

Formalizing neural networks — resolvent

Alternatively, given layers

$$X_0 \xrightarrow{l_1} X_1 \xrightarrow{l_2} \dots \xrightarrow{l_n} X_n,$$

consider the global space

$$X = X_0 \oplus X_1 \oplus \cdots \oplus X_n$$

and the global network function

$$f = l_1 + \cdots + l_n \colon X \to X.$$

A starting point

$$(x_0, 0, 0, \dots, 0) \in X_0 \oplus X_1 \oplus X_2 \oplus \cdots \oplus X_n$$

evolves as follows:

$$\begin{aligned}
&(x_0,\ 0,\ 0,\ \dots,\ 0)\\
&(x_0,\ l_1(x_0),\ 0,\ \dots,\ 0)\\
&(x_0,\ l_1(x_0),\ l_2(l_1(x_0)),\ \dots,\ 0)\\
&\vdots\\
&(x_0,\ l_1(x_0),\ l_2(l_1(x_0)),\ \dots,\ l_n(l_{n-1}\dots(l_1(x_0)))).
\end{aligned}$$

End point is solution of

$$x = f(x) + x_0.$$

Formalizing neural networks — resolvent

Intuition.

The output of a network f with input x_0 satisfies the *machine equation* $x = f(x) + x_0$.

Formalizing neural networks — resolvent

Intuition.

The output of a network f with input x_0 satisfies the *machine equation* $x = f(x) + x_0$.

Definition.

We say that f is a *machine* if, for any x_0, there is a unique solution to

$$x = f(x) + x_0,$$

and this unique solution depends smoothly on x_0.

Formalizing neural networks — resolvent

Intuition.

The output of a network f with input x_0 satisfies the *machine equation* $x = f(x) + x_0$.

Definition.

We say that f is a *machine* if, for any x_0, there is a unique solution to

$$x = f(x) + x_0,$$

and this unique solution depends smoothly on x_0.

In practice we are computing $x = (\mathrm{id} - f)^{-1}(x_0)$.

We call the operator $R_f = (\mathrm{id} - f)^{-1}$ the *resolvent* of f.

Formalizing neural networks — independence

Definition.

Let $f_1, f_2 \colon X \to X$ be endofunctions.

We say that f_1 *does not depend on* f_2 if, for all $a, b \in \mathbb{R}$,

$$f_1(af_2 + b) = f_1(b).$$

Formalizing neural networks — independence

Definition.

Let $f_1, f_2 \colon X \to X$ be endofunctions.

We say that f_1 *does not depend on* f_2 if, for all $a, b \in \mathbb{R}$,

$$f_1(af_2 + b) = f_1(b).$$

Proposition.

Let $f = f_1 + f_2$. If f_1 does not depend on f_2, then

$$R_f = R_{f_2} R_{f_1},$$

i.e., the resolvent of the sum is the composition of the resolvents.

Formalizing neural networks — independence

Definition.

Let $f_1, f_2 \colon X \to X$ be endofunctions.

We say that f_1 *does not depend on* f_2 if, for all $a, b \in \mathbb{R}$,

$$f_1(a f_2 + b) = f_1(b).$$

Proposition.

Let $f = f_1 + f_2$. If f_1 does not depend on f_2, then

$$R_f = R_{f_2} R_{f_1},$$

i.e., the resolvent of the sum is the composition of the resolvents.

Intuition.

Before computing one layer, one must compute its “dependencies”.

Complex architectures via hypergraphs

Hypergraphs allow for edges to connect to arbitrarily large collections of vertices.

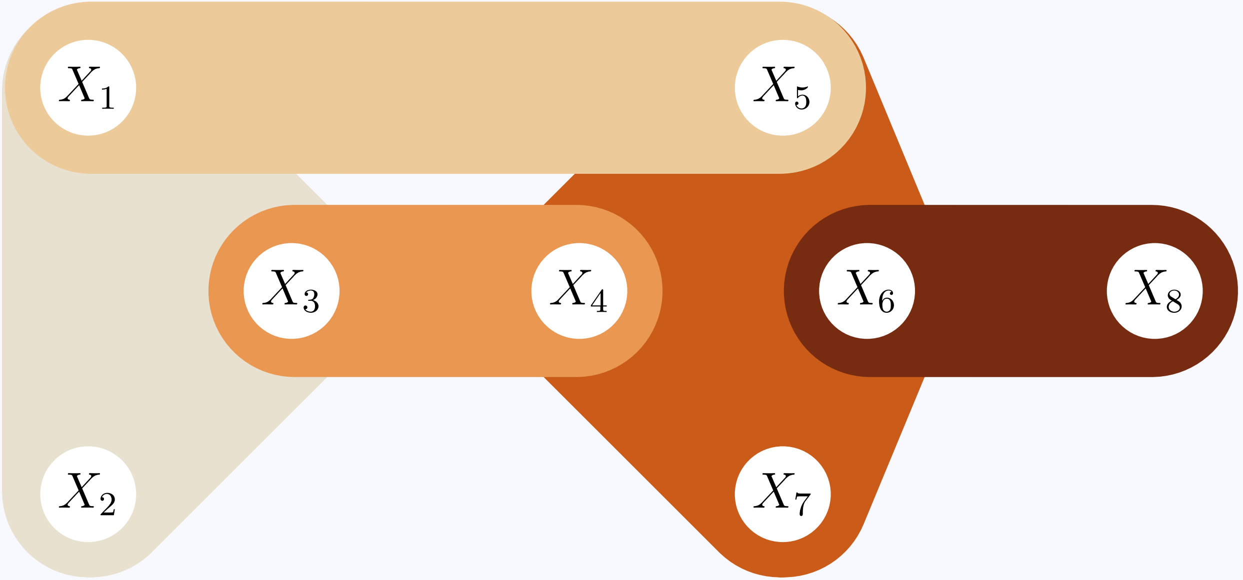
Thus, shortcut connections can be very complex.

Complex architectures via hypergraphs

Hypergraphs allow for edges to connect to arbitrarily large collections of vertices.

Thus, shortcut connections can be very complex.

- $f_1 : X_1 \times X_2 \to X_3$
- $f_2 : X_1 \to X_5$
- $f_3 : X_3 \to X_4$
- $f_4 : X_4 \to X_5 \times X_6 \times X_7$
- $f_5 : X_6 \to X_8$



Machines with *all* shortcuts

In practice, we will often choose to work with all shortcuts.

Recipe.

- Start with a layer (dense or convolutional) whose input and output share a common index space I.
- Partition the index space $I = I_1 \sqcup I_2 \sqcup \cdots \sqcup I_n$.
- Set to 0 weights connecting I_i to I_j with $i \geq j$.

Machines with *all* shortcuts

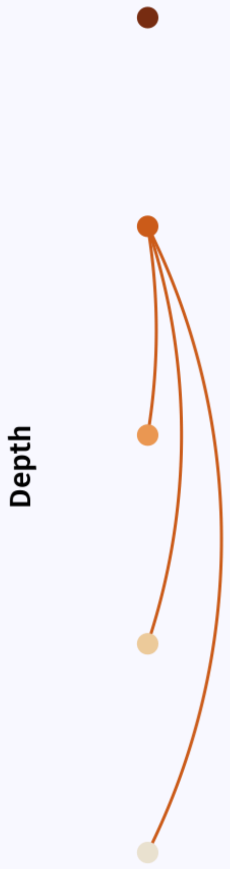
In practice, we will often choose to work with all shortcuts.

Recipe.

- Start with a layer (dense or convolutional) whose input and output share a common index space I.
- Partition the index space $I = I_1 \sqcup I_2 \sqcup \cdots \sqcup I_n$.
- Set to 0 weights connecting I_i to I_j with $i \geq j$.

Examples.

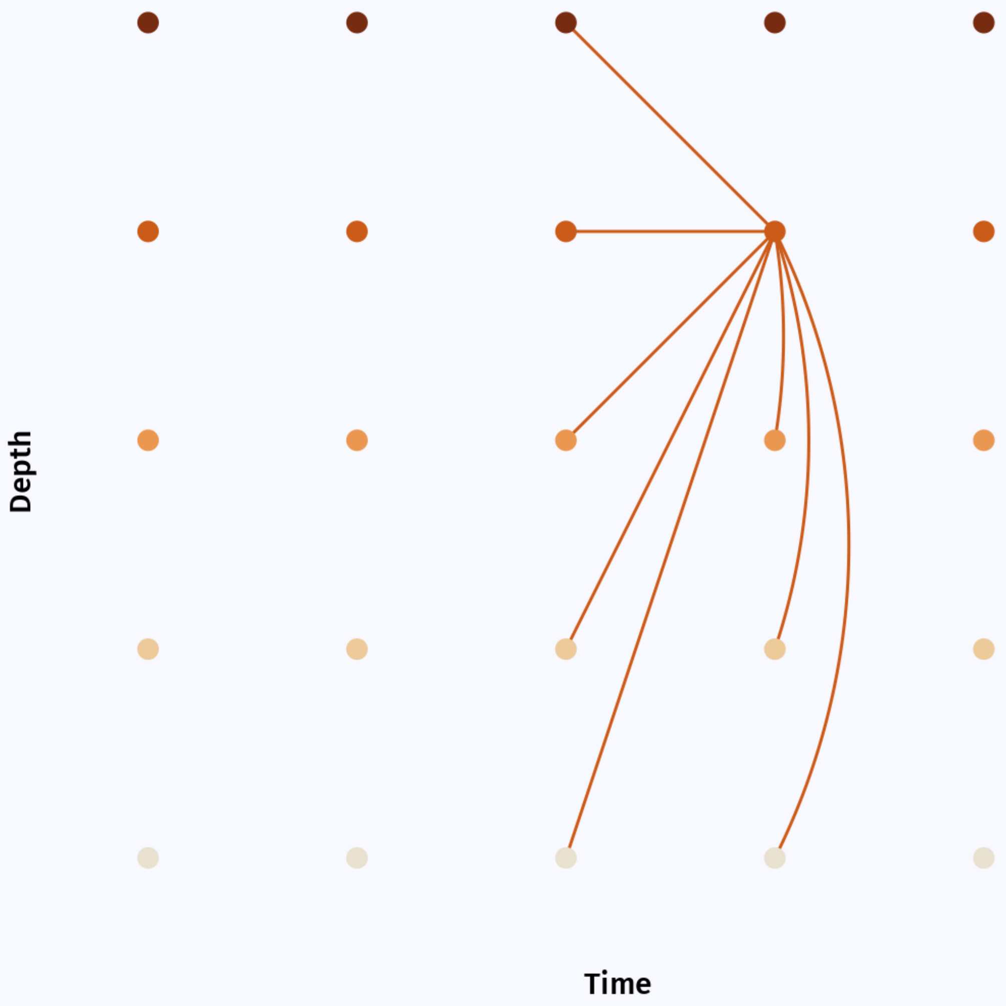
- Dense machine.
- Convolutional machine.
- Time machine (recurrent/convolution hybrid).



Feedforward architecture

Feedforward machine with all shortcuts.

Each node receives inputs from all nodes of smaller depth.



Recurrent / convolution hybrid

Unrolled representation.

Each node receives inputs from

- nodes of smaller depth from same timestep,
- all nodes from previous timestep.

Optimization

Definition.

A parametric machine is a smooth parametric function $f(p, x)$ such that

- $f(p, -)$ is a machine for all choice of parameters p,
- the resolvent $R_f(p, x_0)$ is jointly smooth in p and x_0.

Optimization

Definition.

A parametric machine is a smooth parametric function $f(p, x)$ such that

- $f(p, -)$ is a machine for all choice of parameters p,
- the resolvent $R_f(p, x_0)$ is jointly smooth in p and x_0.

As in classical deep learning, after computing the output

$$x = f(p, x) + x_0,$$

we evaluate the loss function

$$\mathcal{L}(x).$$

Then, we backpropagate the error through the machine and obtain

$$\frac{\partial \mathcal{L}}{\partial p}.$$

Optimization

Proposition.

The backward pass of a machine is again a machine.

Optimization

Proposition.

The backward pass of a machine is again a machine.

Intuition.

Let us consider the case where f is linear.

The backward pass is simply the adjoint.

The resolvent of the adjoint is the adjoint of the resolvent:

$$(\mathrm{id} - f^*)^{-1} = ((\mathrm{id} - f)^{-1})^*.$$

Optimization

Proposition.

The backward pass of a machine is again a machine.

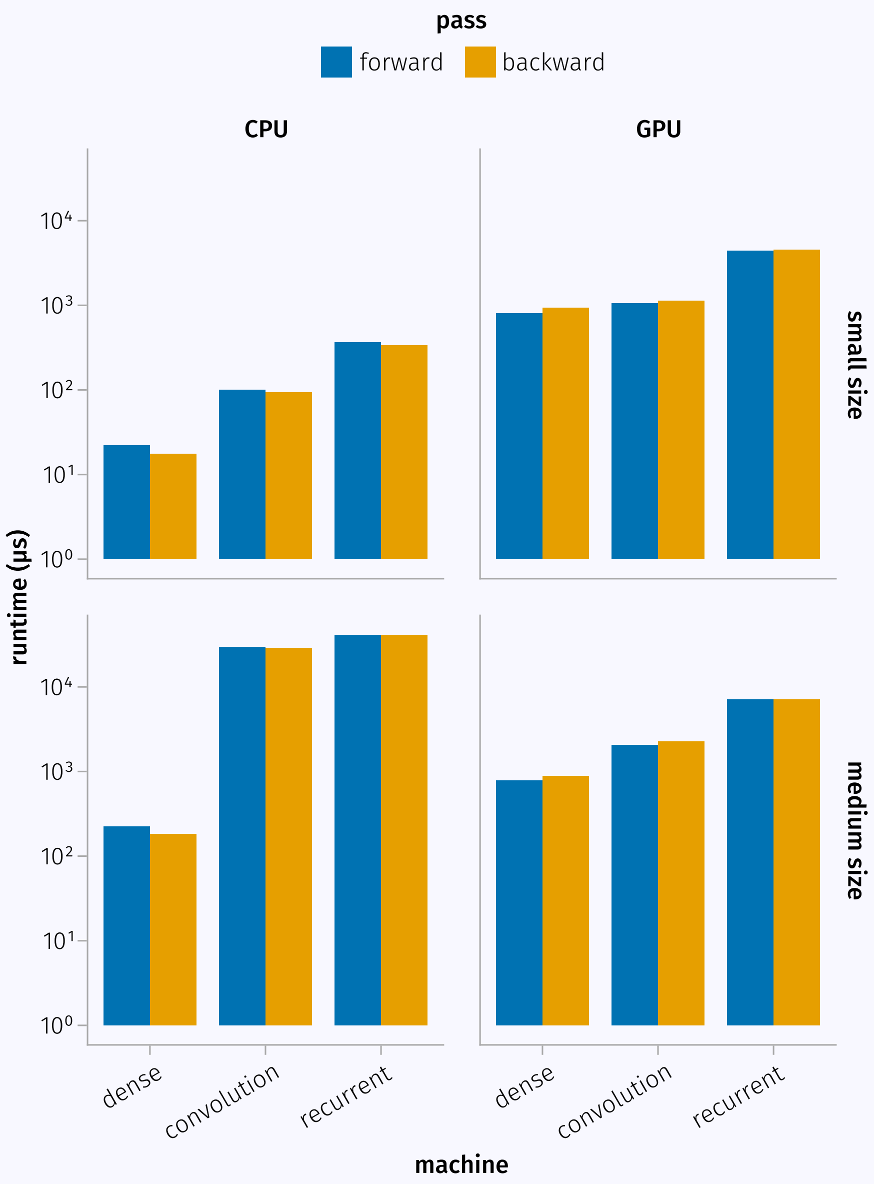
Intuition.

Let us consider the case where f is linear.

The backward pass is simply the adjoint.

The resolvent of the adjoint is the adjoint of the resolvent:

$$(\mathrm{id} - f^*)^{-1} = ((\mathrm{id} - f)^{-1})^*.$$



Conclusions

- Hand-crafted architectures require highly trained experts and time-consuming fine-tuning.
- We created a formal environment in which complex architectures and, in general, machines can be described.
- The resolvent generalizes the computation of a neural network.
- Complex machines can be built from smaller ones.
- In theory and in the examples we consider, the backward pass is analogous to the forward pass in terms of
  - structure,
  - computational cost.