
THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

Categorical Deep Learning

An Algebraic Theory of Architectures

Bruno Gavranović
Symbolica

SYCO 12

16.04.2024.

Joint work with

Paul Lessard

–

Symbolica

Andrew Dudzik

–

Google DeepMind

Tamara Von Glehn

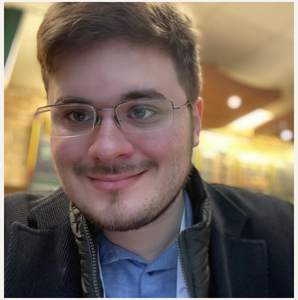
–

Google DeepMind

João Guilherme Araújo

–

Google DeepMind



Petar Veličković

–

Google DeepMind / University of Cambridge

What is this paper about?

Categorical Deep Learning: An Algebraic Theory of Architectures

Bruno Gavranović [*1 2] **Paul Lessard** [*1] **Andrew Dudzik** [*3]
Tamara von Glehn [3] **João G.M. Araújo** [3] **Petar Veličković** [3 4]

Abstract

We present our position on the elusive quest for a general-purpose framework for specifying and studying deep learning architectures. Our opinion is that the key attempts made so far lack a coherent bridge between specifying *constraints* which models must satisfy and specifying their *implementations*. Focusing on building a such a bridge, we propose to apply category theory—precisely, the universal *algebra of monads* valued in a 2-category of *parametric maps*—as a single theory elegantly subsuming both of these flavours of neural network design. To defend our position, we show how this theory recovers constraints induced by geometric deep learning, as well as implementations of many architectures drawn from the diverse landscape of neural networks, such as RNNs. We also illustrate how the theory naturally encodes many standard con-

neural networks can be specified in a *top-down* manner, wherein models are described by the *constraints* they should satisfy (e.g. in order to respect the structure of the data they process). Alternatively, a *bottom-up* approach describes models by their *implementation*, i.e. the sequence of tensor operations required to perform their forward/backward pass.

1.1. Our opinion

It is our **opinion** that ample effort has already been given to both the top-down and bottom-up approaches *in isolation*, and that there hasn't been sufficiently expressive theory to address them both *simultaneously*. *If we want a **general** guiding framework for **all** of deep learning, this needs to change.* To substantiate our opinion, we survey a few ongoing efforts on both sides of the divide.

One of the most successful examples of the top-down framework is *geometric deep learning* (Bronstein et al.,

Summary

- Theory covering numerous deep learning architectures
- A natural step forward from *Geometric Deep Learning*
- Less about CT, more about application
- Exciting things ahead

Plan for today

I. What is Deep Learning?
II. Geometric Deep Learning
III. Categorical Deep Learning
IV. Next steps

I.

What is Deep Learning?

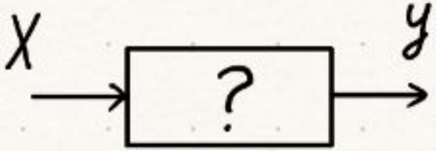
What is Deep Learning?

- Science and engineering of **finding structure** in unstructured data

- Iterative function optimisation from a input-output samples

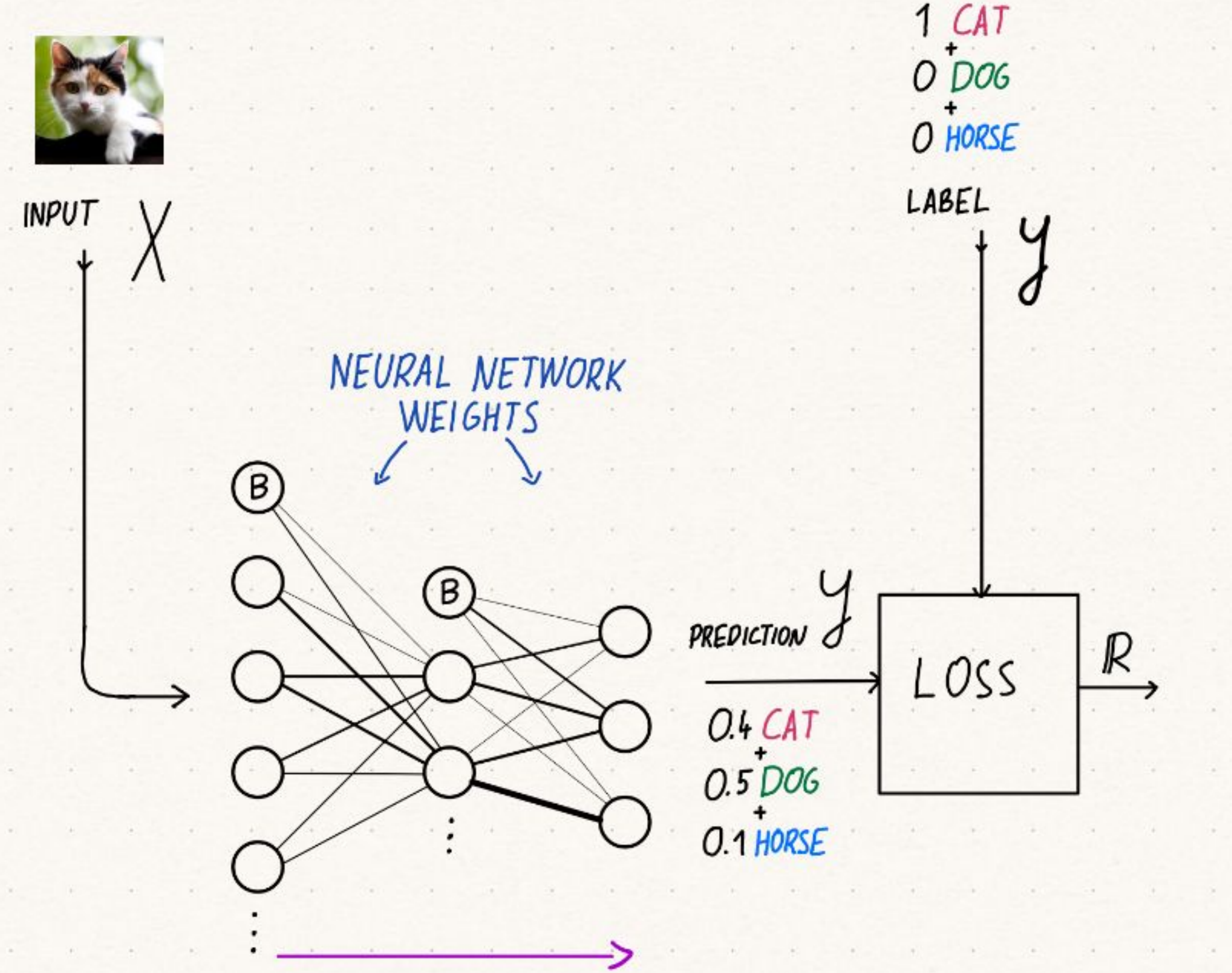
- Requirement: differentiability

SUPERVISED LEARNING WITH NEURAL NETWORKS
IN ONE SLIDE:



TASK: FIND A FUNCTION $X \longrightarrow y$ THAT BEST FITS

A DATASET: List $X \times y$


1 CAT
+
0 DOG
+
0 HORSE
INPUT
X
LABEL
y
NEURAL NETWORK
WEIGHTS
B
B
PREDICTION
y
LOSS
R
0.4 CAT
+
0.5 DOG
+
0.1 HORSE

What's been done so far?

1

Backprop as Functor: A compositional perspective on supervised learning

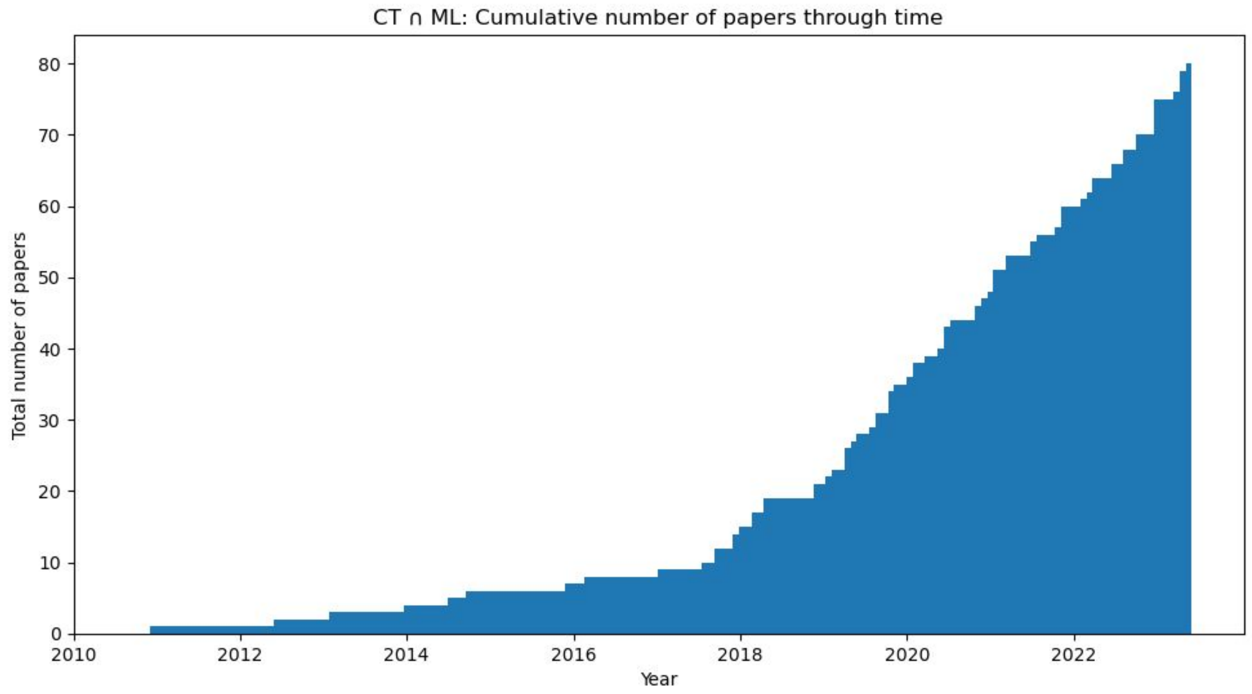
Brendan Fong

Department of Ma
Massachusetts Institute

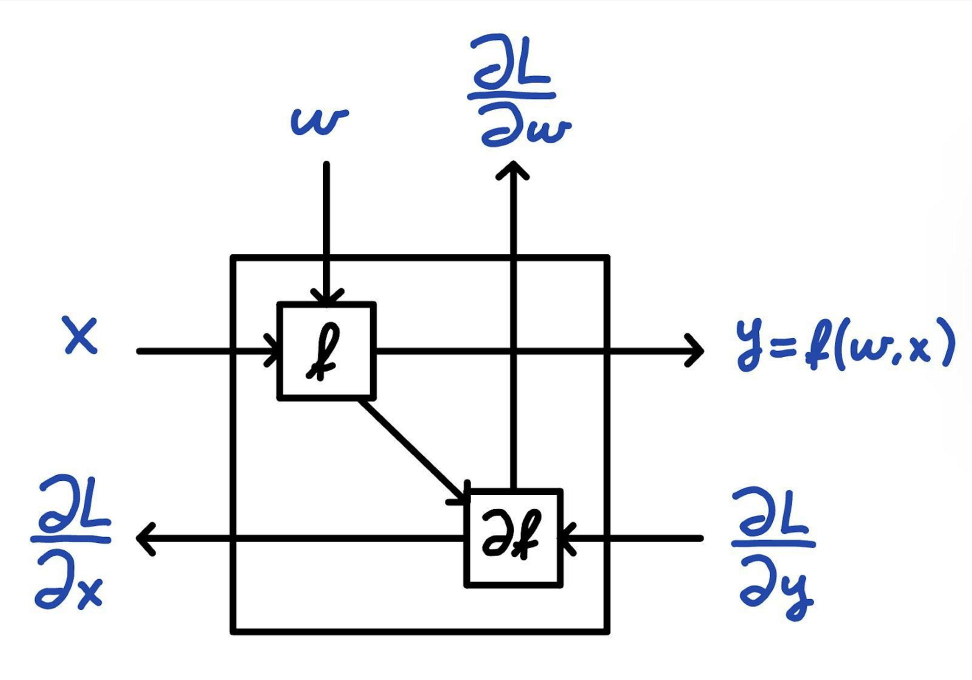
Categorical Foundations of Gradient-Based Learning

G.S.H. CRUTTWELL, Mount Allison University, Canada
BRUNO GAVRANOVIĆ and NEIL GHANI, University of Strathclyde, UK
PAUL WILSON and FABIO ZANASI, University College London, UK

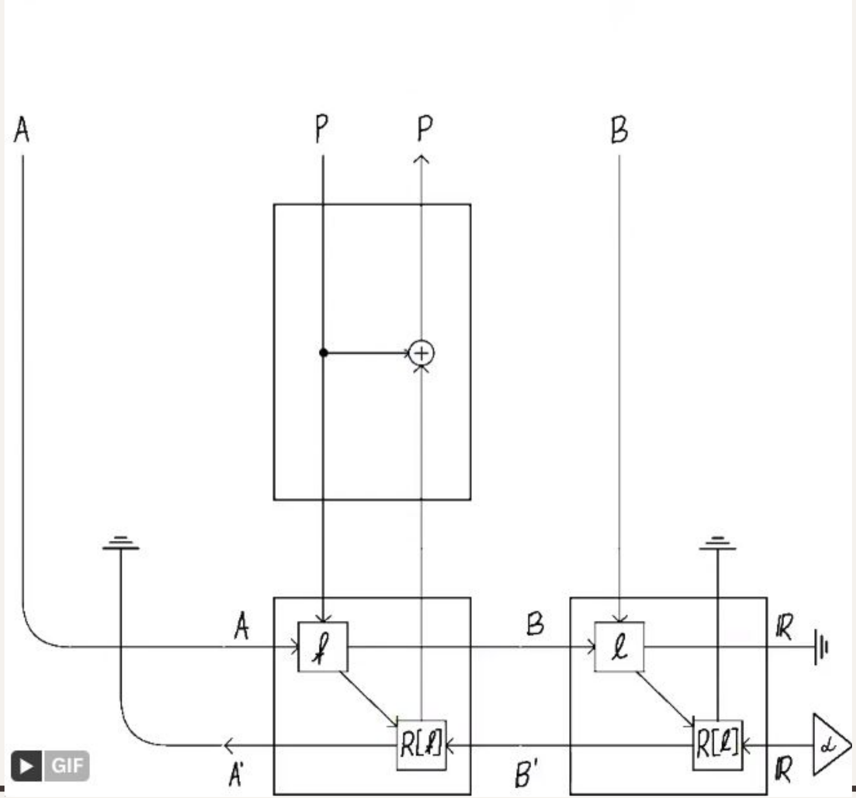
We propose a categorical semantics of gradient-based machine learning algorithms in terms of lenses, parametrised maps, and reverse derivative categories. This foundation provides a powerful explanatory and unifying framework: it encompasses a variety of gradient descent algorithms such as ADAM, AdaGrad, and Nesterov momentum, as well as a variety of loss functions such as as MSE and Softmax cross-entropy, shedding new light on their similarities and differences. Our approach to gradient-based learning has examples generalising beyond the familiar continuous domains (modelled in categories of smooth maps) and can be realized in the discrete setting of boolean circuits. Finally, we demonstrate the practical significance of our framework with an implementation in Python.


CT ∩ ML: Cumulative number of papers through time
Total number of papers
0
10
20
30
40
50
60
70
80
2010
2012
2014
2016
2018
2020
2022
Year

Neural networks as parametric lenses…



… and supervised learning as its composite.



Building a Neural Network from First Principles using Free Categories and Para(Optic)

Apr 15, 2024 • Zanzi Mihejevs • machine learning, categorical cybernetics, functional programming

Introduction

Category theory for machine learning has been a big topic recently, both with Bruno's thesis dropping, and the paper on using the Para construction for deep learning.

In this post we will look at how dependent types can allow us to almost effortlessly implement the category theory directly, opening up a path to new generalisations.

I will be making heavy use of Tatsuya Hirose's code that implements the Para(Optic) construction in Haskell. Our goal here is to show that when we make the category theory in the code explicit, it becomes a powerful scaffolding that lets us structure our program.

All in all, our goal is to formulate this: A simple neural network with static types enforcing the parameters and input and output dimensions.

```
import Data.Fin
import Data.Vect
```

…where to now?

- Deep Learning is not just about backprop

- Key problem in deep learning:

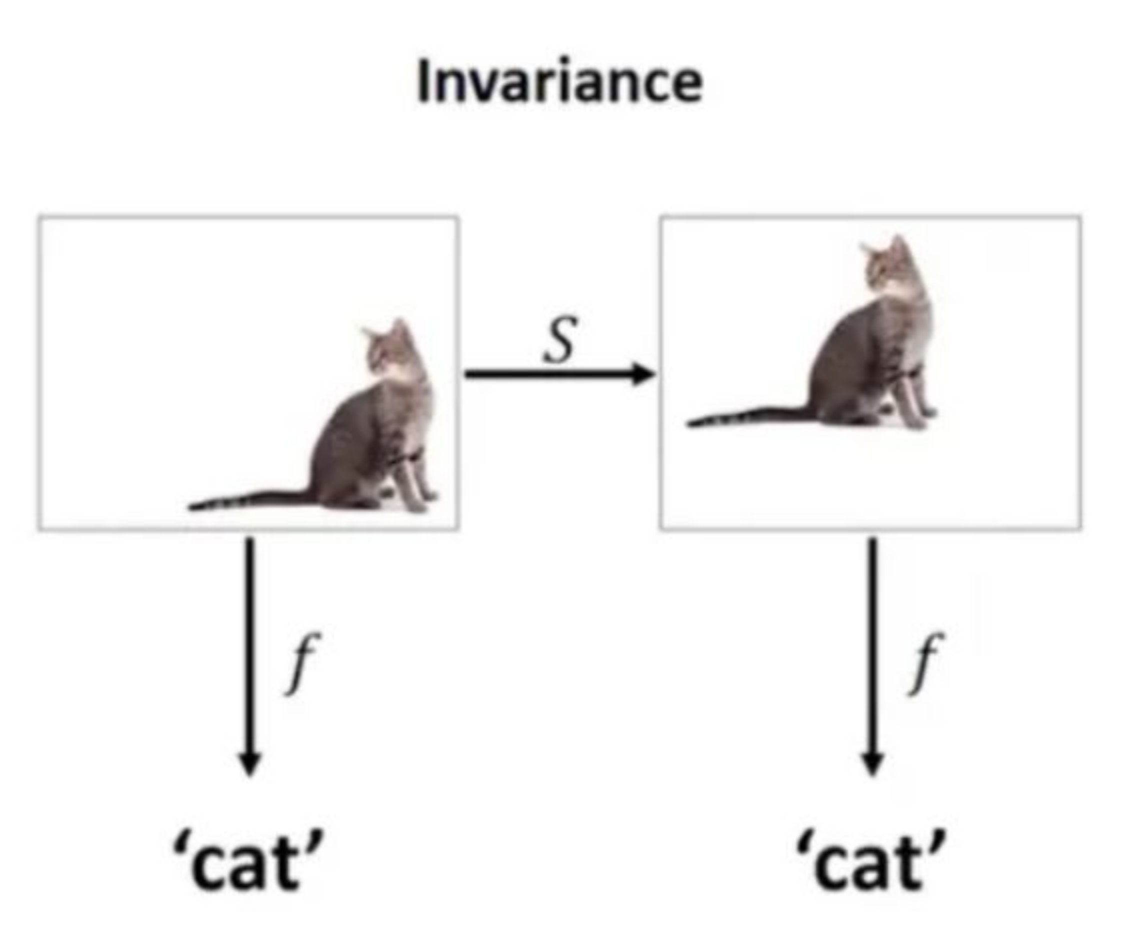
- Designing the **architecture of a neural network**: structure of its forward pass

What's wrong with the above picture?

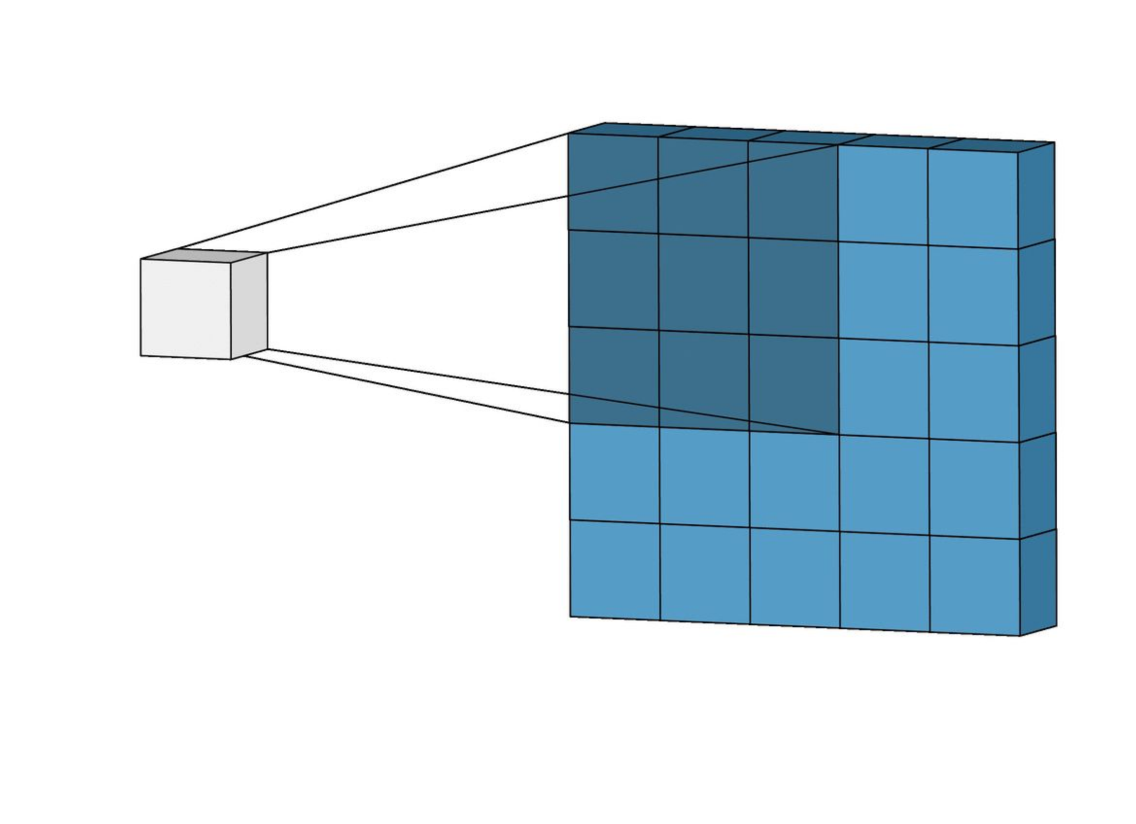
- Image data contains spatial information (Width/height/color encoding)
- By squashing it into a list, *we erase information useful for learning*
- Structure is useful for learning!

‘cat’

'cat'



Images taken from these notes.



Structure enables learning

- We encode priors into architecture, specifying how weights are reused in multiple places: *weight tying*
- Weight tying reduces parameter count
- Enables learning, as a feature has to be only learned once

Network architectures can today be…

Recurrent

Topological

Autoregressive

Convolutional

Graph

Autoencoding

Recursive

Generative-Adversarial

Networks today can be…

Huge.

GPT2

345 million

GPT3

175 billion

GPT4

1.73 trillion

GPT4

That's >700GB of storage just for weights.

GPT4

It cost $63M dollars to train.

GPT4

Completely inscrutable.

How can we understand them?

- Numerous issues:

- Explainability
- Fairness
- Regulation
- Hallucinations

- There’s large scale deployment of these models in practice

The goal?

- Providing a language that can help us understand existing, and design new architectures
- What kind of logics/theories is neural network using in coming to its conclusions?
- Can we use any kind of structural theory for this?

II.

Geometric Deep Learning

Geometric Deep Learning

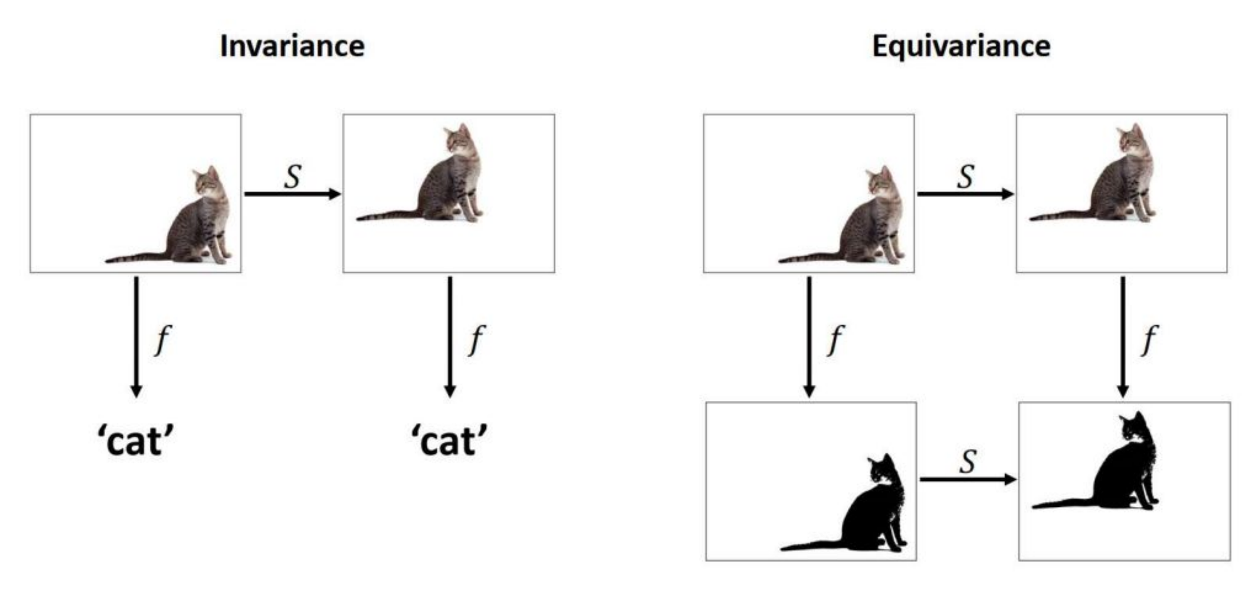
Geometric Deep Learning
Grids, Groups, Graphs,
Geodesics, and Gauges

Michael M. Bronstein[1], Joan Bruna[2], Taco Cohen[3], Petar Veličković[4]

May 4, 2021

Geometric Deep Learning

- Studies stability of inputs under transformation



Architectures as equivariant maps

- Transformations we can do to an input datapoint = group actions

- Maps which preserve these actions = group action homomorphisms

A bit more formally

- Let G be a group, and consider a G-action on X, and a G-action on Y

- A map f:X->Y is G-equivariant if for all g:G and x:X it holds that

$$f(g \blacktriangleright x) = g \triangleright f(x)$$

Group of…

- Translation
- Rotation
- Scaling
- Reflection
- Permutation
- …

How do we use this in neural networks?

- By instantiating the diagram in the appropriate category, such as FVect, we can represent maps as matrices.

- Then the equation $f(g \blacktriangleright x) = g \triangleright f(x)$ gives rise to an equation between matrices, specifying a *weight tying* scheme.

GDL covers a wide array of architectures

- Group convolutional neural networks
- Graph neural networks
- Topological neural networks
- …

- Generally, GDL has been a successful story

But…

Major flaw:

only covers equivariance with respect to *invertible* operations

GDL cannot handle

- Recursion
- Transition functions of an automaton/dynamical system
- Aggregation step of a dynamic programming algorithm
- Programs which write to or read from external memory

- How do we bridge this formalism with data types, general algorithms, branching, and other concepts from CS?

More generally…

- How do we specify the kinds of reasoning networks use?

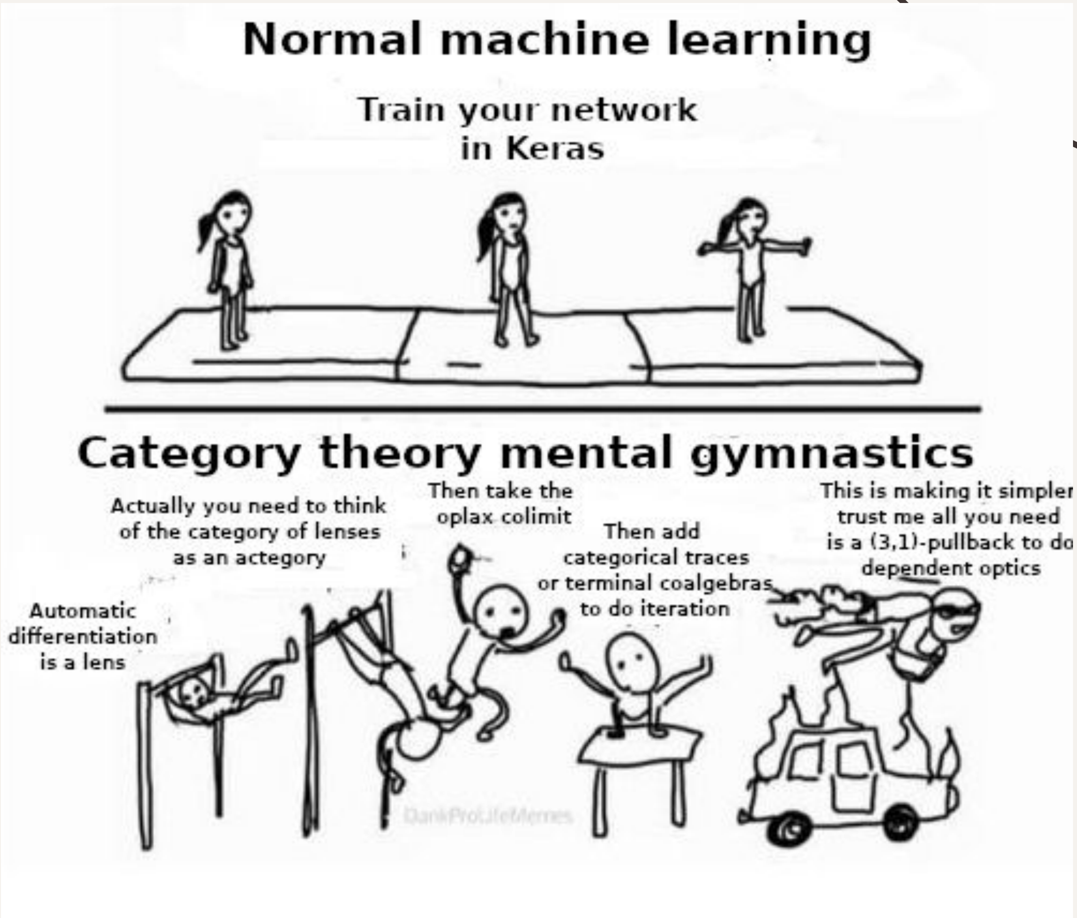
- We want networks to
  - Reason algorithmically
  - Form plans, and then execute them
  - Use (co)inductive reasoning

- … and do so in verifiable, and explainable ways.

III.

Categorical Deep Learning

In other words…



Group actions	→	Algebras for the group action monad
Equivariant maps	→	*Algebra homomorphisms* for the group action monad

In other words,

$$
\begin{tikzcd}
G \times X \arrow[d, "\blacktriangleright"'] & & G \times Y \arrow[d, "\triangleright"] \\
X & & Y
\end{tikzcd}
$$

In other words,

$$
\begin{tikzcd}
G \times X \arrow[r, "G \times f"] \arrow[d, "\blacktriangleright"'] & G \times Y \arrow[d, "\triangleright"] \\
X \arrow[r, "f"'] & Y
\end{tikzcd}
$$

In many ways, a trivial step

- But one which completely captures the idea of equivariance

- … and allows us to generalise in the right way:
  - **Get rid of invertibility:** G only needs to be a monoid for Gx- to be a monad
  - **Capture more structured operations:** We didn't need a *monad*.

Endofunctors work too!

- Consider the endofunctor $1 + A \times - : Set \to Set$
  - The set List(A), together with the map $1 + A \times List(A) \to List(A)$ is its algebra

- Consider the endofunctor $A + (-)^2 : Set \to Set$
  - The set BTree(A) of binary trees with A-labelled nodes, together with the map $A + BTree(A)^2 \to BTree(A)$ is its algebra

- ...

Folds as algebra homomorphisms - Lists

$$\begin{tikzcd}
1 + A \times \mathsf{List}(A) \arrow[r, "1+A\times f_r"] \arrow[d, "{[\mathsf{Nil},\mathsf{Cons}]}"'] & 1 + A \times X \arrow[d, "{[r_0,r_1]}"] \\
\mathsf{List}(A) \arrow[r, "f_r"'] & X
\end{tikzcd}$$

Here f_r is implemented by recursion on input, structural in nature:

$$f_r(\mathsf{Nil}) = r_0(\bullet)$$
$$f_r(\mathsf{Cons}(h, t)) = r_1(h, f_r(t))$$

Folds as algebra homomorphisms - Trees

$$\begin{tikzcd}
A + \mathsf{Tree}(A)^2 \arrow[r, "A+f_r^2"] \arrow[d, "{[\mathsf{Leaf},\mathsf{Node}]}"'] & A + X^2 \arrow[d, "{[r_0,r_1]}"] \\
\mathsf{Tree}(A)^2 \arrow[r, "f_r"'] & X
\end{tikzcd}$$

Here f_r is implemented by recursion on input, structural in nature:

$$f_r(\mathsf{Leaf}(a)) = r_0(a)$$
$$f_r(\mathsf{Node}(l,r)) = r_1(f_r(l), f_r(r))$$

Coalgebras!

- We can dualise the entire story

- Take the endofunctor Ax-:Set -> Set
  - Stream(A) is its coalgebra, together with the map Stream(A)->AxStream(A)

- Take the endofunctor [I, O x -]: Set -> Set
  - The set of Mealy machines with inputs I and outputs O is its coalgebra

NN cells as (co)algebras of 2-endofunctors!

Folding recurrent neural network

$$1 + A \times S \xrightarrow{(P, \mathsf{cell}^{\mathsf{rcnt}})} S$$

Unolding recurrent neural network

$$S \xrightarrow{(P, \langle \mathsf{cell}_o, \mathsf{cell}_n \rangle)} O \times S$$

Recursive neural network

$$A + S^2 \xrightarrow{(P, \mathsf{cell}^{\mathsf{rcsv}})} S$$

Full recurrent neural network

$$S \xrightarrow{(P, \mathsf{cell}^{\mathsf{Mealy}})} (I \to O \times S)$$

"Moore machine" neural network

$$S \xrightarrow{(P, \mathsf{cell}^{\mathsf{Moore}})} O \times (I \to S)$$

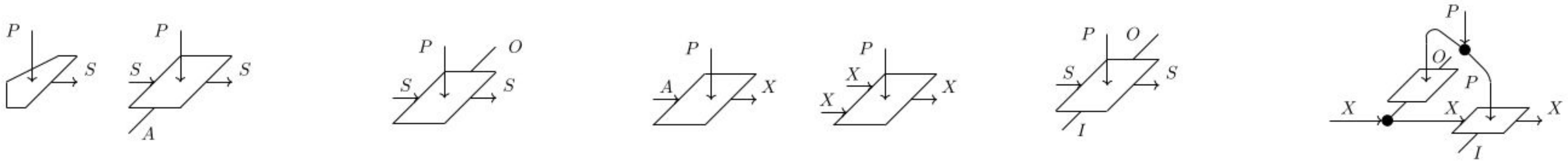
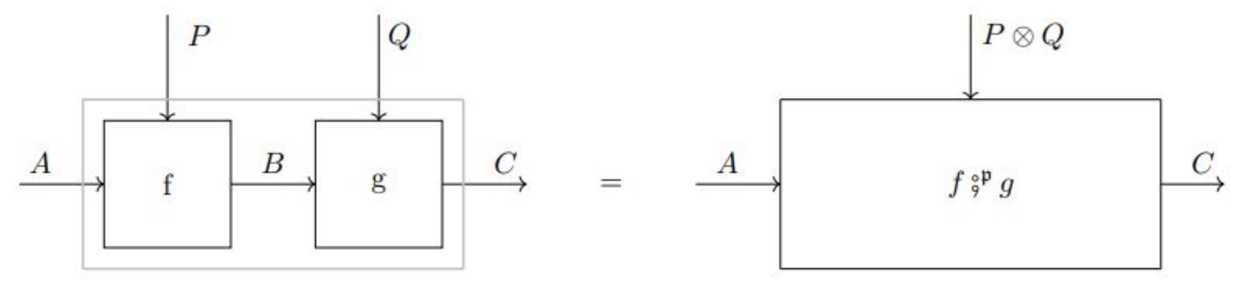


Figure 1: Parametric (co)algebras provide a high-level framework for describing structured computation in neural networks.

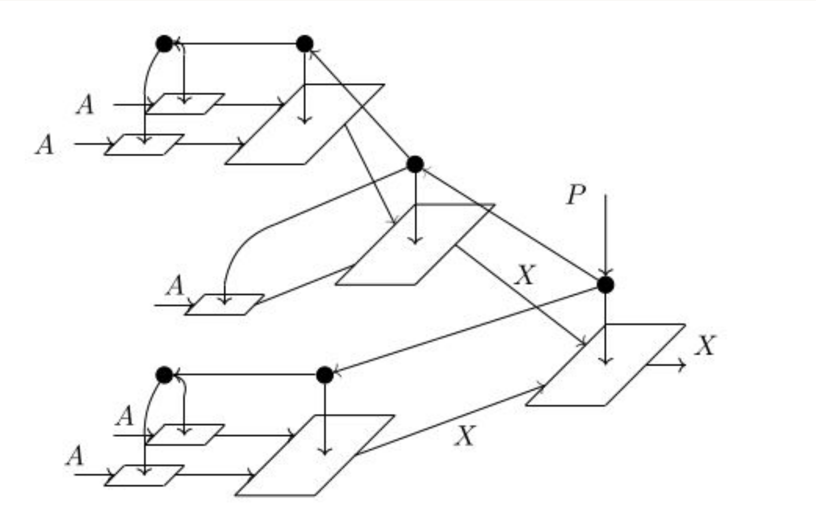
The 2-category Para

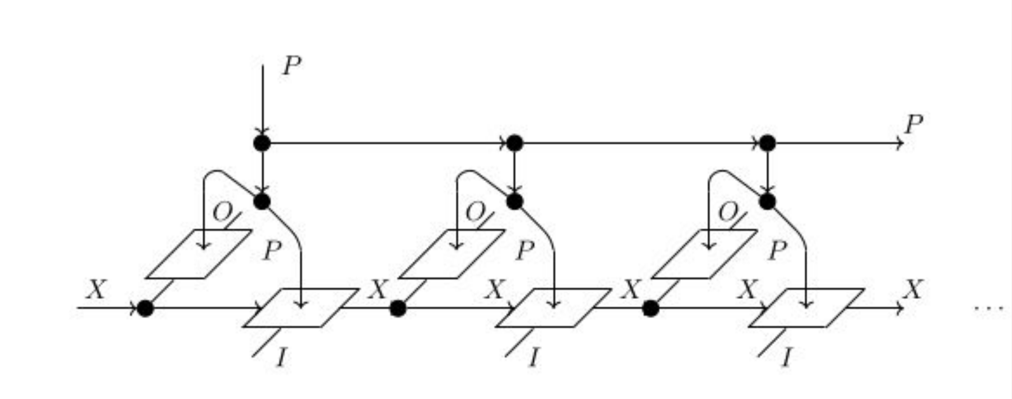
- Neural networks have nonlinearities, and we often need to explicitly track parameters
- In 2-category Para, objects are still *sets*, but a map A -> B is now a *parametric* function, a choice of (P, f), where f:PxA->B

- Morphisms are composed by chaining their computations



Weight sharing happens automatically!





Let’s recap

- Unified 2-dimensional categorical framework

- Capturing not only invertible group actions, but concepts from computer science such as trees, lists, automata, and general data types

- Allows us to talk about *structural recursion* in a principled manner

Future work

- Coalgebra to algebra morphisms model dynamic programming

$$
\begin{tikzcd}
F(P) \arrow[r, "\text{solve subproblems}"] & F(S) \arrow[d, "\text{combine solutions}"] \\
P \arrow[u, "\text{break into subproblems}"] \arrow[r, "\text{solve problem}"'] & S
\end{tikzcd}
$$

Future work

- The prevalent paradigm of machine learning is that we bake in priors ourselves into neural networks

- The Transformer architecture, specifically, the Attention mechanism challenges this assumption

Learning equivariance

The Lie Derivative for Measuring Learned Equivariance

Nate Gruver* **Marc Finzi*** **Micah Goldblum** **Andrew Gordon Wilson**
New York University

Abstract

Equivariance guarantees that a model's predictions capture key symmetries in data. When an image is translated or rotated, an equivariant model's representation of that image will translate or rotate accordingly. The success of convolutional neural networks has historically been tied to translation equivariance directly encoded in their architecture. The rising success of vision transformers, which have no explicit architectural bias towards equivariance, challenges this narrative and suggests that augmentations and training data might also play a significant role in their performance. In order to better understand the role of equivariance in recent vision models, we introduce the Lie derivative, a method for measuring equivariance with strong mathematical foundations and minimal hyperparameters. Using the Lie derivative, we study the equivariance properties of hundreds of pretrained models, spanning CNNs, transformers, and Mixer architectures. The scale of our analysis allows us to separate the impact of architecture from other factors like model size or training method. Surprisingly, we find that many violations of equivariance can be linked to spatial aliasing in ubiquitous network layers, such as pointwise non-linearities, and that as models get larger and more accurate they tend to display more equivariance, regardless of architecture. For example, transformers can be more equivariant than convolutional neural networks after training.

We can some equivariances, but not others

Limits of Transformer Language Models on Learning Algorithmic Compositions

Jonathan Thomm [1 2] Aleksandar Terzic [1 2] Geethan Karunaratne [1] Giacomo Camposampiero [1 2]
Bernhard Schölkopf [2 3] Abbas Rahimi [1]

Abstract

We analyze the capabilities of Transformer language models on learning discrete algorithms. To this end, we introduce two new tasks demanding the composition of several discrete sub-tasks. On both training LLaMA models from scratch and prompting on GPT-4 and Gemini we measure learning compositions of learned primitives. We observe that the compositional capabilities of state-of-the-art Transformer language models are very limited and sample-wise scale worse than relearning all sub-tasks for a new algorithmic composition. We also present a theorem in complexity theory, showing that gradient descent on memorizing feedforward models can be exponentially data inefficient.

potheses on how well it learns a composition of them (within the same training routine):

H1 A Transformer language model learns the composition with a constant number of samples.

H2 A Transformer relearns the sub-tasks for the composition and needs as many samples as the most difficult sub-task needs.

H3 A Transformer language model needs as many samples as the sum of relearning all sub-tasks for learning the composition.

H4 A Transformer language model needs more data samples than in H3.

One can expect that an optimal learner can solve a new composition of sub-tasks within a low constant number of

Transformer-Based Models Are Not Yet Perfect At Learning to Emulate Structural Recursion

Shizhuo Dylan Zhang *shizhuo2@illinois.edu*
University of Illinois Urbana-Champaign

Curt Tigges *curt@eleuther.ai*
EleutherAI

Zory Zhang *zoryz2@illinois.edu*
University of Illinois Urbana-Champaign

Stella Biderman *stella@eleuther.ai*
EleutherAI
Booz Allen Hamilton

Maxim Raginsky *maxim@illinois.edu*
University of Illinois Urbana-Champaign

Talia Ringer *tringer@illinois.edu*
University of Illinois Urbana-Champaign

Abstract

This paper investigates the ability of transformer-based models to learn structural recursion from examples. Recursion is a universal concept in both natural and formal languages. Structural recursion is central to the programming language and formal mathematics tasks where symbolic tools currently excel beyond neural models, such as inferring semantic relations between datatypes and emulating program behavior. We introduce a general framework that nicely connects the abstract concepts of structural recursion in the programming language domain to concrete sequence modeling problems and learned models' behavior. The framework includes a representation that captures the general *syntax* of structural re-

Interested in this? We're hiring.

- Symbolica - new startup developing foundational models for structured reasoning, at scale
- Raised $31M funding round
- Opening up offices in London, soon AUS
- Check out our job descriptions!

Principal Category Theory Scientist - UK - Categorical Deep Learning

Remote

$165,000 - $230,000 per year · Full time

Senior Category Theory Scientist - UK - Categorical Deep Learning

Remote

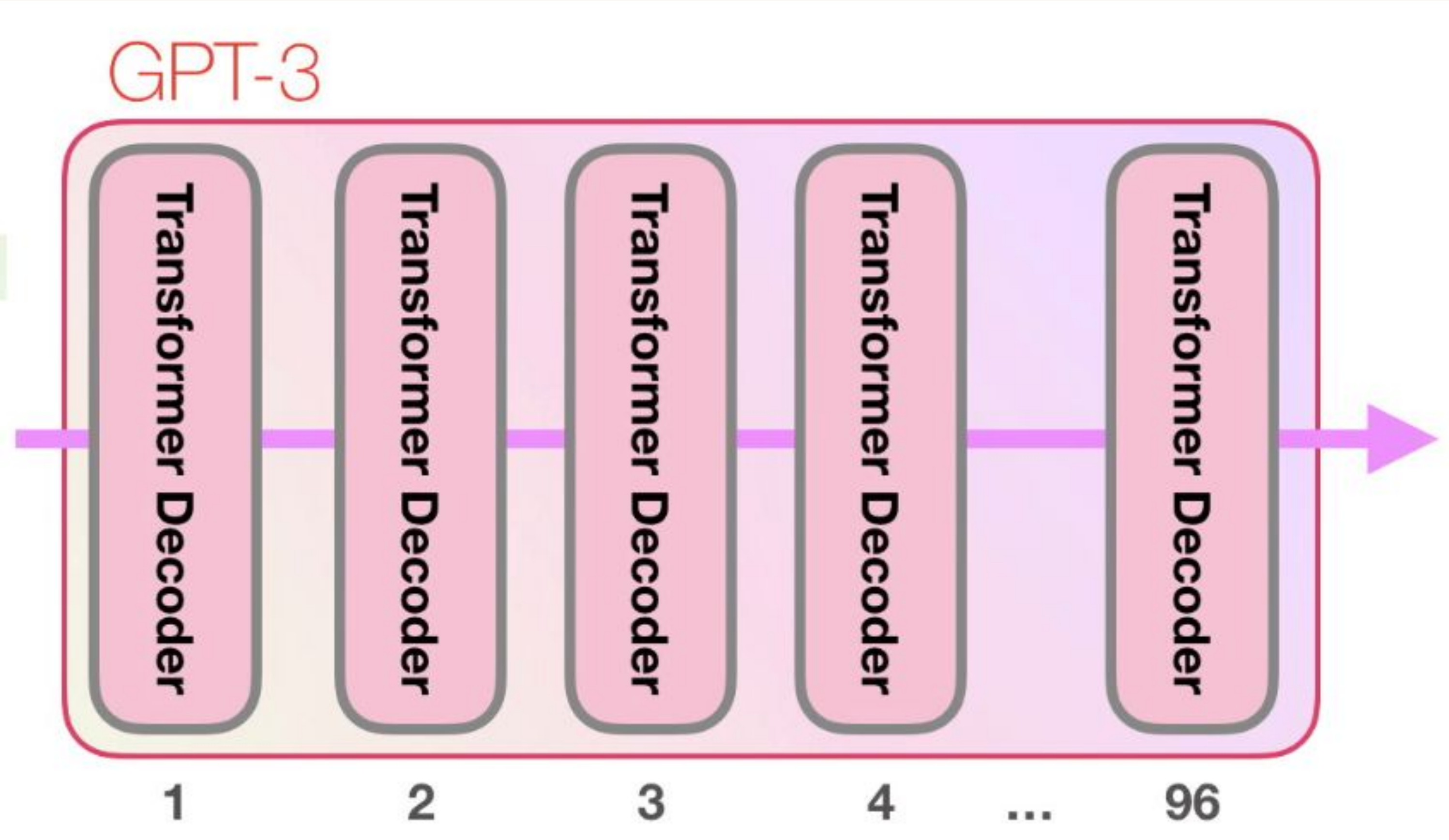
$115,000 - $140,000 per year · Full time

Category Theory Scientist - UK - Categorical Deep Learning

Remote

$75,000 - $100,000 per year · Full time


Always has been
Wait, it's all
category theory?
imgflip.com


GPT-3
Transformer Decoder
Transformer Decoder
Transformer Decoder
Transformer Decoder
Transformer Decoder
1
2
3
4
...
96