Deep Causality Presentation Notes

Hi, this presentation is about DeepCausality, a computational causality library for the Rust programming language that leans on a set of new concepts to tackle rather fundamental challenges in ML hence my proposal to host DeepCausality at the LF. **Let's jump right into it.**

Slide 2: Why donating?

- Vendor-neutral position helps to build trust.
- Open governance instills credibility and trust.
- Lastly, growing the community through increased visibility is important.

Slide 3: Agenda

- First, we look at the actual problem DeepCausality aims to solve.
- Next, we take a closer look at the implied challenges,
- Then, we look at what DeepCausality brings to the table,
- And finally, it's time to take a brief look at the next steps.

Slide 4: Headline: Problem Let's start with the first item of the agenda, the problem.

Slide 5: Problem

- It all started with modelling volatility in financial markets.
- Basically, financial markets are:
  - Interrelated
  - Subject to internal and external context
  - Essentially, you are modeling time-varying complex spatial-temporal patterns
- Deep Learning hits its limits because it is:
  - Context-blind
  - World-model-blind
  - Data-relation-blind
- Let's see where each of those items come from and what it actually means:

Slide 6: Root Cause

- Context blindness:
  - Context blind meaning a DL model only knows the data at hand
  - During training, DL aims to replicate the function describing the original data which is the definition of the underlying the universal approximation theorem.
- World-model-bling
  - means DL does not know how the data were generated and therefore, it does not cope well with, for example, shifting data-distribution, or any other structural change in the data it uses.
- Data-relational-blindness
  - Means DL does not know how data relate to previous data from the same series.
  - More profoundly, it cannot know that because the core assumption of DL is that each data point is independent from all the other data points.
  - Ok, you may ask, how much of an issue is that?

Slide 7: DARPA

- DARPA acknowledges the problem and states that current limitations in today's ML roots
    - lack of context,
    - lack of relations between data set and a
    - lack of a modelled causal mechanism.
- Fundamentally, DARPA has a point here because it is true that in many cases, data are inter-related, contextualized, and the product of a generative causal mechanism.

Slide 8: Headline: Challenges Let's move to the next item of the agenda, the challenges.

Slide 9: There are three big challenges to tackle when dealing with causality modelling

1) Adding context & background knowledge
2) Defining relations between observations
3) Modelling a **causal** mechanism

Let's look briefly at each one:

Slide 10: Context & background knowledge

- Fundamentally, context is independent of the model, which also means multiple models in the same domain can share the same context. There is utility.
- Context data are non-triviala because the differ over,
    - Time,
    - Time scale
    - Space,
    - And space time
- Also, context evolves over time hence needs regular updates.

Slide 11: Relations between observations

- This isn't exactly trivial either because data may relate to each other in
    - Time meaning previous data serve as reference point
    - Space: Locations may serve as reference point
    - And SpaceTime basically meaning geometric patterns inform data in a progressing time continuum.

Slide 12: Causal Relations (1/2)

- Let's start with a mini refresh of the basics:
- Correlation basically means two or more things happening at the same time
- Correlation can be reversed, meaning rain and umbrella is same as umbrella and rain
- Causation, however, has a more stringent definition:
    - IF A then B; AND if NOT A then NOT B
- Meaning, only if a cause means and effect and no cause means no effect, then there is a causal relation. Therefore, causation cannot be reversed
  Finding causal mechanism and structures is exceptionally hard and now lets lool why that is the case…

[Quarter TIME: 7.5 min]

Slide 13: Causal Relations (2/2)

- Single cause, single effect. Easy.
- Multi cause, single effect, okay, that's a bit more realistic but still doable
- Multi-cause, multiple effects, that's where things get interesting and it's actually quite common.
- However, in reality, there are actually multiple stages of multiple causes leading to multiple effects and you may imagine that as a complex network
- And then causality is contextual…
- **At this point, it is very clear why DARPA funds research in this field/**
- However, people working on computational causality, so lets take a brief look at what's already available today.

Slide 14: Solution space

- Judea Pearl obviously is the founding father of computational causality.
- A lot of great research happens at UCLA, John Hopkins, Harvard, and Columbia.
- Also, the industry is doing excellent work at Microsoft, Uber, and Netflix
- Lastly, I just want to briefly highlight Lucien Hardy because he is perhaps the most unique outlier by applying causality to the emerging field of quantum gravity. As we will see later, much of DeepCausality roots in Hardy's comprehensive work of uniform causal structures. Truly fascinating work. Let's share some observations

Slide 15: **Observations**

- In a nutshell, a lot of work is done in Academia and Industry alike
- Very diverse projects, so it's impossible to compare them side by side
- In general, algorithms and applications is in focus
- I highlight a few identified gaps:
  - Rust is not explored yet
  - Contextual causal inference is not explored yet
  - Causal hyper geometric structures not yet explored

Based on these observations, I've drafted some success factors

Slide 16: Key factors to success

1) Causal structures:
    a. I really want to stress the importance of structural support for interrelation between causes, data, and context as a key success factor
2) Then, context is important to relate causal relations to contextual data
3) Lastly, you really want full explanation

Slide 17: Headline: DeepCausality: Let's move to the third item of the Agenda and look at how DeepCausality helps.

DeepCausality is a hyper-geometric computational causality library for Rust that enables fast and deterministic context-aware causal reasoning over complex multi-stage causality models. Let's first look at how it differs from deep learning.

Slide 18: How DC differs from DL
I emphasize that Deep Learning and Deep Causality operate at two different ends of the spectrum and as a result are quite commentary.

Slide 19: Not all causal problems are created equal

- Causal problems can be arranged across three categories
    - Simple vs complex
    - Static vs dynamic
    - Deterministic vs Probabilistic
- Each of these categories has very different requirements, as seen on the slides
- The main idea to tackle these rather disjoint requirements is layers … (next slide

Slide 20: Four layers of causal reasoning

- Fundamentally, the domain of causal reasoning can be organized in these four layers
- **The key take away here is the first principle nature of causality combined with explicit assumptions**
- **These four layers directly inform the architecture of DeepCausality**

Slide 21: Architecture excluding context

- Here, we're looking at the core architecture of DeepCausality excluding context
- I tried to keep the color scheme consistent with the layers form the previous slides
- As we can see, each layer has been translated into a trait with a matching type.
    - Observable
    - Assumable
    - Inferable
    - Causable
- Also, each layer has been translated into a reasoning trait that provides a default implementation for extending collections.
    - Observable reasoning
    - Assumable reasoning
    - Inferable reasoning
    - Causable reasoning
- I want to emphasize that the causal reasoning requires four traits because the hyper graph implementation is separate from the collection extension.
- Before looking into the implementation techniques, I just want to give a brief overview of the context architecture.

Slide 22: Context Architecture

- To the left, we have the contextuable trait and the contextuable graph trait which them is implemented in the matching type
- To the right, we see four traits that specify date, time, space, and spacetime with a corresponding default type.
- The big idea is that each unit of context, called a contextoid, might store a unit of data, time, space, or spactime.
- To make all that work in Rust, lets look at the underlying implementation principles.

Slide 23: Implementation principles

- There are just three big ideas used in the implementation.
    - Protocol bases causality
        - There is currently an effort at Apple to bring end to end differential programming into the Swift compiler.
    - Recursive isomorphic data structures
        - Technically self-referential types
    - Disjoint algebraic types.
        - Basically, just nested Enums.
- DeepCausality was implemented with all these three techniques.
- And now, let's look at the four pillars of DeepCausality

[HALF TIME: 15 min]: 15 more minutes

Slide 24: Four pillars of DeepCausality

Slide 25: Hypergeometric deep causality

- Let's begin with the first pillar, hypergeometric deep causality

Slide 27:

- Let's move on to the second pillar of deep causality, recursive causal data structures

Slide 28: Recursive causal data structures

- Because abstract data structures are a bit hard to imagine, I have drawn a diagram

Slide 29 (image): Recursive Causal Data Structures

- The big circle is the causaloid that stores the hypergraph.
- Each node of the hypergraph is another causaloid that might be a collection, another graph, or a singleton instance.
- Zooming into the causaloid i#3, we see it is a collection of another three causaloids.
- Let's briefly touch on the implementation.

Slide 31:
Let's move on to the third pillar of deep causality, the hypergraph context

Slide 32: Hypergraph context

- As stated at the beginning, context touches data over time, space, and space-time
- Let's start with the conventional notion of these
    - Time
    - Space
    - SpaceTIme
    - Data context
- When modeling a comprehensive data context, the conventional notion lacks quite a bit due to lack of uniform structure. Let's see what DeepCausality brings to the table

Slide 33: Hypergraph context in DeepCausality

- Uniform approach of representing time, space, spacetime, and data, as monoid elements,
- Modelling context becomes quite a bit easier,
- but we need one more bit

Slide 34: Contextoid

Slide 35 (image): Hypergraph context

- To the right side, we see the causal model from the previous slides.
- To the left side, we see an example context containing some arbitrary data structured in a time-graph. A time graph encodes time as a hyper-graph
- The causal function inside a singleton causaloid may then reference some context graph data to inform its reasoning.

Slide 36: Implementation

[ThreeQuarter TIME: 22.5 min]: 7 more minutes

Slide 37: Causal Model

- When you combine a causaloid with a context, you get a contextualized causal model
- Again, a bit hard to imagine so I have drawn a diagram

Slide 38: Contextualized Causal Model

- To the left, we see the same sample context as before
- To the right, we see the recursive causal hypergraph embedded into a causaloid
- To contextualize a complex causal model,
    - Each node-causaloid may refer to one or more piece of context data and use it in its causal function.
    - The context is accessed through an immutable reference. The idea is that the model never changes its context.
    - For a static context, accessing nodes is relatively easy. Example code is in the repository.
    - When generating and updating a dynamic context, accessing context nodes requires careful consideration as how the causaloid can determine the index position in the context graph. I am currently developing a code example for this scenario
- Let's take a brief look at the implementation.

Slide 39: Implementation

- The main take-away here is, to customize the context you have to
    - First extend the super trait, say tempoid
    - Then implement both, your custom trait and the super trait
    - Use your custom type when constructing a context
    - Lastly, import your custom trait in you causaloid so you can access custom functionality
- Again, code example in the repository illustrates the nitty-gritty details

Slide 40:
Let’s briefly touch the last pillar of deep causality, causal state machines

Slide 41: causal state machines

In the interest of time, I just want to highlight that causal state machines emerged from a particular requirement to have a state machine that can be generated dynamically at runtime without prior knowledge of all states and that was simply not possible with finite state machines.

causal state machines were specifically designed for dynamic control systems and that’s were there actually used.

Slide 42: Skip if necessary
- For details, please look at the code and example in the repo

Slide 43: Skip if necessary

Slide 44: Value of DeepCausality

Slide 45: Headline: Next Step:
- Let’s move to the last item of the Agenda, the next step

Slide 46: Next Step
- DeepCausality is still in its infancy but already established some groundwork such as context, novel causal structures, and causal state machines.
- Next step would be supporting multiple contexts and causal learning
- Causasl learning may take a while until its clear how it can be done on hyper geometrict structures

Slide 47: Information for proposal
- In terms of potential collaborations within the LF AI & Data, I have selected some complimentary projects. Details need to be explored.

[TIME’s UP: 30 min]:

Slide 48: Thank You
- Thank you to the LF for inviting me to present DeepCausality
- Thank you for everyone who dialed into the Zoom meeting today
- And now it’s time for the Q&A