Applications Of Recursion Schemes On Neural Networks

Minh Nguyen
Department of Computer Science, University of Bristol
mn15104@bristol.ac.uk

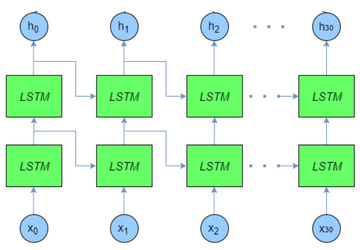
Supervisor: Dr. Nicolas Wu

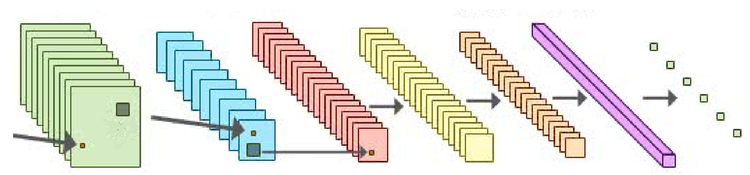
Motivation

There are multiple narratives to which deep learning can be understood. Neuroscience draws analogies to biology. A representative narrative focuses on the transformations of data and the manifold hypothesis. The probabilistic narrative interprets neural networks as finding latent variables. Although these perspectives aren't mutually exclusive, they present very different ways of expressing deep learning. An uninvestigated narrative is the relationship between neural networks and functional programming. This project discusses how the structure and learning process of neural networks can be represented with recursion schemes.

Specifically, I demonstrate implementations of:

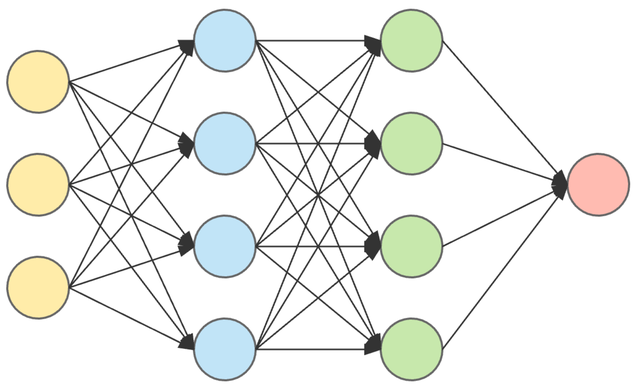
- Fully Connected Networks
- Convolutional Networks
- Deep LSTM Networks





Neural Networks

Neural networks are represented by a graph of nodes and edges containing hyper-parameters. The simplest models are feed-forward networks, where subsets of nodes exclusively belong to individual layers which are connected side-by-side.



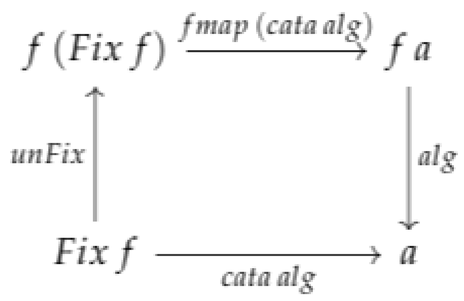
The process of learning a neural network consists of two stages: **forward propagation** and **back propagation**.

Recursion Schemes

Nested structures are very common in programming environments. However, with each recursive structure, any corresponding recursive functions we define must be type-specific to it. This prompts the need to have a generalisation over recursive traversals which abstracts away recursion from the data type. By decoupling how a function recurses over data from what the function actually does, we reduce cognitive overhead and can focus on the core behaviour of the function. This entire notion is captured through **recursion schemes**.

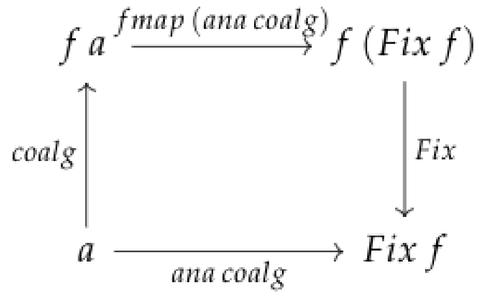
Catamorphisms

Folds allows us to evaluate recursive data structures without having to write recursive functions. **Catamorphisms** are a generalisation over folds, such that we don't have to be type specific to the data type.



Anamorphisms

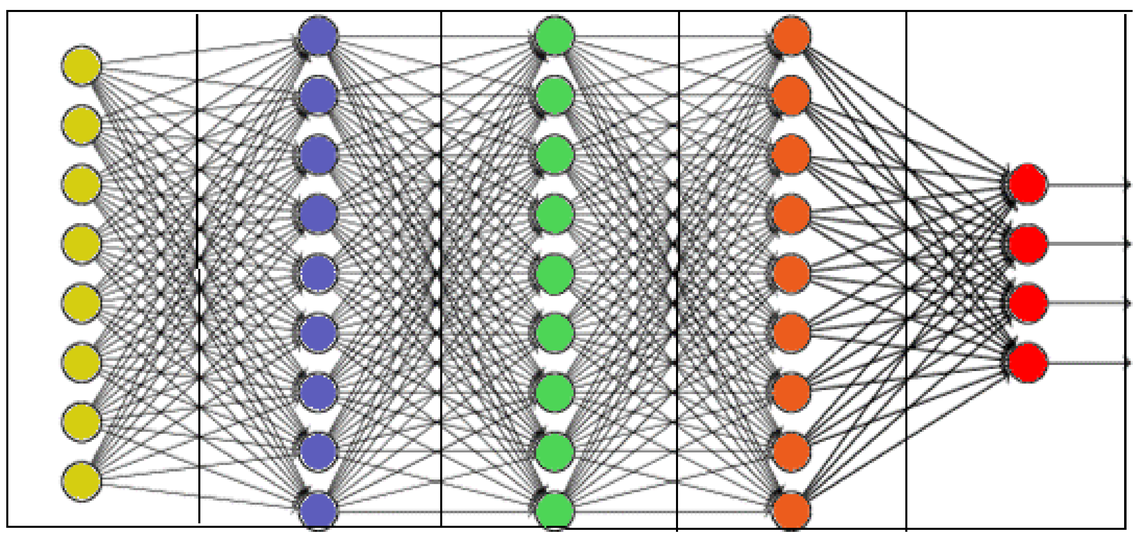
Unfolds let us take a seed value and generate type-specific recursive data structures without defining recursive functions. **Anamorphisms** are a generalisation over unfolds such that we don't have to be type specific to the data type.



Forward And Back Propagation As Catamorphisms And Anamorphisms

• **Forward propagation** consists of passing an input to the first layer of a neural net, with every layer performing a computation using its hyperparameters to produce an output which is propagated to the next layer. This eventually leads to an output. Evaluating a neural network to acquire an output is what a fold essentially achieves. If we can fold over a neural network to get an output, then this can be modelled as a **catamorphism**.

We can in fact visualise a neural network as a list of layers, where we fold over them with a function which performs forward propagation between each layer.



• **Backward propagation** traverses in the opposite direction. From the last layer, every layer performs a computation using its output and back propagated variables from the next layer to update its hyperparameters. Using the output as a seed value to generate an updated neural network is what an unfold essentially achieves. If we can unfold over an output to get a neural network, then this can be modelled as an **anamorphism**.

• The composition of a catamorphism and an anamorphism is called a **metamorphism**; hence updating a neural net with a given input sample can be modelled by a metamorphism.

Existing Work

Currently there exists no formal research recognising recursion schemes or functional programming as being an innate behaviour of neural networks - the only two acknowledgements are presented as hypotheses.

One briefly compares the recursive neural network (RNN) to a catamorphism, stating "*the connection to category theory has only recently been recognised, and detailed analysis in this context has not yet been carried out to the best of our knowledge.*"

The other compares various models to functional programming concepts and makes a specific resemblance between tree nets to catamorphisms and inverse tree nets to anamorphisms. this is a limited perspective. The writer comments "*I expect this idea is wrong, because most untested ideas are wrong. But it could be right, and I think it's worth talking about.*"

This emphasises the state of the relationship of functional programming and recursion schemes with neural network; I believe recursion schemes can be applicable to a broad range of neural network models.