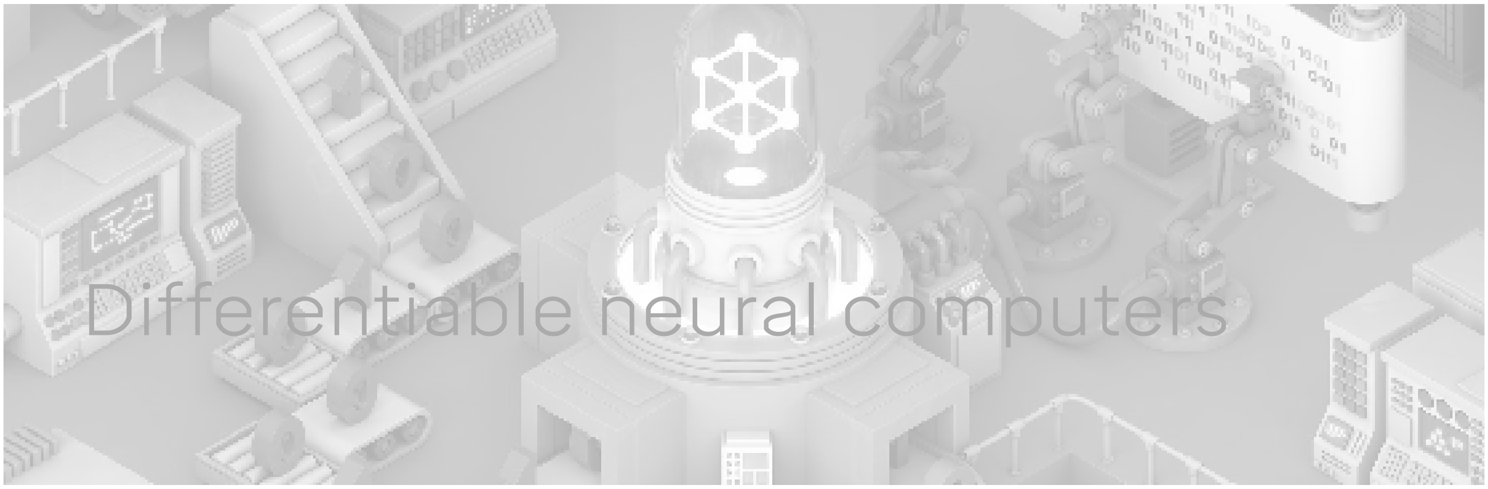

Differentiable neural computers

In a ***recent study in Nature****, we introduce a form of memory-augmented neural network called a differentiable neural computer, and show that it can learn to use its memory to answer questions about complex, structured data, including artificially generated stories, family trees, and even a map of the London Underground. We also show that it can solve a block puzzle game using reinforcement learning.*

Plato likened memory to a wax tablet on which an impression, imposed on it once,
d remain fixed. He expressed in metaphor the modern notion of plasticity – that
inds can be shaped and reshaped by experience. But the wax of our memories
not just form impressions, it also forms connections, from one memory to the next. Philosophers like John Locke believed that memories connected if they were formed nearby in time and space. Instead of wax, the most potent metaphor expressing this is Marcel Proust's madeleine cake; for Proust, one taste of the confection as an adult undammed a torrent of associations from his childhood. These episodic memories (event memories) are known to depend on the hippocampus in the human brain.

Today, our metaphors for memory have been refined. We no longer think of memory as a wax tablet but as a reconstructive process, whereby experiences are reassembled from their constituent parts. And instead of a simple association between stimuli and behavioural responses, the relationship between memories and action is variable, conditioned on context and priorities. A simple article of memorised knowledge, for example a memory of the layout of the London Underground, can be used to answer the question, “How do you get from Piccadilly Circus to Moorgate?” as well as the question, “What is directly adjacent to Moorgate, going north on the Northern Line?”. It all depends on the question; the contents of memory and their use can be separated. Another view holds that memories can be organised in order to perform computation. More like lego than wax, memories can be recombined depending on the problem at hand.

Neural networks excel at pattern recognition and quick, reactive decision-making, but we are only just beginning to build neural networks that can think slowly – that is, deliberate or reason using knowledge. For example, how could a neural network store memories for facts like the connections in a transport network and then logically reason about its pieces of knowledge to answer questions? **In a recent paper**, we showed how neural networks and memory systems can be combined to

ly. These models, which we call differentiable neural computers (DNCs), can
from examples like neural networks, but they can also store complex data like
uters.

In a normal computer, the processor can read and write information from and to random access memory (RAM). RAM gives the processor much more space to organise the intermediate results of computations. Temporary placeholders for information are called variables and are stored in memory. In a computer, it is a trivial operation to form a variable that holds a numerical value. And it is also simple to make data structures – variables in memory that contain links that can be followed to get to other variables. One of the simplest data structures is a list – a sequence of variables that can be read item by item. For example, one could store a list of players’ names on a sports team and then read each name one by one. A more complicated data structure is a tree. In a family tree for instance, links from children to parents can be followed to read out a line of ancestry. One of the most complex and general data structures is a graph, like the London Underground network.

When we designed DNCs, we wanted machines that could learn to form and navigate complex data structures on their own. At the heart of a DNC is a neural network called a controller, which is analogous to the processor in a computer. A controller is responsible for taking input in, reading from and writing to memory, and producing output that can be interpreted as an answer. The memory is a set of locations that can each store a vector of information.

A controller can perform several operations on memory. At every tick of a clock, it chooses whether to write to memory or not. If it chooses to write, it can choose to store information at a new, unused location or at a location that already contains information the controller is searching for. This allows the controller to update what

er needed. When the controller does write, it sends a vector of information to hosen location in memory. Every time information is written, the locations are connected by links of association, which represent the order in which information was stored.

As well as writing, the controller can read from multiple locations in memory. Memory can be searched based on the content of each location, or the associative temporal links can be followed forward and backward to recall information written in sequence or in reverse. The read out information can be used to produce answers to questions or actions to take in an environment. Together, these operations give DNCs the ability to make choices about how they allocate memory, store information in memory, and easily find it once there.

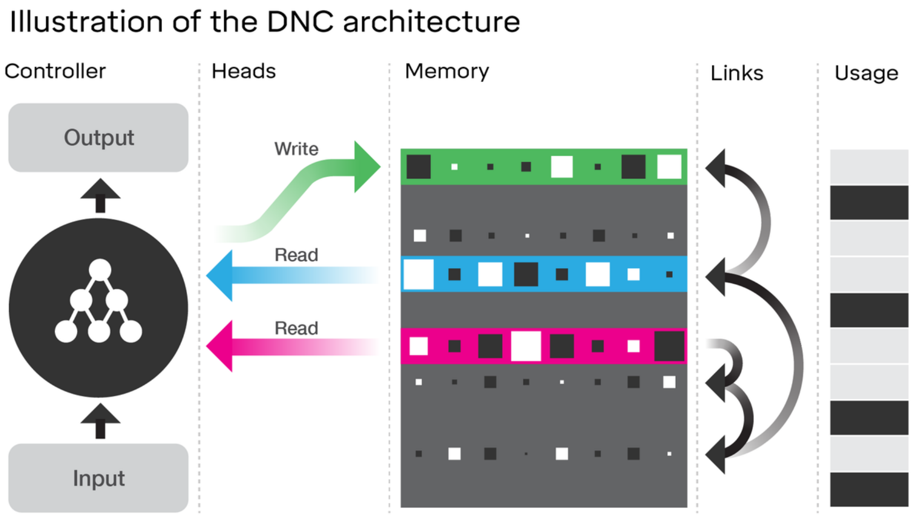

Illustration of the DNC architecture. The neural network controller receives external inputs and, based on these, interacts with the memory using read and write operations known as 'heads'. To help the controller navigate the memory, DNC stores 'temporal links' to keep track of the order things were written in, and records the current 'usage' level of each memory location.

e non-technical reader, it may seem a bit odd that we have repeatedly used ;es like "the controller can" or "differentiable neural computers ... make :es". We speak like this because differentiable neural computers learn how to use memory and how to produce answers completely from scratch. They learn to do so using the magic of optimisation: when a DNC produces an answer, we compare the answer to a desired correct answer. Over time, the controller learns to produce answers that are closer and closer to the correct answer. In the process, it figures out how to use its memory.

We wanted to test DNCs on problems that involved constructing data structures and using those data structures to answer questions. Graph data structures are very important for representing data items that can be arbitrarily connected to form paths and cycles. In the paper, we showed that a DNC can learn on its own to write down a description of an arbitrary graph and answer questions about it. When we described the stations and lines of the London Underground, we could ask a DNC to answer questions like, “Starting at Bond street, and taking the Central line in a direction one stop, the Circle line in a direction for four stops, and the Jubilee line in a direction for two stops, at what stop do you wind up?” Or, the DNC could plan routes given questions like “How do you get from Moorgate to Piccadilly Circus?”