Forthcoming in *Mind & Language*
Please cite published version, doi.org/10.1111/mila.12387

Moving Beyond Content-Specific Computation in Artificial Neural Networks

Nicholas Shea

Abstract

A new wave of deep neural networks (DNNs) have performed astonishingly well on a range of real-world tasks. A basic DNN is trained to exhibit, in parallel, a large collection of different input-output dispositions. While this is a good model of the way humans perform some tasks automatically and without deliberative reasoning, more is needed to approach the goal of human-like artificial intelligence. Indeed, DNN models are increasingly being supplemented to overcome the limitations inherent in dispositional-style computation. Examining these developments, and earlier theoretical arguments, reveals a deep distinction between two fundamentally different styles of computation, defined here for the first time: content-specific computation and non-content-specific computation. Deep episodic RL networks, for example, combine content-specific computations in a DNN with non-content-specific computations involving explicit memories. Human concepts are also involved in processes of both kinds. This suggests that the remarkable success of recent AI systems, and the special power of human conceptual thinking are both due, in part, to the ability to mediate between content-specific and non-content-specific computations. Hybrid systems take advantage of the complementary costs and benefits of each. Combining content-specific and non-content-specific computations both has practical benefits and provides a better model of human cognitive competence.

Running head: Moving Beyond Content-Specific Computation

Keywords
computation; deep neural networks; distributed representation; content-specific; explicit memory; concepts

Contents

(1) Introduction

For those of us who have long championed artificial neural networks (ANNs), 2012 was a good year. Finally, ANNs were starting to out-perform classical computational architectures on some tasks. Their promise – to be a powerful supplement to the undoubted prowess of classical computation – was beginning to be fulfilled. Using fast, special-purpose computer chips, a variety of technical tricks, and huge databases of training data, neural networks with multiple hidden layers began to out-compete all other computational systems on several benchmarks (Buckner, 2018). These deep neural networks (DNNs) have now demonstrated impressive abilities in one domain after another, including: image classification (Krizhevsky et al., 2012; Eslami et al., 2018), strategic games (computer games: Mnih et al., 2015; Go: Silver et al., 2016), natural language processing (Bahdanau et al., 2014; Brown et al., 2020; Floridi & Chiriatti, 2020) and protein folding (Senior et al., 2020; Jumper et al., 2020 [abstract]).

In its basic form, a DNN is trained to react to a range of inputs with appropriate outputs. Training requires a lot of data, carefully curated to avoid falling into unhelpful local minima (Botvinick et al., 2019; Marcus, 2018). It also calls for prodigious amounts of processor time, with huge numbers of individual training events. A basic DNN is effectively trained to exhibit, in parallel, a range of different input-output dispositions. Once trained, DNNs prove to be relatively inflexible. When tasked with learning something new – acquiring an additional input-output disposition – they are always at risk of catastrophically forgetting what has been so laboriously learnt to date (French, 1999).

Nevertheless, this is probably a good model of tasks that humans perform automatically, without deliberative reasoning. To approach the goal of human-level AI, other computational capacities are required. Modellers have experimented with many ways of supplementing the basic DNN architecture to circumvent its limitations. New developments are emerging all the time. Some of these supplements make use of a style of computation that is quite different from the disposition-style computations performed by a basic DNN. I use 'disposition-style' to capture the rough but intuitive idea that a computational step may operate within a specific domain by directly mapping a range of particular inputs to a range of particular outputs. The aim of this paper is to highlight this distinction and to capture it precisely. To do that I will introduce a distinction between 'content-specific' and 'non-content-specific' computation (defined in section 4).

One standard view in cognitive science is that ANNs (connectionist systems) have a fundamental limitation in the structure of their representations: that they lack compositionality (Fodor & Pylyshyn, 1988; Smolensky, 1988; Fodor & McLaughlin, 1990). That is, their representations are not built out of recombinable elements. While agreeing with the importance of compositionality, the distinction in this paper runs deeper. It is found, not in the way representations are structured, but in how they are processed, in the kinds of computations which the system can perform. I build on existing views of computational capacities, in particular the importance of variable binding and of role-filler independence. My category of non-content-specific computation generalises both of these ideas, in different ways. Although the capacity for variable binding or role-filler independence is often taken to be simply superior, the contrast between two forms of computation that I draw here allows

us to see why there are substantial benefits in a system that has access to both forms of computation. Giving DNN models the capacity for non-content-specific computation is an essential step on the road to building fully-functional human-like artificial intelligence. Human cognition has the benefits of this hybrid, and I argue that in fact concepts act as mediators between the two forms of computation. That puts concepts at the heart of the special power of human cognition.

The paper is structured as follows. Section 2 looks at memory and at models that supplement the basic DNN architecture with an 'episodic memory' store: a record of each stimulus or situation encountered. Section 3 connects this advance to an older literature on the benefits of separating memory from computation. Section 4 argues that to use explicit memories in this way calls for a new computational capacity, the ability to perform non-content-specific computations. Section 5 illustrates this distinction by reference to human conceptual thought. I argue that human concepts are involved in computations of both kinds, and mediate between them. With the distinction then clearer, section 6 circles back and asks which transitions within a DNN model count as content-specific and which as non-content-specific. Finally, section 7 describes how the distinction relates to several other ways that theorists have sought to contrast ANNs with classical computational systems.

(2) Obstacles and Solutions

ANNs are trained to have a certain input-output profile. For example, one set of images is mapped to the output "train", another set to the output "cherry", and so on. During training, inputs are presented for which the desired outcome is known. Gradually, the strength of the connections between network units is adjusted. By cycling through a wide range of inputs, making tiny adjustments each time, the network can eventually be trained to produce the right output for all of the trained inputs. Various arrangements help with this endeavour, like depth, and training some layers before others. These also encourage the network to find a solution which generalises to new samples (Bottou, 2014). The trained system is thus designed to approximate a desired non-linear function from input space (e.g. image bitmaps) to output space (e.g. labels) (Botvinick et al., 2020).

Memory and computation are combined, unlike in a classical computer. The computations performed by the trained system map specific regions of input space to specific regions of output space. Given input I_1, it outputs O_1. Its memory of training episodes is implicit in its disposition to make that transition. The whole weight matrix has been adjusted to achieve the $I_1 \rightarrow O_1$ disposition, while at the same time preserving all the other dispositions called for by the training set: $I_2 \rightarrow O_2$, $I_3 \rightarrow O_2$, $I_4 \rightarrow O_3$, and so on. The computation requires it to do something different with inputs in the I_1 region from what it does with inputs in the I_2 region. Rather than transforming all inputs according to a common principle, to a first approximation the appropriate output to inputs in region I_1 says nothing about how the network should respond to inputs in region I_2.

Precisely because of this, there is a constant threat that the adjustments needed to ensure $I_2 \rightarrow O_2$ will undermine those put in place to achieve $I_1 \rightarrow O_1$ (French, 1999). That it was even possible, for real world data sets, to find a weight matrix which would achieve all the

different required input-output dispositions at the same time was a significant discovery. It turned out that, even when there are hundreds or thousands of different input-output dispositions to be achieved, learning by backpropagation of error can find a solution that achieves them all at once. Even then, there is an ever-present risk of interference – catastrophic forgetting – if the trained network is tasked with learning more. That is because the way inputs in the I_1 to I_n regions are transformed into outputs is often uninformative about what should be done with a new input I_{n+1}, unlike when inputs are processed according to a common principle which does not depend on their specific content.

A note of representations and levels. I am concerned with representations at the level of distributed patterns of activation, not the activity of individual nodes. In a DNN architecture, the functional level that connects with what the system has been trained to do – the tasks it has been designed to perform (Shea, 2018) – consists of the transformation of distributed patterns of activation from input layer, through intermediate layers, to output layer. In a representational explanation which captures the way it performs its task (Cichy et al., 2016; Kriegeskorte & Diedrichsen, 2019), the representations are distributed patterns of activation in the state space of each layer (Shea, 2007). This is the level at which it has been trained to implement a mapping that takes different regions of input space to different outputs. At lower levels we find common principles at work, for example in the way the activation of individual units is convolved with a weight matrix, summed, and passed through an activation function. At a still lower level we find at work the common principles of solid state physics. DNN models are implemented in classical computers. At the level of implementation, we find variables and what I will characterise as non-content-specific computations. My claim about content-specific computation in DNNs applies at the level of the neural network model (however it is implemented): the level where the representations are distributed patterns of activation. This is the level of interest for us: it is where the DNN architecture is distinct from a classical computational architecture.

The recent success of DNNs is not just a matter of brute force statistical learning. It also depends on direct hand-design of architectural features and algorithmic biases (Botvinick et al., 2019, p. 417). For example, convolutional DNNs build in an architectural bias that capitalises on the translational invariance of images. In this case there are interesting parallels with the way real brains perform the same task (Yamins et al., 2014; Cichy et al., 2016). The architectural features in the model may correspond to evolved architectural constraints and learning biases in the brain. The critical periods observed for some forms of human learning (Werker et al., 1981) may be an adaptation to prevent catastrophic forgetting of what has been learnt by the brain. This suggests that DNNs are a good model of some cognitive tasks, tasks that humans perform rapidly and without deliberate reasoning. Where the system has access to enough experience, and the task environment is sufficiently stable, this proves to be an excellent computational solution. Although learning-heavy, it is computation-light. In exercising its trained disposition, the appropriate output is generated quickly. That is quite unlike, for example, the processing time called for in model-based system that performs a tree search through a causal model of the domain. A deep network proceeds via several intermediates, but each is typically a direct input-output step, a disposition-style computation.

These points apply equally to the newer AI systems that combine deep neural networks with reinforcement learning: deep RL. In deep RL, computational outputs are not specified directly, as in a DNN doing supervised learning to produce labels in response to images, say. Instead the system relies on a reinforcement signal to learn a non-linear mapping from inputs to state values or action values (Botvinick et al., 2020). These systems retain the computational efficiency of earlier DNNs, and share the same drawbacks, being data inefficient and relatively inflexible when outcome values change (Hassabis et al., 2017, p. 252).

Botvinick et al. (2019) describe two leading techniques deployed to overcome these limitations. The first is to train a deep RL network on a range of related reinforcement learning tasks, so-called 'meta-RL'. Standardly, DNNs bring to a problem only a weak inductive bias (assumptions about the task domain). They learn very slowly. Increasing the learning rate leads to catastrophic interference between different input-output mappings. With meta-RL, the way activity unfolds over time in the trained network, without changing connection strengths, achieves something that amounts to reinforcement learning. The dynamics of the trained network effectively implements an RL algorithm for rapidly learning the contingencies in a particular RL problem. Its training has given it an overall inductive bias that is appropriate to the kinds of RL problems it will face. For example, a meta-RL system might be tasked with learning which pictures are likely to be rewarded. The system's dynamics keeps track of the changing probabilities that different stimuli will deliver rewards. When state values or probabilities change, the system can rapidly adjust to the new environment without having to alter the weight matrix. The disposition to do rapid reinforcement learning of the appropriate kind is embedded within the input-output profile of the trained dynamic system.

The second tactic described by Botvinick et al. (2019) is to add episodic memory: explicit representations of previously encountered states and rewards (Blundell et al., 2016; Pritzel et al., 2017; Gershman & Daw, 2017). An episodic memory store records each state encountered, together with the total reward received in subsequent time-steps (discounted as they occur further in the future). For example, in a system learning to play video games, each state is a representation of the array of pixels on the screen (embedded in a lower dimensional state space). What counts as a reward is winning the game, so the discounted future reward in a state is effectively an estimate of the probability of winning the game starting from that state (i.e. that arrangement of pixels). A model-free RL system would simply record a reward value for each state in a fixed list of states, updating these values when reward is delivered (e.g. using temporal difference learning). A system with an episodic memory store can record what happened each time a given state was encountered, storing each as a separate entry. It can also store actions and record the rewards achieved in response to each action that it performed on every encounter with a given state. When encountering a new state, the value of each potential action is calculated by averaging the action values recorded for similar states, weighted by the degree of similarity to the current state. For example, encountering a slightly different array of pixels for the first time, it will look at the rewards received when starting from closely-related frames (states that are nearby in the state space in which pixel space is embedded). At the same time, a DNN uses gradient descent learning based on the reward signal gradually to improve the embedded state representations (Pritzel et al., 2017; see also Wayne et al., 2018).

Episodic deep RL is a 'non-parametric' model, since the number of parameters whose values need to be learnt is not fixed. The amount of storage dedicated to memory expands as more states are encountered and stored in episodic memory. The computation of value does, however, become increasingly demanding as more episodes are stored, since they must all be searched in order to find the k nearest neighbours to enter into the similarity computation (Blundell et al., 2016, p. 3; Yang et al., 2020, p. 129276). The advantage is that the system can do one-shot learning about a newly encountered state, giving it a good chance of acting appropriately when encountering that state again.

An early example of this architecture was the differentiable neural computer (DNC) of Graves et al. (2016). Although the DNC has proven difficult to scale to high dimensional problems (Rae et al., 2016), it has been an important step forward, leading to further developments (Putin et al., 2018; Chen et al., 2021). The general idea of supplementing a DNN with an explicit memory store has proven useful in many domains (e.g. Jaderberg et al., 2019). I will focus on the DNC as an early, specific example.

When the system has a store of explicit memories, a further step is to use 'experience replay' to drive further learning in a DNN (Hassabis et al., 2017; Liu et al., 2020). Representations of past states and rewards are used 'offline' to drive learning in a DNN, as well as 'online' when deciding how to act on encountering a new state. Experience replay allows a system to learn long-distance associations that they have not directly encountered. For example, having observed the pairs like 'hat'-'dog' and 'dog'-'key' during training, the system can learn that 'hat' goes with 'key', even though they have never seen these stimuli paired together before (Banino et al., 2020).

AI systems have also begun to introduce some forms of compositionality, representing states using recombinable elements (Higgins et al., 2016; Eslami et al., 2016; Lake et al., 2017; Santoro et al., 2017; Madarasz & Behrens, 2019; Locatello et al., 2020). Furthermore, the model-free learning of episodic deep RL can be combined with model-based reasoning about the task. Unlike a model-free system, a model-based system represents aspects of the structure of its environment, for example the way actions cause particular outcomes, or the way states are related to each other spatially or causally (Lake et al., 2017). For example, AlphaGo was given the rules of Go – a world-model of how states of the game unfold. It could then search through branching trees to see how a game might unfold from a given position (Silver et al., 2016). Deep RL systems often fail in environments where actions make a big difference to the outcomes and to the rewards accessible from a given state (Hassabis et al., 2017; Botvinick et al., 2020). Separating learning about how states are structured from learning about rewards has helped overcome this problem (Madarasz & Behrens, 2019). Nevertheless, we are still a long way from seeing flexible planning using recombinable elements (Hassabis et al., 2017; Marcus, 2018; Botvinick et al., 2020), presenting as it does the potential for a computationally demanding combinatorial explosion.

In summary, although the dispositional approach has considerable mileage, e.g. in meta-RL systems, adding a store of explicit memory has opened up a whole swathe of new computational possibilities, developments that are already beginning to overcome the limitations of the earlier DNN and deep RL systems. This advance comes at a computational

cost. As more memories are stored and used in computations, working out how to act becomes increasingly demanding of processing time and power.

(3) Adding Explicit Memory to a Dispositional System

Graves et al. (2016) identify two key attributes which give the DNC an advantage over the basic DNN architecture. This section looks at the first, concerning memory. That lays the groundwork for a second attribute, less obvious but arguably more fundamental, that we turn to in section 4.

The first advantage of the Graves et al. model is that its way of storing information about the past is extensible. Learning is 'non-parametric': it is not just a matter of adjusting a fixed set of parameters (weights) to deal with the training set. As well as being a more efficient use of storage, this also helps with interference, since learning about a new episode does not immediately require any change to what has been learnt already. A new episode is stored explicitly, using new resources.

The discovery that there are practical benefits to storing memory explicitly chimes with an older, more theoretical argument. Long before the recent advances in DNNs, Charles R. Gallistel was arguing that it is of critical importance to separate memory from computation (Gallistel, 2006, 2008; Gallistel & King, 2010). He has two arguments. The first is a proof in the theory of computation. It is based on the limitations of a finite state automaton. What this tells us about neural networks, artificial or biological, depends on tricky issues about implementation. Computational theory implies that a powerful computation must be able, in some sense, to write information to memory and read it to inform behaviour. What it does not tell us is whether gradient descent learning in a standard DNN implements this principle in some way (Morgan, 2020).

Whichever way that issue turns out, Gallistel & King (2010) have a second, more pragmatic argument. They observe that across huge amounts of experimentation and innovation in practical computing machines in the last 80 years, all the ones which have done anything useful (as of 2010) have built in an architectural separation between computation and explicit memory (Gallistel & King 2010, p. 144). The basic ANN architecture makes no such separation. That makes it resource-intensive for the system to deal with situations where the appropriate output depends on past events, or on a long chain of inputs.

ANNs combine computation and memory together. They "remember" the past by changing the weight matrix, which is the basis for their computational dispositions. Training changes the way distributed representations are processed, that is, it changes the computational processes operative at this level. Suppose which output is appropriate at a time depends upon an input n time steps earlier. For example, it might be best to turn left at the lone pine now because you observed nearly-ripe fruit at a different location some days ago. How the organism should react to a range of things it could observe at the current time may be completely different depending on what was observed at various points in the past. The system needs to be set up in such a way that it can react appropriately.

We could consider the whole chain of states encountered in the last n time steps as a single input, with respect to which there is some appropriate output: I_1 (unripe fruit), I_2, …, I_n should lead to output O_1, whereas I_1' (ripe fruit), I_2, …, I_n calls for a different output, O_1'. A machine could be set up so that it is in a state which disposes it to deal appropriately with both of these (long) inputs. Its input-output characteristics map I_1, I_2, …, I_n to O_1 and I_1', I_2, …, I_n to O_1'. The problem is, to do that the system must have dispositions to react to a very large number of inputs. It needs computing machinery ready to react in an individual way to each. Just as a look-up table needs one line for every input it might encounter, a dispositional-style computer needs to have a physically realised computational state to underpin the input-output disposition called for by every chain of n states it could encounter. The number of possibilities is exponential in the length of the input.

Dealing with past states thus runs up against the 'infinitude of the possible' (Gallistel & King 2010, p. 136). A system that relies only on input-output dispositions can remember the past only by changing the state of the processor. A machine with explicit memory for past inputs can side-step this problem by writing an input to memory and then computing with it only when subsequently relevant. For this reason, Gallistel & King argue that, in practical terms, 'a machine with read/write memory [is] vastly more capable than a machine that can remember the past only by changing the state of the processor' (p. 147).

Gallistel & King (2010, pp. 43-54) argue that dealing with the past is a matter of implementing an input-output function that takes many arguments as input (n arguments, if you need to take account of the past n time steps). A function of n arguments can be decomposed into a series of functions of two arguments. The output of one function acts as the input to the next. But a function of two arguments cannot be decomposed into a series of functions of one argument. The computational step calculating that function has to take both arguments as input and compute with them together. In a classical computer a function of n arguments is implemented by breaking it down into a series of functions of two arguments and calculating each using the same processing machinery. (As we will see, this depends on availability of a computational process that is independent of the particular values of the task variables.) The dispositional-style way of implementing a function of n arguments is to dedicate processing machinery to each of the possible n-argument inputs. For example, GPT-3 can produce context-relevant outputs by dealing with what are in effect long sequences at input: a few examples of the task it is supposed to perform plus a query prompt (one shot or few shot learning: Brown et al., 2020).[1] Dealing with a long input string by means of a dispositional computation thus requires many more dedicated computational resources than does the classical computational solution. That is reflected in the large number of free parameters in the weight matrix of the DNNs which have been trained to perform real-world tasks.[2]

There are various ways of objecting to Gallistel & King's argument, but even if it is not as decisive as they think, their claim does point to a distinction that is evident in practical cases and has proven to be important. In fact, we can see these two ways of dealing with the past at work in meta-RL and deep episodic RL, respectively (Botvinick et al., 2019). A meta-RL

[1] Unlike meta-RL, the Transformer architecture of GPT-3 achieves meta-learning without recurrence.

[2] GPT-3 uses 175 billion free parameters (Brown et al., 2020).

system is trained on a range of different, related RL problems so that the trained weight matrix is equipped to deal with them all. A particular RL problem, consisting of certain reward contingencies associated with particular stimuli, is presented to the system as a sequence of inputs. The trained system is in a position to deal appropriately with this sequence without changing the weight matrix. For each element of the sequence, it "remembers" that input by going into a new state appropriate to what it should do with the next element in the sequence. It exemplifies a complex if-then disposition suited to dealing with short chains of inputs, its dynamics exhibiting rapid model-free reinforcement learning about these inputs (Wang et al., 2016; Duan et al., 2016; Wang et al., 2018). The success of meta-RL shows that dispositional computation has more mileage than Gallistel & King suppose. So too does the success of meta-learning in the Transformer architecture (Brown et al., 2020). But, these solutions also run up against the infinitude of the possible and the escalating cost of a solution which effectively dedicates computing machinery to a large number of possible input sequences.

Episodic deep RL solves the problem in a completely different way. It makes use of an explicit read-write memory – the principle whose practical efficacy was so striking to Gallistel & King in surveying extant computing machines. The fact that adding explicit memory to a DNN has enabled a new generation of AI systems to overcome the limitations of earlier models demonstrates that relying on this different computational principle does indeed bring practical benefits. It has its own costs – more processing is required to take a decision when many explicit memories have to be taken into account – but the recent results show that this is often a cost worth paying. DNNs with an explicit memory store have proven adept at tasks as diverse as text-based inference, navigating a transport network, designing novel chemical molecules, and playing a multi-player video game (Graves et al., 2016; Putin et al., 2018; Jaderberg et al., 2019; Chen et al., 2021).

The episodic deep RL described by Botvinick et al. (2019) accepts this cost, calling for all episodic memories to be recalled and assessed for similarity in deciding which to enter into the similarity calculation and hence how to act on a new input (Yang et al., 2020). However, being able to retrieve only memories with relevant contents (content-addressable memory) could make that computation simpler. Episodic memory retrieval in the brain appears to be content-based, using pattern completion in the hippocampus to recall just one (or a small number) of relevant memories from a large, sparsely represented set (Wixted et al., 2014; Botvinick et al., 2020). So there are ways that the computational demands inherent in making use of a large store of explicit memories can be mitigated.

In short, since the early success of 2012, the basic DNN architecture has been supplemented significantly to achieve astonishing levels of performance in an increasingly wide range of domains. The moral I venture to draw from these modifications is as follows. Inherent to dispositional computation there is a fundamental limitation, a limitation that is being overcome by architectures like deep episodic RL and the DNC of Graves et al. (2016). Elements of these models work in a completely different way, separating memory from computation – recording past experience by storing explicit memories rather than by adjusting the system's computational dispositions directly.

(4) Non-Content-Specific Computation

The second advantage that Graves et al. identify in their explicit memory system is that it deploys an algorithm that is independent of the particular values of task variables. That is a key insight. The DNC operates on retrieved memories by applying the same procedure to whatever data is retrieved. For example, when encountering a new state, it retrieves all the episodic memories that have already been stored and calculates the similarity of each to the current state. The similarity computation is performed the same way whatever the current input state is. Plus, in calculating similarity it performs the same procedure on each retrieved episodic memory. The algorithm acts in a way that is independent of the particular values of the task variables. That is a fundamental difference from the dispositional style of computation found in a basic ANN.

I have been using the term 'dispositional-style computation' as an intuitive shorthand, but the key point is not dispositions *per se* – any computational step realizes some disposition or other – but the specificity of the collection of dispositions in a trained DNN. So I will introduce the more precise notion of a 'content-specific' computation, to contrast with computations that operate in a uniform way whatever the input (e.g. an algorithm that computes the distance between any two vectors in activation space). Typically a computational step (transition between layers) in a trained basic DNN works as a special-purpose mapping from some states to others. It is somewhat like a tabular solution to an RL problem, where each possible state has its own entry in an exhaustive 'table' of possibilities (Botvinick et al., 2020).[3] Different regions of input space are mapped to disparate regions of output space. In many parts of state space small differences at input lead to large differences at output, as called for by the training set. At the level of distributed representations, few common processing principles are apparent.

To define the distinction between content-specific and non-content-specific computation, I need to appeal to the way contents are involved in making a computational step 'faithful' to content (Shea, 2018). In many cases, being 'faithful' is a matter of truth preservation: the outputs are guaranteed to be true if the inputs are true. In many other cases, a computational step is faithful to content because the conclusions are likely to be true if the premises are true. There is no guarantee that the outputs of an image classification network will be correct but, after training, they are likely to be. A content-specific computational step is one that is faithful to content only because of the specific contents represented at input and output: because inputs in region I_1 tend to be images of trains, say, and output O_1 represents the label "train"; ditto for all the other input-output dispositions trained into the network. The computations at work in a classical computational architecture are more general (as are those at work in the lower-level processing that implements a DNN architecture). They perform regular logical and mathematical functions. These are non-content-specific computations.

Our episodic deep RL systems apply non-content-specific computations to stored memories. That is the second advantage identified by Graves et al. Doing so is not a requirement for making use of explicit memories. For example, recurrent networks like LSTM

[3] The network as a whole does not function like a look-up table, since a function computed by an early processing step can be relied on multiple times in subsequent layers (Buckner, 2019).

hold online a temporary memory of past states (Hochreiter & Schmidhuber, 1997; Hassabis et al., 2017). Although not stored for the long term, that too is a case of explicit memory. However, in the way the memory is used it is treated as another content-specific input, rather than being input into a non-content-specific computation, as in the DNC. So use of explicit memory does not require the capacity for non-content-specific computation. It seems that it is the fact that the memory store in a DNC is non-parametric (indefinitely extensible) that encourages the system to deploy computations that are non-content-specific.

Looking at the other direction, does non-content-specific computation require explicit memory? No. Online inputs can also be treated in a general, non-content-specific way. However, although there is no necessary connection in either direction, in the episodic deep RL models we see that it is particularly useful to combine them: to deal with the past by storing explicit memories and then to compute with them using non-content-specific algorithms. That highlights the benefits that accrue to a system which can supplement content-specific computational dispositions with the capacity for non-content-specific computation.

The capacity for non-content-specific computation need not be hand-crafted into the system. It can itself be learnt, including by a DNN architecture (Garnelo et al., 2016; Garnelo & Shanahan, 2019). For example, a system can learn to tell which item performs a given role in a story even when the answer violates correlations observed in training (Chen et al., 2021). It is instructive that we see the use of an explicit memory store at work here too (in fact, the DNC).

It has long been recognised that variable binding is a key attribute of classical computational systems (Gallistel & King, 2010; Penn et al., 2008; Kriete et al., 2013; Bottou, 2014; Graves et al., 2016; Santoro et al., 2017). The distinction I am making is related, but is more general than variable binding. (It is also more general than the related idea of role-filler independence.) The computation of similarity in an episodic deep RL system is not performed on variables, but on specific episodic memories. The computation is performed in the same way whatever representation is taken as input, but the computation takes place over contentful inputs, not variables (representations whose content is yet to be specified).

To illustrate the contrast, consider the following two short chains of inference:

All sparrows fly, Abe is a sparrow, therefore Abe flies.
$(x - 1)^2 \equiv x^2 - 2x + 1$, $\therefore\ 2x \equiv x^2 + 1 - (x - 1)^2$

The first deploys a computation that is performed in the same way whatever representations are taken as input (sparrows, Abe, flying), but it is carried out on contentful representations not variables. It is non-content-specific, unlike the dispositional transformations from specific inputs to specific outputs trained into a standard DNN. The second, in addition to being non-content-specific, also makes use of variables. Graves et al. (2016) treat these two attributes as a package. I would argue, however, that it is the former which allows episodic deep RL to transcend the limitations of dispositional computation – its ability to deploy algorithms that are independent of the particular values of the representations it takes as input.

Now consider the related idea of role-filler independence. A system capable of role-filler independence can reason about certain relations no matter which items stand in that relation, that is, play that role. For example, if we use line connectors to represent relations of descent in family tree, there is no restriction on who can be represented in the role *parent of*. If a family tree is drawn using pictures of people, as in Figure 1, then the roles are not represented using variables. The pictures represent particular people. Contrast Darwin's famous diagram of a phylogenetic tree, which does use variables (Figure 2). Nevertheless, the representational scheme in Figure 1 exhibits role-filler independence since the relations are represented in a way which does not depend on any particular individual playing a given role. Non-content-specific computations can be performed over variables, but non-content-specific computations can also be performed over contentful representations, provided they exhibit something like role-filler independence.

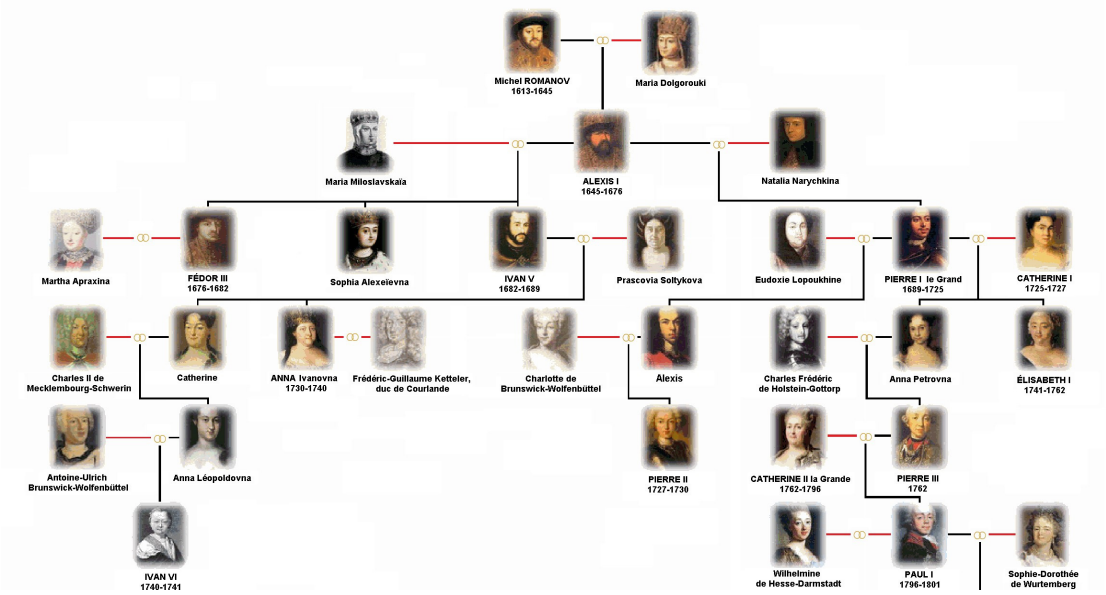

Figure 1. Family tree deploying specific representations (of people). (House of Romanov, from https://commons.wikimedia.org/)

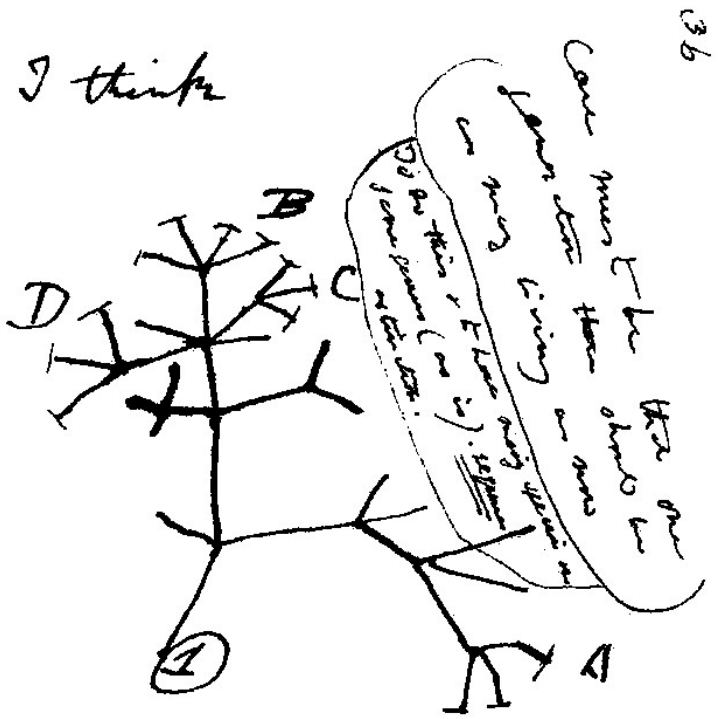

Figure 2. Tree diagram connecting variables. Sketch from Charles Darwin's First Notebook on Transmutation of Species (1837)

Penn et al. (2008) set out a computational model in which role-filler-independent reasoning in a 'physical symbol system' takes place over contentful representations, representations that have been learnt by a connectionist system. They argue that the physical symbol system is needed to account for the way that humans perform a range of relational, analogical and rule-based reasoning tasks. (We can set to one side the question of whether some non-human animals also have this capacity.) Their computational model combines the trained dispositions of a connectionist system with role-filler-independent reasoning in a classical symbolic system. Standardly, researches have focused on the difference in representational structure and compositionality between these two systems. What is striking for our purposes is that their model combines two different computational primitives: content-specific computations in the connectionist system and non-content-specific computations in the symbolic system.

In short, the key innovation is the ability to perform the same procedure on different data. What makes it the right procedure to use is independent of the particular values of the input. It is non-content-specific. This is quite unlike a disposition engine, where the appropriate computational transformation for a possible input I_1 precisely depends on the value of I_1. That is content-specific. Using variables is a particularly flexible way, but not the only way, of doing non-content-specific computations.[4]

[4] That raises the problem of how to achieve variable binding. Gallistel & King (2010) argue that we don't yet know how variable binding is implemented in the brain.

The way I have drawn the distinction may seem quite abstract. In the next section I make it more intuitive by illustrating it in a more familiar realm – in the way we reason with concepts. With the distinction more firmly in hand, I then return to DNNs and ask which computational steps in the models we have been considering are in fact content-specific, and which are not.

(5) Concepts Do Both

So far, I have argued that in the basic DNN architecture, the trained network implements a collection of content-specific computations. Recent developments have seen this supplemented with non-content-specific algorithms, in particular involving explicit representations of previously-encountered states and outcomes. The hybrid models have exhibited improved performance in a range of different tasks. In this section I want to make the content-specific / non-content-specific distinction clearer by showing how it plays out in a familiar domain, the domain of human concepts.

Concepts are so useful precisely because they can figure in both content-specific and non-content-specific computations, and can mediate between the two. Concepts allow us to categorise the objects and events we experience and generalise our experience to new instances. But they are more than just a device of generalisation. Perception involves representations that categorise stimuli and generalise to new samples, but they do not on their own amount to recombinable elements of thought (Eimas et al., 1971). Concepts are representations that figure in our reasoning and deliberate thought (Stanovich & Toplak, 2012; Camp, 2015), sub-propositional elements that are freely recombinable with one another to generate novel thoughts (Evans, 1982; Camp, 2004).

Concepts are involved in content-specific computations. The transition from seeing a certain arrangement of shapes, textures and colours to thinking *that's a cat* is probably well-modelled by DNNs (Yamins et al., 2014; Cichy et al., 2016). The thinker's CAT concept is wired up, as a result of experience, to a region of perceptual input space so that a transition from perception to thought happens relatively automatically. That is an appropriate transition to make. It won't often go wrong, in normal environments, but only because of the specifics of the representations involved. It works because of the specific contents at input and output (certain shapes, colours and textures; *cat*). Content-specific transitions also occur in the other direction, when our thoughts give rise to sensory imagery or expectations – from ROAST CHESTNUTS to a certain image and delicious smell.

Crucially, concepts are represented in such a way that they can be abstracted from any of the particular content-specific dispositions they are embedded in. We can reason with conceptual representations in a domain-general way. We saw an example with the inference about sparrows mentioned above (all sparrows fly, Abe is a sparrow, …). Specific concepts are involved, but the inference pattern works whatever concepts are involved. It is non-content-specific. Probabilistic reasoning can work the same way. Goodman et al. (2015) develop an ingenious model of the way abstract probabilistic reasoning using concepts can be combined with computations that draw on the specific content of those concepts.

The trick needed to get all of this to work is that a concept – the very same representation – can be involved in content-specific computations on some occasions and non-content-specific computations on others. Hummel and Holyoak (2003, 2005) give us a 'how possibly' computational model of how this could be achieved. LISA is a hybrid symbolic-connectionist system which performs syntactic inferences over 'role' representations within 'physical symbol system' (PSS). The basic idea is that categorisation and simulation occur in a perception-like system, which is then linked to a more symbolic-like representation (role representation) on which the system can do logical inference. Thus, the symbolic system is grafted on to a distributed connectionist architecture performing content-specific computations. A more recent model implements concepts as symbolic labels that are grounded in disentangled visual properties learnt by a DNN (Higgins et al., 2018). In either case, the key difficulty is to achieve dynamic role-filler binding without interfering with the role-filler independence required for reasoning within the symbolic system (Penn et al., 2008). In our terms, this is a matter of having content-specific computations in the ANN component connected to constituents of combinatorially-structured representations (concepts) over which non-content-specific computations are performed. This is precisely what the human conceptual system somehow manages to achieve. Indeed, Penn et al. (2008) argue that it is this ability which accounts for the special power of human cognition.

Another promising model of how this can be achieved is Chris Eliasmith's semantic pointer architecture (Eliasmith, 2013). Concepts are modelled as relatively abstract representations in higher layers of an ANN. Vector operations support modes of composition that approximate the syntactic structure of a sentence (e.g. distinguishing between agent and patient) (Eliasmith, 2013, pp. 121-162). But the relatively abstract representations also 'point' to the more specific perceptual contents from which they were abstracted. Computations that go on amongst the pointers are relatively non-content-specific, whereas unpacking a pointer into the less abstract representations to which it points is a content-specific operation. Quilty-Dunn has also argued that pointing is a good way to understand the relation between a concept and the information which it encodes (Quilty-Dunn, 2021). This philosophical model is quite different from Eliasmith's semantic pointer architecture, but shares with it the idea that concepts are a species of representation, pointers, over which computations can occur without calling on the information to which they point. From our perspective what is crucial about pointers is that they support non-content-specific computations.

I would add something further to these models. Concepts can be involved in content-specific transitions of a different sort, ones that take place between conceptual representations directly. Just as a model-free system can learn to recognise patterns in inferences (Lake et al., 2017), a thinker may be disposed to move directly from thinking *Fido is a dog* to *Fido barks*. If made directly, the inference does not depend on representing a minor premise, dogs bark (cf. Lea et al., 2005). Nor is it mediated via sensory representations (imagining a dog and hearing its barking) – although that too is possible. Another example: a disposition to make transitions from *x is whale* to *x is a mammal* may be directly 'wired in' to our concepts WHALE and MAMMAL (Murphy, 2002, p. 209). If we are disposed to make these kinds of transitions between conceptual representations directly, then they are quite different from non-content-specific transitions like the logical inference based on universal

quantification we saw above. Instead, these are content-specific computational steps, occurring within conceptual cognition.

Observing that concepts seem to be involved in computations of both kinds is just the start of the enquiry. It raises many questions. A pressing issue is the problem of relevance-based search – the (philosophical) 'frame problem' (Murphy, 2001; Samuels, 2010). When concepts are being manipulated in ways that are not specific to their particular contents, there are a very large number of other representations that they could be manipulated with. We saw a hint of this with the increasingly demanding operation required to retrieve similar states when an episodic deep RL system builds up an increasingly large store of remembered past states. With concepts, the problem is even more substantial, since their compositional power gives rise to an unbounded number of ways they can be composed into representations to enter into computations. Just highlighting the existence of non-content-specific computations in the human conceptual system does not yet tell us how these substantial problems are or should be overcome. That is a topic for another day.

(6) Which Transitions within a DNN are Content-Specific?

Having seen the idea of content-specific transitions illustrated in the case of concepts, we can now circle back and ask which computational steps in a DNN count as content-specific. A content-specific step is one that is faithful to content only because of the specific contents represented at input and output (§4). To a first approximation, the way the transition deals with one type of input is uninformative about how it should deal with other inputs. A system might be wired up to assume that dogs bark, but that tells us nothing about what if anything it should assume about parrots and flying, say.

Contrast a logical inference, where the way the system should deal with the input 'All men are mortal, Socrates is a man' is very informative about how it should deal with the input 'All meerkats are skittish, Oleg is a meerkat.' It should do something of the same form in response to both, producing the outputs 'Socrates is mortal' and 'Oleg is skittish', respectively. Are there transitions in a DNN that consist of content-specific expectations, wired in as a result of learning, rather than non-content-specific computations, like normalization and regularisation?

That is an empirical question. Furthermore, there is relatively little research on the inner workings of DNNs, compared to the huge amount of work on how to make DNNs better at task performance. Some results indicate, however, that learned transitions between layers are often content-specific. For example, analysis by Güçlü and van Gerven (2015) suggests that the convolutional layers of a DNN trained to classify images act so as to progressively transform distributions of contrast, blobs and edges, through contours, shapes and textures, into irregular patterns and object parts (see Figure 3). This suggests that one distribution of edges is mapped to one specific shape, a different distribution of edges to another. If so, these are content-specific transitions: more like a collection of dog $\rightarrow$ barks cases than like a non-content-specific operation applied uniformly to all inputs.

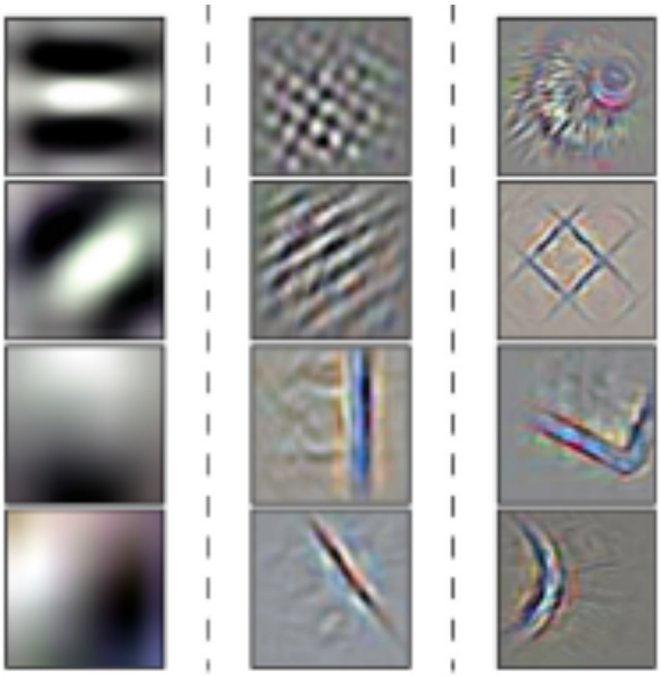

Figure 3. Example reconstructions of the internal representations of the first three convolutional layers of a DNN trained to categorise images (Güçlü and van Gerven 2015).

It is unsurprising that layer-to-layer transitions in a DNN turn out to be content-specific since the system is typically trained end-to-end to exhibit, in parallel, a large collection of content-specific dispositions. To a first approximation, which portions of input space should be mapped to the label 'train' tells us nothing about which portions of input space should be mapped to 'cherry'. However, given a suitable task and a sufficiently rich architecture, we might find that certain components are encouraged to discover non-content-specific computations. For example, there is evidence that, in transformer models of natural language processing, some components specialise in relatively non-content-specific operations, like attending to words in the object role in a sentence, or keeping track of relations of co-reference between elements of the input (Rogers et al., 2020; Henderson, 2020).

Content-specific computations may be the result of relatively domain-general features of the architecture and learning process. For example, convolution builds in the assumption that relevant information in an image is invariant under translation. The weight matrix of a convolutional step is common to many different spatial regions of the input array. That does not imply, however, that the transitions encoded by the resulting weight matrix are non-content-specific. What it does with one pattern of edges, say, may be quite different from what it does with another, even though it applies that wired-in assumption about edges to all spatial regions of the input array. Other features of the architecture and learning process may be specifically designed to filter out nuisance variation in the input (Buckner, 2018). That is a domain-general procedure for homing in on the aspects of the incoming information that are task relevant. But the weight matrix that is learnt as a result may then implement a content-specific transition: precisely because it reacts only to task-relevant information, and ignores the rest, it transforms inputs to outputs in a way that makes sense given the specific contents it is dealing with.

(7) Related Distinctions

ANNs have been distinguished from classical computational systems in many different ways in the existing literature. In this section I will contrast these existing distinctions with the one I have drawn above.

First off, modularity. DNNs work like Fodorian modules, processing domain-specific inputs in a fast, automatic way (Fodor, 1983). That is, however, a quite different distinction. The computations inside a module can be classical, thus non-content-specific. Indeed, Fodor conceived of modules as drawing on their own database of memories (Fodor, 1985, p. 3), naturally understood as explicit representations, and operating on them classically. And although the content-specific computations in a DNN are often encapsulated from information represented elsewhere in the system, that is not compulsory. In a recurrent network, neurons in one layer are affected by processing in higher layers. Representations elsewhere in the system can also act as 'top down' inputs, or affect intermediate layers. So although many DNNs do work like modules, modularity is not a necessary concomitant of dispositional computation. Nor does non-modular imply non-dispositional. Where encapsulation is broken, that can a matter or responding in a content-specific way to inputs from elsewhere in the system (as with LSTM, Hochreiter & Schmidhuber, 1997), or it may cross our Rubicon and introduce non-content-specific computations on explicit memories.

Second, compositionality. The ability to use representations with compositional structure is undoubtedly important. However, elements of compositionality can already be found in DNNs that compute according to content-specific principles. For example, they can learn to extract objects or features from images (Higgins et al., 2016; Eslami et al., 2016; Locatello et al., 2020). Conversely, stored explicit representations need not have compositional structure, although often they do. An episodic memory of a previously encountered state can simply be a record of an input vector, without recombinable compositional structure. In the episodic deep RL discussed by Botvinick et al. (2019) (Blundell et al., 2016; Pritzel et al., 2017), the state representations are stored together with a record of reward subsequently received, so the memory state does have some compositional structure, but this is nothing like the rich grammatical structure of a sentence.

Crucially for our purposes, computations performed on a compositionally-structured representation can be content-specific. We only cross our Rubicon when a structured representation is input into an algorithm that is independent of the particular values of task variables, as with the algorithm in episodic RL that takes state representations and their values as input and calculates their similarity to the current state. Sentences, the paradigmatic compositional representations, freely support these kinds of operations. We saw an example with the inference about sparrows above – replace 'sparrow' with 'robin' and the inference would still be performed in the same way. Reasoning from explicit representations is an effective solution to the challenge that purely dispositional computation faces when dealing with the past. Adding in compositionality vastly increases the possible computations that can be performed with explicit representations, but at the cost of a combinatorial explosion which can prove computationally intractable (Bottou, 2014).

Third, consider the distinction between model-free and model-based RL (Butlin, forthcoming). Un-supplemented, a deep RL system is model-free, learning the values of actions in world states without having any representation of the causal structure that relates action to outcome or world-state to world-state. I noted above that practical applications of DNNs often include a model-based component, like tree search over board positions in the game of Go (Silver et al., 2016; Botvinick et al., 2020). That component makes use of explicit representations of the structure of the environment, i.e. a model of the environment, which in some cases is learnt from experience (Vikbladh et al., 2017). Lake et al. (2017) characterise the hybrid system as being model-based. So model-based reasoning often does make use of explicit memories, and computes over them with non-content-specific algorithms.

On the other hand, model-free RL can also be – and outside DNNs standardly is – implemented in a non-content-specific algorithm, one which calculates and updates state-dependent action values based on reward feedback (Sutton & Barto, 1998; Dayan, 2014). These are stored explicitly and recalled when the same state is encountered again. So model-based reasoning is a further step, one which may call for, but is not entailed by, the capacity for non-content-specific computation. If we compare a content-specific model-free system with a non-content-specific model-based system, the pros and cons of each style of computation are laid bare: the dispositional content-specific system requires lots of samples and becomes resource-intractable unless operating in a restricted domain; for the model-based system, without short cuts, calculating what to do becomes computationally intractable.

The explicit-implicit distinction is relevant here. These terms mean different things to different theorists (e.g. conscious vs. not). Here, I take a representation to be explicit just in case it is a physical particular over which computation takes place (Shea, 2015, 2018). An explicit representation is realised by tokening a vehicle of content. Information retrieved from episodic memory is represented explicitly. Contrast a 'literal' in which the value of a constant like Π is embedded in a processor's input-output dispositions (Gallistel & King, 2010, p. 151). Some of what a system learns about its environment may be encoded in the form of the dispositions it has acquired to move between explicit representations. Such information is represented implicitly. Information that is stored implicitly, in a disposition to move from one tokened representation to another, can only be made use of by tokening the representations between which the disposition subsists (Shea, 2015, p. 81). The weight matrix of a trained DNN stores a huge amount of information implicitly, e.g. the information that inputs in the region I_1 of input space are likely to be pictures of a dog.

Content-specific computation leans heavily on implicit representation, but not everything is implicit. Samples are explicitly represented at input and output, and also in the state space of hidden layers (Yamins et al., 2014; Güçlü & van Gerven, 2015; Cichy et al., 2016). What is true is that, at the level of distributed representations in a basic DNN, memory of the past is entirely implicit – implicit in trained dispositions to move between inputs, internal states and outputs. Storing information implicitly in an input-output disposition calls for a disposition which is specific to the particular contents presented at input. It is by having a specific disposition that the system records that the environment it experienced was one way rather than another. Only when facts about the past are represented explicitly does the

possibility arise of doing non-content-specific computations on that information, thus of deploying algorithms that are independent of the particular values of task variables.

A distinction that is potentially closer is that between rules and associations. There is much controversy about what the difference comes to (Quilty-Dunn & Mandelbaum, 2019). Indeed, the venerable contrast between learning a rule and being trained by association looks simplistic when we consider model-based RL, which can learn about the causal structure of the environment by observing what are usually thought to be associations. The commonsense distinction between rules and associations probably runs together various properties which, while paradigmatically occurring together, can also dissociate (including the properties already mentioned in this section). One precisification, which drills down to a deep computational difference, is precisely the one advocated here. A rule, in computation or reasoning, is a procedure that takes representations as input and processes them in a way that is independent of their particular values. If the rule is that from 'x and y' one can infer 'x', then how that inferential step is performed does not depend on which propositional representations are found at positions x and y. Coming at the same issue from the other direction, a way of understanding what an association is – as a style of computation rather than a form of learning, so that it can be contrasted with a rule – is as a content-specific transition: a case where what has been learned is encoded in a disposition to move from certain specific inputs to certain specific outputs.

Another distinction in the same ballpark is that between symbolic and subsymbolic architectures. These handy labels in fact conjoin many of the forgoing distinctions. In symbolic computation experience is typically represented explicitly, e.g. stored in random access memory, representations have compositional structure, they can enter into models, and are processed using rules. Crucially for our purposes, symbolic architectures rely on non-content-specific algorithms. More than that, they often work with variables. A paradigmatic subsymbolic architecture, like an un-supplemented DNN trained by backpropagation of error to perform some task, has the contraries of all these properties.

Lake et al. (2017) argue that a deeper distinction than symbolic vs. subsymbolic is that between model-building systems and those which just do pattern recognition. Model-building can be done in an architecture that combines symbolic components (e.g. representation of objects, relations, agents and goals) with subsymbolic representation in a trained DNN. The pattern recognition component of their models operates according purely content-specific dispositional principles. What allows their hybrid system to transcend that, as we have seen, is not building a model *per se*, but a computational innovation: the ability to deploy algorithms that are non-content-specific. So I would argue that the content-specific / non-content-specific distinction is a better way to capture the contrast Lake et al. have highlighted.

(8) Conclusion

Recent years have seen impressive improvements in the capabilities of AI systems, with DDNs at the epicentre of an explosion of new tools. The practical project of building computational systems capable of tackling real-world problems can prove remarkably enlightening, revealing important truths about the nature of computation, and of intelligence.

('What I cannot create, I do not understand': Richard Feynman, quoted in Dennett (2017), ch. 15). Adding a store of explicit memories has overcome some of the limitations of the basic DNN architecture. Examining how that works, we have seen that the ability to perform non-content-specific computations is key. Thus, looking at these developments brings to the fore a distinction between two computational styles, the content-specific computations characteristic of a trained basic DNN, and the non-content-specific computations exemplified by the calculations performed on stored explicit memories.

There are, of course, many differences between the architecture of a basic ANN and that of a classical computer. The symbolic / subsymbolic contrast is something of a catch-all for many of these difference. The content-specific / non-content-specific distinction is more precise; also more fundamental than distinctions like pattern-recognition vs. model-building. Philosophical work on ANNs has largely focused on representational structure and the question of whether distributed representations are or can be compositional. In this paper, recent developments in neural network modelling, together with older arguments about computation, have shown us that it is equally important to examine computational processes. Only when we turn from structure to computation does it become clear that there is a significant difference between two fundamentally different styles of computation, the content-specific and the non-content-specific. The recent success of AI systems in solving real world problem is due, in no small measure, to their ability to combine content-specific with non-content-specific computational processes, trading on the complementary costs and benefits of each. This brings them closer to modelling the distinctive mix of capacities involved in human cognition and represents another step on the road towards artificial general intelligence.

Acknowledgements

I am particularly grateful to the following for helpful questions, comments and suggestions: Ali Boyle, Patrick Butlin, Rosa Cao, Shamil Chandaria, Randy Gallistel, Raphaël Millière and Dimitri Mollo; audiences at the Deep Learning Reading Group at Columbia, the Cognition Academy at the Max Planck School of Cognition in Berlin and the IP Lab Meeting; and two reports from *Mind & Language*. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No. 681422 (MetCogCon).

References

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Banino, A., Badia, A. P., Köster, R., et al. (2020). Memo: A deep network for flexible combination of episodic memories. *arXiv preprint arXiv:2001.10913*.

Blundell, C., Uria, B., Pritzel, A., et al. (2016). Model-free episodic control. *arXiv preprint arXiv:1606.04460*.

Bottou, L. (2014). From machine learning to machine reasoning. *Machine learning, 94*(2), 133-149.

Botvinick, M., Ritter, S., Wang, J. X., et al. (2019). Reinforcement Learning, Fast and Slow. *Trends Cogn Sci, 23*(5), 408-422. doi:10.1016/j.tics.2019.02.006

Botvinick, M., Wang, J. X., Dabney, W., et al. (2020). Deep reinforcement learning and its neuroscientific implications. *Neuron, 107*(4), 603-616.

Brown, T. B., Mann, B., Ryder, N., et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Buckner, C. (2018). Empiricism without magic: Transformational abstraction in deep convolutional neural networks. *Synthese, 195*(12), 5339-5372.

Buckner, C. (2019). Deep learning: A philosophical introduction. *Philosophy Compass, 14*(10), e12625.

Butlin, P. (forthcoming). Cognitive Models are Distinguished by Content, Not Format. *Philosophy of Science*.

Camp, E. (2004). The generality constraint and categorial restrictions. *The Philosophical Quarterly, 54*(215), 209-231.

Camp, E. (2015). Logical Concepts and Associative Characterizations. In E. Margolis & S. Laurence (Eds.), *Conceptual mind: New directions in the study of concepts.* (pp. 591-621). London / Cambridge MA: MIT Press.

Chen, C., Lu, Q., Beukers, A., et al. (2021). Learning to perform role-filler binding with schematic knowledge. *PeerJ, 9*, e11046.

Cichy, R. M., Khosla, A., Pantazis, D., et al. (2016). Comparison of deep neural networks to spatio-temporal cortical dynamics of human visual object recognition reveals hierarchical correspondence. *Scientific reports, 6*, 27755.

Dayan, P. (2014). Rationalizable irrationalities of choice. *Topics in cognitive science, 6*(2), 204-228.

Dennett, D. C. (2017). *From bacteria to Bach and back: The evolution of minds*: WW Norton & Company.

Duan, Y., Schulman, J., Chen, X., et al. (2016). RL2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*.

Eimas, P. D., Siqueland, E. R., Jusczyk, P., et al. (1971). Speech perception in infants. *Science, 171*(3968), 303-306.

Eliasmith, C. (2013). *How to build a brain: A neural architecture for biological cognition*. Oxford: OUP.

Eslami, S., Heess, N., Weber, T., et al. (2016). Attend, infer, repeat: Fast scene understanding with generative models. *arXiv preprint arXiv:1603.08575*.

Eslami, S. A., Rezende, D. J., Besse, F., et al. (2018). Neural scene representation and rendering. *Science, 360*(6394), 1204-1210.

Evans, G. (1982). *The Varieties of Reference*. Oxford: O.U.P.

Floridi, L., & Chiriatti, M. (2020). GPT-3: Its Nature, Scope, Limits, and Consequences. *Minds and Machines*, 1-14.

Fodor, J. A. (1983). *The Modularity of Mind*: MIT Press.

Fodor, J. A. (1985). Précis of *The Modularity of Mind*. *Behavioral and Brain Sciences, 8*.

Fodor, J. A., & McLaughlin, B. (1990). Connectionism and the problem of systematicity: Why Smolensky's solution doesn't work. *Cognition, 35*, 183-204.

Fodor, J. A., & Pylyshyn, Z. W. (1988). Connectionism and Cognitive Architecture: A Critical Analysis. *Cognition, 28*, 3-71.

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences, 3*(4), 128-135.

Gallistel, C. R. (2006). The nature of learning and the functional architecture of the brain. *Psychological science around the world, 1*, 63-71.

Gallistel, C. R. (2008). Learning and representation. *Learning and memory: A comprehensive reference, 1*, 227-242.

Gallistel, C. R., & King, A. P. (2010). *Memory and the computational brain: Why cognitive science will transform neuroscience*: John Wiley & Sons.

Garnelo, M., Arulkumaran, K., & Shanahan, M. (2016). Towards deep symbolic reinforcement learning. *arXiv preprint arXiv:1609.05518*.

Garnelo, M., & Shanahan, M. (2019). Reconciling deep learning with symbolic artificial intelligence: representing objects and relations. *Current Opinion in Behavioral Sciences, 29*, 17-23.

Gershman, S. J., & Daw, N. D. (2017). Reinforcement learning and episodic memory in humans and animals: an integrative framework. *Annual review of psychology, 68*, 101-128.

Goodman, N. D., Tenenbaum, J. B., & Gerstenberg, T. (2015). Concepts in a probabilistic language of thought. In E. Margolis & S. Laurence (Eds.), *The Conceptual Mind : New Directions in the Study of Concepts*. Cambridge, MA: MIT Press.

Graves, A., Wayne, G., Reynolds, M., et al. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature, 538*(7626), 471-476.

Güçlü, U., & van Gerven, M. A. (2015). Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream. *Journal of Neuroscience, 35*(27), 10005-10014.

Hassabis, D., Kumaran, D., Summerfield, C., et al. (2017). Neuroscience-inspired artificial intelligence. *Neuron, 95*(2), 245-258.

Henderson, J. (2020). The unstoppable rise of computational linguistics in deep learning. *arXiv preprint arXiv:2005.06420*.

Higgins, I., Matthey, L., Glorot, X., et al. (2016). Early visual concept learning with unsupervised deep learning. *arXiv preprint arXiv:1606.05579*.

Higgins, I., Sonnerat, N., Matthey, L., et al. (2018). Scan: Learning hierarchical compositional visual concepts. *International Conference on Learning Representations*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation, 9*(8), 1735-1780.

Hummel, J. E., & Holyoak, K. J. (2003). A symbolic-connectionist theory of relational inference and generalization. *Psychological Review, 110*(2), 220.

Hummel, J. E., & Holyoak, K. J. (2005). Relational reasoning in a neurally plausible cognitive architecture: An overview of the LISA project. *Current Directions in Psychological Science, 14*(3), 153-157.

Jaderberg, M., Czarnecki, W. M., Dunning, I., et al. (2019). Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science, 364*(6443), 859-865. doi:10.1126/science.aau6249

Jumper, J., Evans, R., & al., e. (2020). *High Accuracy Protein Structure Prediction Using Deep Learning*. Paper presented at the Fourteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstract Book). https://predictioncenter.org/casp14/doc/CASP14_Abstracts.pdf

Kriegeskorte, N., & Diedrichsen, J. (2019). Peeling the onion of brain representations. *Annual review of neuroscience, 42*, 407-432.

Kriete, T., Noelle, D. C., Cohen, J. D., et al. (2013). Indirection and symbol-like processing in the prefrontal cortex and basal ganglia. *Proceedings of the National Academy of Sciences, 110*(41), 16390-16395.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 25* (pp. 1097-1105). New York: Curran Associates, Inc.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., et al. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences, 40*.

Lea, R. B., Mulligan, E. J., & Walton, J. L. (2005). Accessing distant premise information: How memory feeds reasoning. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 31*(3), 387.

Liu, Y., Mattar, M., Behrens, T., et al. (2020). Experience replay supports non-local learning. *bioRxiv*.

Locatello, F., Weissenborn, D., Unterthiner, T., et al. (2020). Object-centric learning with slot attention. *arXiv preprint arXiv:2006.15055*.

Madarasz, T., & Behrens, T. (2019). *Better transfer learning with inferred successor maps.* Paper presented at the Advances in neural information processing systems.

Marcus, G. (2018). Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*.

Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529-533.

Morgan, A. (2020). Against neuroclassicism: On the perils of armchair neuroscience. *Mind & Language*.

Murphy, D. (2001). Folk psychology meets the frame problem. *Stud. Hist. Phil. Biol. & Biomed. Sci., 32*(3), 565-573.

Murphy, G. L. (2002). *The Big Book of Concepts*. London / Cambridge, MA: MIT Press.

Penn, D. C., Holyoak, K. J., & Povinelli, D. J. (2008). Darwin's mistake: Explaining the discontinuity between human and nonhuman minds. *Behavioral and Brain Sciences, 31*(2), 109-130.

Pritzel, A., Uria, B., Srinivasan, S., et al. (2017). Neural episodic control. *arXiv preprint arXiv:1703.01988*.

Putin, E., Asadulaev, A., Ivanenkov, Y., et al. (2018). Reinforced adversarial neural computer for de novo molecular design. *Journal of chemical information and modeling, 58*(6), 1194-1204.

Quilty-Dunn, J. (2021). Polysemy and thought: Toward a generative theory of concepts. *Mind & Language, 36*, 158–185.

Quilty-Dunn, J., & Mandelbaum, E. (2019). Non-Inferential Transitions. In T. Chan & A. Nes (Eds.), *Inference and Consciousness* (pp. 151-171). New York, NY: Routledge.

Rae, J. W., Hunt, J. J., Harley, T., et al. (2016). Scaling memory-augmented neural networks with sparse reads and writes. *arXiv preprint arXiv:1610.09027*.

Rogers, A., Kovaleva, O., & Rumshisky, A. (2020). A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics, 8*, 842-866.

Samuels, R. (2010). Classical computationalism and the many problems of cognitive relevance. *Studies in History and Philosophy of Science Part A, 41*(3), 280-293.

Santoro, A., Raposo, D., Barrett, D. G., et al. (2017). *A simple neural network module for relational reasoning.* Paper presented at the Advances in neural information processing systems.

Senior, A. W., Evans, R., Jumper, J., et al. (2020). Improved protein structure prediction using potentials from deep learning. *Nature, 577*(7792), 706-710.

Shea, N. (2007). Content and its vehicles in connectionist systems. *Mind & Language, 22*(3), 246–269.

Shea, N. (2015). Distinguishing Top-Down From Bottom-Up Effects'. In S. Biggs, M. Matthen, & D. Stokes (Eds.), *Perception and Its Modalities* (pp. 73-91). Oxford: OUP.

Shea, N. (2018). *Representation in cognitive science*. Oxford: Oxford University Press.

Silver, D., Huang, A., Maddison, C. J., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature, 529*(7587), 484-489.

Smolensky, P. (1988). On the Proper Treatment of Connectionism. *Behavioral and Brain Sciences, 11*, 1-74.

Stanovich, K. E., & Toplak, M. E. (2012). Defining features versus incidental correlates of Type 1 and Type 2 processing. *Mind & Society, 11*(1), 3-13.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*: The MIT press.

Vikbladh, O., Shohamy, D., & Daw, N. (2017). *Episodic contributions to model-based reinforcement learning.* Paper presented at the Annual conference on cognitive computational neuroscience, CCN.

Wang, J. X., Kurth-Nelson, Z., Kumaran, D., et al. (2018). Prefrontal cortex as a meta-reinforcement learning system. *Nature Neuroscience, 21*(6), 860-868.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., et al. (2016). Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*.

Wayne, G., Hung, C.-C., Amos, D., et al. (2018). Unsupervised predictive memory in a goal-directed agent. *arXiv preprint arXiv:1803.10760*.

Werker, J. F., Gilbert, J. H., Humphrey, K., et al. (1981). Developmental aspects of cross-language speech perception. *Child development*, 349-355.

Wixted, J. T., Squire, L. R., Jang, Y., et al. (2014). Sparse and distributed coding of episodic memory in neurons of the human hippocampus. *Proceedings of the National Academy of Sciences, 111*(26), 9621-9626.

Yamins, D. L., Hong, H., Cadieu, C. F., et al. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences, 111*(23), 8619-8624.

Yang, D., Qin, X., Xu, X., et al. (2020). Sample Efficient Reinforcement Learning Method via High Efficient Episodic Memory. *IEEE Access, 8*, 129274-129284.