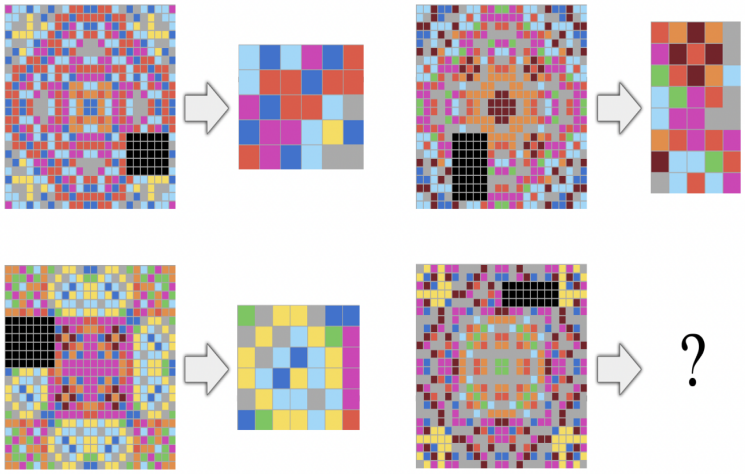
Intelligence and Generalization

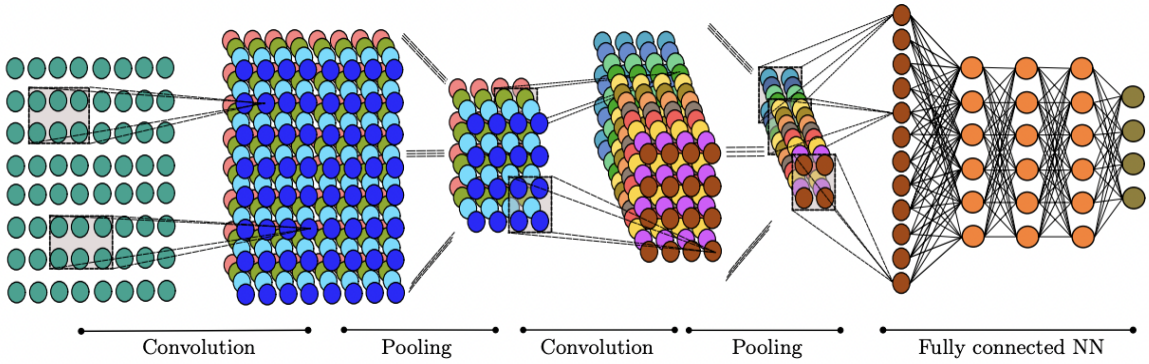


Intelligence and the mind are often intertwined, and it is something we construct throughout our life which happens in numerous stages-from a child to adult. Mind is sought out as a multi-scale hierarchy of temporal prediction modules, which gives rise to many exciting notions such as that of a compression function or prediction function, i.e., there are multiple scales of processing in our brains. Brains are much like a sparse access memory wherein we can dynamically retrieve little bits of information or experience very precisely as and when required.

Abstraction is the engine through which we produce generalization. To understand the complexity of the brain, we can start with the miniature question of what Intelligence is at all. Intelligence is the sensitivity to abstract analogies; that is all there is to it. Intelligence is the efficiency with which we can acquire new skills at tasks we previously didn't know about or prepare for. It is about how well and efficiently we can learn new things (never seen before)- It is not about the skill itself or what we know; it is at its all about efficiency and adaptation. Intelligence is the ability to face future situations given the things we have seen in the past. The way to do it is by abstraction, which can construct some templates or algorithms on the fly that will eventually help explain the novel tasks. It can also handle multiple instances, giving rise to abstraction and generalization capabilities. Revisiting the first principles, the human mind follows both Type1 and Type2 thinking, defining Intelligence as the efficiency of transforming prior information and experience into task solutions. Thus, we need to measure and maximize generalization to progress toward AI.

Intelligence is a system embodied in humans (brain in a body acting in an environment) and hence is highly specialized- it is our ability to convert experience into future skill. Generalization power is the ability to mine previous experience to make sense of future novel situations as discussed in Chollet[2019]. It describes a knowledge differential and characterizes the ratio between known information (prior) and the space of possible future situations. Thus, it can be thought of as our ability to deal with "novelty and uncertainty." If a system in a new environment cannot deviate from what it is encoded to do and trained to do, then it is not intelligent. It should be able to adapt to the new environment efficiently, and cognitive primitives lay the core building blocks for such complex hierarchical computations acquired through millions of years of human evolution. Intelligence is more a process, and the output of that process is a skill. The source of Intelligence and the entity capable of such a process can be coined as intelligent. The discussion of the mind is thus fascinating, and the 'On the Measure of Intelligence' paper aptly captures numerous formulations. I highly recommend you look at it.

Deep Learning

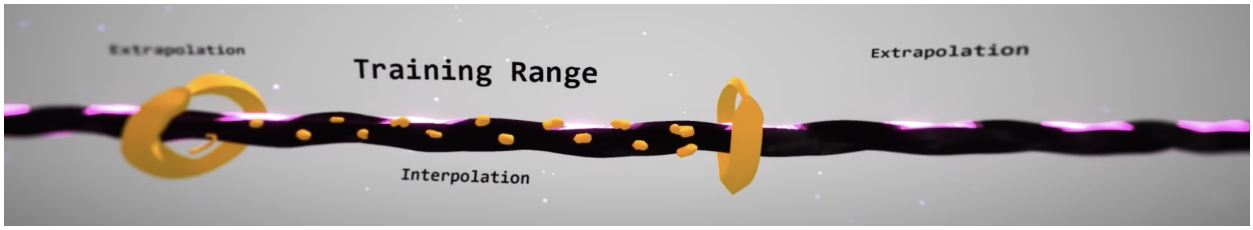


Deep Learning operates in high-dimensional spaces, and things can quickly start to go haywire here. Neural networks are a family of continuous parameterized functions with differential layers stacked on top of one another capable of learning smooth, continuous functions or manifolds of data (disentangling those manifolds through complex mathematical functions)-glorified curve fitting.

Deep Learning is only suited for continuous problems when the data is interpolative and has a learnable manifold and a dense sampling of the entire input space (natural and perceptual data falls in the category of learnable and smooth manifolds). But the real world doesn't have such a static distribution (universe has chaos). Intelligence requires that you adapt to novelty without the help of the engineer who wrote the system, and neural networks sometimes fall well short of that. The most common task of classification is just the mere separation of a bunch of tangled manifolds, and that too only if the surface is learnable. The only way these models generalize is via interpolation-pattern matching and local generalization.

The gradient descent method will most certainly fail to learn when there are discontinuities in its manifolds and thus resort back to memorizing the data. Interpolation on the learned manifold would seem like extrapolation on the original input space. Even if we were to go outside the convex hull of the data even by a tiny bit, then technically, this results in "extrapolation" in high-dimensional spaces. Most of machine Learning(linear regression, logistic regression, SVM) is mere interpolation, but in high-dimensional space, there is no such thing as interpolation; it's purely extrapolation. As it turns out, deep learning is primarily helpful for a vivid range of domains because most things are interpolative by nature (interpolation can mean not just linear but can also mean polynomial, piecewise constant interpolation), and natural data do have learnable manifolds, and deep learning does exceedingly well in acquiring skills for these constructs and also on perception data. Deep Learning models are high-dimensional curves with some constraints (implicit regularization) on their structure given by inductive priors (ex:- CNN's have implicit priors for processing matrix data-images).

The curve has enough parameters to fit almost anything. If we let our model train enough it will simply memorize almost everything, i.e., they are big interpolators of arbitrary manifolds (hacky inductive priors notwithstanding). Because of SGD, as the neural net is trained for more number of iterations, at some point the learned manifold will approximate the original manifold between the underfitting and overfitting regime. At this point, we can make sense of novel points by interpolating on this manifold.



Thus, the power of the model to generalize is actually of the structure of the data and the gradual process of the stochastic gradient method method (SGD) rather than any property of the model itself. Deep neural nets are locally sensitive hash tables and only efficiently use some kinds of data well. Interpolation is the origin of true generalization in deep learning. Interpolation on the native data domain is useless. We need to pull some useful information out of the data.

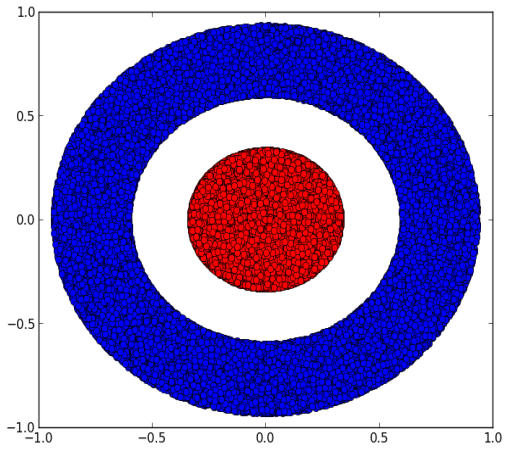
The model demonstrates poor performance on even the simplest of its kind, like the scalar identity function ($f(x)=x$), when we go outside the training data because nothing about the manifold (think of a manifold as a string that goes on forever) is known. The neural nets memorize the space of numbers in their constrained range and cannot go beyond the convex hull of the training data-hardly any generalization at all. But, this is not true for perceptual problems and explains why image models suffer greatly in drawing straight lines. Thus, all perception problems i.e. tasks that involve levels of human intuition (in fact, almost all perceptual problems are entirely interpolative) is probably a good fit for deep learning and those with abstraction and reasoning demands are probably a bad fit for modern deep learning architectures.

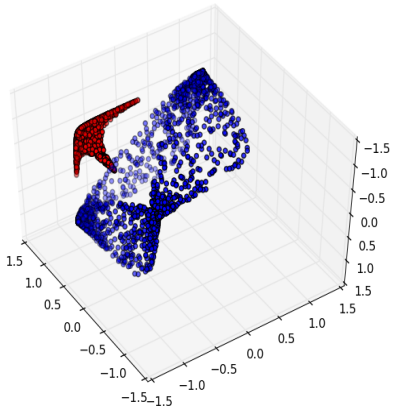
To elaborate on these lines, learnability is a long-standing problem in connectionism-type models that are optimized using gradient information, such as fitting a manifold using the gradient descent method when in fact, the structure I am trying to fit is too discrete (lots of discontinuities) then neural nets will not work at all, and at best it will memorize it. This is the dichotomy of connectionism vs. symbolism methods, abusing DL to perform the latter. Also, finite NN's are not turing complete because programs that run on a turing machine have unbounded memory and time computation which is impossible to encode in any finite NN's right.

ML systems typically approximate functions by relating input variables to output variables-curve fitting. Brittleness world both ways depending on the problem type itself. For example, program synthesis would be highly brittle in classifying cats vs. dogs or even the mnist images, for that case. At the same time, deep learning would be extremely brittle in predicting the digits of pi or prime numbers or sorting a list. For training a deep learning model to predict the next prime number, models will at best, memorize the trained data because the space of prime numbers is not interpolative at all, and the deep learning model will have zero generalization power.

Every single thought in our minds is not simply one or another; instead, it's a combination of Type1 and Type2 thinking. They are enmeshed in everything we think and everything we do. Even our reasoning is guided by intuition which is interpolative in nature. Abstraction is the key to generalization and the way we perform abstraction is the key to generalization and the way we perform abstraction in continuous vs. discrete space; there is a need to find analogies and those analogies are different in both those spaces. Program synthesis allows broad generalization from just a few examples; they build a library of learned concepts and rich primitives, enabling them to perform a horizon of numerous tasks simultaneously, which is a significant deviation from traditional machine learning. Here, rather than trying to interpolate between the training examples, we are constructing an entire program search space and testing if it fits our training data (popular and recent methods like the dreamcoder use a hybrid approach wherein they use neural nets or a neural engine

as a recognition model to guide the search in this discrete program space, and this is achieved by maximizing the posterior over all combinatorial sets of programs present in that space).

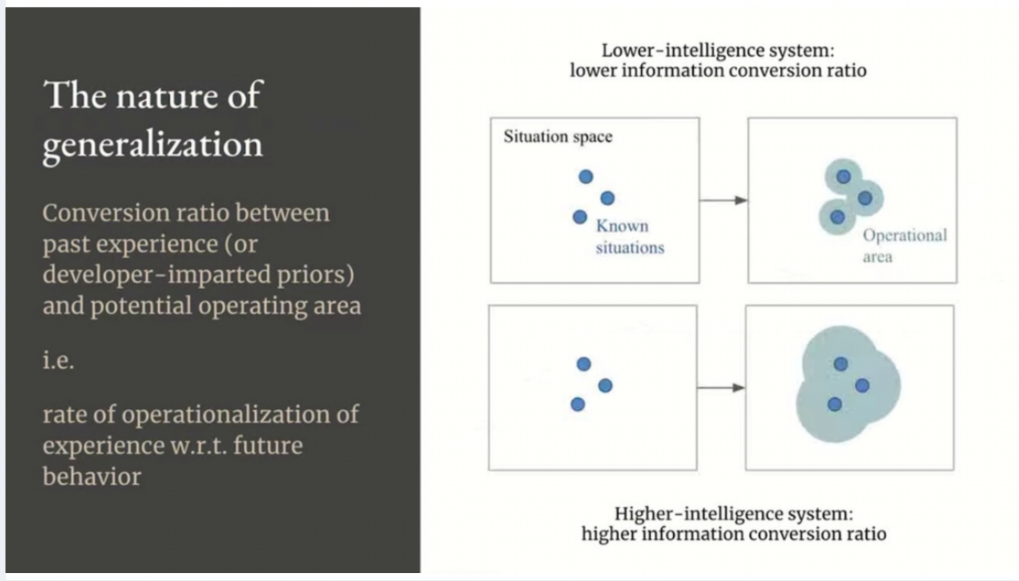




Program synthesis based on discrete search and the DSL can generalize very well to discrete tasks with a few examples (neural nets would need thousands of examples even to fit in basic primitives and learned concepts like addition, multiplication, for loops, while loops and other logical expressions except otherwise baked into the model explicitly as a prior, like an LSTM has a for loop primitive) and the solution is going to be exact because it is the exact discrete algorithm and not a continuous ambiguous model, so it doesn't have any glitched in it's outputs and will be correct and will be lightweight like the transformer model. Neural networks can be thought of as a kind of locality-sensitive hash table; we can cheat with access to massive amounts of data.

As demonstrated with DOTA, the gameplay results were often very brittle as it failed to adapt to even the slightest variations when put up against it (simple rule changes), while humans could easily do this on the fly by combining abstraction and prior experience. Thus, it is always possible to create shortcuts to solve a given task without featuring Intelligence; they reduce novelty and uncertainty via the injection of information as priors and training data. Also, an autoregressive language model such as GPT-3 (statistical language model) hasn't expanded its knowledge of the world, and it is not learning any algorithms on the fly (zero-shot learning).

It has already learned the glitchy representations of the existing tasks during its training, and it is ineffective in tasks requiring broad generalization capabilities like predicting the nth digit of pi. A Turing machine can perform this well due to its unbounded access to memory and time, while a neural network at best approximates these unbounded algorithms but doing so will introduce glitches. Neural nets are finite state machines (FSM) and just like finite state machines they can be augmented with unbounded memory and iteration to yield a turing complete computational model. This indeed states that deep learning models can embed algorithms given sufficient exposure to the training data but can not synthesize novel algorithms that represent a pattern not present in the training data. This is one of the reasons for GPT-3 to be completely ineffective on the ARC challenge, because it has no notion about the world. Since it has been exposed to tons and tons of data, it simply resorts to memorizing these parameters and applying these to new data at inference time. Still, they don't necessarily have any sort of understanding capabilities beyond pattern matching and are non-deterministic in their predictions. System-centric generalization is the ability to adapt to situations not previously encountered by the system/agent itself. Developer-aware generalization is the ability to adapt to situations that could not be anticipated by the creators of the system (unknown unknowns).



Humans can acquire new skills and simulate new knowledge in almost all situations, even with a small amount of data; the kaleidoscope effect, which is surrounded by isomorphisms, like a kaleidoscope, created a remarkable richness of complex patterns from a tiny bit of information. True Intelligence is about casting previous knowledge and information into new information, which are indeed novel in nature. We can cast previous experience into many new types of situations and generate new models on the fly by extrapolating from abstract and prior knowledge. We can also deal with the combinatorial complexity of reality and integrate our knowledge about the world into learning. Humans use a lot of abstract concepts when making these temporal prediction sequences, but AI has relatively very constrained domain adaptation, i.e., there is no adaptation to novelty. There is a need for systems to extrapolate beyond the training data and an innate need for systems capable of better representations, inductions of cognitive models, and a heterogeneous architecture with specialized modules. This requires reasoning as a first-class citizen. The universe is not all about chaos; it has an intrinsic structure to it where many, many things are pretty similar and analogous to each other in ways that go beyond our imagination.

The brain is for long thought as a problem-solving algorithm. Still, it is a big spawn which means that exposure to any given environment will absorb experiences from that given environment and thus enables reasoning capabilities in future novel situations. Francois Chollet uses the notion of algorithmic information to precisely quantify the conversion ratio, which characterizes the relationship between our known priors and experience priors and the space of possible future situations. Also, the notion of optimizing for task-specific skills will have the model taking all sorts of shortcuts over every dimension not captured by our metric. Thus, if a system takes shortcuts, it is not intelligent because skill is orthogonal to generalization power (modern approaches do not display generalizable cognitive abilities but simulate this deception by using more training data and taking shortcuts).

References

1. Chollet, François. "On the measure of intelligence." *arXiv preprint arXiv:1911.01547* (2019).
2. Clark, Peter, et al. "Think you have solved question answering? try arc, the ai2 reasoning challenge." *arXiv preprint arXiv:1803.05457* (2018).