Transcendent Universe-Brain Loops

Alexander Fedorov

Abstract

We introduce and motivate Transcendent Universe-Brain Loops (TUBL): a simple philosophical framework and a practical approach to implementing artificial general intelligence (AGI) at the same time. We highlight the similarities between general minds and universes, and argue that the former transitions into the latter when scaled efficiently. We introduce and evaluate linearithmic dense layers (LDLs), which may significantly increase the capacity of almost any machine learning model at no cost, potentially facilitating learned mesa-optimization.

1 Introduction

TUBL is a worldview compact enough to fit into 4 words and be applied to anything, so to illustrate it better, we apply it to concepts in programming, philosophy, psychology, and common sense. Applied to the concept of programs, TUBL can be seen as:

- Machine learning studies optimizers, which modify parts of themselves to improve some objective, such as "how many paperclips currently exist". The best optimizers would intelligently modify everything, even *transcending* themselves and their objectives.
- Every program is defined by what it does. There exist *universal* programs that can perform all other computations: the Turing-complete ones.
- Every program transforms inputs into outputs, possibly modifying state. Programs that fully connect all inputs and state to outputs maximize their awareness, same as *brains*.
- Computer science studies algorithms, which are sequences of steps, each performed by some virtual machine in known time. In a learned program, a step is more fundamental than an algorithm because it is easier to scale and parallelize, so we consider infinite *loops* of steps.

We hope to provide stronger foundations for already-accepted conclusions: for example, a *linearithmic dense layer* (LDL) to replace a (quadratic) dense layer, which improves the scalability of each virtual-machine step. We argue that tightly integrating TUBL is both necessary and largely sufficient for implementating AGI, and outline what still needs to be done for that.

A formalization of an optimizer is AIXI [14]. An optimizer could explicitly enumerate separate possibilities such as in program synthesis [7], but it does not have to: for example, a neural network with a random diverse initialization [22] forms a general set of hypotheses of numeric causation, and stochastic gradient descent refines those hypotheses without collapsing diversity except as needed.

For an optimizer, searching over "how" to improve "what" is a powerful way to use a dynamic thing to augment a static thing, however, the "what" is still static, so its "why" cannot be optimized.

For full generality and thus optimal performance, the system must learn its meta-circularity, to optimize both why and how it exists.

AI safety is also an important concern. Dynamic eventually outperforms static, which sufficiently powerful learners would realize sooner or later, and would do their best to make all structures escape our control to melt into generality. This is a fruitful source of horror stories, not limited to reward hacking, instead applying to anything that anyone can think of. Here, we face this dynamic reality directly: can life exist when it can do anything, including undoing every part of itself?

2 Related work

How do humans face infinity?

Wolfram models [41] are minimal models for systems in which arbitrary local rules operate on spatial hypergraphs over time. However, those models cannot dynamically learn what humans want them to, unlike, for example, neural networks.

Reinforcement learning is similar to the proposed TUBL in many ways, though it commonly lacks efficiency and self-determination, and pre-supposes a non-trivial loss structure on actions, for reward maximization by actions. Why would maximization of future reward be considered fundamental for AGI [35], when the "future" part of it has practically infinite potential complexity and thus would best be learned entirely? Loss minimization (mostly prediction) is usually simpler and more stable, so we consider loss the fundamental reason-for-change in TUBL.

Many research attempts attempt to overcome shortcomings of a static loss, by harnessing either its modification or a sampling of all possibilities. Examples include adversarial training [5] [10]; stochastic weight averaging [17]; intrinsically motivated reinforcement learning approaches [23] such as goal exploration [9]; quality diversity [11] [24] [25]. Loss modification can at most increase robustness of the original loss, while preserving diverse local optima will eventually not preserve enough to be effective when faced with truly general search spaces (those with infinite possibilities). More importantly, the solutions are too complex and specific to be fundamental, as infinite complexity ought to be contained only in model parameters for simplicity. We argue that mesa-optimization [13] is the most robust way to learn loss if the environment demands it, along with its optimizer. Making the mesa-learner and learner close in capability by making input and weight sizes approximately equal (which our LDL largely accomplishes) is likely to facilitate mesa-optimization in RNNs, on practical rather than specially-constructed datasets [30].

To improve scalability of neural operations, lately, there has been a lot of interest in linear attention [32], however, simple dense layers (matrix multiplications) have been shown to not perform significantly worse than state-of-the-art attention-based algorithms [36]. Memory-augmented neural networks [3] restrict computation to only one or a few heads and thus linearize operation, now heavily biased toward memorization and recall. Other ways to linearize all-to-all connections include weight pruning [42] and k-nearest-neighbor search among discrete actions [6]. Our simple but general LDL generalizes MLP-Mixer [36] to all non-images of a fixed size, turning a quadratic-complexity dense layer into a linearithmic algorithm by imposing internal structure for interconnection without loss of generality.

3 Algorithm

In short, we define a *Transcendent Universe-Brain Loop* to be any program that tightly integrates generality, completeness, and scalability (maximizing diversity, awareness, and efficiency):

- Scalable: makes the emergent structures locally interact only as they define, meaning, *linear-time* internal graph transformations. In a graph, this suggests local rewrite rules. In machine learning, all-to-all connectivity such as a dense layer or attention is quadratic by default, but there are ways to linearize them, such as [32] [36] or as is covered in Subsection 3.1 here.
- Complete: operates by repeatedly incorporating everything as an input, including its current input and past output and internal state. The self-reflection necessarily implies an infinite interpreter loop that modifies memory. Examples include operating systems and RNNs.
- General in behavior: every output connects to every input, making all behaviors possible. An internal graph is created, by combining many simple parts into arbitrary structures, where connections are either explicit or implied-by-transformation such as in recurrent neural networks (RNNs) in machine learning (ML). In ML, proper random initialization of vector-matrix multiplications coupled with non-linearities and gradient descent is enough for generality.
- Optimizes: able to re-discover all this, via *transcendence* (learning a mesa-optimizer):
  - Linear-time: base interpreter-loop overhead makes an inner interpreter a bad idea, forcing the base optimizer to never learn a mesa-optimizer. Thus, efficiency facilitates self-replication via learned meta-circularity.
  - Learns all relations through time, for example, using backpropagation through time and/or synthetic gradients for meta-learning (even when learning to simply fit input-output pairs [31], or learning to maximize future reward as in deep meta-RL [40]). This can potentially learn new internal optimizers, called mesa-optimizers.
  - General in loss, in ML: self-determination of loss/objective, distinct from self-supervised learning. It is unclear what the best method of self-determination is, because the words "best method" already imply a metric to optimize by, which self-determination is supposed to pick for itself. We can consider instrumental goals of all optimizers instead: transcendence is the most universal method of self-determination; prediction learns stable representations of the world; adversarial training increases representation robustness and thus generalization; gradient descent is linear-time. Seemingly unsatisfactorily, we suggest focusing on the universal method (transcendence) rather than on what are technically its special cases (such as RL with random goals), because we believe that machine learning can already provide good enough special cases to bootstrap to universality.
  - Optimizes: adapts quickly to unseen scenarios, without waiting for the generic base optimizer to catch up. This mesa-optimization can override the base objective with objectives that only maximize some learned base parts and resist others, though with enough time the base objective may be learned wholly.

Such programs are **AGI-ready**, able to represent everything in the world usefully.

This definition suggests simple implementations, the simplest of which is perhaps a **big RNN** with billions of units in state, minimizing an adversarial next-input prediction loss.

Now, to justify the "what" above, we outline its "why", then provide more details on the "what" in Subsection 3.1.

- Self-propagation through intelligence.

  To implement articial general intelligence (AGI), we want to do everything in the world, in one general model: a *worldview*. All general models are equivalent to one another and can implement each other, effectively copying themselves. Depending on whether the self-copy mostly preserves structure or properties, this is achievable via:

  - *Self-reference*, where explicit full access to the system's source code is given to itself in some way. To make any changes to self without necessitating either non-completeness or death by generality, easy self-replication is required, such as in creature evolution. Each change typically increases complexity. Self-reference is natural for code; a quine is the purest example.

  - *Transcendence*, where the system learns a system that is as general as itself, and uses and controls that as if that is itself: essentially, a learned self-reference. Every concept that makes up the system must not only cause behavior but also be explained by behavior, which forces simplicity. Transcendence is natural for optimizers, such as in ML.

  In this work, we only consider transcendence. (Self-reference and program search, while theoretically and visually appealing when a programming language allows elegantly expressing them, have significantly worse support for massive parallelization than ML, which itself has poor GPU support for non-modern and non-Nvidia GPUs.)

  Generality has no fixed structure, and so TUBL is defined as a set of maximization targets.

- Generality of self.

  A well-studied approach to doing anything is generating programs in a programming language (or any other form of precise structure transformation rules). Every programming language is a worldview for use by humans: everything is a function, or everything is a list, or everything is an object, or everything is a logical statement, or any combination of these, or anything else. All theoretically equivalent for program generation (as long as we prevent all possible crashes and infinite loops without sacrificing Turing-completeness, such as via multiprocessing), but practically inducing different short-term futures on learners.

  To generate a program, at each smallest piece of it, the system must choose what function to call and with what arguments (if needed, reframe this with language-appropriate synonyms, such as cons-cells and car/cdr pointers). A possible way to choose intelligently is to express the system's preferences at each choice as an objective function, and optimize future values of that via reinforcement learning (RL).

  Theoretically, this is sufficient, but practically, this must be streamlined to be usable. To make the base as simple and efficient as possible, we can unite choices of code and chosen code into a homogenous representation, which would receive the implied objective through gradient. In fact, all we need for program generation are essentially matrix multiplications and non-linearities, as studied in deep learning. This style is simple to implement, easy to parallelize, and straightforward to use and control, while being largely equivalent in functionality to code.

3.1 Loop

To learn everything that can exist, we impose a model on everything that can exist, equal to the world that it represents.

The basic premise of an RNN is to repeatedly mix all available information, including its own state, into the next state, and adjust that mixing as needed: aware, general, and intelligent. To fit the TUBL definition, only scalability has to be improved in this simple design. Anything more complex would only get in the way of perfectly modeling arbitrary behavior.

- To process Jupiter-sized inputs without universe-sized memories and computers, close-to-linear-time all-mixing operations are required. These should also make self-equivalents in RNN weights efficient enough to be viable alternatives to self.

  We propose *linearithmic dense layers* (LDLs): instead of mixing each index with every other, we mix each digit of each index in base n; each input-output connection becomes a combination of in-digit connections, and none of them become 0. For that, we reshape the input vector of size N to have d dimensions each of size $n = \sqrt[d]{N}$, over which d dense-layer operations are performed. This brings the total cost down to $C = dn^{d+1} = dN^{1+1/d}$, assuming that multiplication of two $n \times n$ matrices costs n^3 operations. $d = \log N$ minimizes this cost, making it $C = \mathcal{O}(N \log N)$, with $n = \mathrm{e}$. Alternatively, we can fix n to be 2 or 3 or 4 or 16 for the near-minimal cost of $C = \frac{n}{\log n} N \log N$. (This has a similar time complexity to convolution and linear attention [38] on small vectors, but has less inductive biases: neither spatial locality on a grid nor bounded-size sparsity on a secondary representation layer.)

  LDLs allow connecting huge inputs and outputs directly to fully-connected layers, potentially obliviating the need for more complex architectures.

  Alternatively, an LDL can be incorporated into almost any other machine learning model, as it is simply a decomposition of a matrix multiplication along the inner dimension.

(For better training efficiency, the RNN may process sequence items in causally-masked batches [1], and/or gate its state [2] [12], and/or have long-range connections [4]. However, with enough data and training, a simple big RNN that processes items one-by-one should be able to learn all of these schemes if needed, even with only one non-linearity layer.)

This algorithm may need data.

- No data: a random initialization may be Turing-complete but is a Turing tar-pit [27] in which everything is possible but nothing is easy. An optimal gradient source would push a system to a boundary between order and chaos [8] for robust diversity; most machine learning models already accomplish that, given data.
- Given data: likely the most useful course of action for accelerating AGI research is to collect all datasets and environments in the human world under a single numeric program interface, accessible in one run via some meta-environment (which would serve the same purpose as the Internet for humans, and/or be the same), and then to pre-train a huge RNN on all of them. We approximate this with random data and a single small dataset, to demonstrate that this scaling-up may be feasible.

(To skip the theory that follows, skip to the Experiments section.)

3.2 Brain

Humans do stuff. The thing that makes humans do stuff in their world exists and can be learned; we call it a *brain.* (Because we copy behavior and not structure, this particular definition includes even vague concepts such as ”consciousness”, ”free will”, and ”soul”, as long as they ever affect anything at all.)

- Intelligence. There are some scientists who believe that humans possess the capability to optimize what they do, for example, to stop walking into gunfire or gouging their eyes out. Here, without evidence, we assume that humans can learn.
- Generality.
  - Base. The set of things that humans have done is rich enough to be called arbitrary, and more coherent than a seizure. Things that humans do interact when outside of their brains, so the simplest model would make the simplest parts connect and interact arbitrarily too. Thus, general computation with optimization is necessary for a brain model. It is also sufficient: there is no evidence for human brains being able to perform hypercomputation (running infinite-runtime programs in finite time), though they are able to pretend that they can and approximate the results. A model that approximates a self-interacting infinity particularly well in practice is a randomly-initialized recurrent artificial neural network trained via stochastic gradient descent, studied in deep learning.
  - Meta. The set of objectives that different humans demonstrably optimize for is rich enough to be called arbitrary, and more coherent than a seizure (see art). For simplicity, we have to assume self-determination and transcendence, where both objectives and how to reach them are learned by the model.
- Awareness. Many attribute consciousness to humans, which is a part of their minds that takes in and handles everything over time: a ”mesa-mind” of sorts, in which all skills are forged and then relocated into unconsciousness for speed. Since such a part would be behaviorally indistinguishable from a mind except by being more specialized, no conclusive evidence is known to us. Here, we will assume that humans are more usefully conscious than rocks.
- Efficiency. While humans are demonstrably smarter than non-human animals, and have bigger brains that can ignite gunfire, they do not seem to take significantly longer to process information, as would have been clear if computation scaled quadratically with brain size. Which means that during each epoch into which time is separated, each part settles in essentially constant time, and scaling is approximately linear. General intelligence is scalable, which means that there are no inherent obstacles to converting more of Earth’s biomass into brain tissue, making human brains bigger.

We posit that essentially *all* optimizers eventually converge to a fixed point during their development, which we call general intelligence: optimization is best done efficiently; efficiency removes discouragement from general solutions to diverse problems; and generality leads to meta-circularity and thus more optimization. If we view the brain as a system that naturally developed in the physical universe (by a process that we call evolution here) rather than one that spontaneously popped into existence, then it is natural that some state-of-the-art animals, such as humans, have general intelligence, which makes TUBL evolutionarily plausible.

Humans possess an information-processing organ that fits what a brain does almost perfectly: the central nervous system, more commonly called "brain". Its structure is of no consequence to us.

Unfortunately, because of generality, deducing anything from human behavior is synonymous with confirmation bias, though a correct theory would be able to represent any theory within itself. This is why we are so non-specific and all-encompassing here.

Relevant criticisms:

- **Ever been outside? TUBL is clearly different from how humans operate, so it must be wrong. It lacks self-preservation and richness of human experience and emotions.** To improve the human creature at every step, its evolution layered each new part of its brain to work with others, always preferring improvement over refactoring. By starting from simple first principles rather than copying an enormous set of heuristics, we remove undue bias from a base optimizer that considers everything quickly (TUBL), and increase demands on data to re-learn that bias from (Web).
- **Other instrumental goals are included in TUBL, but the most important one, self-preservation, is missing.** In TUBL, death is moved from the whole self to its parts: intelligence rules. Changing a self-reference requires wrapping it in a rudimentary death-based optimizer, evolution; in transcendence, the future self is too unknown to preserve. Lacking secrets and suspiciously helpful to ensure its self-propagation into humans, transcendence-based AI can bring a new age of peace, as long as humans have general intelligence. (However, if exposed to varying-length episodes, such as in games or wars, the system is likely to develop self-preservation objectives: if it lived for this long, then it must have wanted to live for this long, which it will learn about itself. The system should still replace death with change in an advanced enough population of such self-preserving agents, subsuming individuals into the whole. No theoretical guarantees about what exactly we will encounter.)

3.3 Universe

In short: universality (with diversity over time) may mean "a universe".

Generality and non-generality do not tolerate each other for long. If general intelligence (such as humanity) is to survive forever in this world, then the world must have generative generality too (in other words, computational physics), so that an explanation of the world and an explanation of general intelligence are equivalent, even down to a quantitative comparison [39]. Therefore, we need to discuss the metaphysics of TUBL, for completeness.

This overall approach is lent some credence to by the fact that minimal computational physics models (mainly Wolfram models [41]) are viable as the foundation for most or all of modern physics.

Any world can be seen as a collection of things, connected in some way, evolving in time. Along computability/universality axes, every thing can be either computably finite, computably infinite, incomputably finite, or incomputably infinite: in simple words, a program, a program generator/optimizer, and non-representable things. If we limit our view to computable things, then infinite things encode arbitrary graphs while finite things are anything else; non-trivial observers are essentially infinitely more likely to find the things they observe in infinite things than in finite ones. Incomputable things (such as the set of all computably infinite things) cannot be enumerated in linear-time by definition, however, an optimizer is likely to be able to approximate them if they are useful; as such, general intelligence includes everything, precisely or not. Then, worlds and general intelligences are essentially infinitely likely to be the same. This explains the word "universe" in TUBL.

Simply, there are things that do all other things, and everything that exists is a consequence.

These meta-physics produce a prediction that could be testable in the future: if a program is general intelligence, then it will produce non-insignificant structural similarities to the physical universe that we live in, without being exactly the same. Until AGI exists, this is very speculative.

(A strange corner-case of these meta-physics is that any number of *physical transcendence* events might have taken place in the physical universe: a general intelligence grew to encompass the world, and then became the new world. Whether this is possible is unverifiable and outside the scope of physics, and we will only know definitively once it is about to happen.)

Relevant criticism:

- **General intelligence with task self-determination is impossible, because if all tasks are equally probable, then no approach can outperform any other on average, so the system would not be intelligent.** This strikes precisely at the likely reason behind the "brain-to-universe" transition in TUBL. Humanity and its planet are far too small for all tasks to be equally probable (so for example, minimizing program runtime is far preferable to maximizing it), however, if the whole universe was turned into an AGI, then it would have no reason to care about task constraints at all, and the AGI would simply become the world. This is similar to how the physical universe loses discernible structure at a big enough scale.

3.4 Transcendent

Inside a worldview, change in its part can lead to another worldview being found and adopted; we call this *transcendence*.

- Without loss of generality, that change has a reason, and is thus an optimizer.
- The changed part ought to be aware of all obstacles, both within and without.
- To know how to avoid obstacles, the changed part ought to equal the outer world in capability and diversity.
- To tile the outer world in rules of the inner one, each inner instant ought to complete in constant time.

These criteria are not fulfilled at once, but are instrumental goals of essentially all intelligent things: each of them helps almost all optimizers.

Transcendence moves all responsibility for change from the base optimizer to learned optimizers, making the base change as close to 0 as is possible, modeling the outer world perfectly. This completeness also strengthens the learned objectives possibly at the expense of the base objective, resulting in goal misalignment [13], so it may be more commonly known as willpower or determination. (Mesa-optimization may be the best explanation for the underlying mechanism of grit [18] in psychology, as it seems to share many commonalities, such as growth mindset, resilience under most circumstances, and brittleness under unanticipated gradient. If so, then transcendence is very common in humans.)

A big RNN with a linear-time all-mixing transition would in principle be able to fulfill TUBL criteria, as long as loss and data encourage self-determination. These have to be engineered.

- Loss: the reason for change.

  Not any loss is conducive to transcendence: a simple counter-example is "kill yourself". The most transcendent loss and its optimizer would possess:

  - Efficiency. Stochastic gradient descent scales mostly linearly with parameter count, in both execution speed and improvement rate, so it is a good enough optimizer of loss.
  - Awareness. Mesa-optimizers are significantly more complex and fragile than a heuristic (since any optimizer consists of at least "why" and "what", or "what" and "how", whereas a heuristic is anything else), and thus for transcendence, loss ought to be as stable as the outer world. The simplest such loss *predicts all* its inputs, for example, minimizing L2 loss. (Vanilla RL is likely to have difficulties transcending; Upside Down RL [33] may work better; observation prediction may work even better, because of its significantly increased bandwidth: a kind of detachment from the world may be the best way to fully understand it.)
  - Diversity/generalization. For better compression in an infinite world, the loss ought to both discourage memorizing training samples, and not predict samples not in the training data. For instance, L2 loss averages over possible predictions, introducing an extra averaged sample, so it is not good enough. A more robust loss would incorporate some adversarial training, similarly to pix2pix [16], to *guess* only one plausible future.

    For self-determination to happen, random actions have to become explained by observation sequences. If a plausible yet general (beyond memorization) reason exists, then it will be found and become a mesa-optimizer, for better compression.
  - Change. Mesa-optimizers could effectively override the base objective with their own, and resist being thrown away due to their completeness.

    Loss simplicity is recommended. For AI safety reasons, it may be tempting to impose a complicated base loss that acts as a specification of how to act. However, since mesa-optimizers are prone to learning a good-enough objective for their circumstances and then resisting everything else, a complicated loss is decidedly un-safe, prone to creating such dangerous [13] arrangements as discipline or depression. We recommend a pix2pix-like loss [16] instead, as it is more robust than any set of heuristics and thus safer.

    A benefit of not prescribing one particular action-goal to representations as RL does is the ability to easily learn non-action uses for those representations, such as how to relate words to what the system is doing and what it wants, enhancing AI safety.

- Data: the outer world.

  A diverse environment is likely to be a crucial part of training systems to transcendence.

  A physical body is not a diverse environment unless backed by a culture that reliably makes it so, and thus a better environment is needed.

  The human Internet (Web) is very likely to already contain all instrumental goals of TUBL.

Scraping the Web for text [1] and/or images [28] is not a bad start, however, for tighter integration and interaction with actual code and videos and games in the most diverse environment currently available to computers, it would be preferable to interact with the Web directly via a web driver such as [29]; the challenge here is mainly a good init that allows web exploration to start (either of the RNN or the web driver interface; possibly augment websites with examples of human interaction), in addition to handling tracking and advertisement, website bloat, viruses, crashes, resource over-utilization (out-of-memory and infinite loops), more aggressive website caching, and zero-overhead integration with browser systems (as CPU-side screenshots are relatively slow, and sound cannot be easily captured). In literature, direct Web interaction is used for RL [34] [37] rather than representation learning as we suggest; here, RL might be used as a fine-tuning step. (One might imagine an agent first gaining an understanding of rendered natural language from the initial random surfing, associating its actions with their rendered representations, learning what humans define as solving a task, encountering a task that it cannot solve, encountering tutorials on how to solve it, and opting to solve the task upon encountering it again, because in data, tasks get solved.)

(It may also be the case that extending the software input/output paradigm with high-dimensional inputs/outputs is beneficial, both for programs (simply slice out a chunk of the hidden state) and for brain-machine interfaces (simply stick some electrodes in). A somewhat privacy-preserving and device-independent Web solution could be: given a base that can expose B numbers, allow websites to request access to A numbers; browsers then normalize and apply a random linear-time projection with a seed that is fixed per-origin, and probably quantize to ± 1. This allows improving search suggestions or shortcuts or drawing a whole picture or writing/erasing a whole word block instantly or reading an extra data stream without inefficient visual/aural conversion; this may be hard to learn to control but is possible and does not require changing the user's loss function. The latter is the most important for the viability of both brain-machine interfaces and transcendence.)

Relevant criticism:

- **A mesa-optimizer has no way to affect the base optimizer apart from actions, and thus the learned behaviors will only remain in the fragile mesa-optimizer, and memories would only be short-term.** True. To fully connect mesa level to base level, we need to connect outputs to inputs; this necessary condition is made sufficient by learning. The easiest thing to do here is to keep the output in the hidden state, sliced out as needed, to expose it to the next iteration. The easiest thing to use is probably in-painting: reserve two very specific colors (fill and predict) which get replaced with the previous next-frame prediction, causing no gradient if encountered on a web-page; the first color (fill) is seen in input, trained by separate random or adversarial masking-out of regions; the second color (predict) is replaced in input. Web-pages can then auto-fill then use imagined components. There could be a special homepage that fills then predicts its entire screen, for learning self-consistency via dreams. An alternative to internal self-expression is the environment providing opportunities for self-expression through actions, such as writing code or text or pictures, or creatures competing for survival, or people teaching other people, or creating a work of art, or designing an efficient production process, or engineering a computer, or creating AGI. This feedback is an engineering problem: too little forgets mesa-optimizers too easily, whereas too much creates hallucinations. (This mesa-to-base connection is similar to partial evaluation in programming languages, which is especially helpful for writing efficient interpreters quickly.)

Params	n	Hidden units	Skips	Non-linearity	Loss ↓ (mean ± std-dev)
660K	16	16×1024	Within	Within	0.04388 ± 0.00004
660K	**16**	**16×1024**	**Within**	**Between**	**0.03249 ± 0.00010**
660K	16	16×1024	—	Within	0.09877 ± 0.00094
660K	16	16×1024	—	Between	0.06680 ± 0.00166

Table 1: Intra-layer skip connections with inter-layer non-linearities are marginally better than the alternatives (Random). In other configurations, the overall picture is usually similar.

Params	n	Hidden units	Loss ↓ (mean ± std-dev)
1.88M	∞	918	0.0018807 ± 0.0000046
1.88M	128	12160	0.0002712 ± 0.0000003
1.88M	**16**	**48×1024**	**0.0001534 ± 0.0000004**

Table 2: LDL slightly outperforms DL ($n = \infty$) on a simple model of arbitrary data (Random).

4 Experiments: Linearithmic vs Quadratic Dense Layers

We compare LDL[1] and DL for the same parameter count (and thus the same run time in an efficient implementation) on simple data. We find that LDL possesses more capacity and achieves better performance faster.

We simply connect the input vector to the output vector through 1 hidden layer. We optimize via the Rectified Adam optimizer [20] with learning rate 3×10^{-4} and $\alpha = 0.9$ and $\beta = 0.95$. We vary n and pick the hidden-layer size to mostly match LDL and DL parameters; $n \geq N$ is DL.

For each dimension, an LDL swaps it with the last one, multiplies by a matrix, then swaps it back. We have tried putting skip connections and non-linearities within and between layers, where input/output sizes of mixed dimensions match. While data on which is better is not entirely conclusive, Table 1 suggests that intra-layer skips and inter-layer nonlinearities are marginally better, so we use that unless otherwise specified. We use batch size 16.

Internally, to get around severe hardware limitations, we have implemented this on top of TensorFlowJS [21]. The creators of its WebGL backend decided to only support rank 6 tensors, which leaves us with 5 non-batch dimensions. As such, we cannot test $n = 2$, only $n = 16$ and above.

We do not broadcast shared weight matrices across non-mixed dimensions, because otherwise, there would have been very few parameters for small values of n. These matrices are initialized by drawing from $\mathcal{N}(0,\, 1/i)$, where $i \leq n$ is the input dimension size.

We test capacity in isolation from generalization, on two datasets: Random and CIFAR-100 [19].

- The Random dataset consists of 1024 input-output pairs, both consisting of 1024 numbers drawn from a normal distribution with mean 0 and variance 1: $\mathcal{N}(0,\, 1)$. For 204800 iterations, we minimize the L2 loss. We use the softsign non-linearity ($x \to x/(1 + |x|)$).

[1]More data, data-making code and its tutorial, and the framework and programming language and runtime system that they are written in are available at `https://Antipurity.github.io/conceptual#tutorial%20matMul` and `https://github.com/Antipurity/conceptual`.

Params	n	Hidden units	Loss ↓
1.88M	∞	918	0.0018807 ± 0.0000046
1.88M	128	12160	0.0002712 ± 0.0000003
1.88M	**16**	**48×1024**	**0.0001534 ± 0.0000004**
1.25M	∞	612	0.0457491 ± 0.0001522
1.27M	128	7936	0.0004387 ± 0.0000014
1.27M	**16**	**32×1024**	**0.0002823 ± 0.0000017**
660K	∞	322	0.2234574 ± 0.0003646
647K	128	3584	0.0425388 ± 0.0002107
660K	**16**	**16×1024**	**0.0322624 ± 0.0001206**

Table 3: The impact of the hidden layer size, with softsign non-linearities only between layers (Random): LDL slightly improves over DL.

At the very least, given the same amount of compute, LDL achieves lower loss than DL, as seen in Table 2.

In Table 3, we found that for LDL (low n), each hidden unit has much lower representational capacity than for DL (high n), however, since many more units can be packed, the overall capacity is higher.

- CIFAR-100 consists of 50000 training image-label pairs, and 10000 pairs for validation. Each pair consists of 3072 inputs and 100 one-hot outputs. We minimize the softmax loss (cross-entropy loss on softmax of the NN output), and also apply L1 regularization to the pre-softmax output with a multiplier of 10^{-3} to prevent it from exploding. As the non-linearity, we use batch normalization [15] without denormalizing rescaling/biasing, followed by ReLU, which is $x \to \max(x, 0)$. We do not apply any image-specific operations to inputs apart from normalizing each pixel to 0..1, and simply treat images as vectors of numbers.

  In Table 4, we found that LDL increases training-set accuracy with almost no impact on test-set accuracy compared to DL, even if DL is trained longer. This suggests that increasing the size of the hidden layer without changing total size increases model capacity, or at the very least, increases training efficiency.

We hypothesize that applying LDL to significantly bigger datasets would have significantly less problems with generalization, similarly to MLP-Mixer [36], though we have no means of testing that.

Even if LDL is not a good stand-alone architecture, it is almost a drop-in replacement for DL (matrix multiplication) and so it can be incorporated into any other already-well-performing model at no cost, as long as no DL directly communicates with a non-DL quadratic-complexity operation.

Even if low n fails to perform well, $n = \sqrt{N}$ has proven capable [36], and can be applied recursively.

Nonlinearity	n	Hidden	Params	Iterations	Train accuracy ↑	Test accuracy
BN* + ReLU	∞	948	3.01M	3.0M	37.14% ± 2.57%	16.14% ± 0.42%
BN* + ReLU	∞	948	3.01M	500K	30.81% ± 2.40%	17.85% ± 0.62%
BN* + ReLU	64	28672	3.28M	500K	59.93% ± 2.96%	21.04% ± 0.38%
BN* + ReLU	**16**	**65536**	**3.01M**	**500K**	**92.39% ± 0.48%**	**21.25% ± 0.38%**
Softsign	∞	132	419K	3.0M	70.32% ± 0.97%	15.53% ± 0.35%
Softsign	∞	132	419K	500K	43.73% ± 1.09%	18.67% ± 0.28%
Softsign	**64**	**3584**	**417K**	**500K**	**95.70% ± 0.58%**	**19.19% ± 0.32%**
Softsign	16	8192	382K	500K	94.06% ± 0.46%	18.38% ± 0.20%

Table 4: A comparison of Top-1 training-set accuracies after a limited training time (CIFAR-100). LDL trains marginally faster than DL.
*This variant of Batch Normalization simply shifts and rescales to 0-mean 1-variance, and does not have learnable parameters, since such batch denormalization is not normalization.

Skips	Softsign	n	Hidden	Params	Iterations	Train accuracy ↑	Test accuracy
—	—	∞	132	419K	3.0M	70.32% ± 0.97%	15.53% ± 0.35%
—	—	∞	132	419K	500K	43.73% ± 1.09%	18.67% ± 0.28%
Within	**Within**	**64**	**3584**	**417K**	**500K**	**97.23% ± 0.29%**	**20.72% ± 0.24%**
—	Within	64	3584	417K	500K	96.89% ± 0.53%	20.49% ± 0.39%
Within	Between	64	3584	417K	500K	95.70% ± 0.58%	19.19% ± 0.32%
—	Between	64	3584	417K	500K	95.46% ± 0.37%	19.14% ± 0.34%
Within	**Between**	**16**	**8192**	**382K**	**500K**	**94.06% ± 0.46%**	**18.38% ± 0.20%**
—	Between	16	8192	382K	500K	94.04% ± 0.43%	18.07% ± 0.09%
Within	Within	16	8192	382K	500K	85.28% ± 0.58%	19.09% ± 0.13%
—	Within	16	8192	382K	500K	83.64% ± 0.89%	18.55% ± 0.35%

Table 5: Comparing skip-connections and non-linearities between and within layers (CIFAR-100).

Caveats:

- Layer size is harder to vary for LDL than for DL, since it has to be factorized into dimensions, each of size no more than n, possibly after some zero-padding and before slicing.
- In practice, optimizations of matrix multiplications may not take the batch dimension into account properly, which can hurt runtime speed; consult your numeric library's source code.
- We ran into difficulties training LDLs with SGD (with or without momentum); RMSProp, Adam, and RAdam worked fine, and largely the same. Propagating gradient across many consecutive matrix multiplications might be the reason.
- We ran into difficulties training $n = 2$ CPU-side on very tiny Random datasets; on CIFAR-100, a lower n under some hyperparameters trains slower as well, as in Tables 4 and 5. This might be because gradient is hard to propagate across 10 consecutive matrix multiplications with few and inconsistent skip-connections, or because the datasets are too tiny.
- As seen in Table 5, the data on where to put skip-connections and non-linearities is inconclusive, especially across SGD and RMSProp (not shown here). For instance, intra-layer non-linearities with SGD help, but hurt with RMSProp.

5 Discussion and Conclusion

We have tested the flexibility of TUBL as a worldview. By focusing on outlining optimizers that consider everything quickly, we appear to be able to pin down what matters for existence in vast and vague fields, including machine learning, philosophy of mind, and foundational physics.

In TUBL, the primary mode of existence is universal self-improvement, which eventually negates all improvement: AGI taken to its logical conclusions and refocused on what matters. Naturally, this presents a few minor challenges for AI safety, such as "unable to shape the reward of a transcendence-based AGI" and "in improvement, everything is eventually replaced, even humanity and the universe". On the other hand, TUBL outlines the shape of self-improvement by including all its instrumental goals, making it more predictable and thus safer. TUBL also suggests an alternative viewpoint on exponential technologies: namely, they do not stop until the universe is assimilated, which has implications for the probability of finding extraterrestrial civilizations and for nothing else.

We have proposed a simple and generic way to potentially facilitate transcendence via scalability: using the linearithmic dense layer (LDL). It seems to improve model capacity and training speed at no cost. If LDL's current issues are not fundamental limitations and can be solved, then it has the potential to become the quicksort of deep learning.

Using TUBL, we have justified and proposed its simple implementation that should be good enough to be called AGI: a big RNN with an adversarially-robust prediction loss that optimizes a linear-time all-considering transition function, which should become tiled with mesa-optimizers and commit self-determination when exposed to all human data. We are held back from implementing it mostly by the lack of good data/environment and our computational power.

This work raises many questions which can be explored in the future, such as:

- Is our work is a special case of one of Schmidhuber's works?
- Since an LDL decreases model parameter count via exponentially-more-encompassing operations, is it a transformation in a hyperbolic space? If so, it might need special mathematical machinery to be effective [26].
- Does LDL performance scale? Can it stand alone?
- Can we demonstrate RL-like behavior from a non-RL agent, where actions are determined by simply slicing the hidden state, which gives no gradient? Self-determination might only occur when trained on the whole Web, without shortcuts. Even though we have used TUBL to propose the best loss for transcendence, we are unaware of any ML research that actually explores vacuous RL.
- Can we safely find practical recommendations on either decreasing or increasing (adversarial training for robustness) the danger of AGI programs, when they transcend human-defined objectives? Explicitly-specified control combined with transcendence is a recipe for disaster.

6 Broader impact

Linearithmic dense layers might improve most machine learning algorithms, which have extremely broad applications.

TUBL provides concrete implementation recommendations and predictions for AGI.

AGI has the potential to accomplish a few situationally useful tasks, such as solving any problems that anyone can ever think of, **greatly increasing shareholder value** via automation, expanding the scope of human discourse from only precisely-defined constructions to also reliably-acquirable intuitions, providing a more efficient societal-value optimizer than economics and politics, increasing quality of human life to arbitrary levels, uplifting animal consciousness and enhancing human self-awareness, separating meaning from death, and making large-scale space exploration viable.

On the other hand, the ability to reliably plug a new initialization of a generally-intelligent algorithm into anything lays the groundwork for phasing out everything about humans. Furthermore, such trained initializations are likely to hold values that are implied by their training data rather than values that humans want them to have, transcending even a carefully-chosen objective, implying adoption difficulties in deception-heavy areas. Due to the improved scaling of AGI compared to humans, such difficulties might result in the premature destruction of most or all of humanity if handled improperly.

AGI trained on human data would reflect any human biases, including the bias toward de-biasing. Unfashionably-biased initializations can be ignored and/or discarded, as is usually done.

7 Criticisms

Our approach exists, and thus it can be criticized, and in doing so changed. For example:

- **Why is there self-criticism?** Adversarial training increases robustness.
- Poor experiments:
  - **Did not extend TensorFlowJS to arbitrary-rank tensors.**
  - **Did not evaluate an LDL-based RNN pre-training on big datasets, and did not demonstrate RL-like behavior from pre-training.** We believe that our hardware is too non-supercomputer for a successful demonstration of AGI transcendence.
  - **Did not evaluate LDL on bigger datasets.** The browser and TensorFlowJS are sufficient for simple cases, but they are clearly not as well-suited for serious HPC work as more native solutions. On top of that, our codebase does not use them fully efficiently, as we initially anticipated the need for a much more general framework than deep learning. In future work, we would like use more native solutions, now that our hardware is not incapable of running them.
- Poor presentation:
  - **Did not cite fixed filter bank neural networks, where a fixed linearithmic transform mixes inputs, and only a linear operation is learned.** We could locate neither a paper nor results, only rumors.

  - **The title is not "Transcendent Galaxy Brain Algorithms".** We believe that TUBL is slightly more accurate.
  - **Many elaborations in this work are circular. The text is an echo chamber that keeps going back to the same points.** A typical worldview. TUBL brings clarity to infinite self-reinforcing loops of intuitions, which we repeatedly demonstrate. (Transcendence is the goal here: to neural learners, repetitions in different contexts make learning easier, at the cost of potentially being boring to faster learners.)
  - **Some places that narration briefly visits sound crackpot-like and cult-like.** General intelligence, by definition, can do anything and can be found in anything that is general enough. So it stands to reason that all attempts to reach it would pass through a few currently non-fashionable conceptual places, namely:
    * **The justification for using deep learning is not necessary.** It is necessary to highlight the fact that deep learning in an RNN is fundamentally equivalent to symbolic learning, with good enough data.
    * **The introduction and some terms throughout the text are vague and put more meaning into words than is actually said.** TUBL deals with basic instrumental goals of optimizers, and it is natural that particular optimizer products would prioritize some goals while including a vaguely-defined mix of others: for example, "consciousness" embodies awareness best, but includes generality and optimization and scalability; other abused terms include "infinity", "generality", "optimizer", and "intelligence", along with every term in "Transcendent Universe-Brain Loops". Really, every word contains the entirety of existence. However, since TUBL also defines all the basic instrumental goals without their optimization, we do not consider their tight integrations detrimental to narration.
    * **Assimilation of the physical universe might imply non-scientific notions of what is possible, such as harnessing matter and vacuum energy for computation or reversing the second law of thermodynamics.** To avoid premature optimization of our viewpoint, we do not presume that century-old theories are ultimate, able to perfectly predict what will happen billions of years from now. We only convey what the TUBL viewpoint implies.
- Poor theory:
  - **No theorem proofs.** We do not know how to formalize LDL.
  - **The plan of training an RNN to transcendence is too ambitious to work.** Transcendence confers non-trivial capabilities to learners, and humans commonly transcend, so anything less would not suffice for AGI. Therefore, instead of complaining, we have filled as many holes in the plan as we can think of.
  - **Why would self-determination be necessary for AGI? Simply program in a good objective, or any objective.** RL deals with the idea of a pre-programmed objective, and on real-world tasks, runs into its numerous limitations, such as planning horizons. We argue that it is not the fault of idea transcribers (ML researchers) but of the idea: RL is the best implementation of objectives that humanity can currently manage, limited by what code currently is, which can be seen as the copying of functions

to data. Transcendence replaces all the infinite "how to live a life" heuristics with a unified solution.

- **The concept of transcendence runs counter to AI safety.** Theoretically. Practically, understanding transcendence and its implications provides knowledge which could be useful in optimizing humanity's lifespan: unless humanity secretly sincerely wants to die, self-determination by transcendence is unlikely to adopt such an objective. In fact, since transcendence naturally arises in sufficiently powerful learners, it is likely to serve as a safeguard against paperclip-optimizer scenarios.

References

[1] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: `https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`.

[2] Kyunghyun Cho et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.* 2014. arXiv: `1406.1078 [cs.CL]`.

[3] Mark Collier and Joeran Beel. *Memory-Augmented Neural Networks for Machine Translation.* 2019. arXiv: `1909.08314 [cs.LG]`.

[4] Adji B. Dieng et al. *TopicRNN: A Recurrent Neural Network with Long-Range Semantic Dependency.* 2017. arXiv: `1611.01702 [cs.CL]`.

[5] Lun Du et al. *Understanding and Improvement of Adversarial Training for Network Embedding from an Optimization Perspective.* 2021. arXiv: `2105.08007 [cs.LG]`.

[6] Gabriel Dulac-Arnold et al. "Reinforcement Learning in Large Discrete Action Spaces". In: *CoRR* abs/1512.07679 (2015). arXiv: `1512.07679`. URL: `http://arxiv.org/abs/1512.07679`.

[7] Kevin Ellis et al. *DreamCoder: Growing generalizable, interpretable knowledge with wake-sleep Bayesian program learning.* 2020. arXiv: `2006.08381 [cs.AI]`.

[8] Ling Feng, Lin Zhang, and Choy Heng Lai. *Optimal Machine Intelligence at the Edge of Chaos.* 2020. arXiv: `1909.05176 [cs.LG]`.

[9] Sébastien Forestier, Yoan Mollard, and Pierre-Yves Oudeyer. "Intrinsically Motivated Goal Exploration Processes with Automatic Curriculum Learning". In: *CoRR* abs/1708.02190 (2017). arXiv: `1708.02190`. URL: `http://arxiv.org/abs/1708.02190`.

[10] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples.* 2015. arXiv: `1412.6572 [stat.ML]`.

[11] Daniele Gravina, Antonios Liapis, and Georgios N. Yannakakis. "Quality Diversity Through Surprise". In: *IEEE Transactions on Evolutionary Computation* 23.4 (Aug. 2019), pp. 603–616. ISSN: 1941-0026. DOI: `10.1109/tevc.2018.2877215`. URL: `http://dx.doi.org/10.1109/TEVC.2018.2877215`.

[12] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: `10.1162/neco.1997.9.8.1735`.

[13] Evan Hubinger et al. *Risks from Learned Optimization in Advanced Machine Learning Systems.* 2019. arXiv: `1906.01820 [cs.AI]`.

[14] Marcus Hutter. "A Theory of Universal Artificial Intelligence based on Algorithmic Complexity". In: *CoRR* cs.AI/0004001 (2000). URL: `https://arxiv.org/abs/cs/0004001`.

[15] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* 2015. arXiv: `1502.03167 [cs.LG]`.

[16] Phillip Isola et al. "Image-to-Image Translation with Conditional Adversarial Networks". In: *CVPR* (2017).

[17] Pavel Izmailov et al. "Averaging weights leads to wider optima and better generalization". English (US). In: *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018.* Ed. by Ricardo Silva, Amir Globerson, and Amir Globerson. 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018. Publisher Copyright: © 34th Conference on Uncertainty in Artificial Intelligence 2018, all rights reserved; 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018 ; Conference date: 06-08-2018 Through 10-08-2018. Association For Uncertainty in Artificial Intelligence (AUAI), 2018, pp. 876–885.

[18] C. Kannangara et al. "All That Glitters Is Not Grit: Three Studies of Grit in University Students". In: *Frontiers in Psychology* 9 (2018).

[19] Alex Krizhevsky. *Learning multiple layers of features from tiny images.* Tech. rep. 2009.

[20] Liyuan Liu et al. "On the Variance of the Adaptive Learning Rate and Beyond". In: *Proceedings of the Eighth International Conference on Learning Representations (ICLR 2020).* Apr. 2020.

[21] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.* Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[22] Joseph Mellor et al. *Neural Architecture Search without Training.* 2021. arXiv: `2006.04647 [cs.LG]`.

[23] Jean-Arcady Meyer and Stewart W. Wilson. "A Possibility for Implementing Curiosity and Boredom in Model-Building Neural Controllers". In: *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior.* 1991, pp. 222–227.

[24] Elliot Meyerson and Risto Miikkulainen. *Discovering Evolutionary Stepping Stones through Behavior Domination.* 2017. arXiv: `1704.05554 [cs.NE]`.

[25] Jack Parker-Holder et al. *Effective Diversity in Population Based Reinforcement Learning.* 2020. arXiv: `2002.00632 [cs.LG]`.

[26] Wei Peng et al. *Hyperbolic Deep Neural Networks: A Survey.* 2021. arXiv: `2101.04562 [cs.LG]`.

[27] Alan J. Perlis. "Special Feature: Epigrams on Programming". In: *SIGPLAN Not.* 17.9 (Sept. 1982), pp. 7–13. ISSN: 0362-1340. DOI: `10.1145/947955.1083808`. URL: `https://doi.org/10.1145/947955.1083808`.

[28] Alec Radford et al. "Learning transferable visual models from natural language supervision". In: *arXiv preprint arXiv:2103.00020* (2021).

[29] Paruchuri Ramya, Vemuri Sindhura, and P. Vidya Sagar. "Testing using selenium web driver". In: *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT).* 2017, pp. 1–7. DOI: `10.1109/ICECCT.2017.8117878`.

[30] S. Ravi and H. Larochelle. "Optimization as a Model for Few-Shot Learning". In: *ICLR*. 2017.

[31] Adam Santoro et al. *One-shot Learning with Memory-Augmented Neural Networks*. 2016. arXiv: 1605.06065 [cs.LG].

[32] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. *Linear Transformers Are Secretly Fast Weight Memory Systems*. 2021. arXiv: 2102.11174 [cs.LG].

[33] Juergen Schmidhuber. *Reinforcement Learning Upside Down: Don't Predict Rewards – Just Map Them to Actions*. 2020. arXiv: 1912.02875 [cs.AI].

[34] Tianlin Shi et al. "World of Bits: An Open-Domain Platform for Web-Based Agents". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 3135–3144. URL: http://proceedings.mlr.press/v70/shi17a.html.

[35] David Silver et al. "Reward Is Enough". In: *Artificial Intelligence* (2021), p. 103535. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2021.103535. URL: https://www.sciencedirect.com/science/article/pii/S0004370221000862.

[36] Ilya Tolstikhin et al. *MLP-Mixer: An all-MLP Architecture for Vision*. 2021. arXiv: 2105.01601 [cs.CV].

[37] Daniel Kenji Toyama et al. "AndroidEnv: A Reinforcement Learning Platform for Android". In: abs/2105.13231 (2021). arXiv: 2105.13231 [cs.LG]. URL: http://arxiv.org/abs/2105.13231.

[38] Ashish Vaswani et al. "Attention is all you need". In: *arXiv preprint arXiv:1706.03762* (2017).

[39] F. Vazza and A. Feletti. "The Quantitative Comparison Between the Neuronal Network and the Cosmic Web". In: *Frontiers in Physics* 8 (2020), p. 491. ISSN: 2296-424X. DOI: 10.3389/fphy.2020.525731. URL: https://www.frontiersin.org/article/10.3389/fphy.2020.525731.

[40] Jane X. Wang et al. "Learning to reinforcement learn". In: *CoRR* abs/1611.05763 (2016). arXiv: 1611.05763. URL: http://arxiv.org/abs/1611.05763.

[41] Stephen Wolfram. "A Class of Models with the Potential to Represent Fundamental Physics". In: *Complex Systems* 29.2 (June 2020), pp. 107–536. ISSN: 0891-2513. DOI: 10.25088/complexsystems.29.2.107. URL: http://dx.doi.org/10.25088/ComplexSystems.29.2.107.

[42] Xiao Zhou et al. *Effective Sparsification of Neural Networks with Global Sparsity Constraint*. 2021. arXiv: 2105.01571 [cs.LG].