ChatGPT

Minimal Code, Maximal Field – Towards a "Donut" Framework in DoA

Minimal Code, Maximal Field Philosophy: The idea of "minimal code, maximal field" is that **a small, simple core of instructions can generate a disproportionately rich and extensive outcome**. In other words, systems following this philosophy achieve *maximum effect or coverage ("field")* from *minimal description or code*. This principle appears across domains – from information theory and mathematics to engineering, neural networks, and even cognitive science. A useful mental image is a **donut metaphor**: a tiny seed or hole in the center leading to a large ring of substance around it. In the context of DoA (our "Donut-of-Attention" system), we want to design algorithms where **short intention seeds (minimal input)** yield **rich focus or output (maximal field)**. Below we survey key concepts exemplifying this philosophy – **Kolmogorov complexity & fractals, compressive sensing, sparse neural networks (Lottery Ticket Hypothesis)**, and **cognitive/attention models** – and then propose a conceptual framework for applying the "minimal code, maximal field" approach to DoA.

Algorithmic Complexity and Fractals – Simple Rules, Infinite Complexity

In **algorithmic information theory**, *Kolmogorov complexity* measures the shortest program (minimal code) needed to produce a given output. Formally, it is *"the length of a shortest computer program that produces the object as output"* for a given programming language [1]. If an object (say, a string or an image) has a concise algorithmic description, its Kolmogorov complexity is low – meaning a small code can generate it. This concept embodies **Occam's razor** in a computational sense: the simplest complete description is the best model [2] [3].

A classic example is **fractal geometry**. Many fractals are defined by a short recursive formula yet produce infinitely detailed patterns. The *Mandelbrot set*, for instance, is generated by the iteration $z_{n+1} = z_n^2 + c$ – a few symbols of code. **Storing a detailed image of the Mandelbrot set could take tens of megabytes, but the Kolmogorov complexity is far smaller** since a compact program can reproduce the image [4]. In other words, the fractal's *minimal code* (the formula and parameters) expands to a vast *field* of intricate detail.

Figure: A close-up of the Mandelbrot fractal demonstrates how a minimal formula can generate endless complexity. Simply storing the raw pixels of such an image would require ~23 MB of data, but a short program encoding the Mandelbrot set definition and coordinates can reproduce it (implying a much lower Kolmogorov complexity) [4]. This exemplifies "minimal code, maximal field," as a tiny algorithm yields a richly detailed structure.

Fractals appear not only in math but in nature (fern leaf patterns, coastlines, etc.), suggesting the world itself leverages compact generative rules. The **Kolmogorov complexity** of these patterns is low even though their observed complexity is high. This teaches us that **compressibility** (existence of a simple

generative code) underlies many complex phenomena. For DoA, this inspires us to seek *simple representations that can be expanded* – akin to a donut's small center generating a broad ring. In practice, we aim to identify *minimal descriptors* (the "code" or *intention seed*) that still capture the essence needed to unfold a large outcome.

Compressive Sensing – Less Data, Full Signal

In engineering and signal processing, **compressed sensing** (or compressive sampling) is a paradigm that explicitly achieves "minimal input, maximal reconstruction." *Compressed sensing allows one to reconstruct a signal from far fewer measurements than traditional methods require, by exploiting the signal's sparsity* [5]. Essentially, if the underlying signal has a simple or sparse structure, we don't need to measure every detail – a small number of cleverly chosen samples will do.

Key idea: Rather than sampling at the Nyquist rate (which says sample at least twice the highest frequency), compressed sensing assumes the **signal is sparse** in some basis (i.e. most coefficients zero). Then one can collect **M << N** linear measurements and still recover the N-length signal exactly. Two conditions enable this magic [6]:

- **Sparsity:** The signal has a *minimal underlying description* (only K significant components out of N, with K << N). This is the "minimal code" – only K degrees of freedom matter.
- **Incoherence (or randomness in sampling):** We sample the signal in a way that spreads out information, ensuring each measurement captures a broad mix of signal components. Random projections or certain structured matrices are used as measurements, guaranteeing that no information is completely missed.

Using optimization algorithms (or greedy algorithms), one then **reconstructs the full signal by finding the solution vector that fits the measurements and has minimum L1-norm (i.e. is sparsest)**. This typically recovers the true sparse signal exactly [7]. In simple terms, the reconstruction *fills in the blanks* by leveraging the sparsity prior.

An illustrative example is the **single-pixel camera**. Instead of an array of millions of pixel sensors, it uses one photodetector and a spatial light modulator (like a digital micromirror device) to take many random combined measurements of a scene. Amazingly, *"the single-pixel camera can recover images from fewer measurements than the number of reconstructed pixels"* by using compressed sensing algorithms [8]. The final image (rich field of pixels) is obtained from a very limited set of observations (minimal data), thanks to the assumption that the image has structure (e.g. is sparse in some transform domain) and the power of the reconstruction algorithm.

Technical aside: Compressive sensing typically solves a problem of the form $y = A\,x$ (an underdetermined linear system with far fewer measurements y than signal coefficients in x) by adding a sparsity constraint. The recovery is done by solving an optimization like:

$$\min_{x} |x|_{1} \quad \text{s.t.} \; A x = y,$$

which finds the smallest L1-norm solution consistent with the measurements – under theoretical conditions, this recovers the true sparse x. This is a convex relaxation of seeking the truly minimal support solution (an NP-hard L0 minimization). The success of this approach, proven in works by Candès, Tao, Donoho, and

others, shows that **a remarkably small amount of information (measurements) can yield the full signal** if one intelligently exploits *hidden simplicity* (sparsity) [5].

For the DoA donut framework, compressive sensing provides a **pattern**: **find a compact representation (sparse features) and a way to *decode* it into a full signal**. The "field" (full signal) was always latent in the "code" (sparse coefficients); we just needed the right lens to expand it. This suggests we design systems that *store or identify sparse, meaningful pieces of information and expand them to rich outputs*. It also suggests a way to measure success: how many pieces of input are required to achieve a certain richness of output – the fewer, the better.

Sparse Subnetworks in Neural Nets – Lottery Tickets and Compressive Models

Machine learning offers analogous examples where **a small subset of a model can produce performance comparable to the full model** – again highlighting minimal structure yielding maximal results. The **Lottery Ticket Hypothesis (LTH)** is a recent idea that in a large neural network, there exist *"sparse subnetworks ('winning tickets') that – when trained in isolation – reach test accuracy comparable to the original network"* [9]. In other words, a dense network has a *needle in the haystack*: a much smaller network (with a fraction of the parameters) that can be as good after appropriate training.

Researchers Frankle & Carbin (2019) showed that standard pruning methods (iteratively removing the smallest-magnitude weights) can uncover these winning tickets [10]. For example, in networks for MNIST or CIFAR-10, they found subnetworks with only **10-20% of the original weights** that could be trained to match or even exceed the original accuracy [11]. Beyond that sparsity level, the small networks actually sometimes *learn faster and generalize better* than the full network [11]. This is a striking confirmation that **the "essence" of a model's capability may lie in a very compact sub-part**. The rest of the weights are like extra "code" that isn't fundamentally necessary – perhaps they help the full model train in practice, but they can be pruned for inference.

Compressive models in ML refer broadly to techniques for reducing model size while retaining performance. The lottery ticket is one paradigm (find a smaller architecture within a big one). Other methods include **knowledge distillation** (where a large "teacher" model's behavior is distilled into a smaller "student" model), and **quantization** or low-rank factorization (reducing numerical precision or complexity). All these aim at the same outcome: **minimal model, maximal task performance**.

Why it matters for minimal code philosophy: If a huge deep network can be reduced 10-fold or more with little loss in accuracy [12], this means the *effective information* needed for the task was much smaller than it appeared. The *minimal code* was hiding in the larger code. For DoA design, this implies we should focus on **finding the smallest sufficient structures** – e.g. identifying "winning ticket" configurations in whatever system we build – that still yield the full desired outcome. It also suggests a benchmarking idea: measure the *sparsity or brevity* of a solution needed to achieve a certain result (like how many neurons or rules are truly needed). An ideal "DoNUT" (Donut-of-Attention) system might dynamically prune away extraneous parts, focusing only on a concise sub-network or rule set that carries the day.

Implementation note: In practice, to identify a lottery ticket, one procedure is: train a full network to reasonable performance, prune a large fraction of weights (those deemed least important), *reset* the

remaining weights to their initial values, and then retrain only that subnetwork. If it learns well to the same performance, it's a winning ticket [9]. This process can be iterated to find even smaller tickets. Although originally this was done with iterative magnitude pruning, recent research has looked at early pruning or even finding winning tickets at initialization. The takeaway is that **building big and then compressing** is one way to reveal minimal effective structure – a hint that sometimes *starting with excess can help find the optimal minimal*. For DoA, however, we might aim to directly *generate or evolve minimal structures* for maximal output, rather than pruning down from a bloated start.

Cognitive and Attention Models – Prediction and Focus from Minimal Clues

Cognitive science provides intuitive parallels to the "minimal input, maximal perception" idea. Human brains are remarkably adept at *inferring a lot from a little*. Two interrelated concepts illustrate this: **predictive coding** in perception, and **thin-slicing** in social cognition.

Predictive Coding (Bayesian brain): Our brain is thought to function not just as a passive data recorder, but as a *proactive model-builder*. According to the predictive coding theory, the cortex constantly generates top-down predictions of sensory inputs and only the **errors** (differences between prediction and actual signal) are propagated upwards [13]. This means if the brain's internal model can explain the incoming data, no news is good news – nothing (or very little) needs to be transmitted. Perception is thus largely "filled in" by expectations, and *"signals from the external world only shape perception to the extent that they are propagated up the cortical hierarchy in the form of prediction error"* [13]. In effect, the brain **compresses sensory information by focusing on the unexplained pieces**. The bulk of visual experience, for example, might be generated by your brain's model (based on past experience and context), with your eyes just correcting the small deviations. This is analogous to a **decompression algorithm**: the brain has a generative model (like a code) that expands into a full sensory scene, and sensory data only tweak it at the margins. The minimal code here is our prior knowledge and predictions; the maximal field is the rich perception of the world we experience.

This predictive mechanism has ties to **attention** as well. Since not all prediction errors are equally important, the brain applies a sort of weighting (precision) to them – effectively *attention* is thought to amplify certain error signals that matter [14]. This ensures that only salient, informative differences (e.g. a tiger moving in the grass) capture processing resources, while expected, redundant inputs are ignored. From the "minimal code" viewpoint, attention is about *selecting the few critical pieces of new information* to process and letting our internal model generate the rest. The brain thereby achieves efficient cognition: huge amounts of sensory input are thrown away or down-weighted, and we concentrate on a small set of cues that have large explanatory power.

Thin-Slicing: A more high-level cognitive phenomenon is our ability to make **surprisingly accurate judgments from very limited observations**. Psychologists Nalini Ambady and colleagues showed that people can form valid impressions of someone (e.g. a teacher's effectiveness) from just a few seconds of silent video – a "thin slice" of behavior – that correlate strongly with opinions formed after semester-long exposure [15] [16]. In one experiment, observers watched **2-second clips** of teachers with no sound and rated their teaching; these ratings matched the evaluations given by students who had spent an entire semester with the teacher [17] [16]. Moreover, increasing the clip length to 5 or 10 seconds did not significantly change accuracy – *most of the relevant social information was conveyed almost instantaneously*

[18] . This finding underscores how *a minimal snippet (visual cues, body language) can let the human mind infer a large array of traits or outcomes*. The viewers' brains, in essence, *filled in the rest* (extrapolating the teacher's personality and competence) from just a thin data slice. It's an example of maximal field (a nuanced social judgment) from minimal code (a flash of observation).

Thin-slicing research aligns with the idea that humans have internalized many patterns (through experience or evolution) that allow **rich reconstruction from sparse data**. Just as a few measurements can reconstruct an image in compressive sensing, a few cues can reconstruct a person's likely behavior or a scene's gist in our minds. Other everyday examples include: recognizing a familiar song from just a few notes, or recalling a detailed memory from a faint smell. The brain stores *compressed representations* of these phenomena and can explode them into full perceptions or memories upon encountering a hint.

Lesson for DoA: Cognitive models emphasize the importance of a **good model or prior** to enable minimal inputs to yield maximal understanding. If our DoA system can be endowed with strong priors or knowledge (analogous to the brain's learned model of the world), then a small hint or "intention seed" can trigger it to retrieve or generate a wealth of relevant information. Also, implementing an **attention mechanism** would let the system focus on the few key inputs (or key parts of its knowledge) that have outsized influence on the outcome. In summary, the brain's example suggests we design DoA as a **predictive, generative system** that treats inputs as prompts to instantiate a much larger latent structure (filling in gaps using its own learned model), and that we ensure the system can identify which small inputs are critical (attention) to drive the expansion.

Toward a "Donut" Framework for DoA – Designing with Minimal-Maximal in Mind

Bringing these threads together, we can propose a conceptual **Donut Framework** for DoA that encapsulates "minimal code, maximal field." The framework would ensure that a *small core input* (the "donut hole") is leveraged to produce a *large, rich output space* (the "dough"). To achieve this, a DoA system can incorporate the following design principles and patterns:

- **Compressive Representations:** Just as sparsity and compression appear in many domains, represent knowledge or data in a compressed form. Identify the **minimal sufficient set of features** or components that can generate the rest. This could mean finding a **"lottery ticket" substructure** in a model (minimal network connections that suffice), or compressing information into a lower-dimensional latent vector. *Example pattern:* Use an autoencoder to encode a complex input into a small embedding, then decode it back – ensuring the embedding (minimal code) retains maximal pertinent info.

- **Generative Expansion Mechanisms:** Include processes that can **expand a short code into a detailed output**. This could be an iterative rule (like fractal generation) or a deep generative model (like a decoder in an LLM or GAN). The key is a reliable way to go from *seed to field*. *Example pattern:* In natural language, a large language model (LLM) takes a brief prompt and generates a long, coherent essay – the model's weights contain the latent structure of language, enabling expansion. In DoA, given a tiny "intention" input, the system might use a generative model to create a rich set of interpretations, scenarios, or detailed plans from it.

- **Multi-Scale Iteration (Fractal-like Refinement):** Borrowing from fractals and predictive coding, design the system to refine outputs across scales or iterations. A small initial output can be treated as a coarse outline that is then *iteratively refined*, adding layers of detail. Each iteration uses the same simple rules or model to add complexity (like zooming into a fractal or successive prediction error corrections in predictive coding). *Implementation idea:* The system might start with a broad guess based on the intention seed, then repeatedly focus on gaps or errors (areas where detail is lacking or predictions are off) to enrich the output. This ensures the complexity grows from the initial simple structure, much like a donut rising from a small ring to a fluffy shape.

- **Attention and Focus:** Internally, the system should **focus computational resources on key elements** of the input or intermediate state that promise the biggest expansion. This is analogous to how a small region of the fractal formula yields a huge swirl of detail, or how the brain's attention picks a small unexpected cue to investigate. Concretely, this could involve mechanisms to rank which parts of the generated output need more detail or which aspects of the input are most informative, and then concentrating expansion there. An attention module could guide the generative process to ensure the *most is made out of each bit of input*.

- **Sparse Adaptive Modules:** In line with the lottery ticket idea, the architecture could be designed to **activate only a small subnetwork** for a given task/intention. Rather than always using a massive network (full code), the system dynamically selects a compact subset of its network or rules that are relevant to the intention seed. This is like having multiple "donut recipes" and picking the minimal one that yields the desired flavor. Such an adaptive, sparse activation not only aligns with minimal code use, but often yields efficiency gains.

- **Benchmarking Expansion Ratio:** To evaluate and drive this philosophy, we establish metrics for **expansion ratio** – how much output (in terms of information content, detail, or task performance) is obtained per unit of input instruction or model size. For example, we might measure Kolmogorov complexity of outputs vs inputs, or measure the performance of increasingly pruned models to see how low we can go without loss. A successful DoA design would consistently demonstrate high expansion ratios (like reconstructing a high-resolution signal from few samples, or achieving strong results with tiny models). One could create benchmark tasks where the input "intention" is extremely brief (a few words or a simple signal) and the score depends on the richness and accuracy of the system's generated response. This encourages solutions that **maximize output richness for minimal input complexity**.

In summary, the Donut Framework for DoA envisions a system that **stores or learns rich structure (the dough) but interacts through minimal seeds (the donut hole)**. It is a marriage of compression and generation: compress knowledge into tight representations, and unleash it via generative expansion when prompted by a small cue. Each concept we reviewed provides a piece of this puzzle. **Kolmogorov complexity and fractals** teach us that *simple programs can encode vast detail*, inspiring us to seek those simple programs. **Compressive sensing** teaches us to *find the hidden sparsity* and leverage it to reduce inputs. **Sparse neural networks (LTH)** show that *most of a large system can be redundant*, and finding the right small part can do the job – pushing us to algorithmically discover the minimal critical subset. **Cognitive predictive models** demonstrate the power of *strong internal models and focus* to infer whole realities from tiny signals, suggesting our systems should incorporate prior knowledge and selective attention.

By aligning our design with “minimal code, maximal field,” we aim for DoA systems that are not only efficient, but also elegant and insightful – they will reveal what the *essential ingredients* are for complex outcomes. Such systems could take a **“short intention seed” and generate a** “rich focus”**, much like planting a tiny acorn that grows into a sprawling oak tree. The Donut metaphor reminds us that** a tiny center can support a wide periphery** – by design, our algorithms should do the same, giving us maximal insight and output from minimal guidance.

Sources:

- Kolmogorov complexity and MDL principle [1] [19] ; fractal example (Mandelbrot set compression) [4] .
- Compressive sensing fundamentals [5] [8] .
- Lottery Ticket Hypothesis (sparse subnetworks in neural nets) [9] [11] .
- Predictive coding in the brain (top-down predictions, errors) [13] ; Attention as precision of prediction error [14] .
- Thin-slicing phenomenon (accurate judgments from brief clips) [17] [16] .

---

[1] [4] Kolmogorov complexity - Wikipedia
https://en.wikipedia.org/wiki/Kolmogorov_complexity

[2] [3] Minimum description length - Wikipedia
https://en.wikipedia.org/wiki/Minimum_description_length

[5] [6] Compressed sensing - Wikipedia
https://en.wikipedia.org/wiki/Compressed_sensing

[7] [PDF] Compressive sampling - Emmanuel Candès
https://candes.su.domains/publications/downloads/CompressiveSampling.pdf

[8] Single-pixel imaging - Wikipedia
https://en.wikipedia.org/wiki/Single-pixel_imaging

[9] [10] [11] [12] [1803.03635] The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks
https://arxiv.org/abs/1803.03635

[13] [14] Predictive coding - Wikipedia
https://en.wikipedia.org/wiki/Predictive_coding

[15] [16] [17] [18] Thin-slicing - Wikipedia
https://en.wikipedia.org/wiki/Thin-slicing

[19] Kolmogorov Complexity
https://www.ethansmith2000.com/post/minimum-description-length