Title
To Fix the Image in Memory: Adaptive Analog Coding in Emerging Memory Systems

Author
Zarcone, Ryan

Publication Date
2020

Peer reviewed|Thesis/dissertation

To Fix the Image in Memory: Adaptive Analog Coding in Emerging Memory Systems

by

Ryan Zarcone

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Biophysics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Associate Professor Michael R. DeWeese, Chair
Professor Bruno Olshausen
Professor H.S. Philip Wong
Professor Frédéric Theunissen

Spring 2020

To Fix the Image in Memory: Adaptive Analog Coding in Emerging Memory Systems

Abstract

To Fix the Image in Memory: Adaptive Analog Coding in Emerging Memory Systems

by

Ryan Zarcone

Doctor of Philosophy in Biophysics

University of California, Berkeley

Associate Professor Michael R. DeWeese, Chair

Motivated by (i) nature's ability to perform reliable, efficient computation with stochastic components, (ii) the end of Moore's Law (and other associated scaling laws) for our current computational paradigm, and (iii) the exponentially increasing amount of data (especially of the image variety) generated over the last decade, we examine herein the ability of analog valued emerging memory devices to directly store analog valued data. Specifically, we start by recasting the problem of data storage as a communication problem, and then use tools from the field of analog communications and show, using Phase Change Memory (PCM) as a prototypical multi-level storage technology, that analog-valued emerging memory devices can achieve higher capacities when paired with analog codes. Further, we show that storing analog signals directly through joint coding can achieve low distortion with reduced coding complexity. We then scale the problem up to store natural images on a simulated array of PCM devices. Here, we construct an autoencoder framework, with encoder and decoder implemented as neural networks with parameters that are trained end-to-end to minimize distortion for a fixed number of devices. We show that the autoencoder achieves a rate-distortion performance above that achieved by a separate JPEG source coding and binary channel coding scheme. Next, we demonstrate, this time by experiment, an image storage and compression task by directly storing analog image data onto an analog-valued Resistive RAM (RRAM) array. A joint source-channel coding algorithm is developed with a neural network to encode and retrieve natural images. This adaptive joint source-channel coding method is resilient to RRAM array non-idealities such as cycle-to-cycle and device-to-device variations, time-dependent variability, and non-functional storage cells, while achieving a reasonable reconstruction performance of ~ 20 dB using only 0.1 devices/pixel for the analog image. Finally, in an attempt to explicitly tackle device-device variation and drift, we use data from a commercial fabrication facility at TSMC and demonstrate preliminary results showing the ability to create an effective drift model capable of inferring values stored at previous times.

$$f\left(F_7^{63}\left(10^{100}\right)\right)$$

$$numbers \xrightarrow{f} epigraph\ quality$$

Contents

List of Figures

List of Tables

Acknowledgments

To my friends, family, and mentors:[1] I cannot begin to thank you for all you have done for me on this journey. You've supported me physically and spiritually, and I hope that I can return your kindness in the years to come. When I was deciding where to go for grad school, a professor told me that choosing a grad school is like choosing a spot to set up camp. It's a really important decision, but more important than the spot is who your camp-mates are. You're the best camp-mates I could have asked for, and I'm looking forward to continuing this adventure with you. Let's go!

[1] I apologize to all of you. As I'm sure will be no surprise to any of you, I'm writing this almost literally at the eleven-and-a-halfth hour and thus do not have enough time to properly thank you here. I would have liked (and honestly did have grand plans) to thank each of you personally here – a la the great Acknowledgment sections of the theses of my good friends Charles Frye and Mayur Mudigonda – but alas, clichés[I] are all I'll have time for.

[I] But please don't let the banality of these statements diminish the intent behind them. Clichés exist for a reason, and I really do mean what I say here.

Abstract

To Fix the Image in Memory: Adaptive Analog Coding in Emerging Memory Systems

by

Ryan Zarcone

Doctor of Philosophy in Biophysics

University of California, Berkeley

Associate Professor Michael R. DeWeese, Chair

Motivated by (i) nature's ability to perform reliable, efficient computation with stochastic components, (ii) the end of Moore's Law (and other associated scaling laws) for our current computational paradigm, and (iii) the exponentially increasing amount of data (especially of the image variety) generated over the last decade, we examine herein the ability of analog valued emerging memory devices to directly store analog valued data. Specifically, we start by recasting the problem of data storage as a communication problem, and then use tools from the field of analog communications and show, using Phase Change Memory (PCM) as a prototypical multi-level storage technology, that analog-valued emerging memory devices can achieve higher capacities when paired with analog codes. Further, we show that storing analog signals directly through joint coding can achieve low distortion with reduced coding complexity. We then scale the problem up to store natural images on a simulated array of PCM devices. Here, we construct an autoencoder framework, with encoder and decoder implemented as neural networks with parameters that are trained end-to-end to minimize distortion for a fixed number of devices. We show that the autoencoder achieves a rate-distortion performance above that achieved by a separate JPEG source coding and binary channel coding scheme. Next, we demonstrate, this time by experiment, an image storage and compression task by directly storing analog image data onto an analog-valued Resistive RAM (RRAM) array. A joint source-channel coding algorithm is developed with a neural network to encode and retrieve natural images. This adaptive joint source-channel coding method is resilient to RRAM array non-idealities such as cycle-to-cycle and device-to-device variations, time-dependent variability, and non-functional storage cells, while achieving a reasonable reconstruction performance of ~ 20 dB using only 0.1 devices/pixel for the analog image. Finally, in an attempt to explicitly tackle device-device variation and drift, we use data from a commercial fabrication facility at TSMC and demonstrate preliminary results showing the ability to create an effective drift model capable of inferring values stored at previous times.

Chapter 0

Introduction

0.1 Motivations

0.1.1 Scientific: Perform Computations with Stochastic Components

Nature. Much wow.[2]
Cells are astoundingly energy-efficient sensing and actuating machines. Cells process their electrical, chemical, and mechanical inputs with highly noisy and imprecise parts. Nevertheless, they perform highly complex, sensitive, and collectively precise hybrid analog-digital signal processing such that reliable actions are carried out [105].[3]

If we look specifically at neurons, bodies' "computational hardware", the comparison to our own engineered systems is quite remarkable. To quote (very rough numbers from) Carver Mead[4] [78]:

> [The] ultimate silicon technology that we can envision today will dissipate on the order of 10^{-9} J of energy for each operation... We can compare these numbers to the energy requirements of computing in the brain. There are about 10^{16} synapses in the brain. A nerve pulse arrives at each synapse about ten times/s, on average. So in rough numbers, the brain accomplishes 10^{16} complex operations/s. The power dissipation of the brain is a few watts, so each operation costs only

[2] I'll be breaking this dissertation up into many subsections so that the one person[I] who reads this can skip around to the parts they care about.

[3] See this reference for some great examples of biological efficiency.

[4] John Von Neumann was one of the first to realize that neurons must operate at a fairly low level of precision, and that no known computing machine could (or can) operate reliably on such low precision. Von Neumann called this "a deterioration in arithmetics for an improvement in logics" [120].

[I] I've been repeatedly told that it's likely no one will ever read your thesis. If this is the case, then a typical thesis, I suppose, is a graduate student talking to themselves.

> 10^{-16} J... [The brain is] a factor of 10 million more efficient than the best digital technology that we can imagine.

We have far to go

If we look at our own computational paradigm, there's a massive gulf between where we are and what biology is capable of. But it doesn't end there. Biology isn't the only thing that computes. We can generalize some of our thinking about computation and look at nature as a whole. The computational mechanics archive summarizes the situation well [24]:

> One quickly comes to the conclusion [then] that contemporary notions of *computation* and of *useful information processing*, colored as they are by the recent history of digital computer technology, must be extended in order to be useful within empirical science. Why? Because the processes studied by natural scientists involve systems that are continuous, stochastic, spatially extended, or even some combination of these, and these characteristics fall strictly outside the purview of discrete computation theory.

Part of the progress towards approaching nature's computational abilities has come (and will continue to come) from realizing the importance of error (i.e. "stochasticity") considerations from the start. Von Neumann's view was exactly this [119]:

> Our present treatment of error is unsatisfactory and ad. hoc. It is the author's conviction, voiced over many years, that error should be treated by thermodynamical methods and be the subject of a thermodynamical theory, as information has been by the work of L. Szilard and C. E. Shannon.

Looking forward

Part of the solution to "catch up" with nature will be to use physics (or more generally the dynamics of systems) to do computation for us. This isn't a new idea. Many of those involved in the founding of computing and those studying the future of electronic systems have discussed this in some form or another. See [119], [120], [112], [33][5], and [78] for some great discussions.

A lot of work has been done on trying to understand the trade-offs between computations and the resources used for said computations. One paper that I think lays a nice intuitive foundation for this is "A universal tradeoff between power, precision and speed in physical communication" [74]. In addition to this, a recent review paper, "Thermodynamic Computing" [22], summarizes a lot of the great work that has happened in this area over last few

[5] A decent dead canary for a paradigm mismatch is when you see terrible scaling (something we see in simulating many physical systems, including quantum and biological). Also see [1] for a great discussion of the importance of scaling considerations.

decades.[6]

If we take a look again specifically at biological systems, we might be able to find further inspiration. Living systems continuously adapt to *embody* efficient general solutions to the problems they encounter . As a particular instantiation of "using physics to compute", some roboticists have taken inspiration from animals in designing robots for specific motor tasks. This idea, "morphological computation", uses the physical body as a computational resource for the problem at hand [41].[7]

Looking lower, as already mentioned, cells have evolved complex biochemical reaction networks for doing all sorts of computations. As we begin to understand the specifics of these processes in more detail, we'll (potentially) be able to engineer complex cellular computing systems [128][8].

Finally, specifically looking at specialized biological computational machinery (i.e. neurons/brains), there's been progress on trying to understand the origin of these systems' efficiency. There's been a lot of great work in this area, but two pieces that stand out to me are Raoul Sarpeshkar's "Analog Versus Digital: Extrapolating from Electronics to Neurobiology" [99] and Sterling and Laughlin's book "Principles of Neural Design" [108]. Both of these take, in some sense, an engineer's perspective on the brain. And, as I too believe that what one cannot create one does not understand, this perspective smells like one of the best ways to understand brains.

Charitably, each of these works summarizes the principles they've uncovered. And, as I think these are sufficiently important to always keep in mind, I'll reproduce the main ones here (and with that end this long, prosaic rant on performing computations with stochastic components). Please see Table 1.1[9].

0.1.2 Engineering: End of Moore's Law ⇒ Need to Move Away from Discrete, Deterministic Paradigm

Here, I'd like to summarize from two papers and use them as frames (and as jumping off points for some other key papers) for the current state of the computational electronics field-/industry.

Thermodynamic Computing [22]
In the 1960s two "laws" regarding the current computing paradigm were observed. The first was Moore's Law, in which Gordon Moore noted that the number of transistors on a chip doubles approximately every 18 months. The second was Dennard Scaling, an observation by

[6]For those who are strapped in and ready to spelunk down in the 10-dimensional rabbit-hole, I suggest looking at [9], [124], and [125].

[7]See [56] (and check out the company founded by the last author, *Agility Robotics*) for an awesome example of how exploiting the interplay between passive dynamics and control can yield robust locomotion.

[8]And see [47] for a great review.

[9]Oh exquisite table, if only I had had more time to make you... A lightning flash. Too late! Ô toi que j'eusse aimée, ô toi qui le savais!

Neurobiological Principles to Remember	
Analog vs Digital	Principles of Neural Design
∗Analog computation is efficient at low-precision processing, and digital computation is efficient at high-precision processing.	∗Compute with chemistry
∗The advantages of analog computation arise from its exploitation of physical primitives for computation. The advantages of digital computation arise from its multiwire representation of information and information processing, and from its signal restoration properties.	∗Compute directly with analog primitives
∗Analog computation that distributes its precision and processing resources over many wires is maximally efficient at a certain signal-to-noise ratio per wire, due to the trade-offs between computation and communication.	∗Combine analogue and pulsatile processing
∗In neurobiological systems, where communication costs are relatively low compared with communication costs in silicon, the optimal signal-to-noise ratio per wire is lower than that in silicon. Thus, we believe that nature was smart to distribute computational resources over many noisy neurons (dendrites and somas) and communicate information between neurons over many noisy wires (axons).	∗Sparsify (only small subset of resources used for given situation)
∗Since the brain appears to be extremely efficient in its information processing and hybrid representations are the most efficient representations in massively complex systems, it is likely that the brain uses hybrid representations.	∗Send only what is needed
∗From numerous examples, it is qualitatively clear that in neurobiology, the reduction of noise is accomplished through resource consumption as it is in electronics. Neurobiology and electronics behave similarly because physical and mathematical laws such as the laws of thermodynamics and the law of large numbers do not change with technologies. It is such laws that, with a few technology-dependent parameters, determine noise resource equations.	∗Send at the lowest acceptable rate
	∗ Minimize wire ∗ Make components irreducibly small ∗ Complicate (to optimize) ∗ Adapt, match (capacity), learn, and forget

Table 0.1: Summary of main principles from [99] and [108].

Robert H Dennard that as transistors reduced in size the circuits designed with them would increase in speed. However, in the 60 years since these laws were recorded, we have begun to see the limitations of them in the computing paradigm. As devices have become smaller and smaller, we have struggled to eliminate the effects of thermodynamic fluctuations that occur at the nanometer scale, driving up the power densities and cost of building next-generation semiconductor chips, slowing Moore's Law, and eventually ending Dennard Scaling. This performance limit is known as the "power wall", and our inability to breach this wall has

killed the advancement of microprocessor performance.

This paper claims that Moore's Law is still kicking, but I think a better framing of the law (as it's inextricably intertwined with economic motivations) is the number of transistors/$. And, as indicated by an article in the Economist, "After Moore's Law" [29], this law is dead.

The main reasons for this death are summarized nicely in "A Neuromorph's Prospectus" [10]: You can think about the surface of a transistor as a road that cars (electrons) travel down (see Fig. 0.1). As transistors shrink to nanometer scales, the number of "lanes" of traffic shrinks to single digits. Dangling bonds at the silicon-silicon-dioxide interface of transistors are "traps" for these electrons. If thermal fluctuations can break these bonds quickly, the traps yield stochastic transistors (i.e. the current through the device fluctuates stochastically). If the bonds aren't broken quickly, the traps yield heterogenous transistors (i.e. the current going through a given device will vary depending on trap density). Either of these types of error in a transistor with a few lanes would be catastrophic for a computer. To address this trap problem, the fabrication industry switched from a planar design to a 3D "fin" design (see Fig. 0.1). This increased the "lanes" but significantly increased the cost of production (hence the dip seen in 2014; inset of Fig. 0.2). But this strategy is not scalable. The smallest transistor theoretically possible with current technology would have an 4% chance of having one of these traps. Such an error rate would be insurmountable in the current computing paradigm.

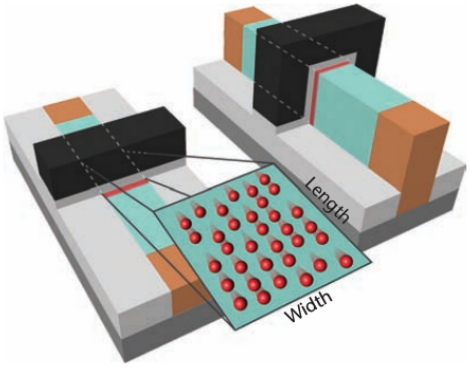

Fig. 0.1. Planar (left) vs. 3D "fin" design (right) for transistors. Electrons (red) travel along surface in "lanes". © 2017 IEEE. Reproduced with permission. Original source: [10]

Or, in the words of Jan Rabaey and Sharad Malik [91]:

> As the critical dimensions of CMOS devices reach the nanometer range, we are approaching some fundamental limits, making routine technology scaling no longer a viable solution. This will force the adoption of drastically different strategies for manufacturing, device engineering, and design creation. The spiraling cost associated with some of these technologies challenges an important corollary of Moore's law: that the cost per function is reduced proportionally with technology

scaling. If this no longer holds, the economic drive behind the continuation of technology scaling will quickly vanish.

See Fig. 0.2 for a great summary of all the above trends.

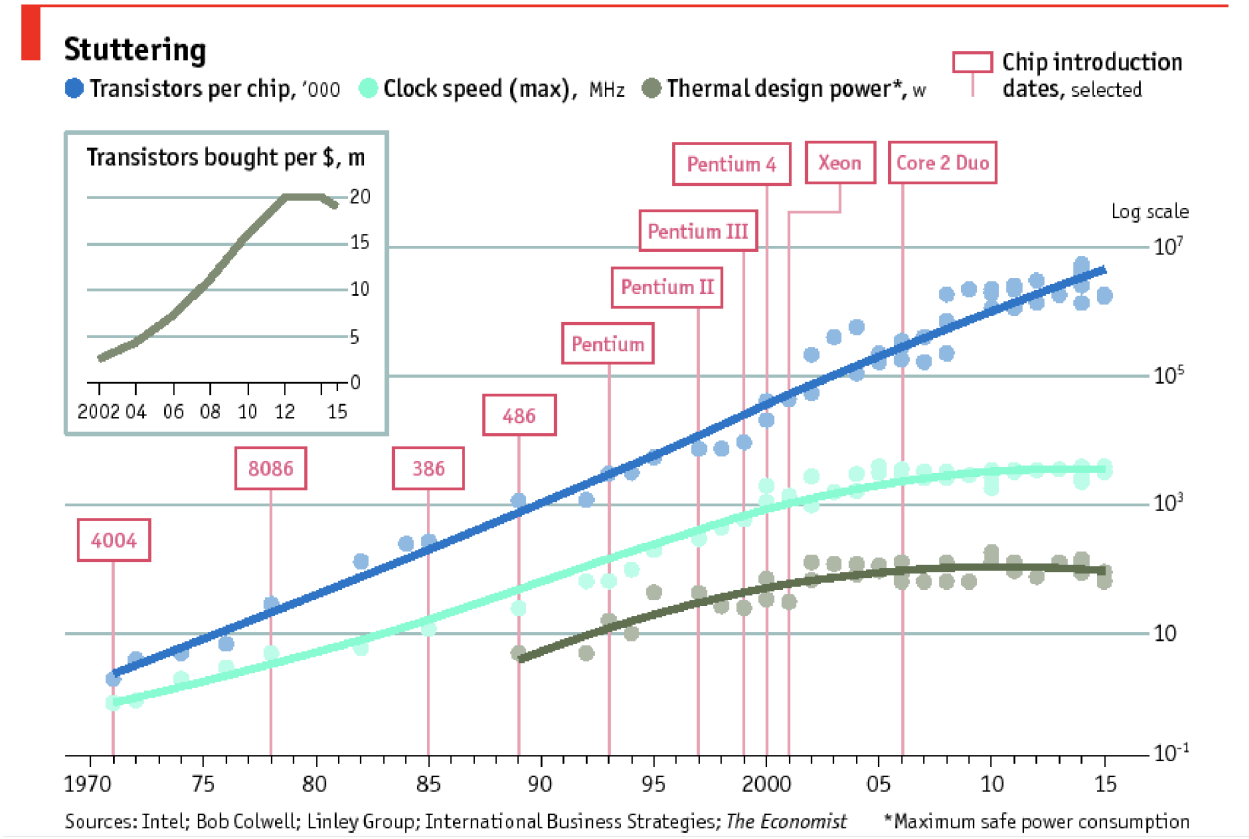

Fig. 0.2. Plot of scaling trends for different technologies summarizing the main points of the corresponding section: Denard scaling laws have stopped while the number of transistors per dollar has actually decreased. © 2016 The Economist. Reproduced with permission. Original source: [29].

The Institute of Electrical and Electronics Engineers (IEEE)'s Rebooting Computer Initiative is focused on addressing the issue of slowing computer power by understanding how different approaches to computation impact energy efficiency on various levels of the computing stack, from devices to algorithms. Their observations are in Fig 0.3 below, which shows four different approaches to the future of computing, and the relative level of disruption those approaches would cause to different levels of the computing stack.

Challenges and Solutions for Late and Post Silicon Design [91]
Interestingly, we've already started to see the "complicating to optimize", mentioned previ-

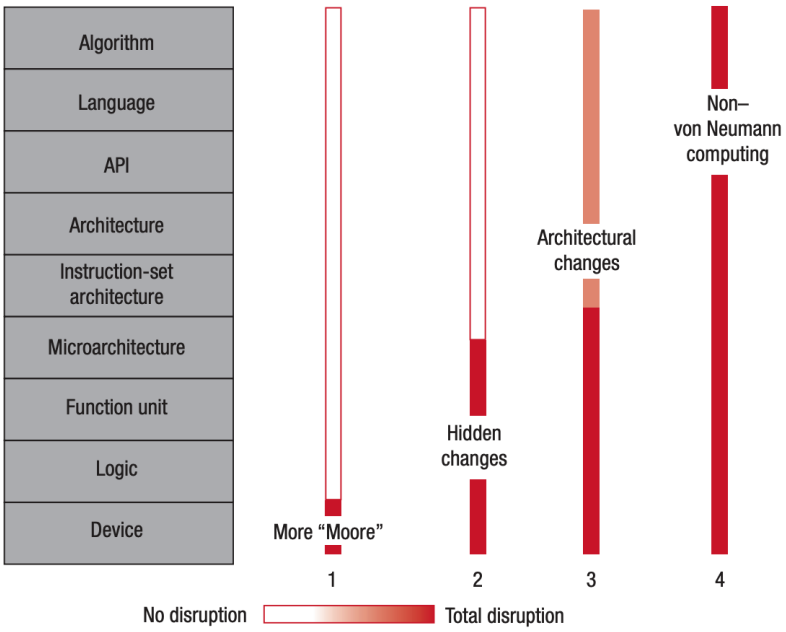

Fig. 0.3. In response to the slowing of long cherished scaling laws (see Fig. 0.2), IEEE started the Rebooting Computing Initiative in 2012 to rethink computing all the way from devices to algorithms. This is an illustration of how different approaches to the future of computing could change different levels of the computing stack. Computing with stochastic components is a "level 4" approach. Reproduced with permission. © 2017 IEEE. Original source: [21]

ously by Sterling and Laughlin as principles of neural design, in our computing hardware: As laid out by Rabaey and Malik:

> Specifically, integrated systems are becoming increasingly more heterogeneous, combining various computation and communication paradigms. On the computation side, this includes hardwired and programmable logic blocks as well as domain-specific and general-purpose programmable processors. On the communication side, this includes traditional busbased architectures, emerging packet-switched networks on a chip, and memories with varying properties. In addition to these, analog (including RF) subsystems enable external signal interfaces. In light of the divergence of the technology roadmap we have just discussed, an integrated system could consist of more than one chip in a single package.

From this perspective, the approaches we'll be discussing later are very much in line with emerging trends in computing (i.e. specializing hardware/software to optimally perform

specific, but common, tasks): power considerations are already pervasive throughout design considerations and can no longer be separated from the overall design flow. This means that performance metrics are transitioning from things like MIPS – million of instructions per second – to things like MIPS/Watt .

In this work, Rabaey and Malik describe a road-map for addressing the "long-term woes of electronic systems design" with various time horizons. The two most related directions for our work here are "The Age of Error Resiliency" and the "Age of Randomness". Below, I'll reproduce the sections most relevant to the current discussion:

The Age of Error Resiliency :

- Systems will experience dynamic variations over time and will be subject to numerous failure mechanisms
- Adaptability to changing conditions and resiliency against failures and operating-condition variations must be intrinsic properties of the next generation of computation and communication systems
- Without these safeguards, the design will be uneconomical or the product's lifetime will be too short to be of any value
- Note that perfect adaptivity and resiliency are not necessary at every single individual level, and can be accomplished by cooperation between different levels of the design stack. Individual levels (such as device or logic) can occasionally fail as long as the overall system shows correct behavior

The Age of Randomness (The far beyond) :

- Ultimately, the combined factors of variation and reliability will be such that maintaining traditional computational models – such as purely deterministic Boolean logic – becomes untenable
- The only possible solution at that point is to slowly but surely abandon them in favor of statistical models that inherently support error-resiliency
- The domains of signal-processing, communications and information theory provide ample examples on how such models could work
- It is further our conjecture that such computational models will be of essence in the post-silicon nano-device age with its promise of uncountable devices, operating under conditions of tiny noise margins, random variations and unreliability in computation

Going further and looking at future trends, we're seeing a fundamental shift in the kinds of devices being manufactured and deployed. We're moving to a model where an infrastructural core (e.g. servers) talks to a network of mobile devices (e.g. smartphones) which in tern talk

to a dense network of devices in a "sensory swarm" (e.g. fitness trackers) – potentially more than a trillion sensory nodes per year will be deployed in the near future, with the majority of these communicating wirelessly [92] (see Fig. 0.4[10]).

This culminates in what has been called Ambient Intelligence [25]:

> ...the 'embedded-everywhere' world in which all objects around us become intelligent microsystems interacting with each other, and with people, through wireless sensors and actuators... a vision of a world in which people will be surrounded by networked devices that are sensitive to, and adaptive to, their needs....The design of [these] wireless-transducer-network devices requires creative engineering to get to the ultimate limits of miniaturization, cost reduction, and energy consumption. This leads to the need for "more-than-Moore" that is a cost-effective integration of CMOS with MEMS, optical- and passive- components, new materials, bio-silicon interfaces, lifelong autonomous energy sources, and grain-size 3D packaging. The complexity is not in the number of transistors, but in combining technologies, circuit- and global- networking architectures to obtain utmost simplicity for the sensor nodes themselves.

A final tangential thought: while we're certainly approaching this futuristic, (scientifically-) romantic vision, we must remember that we're far from being able to effectively engineer these swarms (i.e. to call them "intelligent"). The Michael Jordan of computer science [11] has a great article where he discusses challenges we'll face in building this "Intelligent Infrastructure" [59]:

> ...the principles needed to build planetary-scale inference-and-decision-making systems of this kind, blending computer science with statistics, and considering human utilities, [are] nowhere to be found... Whereas civil engineering and chemical engineering built upon physics and chemistry, this new engineering discipline will build on ideas that the preceding century gave substance to, such as information, algorithm, data, uncertainty, computing, inference, and optimization. [This] "Intelligent Infrastructure", whereby a web of computation, data, and physical entities exists that makes human environments more supportive, interesting, and safe ... is beginning to make its appearance in domains such as transportation, medicine, commerce, and finance, with implications for individual humans and societies. This emergence sometimes arises in conversations about an Internet of Things, but that effort generally refers to the mere problem of getting 'things' onto the Internet, not to the far grander set of challenges associated with building systems that analyze those data streams to discover facts about the world and permit 'things' to interact with humans at a far higher level of abstraction than mere bits.

[10] Press F to pay respects for the flip phone.

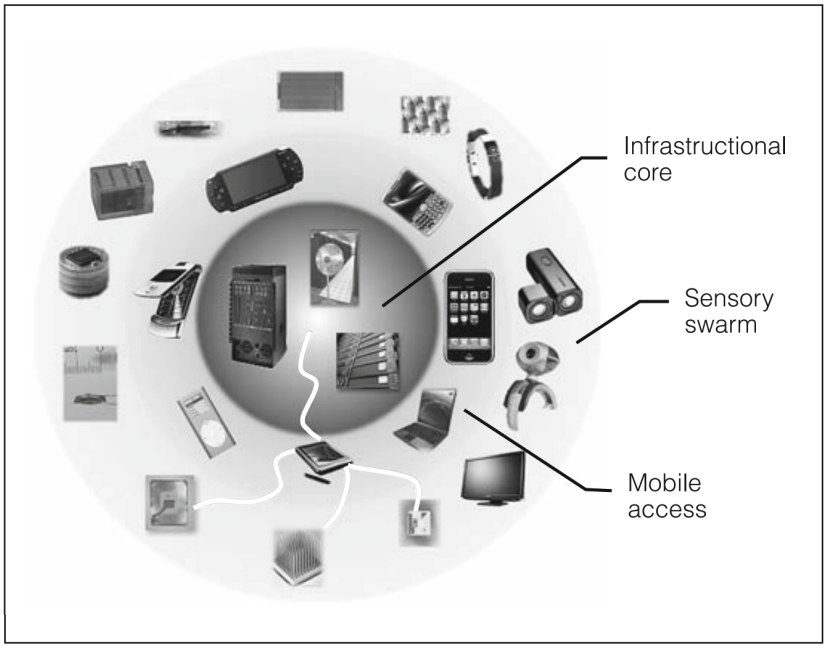

Fig. 0.4. Evolving computational device ecosystem. Importantly, a (growing) majority of the devices are at the "edge" or part of the "sensory swarm". Reproduced with permission. © 2008 IEEE. Original source: [92]

0.1.3 Focus: Storage as a Particular Type of Computation

You have to start somewhere

Considering (i) communication is a fundamental prerequisite for more complex computations [74] and (ii) there's a whole field (that's over 70 years old) dedicated to the study of communication, communication seems like a great place to start.

On a more practical note, there's been an exponential increase in the amount of data humans have generated over the last few decades (see Fig. 0.5). More specifically, a paradigm shift in the type and quantity of storable information is currently underway. Internet-connected devices are projected to reach 50 billion, more than 6 devices per person, by the year 2020 [32]. Much of the data produced by these devices, such as pixel intensities from cameras, sound recordings from microphones, and time-series from sensors, comes from signals that are intrinsically analog-valued or many-valued (> 100 values) and ordinal. These signals are also highly redundant and compressible, with a large degree of their joint activity explained by a smaller number of factors than the intrinsic dimensionality of the signal. Furthermore, an increasing portion of the operations performed on the data are analog, with much of it being used for statistical inference or human perception (e.g. object-detection for images).

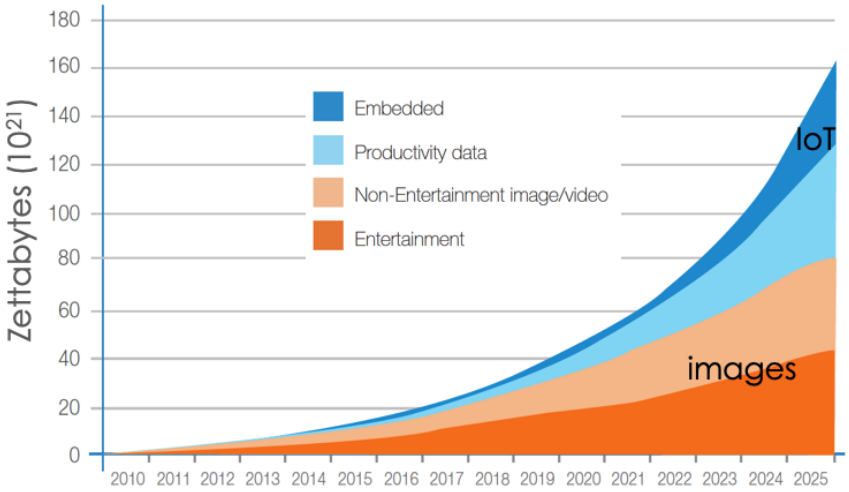

Fig. 0.5. Plot showing the exponentially increasing amount of data generated per year, by type. Importantly, the majority of this data is image data, (in either static or video form) and a growing portion is from embedded or "Internet of Things" devices (with much of this data being natively continuous). Reproduced with permission. IDC White Paper, sponsored by Seagate, Data Age 2025: The Digitization of the World from Edge to Core, November 2018)

A concurrent paradigm shift is underway in the media we use to store data. With this rapid increase in the amount of data generated has come an increase in research aimed at designing and characterizing smaller and more power-efficient devices for information storage – i.e. "emerging memory devices". Early storage media such as phonograph records and VCR tapes relied on perturbing an analog-valued state (wax height and magnetic polarization, respectively). Digital computation led to the popularity of binary storage representations that inhibit noise propagation and utilize the concurrently developed theories of binary error-correcting codes [76] (see Table 0.1). However, many emerging memory technologies have shifted back to analog-valued media to create multi-level devices that fill the need for inexpensive, high-density storage. MLC-Flash, Phase Change Memory (PCM), Resistive RAM (RRAM), and Conductive Bridge RAM (CBRAM) are all examples of technologies that have an analog state (threshold voltage or resistance) determined by the gate-charge, resistive amorphous capping region, and conducting filament, respectively [127], [126]. One of the goals of many researchers and developers studying emerging memory technologies is to create a "Storage Class Memory" [16]. This is, essentially, a system of memory that combines the benefits of storage systems like HDDs (e.g. high density, long life) with the benefits of "high performance" systems like DRAM (e.g. fast access time). From this traditional perspective, the emerging memory devices we will be considering in this work are approaching a Storage Class Memory system in that they have the ability to reach high

densities (through 3D integration) with very fast access times (∼100 ns).

The aforementioned trends of (i) analog data generation and operation and (ii) analog storage media development motivated us to investigate the use of analog coding in emerging memory systems. We examine here the potential for memory systems that utilize analog representations – analog-valued signals stored in analog-valued devices – to achieve higher capacity and reduced coding complexity, leading to reduced latency and increased energy efficiency.

0.1.4 Specifically: Image Storage

The most abundant form of data stored and processed on electronic devices is now image data, in both static and video form, and this dominance is only expected to increase [30] (see Fig. 0.5).

At the same time, the scientific field dedicated to studying natural images has made progress on understanding natural image structure. There's a long history of research that's made progress on modeling the structure found in natural images – see [98], [107], [137], and [57] for some great overviews.

Virtuous Loop
Scientific → Practical: an implicit (and sometimes explicit) hypothesis of neuroscientists studying natural image statistics is that brains evolved to take advantage of the structure of the world said brains are embedded in [34]. Thus, by learning about the structure of said world, neuroscientists might gain some insights into the functions of brains. As will be discussed later, compression essentially is the process of removing structure. Thus, if we're aware of some structure present in the data we want to compress, we can explicitly remove that structure to get a reduced representation. This has happened with images (and video) over the years, with understandings in spatial (or temporal) structure leading to advances in compression (e.g. Study of Gabor wavelets and the development of JPEG 2000 [93]). Practical → Scientific: (Essentially) simultaneously, the desire to perform compression has lead to a better understanding of natural image statistics. See, for example, the development of color spaces and the psychophyiscs of color vision [60][129][53].

0.1.5 Motivation: TL;DR:

Scientific understanding → engineering considerations → specific case of storage → specific case of image storage [11]

- Scientific: understanding computations with stochastic dynamical systems
  - Nature's amazing computational efficiency comes from using the massively-parallel

[11] Alas, many of us grad students end up grasping only the smallest of nutshells for the infinite space we call ourselves kings of, despite the enormity of the giants we stand on.

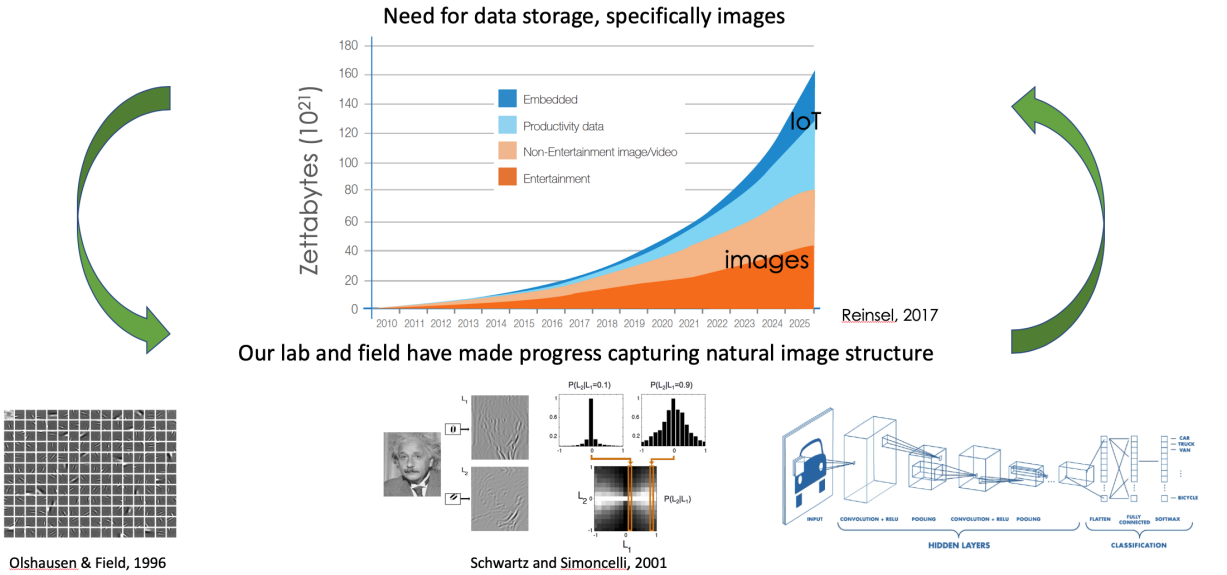

Fig. 0.6. "Virtuous loop" between scientists studying natural images statistics and engineers looking to compress images. Those studying natural images attempt to capture their structure (e.g. sparse coding (bottom left), divisive normalization (bottom middle), and deep neural networks (bottom right)). As compression involves the removal of structure, what is captured by those studying natural images can be used to compress images. At the same time, those seeking to compress images look for redundancies (i.e. structure) in them to try and remove (e.g. dependencies between different color channels in video). These redundancies can then be incorporated into a deeper understanding of images (and the way they are processed by creatures with visual systems; e.g. in the psychophysics of color perception). Each portion reproduced with permission. © 1998 Nature, © 2001 Nature, © 2014 MathWorks.

dynamics of in-precise systems to "compute"

- Engineering: end of Moore's Law ⇒ need to move away from discrete, deterministic paradigm
  - as we scale our computational systems down to the nanometer scale, we're realizing that physics will no longer let us remain within the traditional Turing abstraction – Maxwellian Demons are in the details
- Focus: storage as a particular type of computation
  - Have to start somewhere, and since we can think about storage as communication over time, and since communication is a fundamental prerequisite for more complex computations, we'll start with storage. But also: ∼*Big Data*∼.

- Specifically: Image Storage
  - Images (in static and video form) are the majority of our exponentially increasing data, so finding more efficient ways to store these will have practical value. As storage involves compression, and as compression is the removal of structure from a signal, and as the study of natural image statistics has uncovered some of the structure in images, we can use some of the ideas from the study of natural images to help with storage. At the same time, trying to do a better storage job might help us uncover more of the structure in natural images (something neuroscientists care about).

0.2 Main Results

0.2.1 Published Works

The first three chapters of this dissertation are a summery of results presented in three previously published works:

- Analog coding in emerging Memory Systems [132]
- Joint source-channel coding with neural networks for analog data compression and storage [133]
- Error-Resilient Analog Image Storage and Compression with Analog-Valued RRAM Arrays: An Adaptive Joint Source-Channel Coding Approach [135]

0.2.2 Storage as a communication problem; single analog variable

Here, we recast the storage problem as a communication problem. This then allows us to use ideas from analog coding and show, using phase change memory as a prototypical multi-level storage technology, that analog-valued emerging memory devices can achieve higher capacities when paired with analog codes. Further, we show that storing analog signals directly through joint coding can achieve low distortion with reduced coding complexity. Specifically, by jointly optimizing for signal statistics, device statistics, and a distortion metric, we demonstrate that single-symbol analog codings can perform comparably to digital codings with asymptotically large code lengths. These results show that end-to-end analog memory systems have the potential to not only reach higher storage capacities than discrete systems but also to significantly lower coding complexity, leading to faster and more energy efficient data storage.

0.2.3 Image storage with neural networks; simulating multi-variable case

Here, we provide an encoding and decoding strategy for efficient storage of analog data onto an array of Phase-Change Memory (PCM) devices. The encoder and decoder are implemented as neural networks with parameters that are trained end-to-end to minimize distortion for a fixed number of devices. Similar to [7], we find that incorporating divisive normalization in the encoder, paired with de-normalization in the decoder, improves model performance. We show that the autoencoder achieves a rate-distortion performance above that achieved by a separate JPEG source coding and binary channel coding scheme. These results demonstrate the feasibility of exploiting the full analog dynamic range of PCM or other emerging memory devices for efficient storage of analog image data.

0.2.4 Image storage with neural networks; experimental realization with device non-idealities

Here, we demonstrate, this time by experiment, an image storage and compression task by directly storing analog image data onto an analog-valued RRAM array. A joint source-channel coding algorithm is developed with a neural network to encode and retrieve natural images. This adaptive joint source-channel coding method is resilient to RRAM array non-idealities such as cycle-to-cycle and device-to-device variations, time-dependent variability, and non-functional storage cells, while achieving a reasonable reconstruction performance of 20 dB using only 0.1 devices/pixel for the analog image.

0.2.5 Image storage with neural networks; explicitly tackling device-device variability and drift

Here, we're using data from a commercial fabrication facility at TSMC (which has 1 Million device arrays) in an attempt to get enough data to model drift and, potentially, variation across arrays. Preliminary results show that we're able to create an effective drift model and infer original stored values. Additionally, further preliminary work shows that we might be able to learn transformations around the array to account for device-device variation (but this is still very early).

0.3 Conclusion

Motivated by (i) the search for a deeper, more comprehensive understanding of computation and (ii) the practical limitations of a significant portion of our current computational paradigm, we have explored here a new approach for storing analog-valued media in emerging analog-valued devices showing that analog coding strategies have the potential to create robust storage with relatively low complexity. Designing systems to match the statistics of

the source, channel, and task-at-hand provides unique opportunities not captured by systems that try to enforce determinism.

With the rise of machine learning applications, the nature of computation is changing, and many of the data-centric applications of the present and future do not require perfect data retrieval. With the data deluge faced by internet companies, it is often not the storage of data that is important, but the storage of information which performs well when retrieved for human or statistical machine inference (e.g., perception of audio and video). While losing some generality, representations that reduce the redundancy of signals to match the statistics of storage media and applications can be more efficient than universal representations.

In one sentence: **Guided by the idea that computing should (and will likely soon need to) match physical primitives with desired computations, we have come up with a way of storing analog data, such as images – the most abundant data in the world – on a set of emerging memory devices.**

Chapter 1

Storage as a communication problem; single analog variable

Whuh-oh! It's that
retro-muto-thingamabob!

Michelangelo

We'll first start by looking at the storage problem for a single variable (see Fig. 1.1). The key we found to making progress on this problem: recast storage as a communication problem. I.e. think about the analog memory device as a noisy channel. First step in this is to realize that what we're really dealing with then is distributions $P(S), P(V), P(R|V), P(R)$, and $P(\hat{S})$ (see Fig. 1.2). This is a general framework that will allow us to consider many, qualitatively different devices.

Fig. 1.1. General goal: Take samples S (from distribution $P(S)$) and transform them with a function F to then be sent through a single channel (the red dot). The value sent through the channel is then passed to the decoding function G which tries to reconstruct the original sample, $\hat{S}$, resulting in a distribution $P\left(\hat{S}\right)$).

This then allows us to utilize tools from the study of communications, Information The-

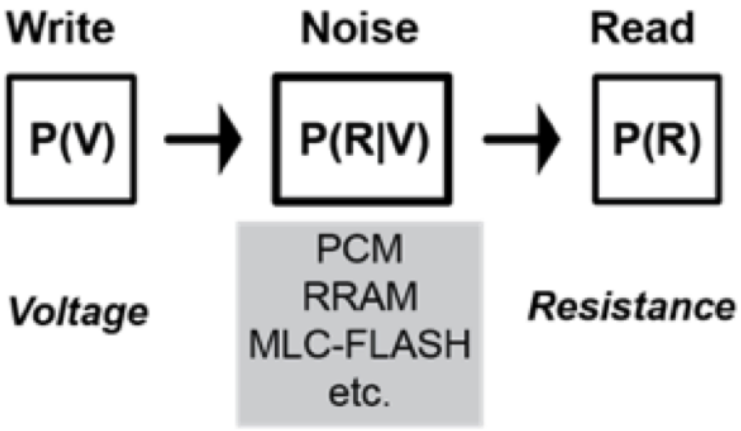

Fig. 1.2. Storage as a Communication Problem. (a) The capacity (Eq. 1) is determined by the ability to infer a write voltage pulse distribution $P(V)$ from the read resistance distribution $P(R)$. The conditional dependence between the two, $P(R|V)$, is unique for each technology and pulsing scheme. While we chose a particular technology (and therefore a particular $P(R|V)$) to illustrate the main points here, the framework could be applied to any emerging memory device (e.g. RRAM, STT-MRAM).

ory, to frame and tackle the problem. For all those following along at home [12], here's a crash course on the relevant information theory:

1.1 Background

In classic form, the gladiatorial nature of Wikipedia article contribution for popular topics has yielded some pretty good overviews of the concepts I'll be speed-outlining here. I'll mostly follow their structure here, but if you'd like more info on any one of these specific topics, I recommend starting with a combination of the Wikipedia summary, The Bible [23], and [76].

On a final note, like most foundational topics, there are many different ways to introduce/describe/illustrate them. For example, there are many different ways to introduce the concept of "information". Below I'll only give one way that fits nicely into the framework we'll be working with, but there are many others that might fit better for different problems (e.g if we were dealing with discrete variables for our main problem, thinking about entropy as the optimally coded message length might be more intuitive).

[12] Or, for those who aren't already on the bandwagon [103].

1.1.1 Information

We can think about the "information" of a random variable as something like "the amount that we have learned after measuring the outcome of the variable" or how "surprised" we are (or should be) when a particular outcome happens. Thought about in this way, it's essentially an alternative way of expressing the probability of an event. Shannon set this definition up to meet some basic requirements:

- A probability 1.0 outcome is completely unsurprising and thus yields zero information (e.g. we will learn nothing/will never be surprised by the outcome of coin toss with a coin that has heads for both sides).
- The less probable an outcome is, the more surprising it is and, thus, the more we learn (e.g. a coin that has $P = 0.99$ heads that comes up tails would be very surprising).
- If two independent outcomes are measured separately, the total amount of surprise should be the sum of the surprise for each (e.g. the surprise of separate coin flips should add).

It turns out that there's a unique function (up to a constant) that meets these requirements!

$$I(x) \coloneqq -\log_b [P(x)] = \log_b \left[\frac{1}{P(x)}\right] \tag{1.1}$$

The base of the log is the constant mentioned above and determines what "informational units" you'd like to use for your problem. E.g. $\log_2()$ is bits (binary units) while $\log_e()$ is "nats" (natural units). Also, though the first form of the definition is easier/cleaner to write, I think it's more helpful to think of the information or "surprisal" as the second: log of one over a probability (as opposed to a negative log which isn't as intuitive).

So, in summary, the amount of information a measurement outcome has is equal to it's "surprise".[13]

[13] Of course, we can't scrutanize these abstractions too closely. Only up the point that they are useful for our purposes. For as briefly mentioned previously, information is in some sense *physical*, and we should keep that in mind when dealing with it (especially when we're talking about doing computations with "continuous" devices).[I]

[I] Fundamentally, there's a finite amount of information that can be contained in a given space.[i]

[i] If you've chewed on the requisite amount of creative paper, this is formalized in the holographic bound, a modification of the Bekenstein bound: $S = A/\left(L_p^2\right)$, where L_p is the planck-length and A is the surface area of the membrane containing the system (i.e. surface area of a black hole the size of the system).[A]

[A] That's all to say, don't think you can perform hypercomputation just because you've got the magic of the $\mathbb{R}$s on your side.

1.1.2 Entropy

The entropy of a random variable is the expected amount of surprise over all possible outcomes:

$$H(x) \coloneqq \mathbb{E}[I(x)] = \sum_x P(x) I(x) \tag{1.2}$$

So, for example, the entropy of a fair coin is 1 bit (i.e. you get 1 bit of information each time you measure it).

1.1.3 Mutual Information

The mutual information between two random variables is the amount of information you get about one random variable by observing the other (i.e. how much information is "shared" between the two).[14]

$$I(X;Y) \coloneqq \sum_x \sum_y P(x,y) \log \left[\frac{P(x,y)}{P(x) P(y)} \right] \tag{1.3}$$

Some sanity checks: (i) if X and Y are independent, they should share zero information. (ii) if X is a deterministic function of Y (or vice versa), then all the surprise should be contained in one of the variables (so the mutual information should just be the entropy of X (or Y). These check out (exercises left to the two people who read this).

1.1.4 Channel Capacity

In information theory, a channel is defined as the conditional distribution between two variables: $P(Y|X)$, where X is defined as the "input" variable and Y is defined as the "output" variable. Given a source distribution over X and a channel (i.e. a $P(Y|X)$), the channel capacity is defined as the maximum amount of mutual information between two variables over all possible input distributions:

$$C \coloneqq \max_{P(x)} I(X;Y) \tag{1.4}$$

Another way of thinking about the capacity is as the maximum amount (over all possible input distributions) of information you can communicate over a channel (this will be important later).

[14] Things get pretty hairy/ill-defined if you start talking about more than two variables, so we'll swiftly sweep that under the rug.

1.1.5 Rate Distortion

I've avoided the discussion of continuous variables up until this point to avoid having to discuss measures and such , but it's important here: the magic of the $\mathbb{R}$s implies that we need an infinitely long sequence of numbers to specify a sample from, say, (you guessed it) a Gaussian distribution. But no one has time for that (or the resources), so we need to approximate this. Essentially, rate-distortion tries to analyse the trade-off between how well you describe the outcome of a random variable and how many resources you used for this description. I.e. it tries to quantify the trade-off between compression (your description of the variable) and distortion (how much your description distorts the outcome). The "how well" or "distortion" is up to you – you must specify a "distortion metric". So, unlike for channel capacity, rate-distortion inherently has a degree of subjectivity (you must specify what aspect of the variable you care about). Given (i) a distribution for the random variable and (ii) a particular scheme for coding the variable, the rate-distortion function tells you the best you can possibly do in terms of this trade-off:

$$R\left(D\right) = \min_{P(\hat{x}|x):\sum_x \sum_{\hat{x}} P(\hat{x}|x)P(x)d(x,\hat{x})\leq D} I\left(X;\hat{X}\right) \tag{1.5}$$

Essentially, what all these squiggles mean is the following: given some way of transforming the data X into another variable $\hat{X}$, given by $P\left(\hat{x}|x\right)$, $R\left(D\right)$ tells you the smallest representation you can have given a particular desired distortion limit. In practice, analytically calculating this ins't always possible (and doing it numerically can be challenging for high-dimensional sources), so $R\left(D\right)$ isn't known for many source-distortion pairs.

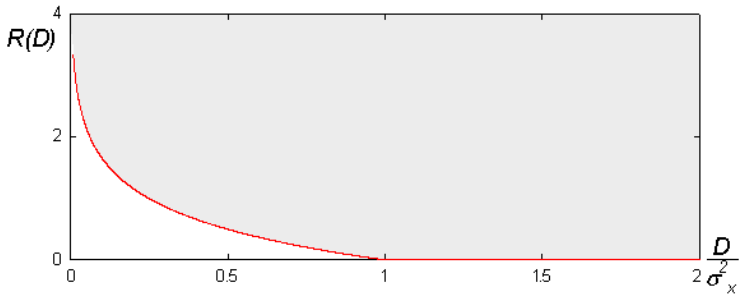

Fig. 1.3. Classic example of the rate-distortion function for a Gaussian source with variance σ^2 and squared-error distortion. Gray region (and red line) are achievable, white region is unachievable. Reproduced with permission. Original source: Rate–distortion theory. In Wikipedia, The Free Encyclopedia.

[15] Technically, there are two "Blahut-Arimoto" algorithms, one for channel capacity and one for the rate-distortion function. They're both special cases of a general iterative optimization algorithm (which also includes the expectation-maximization algorithm as a specific case), but since we only care about channel capacity here, this is the one we'll discuss. Conceptually, they're very similar.

1.1.6 Blahut-Arimoto Algorithm

The motivation for the Blahut-Arimoto algorithm[15] is that the channel capacity is difficult to calculate analytically for most channels. The idea then is to use an optimization based strategy to find the capacity (and the capacity achieving source distribution). Essentially, the trick is to first rewrite the channel capacity in the following form:

$$C = \max_{r>0} \max_{Q} \sum_{x} \sum_{y} r(x) P(y|x) \log \left[\frac{Q(x|y)}{r(x)} \right] \tag{1.6}$$

and then use an alternating optimization:

$$Q^{(k)}(x|y) = \frac{r^{(k)}(x) P(y|x)}{\sum_{x'} r^{(k)}(x') P(y|x')} \tag{1.7}$$

$$r^{(k)}(x) = \frac{\prod_y Q^{(k-1)}(x|y)^{P(y|x)}}{\sum_{x'} \prod_y Q^{(k-1)}(x|y)^{P(y|x)}} \tag{1.8}$$

An analogy I like for this comes from Raymond Yeung's "A First Course in Information Theory" [131]:

> Suppose a hiker wants to reach the summit of a mountain. Starting from a certain point in the mountain, the hiker moves north–south and east– west alternately. (In our problem, the north–south and east–west directions can be multi-dimensional.) In each move, the hiker moves to the highest possible point. The question is whether the hiker can eventually approach the summit starting from any point in the mountain[16].

Armed with these tools, we can now understand an interesting phenomenon: fractional bits of information (key to our scenario, as explained later). See Fig. 1.5.

In traditional coding, the problem is split into two phases: source coding and channel coding. The goal of source coding is to remove redundancy (structure) in the signal. The goal of channel coding is to add controlled redundancy to protect against errors that will be introduced by the channel (e.g. parity-bits to guard against a bit being flipped). These two stages of coding are typically considered in the case of "block coding": here, many samples from a source are taken and coded into "blocks" (strings of samples). E.g. let's assume we were trying to send samples from a biased coin over a binary symmetric channel (simply, a channel that will flip a bit with a specific small probability). Optimal source coding would take m samples of this coin and treat these blocks of length m as the messages to be sent. Then, an entropy-coding scheme (e.g. Huffman coding) would be used to create source coded "words" whose average length was equal to the entropy of the coin. Finally, a channel coding

[16] Answer: yes.

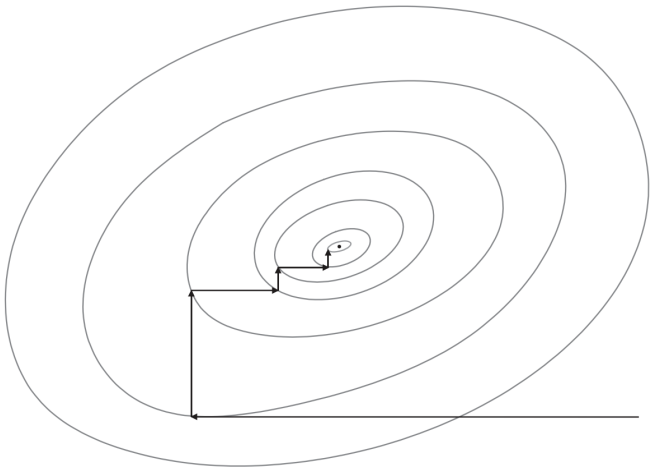

Fig. 1.4. Illustration of alternating optimization used in BA algorithm. Reproduced with permission. 2012 Springer Nature. Original source: [131]

scheme would be used to add back in bits so that the messages would be robust to bit flips by the channel.

To get very concrete: consider a coin with $P = 0.8$ heads. The entropy of this coin is $\simeq 0.72$ bits. Via a Huffman coding scheme, we can take blocks of 2 and code these in the following way: $00 \to 0$, $01 \to 10$, $10 \to 110$, and $11 \to 111$. This yields an average length per coin flip of 0.74. Close to the entropy! We could get closer if we went with larger blocklengths, but (i) the point is hopefully clear and (ii) the three people reading this probably don't care. To channel code, we could use something like a Hamming(7,4) code [121] – this takes in 4 bits and maps them to strings of 7 bits (so there are 3 parity bits). The details of this are straightforward but too long to include here, so I'll just give the intuition: let's assume that the channel has a low enough error probability to expect up to 1 bit flip per 7 uses. If we think of messages of length 4 as points in a space, the idea is that we'd like to map these to a new space where all points within 1 bit flip get mapped back to the correct 4 bit point. If we define the "Hamming distance" as the number of bit flips a message is away from another, then what we desire is a "Hamming ball" around each message such that all messages within a ball of 1 bit get mapped to one message. Thus, when the channel inevitably flips one of the message bits, the receiver will be chill as the flipped message will get mapped back to the original message. It turns out that a Hamming(7,4) code will do this by taking messages of length 4 and mapping them to a larger space of length 7. All words in this new space are at least 3 bits away, so taking one of these messages and decoding it will get back the intended original 4 bit message. So, by sending the message into a larger space, we can correct for errors by essentially making the channel behave deterministically (by separating the intended messages far enough in this new space).

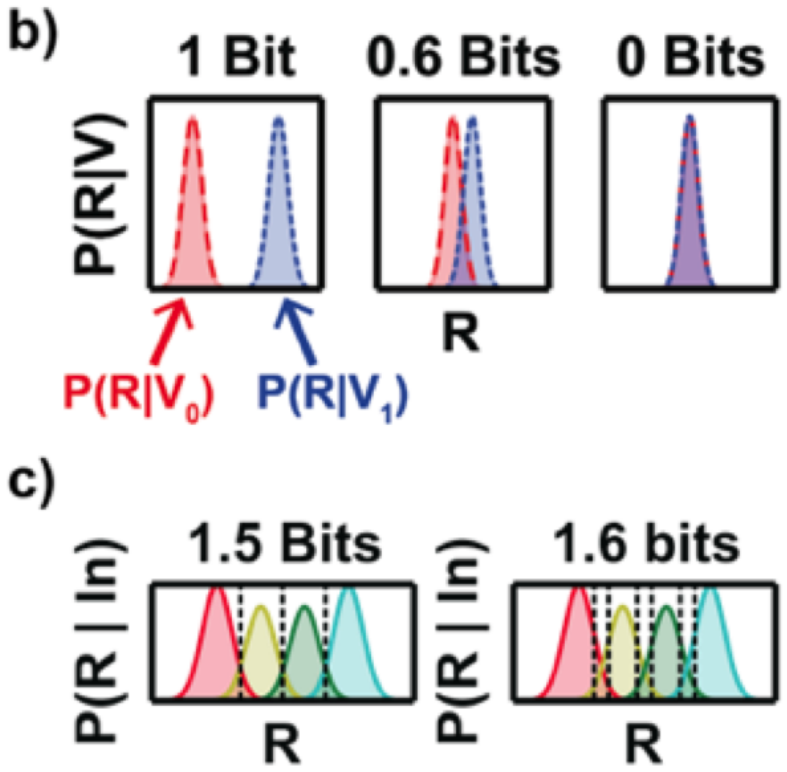

Fig. 1.5. b) Example of two state system, with input voltages V_0 and V_1. If there is no overlap between the output distributions (left), the capacity is $\log_2$ (#write states) = 1 bit. For complete overlap (right), the capacity is 0 bits, as it is impossible to infer anything about the inputs from reading the outputs. For partial overlap (center), redundancy needs to be added to the signal to correctly infer the inputs, reducing the effective bits/device. c) 'Soft information' increases capacity. For a given number of write states (4), the capacity increases with number of read states (4 left, 7 right). The read resistance is discretized into bins separated by black dotted lines. Even though there are only 4 input states, the extra read states increase the capacity of the system by providing 'soft information', the degree of belief that a read value belongs to each write value. Further read states provide greater granularity of belief values, allowing for easier inference of the input values. Analog codes, such as artificial neural networks, can use the actual resistance values, intrinsically benefiting from high granularity.

Note that what we've done here is take a single dimensional variable, take many samples to source code, and then take strings of these source coded words and channel code them by adding parity bits (putting them in a larger space). See Fig. 1.6

Importantly though, this splitting of source and channel coding is only guaranteed to

be able to reach the optimal solution in the limit of asymptotically long blocklengths (for our above example, the case where we take many many samples from the coin, source code these, and then take blocks of these and send them into a very large space so that they're sufficiently separated). Again, the intuition for why this is the case is that in traditional coding, many samples are taken so that we can jam as many words in the space as possible. Then, we try and force an approximately deterministic channel – we blow the sample up into a high dimensional space where things look deterministic.[17]

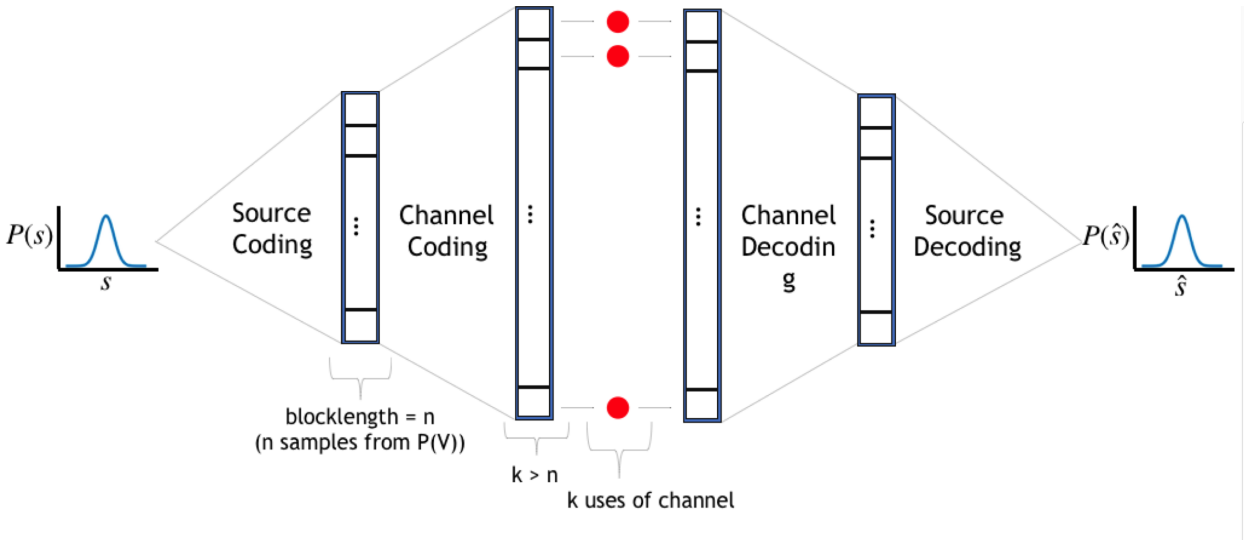

Fig. 1.6. Traditional separate approach to coding a continuous signals. In the first step, source coding, many samples are taken from the source and put into a "block". Typically, this is "vector quantization". The idea is that the signal space can be packed with many cells (see Fig. 1.7) so that, on average, the squared error between a sample and the center of the cell its assigned to is fairly small (decreasing in size with the number of cells you can pack in there). This in effect blows the signal up into a higher dimensional space as you're now considering sample sequences or *vectors*. The next step, channel coding, adds "controlled redundancy" to these sample vectors, increasing the dimensionality even further. The idea here is that given a certain level expected amount of channel noise, with enough of the right kind of redundancy, you'll be able to essentially deterministically transmit long sequences without error (the noise will only push you around so much in this high dimensional space that you have embedded the original signal). For reference, these two steps together, source encoding and channel encoding, correspond to F in Fig. 1.1 (reciprocally for G and the decoding steps).

There are two major limitations to this approach: (i) it requires long system delays and

[17]Of course, as the Library has already noted, random strings with sufficient length contain *All* - the gnostic gospel of Basilides, why the Answer is 42, and what happened to the Berenstein Bears.

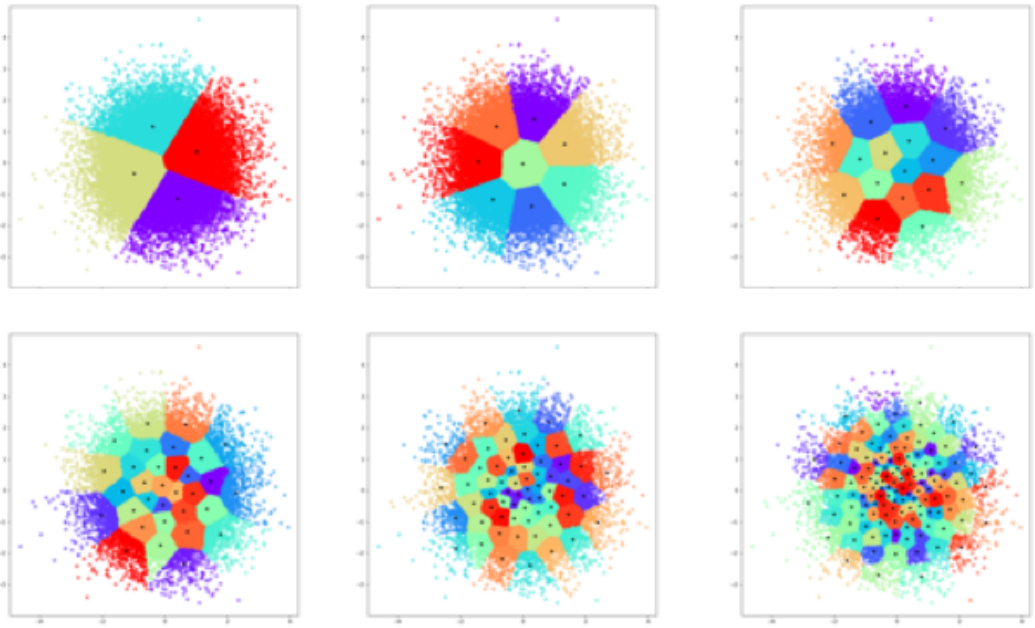

Fig. 1.7. Example of how vector quantization can work when just encoding two samples from a normal distribution (with resulting joint distribution shown here). See how "cell packing" tries to place cells in areas with largest number of samples. So the more cells you have, the finer they can be squeezed into the space.

high complexity to implement and (ii) there are no guarantees on robustness with changing channel conditions (usually does extremely poorly). As summarized by Romero et al. [97]:

> Channel codes are designed to protect data at a certain channel signal to noise ratio (CSNR). If the CSNR increases, the channel codes will over-protect the data, which leads to low efficiency. On the other hand, if the CSNR decreases, the channel codes will no longer offer sufficient protection, and this can lead to a breakdown in system performance.

How did we get here?
(Read if you'd like a short summary[18] of electronic communication/computation, skip if your time is valuable.)

[18]Though my (surely unmet) goal here is to search for the descent – not the erecting of foundations; to show the heterogeneity of what was imagined consistent with itself – here I too will deform; make groan and

A Genealogy of Electronic Communication/Computation		
Communication Over Space	Communication Over Time	ISO numeric Code
1830s: Telegraph	1860s: photopaper	1870s: Wheel-and-disc mechanical integrator
1870s: Telephone	1860s: microfilm	1890: Tabulating Machine (kernel to IBM)
1890s: Radio	890s: roll film	1930s: Mechanical differential analyzer
1920s: TV	1890s: punch cards	1930s: Turing does Turing stuff
1950s: **Semiconductor Era Begins**	1930s: LP/Vinyl	1940s: Electromechanical relay computers
1960s: Commercial Telecommunications Satellite	1940s: Williams Tube	1940s: Colossus electronic computer
1970s: Computer networks/Internet	1950s: Drum memory	1940s: Turing-Welchman electro-mechanical Bombe
1980s: Cell phones	1950s: Magnetic tape	1940s: Turing's "Proposed Electronic Calculator" report
1990s: Telecommunications networks almost entirely digitized	1950s: Hard disc drive	1940s: von Neumann's "EDVAC" report
1990s: (Digital) wireless network revolution begin	1970s: Floppy disc	1940s: ENIAC electronic computer (first general-purpose computer)
2000s: "Smart" phones	1970s: Compact cassette	1940s: Shockley and Co. invent transistor at Bell Labs
	1970s: VHS	1950s: Pilot Model ACE
	1970s: DRAM	1950s: Integrated circuit ("computer chip")
	1980s: Compact disc	1960s: First computer with GUI [20]
	1980s: Flash Memory	1960s: Microprocessor
	1990s: DVD	1970s: first personal computers enter market
	2000s: Blu-ray	1980s: explosion of personal computers
	2000s: Solid State Drive	1990s: "mobile computing" begins [21]

Table 1.1: Summary of key developments in electronic communications and computations.

For the purposes of this work, we'll consider "storage" as communication over time, so we'll also consider storage media

. Like any single, feedforward description of a highly coupled, nonlinear, recurrent [19] dynamical system – e.g. a history – this description is incomplete.

The events are a fascinating interplay between scientific, technological, and cultural products : Humans originally built communications devices (over space and time) to communicate "natural data" (i.e. data produced for and consumed by us "human computers"): images, sounds, words . But we then started to transition to symbolic representations to communicate with computers, whose representations were designed to be digital , and we thus

protest (but not explicitly tribute).

[19] While it is true that "future" events can't causally influence the "past", new information acquired in the future can change the way we see the past, and for us story-telling creatures [46], that's basically equivalent.

contorted our communications/representations to fit this model. A focus on improving IC technologies and digital signal processing lead to great advances in digital communication systems, fully realizing the "digital revolution" (e.g. Moore and Denard scaling laws). I.e. Digital communication overtook analog in large part due to advances in DSP (including quantization and error control techniques – necessary for communicating symbolic computer instructions) and the prevalence of digital processors. But, looking now and forward, as digital computation/communication is, in some sense, an extreme form of analog (in the way we do it at least), we're starting to reach the limits of what we can compute/communicate without worrying about what's "below" (e.g. the aforementioned laws are breaking down). Furthermore, as we start to, exponentially, want to go back to communicating more "cognitive data"; data about processes and systems with structure that's meaningful to human computers (or their artificial duplicates), we're finding that the way we're communicating and computing isn't the right match. End of tangent.

Back to the main point
So, in summary, the traditional approach tries to force the channel to be deterministic by spreading out the code words in a larger space, in such a way that errors introduced by the channel all get mapped to the correct source points (e.g. idea of a "hamming ball").

But there's another way of looking at the problem of coding. For many cases, we don't need errorless communication. We want to capture some statistical structure in the data for some downstream task (and it's ok if each sample is little off). Importantly, essentially any communication system is a specific case of the general mapping in Fig. 1.9. A typical digital scheme is just an extreme case – it's still just a mapping from the source space R^m to the channel space R^k and then back to the reconstruction space [3]. One can think of the goal of coding as finding a mapping that makes the input distribution "probabilistically well-matched" to the channel. Traditional coding achieves this by essentially removing the randomness introduced by the channel. But it does this often at a huge cost to complexity (something not really considered in standard information theory[22]).

Our proposal then is to take this "probabilistically well-matched" perspective and try to find efficient mappings. When we take this perspective, we think about this mapping as "Joint Source-Channel Coding" (JSCC). Here, we're considering both the statistics of the source and the channel and designing our mappings with these in mind (as opposed to the traditional approach which thinks about them separately).

Of course, this in some sense is just another of the countless instances of the engineering principle "specialize to optimize": the traditional separate approach is highly general in that you can design source coders without thinking about the specific channel to be used (and vice versa). But this comes at a complexity cost. Alternatively, you can employ a joint strategy, which can be significantly more efficient (and even outperform the separate approach when

[22]See [103], [74], [22], and [130] for some interesting thoughts on this.

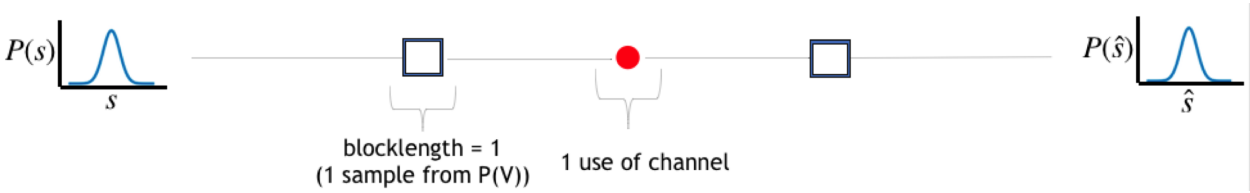

Fig. 1.8. In the joint coding case that we're considering, single symbols are mapped to the channel (i.e. blocklength = 1). Instead of trying to make the system asymptotically deterministic (like in the separate case), this joint case tries to structure the encoding function in such a way that the distribution that results as the input to the channel (what's sent in from the left side of the red dot) is probabilistically "well-matched" to the channel statistics. (see [44] for a great discussion of the details of this).

you constrain complexity[23,24]).

Below I'll give a cartoon example to illustrate the point, and then will, finally, get to what we've done. The general picture is outlined in Fig. 1.9 (we'll be doing the 1-D case for this chapter but will be scaling things up in subsequent chapters). Let's assume X is 1-D Gaussian and that the Channel is a Gaussian channel (i.e. an "Additive White Noise Gaussian" (AWGN) channel). It turns out that the capacity achieving source distribution for this scenario is a Gaussian! So the optimal solution is to do no coding (just simple scaling) and send symbols directly through[25] (Fig. 1.10). But the traditional approach destroys this favorable condition by attempting to create a deterministic channel (with high probability). So if we assume that we start off with some initial distribution that's non-Gaussian and we have a Gaussian channel, the goal then is to transform the input into a distribution that's well-matched (in this case, a Gaussian; see Fig. 1.11).

I lied 0xF0 0x9F 0x92 0xA9 (f09f92a9) ... one last thing: I'll quickly review here some of the great work people have done for analog coding – i.e. "standard communications". Again, we can reformulate the problem of storing analog data on an emerging memory device as that of transmitting a discrete-time, continuous alphabet source over a discrete-time, additive noise channel. When thought of in this manner, we can then look at the general problem of analog coding in the field of communications and see what ideas can be brought to bear on this specific problem. This field has a rich history, with the original treatment of this problem dating back to Shannon [104] and Kotelnikov [69]. Their proposed analog coding scheme was based on the use of space-filling curves. The use of space-filling curves was then significantly extended in the work of [42], [19], [94], [113], [35], [51], [52], and [55]. In these works, space-filling curves were investigated for the transmission of Gaussian

[23]See [68]; but also rememeber that there's no free lunch, and joint coding limits you to mappings tied to specific source-channel pairs.

[24]Though it should also be noted that, in general, the best source-channel schemes with finite delay, either separate or joint, may not reach the asymptotic coding bound [102], [44].

[25]See [44] and [3] for great discussions of this and extensions to higher dimensional cases.

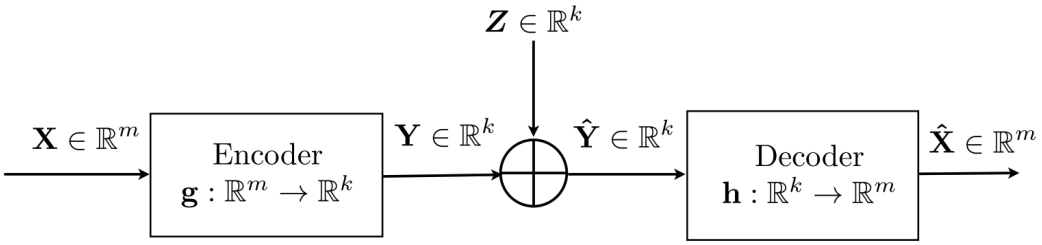

Fig. 1.9. In the most general case, a signal X of dimensionality m is encoded by an encoding function h into a signal Y of dimensionality k (where k can be less than, equal to, or greater than m). This signal is then sent over a channel which adds noise to each component of the signal. The resulting signal, $\hat{Y}$, is then sent through a decoder to be reconstructed into an estimate of the original signal, $\hat{X}$. Importantly, virtually all communication schemes are a special case of this general framework (including traditional digital approaches). Reproduced with permission. © 2014 IEEE. Original source: [3]

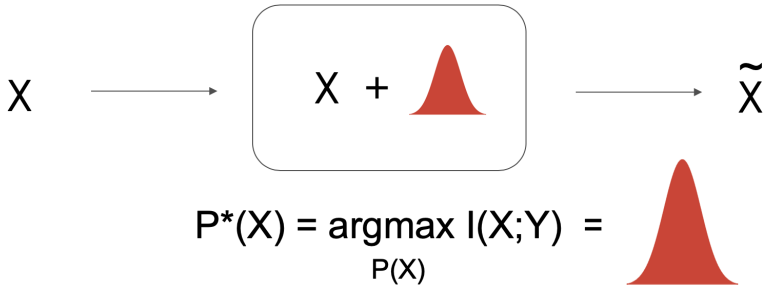

Fig. 1.10. Simple illustration of a Gaussian source going over a Gaussian channel. Here, it turns out that the capacity achieving source distribution is just a Gaussian. I.e. a Gaussian distribution is probabilistically well-matched for a Gaussian channel, so no coding is needed (just a simple scaling). See [44] and [3] for more details and some good examples.

sources over AWGN channels. These ideas, primarily using variants of the "Archimedean spiral", were further extended by [37], [36], [50], and [97] to the case of different kinds of channels. [38] looked at the operations for actually implementing the Archimedean spiral scheme and showed it was significantly more efficient for communicating over the Gaussian channel compared to the all-digital solution. Following these, a major breakthrough was made in 2014 [3] when it was shown that instead of using parameterized spirals, a functional optimization approach could be used to achieve state-of-the-art performance on the problem of Gaussian sources over AWGN channels.

So, finally, here, inspired by the general success of optimization-based methods and uti-

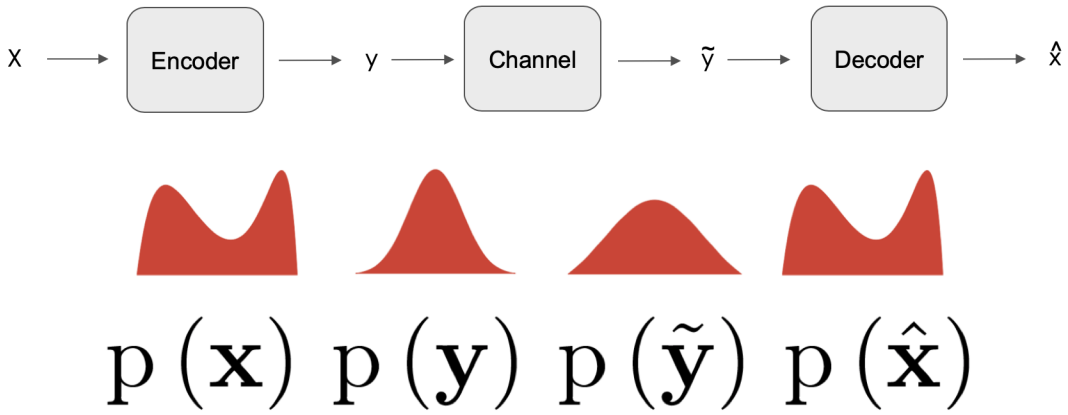

Fig. 1.11. Now let's assume the source is non-Gaussian (in the case of this illustration, some funky bi-modal distribution). The goal of coding is still the same: we must find a way to transform this distribution into something that is well-matched to the Gaussian channel. But we know from the previous example that Gaussians are well-matched for Gaussians, so this amounts to finding a way to transform this non-Gaussian channel into a Gaussian, and then from a Gaussian back into an estimate of the original.

lizing ideas from the analog communications literature to frame the problem, we show that using analog codes with analog emerging memory devices can improve system performance over digital codes in both the separate coding and joint coding regimes.

1.2 Setup

Getting past all the convoluted explanations and jargon that follows, Fig. 1.12 is the main picture to have in mind.

Capacity
The devices are perturbed by voltage pulses of different magnitudes drawn from the input probability distribution P(V). These pulses modulate a device's resistance, resulting in an output resistance distribution $P(R) = \int P(R|V)\,P(V)\,dV$. The conditional dependence of the read resistance on the write voltage, $P(R|V)$, defines the noisy channel through which information must be communicated. In the limit of an infinite blocklength code, the capacity is the number of bits of information per use of the device that can be communicated without error [6]. It is equivalently given by the maximum mutual information between the input and output distributions,

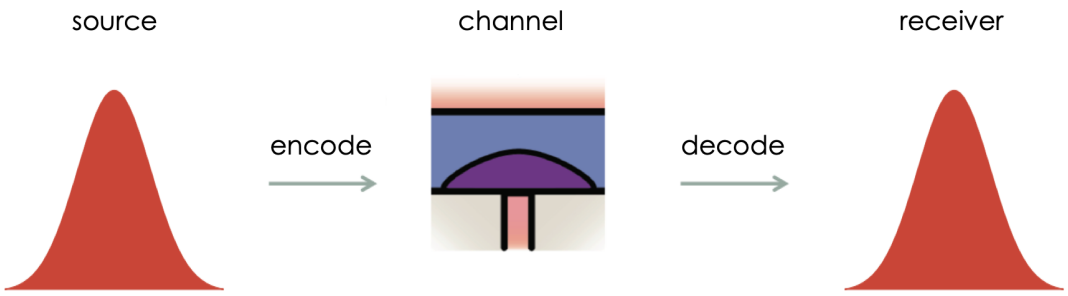

Fig. 1.12. What we want here is to take samples of a Gaussian distributed variable and store these individual samples on a single PCM device (channel) – one sample per one use of the device. E.g. let's say we drew $s = 0.1$. We then want to come up with a way of storing that 0.1 on the device such that we can later retrieve it with some degree or reliability. Importantly though, we don't care about getting the exact number right. Just that the distribution of these numbers follows the original distribution if we repeat this process with many times.

$$C = \max_{P(V)} \sum_{V} P(V) P(R|V) \log_2 \frac{P(R|V)}{P(R)} \tag{1.9}$$

The capacity achieving input distribution given by equation 1.9 exhibits an optimal trade-off between having as many input states as possible and having as little overlap in the output distributions as possible. As a simple example, consider a two-state system shown in Fig. 1.5b, with input voltages V_0 and V_1. If there is no overlap between the output distributions $P(R|V_0) P(V_0)$ and $P(R|V_1) P(V_1)$ (left panel), the capacity is the log (base two) of the number of input states (1 bit). If there is complete overlap (right panel), the capacity is 0 bits, as it is impossible to infer V from R. For cases of partial overlap, redundancy needs to be added to the signal to correctly infer the inputs, reducing the effective bits/device (center panel). Increasing the number of read states can also increase the capacity for a given number of write states (Fig. 1.5c). Practically, having more read states than write states gives additional information in the form of greyscale belief as to which input state they belong. This 'soft information' is currently used by MLC-Flash LDPC decoders to improve inference during belief propagation decoding [63]. Intuitively, analog representations that use the resistance values directly can have higher capacity because they operate at higher granularity (ultimately limited by circuit noise).

$P(R|V)$
In order to calculate capacity, we need to get $P(R|V)$.

1.2.1 Memristors (a.k.a. emering memory devices)

But first, as the devices we'll be considering are a unique type of electronic circuit element, I'll give a brief summary of what this general class of devices is and how they work.

If we're being sloppy with our terminology, we can call the devices we'll be using here "memristors" (a cheeky portmanteau of "memory" and "resistor"). The memristor was theorized by Leon Chua in the 1970s as the fourth element of a "fundamental quartet" of electronic circuit elements, along with the capacitor, inductor, and (yes) the resistor. The original elegant formulation of this quartet (see Fig. 1.13) has

$$I = G(\phi) \cdot V$$
$$\dot{\phi} = V$$

However, there are a number of works that (cogently) argue that such a device isn't physically possible (see the Wikipedia article for a great summary [122]). For our purposes though, what we really mean here is a two terminal electronic circuit element that shows a "pinched" hysteresis loop[26] in a voltage vs. current plot. The key property we care about here is that the device has a variable dependence between current and voltage, with a memory of past voltages (or currents).

On a final general note, as these are relatively new, highly non-standard channels, rigorous fundamental limits are still unknown, and coding techniques for these devices are practically non-existent.

1.2.2 Device Operation

Device operation is carried out in the following manner: as diagrammed in Fig. 1.15, a PCM material can either be in a conductive crystalline structure (Poly GST, light blue) or a resistive amorphous structure (α-GST "cap", light purple). The resistance of the PCM cell is determined by the size and structure of the resistive amorphous cap, with larger caps yielding higher resistances. The resistance is set to a high resistance state by a short current pulse (Fig. 1.15, top panel, left-most light blue pulse) that first melts the film through Joule heating. The short pulse then quickly quenches the material, causing it to form a resistive amorphous cap. The resistance can be decreased by slowly passing a lower current pulse (Fig. 1.15 top panel, second pulse from the left, light red) that heats the material above its crystallization temperature, annealing the amorphous cap, allowing some of the crystalline structure to slowly reform while cooling. Different resistance levels can then be achieved by applying different magnitudes of current pulses (e.g. the two different light red pulses in Fig. 1.15, top panel), resulting in different volumes of resistive amorphous and conducting crystalline phases [127]. In general, closed form mathematical models do not do a good job

[26] Technically the hysteresis in actual devices like RRAM has been found to not actually go through the origin [114] .

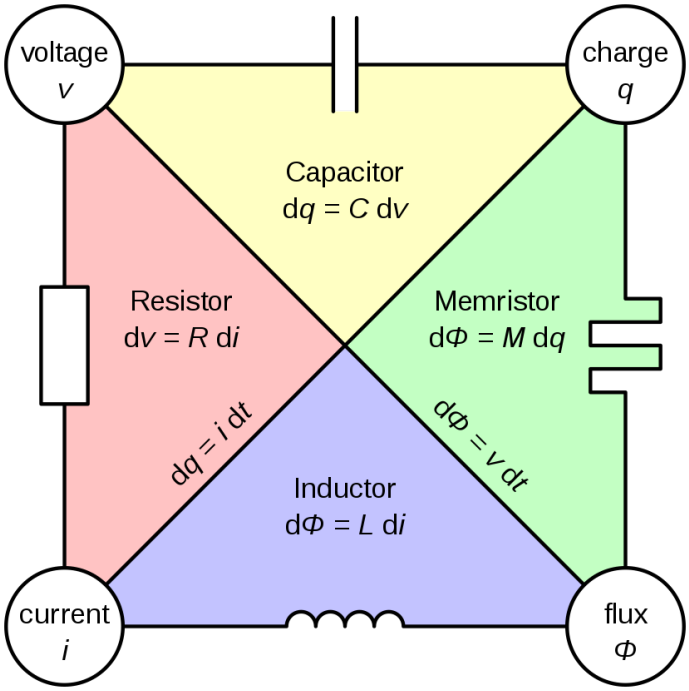

Fig. 1.13. Fundamental circuit elements and their relationships, as originally conceived by Chua [18]. Reproduced with permission. Original source: [122]

of describing the dynamics of emering memory devices. To describe the dynamics of these devices requires highly detailed finite-element modeling (e.g. COMSOL [28]), and even this is not very accurate in most circumstances because there are intricate interactions between the electric field, current, and temperature-dependent conductivity of the GST that all must be simulated self-consistently [73], [58]. For these reasons, we construct a numerical model of these devices for our study based on empirical measurements. We describe the data collection and model in the following paragraphs.

For this study, we performed pulsed resistance measurements of seven different devices (Fig. 1.15b, circuit diagram for one such device) on a 100-device PCM array. Details of fabrication and characterization of these arrays can be found in previous reports [13], [20], [31]. Each cell is equipped with its own access transistor (Fig. 1.15b) to prevent cross-talk and sneak paths in the array. Current flowing through the cell is controlled by the gate 'word line' voltage (V_{WL}), which we pulse to control the shape and magnitude of current pulses. To ensure independence between pulses, we first 'RESET' the cell to a high resistance state with a short melting pulse. We then apply a longer variable-magnitude 'Partial SET' pulse to decrease the resistance of the cell. This facilitates analysis by constraining device behavior to a memoryless zeroth-order Markov process. For example, let us say that at t = 1 second

the value of $V_{WL} = 1.0$ Volts was applied across the device. This will result in a resistance that is drawn from the conditional distribution $P(R|V_{WL} = 1.0)$. Now because of the way the device is being pulsed, this conditional distribution will be the same regardless of what the previous resistance value was. I.e. if the process had memory, $P(R|V_{WL} = 1.0)$ could be different depending on what R at t = 0 seconds was. But since the process does not have memory, $P(R|V_{WL} = 1.0)$ will always be the same. [27]

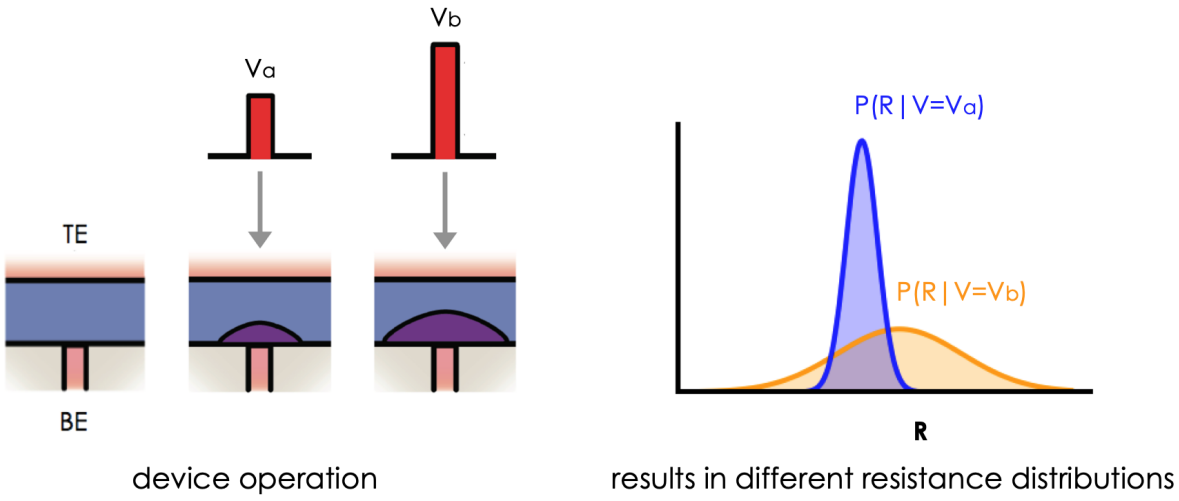

Fig. 1.14. Illustration of device operation. In this example, the default state for the device is crystalline (light purple). By applying a voltage pulse across the top and bottom electrodes (TE, BE), electrons flow through the material and heat a portion of it past the melting point (here shown in darker purple). As the pulse falls quickly (i.e. the width of the red spike is small), the device is quenched, with the portion that melted unable to relax back down to the crystalline state. Instead, this portion ends up in an in-between, "amorphous" state (again, darker purple). As the crystalline state has a low resistance (electrons can travel through it easily) and the amorphous state has a higher resistance (the electrons have a harder time passing through the partially disordered lattice), the resistance of the PCM device is determined by the amount of material in the amorphous state. And, the magnitude of the voltage pulse determines the amount of material that undergoes this phase transition process. Importantly though, there are uncertainties in the initial conditions of the device and write process. Thus, each time you apply the same voltage pulse you will get a slightly different value of resistance. If we do this many times, a distribution of resistances will result for each voltage pulse. Additionally, different voltages have different resistance distributions. E.g. the distribution resulting from V_a (blue) might have a lower mean and a smaller variance than the distribution resulting from V_b (gold).

[27] It should be noted that a digital system would need to apply a similar pulse scheme if it was to achieve greater than 1 bit of information transmission across the channel.

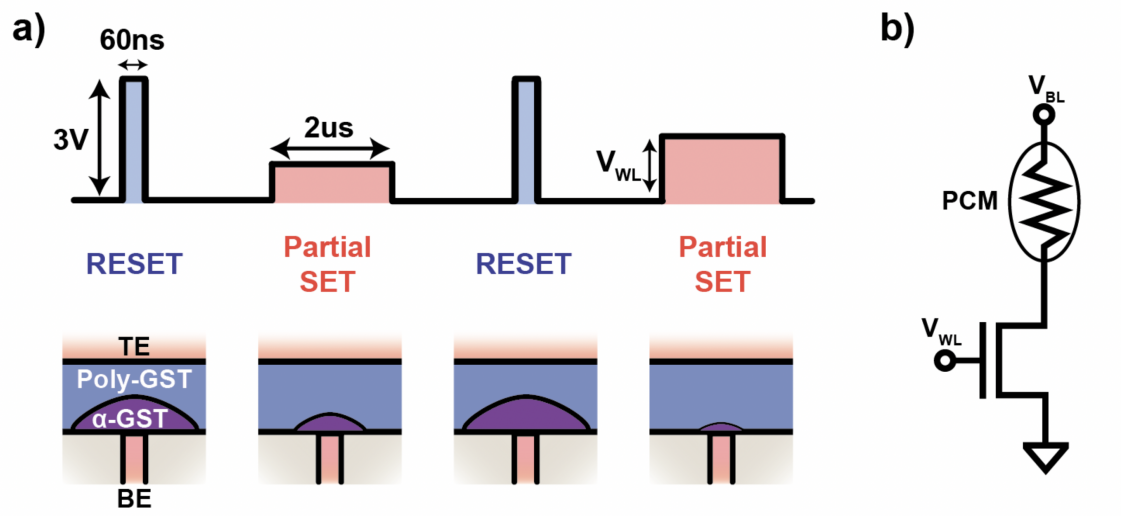

Fig. 1.15. Device Operation for our particular case. (a) Illustration of the pulsing scheme. Between each measurement, the cell is consistently "RESET" into the high resistance state via a short current pulse that melts the region above the bottom electrode (BE) and quickly cools to form a resistive amorphous cap (left). The cell is then "Partial SET" to a lower resistance with a long wordline (V_{WL}) voltage pulse that anneals the amorphous cap region (second from the left). Larger voltage pulses (larger V_{WL}) create smaller amorphous caps (far right) and thus yield lower resistances. (b) Circuit diagram of a single device within the 10 x 10 PCM array. Devices in the array are individually addressed by applying voltage V_{WL} to the gate of the access transistor. PCM resistance is modulated by current flow whose magnitude is controlled via applying a large bitline voltage (V_{BL} = 3 V) and pulsing the wordline.

If we take all of these distributions together, we get a conditional distribution, $P(R|V)$ (see Fig. 1.18 for an example distribution).

As mentioned earlier, device-to-device variation and resistance drift will have a dramatic effect on the capacity of PCM arrays. The channel noise characteristics are determined by both the memory controller and memory devices, and indeed multi-pulse read-verify control schemes have proven effective at achieving high capacities despite device-to-device variation and resistance drift [82], [87]. As a compromise between simplicity and realism, we ignore drift in our current treatment and approximate a simple memory controller to measure the responses of seven different devices on this 100-device array.

For each of the seven different devices, we first collect 120 trials at each voltage level. With seven devices, this corresponds to 840 points at each voltage level. For this study, we used 100 voltage levels. To then calculate a continuous probability density for $P(R|V_{WL})$, we first performed Gaussian kernel density estimation (KDE) of the distribution of resistances resulting from each voltage level. That is, for each of the 100 voltage levels, v, a Gaussian KDE of $P(R|V_{WL} = v)$ was made from the 840 measured points. To estimate $P(R|V_{WL} = v)$

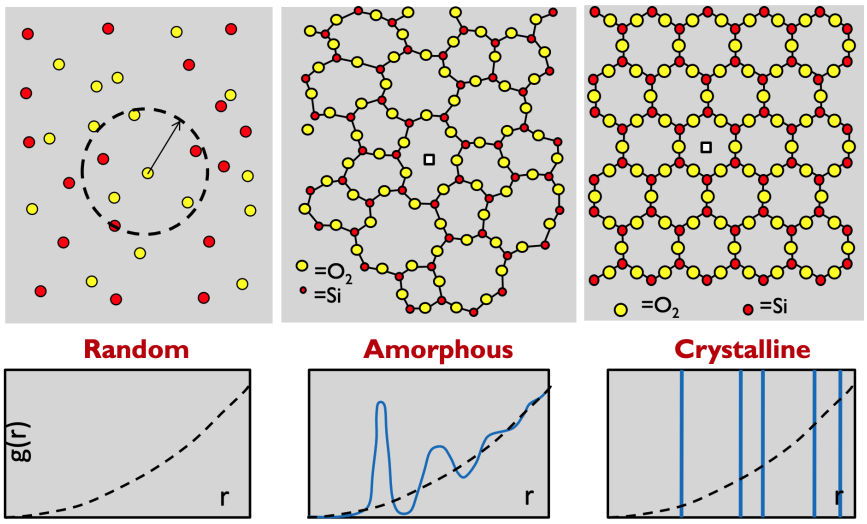

Fig. 1.16. Illustration of difference between crystalline and amorphous phase. Top left: a random or "completely disorded" phase. (un-normalized) Radial distribution function, $g(r)$, monotonically increases for a random arrangement of molecules (e.g. a gas) (would be constant if normalized by density). For a crystalline material, not all values are possible – because of the fixed lattice, you will only find other particles at certain specific distances (r). Here there is both short- and long-range order. In-between these two, you find the amorphous phase, which has short-range order (i.e. looks somewhat ordered locally (i.e. are more likely to find particles at specific distances from each other than others; hence the peaks in $g(r)$)), but long-range disorder. Thus, it locally looks like a crystal but over large scales looks more like a liquid or gas. Reproduced with permission. Original source: Muhammad A. Alam's Course notes: ECE 695: Reliability Physics of Nano-Transistors - Lecture 5: Amorphous Material/Interfaces.

for v's that were not measured, we simply interpolate between the two closest $P(R|V_{WL} = v)$ that we have measurements for.

Plotted in Fig. 1.19a are 1 standard deviation above and below the mean for each of the seven devices measured (interpolating between the 100 measured v's to get the smooth estimates shown). As can be seen, the different devices exhibit qualitatively similar behavior with slightly offset RESET resistances and slopes of annealing. We approximate the very simple single-pulse memory controller, mentioned above, by normalizing the RESET resistances between devices through measuring the relative resistance (to the RESET resistance) rather than the absolute resistance, which can be accomplished with two resistance reads and a divide operation.

Another way of thinking about this with Fig. 1.19a in mind is that we're simply applying a vertical shift to each of the curves so that their left most points align, yielding the curves seen in Fig 1.19b. Now, in this new space, the relative resistance value of a point is the logarithmic

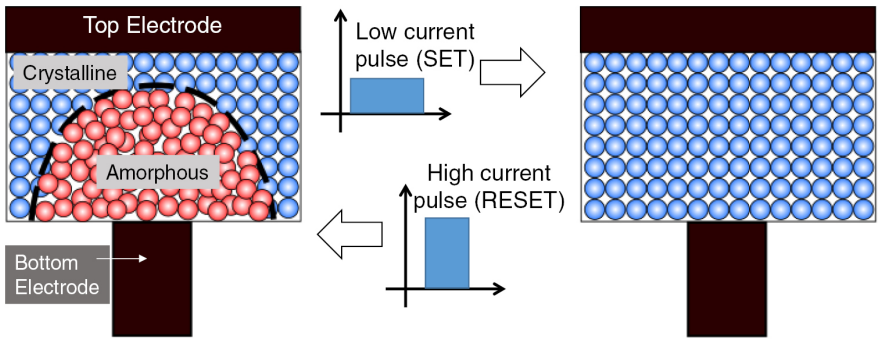

Fig. 1.17. Illustration of the what this looks like for our specific case in terms of phase of material and the pulsing scheme. Reproduced with permission. © 2019 Journal of Physics D: Applied Physics. Original source: [101]

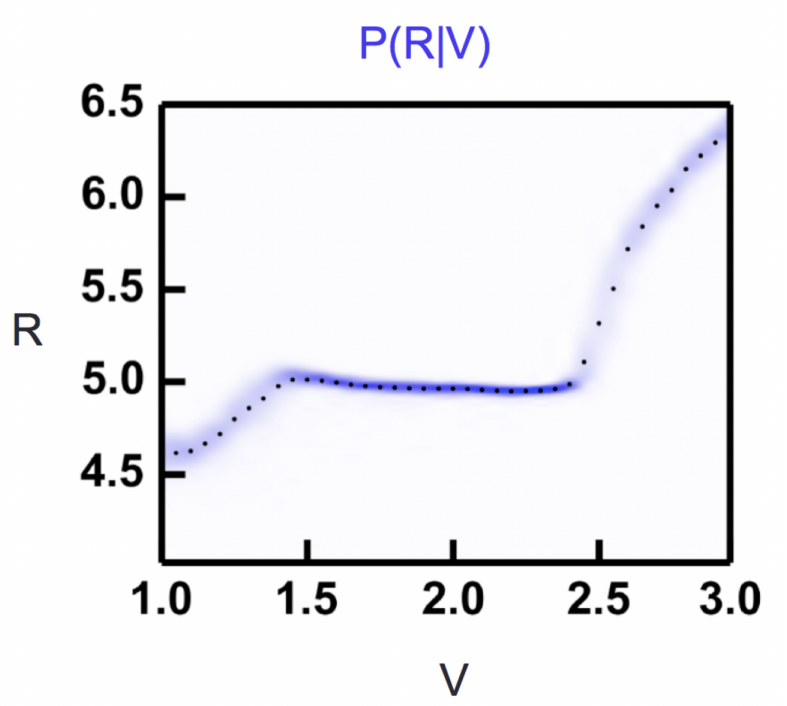

Fig. 1.18. Conditional density, $P(R|V)$, for an example PCM device (not one of the seven used in this study). Darker color indicates higher probability density at that location.

ratio between R_{RESET} and the measured resistance, i.e. $\log_{10}(R_{\text{RESET}}/R)$ (y-axis, Fig 1.19b). Thus, all the points to the farthest left are centered around 0.0, by construction (as this is $\log_{10}(R_{\text{RESET}}/R_{\text{RESET}})$), while all the points that are, for example, 10 times smaller than R_{RESET} are at 1.0.

The effect of this normalization on the actual measured resistances is that it brings all of the points for each voltage closer together. Thus, the seven curves described previously from Fig. 1.19a now lie on top of each other (Fig. 1.19b) as the seven sets of 120 points that yielded each curve are now much closer together. Finally, when the Gaussian KDE (with interpolation) is done on this new, renormalized set of points, the $P\left(R|V_{WL}\right)$[28] shown in Fig. 1.19c results.

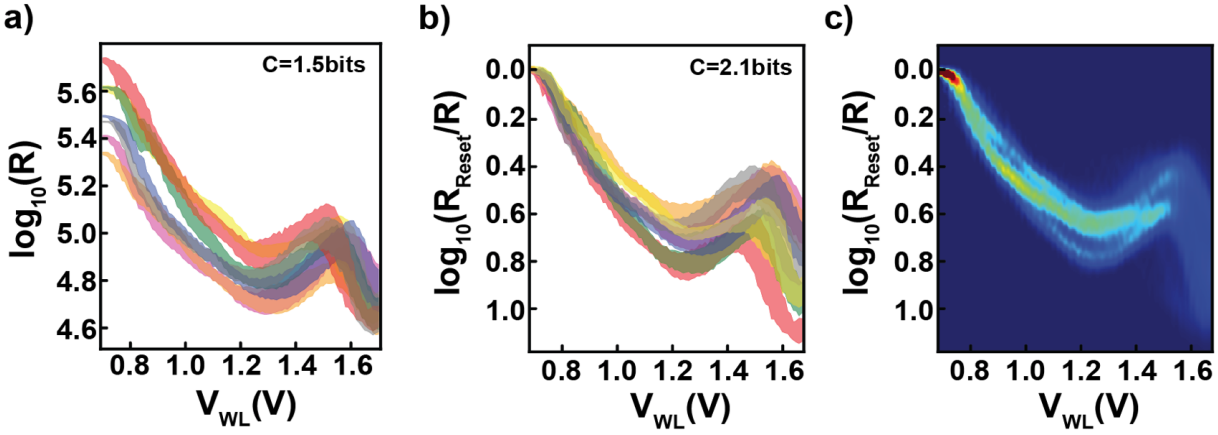

Fig. 1.19. Measuring $P\left(R|V\right)$. (a) Plot of the data collected for 100 different values of voltage pulse V_{WL} for seven different devices shown superimposed (each in a different color). Rather than plotting the 84,000 raw data points collected, we instead plot filled-in curves where the top and bottom boundary of each curve are +/- one standard deviation from the mean of R, respectively. This gives a better sense of where most of the raw data is concentrated for each of the seven devices. Values of V_{WL} between those collected (and the corresponding +/- standard deviations) are linearly interpolated. Lumping all seven devices together into a single 'virtual device' yields a capacity of 1.54 bits. Notice that the different devices exhibit qualitatively similar behavior but with slightly offset RESET resistances and slopes of annealing. (b) The same as in (a), but now with normalized RESET resistances (achievable via a simple single-pulse memory controller). This normalized data yields a capacity of 2.08 bits. (c) Heat map of the conditional distribution, $P\left(R|V_{WL}\right)$, estimated using Gaussian KDE and linear interpolation on the normalized data points.

[28]More precisely, this is $P\left(\log_{10}\left(R_{\text{RESET}}/R\right)|V_{WL}\right)$, but for brevity and clarity we'll refer to $\log_{10}\left(R_{\text{RESET}}/R\right)$ simply as R in the rest of the text.

1.3 Results

1.3.1 Capacity

From this empirically obtained $P(R|V_{WL})$ we then solve for the capacity-achieving input distribution, $P_{\text{cap}}(V_{WL})$, using the Blahut-Arimoto algorithm. The effect of the simple single-pulse memory controller is that it increases the capacity of the channel from 1.54 bits to 2.08 bits (for reference, the average capacity for any individual channel is 2.5 bits). While more realistic arrays would contain more variation and drift, they would typically also be paired with more powerful controllers, regardless of the coding scheme (either analog or digital). Thus, as they would affect both schemes, we will not factor them into our central comparison of analog vs. digital coding for emerging memory devices. Furthermore, opportunities exist to jointly learn the behavior of the controller with the encoder and decoder – e.g., with a neural network – which is in fact an avenue we explore here later.

As the Blahut-Arimoto algorithm technically only applies to discrete distributions, we use a discrete approximation to the channel, finely discretizing the $P(R|V_{WL})$ that we calculated as discussed above. Our criterion for sufficient discretization is that the capacity no longer increases with the addition of more states (e.g. >1000 states). As the goal is to optimize the trade-off between the number of output states and these states' overlap (Fig. 1.20a), the number of nonzero values in $P_{\text{cap}}(V_{WL})$ eventually saturates. That is, beyond a certain number of discrete input states, the addition of more states does not increase the mutual information (if the full analog output range is used). For this channel, saturation is reached at 13 states[29]. Note that the output distributions for these states partially overlap and cover the full range of R (Fig 1.20a, bottom panel), meaning that higher capacity can only be achieved if the full analog output range is used (as opposed to the traditional discrete case where there is no overlap). Put another way, any decoding algorithm that is to achieve the channel capacity must perform inference over the entire analog range of R.

As mentioned previously, the capacity is the maximum possible rate of information transmission over a given channel. To gain a better understanding of the effect of discretization on the rate of information transmission, we examine the case where there are a finite number of read and write states. This analysis is done to illustrate and make quantitative, for the case of the device we're considering, the idea of "soft information" presented in Fig. 1.5. We model this discretization behavior by marginalizing the continuous probability $P_{\text{cap}}(V_{WL})$ between discrete read levels, creating a reduced discrete channel. We then use simulated annealing to search for optimal values of the read levels to maximize the rate of information transmission. Fig. 1.20b demonstrates how the capacity of this reduced channel increases with the number of read levels for a given number of write levels. In each case, the capacity monotonically asymptotes as the number of read levels increases, corresponding to the case of an analog read circuit. As the number of allowed input levels increases above 13, the optimal number for the full analog channel, the discrete capacity asymptotes to the analog

[29]Non-uniform input state probabilities like those here are difficult to achieve for many applications, but setting them equal only decreases the channel capacity by 5%.

capacity of 2.08 bits. Put another way, the maximum possible rate of information transmission is only possible in the limit that the full analog range for the output is used. As analog systems inherently operate over the full continuous range of values, it is possible for them to achieve the capacity. This use of the full continuous output range can be thought of as a necessary condition to achieve the maximum possible rate of information transmission over the channel. Of course, developing the specific code that achieves this maximum information transmission is an unsolved problem. I.e. the capacity tells you what the upper bound is, but not how to achieve it. This discretization analysis shows that it is necessary to consider the full continuous range of outputs to achieve this bound, but it too does not show you how to achieve it.

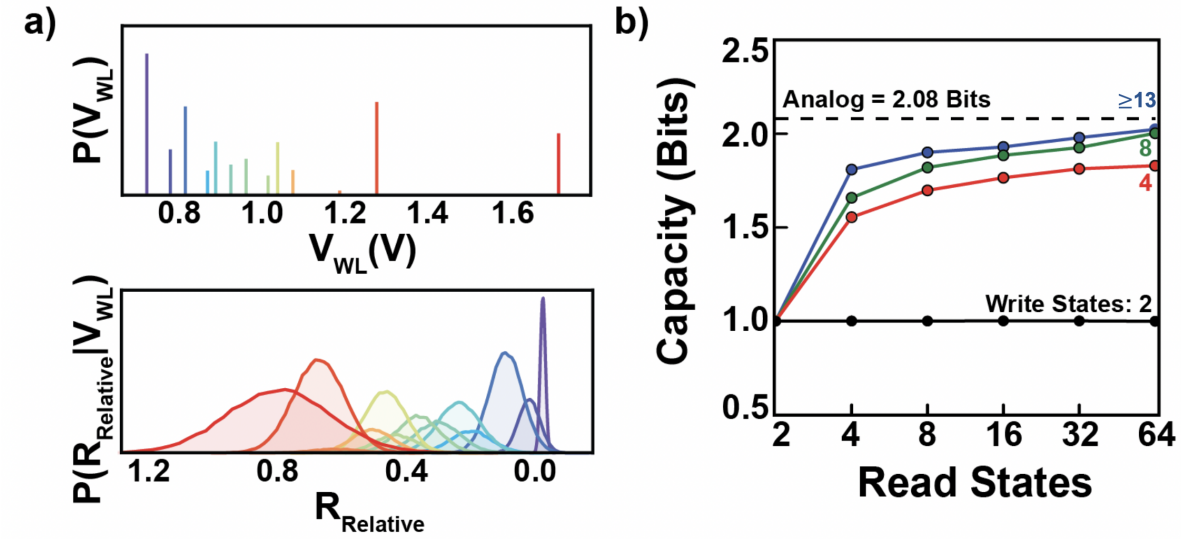

Fig. 1.20. Calculating Channel Capacity. (a) Capacity achieving input distribution, $P(V_{WL})$, and corresponding output distributions, $P(R_{\text{Relative}}|V_{WL})$), shown for each of the discrete levels in $P(V_{WL})$ (each in a different color)[Note: Here we have replaced $\log_{10}(R_{\text{Relative}}/R)$ with R_{Relative} to not clutter the figures.]. The optimal input distribution contains 13 discrete levels, spaced so as to minimize overlap in the output distributions (additional input states beyond this can be used, but they do not increase the capacity). Note that even though the input distribution is over a finite number of states, the output distribution covers the full analog range of R. (b) Discrete capacity as a function of the number of read and write states. Limited by the number of write states, capacity increases with the number of read states due to the creation of 'soft information'. For more than 13 write states, the discrete capacity asymptotes to the analog capacity as the number of read states increases. Thus, error-correcting codes that utilize analog circuits and the actual cell resistance values (such as those in an artificial neural network) can achieve the highest possible rates.

1.3.2 Joint Coding

Finally, we explore an approach to directly store analog-valued information in the resistance values of the PCM devices. While many interesting opportunities exist for exploiting the rich statistics of natural signals, for this preliminary study, we restrain ourselves to the canonical case of a single Gaussian source signal, P(S), with an MSE distortion metric. The most naïve approach to store analog values would be to employ no coding and directly map source values, S, into voltages and directly provide reconstructions, $\hat{\mathrm{S}}$, from resistances. Such a naïve approach leads to an "effective channel" (conditional probability distribution of reconstruction, $P\left(\hat{S}|S\right)$) identical to the original channel, $P_{\mathrm{cap}}\left(V_{WL}\right)$. While this is the ideal approach for a Gaussian source with Gaussian channel and MSE distortion [44], the nonlinear PCM device is far from a Gaussian channel. Correspondingly, the naïve implementation exhibits a low SNR (<0dB) at a rate of 1 device/symbol.

Given that we are working with a Gaussian source and MSE distortion, we would like a parameterized model to learn an encoding and decoding function that minimizes the average cost (to bring back and slightly modify our old friend, see Fig. 1.21). We start with a parameterized model:

$$V = F\left(S\right) \tag{1.10}$$

$$\hat{S} = G\left(R\left(V\right)\right) \tag{1.11}$$

where F and G are nonlinear mappings corresponding to the encoder and decoder, respectively. We then solve numerically for the mappings that minimize the distortion,

$$F^*, G^* = \underset{F,G}{\operatorname{argmin}}\, D = \underset{F,G}{\operatorname{argmin}} \iint \left(\hat{S} - S\right)^2 P\left(\hat{S}|S\right) P\left(S\right) d\hat{S} dS \tag{1.12}$$

The encoding and decoding functions are parameterized by a sum of un-normalized Gaussians. Specifically, F(.) and G(.) are described by:

$$y_\alpha\left(x\right) = \sum_{i=1}^{N} y_{\alpha_i} K\left(\frac{x - x_i}{\sigma}\right) \tag{1.13}$$

$$K\left(\frac{x - x_i}{\sigma}\right) = \frac{e^{\frac{-(x-x_i)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}} \tag{1.14}$$

Here, the Gaussian centers, x_i, as well as their widths, σ, are fixed, and the only variables to be adapted are the weights of each Gaussian, $y_(\alpha_i)$. Ultimately, two different y_α are learned, one for the encoder and one for the decoder. The $y_(\alpha_i)$ for each of these are learned by minimizing Eq. 1.12 via gradient descent. A sum of Gaussians is chosen because they are smooth (with well-behaved derivatives for the gradient-based optimization), are easily parameterizable, have support over the whole region considered, and are highly non-linear

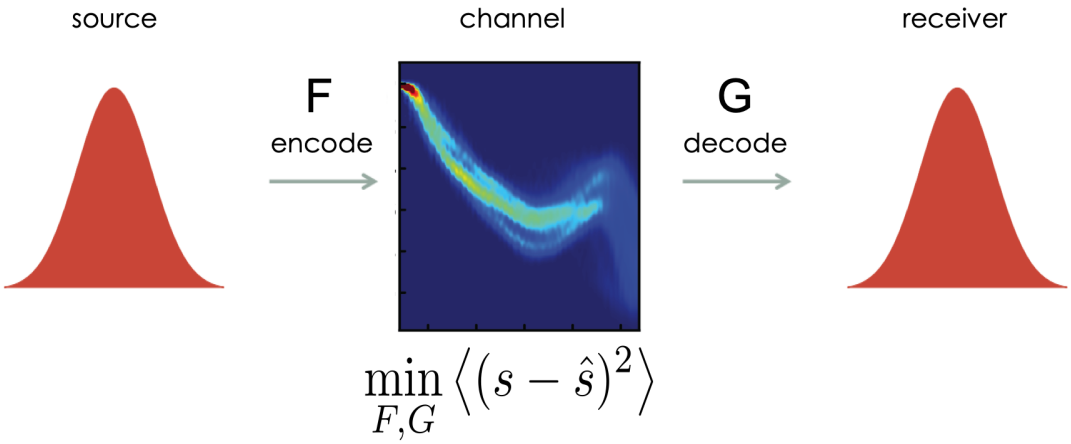

Fig. 1.21. Same as the first plot in this section, but now indicating the specific probabalistic relationship between channel input and output with the plot of $P(R|V)$ replacing the channel picture.

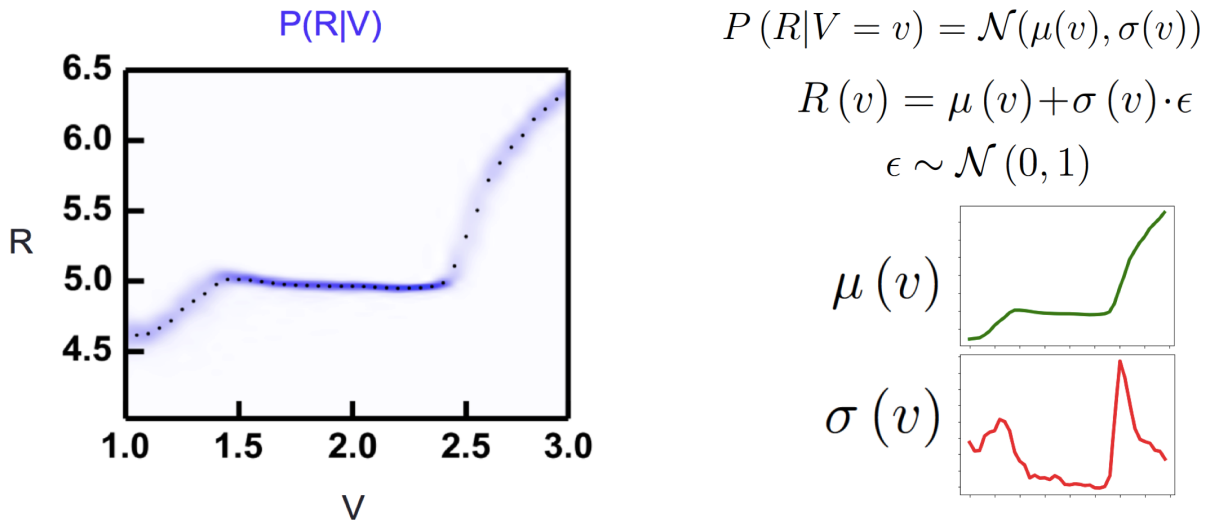

Fig. 1.22. Showing the relationship between the measured $P(R|V)$ and how it is modeled. Modeled as conditionally Gaussian, with mean and variance at each of the Vs measured given by green and red curves, respectively (points in-between those measured are interpolated).

and thus, when combined, able to approximate a large class of functions.

The results of this minimization are presented in Fig. 1.24. Here, the Gaussian distributed variable, S (blue distribution, top of Fig. 1.24a), is passed through the learned

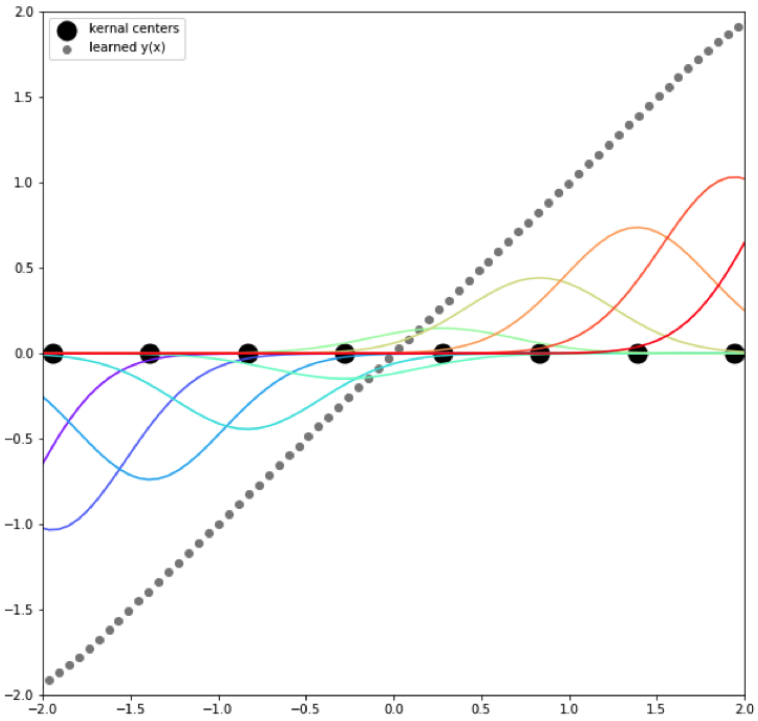

Fig. 1.23. Illustration of how the un-normalized sum of Gaussians can be used to approximate a function. Here the function is just a strait line (given by the gray dotted line). Each gray dot is the sum of all the components beneath it.

encoding function F (red line, Fig. 1.24a). That is, each sample from P(S) is mapped, deterministically, through F into a specific voltage, VWL. For example, a sample of S=0.0 is mapped to approximately 0.9 V. This mapping transforms the original Gaussian distribution into a highly non-Gaussian distribution (red distribution, Fig. 1.24a right and Fig. 1.24b, top). Interestingly, and encouragingly, this distribution is qualitatively similar to $P_{\text{cap}}(V_{WL})$ (Fig. 1.20a). For an easier comparison between the two we have taken the resulting $P(V_{WL})$ and overlaid $P_{\text{cap}}(V_{WL})$ (Fig. 1.25a).[30] It is also interesting that the parameterization for the encoding function, the sum of Gaussians, was chosen for its flexibility in function approximation, and not hand-designed to try and approach $P_{\text{cap}}(V_{WL})$. Thus, the fact that algorithm was able to learn a mapping that produced a distribution that approaches $P_{\text{cap}}(V_{WL})$ is support for the algorithm's generality.

The encoded value then gets stochastically mapped through the channel. That is, each sample voltage results in a draw from the conditional distribution given by $P(R|V_{WL})$.

[30]It is difficult to make a quantitative comparison here as the capacity achieving source distribution is over a finite number of elements whereas this distribution has compact support in different sections.

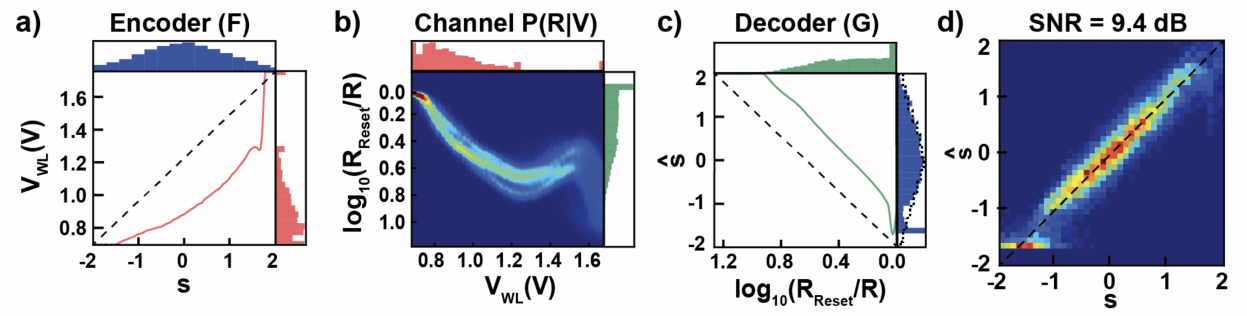

Fig. 1.24. Optimal joint coding of a Gaussian source. (a) Samples, S, distributed according to a Gaussian (blue distribution, top) are mapped through the learned encoding function (red line). For reference, a linear mapping is shown with a dashed line. This encoding transforms the Gaussian distribution into a highly non-Gaussian distribution over V_{WL} (red distribution, right). Encouragingly, this distribution qualitatively matches the capacity achieving source distribution, $P_{\text{cap}}(V_{WL})$ (a quantitative comparison is difficult as the former has compact support over different regions while the latter has finite support). (b) Samples, V_{WL}, distributed according to $P(V_{WL})$ (red distribution, top), are then stochastically mapped through the channel (heat map in center), resulting in a distribution over resistances, P(R) (green distribution, right). (c) Samples passed through the channel, R, are then sent through the decoder, G (green line), which transforms them into estimates of the original input, $\hat{S}$ (again, a linear mapping is shown with a dashed line). This results in a distribution over reconstructed samples, $P\left(\hat{S}\right)$ (blue distribution, right), which is very close to the original Gaussian distribution (indicated with a dashed line). (d) Heatmap demonstrating how samples of S are transformed through the whole pipeline, resulting in reconstructions Shat. Note that as the majority of the mass for S is between [-1,1], this is where the encoding/decoding functions learn to do the best.

For example, V_{WL}(S=0.0) results in a draw from $P(R|V_{WL} = 0.9)$. The distribution of resistances that results from this mapping is again highly non-Gaussian (green distribution Fig. 6b, right and Fig. 1.24c, top) [31] Finally, the read-out resistance is passed through the learned decoding function, G (green line, Fig. 1.24c). The resulting distribution for $\hat{\text{S}}$ is very close to the original Gaussian (with the target distribution for S given by the dashed line). For an easier comparison between these distributions we have also taken the original P(S) and overlaid P($\hat{\text{S}}$) (Fig. 1.25b).

To quantitatively assess the performance of this approach, we can draw many samples from P(S) and check what the corresponding $\hat{\text{S}}$ is. This creates an 'effective channel', $P\left(\hat{S}|S\right)$

[31] Note that the encoder learns to map a large amount of mass in the region of the channel where input-output pairs are easiest to distinguish (red portion of histogram, where the noise is smallest) and none in the "indistinguishable" region of the channel (region with positive slope, approximately 1.2 – 1.6 V).

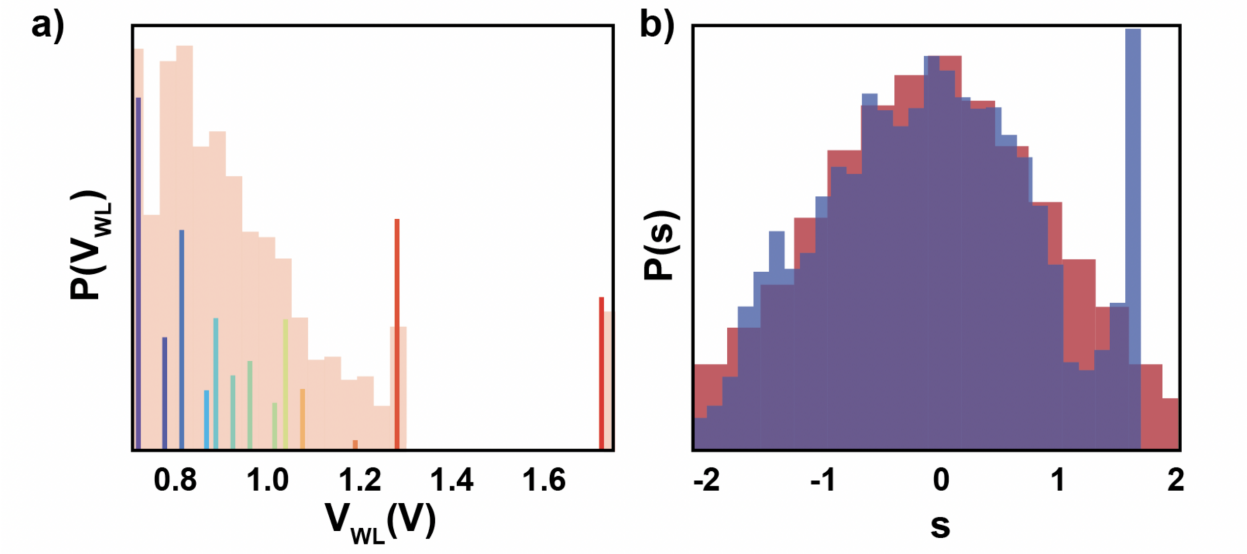

Fig. 1.25. Distributions from learned mapping. (a) Reproduction from Fig. 1.20a of the capacity achieving source distribution, rainbow, overlaid on the distribution produced by the learned encoder (the same distribution as shown in Fig. 1.24a, red, and Fig. 1.24b, red). Notice how the learned mapping captures most of the features of the capacity achieving distribution: largest concentration of mass around the minimum value of V_{WL}, with a taper until a spike around 1.3, followed by zero mass until one final spike at the maximum value of V_{WL}. (b) Reproduction from Fig. 1.24c of the mapped output distribution, blue, overlaid on the source distribution, red. Notice again how the learned mapping does the best job of reconstructing the input in the region [-1,-1], where the majority of the mass is.

(Fig. 1.24d). This analysis shows that samples from P(S) are mapped through the channel and recovered by $\hat{\mathrm{S}}$ with a small amount of scatter around the straight dashed line (corresponding to $\hat{\mathrm{S}} = \mathrm{S}$), yielding an SNR of 9.4 dB. Note how the region between [-1,1], where the majority of S's mass is, is where the encoder and decoder learn to do the best job of transmitting the signal over the channel. This is a consequence of the MSE distortion metric: the encoder and decoder are penalized the most in this region and thus devote their resources to performing good reconstructions here.

As another point of comparison, we can look at a quantity from the analog coding literature: the Optimal Performance Theoretically Attainable (OPTA). This quantity is obtained by equating the channel capacity to the rate distortion function of the source considered, and then solving for the resulting minimum achievable distortion (usually in decibels). Thus, for the Gaussian source and the channel being considered we have:

$$\frac{1}{2}\log_2\left(\frac{\sigma_S^2}{D}\right) = 2.08 \text{ bits} \tag{1.15}$$

This implies that the OPTA is 12.5 dB (3.1 dB from our method). While this is a useful quantity to consider as it provides an upper bound to the performance, we prefer instead to think about the effective number of bits transmitted across the device (obtainable via a similar manipulation as above, but solving for rate instead of distortion). From this perspective, we think that the most surprising and important result of this work is that even with this very simple encoding/decoding scheme (amounting to a single dimensional lookup table), joint coding with these devices achieves a similar SNR to separate coding schemes of high complexity. Specifically, this joint method performs as well as a separate system that first compresses strings of samples from the Gaussian to 1.56 bits per sample (e.g. through vector quantization) and then uses a hypothetical channel coder capable of getting 1.56 bits of errorless communication across the device. Said channel coder is "hypothetical" in that there does not currently exist a channel coder capable of achieving 1.56 bits across the device. In reality, the design of channel coders for specific channels is a challenging task. For example, though the capacity of the standard AWGN was worked out by Shannon over 70 years ago [102], it wasn't until the 1990s that a class of codes was created that could approach the capacity of these channels [77].

Though a channel coder for the separate coding scheme does not exist for these channels, we can assume one does for the purposes of a comparison. We can then further quantify the benefits of our approach by comparing the energy and latency required to perform the above storage task with separate vs. joint coding schemes. For the comparison, an '8KB' PCM array is used, with both separate and joint coding schemes implemented in Samsung 28nm technology. As discussed above, the separate approach requires the use of digital error-correcting codes. These codes require large blocklengths for even modest code rates (e.g., 0.5). The current state of the art – both in terms of performance and efficiency – are LDPC codes. Thus, we use these as the hypothetical channel coder for the comparison. An LDPC code with good decoding performance typically requires a blocklength of 2000 at a code rate of 0.5 (with significantly longer blocklengths for higher rates) [75]. The decoding step alone for this requires 44 pJ/sample. For this storage task, the total energy expenditure for a separate scheme with an LDPC code is 128 pJ/sample. Furthermore, the LDPC decoder requires 10 (or more) iterations to converge. By comparison, a joint coding scheme, achieving the same SNR (i.e. at the same distance from the OPTA, described previously), implemented using a simple lookup table requires only 40 pJ/sample and a single iteration. The decrease in required energy and latency come mainly from (i) the elimination of the LDPC coder – requiring long blocklengths – and (ii) the reduction in PCM write operations – 2 writes/sample $\rightarrow$ 1 write/sample – due to the higher coding rate achievable with joint coding in the small blocklength regime (via the intrinsic error-correction provided by the learned encoding/decoding functions).

While we have only examined the case of single dimensional encodings here, the design landscape for analog encoders/decoders is rich for higher dimensional signals and other compression ratios, making it a promising area for the development of coding circuitry that is low in energy and latency. Lastly, we should emphasize that because the joint framework we proposed is adaptive, it can learn to store different types of analog data on different types

of channels, as diagramed in Fig. 1.2 (e.g., RRAM, MRAM, or other PCM devices).

1.4 Discussion

The results presented here represent only a single model applied to a single type of signal and device. While we only looked at one type of signal and device for illustrative purposes, the methods we used here are very general and can be applied to a wide range of devices and signals. The future design space to explore in this field is extremely rich.

From the experimental perspective, other devices (RRAM, CBRAM, and MLC-Flash) can exhibit dramatically different behavior. Furthermore, various pulsing schemes can produce different types of statistics. For example, iterative pulsing introduces rich conditional dependencies through time that are not currently exploited by multi-bit memory systems and present an interesting opportunity from the perspective of modeling. Cell-to-cell dependencies within the array are also a fact of life that must usually be avoided through hand-engineered systems. Measuring these statistical dependencies can enable codes to be adaptive to the flaws and statistics of individual chips, both at the factory during test time and over the lifetime of use, improving the reliability and uniformity of performance.

From an algorithmic perspective, many powerful analog-valued high-dimensional models such as artificial neural networks exist, leaving a rich space to explore for learning high-dimensional parameterized encodings and compression. Since such models have been able to successfully utilize the statistics of natural media such as images [71] and sound [54], they would seem well-suited for encoding such media in analog-valued devices. While these neural network methods have been successfully applied to the problem of source coding [111], [7], there has still yet to be a thorough treatment of the problem for doing both source and channel coding with emerging memory devices, and we think this is an exciting open area for research. Our own initial work in this direction (see subsequent chapter) has shown that a joint approach with neural networks can outperform traditional separate approaches when storing images on an array of emerging memory devices. In a related work, [12] have looked at the problem of using neural networks for source and channel coding with standard channels, such as the Gaussian and Rayleigh channels. As mentioned above, many of the concepts we have discussed here are regularly applied to analog coding problems in the communications literature, typically in the context of communicating over standard channels. Our focus in this work has been on bringing these ideas into the problem of storage with emerging memory devices.

Chapter 2

Image storage with neural networks; simulating multi-variable case

I can't keep fighting alien
technology with a six foot staff.

Donatello

Let's scale up to natural data with structure and utilize a collection of devices. As discussed above, images are interesting to us for two reasons (i) They're the most abundant data in the world and (ii) Our lab, and the field more broadly that our lab is embedded in, have made progress on capturing natural image structure. And, as part of the goal of storage is compression (and as compression involves the removal of structure), we'd like to utilize our knowledge of the structure of natural images to remove said structure. Based on the discussion in Chapter 1, we seek a mapping from images to a set of emerging memory devices. One way of thinking about an $N \times N$ pixel image is as a point in $\mathbb{R}^{N \times N}$[32]. So, to heavy-handidly put: we're looking for highly expressive, learnable mapping from points in $\mathbb{R}^{N \times N}$ to points in $\mathbb{R}^M$.

2.1 Background

The report of my death was an exaggeration
Despite the "unreasonable effectiveness of data"[49], that is, despite what some might have you believe in the era of Big Data and new "electricities", "human ingenuity", "engineering", or "theory" is more important than ever. The great "success" of machine learning models that crank through terabytes (and kilowatts) to train has left many with the impression that

[32]This is the way we'll think about images here, but it's likely that a better way to think about images is as functions defined on $\mathbb{R}^2$ (see [17] for a great discussion of this and to see some of what it allows you to do).

bigger models with more data are all we need to reach the moon [136].

With that in mind, we'll be using a neural network to parameterize the mapping. BUT, we'll be "engineering" some structure into the network based off of what we know about the structure of natural images (a la spatial-invariance and convolutional neural networks). The general setup looks something like Fig. 2.1).

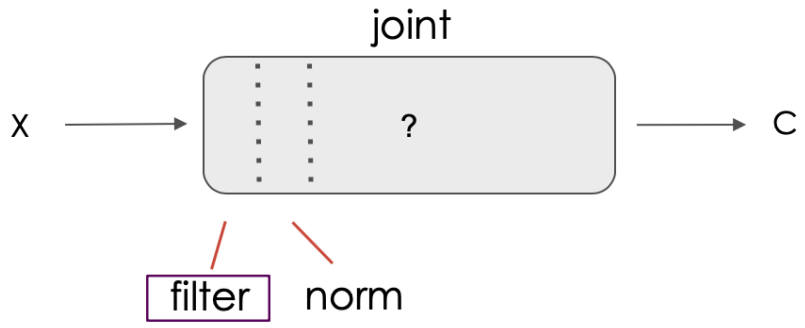

Fig. 2.1. Our joint coder will need many stages of processing to removing the redundancies present in the data while retaining (or replacing) some of this with redundancy for being robust to noise introduced by the channel. With regard to natural images, we know that the first step likely involves some sort of filtering (edge detection) followed by some nonlinear, group normalization of the activity of these edge-detectors (divisive normalization). But after that, it's unclear. So let's try and learn the rest.

As discussed below, the first part of processing is likely some sort of "edge detection" (with filters) followed by a normalization of filter outputs. I.e. we know that images can be described to a decent approximation as a sparse linear combination of "edges" [85]. Where the Gabor filter is given by

$$g\left(x,y;\lambda,\theta,\psi,\sigma,\gamma\right)=e^{\left(-\frac{(x\cos(\theta)+y\sin(\theta))^2+\gamma^2(-x\sin(\theta)+y\cos(\theta))^2}{2\sigma^2}\right)}e^{\left(i\left(2\pi\frac{x\cos(\theta)+y\sin(\theta)}{\lambda}+\psi\right)\right)} \tag{2.1}$$

Now, Let's take two wavelets and look at their activity when convolved over an image. Looking at Fig. 2.3 – Left: natural image, right: two similar but non-overlapping Gabors and respective activity maps that result from their convolution over the image. When looking at joint statistics, we find that the variance of a filter's activity depends, nonlinearly, on the value of another. This is just another way of restating the well known result that coefficients in wavelet sub-bands have nonlinear dependencies. We can model this dependence by saying that the variance of filter activity i goes as the square of filter activity j. If we then divide the activity of each filter by a linear combination of squared neighboring filter activities, we find that the pairwise dependency between each filter removed. Namely, if we now look at the conditional distribution of activity for a filter, we find that its variance remains constant as the activity of other filters changes. This process is called divisive normalization (see Fig. 2.3)[100].

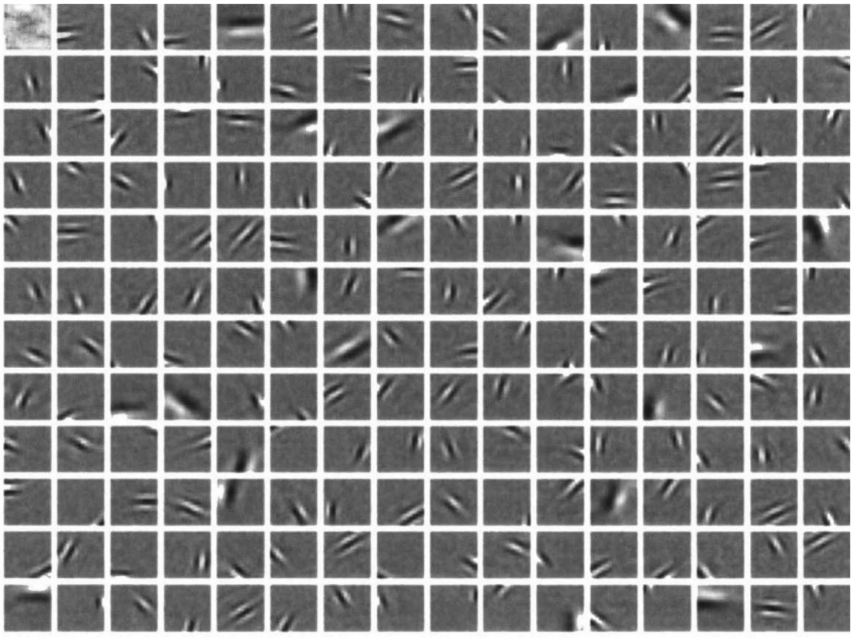

Fig. 2.2. Example of filters learned via sparse coding [85]. Similar "Gabor-wavelet-like" filters are typically learned in convolutional neural networks whose task is to perform image classification. Reproduced with permission. © 1998 Nature.

Beyond this we don't have a great way of parameterizing higher-order structure in images. There has of course been a lot of work on this (see [98], [107], [137], and [57] for some great overviews), but given that (i) it seemed like a neural-network-based approach was good to use here and (ii) the majority of the above "joint" box would have to be learned, we wanted to put some structure in that could easily be introduced into the network (without requiring significant changes in optimization techniques).

[33]There is certainly a lot more going on in the early stages of animal visual systems than simple filtering and divisive normalization [86] (e.g. recurrent computations likely play a hugely important role). I.e. there's a lot more structure that the visual system is capturing. All we're doing here is taking some things that seem to be the case and using them to our advantage for this storage problem. Since we're dipping in to the topic of neuroscience here, it should also be noted that animals aren't cameras. They don't store "photographs" but rather representations [I] that are useful for some downstream action[II]

[I]The photoreceptors of the eye receive 36 Gb/sec of raw data. The retina does 1000:1 compression and this gets processed down to 20 Mb/s! [105]

[II]For animals (especially humans) are meant to act in the world. There's a ton of great literature on the "perception-action"/"sensory-motor" loop . But I apologize. I didn't have time to write a short list of references, so I wrote a long one instead: [8], [83], [62], [118], [89], [67], [66], [110], [117], [45], [39], [14], [96], [40], [109], [43], [26], [27], [88], and [6].

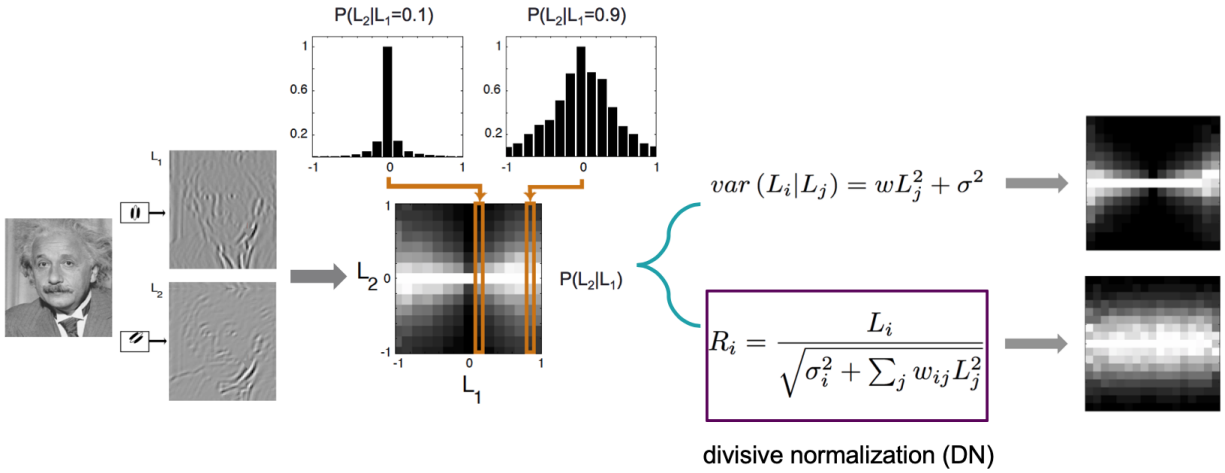

Fig. 2.3. Removing feature dependencies. Example of feature maps from two similar Gabor wavelets (from the same sub-band) convolved over our tamed (mustachioed) metaphysicist. Conditional dependencies exist between one feature's activity and the distribution of the other feature's activity. Specifically, the variance of the later goes like the square of the value of the former (see equation). With this knowledge, we can divide out this increased activity, effectivley removing this dependence (i.e. removing the "bowtie" dependence seen in the example figure).

So, with that, we decided to follow the work of [7] and implement this divisive normalization as the nonlinearity used in the network (explained in more detail below).[33]

2.2 Setup

Now, we'll combine these ideas into a framework that can use them to learn a mapping from images to a set of PCM devices, and then from those PCM back to a reconstruction of the image. We'll be using a neural network to learn this mapping (for the reasons stated above). This type of neural network, mapping from input X to some constrained "latent space", and then back to $\hat{X}$, is called an autoencoder. Below I'll briefly discuss the general framework for neural networks, how they are trained, and how we are specifically parameterizing ours to accomplish our task.

2.2.1 Standard Neural Network

At this point I get the sense that even my grandmother, incapable of operating an iphone as she is, knows what a neural network is. But for the sake of completeness (and to step outside of my own biased bubble), I'll quickly describe the idea here:

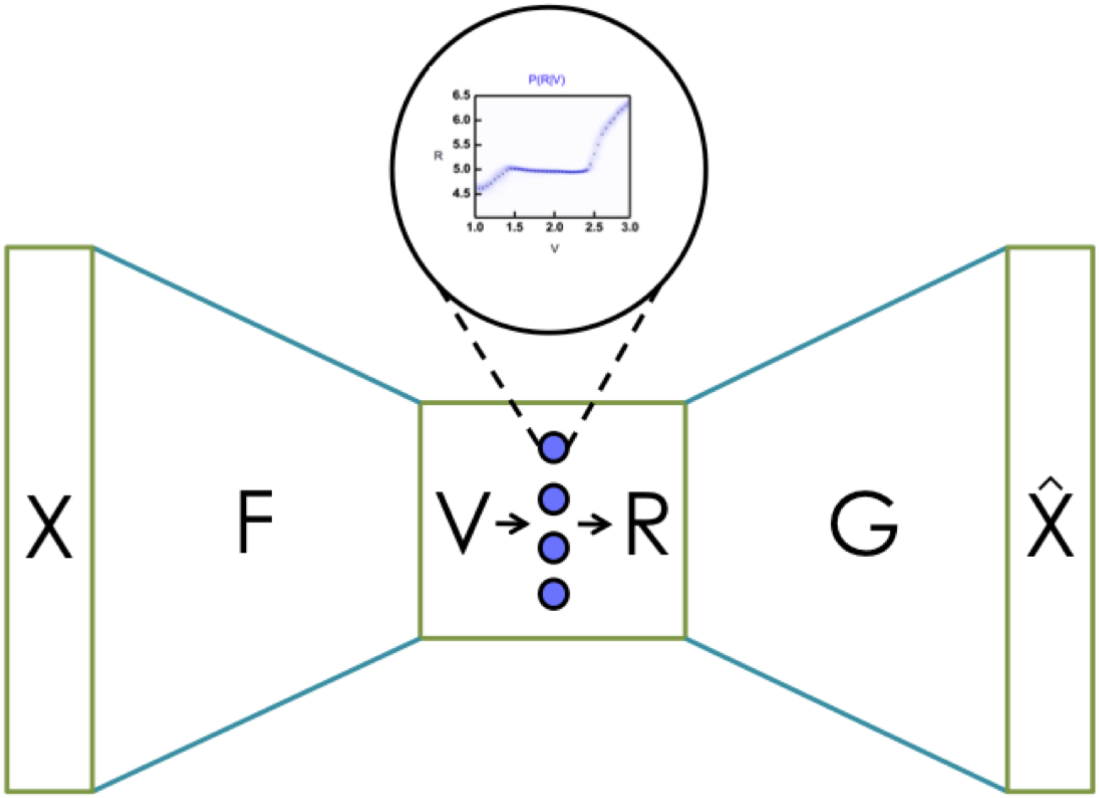

Fig. 2.4. General setup for our framework: image data, X, is transformed by an encoding function, F (here a neural network), into a set of voltages. These voltages are then applied to a set of PCM devices. The resistance of these devices is then read and passed to the decoder, G (also, surprise!, a neural network), which attempts to make a reconsruction of the original image, $\hat{X}$.

At a high level of abstraction, neural networks are a particularly interesting subset of graphs that loosely model biological neural networks. A graph is a collection of lines and points. In the case of neural networks, the lines or "edges" are called "weights". They are directed and connect the points or "neurons". Each edge has a number assigned to it indicating how strongly the neurons are connected. Each neuron computes some sort of function of the edges leading in to it and then passes the output of this function along another edge. The output of each neuron is multiplied by the weight of the edge that connects it to the next neuron. That's it (see Fig. 2.5 for a simple, but common, example). Depending on the particular network, the function that the neuron computes may be different (e.g. point-wise linear, point-wise nonlinear, population non-linear, stochastic, reccurent). And depending on the desired overall computation the network will be arranged in certain ways

with various connectivity structures and neuron functions.

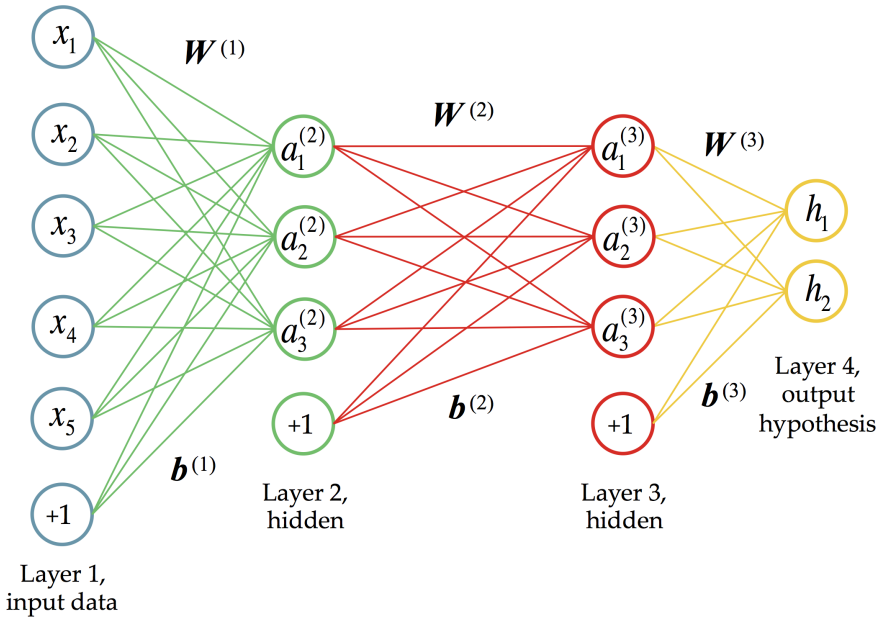

Fig. 2.5. Example of a typical feedforward network used in a classification task. Reproduced with permission. Original source: [106]

In the past decade there's been an (overwhelming) explosion of research in this area with a variety of different kinds of network architectures proposed[34]. A couple of folks over at the Asimov Institute have put together a great summary chart for the broad types of neural networks. I highly recommend anyone interested in learning more about the subject (or a specific kind of network) to check out their webpage. As the networks I'll be discussing are either outlined there or are composed of combination of networks there, I'll include their chart here.[35]

2.2.2 Backpropagation

But what about the "learning", you might be asking? People talk about training these things and getting them to learn. Where's that in this picture? Good question! Essentially, people

[34]While many of these are new designs, it's safe to assume that some form or another of the design you propose was made by Jürgen Schmidhuber in the 90s.[I]

[35]Plus it's just an awesome chart that synthesizes a ton of work and should be presented as much as possible.

[I]And by David Zipser in the 80s [80].[i]

[i]Thank you Dr. Paiton for this great find (and scholarly observation.)

use a clever application of calculus: the chain rule for derivatives. The idea is pretty clear when we think of a neural network in a slightly different way: as a computational graph.

Computational graphs are a slick way of reformulating mathematical operations as a specific kind of graph. Here, edges are simply what takes results of one computation to another, and nodes are either the inputs to a computation or a computation themselves. For example, if we wanted to represent the equation $a = b + c$, we have two nodes, b and c that then go into a third node, a, that sums them up. We could repeat this process by adding more nodes and edges to our graph. Eventually, we could even reproduce Fig. 2.5 with this strategy (just add extra nodes to represent the weights and biases).

Now, when we build a neural network, we want to know what value of weights (and biases) are needed to give a good output (given by some evaluation metric). This amounts to asking how changes in these parameters affect the output. As the direction we care about is: how all parameters connected to a specific output affect that output, we can think about changing that output a little and then seeing how that change propagates backward to all the nodes connected to it. And there you have it! "Changing the output a little" is just the derivative, so what we're really doing is adding together all the paths with derivatives on them. An efficient algorithm for doing this via factoring the paths is called "reverse-mode differentiation" (and this is what backpropagation is doing). For a much better summary of this (with great illustrations), check out Chris Olah's blogpost: [84][36].

2.2.3 Autoencoder

An autoencoder is represented in the third row of Fig. 2.6, first column. The idea is: Let's take an input, map it to some constrained space, and then try to reconstruct the original input from that constrained space. E.g. in the case of the diagram, going from a 4-dimensional input to a 2-dimensional "latent space", and then back to a 4- dimensional reconstruction. There are many reasons one might want to do this, but one common one is that we want to learn something about the "latent factors of variation" in our data. I.e. if we can figure out how to reduce our data to a much smaller representation and still do a decent job of reproducing it, then, one would hope, this representation has captured some of the structure in the data (as compression is the removal of structure).

2.2.4 Convolutional Autoencoder

Referencing Fig. 2.6 again, a convectional autoencoder can be thought of either as a "deep convolutional network" plus a "deconvolutional network" (without the orange output cells from the convolutional piece or the yellow input cells from the deconvolutional piece) or as the "deep convolutional graphics network" with simple hidden units instead of probabilisitc hidden units. It's really just an autoencoder with convolutional units. Convolution in neural

[36] I would highly recommend checking out any of the articles on Chris's blog. They provide some of the best explanations I've found for many concepts in/related to the field.

networks, despite the fancy name, is a very simple concept[37]: though it's jumping the gun a bit, let's use Fig. 2.9 for a reference. For convolution in a neural network, we start off with some input (usually an image of some dimensions length×width×channel[38]). We then have "kernal" which also has length×width×channel, but the first two are smaller than the length and width of the image. What the kernel is doing is "looking" for a particular feature in the image. It simply takes the inner-product of itself with the input at a certain spatial location. This produces a large number if the two are similar and a small one if they are dissimilar. We then save this number in the output tensor[39] and slide the kernel over by one (or more) units (pixels if we're at the first layer) and do the same thing at this new spatial location. After we've done this over the whole input (going both up and down) we'll have a new set of numbers representing how much the particular feature the kernel was looking for is present at different locations in the input. But this itself can be thought of as a sort of "image" (but only has dimensions of length×width, as the inner product summed over the channel dimension). The channel dimension for this output tensor (which is just the input to the next layer) is just the number of filters that you had looking over the image (e.g. if you had 10 different filters, you would get 10 different "channels" or feature maps, one for each filter).

As an example, Fig. 2.9 starts of with an $64 \times 64 \times 1$ pixel image with one channel dimension. This is convolved with 128 different kernels, each of shape $9 \times 9 \times 1$, moving with a stride of 4 (they are moved over 4 pixels each time). This produces a $16 \times 16 \times 128$ output tensor (one channel dimension for each filter). This tensor is now the input for the next layer, which takes 128 kernels of size $5 \times 5 \times 128$ and convolves them over the tensor. After a few more convolutional layers, the final output tensor has dimensions $4 \times 4 \times 28$.

2.2.5 Variational Autoencoder

The motivation for the variational autoencoder [65] is to construct a generative model of some data. The idea here is that we want to be able to sample from an easily sample-able distribution (let's get weird and say a Gaussian) and use these samples to generate samples from the highly non-easily-sample-able data distribution (e.g. natural images). So we'd like something that takes samples from the easy distribution and transforms them into samples from the hard distribution of interest. The variaional autoencoder (third row of Fig. 2.6) accomplishes this by taking the input data and mapping it to a set of parameters.

[37] I'll explain the ideas specifically with reference to our problem but, unsurprisingly, Chris Olah has a nice blog post on convolutions that's worth checking out for a more general discussion.

[38] Where channel refers to the "depth" of the input. The term comes from color images where there are 3 channels: reg, green, and blue. This corresponds to three different images, all with very similar but slightly different information. But it doesn't have to stop at 3. For example, hyper-spectral cameras can have hundreds of channels, each looking at a narrow frequency band. In general, the channel simply is a feature at a specific spatial location.

[39] OK, yes, for those mathematically in the know, this isn't really a tensor. It's technically a holor. But no one knows what the hell that is (and tensor sounds cooler and gets the general idea across), so we'll stick with that.

These parameters are actually the means and variances that parameterize the aforementioned simple distributions – means and variances in the case of a set of Gaussian distributions (one for each latent variable). Then, we sample from distributions with these parameters. We then take these samples and pass them through a decoder that tries to reconstruct the original image. Finally then, once this whole framework is trained, we can throw away the encoder and just draw samples from the Gaussians and, in theory, use the decoder to transform these into realistic looking samples from the interest distribution. But how do we learn the weights of this encoder/decoder when we have these stochastic units in the middle (e.g. the Gaussians)? The idea is that we can actually just reparameterize the distribution so that we separate it into pieces which are stochastic and pieces which are deterministic (the parameters we care about)[40]. We are then able to take derivatives of variables with respect to the parameters of the distribution (as these parameters are deterministic). See Fig. 2.7 and Fig. 2.8 for some illustrations of the idea and [4], [79], and [95] for great discussions of the details.

2.2.6 Putting it All Together

We combine these ideas to construct a convolutional autoencoder with divisive normalization nonlinearities, with stochastic latent units given by the models of the devices that we've been discussing. The main difference between the latent space for our case and that of the variational autoencoder is that we're not trying to learn a generative model of the data. Instead, we have conditional distributions with fixed parameters that need to be dealt with (not learned).

The objective to be minimized in our case is a combination of reconstruction error and a regularization penalty on the activations of the latent space:

$$C = \left\langle \left\| X - \hat{X} \right\|_2^2 + \lambda \left[\max\left(0, v - V_{max}\right) + \max\left(0, V_{min} - v\right) \right] \right\rangle \tag{2.2}$$

Here, max (.) puts a constraint on the latent space activity (for going outside $V_{\max}$ or $V_{\min}$), λ establishes the relative importance of keeping the write voltages within the range of the device, and v (voltages) are the activation values of the final encoding layer (i.e. outputs of f (.)). The full network was trained using ADAM [64] on this cost function.

The nonlinearities in standard neural networks are typically pointwise ReLU, i.e. $a_i' = \max\left(0, a_i\right)$, where $a_i = \sum_j x_j \cdot w_{ij}$, with x indicating the layer's input and w indicating the weights. In contrast, divisive normalization is a population nonlinearity, whose functional form is given by:

$$a_i' = \frac{a_i}{\sqrt{\beta_i^2 + \sum_k \gamma_{ik} a_k^2}} \tag{2.3}$$

[40]Not all distributions can be reparameterized this way, but luckily most of the ones we're good friends with can be.

where β and γ are learned parameters. Divisive normalization implements a local form of gain control that can reduce nonlinear dependencies [100] (mentioned previously).

The architecture we used was similar to that described in [7]. To adjust the compression rate of the network, we varied the number of units in the last layer of the encoder/first layer of the decoder (128 for the low compression, 30 for the high compression), while keeping the number of units in all other layers and the number of layers the same as in [7]. The output of the encoder is passed through a set of model PCM devices and their outputs are passed through the decoder. Thus, in contrast to [7], we are asking the network to perform both source and channel coding.

2.3 Results

As this model is performing joint SC coding , we need comparisons against models with both a source and channel coder. As it is the most commonly used for natural images, we used JPEG for the source coder. For the channel coder, we used two different hypothetical schemes. 1st: (red) binary scheme: i.e. one able to get 1 bit information across the channel. To assess performance on images of natural scenes, we used 24 gray-scale images from the Kodak dataset. We measured two compression levels using our proposed autoencoder framework. The higher capacity network (with 30 filters in the last encoding layer) used 7,680 model PCM devices to store each image, while the lower capacity network (with 12 filters in the last encoding layer) used 3,072 PCM devices. Results from this test are illustrated in Fig. 2.10. We found that the convolutional autoencoder was able to outperform the JPEG + binary channel coder method for both rates. Using the k-nearest neighbors algorithm introduced in [70], we estimated the achieved marginal information transmission rate across a subset of the PCM channels was -1.5 bits/channel. This indicates that it would take more than 1.5 times as many PCM devices to store an image if the binary channel coder were used (as is the case in commercially deployed 3D-Xpoint [48]).

2.4 Discussion

The questions we address are how to optimally store image data an on analog medium, and, more generally, how to optimally perform compression and error correction in this setting. The noise characteristics of PCMs and other analog storage devices are significantly different than traditional devices. Additionally, to optimally utilize PCM devices at their full capacity, it is necessary to develop an analog channel coding scheme (see previous chapter). Instead of hand-designing analog channel coders for storing images on these devices – often an arduous task – we propose an adaptive autoencoder framework that accomplishes joint source-channel coding. We find that our proposed joint source-channel coding scheme is able to achieve a rate-distortion performance that is superior to that achieved by JPEG combined with a binary channel coder. We simulated the input-output behavior of PCM

using measured data from the devices. From this, our proposed network learned about the characteristics of PCMs and adapted the encoder and decoder to effectively use these devices for storing image data. Thus, in principle, our method can adapt to the statistics of a broad range of data types and memory devices, potentially even adapting to changes in device properties over time. Additional challenges will arise when attempting to code images onto a physical PCM array. Specifically, the PCM devices in a fabricated array will likely not be completely independent, as cross-talk in the read/write process can modify their statistics. We simulated a set of uniformly behaving devices, while in reality each storage device would have somewhat different input-output characteristics. Finally, resistance drift will slowly change the properties of the PCM devices over time. We start to address some of these issues in the following chapters.

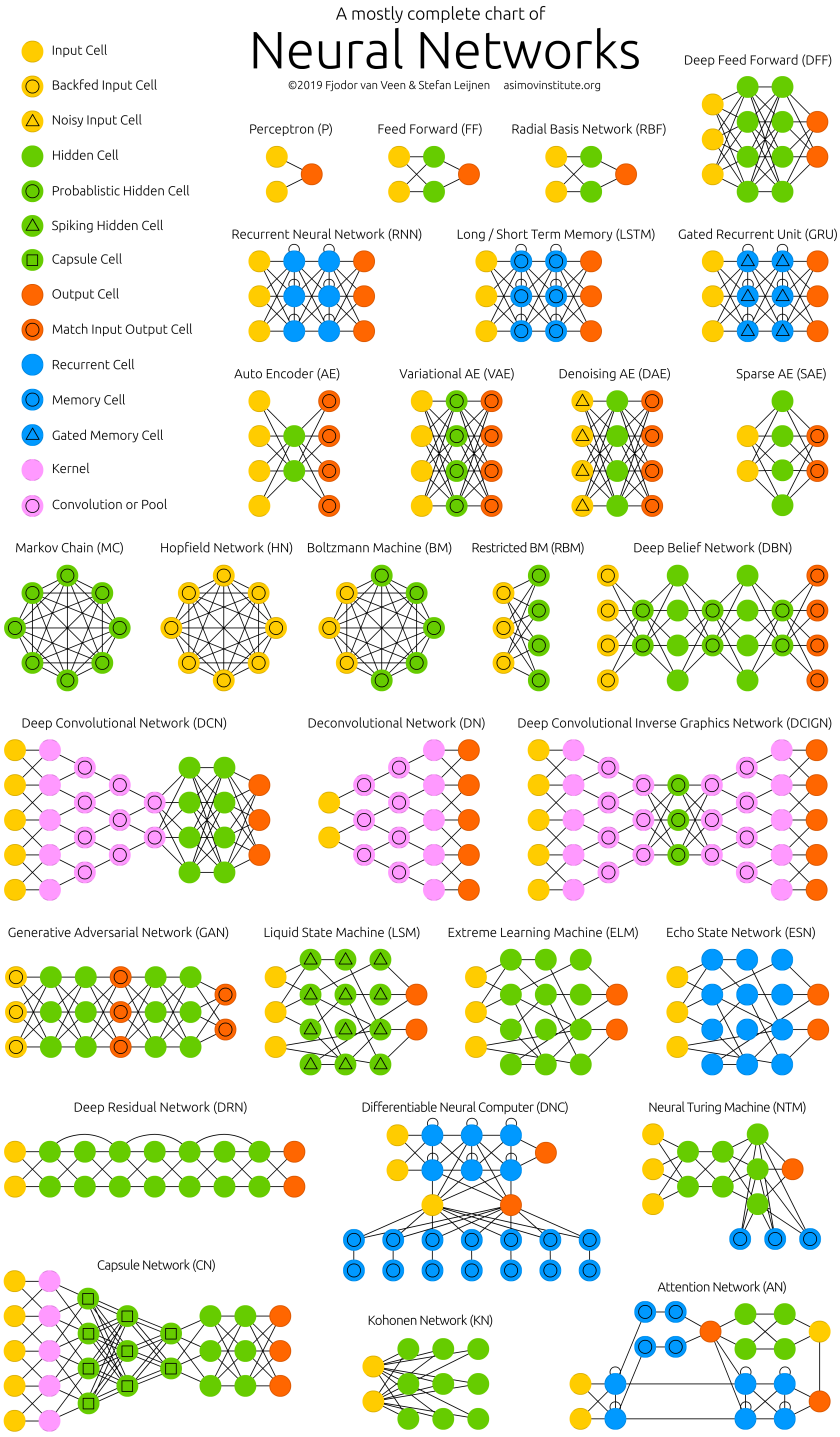

Fig. 2.6. Summary chart of main neural network architectures. Reproduced with permission. Original source: [116]

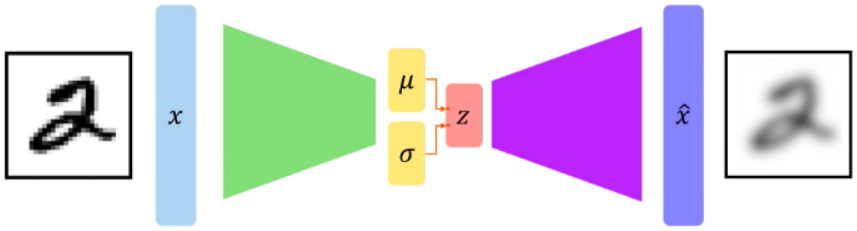

Fig. 2.7. Example of a variational autoencoder trained to generate MNIST digits with latent variables parameterized as Gaussians with means μ and variances σ. Reproduced with permission. Original source: [5].

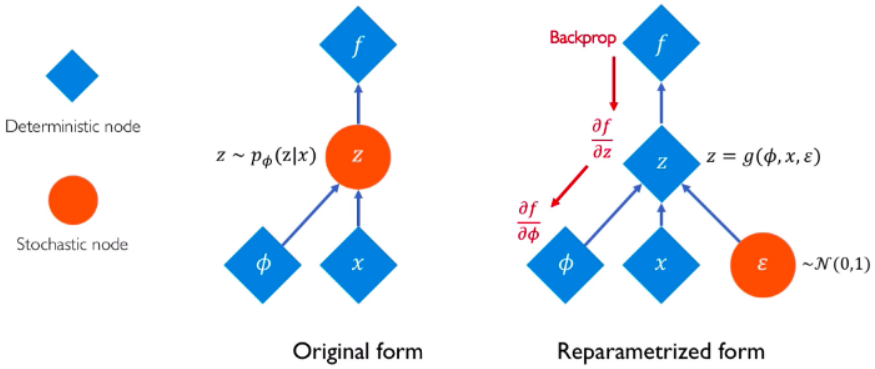

Fig. 2.8. Illustration of reparameterization trick for a Gaussian distribution. Reproduced with permission. Original source: [5].

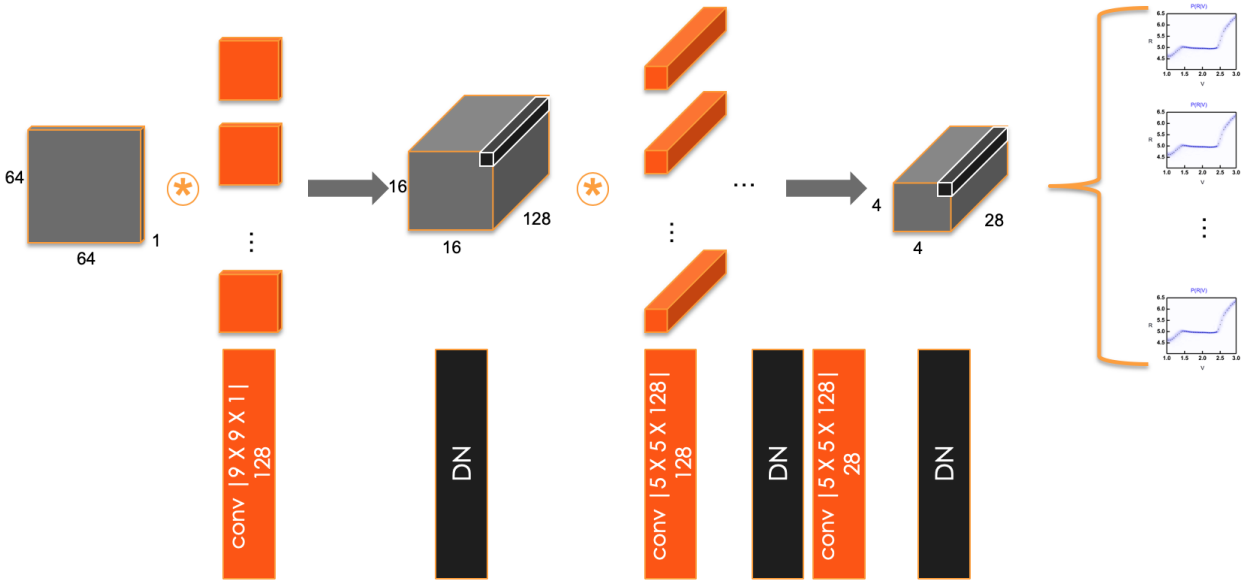

Fig. 2.9. Architecture of the encoding neural network (decoder is just inverse of these steps). A convolutional kernal is taken and slid over the image, taking the inner product at each location and storing this scalar in a tensor (gray rectangular boxes). Because the strides for the convolution are greater than one pixel, the spatial dimension of the maps is reduced each layer. The depth of the tensor is equal to the number of filters used (i.e. one feature map for each filter). Divisive normalization is applied across feature maps at each spatial location (indicated by $1 \times 1 \times n$ black boxes in the figure).

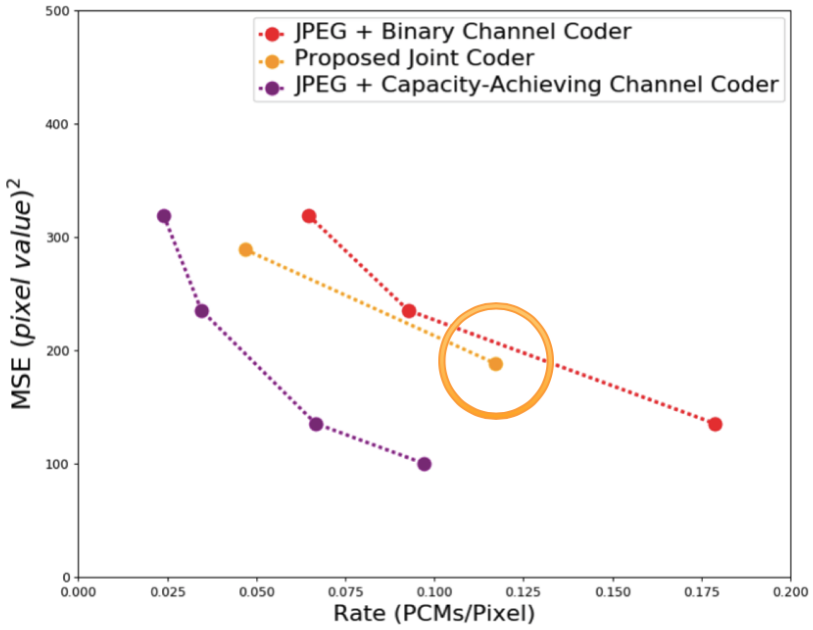

Fig. 2.10. Rate-distortion curve for three different storage methods: In red and purple are results for the JPEG codec combined with two hypothetical channel coders that achieve transmission rates of (respectively) 1 bit and 2.68 bits across each PCM device. The proposed joint coder is shown in yellow. Dots indicate an average MSE achieved for 24 gray-scale images from the Kodak dataset.

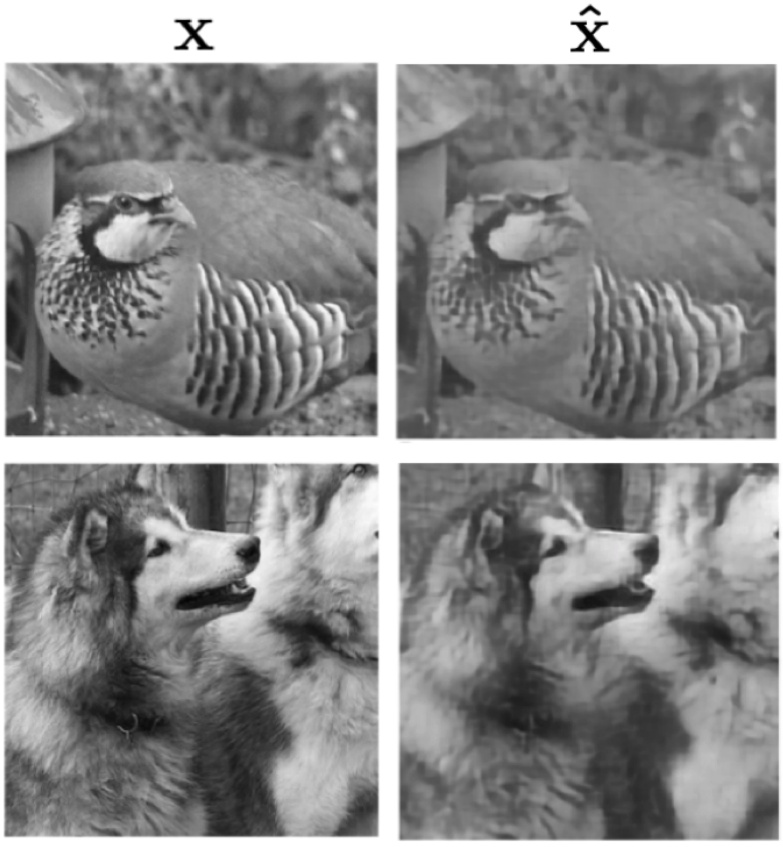

Fig. 2.11. Example storing 256 x 256 pixel natural images onto 7,680 PCM devices. Original images (left) and their reconstructions (right).

Chapter 3

Image storage with neural networks; experimental realization with device non-idealities

> Like Master Splinter said, it's not the weapon that's important, it's the Ninja wielding it.
>
> Leonardo

Let's realize this experimentally where we now have two new issues: device-device variability and drift.

3.1 Background

3.1.1 Device-Device Variation

Within an array, different devices can have slightly to significantly different response properties. A simple case of this was shown in Chapter 1. In the specific array we're dealing with for this current study, a different kind of variability is present: each device will behave, statistically, the same as any other device with the exception of some percentage of devices. This random subset will be "error cells", in that the device will be stuck at some random value around the minimum programable value (more on this below). In general then, developing ways to deal with this variability is an important step towards reliably using these arrays at large scales.

3.1.2 Impermanence (Drift)

Device drift is also a major practical problem in emerging memory devices . The specific physical reasons for why this happens (and the dynamics) depend on the device but essentially thermal fluctuations in the material cause their structures to change over time (and as their resistance is a property of this physical structure, the resistance changes along with it). In the case of RRAM, eventually the filamental bridge breaks, and the two sides are no longer connected . Thus, for example, a device programmed with $R_0 = 5.5$ might drift $R_{100} = 5.3$, $R_{1000} = 5.36$, and $R_{10000} = 5.25$. And, because of the physical mechanism for this drift, the same device starting at the same R_0 might make a different drift trajectory on another run. Modeling this drift and correcting for it is a challenge that we'll tackle later, but for the time being our goal is to see if our scheme can be robust to it.

3.2 Setup

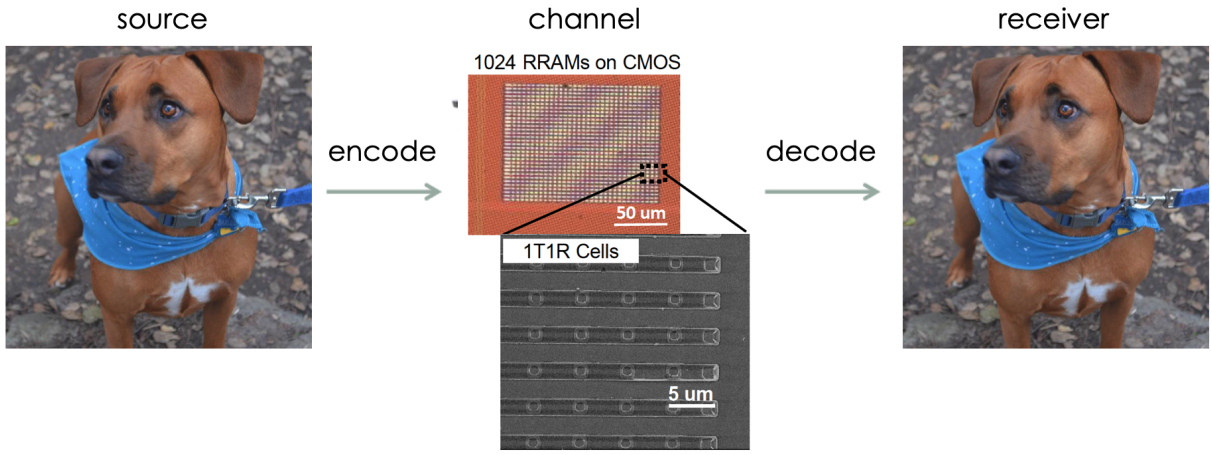

Fig. 3.1. General setup: similar to before, but now instead of an array of simulated devices we have an actual experimental array consisting of 1024 RRAM devices.

3.2.1 RRAM

3.2.2 Modeling $P(R|V)$

Again, from our perspective, we're after the conditional distribution that describes channel dynamics. For our case, we have two sources of noise that we'll be modeling (we'll see the effects of drift later but won't explicitly model it): stochasticity in target values (same idea as we've seen up until this point) and "error cells".

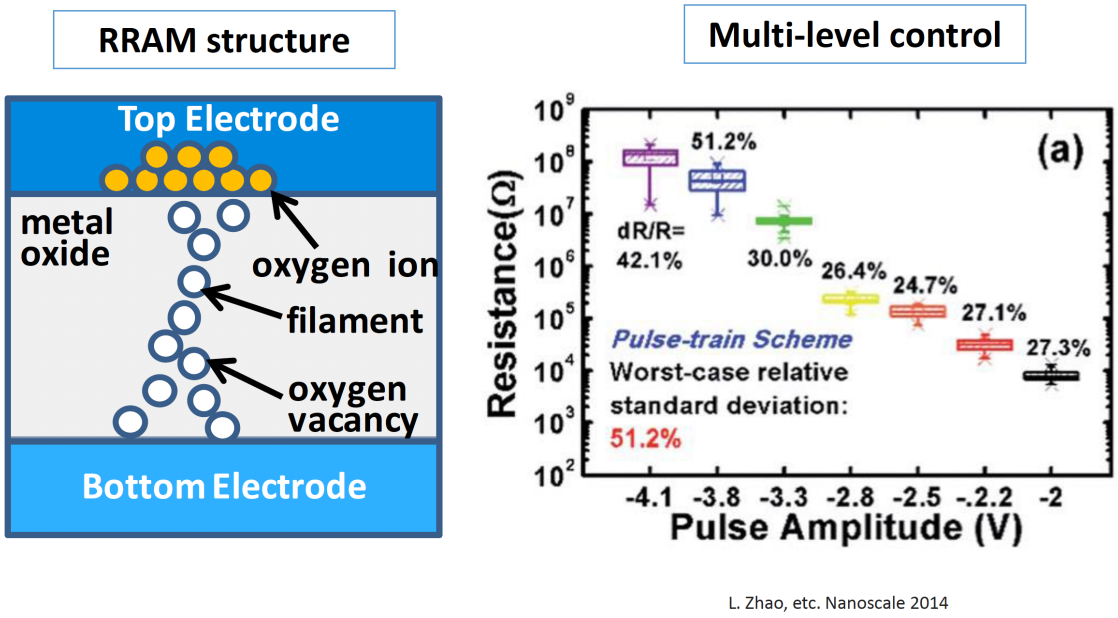

Fig. 3.2. Illustration of RRAM device: similar to PCM in that we have a material sandwiched between two electrodes and the property of this material determines the devices resistance. Here though, a filament grows between the two electrodes. The larger this filament, the lower the resistance. Similar to PCM devices, these devices generally have the ability to reach values along a continuum (right). Reproduced with permission. © 2018 IEEE, © 2014 IEEE. Original sources: [2], [134].

The first comes from the write process. The second comes from the fact that the full dynamic range of resistances for an individual device can differ (i.e. this may vary from device-device). For most instances, this is not an issue, and programming proceeds as planned. However, some small fraction of devices (dispersed randomly throughout the array) have dynamic ranges that are outside the target value. This results in an R_M which is around the minimum value.

So we have this familiar picture, Fig. 3.7, with the same cost function as before (now just a different channel model).

3.3 Results

3.3.1 JPEG Comparison

Again, it's hard to construct a true comparison point without a channel coder for the separate scheme. Additionally, JPEG can't compress down to the level we're at. I.e. we can't extend JPEG curve as low as us but even for it's lowest point we're still beating it.

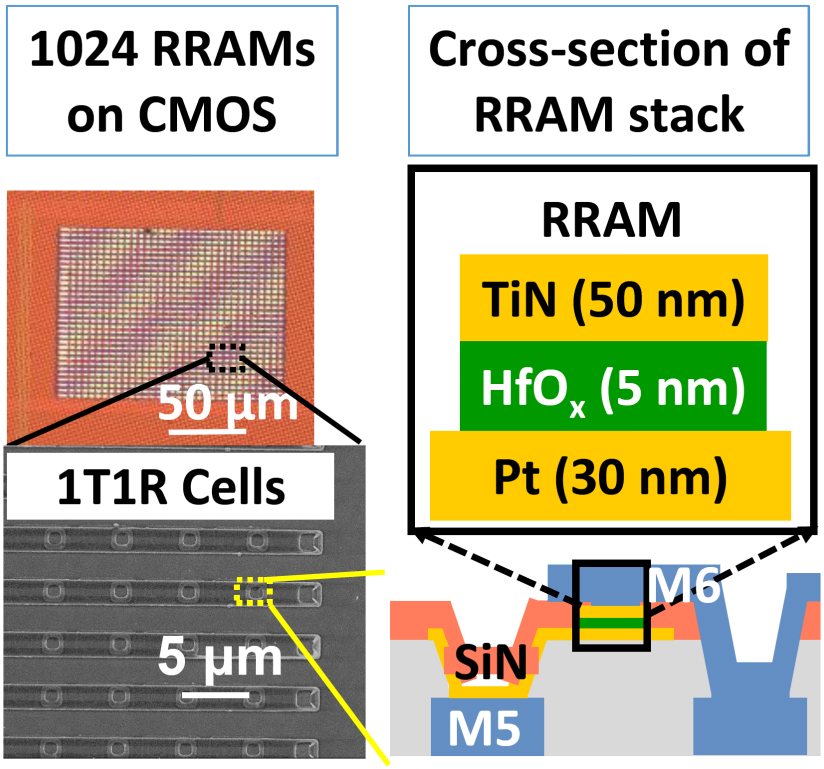

Fig. 3.3. Left: SEM image of the 1K 1T1R analog-valued RRAM array used. Right: Cross-section schematic of the RRAM stack.

3.3.2 Robust to Device Failure (Device-Device Variability

Encouragingly, we trained at one error and found both (i) a graceful degradation as error rate increased and (ii) improved performance as error rate decreased. I.e. we found that the network generalizes across failure rates. Importantly, as mentioned previously, this was one of the major problems with the traditional discrete approach: "catastrophic" performance as error rates go beyond that which the code was designed for and performance "saturation" for error rates lower than that which the code was designed for.

Various error rates (0.5%, 1%, 2%, 5%, and 10%) were dialed in by randomly selecting a set of cells and SETTING them to LRS (Fig. 3.6). Fig. 3.10 shows reconstructions (top) for the original image and corresponding MSE (bottom) for the different error rates (all images are scaled to have pixel values of mean = 0, variance = 1)

3.3.3 Robust to Drift

Again, we see a graceful degradation as drift increases. See Fig. 3.12.

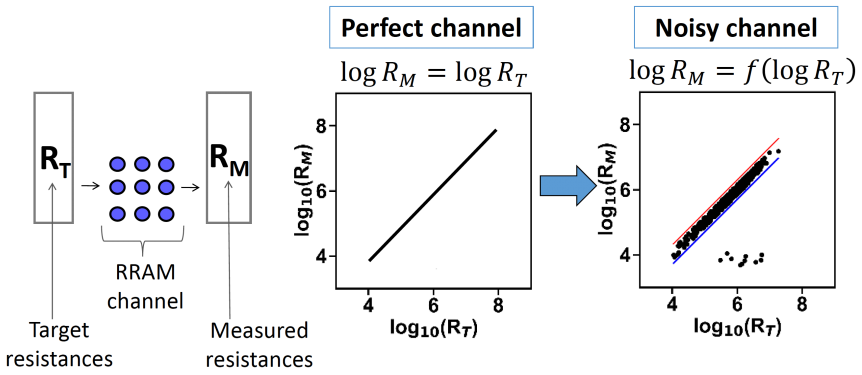

Fig. 3.4. Similar setup as before but now with RRAM devices instead of PCM. In the case of the perfect RRAM channel, $R_T = R_M$, and the input-output relationship would just be a strait line. For our case though, we have noise introduced by the measurement process (shown between the blue and red lines) as well as noise induced by "device failures" (dots at the bottom, outside these lines).

3.4 Discussion

Our method empirically showed error-resilience against device non-idealities such as error cells, cycle-to-cycle and device-to-device variations, and resistance relaxation. This experimental demonstration was very encouraging for us, motivating us to try and directly tackle device-device variation and drift. See the next Chapter.

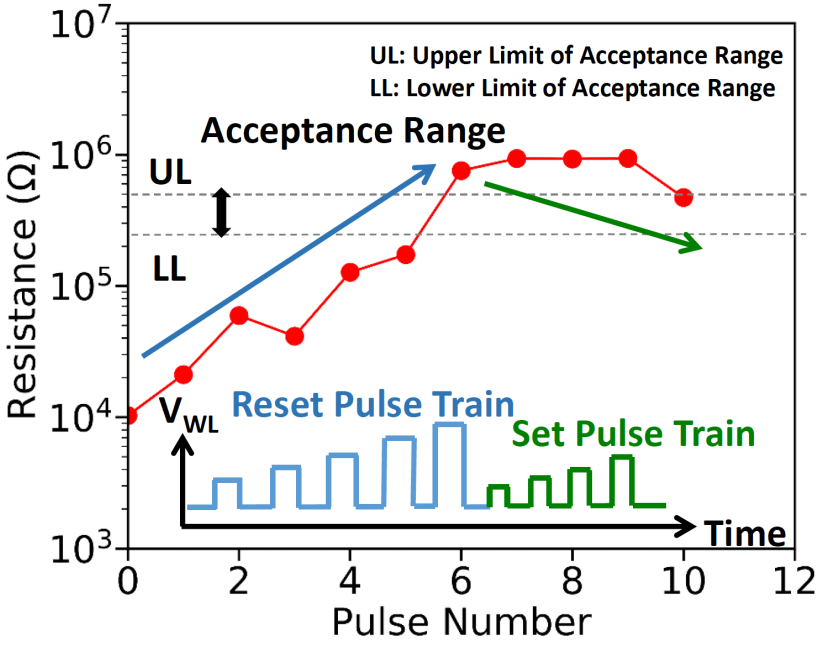

Fig. 3.5. Pulsing scheme to store a value in a device: Example writing with DD-ISPP showing the over-programming can be fixed by programming in the opposite direction, and starting from minimal voltage when changing the direction can minimize programming across the range and thus save energy. Main figure: measured resistance as a function of pulse number. Inset: write pulse train waveform.

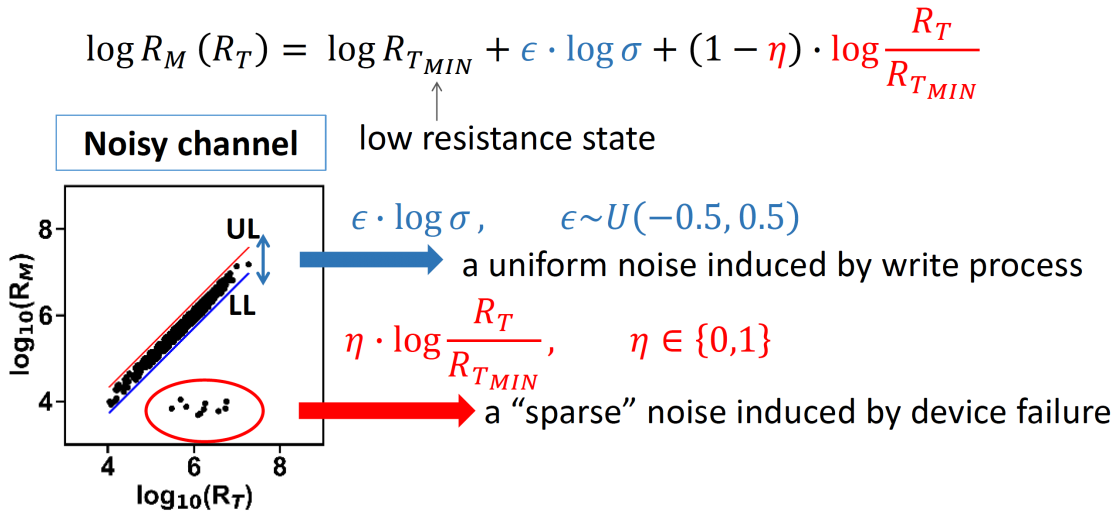

Fig. 3.6. Modeling the different sources of noise in the channels.

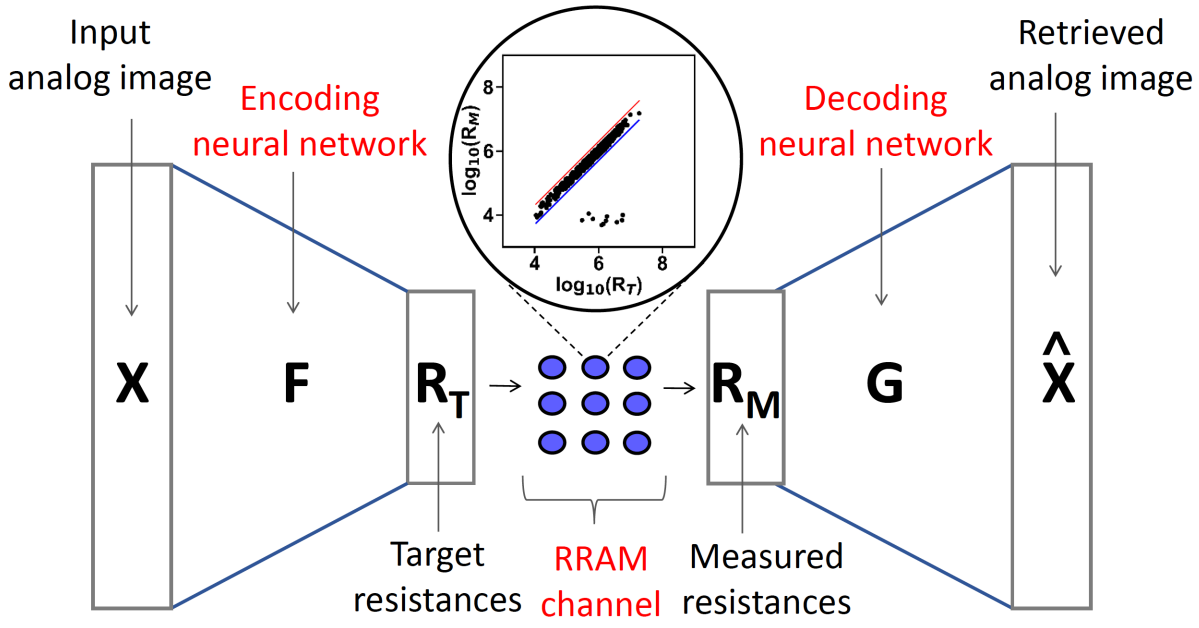

Fig. 3.7. Same setup as in the PCM simulation case, but now using RRAM devices described above.

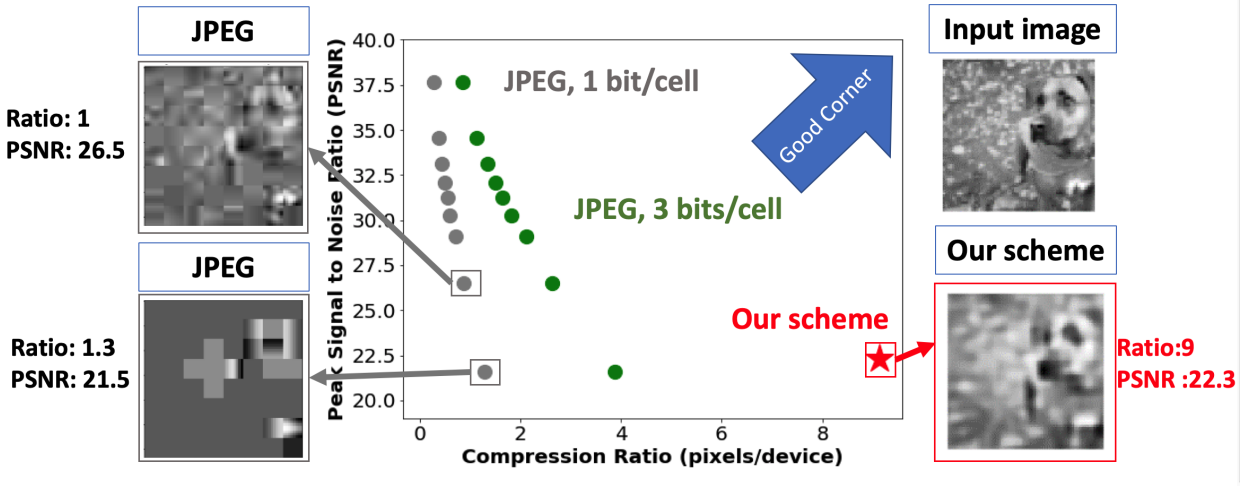

Fig. 3.8. Comparison between our method storing a test image and a set of hypothetical schemes using JPEG source coding followed by a hypothetical channel coder. Note: JPEG is unable to compress down to the level that we are, and even at the level they can compress down to, the images are perceptually much worse.

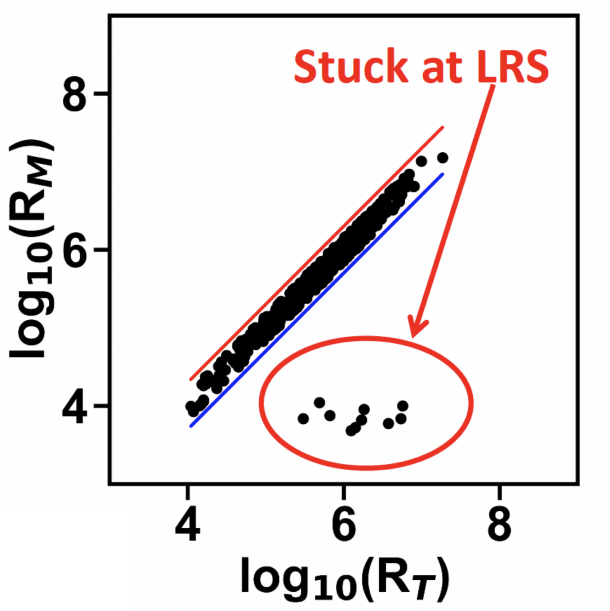

Fig. 3.9. Error cells: looking specifically at cells that were stuck at the low resistance state.

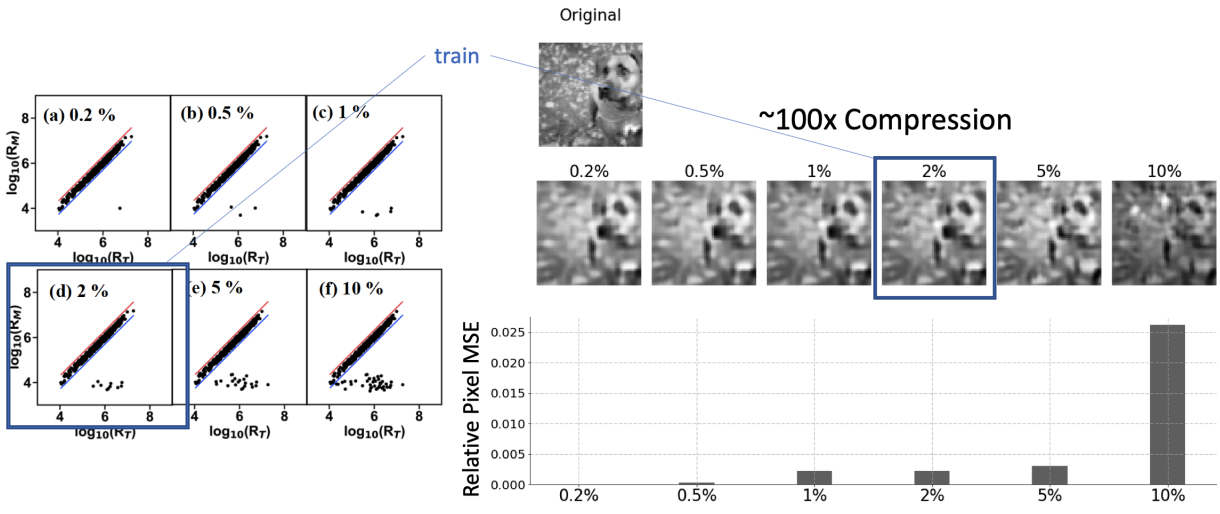

Fig. 3.10. Reconstructions (top) for the original image and their corresponding relative mean squared error (MSE) (bottom) to the MSE of 0.2% error rate (Fig. 12 (b)) for various error rates (0.2%, 0.5%, 1%, 2%, 5%, 10%) of devices. The network is trained with 2% noise. The network shows the ability to generalize across noise values – taking advantage of the lower noise to perform better reconstructions while tolerating errors up to 5% before suffering serious reconstruction degradation.

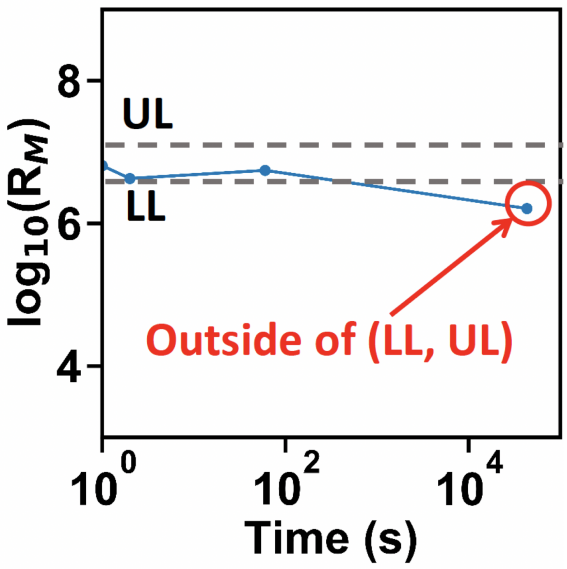

Fig. 3.11. Drift example: value starts off in range but over time drifts outside of expected (modeled) range.

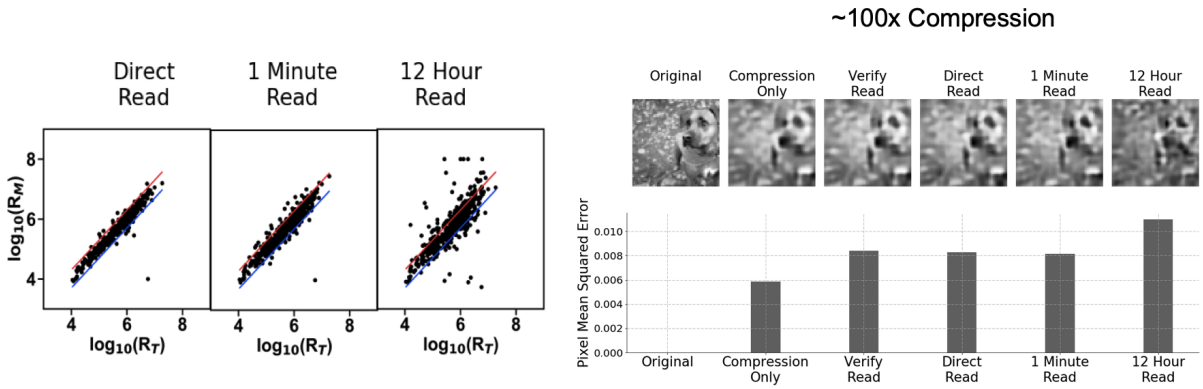

Fig. 3.12. Left: drift of values as time progresses. Right: corresponding reconstruction from drifted values. Note that the reconstruction is robust to a significant portion of values drifting outside of the expected region, even at this high level of compression.

Chapter 4

Image storage with neural networks; future work – explicitly tackling device-device variability and drift.

> I always thought the future of ninjutsu would be taller.
>
> Raphael

Let's scale up and attempt to tackle non-stationarity (in space and time (i.e. device-device variability and drift)). Specifically, let's collect more data from the devices and explicitly try and tackle device-device variation and drift. I.e. would like to get enough data to model drift and, potentially, variation across array. This variation comes form interaction between the devices and how they are laid out and addressed on the array, so it's possible there's a way to model this variation (we just didn't have enough data before). Here, we're using a commercial fabrication facility at TSMC which has 1 Million device arrays.

4.1 Getting Better Device-Device Variation Data:

Now, instead of just one instance, we can store the same image 44 times and look at the raw values and reconstructed samples. See Fig. 4.1 and Fig. 4.2.

4.2 Getting Better Drift Data

Additionally, we can collect drift data for the large collection of devices to more systematically see how drift affects storage. See Fig. 4.3.

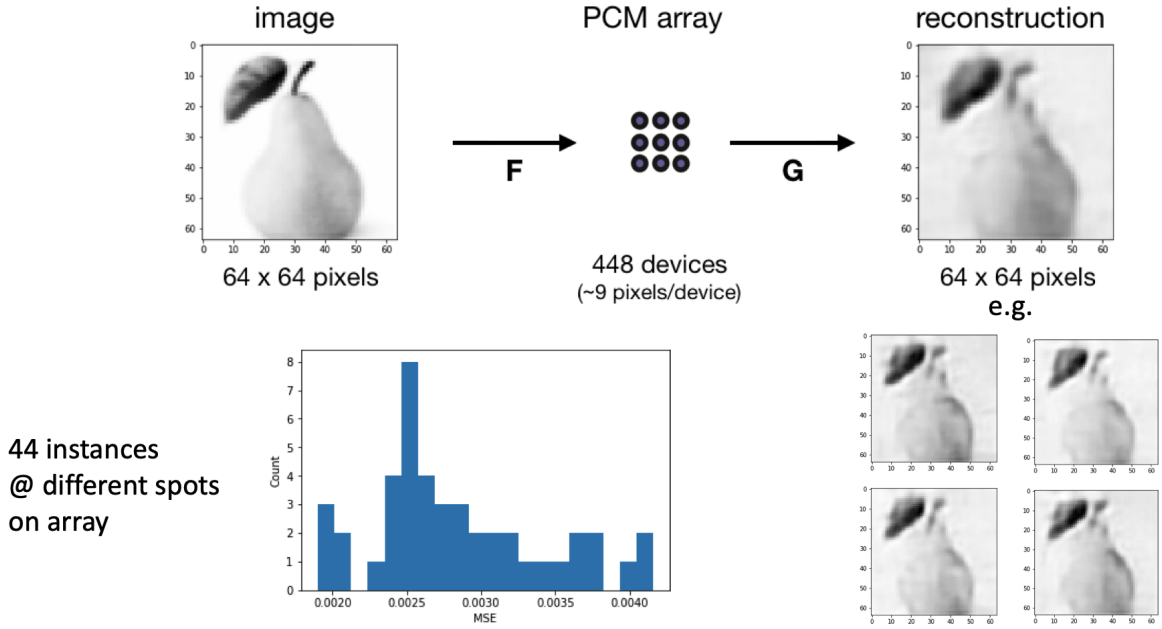

Fig. 4.1. Illustration of multiple-instance storage: here, the image of the pear is put through the encoding neural network, resulting in 448 values. These values are then taken and put on the 1M array in 44 different location (so 44 x 448 values are stored on the array in total). We can then reconstruct these 44 different instances and look at the distribution of MSE values that results around the array.

4.3 Results

4.3.1 Modeling Drift Data

With this new data, we can use Bayesian estimation to infer initial values. It turns out that you can model the distribution of R_t as conditionally Gaussian! (surprise!).

$$\text{Model: } R_t|R_0,t \sim N\left(R_0+\mu t, \sigma^2 t^2\right), \quad f\left(R_t|R_0,t\right) = \frac{1}{\sigma t\sqrt{2\pi}} e^{-\left(R_t-R_0-\mu t\right)^2/\left(2\sigma^2 t^2\right)} \tag{4.1}$$

$$\text{Bayes is bae: } f\left(R_0|R_t,t\right) = \frac{f\left(R_t|R_0,t\right) f\left(R_0|t\right)}{\int_a^b f\left(R_t|R_0,t\right) f\left(R_0|t\right) dR_0} \tag{4.2}$$

$$\text{Flat prior: } f\left(R_0|t\right) = f\left(R_0\right) = \frac{1}{b-a}, \quad f\left(R_0|R_t,t\right) = \frac{f\left(R_t|R_0,t\right) f\left(R_0|t\right)}{\int_a^b f\left(R_t|R_0,t\right) dR_0} \tag{4.3}$$

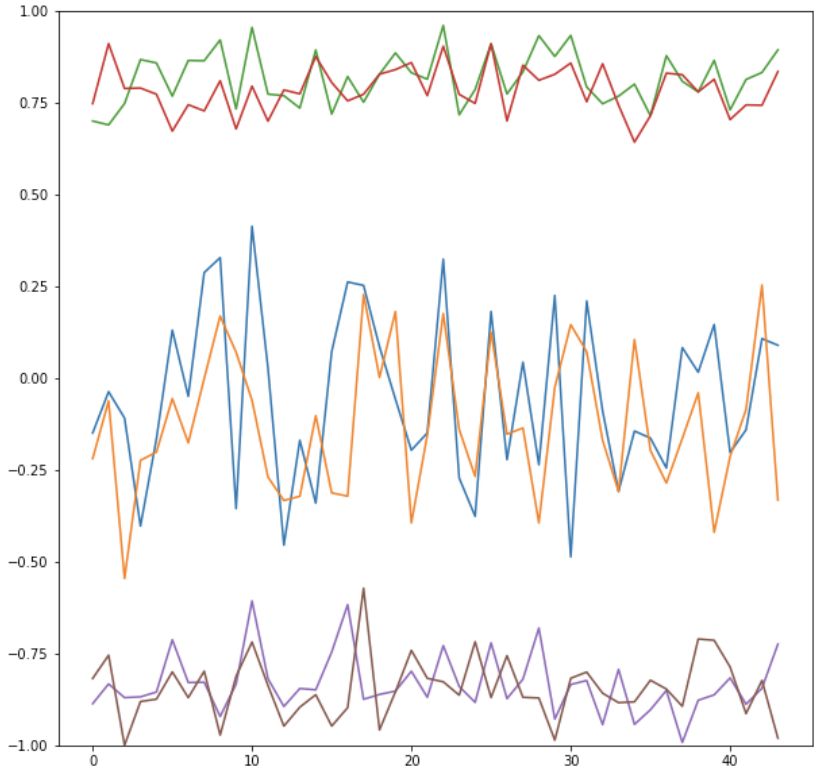

Fig. 4.2. We can look at these 44 different locations and see how values vary inter-location. Here's an example of six different values that were stored in the 44 different locations. The six different values are from three "groups": two are supposed to be around 0.8, two are supposed to be around -0.2, and two are supposed to be around -0.8. We can see that as we look across the locations, the values change with some regularity that we might be able to model and correct for.

Here we use a simple flat prior so we don't have to know the distribution of R for each device (that would be absurd/would require another set of parameters for each memory element).

4.3.2 Modeling Device-Device Variation

Finally, tying back to modeling device-device variation, we can further look at how drift changes with regions and potentially learn transformations around the array (but data on this is very preliminary so this is still work in progress). See Fig. 4.5.

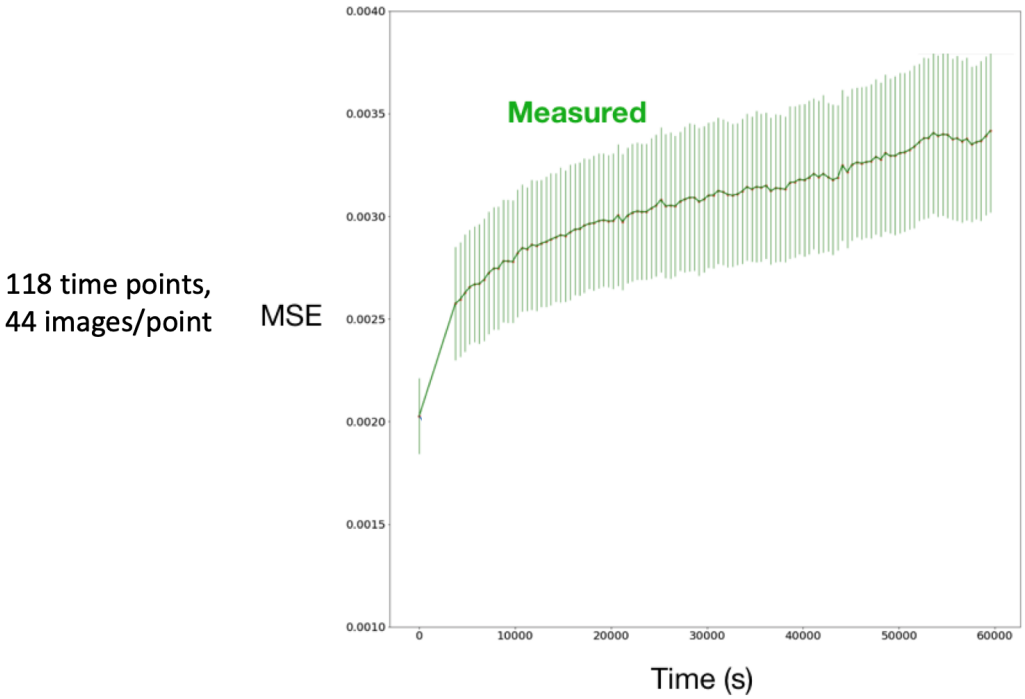

Fig. 4.3. Using these 44 x 448 values, we can see how the values (and thus the reconstructions) drift with time. Shown here are those same 44 instances mentioned previously, but now measured and reconstructed at different times.

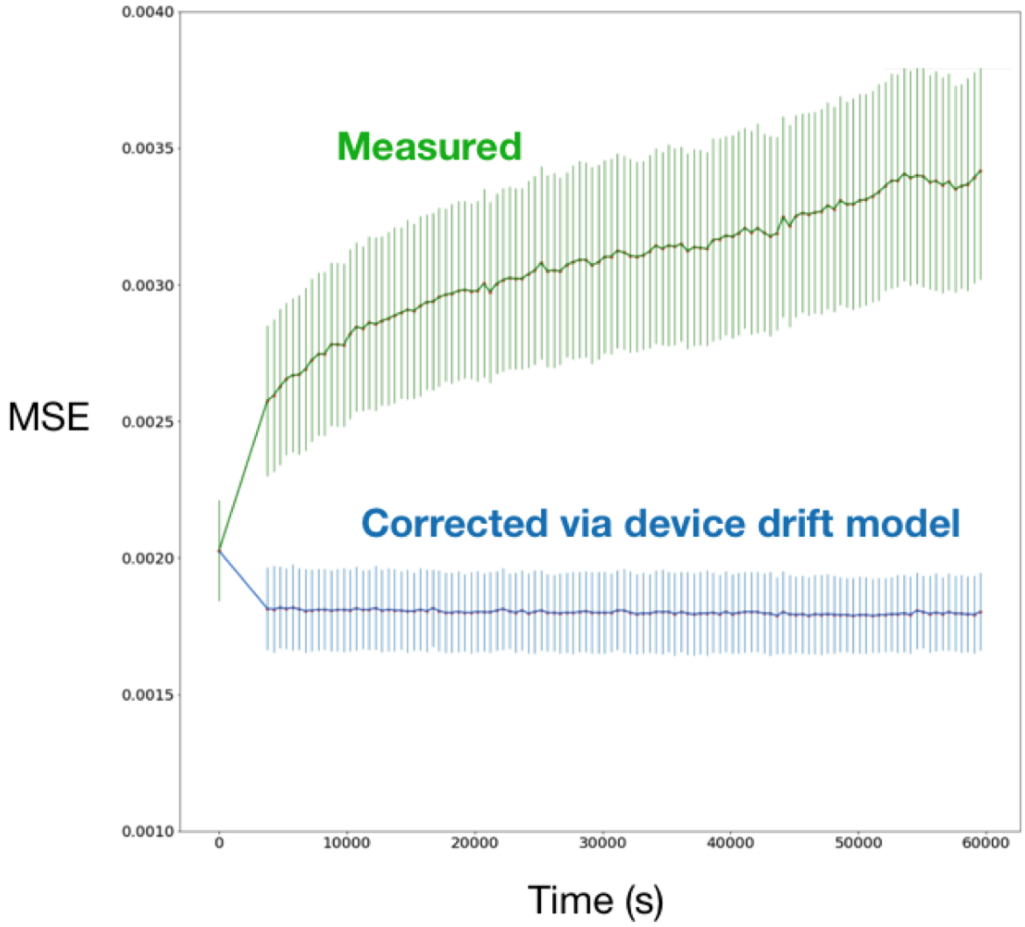

Fig. 4.4. Using the data and model described previously, we can correct for the effect of drift and adjust the values to what we expect they would have been. This results in a significant improvement over the un-corrected case and essentially an elimination of the drift effect (on the timescale measured).

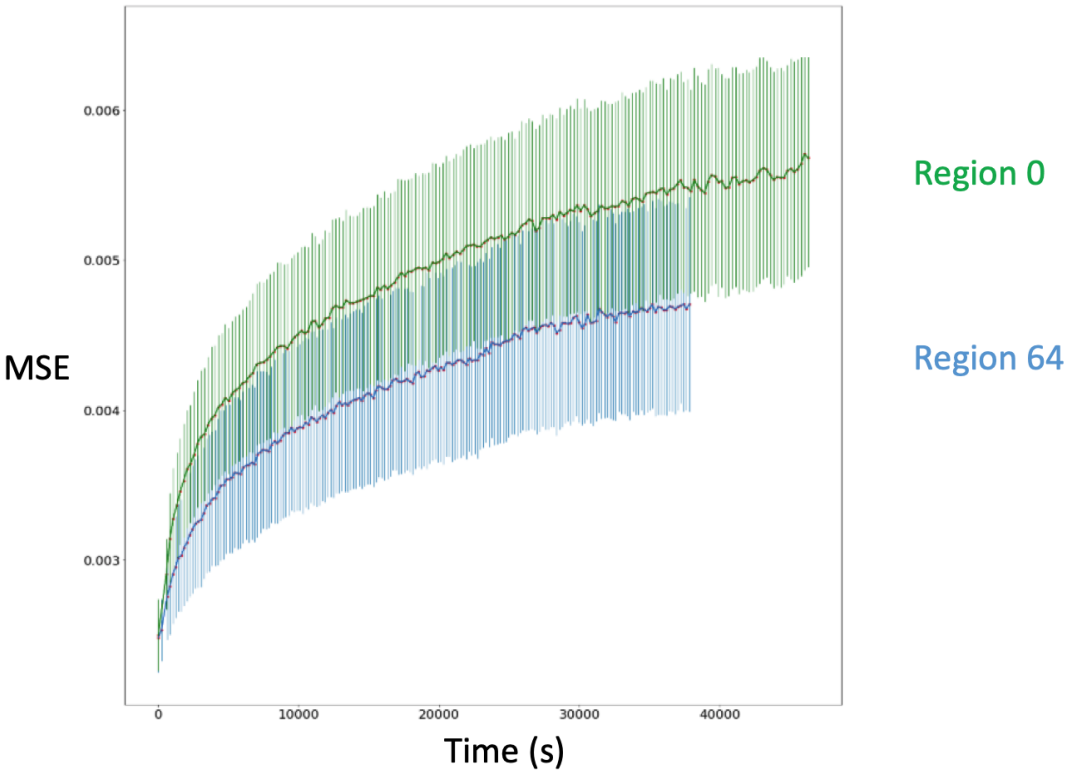

Fig. 4.5. We can repeat the above process but now at two completely separate regions on the array. I.e. take an image, encode it into the 448 values, store 44 instances of these values in one area of the array and then another 44 in another. This can then give us a sense of how drift and device-device effects might interact as we move around the array.

Chapter 5

Conclusion

We have explored a new approach for storing analog-valued media in emerging analog-valued devices showing that analog coding strategies have the potential to create robust storage with relatively low complexity. Designing systems to match the statistics of the source, channel, and task-at-hand provides unique opportunities not captured by systems that try to enforce determinism. However, there is no free lunch, and what such systems gain in efficiency they likely lose in generality.

Universal memory design has been a great boon to the development of digital technology and designing memory systems for specific tasks has many unexplored risks in terms of backward compatibility. That said, with the rise of machine learning applications, the nature of computation is changing, and many of the data-centric applications of the present and future do not require perfect data retrieval. With the data deluge faced by internet companies, it is often not the storage of data that is important, but the storage of information which performs well when retrieved for human or statistical machine inference (e.g., perception of audio and video). While losing some generality, representations that reduce the redundancy of signals to match the statistics of storage media and applications can be more efficient than universal representations.

Efficient integrated circuit implementations of such circuits are also an active area of investigation [81], lending further weight to the idea of analog-valued ECCs that can learn on-chip dependencies and adapt to nonstationary statistics as devices age. Another active area of research is in using analog-valued variable resistors as synaptic elements in such neural circuits [72], [90], [15], leading to the possibility that storage and error-correction could someday be integrated within a single memory substrate.

As briefly mentioned previously, it is interesting to note that from the perspective of neuroscience, neural systems are faced with a very similar task. Given the statistical redundancy of natural stimuli and the stochasticity of perception and signal transmission, efficient coding strategies must be devised to transmit compressed representations of stimuli that are intrinsically robust to neural noise and resource-efficient [61], [115].

For those who have better things to do with their time and have scrolled to the very end, I'll finish with some more general points:

- Analog computation is hard because we have to simultaneously optimize many things...so let's learn it.
- For the particular case of storage, we've looked at how anolog methods can outperform digital ones and have proposed a joint autoencoder framework that learns a mapping between data and devices.
- The framework is able to adapt to many types of devices, but the approach we've presented here is just the starting point.
- End of Moore's Law $\implies$ need to move away from the discrete, deterministic computing paradigm
- Computing should match physical primitives with desired computation (just as nature does). To try and capture Feynman's spirit: We might be in a situation where the shortest description of the phenomena we're interested in are the phenomena themselves. [41] Thus, coming up with ways to develop effective computational abstractions and implementations for things like economic, ecological, and biological systems will be challenging. [42] That's all to say: By golly it's a wonderful problem, because it doesn't look so easy![43]

[41] Feynman specifically referred to quantum processes, but we're finding out that many physical systems are in a similar situation.

[42] But this might actually be futile. See Steven Wolfram's idea of "computational irreducibility" [123] for an interesting perspective that I find appealing.[I]

[43] Hito no yume wa...owara nē!

[I] Though I also take seriously "the long-lingering proposition that thermodynamics drives the self-organization and evolution of natural systems and, similarly, that thermodynamics might drive the self-organization and evolution of future computing" [22]. So there might be hope.

Bibliography

[1] Scott Aaronson. "Guest column: NP-complete problems and physical reality". In: *ACM Sigact News* 36.1 (2005), pp. 30–52.

[2] Deji Akinwande. "Memory, Memristors, and Atomristors". In: *IEEE Micro* 38.5 (2018), pp. 50–52.

[3] Emrah Akyol et al. "On zero-delay source-channel coding". In: *IEEE Transactions on Information Theory* 60.12 (2014), pp. 7473–7489.

[4] Jaan Altosaar. *Tutorial - What is a variational autoencoder?* URL: `https://jaan.io/what-is-variational-autoencoder-vae-tutorial/`.

[5] Alexander Amini and Ava Soleimany. *MIT Deep Learning 6.S191 - Lecture 4*. URL: `http://introtodeeplearning.com/`.

[6] Paul B Badcock, Karl J Friston, and Maxwell JD Ramstead. "The hierarchically mechanistic mind: A free-energy formulation of the human psyche". In: *Physics of life Reviews* (2019).

[7] Johannes Ballé, Valero Laparra, and Eero Simoncelli. "End-to-end optimized image compression". In: *5th International Conference on Learning Representations, ICLR 2017*. 2019.

[8] Anthony J Bell. "Levels and loops: the future of artificial intelligence and neuroscience". In: *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences* 354.1392 (1999), pp. 2013–2020.

[9] Charles H Bennett. "The thermodynamics of computation—a review". In: *International Journal of Theoretical Physics* 21.12 (1982), pp. 905–940.

[10] Kwabena Boahen. "A neuromorph's prospectus". In: *Computing in Science & Engineering* 19.2 (2017), pp. 14–28.

[11] John BohannonApr. *Who's the Michael Jordan of computer science? New tool ranks researchers' influence*. Dec. 2017. URL: `https://www.sciencemag.org/news/2016/04/who-s-michael-jordan-computer-science-new-tool-ranks-researchers-influence`.

[12] Eirina Bourtsoulatze, David Burth Kurka, and Deniz Gündüz. "Deep joint source-channel coding for wireless image transmission". In: *IEEE Transactions on Cognitive Communications and Networking* 5.3 (2019), pp. 567–579.

[13] M Breitwisch et al. "Novel lithography-independent pore phase change memory". In: *2007 IEEE Symposium on VLSI Technology*. IEEE. 2007, pp. 100–101.

[14] Jelle Bruineberg and Erik Rietveld. "Self-organization, free energy minimization, and optimal grip on a field of affordances". In: *Frontiers in human neuroscience* 8 (2014), p. 599.

[15] Geoffrey W Burr et al. "Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element". In: *IEEE Transactions on Electron Devices* 62.11 (2015), pp. 3498–3507.

[16] Geoffrey W Burr et al. "Phase change memory technology". In: *Journal of Vacuum Science & Technology B, Nanotechnology and Microelectronics: Materials, Processing, Measurement, and Phenomena* 28.2 (2010), pp. 223–262.

[17] Yubei Chen, Dylan Paiton, and Bruno Olshausen. "The sparse manifold transform". In: *Advances in Neural Information Processing Systems*. 2018, pp. 10513–10524.

[18] Leon Chua. "Memristor-the missing circuit element". In: *IEEE Transactions on circuit theory* 18.5 (1971), pp. 507–519.

[19] Sae-Young Chung. "On the construction of some capacity-approaching coding schemes". PhD thesis. Massachusetts Institute of Technology, 2000.

[20] GF Close et al. "Device, circuit and system-level analysis of noise in multi-bit phase-change memory". In: *2010 International Electron Devices Meeting*. IEEE. 2010, pp. 29–5.

[21] Thomas M Conte et al. "Rebooting computing: The road ahead". In: *Computer* 50.1 (2017), pp. 20–29.

[22] Tom Conte et al. "Thermodynamic Computing". In: *arXiv preprint arXiv:1911.01968* (2019).

[23] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[24] Jim Crutchfield. *Computational Mechanics Archive*. 2020. URL: `http://csc.ucdavis.edu/~cmg/index.htm`.

[25] Hugo De Man. "Ambient intelligence: gigascale dreams and nanoscale realities". In: *ISSCC. 2005 IEEE International Digest of Technical Papers. Solid-State Circuits Conference, 2005.* IEEE. 2005, pp. 29–35.

[26] Stanislas Dehaene, Hakwan Lau, and Sid Kouider. "What is consciousness, and could machines have it?" In: *Science* 358.6362 (2017), pp. 486–492.

[27] Ezequiel A Di Paolo, Elena Clare Cuffari, and Hanne De Jaegher. *Linguistic bodies: The continuity between life and language*. Mit Press, 2018.

[28] Edmund JF Dickinson, Henrik Ekström, and Ed Fontes. "COMSOL Multiphysics®: Finite element software for electrochemical analysis. A mini-review". In: *Electrochemistry communications* 40 (2014), pp. 71–74.

[29] The Economist. *After Moore's Law*. 2016.

[30] J Ericsson. *Ericsson mobility report,(November)*. 2014.

[31] S Burc Eryilmaz et al. "Experimental demonstration of array-level learning with phase change synaptic devices". In: *2013 IEEE International Electron Devices Meeting*. IEEE. 2013, pp. 25–5.

[32] Dave Evans. "The internet of things: How the next evolution of the internet is changing everything". In: *CISCO white paper* 1.2011 (2011), pp. 1–11.

[33] Richard P Feynman. "Simulating physics with computers". In: *Int. J. Theor. Phys* 21.6/7 (1999).

[34] David J Field. "Relations between the statistics of natural images and the response properties of cortical cells". In: *Josa a* 4.12 (1987), pp. 2379–2394.

[35] Pal Anders Floor and Tor A Ramstad. "Optimality of dimension expanding shannon-kotel'nikov mappings". In: *2007 IEEE Information Theory Workshop*. IEEE. 2007, pp. 289–294.

[36] Pål Anders Floor et al. "Zero delay joint source channel coding for multivariate Gaussian sources over orthogonal Gaussian channels". In: *Entropy* 15.6 (2013), pp. 2129–2161.

[37] Pål Anders Floor et al. "Zero-delay joint source-channel coding for a bivariate Gaussian on a Gaussian MAC". In: *IEEE Transactions on Communications* 60.10 (2012), pp. 3091–3102.

[38] O Fresnedo et al. "Comparison between analog joint source-channel coded and digital BICM systems". In: *2011 IEEE International Conference on Communications (ICC)*. IEEE. 2011, pp. 1–5.

[39] Karl Friston. "Life as we know it". In: *Journal of the Royal Society Interface* 10.86 (2013), p. 20130475.

[40] Karl Friston et al. "Active inference and epistemic value". In: *Cognitive neuroscience* 6.4 (2015), pp. 187–214.

[41] Rudolf M Füchslin et al. "Morphological computation and morphological control: steps toward a formal theory and applications". In: *Artificial Life* 19.1 (2013), pp. 9–34.

[42] Arild Fuldseth and Tor A Ramstad. "Bandwidth compression for continuous amplitude channels based on vector approximation to a continuous subset of the source

signal space". In: *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 4. IEEE. 1997, pp. 3093–3096.

[43] Shaun Gallagher. *Enactivist interventions: Rethinking the mind*. Oxford University Press, 2017.

[44] Michael Gastpar, Bixio Rimoldi, and Martin Vetterli. "To code, or not to code: Lossy source-channel communication revisited". In: *IEEE Transactions on Information Theory* 49.5 (2003), pp. 1147–1158.

[45] Jacqueline Gottlieb et al. "Information-seeking, curiosity, and attention: computational and neural mechanisms". In: *Trends in cognitive sciences* 17.11 (2013), pp. 585–593.

[46] Jonathan Gottschall. *The storytelling animal: How stories make us human*. Houghton Mifflin Harcourt, 2012.

[47] Lewis Grozinger et al. "Pathways to cellular supremacy in biocomputing". In: *Nature communications* 10.1 (2019), pp. 1–11.

[48] Frank T Hady et al. "Platform storage performance with 3D XPoint technology". In: *Proceedings of the IEEE* 105.9 (2017), pp. 1822–1833.

[49] Alon Halevy, Peter Norvig, and Fernando Pereira. "The unreasonable effectiveness of data". In: *IEEE Intelligent Systems* 24.2 (2009), pp. 8–12.

[50] Mohamed Hassanin and Javier Garcia-Frias. "Analog joint source channel coding over non-linear acoustic channels". In: *2013 47th Annual Conference on Information Sciences and Systems (CISS)*. IEEE. 2013, pp. 1–6.

[51] Fredrik Hekland. "On the Design and Analysis of Shannon-Kotel'nikov Mappings for Joint-Source-Channel Coding". In: (2007).

[52] Fredrik Hekland, Pal Anders Floor, and Tor A Ramstad. "Shannon-kotel-nikov mappings in joint source-channel coding". In: *IEEE Transactions on Communications* 57.1 (2009), pp. 94–105.

[53] Christian Herrera. "Psychophysics of Color Vision". PhD thesis. UC Irvine, 2016.

[54] Geoffrey Hinton et al. "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". In: *IEEE Signal processing magazine* 29.6 (2012), pp. 82–97.

[55] Yichuan Hu, Javier Garcia-Frias, and Meritxell Lamarca. "Analog joint source-channel coding using non-linear curves and MMSE decoding". In: *IEEE Transactions on Communications* 59.11 (2011), pp. 3016–3026.

[56] Christian Hubicki et al. "Walking and running with passive compliance: Lessons from engineering: A live demonstration of the ATRIAS biped". In: *IEEE Robotics & Automation Magazine* 25.3 (2018), pp. 23–39.

[57] Aapo Hyvärinen, Jarmo Hurri, and Patrick O Hoyer. *Natural image statistics: A probabilistic approach to early computational vision.* Vol. 39. Springer Science & Business Media, 2009.

[58] Daniele Ielmini and Yuegang Zhang. "Analytical model for subthreshold conduction and threshold switching in chalcogenide-based memory devices". In: *Journal of Applied Physics* 102.5 (2007), p. 054517.

[59] Michael I Jordan. "Artificial intelligence—The revolution hasn't happened yet". In: *Harvard Data Science Review* (2019).

[60] Samruddhi Y Kahu, Rajesh B Raut, and Kishor M Bhurchandi. "Review and evaluation of color spaces for image/video compression". In: *Color Research & Application* 44.1 (2019), pp. 8–33.

[61] Yan Karklin and Eero P Simoncelli. "Efficient coding of natural images with a population of noisy linear-nonlinear neurons". In: *Advances in neural information processing systems*. 2011, pp. 999–1007.

[62] Fred Keijzer. "Representation in dynamical and embodied cognition". In: *Cognitive Systems Research* 3.3 (2002), pp. 275–288.

[63] Jonghong Kim and Wonyong Sung. "Rate-0.96 LDPC decoding VLSI for soft-decision error correction of NAND flash memory". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 22.5 (2013), pp. 1004–1015.

[64] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[65] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[66] Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. "Representations of space and time in the maximization of information flow in the perception-action loop". In: *Neural computation* 19.9 (2007), pp. 2387–2432.

[67] Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. "Tracking information flow through the environment: Simple cases of stigmerg". In: *In: Artificial Life IX: Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems, edited by Pollack, J.* MIT Press. 2004.

[68] Victoria Kostina and Sergio Verdú. "Lossy joint source-channel coding in the finite blocklength regime". In: *IEEE Transactions on Information Theory* 59.5 (2013), pp. 2545–2575.

[69] Vladimir A Kotelnikov. *The theory of optimum noise immunity.* Vol. 34. McGraw-Hill, 1959.

[70] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. "Estimating mutual information". In: *Physical review E* 69.6 (2004), p. 066138.

[71] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[72] Duygu Kuzum, Shimeng Yu, and HS Philip Wong. "Synaptic electronics: materials, devices and applications". In: *Nanotechnology* 24.38 (2013), p. 382001.

[73] ANDREA LEONARDO Lacaita et al. "Electrothermal and phase-change dynamics in chalcogenide-based memories". In: *IEDM Technical Digest. IEEE International Electron Devices Meeting, 2004.* IEEE. 2004, pp. 911–914.

[74] Subhaneil Lahiri, Jascha Sohl-Dickstein, and Surya Ganguli. "A universal tradeoff between power, precision and speed in physical communication". In: *arXiv preprint arXiv:1603.07758* (2016).

[75] Thien Truong Nguyen Ly. "Efficient Hardware Implementations of LDPC Decoders, through Exploiting Impreciseness in Message-Passing Decoding Algorithms". PhD thesis. 2017.

[76] David JC MacKay and David JC Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[77] David JC MacKay and Radford M Neal. "Near Shannon limit performance of low density parity check codes". In: *Electronics letters* 32.18 (1996), pp. 1645–1646.

[78] Carver Mead. "Neuromorphic electronic systems". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1629–1636.

[79] Shakir Mohamed. *Machine Learning Trick of the Day (4): Reparameterisation Tricks*. Oct. 2015. URL: `http://blog.shakirm.com/2015/10/machine-learning-trick-of-the-day-4-reparameterisation-tricks/`.

[80] PAUL Munro and DAVID Zipser. "Image compression by back propagation: an example of extensional programming". In: *Models of cognition: rev. of cognitive science* 1.208 (1989), p. 1.

[81] Emre Neftci et al. "Event-driven contrastive divergence for spiking neuromorphic systems". In: *Frontiers in neuroscience* 7 (2014), p. 272.

[82] T Nirschl et al. "Write strategies for 2 and 4-bit multi-level phase-change memory". In: *2007 IEEE International Electron Devices Meeting*. IEEE. 2007, pp. 461–464.

[83] J Kevin O'Regan and Alva Noë. "A sensorimotor account of vision and visual consciousness". In: *Behavioral and brain sciences* 24.5 (2001), pp. 939–973.

[84] Christopher Olah. *Calculus on Computational Graphs: Backpropagation*. 2015. URL: `http://colah.github.io/posts/2015-08-Backprop/`.

[85] Bruno A Olshausen and David J Field. "Emergence of simple-cell receptive field properties by learning a sparse code for natural images". In: *Nature* 381.6583 (1996), pp. 607–609.

[86] Bruno A Olshausen and David J Field. "What is the other 85 percent of V1 doing". In: *L. van Hemmen, & T. Sejnowski (Eds.)* 23 (2006), pp. 182–211.

[87] N Papandreou et al. "Drift-tolerant multilevel phase-change memory". In: *2011 3rd IEEE International Memory Workshop (IMW)*. IEEE. 2011, pp. 1–4.

[88] Giovanni Pezzulo, Francesco Rigoli, and Karl J Friston. "Hierarchical active inference: A theory of motivated control". In: *Trends in cognitive sciences* 22.4 (2018), pp. 294–306.

[89] David Philipona, J Kevin O'Regan, and J-P Nadal. "Is there something out there? Inferring space from sensorimotor dependencies". In: *Neural computation* 15.9 (2003), pp. 2029–2049.

[90] Mirko Prezioso et al. "Training and operation of an integrated neuromorphic network based on metal-oxide memristors". In: *Nature* 521.7550 (2015), pp. 61–64.

[91] Jan M Rabaey and Sharad Malik. "Challenges and solutions for late-and post-silicon design". In: *IEEE Design & Test of Computers* 25.4 (2008), pp. 296–302.

[92] Jan M Rabaey et al. "Workloads of the future". In: *IEEE Design & Test of Computers* 25.4 (2008), pp. 358–365.

[93] Majid Rabbani. "JPEG2000: Image compression fundamentals, standards and practice". In: *Journal of Electronic Imaging* 11.2 (2002), p. 286.

[94] Tor A Ramstad. "Shannon mappings for robust communication". In: *Telektronikk* 98.1 (2002), pp. 114–128.

[95] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. "Stochastic backpropagation and approximate inference in deep generative models". In: *arXiv preprint arXiv:1401.4082* (2014).

[96] Erik Rietveld and Julian Kiverstein. "A rich landscape of affordances". In: *Ecological Psychology* 26.4 (2014), pp. 325–352.

[97] Sergio Matiz Romero et al. "Analog joint source channel coding for wireless optical communications and image transmission". In: *Journal of Lightwave Technology* 32.9 (2014), pp. 1654–1662.

[98] Daniel L Ruderman. "The statistics of natural images". In: *Network: computation in neural systems* 5.4 (1994), pp. 517–548.

[99] Rahul Sarpeshkar. "Analog versus digital: extrapolating from electronics to neurobiology". In: *Neural computation* 10.7 (1998), pp. 1601–1638.

[100] Odelia Schwartz and Eero P Simoncelli. "Natural signal statistics and sensory gain control". In: *Nature neuroscience* 4.8 (2001), pp. 819–825.

[101] Abu Sebastian, Manuel Le Gallo, and Evangelos Eleftheriou. "Computational phase-change memory: beyond von Neumann computing". In: *Journal of Physics D: Applied Physics* 52.44 (2019), p. 443002.

[102] Claude E Shannon. "A mathematical theory of communication". In: *Bell system technical journal* 27.3 (1948), pp. 379–423.

[103] Claude E Shannon. "The bandwagon". In: *IRE Transactions on Information Theory* 2.1 (1956), p. 3.

[104] Claude Elwood Shannon. "Communication in the presence of noise". In: *Proceedings of the IRE* 37.1 (1949), pp. 10–21.

[105] R Sharpeshkar. *Ultra low power bioelectronics: Fundamentals, biomedical applications, and bio-inspired systems*. 2010.

[106] Sara Sheehan and Yun S Song. "Deep learning for population genetic inference". In: *PLoS computational biology* 12.3 (2016), e1004845.

[107] Eero P Simoncelli and Bruno A Olshausen. "Natural image statistics and neural representation". In: *Annual review of neuroscience* 24.1 (2001), pp. 1193–1216.

[108] Peter Sterling and Simon Laughlin. *Principles of neural design*. MIT Press, 2015.

[109] Alexander V Terekhov and J Kevin O'Regan. "Space as an invention of active agents". In: *Frontiers in Robotics and AI* 3 (2016), p. 4.

[110] Naftali Tishby and Daniel Polani. "Information theory of decisions and actions". In: *Perception-action cycle*. Springer, 2011, pp. 601–636.

[111] George Toderici et al. "Full resolution image compression with recurrent neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5306–5314.

[112] Alan Mathison Turing. "The chemical basis of morphogenesis". In: *Bulletin of mathematical biology* 52.1-2 (1990), pp. 153–197.

[113] Vinay A Vaishampayan and Sueli I Rodrigues Costa. "Curves on a sphere, shift-map dynamics, and error control for continuous alphabet sources". In: *IEEE Transactions on Information Theory* 49.7 (2003), pp. 1658–1672.

[114] Ilia Valov et al. "Nanobatteries in redox-based resistive switches require extension of memristor theory". In: *Nature communications* 4.1 (2013), pp. 1–9.

[115] Lav R Varshney, Per Jesper Sjöström, and Dmitri B Chklovskii. "Optimal information storage in noisy synapses under resource constraints". In: *Neuron* 52.3 (2006), pp. 409–423.

[116] Fjodor van Veen and Stefan Leijnen. *The Neural Network Zoo*. Apr. 2019. URL: `https://www.asimovinstitute.org/neural-network-zoo/`.

[117] Paul FMJ Verschure. "Distributed adaptive control: a theory of the mind, brain, body nexus". In: *Biologically Inspired Cognitive Architectures* 1 (2012), pp. 55–72.

[118] Paul FMJ Verschure, Thomas Voegtlin, and Rodney J Douglas. "Environmentally mediated synergy between perception and behaviour in mobile robots". In: *Nature* 425.6958 (2003), pp. 620–624.

[119] John Von Neumann. "Probabilistic logics and the synthesis of reliable organisms from unreliable components". In: *Automata studies* 34 (1956), pp. 43–98.

[120] John Von Neumann. *The Computer and the Brain*. Yale University Press, 2000.

[121] Wikipedia contributors. *Hamming(7,4) — Wikipedia, The Free Encyclopedia*. [Online; accessed 19-May-2020]. 2020. URL: `https://en.wikipedia.org/w/index.php?title=Hamming(7,4)&oldid=949665414`.

[122] Wikipedia contributors. *Memristor — Wikipedia, The Free Encyclopedia*. [Online; accessed 19-May-2020]. 2020. URL: `https://en.wikipedia.org/w/index.php?title=Memristor&oldid=955296621`.

[123] Stephen Wolfram. *A new kind of science*. Vol. 5. Wolfram media Champaign, IL, 2002.

[124] Stephen Wolfram. "Undecidability and intractability in theoretical physics". In: *Physical Review Letters* 54.8 (1985), p. 735.

[125] David H Wolpert. "Computational capabilities of physical systems". In: *Physical Review E* 65.1 (2001), p. 016128.

[126] H-S Philip Wong et al. "Metal–oxide RRAM". In: *Proceedings of the IEEE* 100.6 (2012), pp. 1951–1970.

[127] H-S Philip Wong et al. "Phase change memory". In: *Proceedings of the IEEE* 98.12 (2010), pp. 2201–2227.

[128] Sung Sik Woo, Jaewook Kim, and Rahul Sarpeshkar. "A digitally programmable cytomorphic chip for simulation of arbitrary biochemical reaction networks". In: *IEEE transactions on biomedical circuits and systems* 12.2 (2018), pp. 360–378.

[129] Bill Wooten and David L Miller. "The psychophysics of color". In: *Color categories in thought and language* (1997), pp. 59–88.

[130] Yilun Xu et al. "A Theory of Usable Information Under Computational Constraints". In: *arXiv preprint arXiv:2002.10689* (2020).

[131] Raymond W Yeung. *A first course in information theory*. Springer Science & Business Media, 2012.

[132] Ryan V Zarcone et al. "Analog coding in emerging Memory Systems". In: *Scientific Reports* 10.1 (2020), pp. 1–13.

[133] Ryan Zarcone et al. "Joint source-channel coding with neural networks for analog data compression and storage". In: *2018 Data Compression Conference*. IEEE. 2018, pp. 147–156.

[134] Liang Zhao and Yoshio Nishi. "Pulse-train measurement techniques: An RRAM test vehicle for in-depth physical understanding". In: *2014 12th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*. IEEE. 2014, pp. 1–4.

[135] Xin Zheng et al. “Error-Resilient Analog Image Storage and Compression with Analog-Valued RRAM Arrays: An Adaptive Joint Source-Channel Coding Approach”. In: *2018 IEEE International Electron Devices Meeting (IEDM)*. IEEE. 2018, pp. 3–5.

[136] Song-Chun Zhu. *How to reach the moon: do we know that we are not doing research in the wrong way? (2010)*. URL: `http://www.stat.ucla.edu/~sczhu/research_blog.html`.

[137] Daniel Zoran. “Natural Image Statistics for Human and Computer Vision”. PhD thesis. Citeseer, 2013.