NX-RESEARCH

White Paper

Introduction to Narrative Computation

A framework for designing AI systems that are accurate, robust, and explainable

by
Femi Fadeyi
Founder
NX-RESEARCH INC.
2023-May

EXECUTIVE SUMMARY

Narrative computation is a framework for designing AI systems that leverages robust and precise descriptions of the computational states and processes of a system using a vocabulary of causal-model objects that is well-founded internally, within the system itself. The purpose of this white paper is to provide an introduction to narrative computation and the research being conducted by NX-RESEARCH to deploy narrative computation into useful AI systems.

Narrative objects (semantically-grounded computational-state descriptions) allow writing 'programs' within a system such that 'programs' can leverage accurate behavioral descriptions external to the system (at multiple levels of abstraction, over multiple epochs, and for multiple scopes of concern), to control the behavior of the overall system itself, in a reliable and robust way.

For example, narrative computation allows writing a 'program' of the following form:

- (pseudocode) <if> AgentSelf 'IS CONFUSED ABOUT REQUEST' <trigger> 'CONCEIVE PLAN TO RESOLVE CONFUSION' <then> 'EXECUTE PLAN';
- The structure of narrative computation systems enable defining contextually-reasonable and specific measures of 'CONFUSION', as well as contextually-reasonable evaluations of 'CONCEIVE PLAN TO RESOLVE CONFUSION' and 'EXECUTE PLAN';
- The incorporation of such contextually-reasonable semantics into the underlying system itself are what make it possible to 'program' narrative computation systems such that they can robustly evaluate themselves and exhibit reasonable-behaviors with consideration of prior goals, current circumstances, novel observations, and novel hypotheses.

At NX-RESEARCH, we have implemented narrative computation approaches into our NX™ systems, which include NX-Utility™, Curious-NX™ and others. Using the narrative computation approach, we are able to ensure that the internal measures and evaluations of computational states, simulations, goals and plans in our NX™ systems are accurate and useful; which we evaluate using our NX-Utility™ modules that focus on external-reasonableness, robustness, explainability, and others. Our current research objectives are to continue development of core architecture features and utility evaluation systems, while we use our Curious-NX™ agent platform to test more complex 'programs' and to evaluate how our architecture and 'programs' behave when provided with larger datasets and under multi-modal conditions. Our hope is that narrative computation systems will demonstrate scalable efficiency and practical performance (e.g. AGIEval by W. Zhong et al.), while retaining the defining characteristics of our NX-Utility™ modules. If this hope is sustained under the current research roadmap, we expect that narrative computation technology will add significant value for mission-critical AI applications that demand accuracy, robustness, and explainability.

This white paper will provide an overview of key concepts, operating principles, and features of the narrative computation framework, explain how narrative computation relates to and differs from other AI approaches, highlight some of NX-RESEARCH's systems and architectures that implement aspects of the narrative computation framework, then finally discuss some forward-looking opportunities and potential limitations of the narrative computation framework in the context of contemporary approaches to addressing critical challenges in AI research.

TABLE OF CONTENTS

1. OVERVIEW

1.1. Background and Motivation for Narrative Computation Research

1.1.1. There is increasing demand for deploying AI systems for myriad use cases and applications.

1.1.2. However, certain mission-critical applications,such as accounting, engineering, autonomous vehicles, medical diagnosis and many others, require a very high degree of accuracy, robustness and explainability that is presently not available.

1.1.3. Certain implementations of deep neural networks, generative AI, and related approaches to AI show significant promise for many applications, especially for text, image, and audio generation - the quality that is currently available indicates these tools will soon become indispensable in many such applications. However, accuracy, robustness and explainability do remain difficult challenges for even the most capable state-of-the-art systems like GPT-4, SAM, PALM-2 and others.

1.1.4. Separately, some historical approaches to AI had advocated for more 'symbolic' AI systems which have different operating principles from gradient-based approaches (like generative AI). Good-Old-Fashioned-AI approaches, such as expert systems, rule discovery systems, were originally promising but found difficulty generalizing, eventually resulting in profound setbacks for the field of AI research.

1.1.5. Despite setbacks, symbolic AI is still survived today by an array of modern approaches such as…

1.1.5.1. ?

1.1.6. Narrative computation requires a system of discrete rules and as such would be reasonable to consider as a form of "symbolic" AI. Separately, though narrative computation is not a gradient-based approach, it is compatible with gradient-based approaches in certain specific ways.

1.1.7. At NX-RESEARCH, we are investigating if narrative computation is a viable framework for improving the accuracy, robustness and explainability of AI systems while remaining efficient and performant.

1.1.8. NX-RESEARCH is a research startup focused on developing AI systems that are accurate, efficient and explainable; our vision is a smarter world where more humans have greater access to enjoy engaging, fulfilling and rich experiences empowered by responsible and explainable AI systems.

1.2. Objectives of this White Paper

1.2.1. A key objective of this paper is to provide an introduction to the conceptual underpinnings of narrative computation, situate the framework among other approaches to AI, and inform the opinions of the technical reader about the narrative computation framework, including potential possibilities and limitations of the approach.

1.2.2. This paper also aims to inform the reader about some of the innovative narrative computation technologies that are under active research and development by NX-RESEARCH; specifically NX-Utility™ and Curious-NX™.

1.3. Target Audience

1.3.1. This white paper is written to be approachable for a general audience, but it may be helpful to have some background understanding of programming concepts. This white paper is especially tailored to inform a technical audience who have an interest in gaining an introductory understanding of how narrative computation proposes to work, especially in contrast to other approaches to AI.

1.4. Scope and Limitations

1.4.1. This white paper will focus on explaining the key concepts and elements of narrative computation, specifically how it relates to and differs from other approaches to AI systems. This white paper will also highlight some of the ongoing research within NX-RESEARCH and discuss some of our systems and architectures.

1.4.2. Importantly, this white paper will NOT dive into the details of implementing narrative computation systems or algorithms; and it will also NOT provide functional details, performance assessments or comparisons to other systems for NX-RESEARCH systems or tools.

1.4.3. And though this white paper acknowledges ethical and social implications associated with AI systems, it does NOT extensively address the ethical considerations for AI research in general, nor for narrative computation in particular.

2. Key Concepts and Definitions

2.1. Narrative Computation

2.1.1.1. The narrative computation approach can be explained as: utilizing narrative objects with robustly reliable and useful semantically-descriptive 'symbols' enables creating systems using narrative programs such that the system itself is describable and robust.

2.1.1.2. If the 'symbols' utilized are very generic (ex: XNOR), they will contain low descriptive value in comparison to more situationally-specific symbols (ex: CURIOSITY_WAS_SATISFIED_POSITIVELY)

2.1.1.3. Designing the underlying set of 'symbols' within a narrative computing system is an art that requires full view of the overall objectives of the system, the particular strategy for organizing symbols, thoughtful planning, careful experimentation and evaluation of 'programming' impact of symbol sets, and judicious allocation of design effort.

2.1.1.4. Writing useful and robust 'programs' is also an art of itself.

2.1.1.5. The central observation of narrative computation is that systems implemented with useful and robust narrative objects and programs, can be designed to evaluate themselves usefully and robustly

2.1.1.6. Self-observation

2.1.1.7. Cost function, Feedback, data, training

2.1.1.8. The design of the reward, prioritization and decision-making systems is not trivial!

2.2. Narrative Objects

2.2.1.1. More details and examples of how narrative objects work

2.2.1.2. Narrative objects are semantically-grounded computational-state descriptions;

2.2.1.3. Where semantically-grounded refers to the notion that the description uses 'symbols' which have meanings within the system that are consistent with our usual understanding of those symbols outside of the system

2.2.1.4. Example would be that for a 'symbol' with a name 'CHECK_IF_SEMANTIC_OBJECT_UNDERSTOOD', the particular routing outcome of following such a 'symbol' should match the expectation described by its label in a consistent and robust way

2.2.1.5. Specifically, consistent implies that the behaviors enforced by following the symbol should have the some consistent expected effect in all circumstances where it is invoked (and importantly it should be explicit in not allowing invocation in ambiguous and inappropriate circumstances)

2.2.1.6. And, robust implies that this design is thoroughly verified across a wide range of operational circumstances and practical edge cases which may occur even with low frequency

2.2.1.7. And computational-state refers to the notion that the description (set of symbols) represents particular aspect or sequence of some computation ('computational state')

2.2.1.8. Example would be that a 'computation' which followed some sequential set of 2 'symbols' must have 'checked reasonableness of questions' then 'decided question was worthwhile', in that specific order

2.2.1.9. Such a description could both specify some pattern for behavior, e.g. for composing some plan for behavior, another e.g. for a designer to 'program' some specific behavioral sequence in advance

2.2.1.10. Such a description could also describe some observed behavior, e.g. for circumstances where complex set of influences drove some outcome behavior over some time period, the occupancy of particular states within that period is a specific and particular trace of which behaviors were executed

2.3. Narrative Programs

2.3.1.1. More details and examples of what narrative programs may look like

2.3.1.2. Narrative programming is the process of writing 'programs' using some defined set of narrative objects / 'symbols':

2.3.1.3. A well-designed set of 'symbols' enables us to 'program' systems that can exhibit useful real world behaviors, such as systems that robustly evaluate themselves inclusive of their observations, hypotheses, and prior goals

2.3.1.4. But importantly, components of the 'programs' are actually implemented within particular 'symbols' and are just conditional routes to other symbols

2.3.1.5. Thus, 'programming' within this system consists of describing conditional branches to other symbols and organizing sets of conditional branches that describe some particular pattern of behavior which is meaningful outside of the system and ideally useful within the system

2.3.1.6. An example of a 'program' might be (pseudocode) if ('SEMANTIC_UNCLEAR') trigger ('_QUERY_TRUSTED_SOURCE') <then> trigger ('CHECK_SEMANTIC_CLARITY')

2.3.1.7. Another example might be (pseudocode) if ('ENERGY_ACCOUNT_LOW') trigger ('REDUCE_EXPLORATION') <and> ('INCREASE_EXPLOITATION')

3. Principles of Operation

3.1. Self-Evaluation

3.1.1. Self-evaluation of objectives, planning processes, actions, and observations

3.1.2. "Learning to Predict by the Methods of Temporal Differences" by Richard S. Sutton (1988): This paper introduces the concept of temporal difference learning, a key component of reinforcement learning algorithms used for self-evaluation and planning in AI systems.

3.1.3.

3.1.4. "Planning by Inference: A Study in Graphical and Relational Probabilistic Models" by Stuart Russell and Brian Milch (2008): This paper discusses the use of graphical and relational probabilistic models for planning and inference, which enables AI systems to evaluate their own actions and observations to improve decision-making.

3.1.5.

3.1.6. "Unifying Logical and Statistical AI" by Pedro Domingos (2009): This paper explores the integration of logical reasoning and statistical learning for self-evaluation and planning in AI systems, bridging the gap between symbolic and statistical approaches to AI.

3.1.7.

3.1.8.

3.1.9.

3.1.10. "Mastering the Game of Go with Deep Neural Networks and Tree Search" by David Silver et al. (2016): This paper presents AlphaGo, a groundbreaking AI system that combines deep neural networks with Monte Carlo tree search for evaluating and planning moves in the game of Go.

3.1.11.

3.1.12.

3.1.13.

3.1.14. "Planning and Learning with Tabular Methods in Markov Decision Processes" by Leslie Pack Kaelbling et al. (1996): This paper discusses the use of tabular methods, such as dynamic programming and Q-learning, for self-evaluation and planning in Markov Decision Processes, a popular framework for modeling decision-making problems in AI.

3.1.15.

3.1.16. "Planning with Continuous Effect Models" by Brian C. Williams and Robert S. Jarvis (1996): This paper explores the use of continuous effect models for planning and self-evaluation in AI systems, allowing for more realistic and flexible representations of actions and their consequences.

3.1.17. Discussion of how

3.2. Planning and Autonomy

3.2.1. Planning, prioritization and decision-making

3.2.2. Simulation

3.2.3. "A Fast Learning Algorithm for Deep Belief Nets" by Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh (2006): This paper introduced the concept of Deep Belief Networks (DBNs) which are capable of learning and representing complex patterns. DBNs have been influential in various areas of AI research, including planning and decision-making.

3.2.4. "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference" by Judea Pearl (1988): This book introduced the theory of Bayesian networks and probabilistic reasoning. It has been foundational in planning and decision-making systems that involve uncertainty and probabilistic models.

3.2.5. "Fast Planning Through Planning Graph Analysis" by Avrim L. Blum and Merrick L. Furst (1997): This paper proposed the Planning Graph heuristic, which is an efficient method for generating and evaluating plans. It has been widely influential in the field of automated planning and has paved the way for subsequent research in plan generation and evaluation.

3.2.6.

3.2.7. "Decision-Theoretic Planning: Structural Assumptions and Computational Leverage" by Stuart Russell and Daniel S. Weld (1995): This paper introduced the decision-theoretic framework for planning, where plans are generated by optimizing a utility function over possible actions. It provides a framework for incorporating decision-making and prioritization into planning systems.

3.2.8.

3.2.9. "Markov Decision Processes: Discrete Stochastic Dynamic Programming" by Richard Bellman (1957): Although not specific to AI, this seminal work introduced Markov Decision Processes (MDPs) as a mathematical framework for sequential decision-making under uncertainty. MDPs have been widely used in AI research for planning and decision-making problems.

3.2.10.

3.2.11. "Planning as Satisfiability" by Henry Kautz and Bart Selman (1992): This paper introduced the concept of reducing planning problems to Boolean satisfiability problems (SAT). The reduction allows planners to leverage existing efficient SAT solvers to solve planning problems and has been influential in the development of planning algorithms.

3.2.12.

3.2.13. "Anytime Dynamic A* for Stochastic Shortest Path Problems" by Maxim Likhachev, Dave Ferguson, Geoffrey Gordon, Anthony Stentz, and Sebastian Thrun (2005): This paper introduced the Anytime Dynamic A* algorithm, which is a planning algorithm that generates approximate solutions incrementally and can be interrupted at any time to return the best solution found so far. It has been influential in planning systems that require real-time decision-making.

3.3. Performance Assessment

3.3.1. resource allocation

4. Features and Benefits

4.1. Accuracy

4.2. Robustness

4.2.1. Learning
4.2.2. Autonomy Open-ended search and exploration
4.2.3. Economy
4.2.4. Useful ideation
4.2.5. Common sense
4.2.6. Decision making
4.2.7. Reasoning
4.2.8. Applications

4.2.8.1. Driving
4.2.8.2. Math
4.2.8.3. Engineering
4.2.8.4. Physics
4.2.8.5. Science

4.3. Explainability

4.3.1.1. Need to fix formatting

4.3.1.1.1. Another paper that might be relevant is “Explainable Artificial Intelligence: Objectives, Stakeholders, and Future Research Opportunities” by Christian Meske et al. This paper describes exemplary risks of black-box AI and the consequent need for explainability[2].

4.3.1.2. Autospection and explanation
4.3.1.3. Self description
4.3.1.4. Narration
4.3.1.5. Moreover, the potential applications of narrative computation are vast and diverse. From virtual assistants and chatbots to complex decision-making systems, narrative computation offers a new framework for tackling the challenges posed by real-world scenarios. By leveraging accurate behavioral descriptions at multiple levels of abstraction, over multiple epochs, and for multiple scopes of concern, narrative computation enables the creation of AI systems that can adapt and respond to dynamic and evolving situations.

4.4. Multimodality

5. Contrasting Narrative Computation with Other Approaches

5.1. Historical Approaches

5.1.1. Expert Systems
5.1.1.1. Expert Systems:
5.1.1.2. Research Paper: "Expert Systems: Principles and Programming" by Joseph C. Giarratano and Gary D. Riley (2004)
5.1.1.3. Paradigm: Knowledge-based Systems
5.1.1.4. System: Dendral (early expert system for chemistry)
5.1.2. Rule-based Systems:
5.1.2.1. Research Paper: "Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project" by Edward H. Shortliffe and Bruce G. Buchanan (1975)

5.1.2.2. Paradigm: Production Systems
5.1.2.3. System: CLIPS (C Language Integrated Production System)

5.1.3. Decision-tree learning

5.1.3.1. decision tree learning, Mitchell's version space learning, Valiant's contributions to PAC learning, statistical relational learning, inductive logic programming, and

5.1.3.2. In contrast to the knowledge-intensive approach of Meta-DENDRAL, Ross Quinlan invented a domain-independent framework for statistical classification, decision tree learning, starting first with ID3[63] and then later extending its capabilities to C4.5.[64] The decision trees created are glass box, interpretable classifiers, with human-interpretable classification rules.

5.1.4. Version space learning

5.1.4.1. Advances were made in understanding machine learning theory, too. Tom Mitchell introduced version space learning which describes learning as search through a space of hypotheses, with upper, more general, and lower, more specific, boundaries encompassing all viable hypotheses consistent with the examples seen so far.[65]

5.1.5. Probably Approximately Correct learning

5.1.5.1. More formally, Valiant introduced Probably Approximately Correct Learning (PAC Learning), a framework for the mathematical analysis of machine learning.[66]

5.1.6. Inductive Logic Programming

5.1.6.1. Inductive logic programming was another framework for learning that allowed logic programs to be synthesized from input-output examples. E.g., Ehud Shapiro's MIS (Model Inference System) could synthesize Prolog programs from examples.[68] John R. Koza applied genetic algorithms to program synthesis to create genetic programming, which he used to synthesize LISP programs. Finally, Manna and Waldinger provided a more general framework for program synthesis that synthesizes a functional program in the course of proving its specifications to be correct.[69]

5.1.7. Cognitive Architectures:

5.1.7.1. Research Paper: "ACT-R: A Theory of Higher-Level Cognition and Its Relation to Visual Attention" by John R. Anderson et al. (1997)
5.1.7.2. Paradigm: Cognitive Architectures
5.1.7.3. Architecture: ACT-R (Adaptive Control of Thought-Rational)

5.1.7.4. Symbolic machine learning encompassed more than learning by example. E.g., John Anderson provided a cognitive model of human learning where skill practice results in a compilation of rules from a declarative format to a procedural format with his ACT-R cognitive architecture. For example, a student might learn to apply "Supplementary angles are two angles whose measures sum 180 degrees" as several different procedural rules. E.g., one rule might say that if X and Y are supplementary and you know X, then Y will be 180 - X. He called his approach "knowledge compilation". ACT-R has been used successfully to model aspects of human cognition, such as learning and retention. ACT-R is also used in intelligent tutoring systems, called cognitive tutors, to successfully teach geometry, computer programming, and algebra to school children.[67]

5.1.8. Case-Based Reasoning

5.1.8.1. Case-Based Reasoning:
5.1.8.2. Research Paper: "Case-Based Reasoning" by Janet Kolodner (1993)
5.1.8.3. Paradigm: Case-Based Reasoning
5.1.8.4. System: CBR-Retrieval (Case-Based Reasoning Retrieval System)
5.1.8.5. As an alternative to logic, Roger Schank introduced case-based reasoning (CBR). The CBR approach outlined in his book, Dynamic Memory,[70] focuses first on remembering key problem-solving cases for future use and generalizing them where appropriate. When faced with a new problem, CBR retrieves the most similar previous case and adapts it to the specifics of the current problem.[71] Another alternative to logic, genetic algorithms and genetic programming are based on an evolutionary model of learning, where sets of rules are encoded into populations, the rules govern the behavior of individuals, and selection of the fittest prunes out sets of unsuitable rules over many generations.[72]

5.1.9. Planning and Plan Recognition:
5.1.9.1. Research Paper: "Fast Planning Through Planning Graph Analysis" by Avrim L. Blum and Merrick L. Furst (1997)
5.1.9.2. Paradigm: Automated Planning
5.1.9.3. System: STRIPS (Stanford Research Institute Problem Solver)
5.1.10. Explainable AI (XAI):
5.1.10.1. Research Paper: "Explainable Artificial Intelligence (XAI)" by Mark Stefik et al. (1986)
5.1.10.2. Paradigm: Explainable AI
5.1.10.3. System: LIME (Local Interpretable Model-Agnostic Explanations)
5.1.11. Natural Language Processing (NLP):
5.1.11.1. Research Paper: "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition" by Daniel Jurafsky and James H. Martin (2020)
5.1.11.2. Paradigm: Natural Language Processing
5.1.11.3. System: Google BERT (Bidirectional Encoder Representations from Transformers)
5.1.12. Knowledge Representation and Reasoning:
5.1.12.1. Research Paper: "A Logic of Knowledge and Belief" by Robert Fagin et al. (1995)
5.1.12.2. Paradigm: Knowledge Representation and Reasoning
5.1.12.3. Architecture: Description Logics (DLs) and Semantic Web Rule Language (SWRL) for knowledge representation

5.2. Contemporary Approaches

5.2.1. Generative AI
5.2.2. GAN
5.2.3. Attention
5.2.4. Transformer
5.2.5. Transformers - Transformers, such as BERT and GPT, are powerful models that leverage attention mechanisms to capture contextual dependencies in sequential data. While narrative computation focuses on describing computational states and processes using a vocabulary of causal-model objects,

transformers excel at capturing relationships within large-scale textual data, but they do not inherently provide a framework for robust behavioral control and evaluations like narrative computation.

5.2.6. Generative AI - Generative AI, exemplified by Generative Adversarial Networks (GANs), focuses on learning to generate new data samples that resemble a given training dataset. While narrative computation involves using behavioral descriptions and state-based reasoning to control AI systems, generative AI is primarily concerned with generating data, such as images, text, or audio, rather than explicitly modeling computational states and processes.

5.2.7. Variational Auto-encoders (VAEs) - Variational Auto-encoders (VAEs) are generative models that aim to learn latent representations of input data, enabling the generation of new samples. While VAEs capture the underlying statistical structure of data, narrative computation focuses more on the precise descriptions of computational states and processes within an AI system rather than explicitly modeling the probabilistic generation of data.

5.2.8. Reinforcement Learning (RL) - Reinforcement Learning (RL) involves an agent learning to interact with an environment to maximize a reward signal. While RL systems can exhibit complex behaviors, narrative computation focuses on the precise descriptions of computational states and processes, enabling robust behavioral control and self-evaluation. RL can benefit from narrative descriptions for specifying goals and context but typically lacks the explicit representation of narratives as described in narrative computation.

5.2.9. Graph Neural Networks (GNNs) - Graph Neural Networks (GNNs) are designed to process and reason about graph-structured data, capturing relationships between nodes and their features. While GNNs are well-suited for graph-based computations, narrative computation focuses on the precise descriptions of computational states and processes within a system, which may or may not involve graph-like structures. Narrative computation emphasizes behavioral control and evaluations beyond the graph-based reasoning capabilities of GNNs.

5.2.10. Deep Reinforcement Learning (DRL) - Deep Reinforcement Learning (DRL) combines deep neural networks with reinforcement learning to learn policies for sequential decision-making tasks. While DRL systems can achieve impressive results in complex environments, narrative computation is more concerned with precise behavioral descriptions and evaluations of computational states and processes, which goes beyond the learning of policies through trial and error.

5.2.10.1. "Deep Learning" by Yann LeCun, Yoshua Bengio, and Geoffrey Hinton (2015): This paper provides an overview of deep learning techniques, including deep neural networks, which have revolutionized many areas of AI research, including self-evaluation and planning.

5.2.10.2. "Human-Level Control through Deep Reinforcement Learning" by Volodymyr Mnih et al. (2015): This paper introduces the concept of deep reinforcement learning, where AI agents learn to evaluate their own actions and observations through trial and error, achieving human-level performance in various Atari 2600 games.

5.2.11. Meta-Learning - Meta-learning focuses on learning how to learn or adapt quickly to new tasks or environments. While meta-learning aims to acquire general learning abilities, narrative computation is centered around the precise descriptions of computational states, processes, and behavioral control within an AI system. Meta-learning can potentially benefit from incorporating narrative descriptions to guide the learning and adaptation process.

5.2.12. Deep Generative Models - Deep generative models, such as Normalizing Flows, aim to learn complex probability distributions and generate new samples from them. While these models provide powerful generative capabilities, narrative computation focuses on precise behavioral descriptions, evaluations, and behavioral control, which go beyond the probabilistic modeling and sample generation objectives of deep generative models.

5.2.13. Papers…

5.2.13.1. Transformers:

5.2.13.2.

5.2.13.3. Research Paper: "Attention Is All You Need" by Vaswani et al. (2017)

5.2.13.4. Model: Transformer (including variants like BERT, GPT, T5, etc.)

5.2.13.5. System: OpenAI GPT-3 (Generative Pre-trained Transformer 3)

5.2.13.6. Generative AI:

5.2.13.7.

5.2.13.8. Research Paper: "Generative Adversarial Networks" by Goodfellow et al. (2014)

5.2.13.9. Model: GAN (Generative Adversarial Network)

5.2.13.10. System: NVIDIA StyleGAN (Style-based Generative Adversarial Network)

5.2.13.11. Variational Auto-encoders (VAEs):

5.2.13.12.

5.2.13.13. Research Paper: "Auto-Encoding Variational Bayes" by Kingma and Welling (2014)

5.2.13.14. Model: Variational Auto-encoder (VAE)

5.2.13.15. System: VQ-VAE (Vector Quantized Variational Auto-encoder)

5.2.13.16. Reinforcement Learning (RL):

5.2.13.17.

5.2.13.18. Research Paper: "Playing Atari with Deep Reinforcement Learning" by Mnih et al. (2013)

5.2.13.19. Paradigm: Reinforcement Learning

5.2.13.20. System: OpenAI Five (deep reinforcement learning system for Dota 2)

5.2.13.21. Graph Neural Networks (GNNs):

5.2.13.22.

5.2.13.23. Research Paper: "Semi-Supervised Classification with Graph Convolutional Networks" by Kipf and Welling (2017)

5.2.13.24. Model: Graph Convolutional Network (GCN)

5.2.13.25. System: DeepChem (deep learning library for drug discovery with GNNs)

5.2.13.26. Deep Reinforcement Learning (DRL):

5.2.13.27.

5.2.13.28. Research Paper: "Human-Level Control Through Deep Reinforcement Learning" by Mnih et al. (2015)

5.2.13.29. Paradigm: Deep Reinforcement Learning

5.2.13.30. System: AlphaGo Zero (deep RL system that achieved superhuman performance in Go)

5.2.13.31. Meta-Learning:

5.2.13.32.

5.2.13.33. Research Paper: "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks" by Finn et al. (2017)

5.2.13.34. Paradigm: Meta-Learning

5.2.13.35. System: MAML (Model-Agnostic Meta-Learning)

5.2.13.36. Deep Generative Models:

5.2.13.37.

5.2.13.38. Research Paper: "Variational Inference with Normalizing Flows" by Rezende and Mohamed (2015)

5.2.13.39. Model: Normalizing Flows
5.2.13.40. System: Glow (Generative Flow with Invertible 1x1 Convolutions)

5.2.14. Multimodality, specifically images and text

5.2.14.1. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention" by Kelvin Xu et al. (2015): This paper introduced an attention mechanism for image captioning, where a deep neural network learns to attend to different parts of an image while generating a textual description.

5.2.14.2.

5.2.14.3. "Visualizing and Understanding Convolutional Networks" by Matthew D. Zeiler and Rob Fergus (2014): The authors proposed a technique called "deconvolutional networks" to visualize and understand the representations learned by convolutional neural networks (CNNs) for image classification tasks.

5.2.14.4.

5.2.14.5. "Deep Visual-Semantic Alignments for Generating Image Descriptions" by Andrej Karpathy and Li Fei-Fei (2015): This paper presented a deep neural network model that generates textual descriptions for images by aligning visual and semantic information using a CNN and a Long Short-Term Memory (LSTM) network.

5.2.14.6.

5.2.14.7. "From ImageNet to Places: A Large-Scale Visual Recognition Challenge" by Bolei Zhou et al. (2014): This paper introduced the Places dataset, which focuses on scene recognition, and proposed the use of deep CNNs for multi-modal tasks involving images and text.

5.2.14.8.

5.2.14.9. "Visual Question Answering" by Stanislaw Antol et al. (2015): The authors introduced the Visual Question Answering (VQA) task, where a model is trained to answer questions about images. The paper describes the VQA dataset and presents a baseline approach that combines CNNs and LSTMs to tackle this task.

5.2.14.10.

5.2.14.11. "Stacked Attention Networks for Image Question Answering" by Zichao Yang et al. (2016): This paper proposed a stacked attention network that utilizes attention mechanisms to answer questions about images. The model dynamically focuses on different regions of an image and combines information from multiple levels of attention.

5.2.14.12.

5.2.14.13. "Learning Cross-Modal Embeddings for Cooking Recipes and Food Images" by Chih-Yao Ma et al. (2015): The authors presented a method to learn joint embeddings of cooking recipes and food images, allowing cross-modal retrieval between the two domains. They used deep neural networks to represent the text and image modalities.

5.2.14.14.

5.2.14.15. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks" by Shaoqing Ren et al. (2016): This paper introduced the Faster R-CNN model, which integrates region proposal networks (RPNs) with CNNs for efficient and accurate object detection in images. The model has been widely used as a backbone for various multi-modal tasks.

6. NX-RESEARCH Systems and Architectures

6.1. NX-Utility™ Modules

6.2. Curious-NX™ Agent Platform

6.3. Research roadmap

6.3.1. Experimentation Setup and Methodology
- 6.3.1.1. Evaluation of Architecture and Programs
- 6.3.1.2. Handling Larger Datasets and Multi-Modal Conditions

6.3.2. Scalable Efficiency and Practical Performance
- 6.3.2.1. Definition of Scalable Efficiency in Narrative Computation
- 6.3.2.2. Measuring Practical Performance (e.g., AGI Eval)
- 6.3.2.3. Maintaining NX-Utility™ Measures in Real-World Scenarios
- 6.3.2.4. Expected Value for Mission-Critical AI Applications

7. Discussion

7.1. Further Research Opportunities

7.1.1.

7.2. Potential Limitations of this Approach

7.2.1.

7.3. Final Remarks and Closing Thoughts

7.3.1. In the preceding sections, we delved…

7.3.2. delve deeper into the conceptual underpinnings of narrative computation, explore its key components and terminology, and examine the benefits it offers over historical and contemporary approaches. We will also present specific examples of narrative programs and discuss the semantics of narrative computation. Additionally, we will introduce the NX™ narrative computation systems and the innovative design principles that ensure accuracy and usefulness of computational measures and evaluations. Finally, we will discuss our ongoing research objectives, including testing complex programs using the Curious-NX™ agent platform, and our expectations for scalable efficiency and practical performance.